طراحی گام به گام پایگاه داده

راه حل پایگاه داده

رهیافت گام به گام برای ساخت پایگاه داده

Thomas M. Connolly
Carolyn E. Begg

مترجم: حامد نجارپور جباری

عنوان	:	راه حل پایگاه : رهیافت گام به گام برای ساخت پایگاه داده
نویسنده	:	Thomas Connolly & Carolyn Begg
مترجم	:	نجارپور جباری، حامد، ۱۳۶۳
شابک	:	978-964-8488-29-6
نسخه	:	الکترونیکی
یادداشت	:	ترجمه فارسی کتاب طراحی گام به گام پایگاه داده
یادداشت	:	عنوان اصلی: Database Solutions: A step-by-step guide to building databases
قیمت	:	رایگان
یادداشت	:	انتشار الکترونیکی این کتاب هیچگونه سود مادی برای اینجانب ندارد . ترجمه فارسی این کتاب و کلیه فعالیتهای مرتبط اعم از آماده سازی و برگرداندن عکسها و نمودارها به فارسی و ترجمه متون و تایپ و ویرایش حاصل یک فعالیت شخصی بوده و بعلت هزینه زیاد نشر کتاب، چاپ کتاب مقدور نشد. و بنابراین تصمیم گرفتم این کتاب را بصورت **رایگان** در اختیار علاقمندان قرار دهم. امیدوارم مفید باشد. همچنین میتوانید برای ارایه پیشنهادات خود با ایمیل من تماس بگیرید. با تشکر
Donate	:	شماره کارت سامان جهت واریز هدایای نقدی در صورت تمایل: 6219 8610 1141 8711 بنام حامد نجارپور جباری jabbary.hamed@gmail.com

خلاصه مطالب

فهرست مطالب

بخش ۱ مقدمه

بخش ۳ **طراحی منطقی پایگاه داده**

بخش ۵ دومین مثال عملی

ضمیمه ها

پیشگفتار

امروزه پایگاه داده چهارچوب زیرین سیستم اطلاعاتی را تشکیل می دهد و راه انجام کارهای بسیاری از شرکتها و اشخاص را بطور اساسی تغییر داده است. توسعه این تکنولوژی طی چندین سال گذشته سیستم های پایگاه داده ای را ایجاد کرده است . اگر به کتابهای مرتبط با پایگاه داده نگاهی بیاندازید کتابهای عالی زیادی را می یابید که بخشی از چرخه حیات توسعه پایگاه داده را مورد بررسی قرار داده اند. با این وجود، کتابهای بسیار کمی نوشته شده است که در آن مراحل طراحی، تجزیه و تحلیل ، پیاده سازی و توسعه پایگاه داده را بطور قابل فهم توضیح داده باشد بطوریکه خوانندگان فنی و هم غیر فنی بتوانند از آن استفاده کنند.

بنابراین هدف اصلی ما ارائه کتابی است که ابتدا برای جامعه تجاری مورد استفاده قرار گیرد که در آن به طور واضح تجزیه، طراحی و پیاده سازی پایگاه داده را توضیح دهد. که این، هم پایگاه داده های ساده با جداول اندک و هم پایگاه داده های بزرگی را از ده تا چندین صد جدول در بر می گیرد. طی بررسی های اولیه که انجام شد چنین بنظر رسید که این کتاب برای مجامع دانشگاهی نیز می تواند مفید باشد و مطالب خیلی ساده و واضحی را از اصول طراحی پایگاه داده ارایه کند که مکمل کتابهای درسی باشد.

متدلوژی که ما در این کتاب برای سیستمهای مدیریتی پایگاه داده ارتباطی (DBMS) ارائه می کنیم طی سالیان سال در محیطهای آکادمیک و صنعتی بارها و بارها تست شده اند. این متدولوژی به دو مرحله تقسیم می شود:

مرحله طراحی منطقی پایگاه داده که در این مرحله ما سعی می کنیم مدلی را از آنچه می خواهیم نشان دهیم طراحی کنیم که این مرحله با نادیده گرفتن جزئیات پیاده سازی انجام می شود.

مرحله طراحی فیزیکی پایگاه داده که در این مرحله سعی می کنیم پیاده سازی در DBMS هدف مانند اکسس، پارادوکس، اوراکل و Informix و ... تحقق بخشیم.

ما هر مرحله را گام به گام ارائه می کنیم. برای طراح بی تجربه انتظار آن را داریم که مراحل به ترتیب گفته شده و راهنمایی های داده شده در سراسر مرحله انجام گیرد. و برای طراح مجرب مراحل ذکر شده می تواند بصورت انتخابی مطالعه شود.

کمک به فهم طراحی پایگاه داده

برای اینکه در فهم موضوعات مهم در استفاده از متد پایگاه داده به شما کمک کرده باشیم مثال عملی جامعی را ارائه می کنیم که بر مبنای یک شرکت اجاره فیلم با نام StayHome ساخته شده است. که همراه با مطالب کتاب ارائه می شود. برای اینکه به شما بیشتر کمک کنیم ما همچنین راه حل هایی اضافی پایگاه داده را در ضمیمه د کتاب فراهم ساخته ایم. هر راه حلی مقدمه کوچکی دارد که شما می توانید آنرا خوانده و پایگاه داده آنرا طراحی کنید قبل از آن به راه حل های ما مراجعه کنید.

مدلهای داده ای متداول

علاوه بر اینکه شما را با تجربیات اضافی طراحی پایگاه داده آشنا کنیم ضمیمه د همچنین شما را با مدلهای داده ای متداول آشنا می کند که می تواند برای شما مفید باشد. در حقیقت چنین تخمین زده شده است که یک سوم از مدلهای داده ای شامل ساختار مشترکی هستند که در بسیاری از شرکتها از آن استفاده می کنند و بقیه دو سوم آنها دارای خصوصیات شرکتی یا صنعتی هستند. بنابراین بسیاری از طراحی های پایگاه داده شامل باز آفرینی پایگاه داده هایی هستند که قبلا در شرکتهای دیگر مورد استفاده قرار داده شده اند. این مدلهای طراحی شده ممکن است در شرکت شما دقیقا بکار نرود اما می تواند نقطه شروعی باشد که با خصوصیات شرکت شما مطابقت کند بعضی از مدلهایی که ما ارائه می کنیم حاوی حیطه های تجاری زیر خواهد بود .

- **ثبت سفارش مشتری**
- **کنترل موجودی**
- **مدیریت پروژه**
- **مدیریت آموزش**
- **مدیریت دارایی**
- **مدیرت منابع انسانی**
- **و مدیریت صورت پرداخت**

چگونگی ایجاد طراحی

بنظر ما مهم است که به شما نشان دهیم چگونه یک طرح پایگاه داده را به یک عمل فیزیکی تبدیل کنید. در این کتاب ما به شما نشان می دهیم که چگونه اولین بررسی موردی را در مایکروسافت اکسس ایجاد کنیم. همچنین نشان می دهیم که چطور طرح پایگاه داده را برای دومین بررسی موردی (کلینیک دامپزشکی بنام PERFECT PETS) در DBMS اوراکل ۸ ایجاد کنیم.

چه کسانی باید این کتاب را بخوانند؟

ما سعی کرده ایم این کتاب را بصورت خود آموز بنویسیم. استثنایی که وجود دارد بخش طراحی پایگاه داده فیزیکی می باشد که لازم است شما از درک خوبی از عملکرد DBMS هدف بایستی داشته باشید. مخاطبین ما همه کسانی هستند که می خواهند پایگاه داده خود را توسعه دهند که شامل افراد زیر بوده و بسیار فراتر از آنها نیز می باشند:

- مدلسازان اطلاعات و طراحان پایگاه داده
- طراحان عملی پایگاه داده و پیاده سازان.
- کارورزان پایگاه داده
- مدیران پایگاه داده و داده.
- استادان علوم کامپیوتر ، آی تی، سیستمهای اطلاعاتی که متخصص طراحی پایگاه داده می باشند.
- دانشجویان پایگاه داده اعم از دانشجویان لیسانس، و فوق لیسانس.
- هر کسی که می خواهد پایگاه داده ای را طراحی کرده و توسعه دهد.

ساختار کتاب

ما ساختار این کتاب را به پنج بخش و چهار ضمیمه تقسیم کرده ایم.

بخش ۱- پس زمینه . ما مقدمه ای به DBMS و مدل ارتباطی در بخش ۱ و ۲ ارائه کرده ایم همچنین چرخه حیات کاربرد پایگاه داده را در بخش ۳ ارائه کرده ایم.

بخش۲- تکنیکهای طراحی و تجزیه و تحلیل پایگاه داده. ما در مورد تکنیکها برای آنالیز پایگاه داده در بخش ۴ بحث می کنیم. همچنین نشان می دهیم که چگونه از بعضی از این تکنیکها جهت تجزیه نیازمندیها برای شرکت اجاره فیلم StayHome استفاده کنیم. ما نشان می دهیم که چگونه دیاگرامهای ارتباط موجودیتی (ER) را جهت استفاده در UML در فصل ۵ رسم کرده و چگونه قواعد نرمال سازی را در بخش ۶ بکار ببریم. مدلهای ER و نرمال سازی تکنیکهای مهمی هستند که در متدولوژی که در بخش ۳ توصیف می کنیم استفاده می شوند.

بخش ۳– متدلوژی طراحی منطقی پایگاه داده. ما رهیافت گام به گام برای طراحی منطقی پایگاه داده را توضیح داده ایم. در مرحله ۱، مدل داده ای منطقی محلی را برای هر دید موجود در شرکت ایجاد کرده ایم. در مرحله ۲ ، هر مدل را به مجموعه ای از جدولهای پایگاه داده نگاشت کرده، و در مرحله ۳ مدلهای داده ای محلی را با هم دیگر جهت تهیه مدل سراسری شرکت ادغام کرده ایم.

بخش٤– متدلوژی طراحی فیزیکی پایگاه داده. ما در این بخش رهیافت گام به گامی را برای طراحی فیزیکی پایگاه داده توصیف کرده ایم. در بخش ۴ ما مجموعه ای از جدول های پایه را برای DBMS هدف طراحی می کنیم. در مرحله ۵، ما ساختمان فایلها و شاخص ها را انتخاب می کنیم. در مرحله ۶، مقدمه ای را جهت کنترل افزونگی برای رسیدن به کارایی مورد قبول در نظر می گیریم. در مرحله ۷، مقادیر امنیت را طراحی می کنیم که داده هایمان را از دستیابی غیر مجاز حفاظت می کنند. بالاخره در مرحله ۸، سیستم عملکری مان را بهبود و باز بینی خواهیم کرد. همانگونه که متذکر شدیم، نشان می دهیم که شما چگونه طراحی را برای کاربرد پایگاه داده StayHome در اکسس مایکروسافت انجام دهید.

بخش ۵– دومین مثال عملی . در فصلهای ۱۸ و ۱۹، ما از طریق دومین بررسی موردی برای کلینیک دامپزشکی Perfect Pets کار خواهیم کرد. ما نشان می دهیم که شما چگونه طراحی را برای کاربرد پایگاه داده Perfect Pets در اوراکل ۸ انجام دهید.

ضمایم. ضمیمه الف دو نمادسازی متداول اصلی ER را بررسی می کند: نمادسازی Chen و نماد سازی Crow's feet. ضمیمه ب خلاصه ای از متدلوژی را جهت راهنمای سریع فراهم کرده است. ضمیمه د مجموعه ای از ۱۵ مدل داده ای متداول را فراهم آورده است

در پایان از همه کسانی که در تالیف این کتاب کمک حال ما شدند و ما را در نشر این کتاب یاری کردند سپاسگذاری می کنیم.

Thomas M. Connolly
Carolyn E. Begg
Glasgow, September 1999

فصل ۱

مقدمه

آنچه در این فصل خواهید آموخت :

- برخی از کاربردهای متداول سیستمهای پایگاه داده.
- معنی اصطلاح پایگاه داده.
- معنی اصطلاح سیستم مدیریت پایگاه داده DBMS.
- مولفه های اصلی محیط DBMS.
- سرویس هایی که یک DBMS بایستی فراهم کند.
- مزیت و اشکالات DBMS.

پایگاه داده امروزه به عنوان بخش جدا ناپذیری از زندگی روزمره ما شده است بطوریکه اغلب متوجه نیستیم که در حال استفاده از آن هستیم. برای شروع بحث مان درباره پایگاه داده، ابتدا به طور مختصر بعضی از کاربردهای آن را مورد بررسی قرار میدهیم. در این بحث پایگاه داده را به عنوان مجموعه ای از داده های به هم مرتبط و سیستم مدیریت پایگاه داده (DBMS) را نرم افزاری در نظر میگیریم که وظیفه کنترل و مدیریت دسترسی بر داده ها را بر عهده دارد. همچنین به صورت کلی از اصطلاح سیستم پایگاه داده به عنوان مجموعه ای از برنامه های کاربردی مرتبط با پایگاه داده استفاده خواهیم کرد. در بخش بعدی فصل به کارکردهای نمونه یک DBMS مدرن نگاهی خواهیم انداخت و اجمالا امتیازات و اشکالات DBMS را مورد بررسی قرار میدهیم.

۱- ۱ مثالهای استفاده از سیستم های پایگاه داده

خرید از فروشگاه

وقتی از فروشگاه محله تان کالایی را میخرید فروشنده فروشگاه احتمالابه یک پایگاه داده دسترسی دارد. مثلا شخصی که کالا را بررسی می کند ممکن است از یک وسیله بارکدخوان برای خواندن مشخصات اجناس خریداری شده استفاده کند. که خود این وسیله نیز به یک برنامه کاربردی متصل است که از بارکد برای پیداکردن قیمت کالا از پایگاه داده محصولات استفاده میکند.

خرید توسط کارت اعتباری

زمانی را بیاد آورید که کالایی را با استفاده از کارت اعتباری میخرید، فروشنده معمولا اعتبار کارتتان را بررسی میکند. این بررسی ممکن است با تلفن یا به طور اتوماتیک توسط کارت خوان متصل به سیستم کامپیوتری صورت گیرد. در هر دو حالت پایگاه داده ای وجود دارد که اطلاعات خریدهای انجام شده توسط کارت اعتباری در آن ذخیره می شود. همچنین جهت

بررسی اعتبار کارت، برنامه کاربردی وجود دارد که تمام خریدهای انجام شده را با استفاده از شماره کارت اعتباری بررسی کند. هنگام تایید خرید جزئیات آن به پایگاه داده افزوده می شود.

استفاده از کتابخانه عمومی

شما هنگامیکه برای مطالعه به کتابخانه می روید، احتمالا پایگاه داده ای وجود دارد که در آن همه جزئیات مربوط به کتابهای موجود درکتابخانه، و همچنین اطلاعات مربوط به خوانندگان کتابها، رزرو کتابها و غیره موجود میباشد. همچنین ایندکس کامپیوتری وجود دارد که به خوانندگان اجازه میدهد کتاب خود را از روی مشخصات مولف، عنوان، و یا موضوع کتاب پیدا کنند. سیستم پایگاه داده رزرو کتابها را مدیریت میکند و به خوانندگان اجازه میدهد که کتابی را رزرو و یا اگر کتابی در دسترس بود با ایمیل آنها را با خبر کنند. کتابها همچنین دارای بارکدی همانند آنچه در مثال فروشگاه گفته شد که اطلاعات ورود و خروج کتابها را بررسی میکند.

رزرو بلیط مسافرت در آژانس مسافرتی

شما هنگامیکه از آژانس مسافرتی درباره تعطیلات پرس و جو میکنید، آژانس احتمالا به چندین پایگاه داده که شامل جزئیات تعطیلات و پرواز است دسترسی دارد. وقتی مقصد سفر را رزرو میکنید، سیستم پایگاه داده همه ترتیبات رزرو لازم را انجام میدهد. در این مورد، سیستم باید مطمئن شود که دو آژانس مختلف تعطیلات مشابه را رزرو یا صندلیهای موجود در پرواز را دوباره رزرو نکنند. برای مثال، اگر تنها یک صندلی در پرواز از تبریز به تهران باقی باشد و دو آژانس سعی کنند درآخرین لحظه به طور همزمان صندلی را رزرو کنند، در این صورت سیستم این موقعیت را تشخیص داده، و تنها اجازه میدهد که یکی از آنها صندلی را رزرو کند، و به طور خودکار آژانس دیگری را باخبر می کند که صندلی خالی دیگر وجود ندارد. همچنین آژانس مسافرتی ممکن است، پایگاه داده مجزایی نیز برای صورتحساب داشته باشد.

در مثالهای بالا چند کاربرد نمونه استفاده از پایگاه داده را دیدیم و همانگونه که مشاهده میشود در تمامی اینها پایگاه داده نقش کلیدی را بازی میکند. برای سیستمی که خواسته های کاربران نهایی[۱] را برآورده سازد، به یک پایگاه داده ساختیافته نیاز داریم. ایجاد چنین ساختاری *طراحی پایگاه داده*[۲] نام دارد. هدف ما در این کتاب تمرکز بر این فعالیت عمده میباشد که میخواهیم به آن بپردازیم. پایگاه داده چه کوچک باشد یا بزرگ به هر حال، طراحی آن یک موضوع مفهومی است، و متدلوژی ارائه شده در این کتاب به شما کمک میکند که پایگاه داده خود را به طور صحیح و به آسانی با استفاده از رابطه طراحی کنید. داشتن پایگاه داده ای که به خوبی طراحی شده به شما اجازه میدهد تا سیستمی ایجاد کنید که نیازهای کاربران را برآورده کرده و درعین حال، کارائی قابل قبولی را نیز فراهم آورد.

۱-۲ رهیافت پایگاه داده

در این بخش، چند تعریف رسمی از اصطلاح *پایگاه داده* و *سیستم مدیریت پایگاه داده (DBMS)*[۳] ذکر میکنیم که در بخش آتی آنها را مورد استفاده قرار خواهیم داد.

۱-۲-۱ پایگاه داده

اجازه دهید تعریف پایگاه داده را با جزئیات مورد بررسی قرار دهیم تا این مفهوم را کاملا درک کنیم. پایگاه داده، انبار بزرگ و منفردی از داده هاست، که به طور همزمان توسط چندین سازمان و کاربران مورد استفاده قرار میگیرد. تمامی داده های مورد

---

۱ End User
۲ Database Design
۳ Database Management System

نیاز توسط این کاربران با کمترین میزان تکرار یکی شده اند. و از همه مهمتر اینکه پایگاه داده در حالت عادی متعلق به هیچ سازمان یا فردی نیست بلکه یک منبع مشترک میان آنهاست.

پایگاه داده علاوه بر اینکه داده های عملیاتی یک شرکت را ذخیره می کند، بلکه توضیحات این داده ها را نیز نگه میدارد. به این دلیل به پایگاه داده *مجموعه رکوردهای یکی شده خود تعریف*[1]، نیز اطلاق میشود. شرح داده ها، یعنی **ابرداده ها**[2] - یا 'داده هایی درباره داده ها'، بنام **کاتالوگ سیستم**[3] یا **دیکشنری داده**[4] خوانده میشود. طبیعت خود تشریح پایگاه داده است که استقلال داده ها را برآورده میسازد. این بدان معنی است که اگر ساختمان داده جدیدی به پایگاه داده اضافه شود یا ساختمان داده کنونی موجود در پایگاه داده تغییر یابد در این صورت برنامه کاربردی که از پایگاه داده استفاده میکند بدون تغییر باقی میماند، یعنی به طورمستقیم به آنچه که تغییر میکند وابسته نیست. برای مثال اگر ستون یا رکورد جدیدی و یا جدول جدیدی ایجاد کنیم، کاربردهای موجود بدون تاثیر باقی میمانند. با این وجود اگر ستونی را از جدولی که برنامه کاربردی از آن استفاده میکند حذف کنیم، در این صورت برنامه کاربردی توسط این تغییر تحت تاثیر قرار گرفته و بر طبق آن باید تغییر یابد.

آخرین اصطلاح در تعریف پایگاه داده که بایستی توضیح دهیم عبارت **'بصورت منطقی مرتبط'** میباشد. ما زمانیکه اطلاعات مورد نیاز شرکت را تحلیل میکنیم، همواره سعی میکنیم اشیا مهم و ارتباطات منطقی که میان آنها وجود دارد را شناسایی کنیم. متدلوژی را که در این کتاب برای طراحی پایگاه داده ارائه خواهیم داد روشی را جهت شناسایی این اشیا و ارتباطات منطقی بین آنها ارائه خواهد کرد.

۲-۲-۱ سیستم مدیریت پایگاه داده (DBMS)

شاید از مواقعی پیش آمده که از خودتان پرسیده باشید که DBMS چیست؟ در جواب این سوال باید بگوییم که یک DBMS نرم افزاری است که با کاربران، برنامه های کاربردی، و پایگاه داده متقابلا در ارتباط است. علاوه بر این، DBMS به کاربران اجازه میدهد که اطلاعات خود را درج، بهنگام، حذف یا بازیابی کنند. داشتن مخزن داده مرکزی برای همه داده ها و شرح داده ها به DBMS اجازه میدهد امکانات پرس وجوی جامعی به این داده ها را که *زبان پرس وجو* نام دارد فراهم سازد.

استفاده از زبان پرس و جویی (همچون SQL) مشکلات موجود با سیستمهای قبلی را که کاربران در آن با تعداد زیادی از پرس وجو ها کار میکردند که کثرت پرس و جوها مشکلات عمده مدیریت نرم افزار را باعث میشد را کم کرد. در بخش بعدی درباره سرویسها و کارکردهای نمونه DBMS بحث خواهیم کرد.

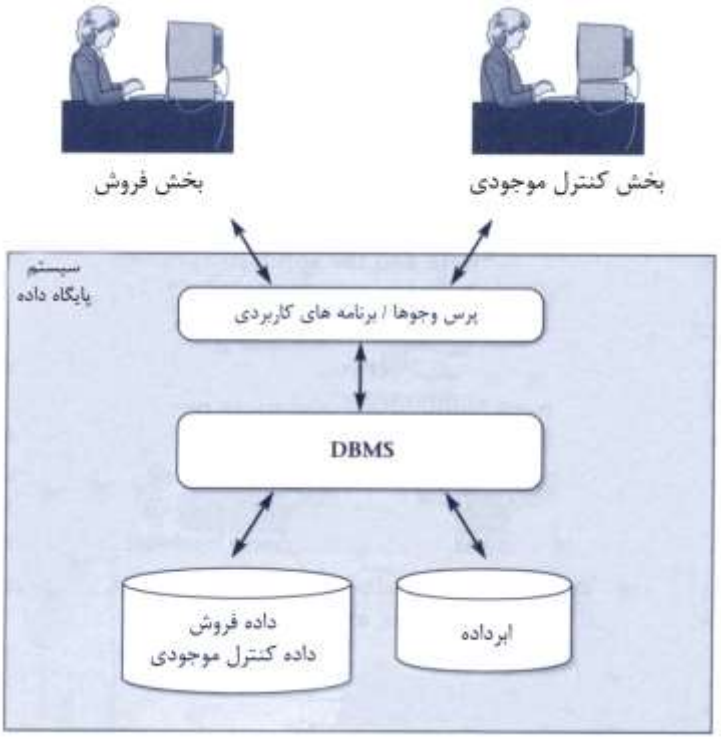
> زبان پرس و جوی ساختیافته (اس- کیو- ال یا بعضا سی- کوئل تلفظ میشود)، یک زبان پرس و جوی مهم برای DBMS های رابطه ای میباشد.

---

[1] Self describing collection of integrated records
[2] Meta-Data
[3] System catalog
[4] Data dictionary

شکل ۱-۱

استفاده بخشهای فروش و کنترل موجودی از DBMS .



شکل ۱-۱ رهیافت پایگاه داده را نمایش میدهد. این شکل قسمت کنترل فروش و موجودی را با استفاده از برنامه های کاربردی شان جهت دسترسی به پایگاه داده از طریق DBMS نشان میدهد. هر مجموعه ای از برنامه های کاربردی اداری، ثبت داده، نگهداری داده، و تولید گزارش را کنترل میکند. ساختمان فیزیکی و ذخیره سازی داده توسط DBMS مدیریت میشود.

۳-۲-۱ دیدها

با کارکردی که در بالا توصیف شد، DBMS ابزاری بسیار قدرتمند است. با این وجود، کاربران نهایی علاقه ای به اینکه کارها را آسان یا پیچیده میکند ندارند، و دلیلی که می آورند این است که DBMS کارها را پیچیده میکند چون ممکن است داده هایی را مشاهده کنند که واقعا برای کار خود به آنها نیاز ندارد. در تشخیص این مشکل، DBMS وسیله دیگری را بنام *مکانیزم* دید (View mechanism) فراهم آورده است؛ که به کاربر این امکان را میدهد که فقط دید مختص به خود را از پایگاه داده داشته باشد. دید زیرمجموعه ای از پایگاه داده است. دیگر امتیازات دیدها را میتوان بدین صورت نام برد:

- *دیدها سطحی از امنیت را فراهم میسازند.* دیدها میتوانند مانع دیده شدن برخی داده ها به کاربران شوند. برای مثال، میتوانیم دیدی را ایجاد کنیم که به مدیر شعبه و بخش مالی اجازه دهد که تمامی داده های مربوط به پرسنل، که شامل جزئیات حقوق است را ببینند. بعلاوه میتوانیم دید دومی را ایجاد کنیم که جزئیات حقوقی را که پرسنل دیگر استفاده میکند را مخفی سازد.
- *دیدها مکانیزمی را جهت سفارشی کردن ظاهر پایگاه داده فراهم میسازد.* برای مثال، بخش کنترل موجودی ممکن است بخواهد که ستون نرخ کرایه روزانه فیلمها را فراخوانی کند.
- دید *میتواند* حتی زمانیکه پایگاه داده زیرین نیز تغییر کند *تصویر غیرقابل تغییر و سازگاری از ساختار پایگاه داده ارائه* دهد(مثلا، ستونهایی افزوده یا حذف شود، ارتباطات تغییر کند، فایلهای داده ای از هم جدا شوند، دوباره سازماندهی شوند، یا تغییر نام داده شوند). اگر ستونهایی از / به فایل داده ای حذف یا اضافه شوند، و این ستونها توسط هیچ دیدی

مورد نیاز نباشند، دید توسط این تغییر تاثیر نمی پذیرد. بنابراین همانطور که در بخش ۱-۲-۱ توضیح خواهیم داد، دید کمک میکند تا استقلال داده ای افزونی که توسط کاتالوگ سیستم فراهم شده را بوجود آورد.

۱-۲-۴ مولفه های محیط DBMS

پنج مولفه اصلی محیط DBMS عبارتند از : سخت افزار، نرم افزار، داده، رویه ها و افراد.

(۱) *سخت افزار* همان سیستمهای کامپیوتری است که DBMS و برنامه های کاربردی روی آن اجرا میشوند. که میتواند محدوده ای را از PC ها تا پردازنده مرکزی و یا شبکه های کامپیوتری در برگیرد.

(۲) *نرم افزار* منظور همان نرم افزار DBMS و برنامه های کاربردی، با همدیگر بهمراه سیستم عامل، و نرم افزار شبکه در صورتیکه DBMS در شبکه مورد استفاده قرار گیرد.

(۳) *داده* داده مانند پلی بین مولفه های سخت افزار و نرم افزاری و مولفه های انسانی عمل میکند. همانطور که گفته شد پایگاه داده در برگیرنده داده های عملیاتی و ابرداده ها میباشد ('داده درباره داده').

(۴) *رویه ها* دستور العمل ها و قواعدی است که طراحی و استفاده از پایگاه داده را کنترل میکند. دستورالعملهایی در مورد نحوه ورود به DBMS ،گرفتن نسخه پشتیبان از پایگاه داده، و چگونگی کنترل خرابی سخت افزار و نرم افزار میباشد.

(۵) *افراد* که شامل طراحان پایگاه داده، مدیران پایگاه داده (DBA)، برنامه نویسان کاربردی، و کاربران نهایی میباشد.

۱-۲-۵ ساختار DBMS

قبل از ظهور وب، DBMS به طورکلی به دو بخش تقسیم بندی میشود:

- برنامه **سرویس گیرنده**[۱] که منطق پردازش تجاری و داده همچنین رابط های میان کاربران را کنترل میکند؛
- برنامه **سرویس دهنده**[۲] (که بعضا موتور DBMS نیز نامیده میشود) دسترسی به پایگاه داده را مدیریت و کنترل میکند.

این ساختار به نام **ساختار دوطرفه کلاینت/ سرور** معروف است. شکل ۲-۱ ساختار ساده شده کلاینت / سرور شرکت کرایه فیلمی که *StayHome* نامیده میشود را نشان میدهد. شکل فوق پایگاه داده متمرکز شده و سرور واقع در سیاتل و تعدادی از کلاینت هایی، که در بعضی از شعبه های اطراف ایالات متحده قرار گرفته است را نشان میدهد.

در اواسط سال ۱۹۹۰ برنامه های کاربردی به طور بالقوه پیچیده شدند و به صدها و هزاران کاربر نهایی گسترش یافتند، و طرف کلاینت این ساختار دارای دو مشکل گردید:

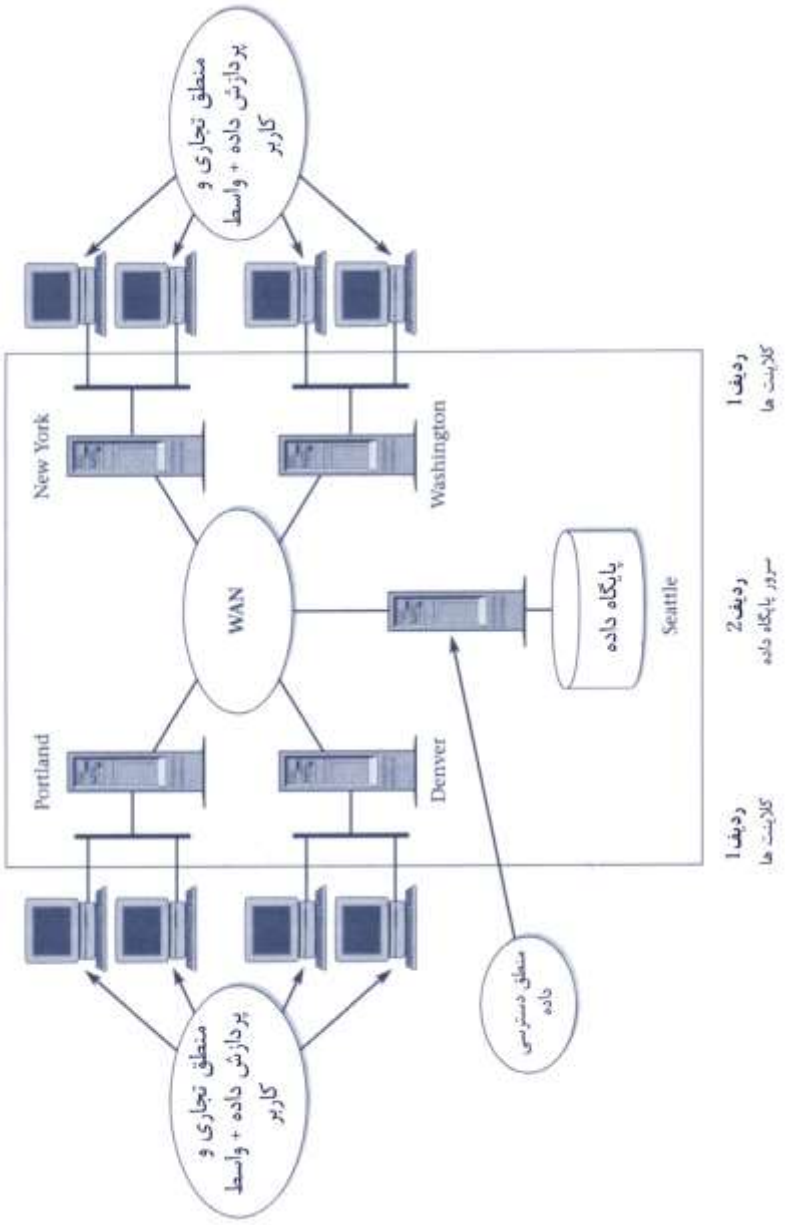
- کلاینت 'چاق' [۳] ، به منبع قابل توجهی درکامپیوتر کلاینت نیاز دارد تا به طور موثر اجرا شود. این شامل فضای روی دیسک، RAM ، و قدرت CPU میباشد.
- هزینه قابل توجه، مدیریت طرف مشتری.

---

[۱] Client
[۲] Server
[۳] Fat client

شکل ۲-۱

پیکربندی ساده شده دو طرفه کلاینت - سرور پایگاه داده *StayHome* .



در سال ۱۹۹۵، تغییرات تازه ای در مدل دو طرفه کلاینت/سرور پدیدار شد تا این دو مشکل را حل کند و بنام **ساختار سه طرفه کلاینت/ سرور** مطرح شد. این ساختار تازه، سه لایه پیشنهاد کرد که هر لایه به طور بالقوه در محل مجزایی اجرا شود:

(۱) لایه واسط کاربر، که روی کامپیوترکاربر نهایی اجرا میشود (*کلاینت*) .

(۲) لایه پردازش داده و منطق تجاری[۱]. این نصف دیگر روی سرور اجرا میشود و اغلب به آن **سرور کاربرد**[۲] نیز اطلاق میگردد. هر سرورکاربرد برای ارائه خدمت به چندین کلاینت طراحی شده است.

(۳) DBMS، که داده های موردنیاز توسط نصف دیگر را ذخیره میکند. این نیمه روی سرور دیگری بنام **سرور پایگاه داده** اجرا میشود.

طراحی سه ردیفه فوق مزیتهایی نسبت به دو ردیفه دارد، همچون:

- کلاینت 'نازک' (Thin client) ، به سخت افزار ارزانتری نیاز دارد.
- تعمیر برنامه کاربردی ساده شده، در نتیجه متمرکز ساختن منطق تجاری برای چندین کاربر نهایی داخل یک سرور برنامه کاربردی میباشد. این عمل نگرانی توزیع نرم افزارهایی را که در مدل دو طرفه قدیمی مشکل زا میباشد حذف میکند.
- افزودن پیمانه ای بودن، باعث میشود که یک ردیف به راحتی با ردیف دیگری بدون تاثیری بر دیگری جایگزین شود.
- سادگی متعادل بار شدن، در نتیجه جداسازی منطقی تجاری هسته از عملیات پایگاه داده است. برای مثال مانیتور کردن پردازش تراکنش (TPM) میتواند جهت کاهش ارتباطات به سرور پایگاه داده استفاده شود. (TPM برنامه ای جهت کنترل انتقال داده بین کلاینت ها و سرورهاست تا محیطی سازگار جهت پردازش تراکنش برخط (OLTP) را فراهم کند.)

امتیاز دیگر ساختار سه طرفه این است که اگر از مرورگر وب به عنوان کلاینت 'نازک'، و سرور وب به عنوان سرور کاربرد استفاده شود در این صورت میتواند به محیط شبکه نگاشت شود. ساختار سه طرفه کلاینت- سرور در شکل ۳-۱ نشان داده شده است.

۳-۱ عملکرد DBMS

در این بخش، عملیات و سرویس هایی که امروزه از یک DBMS انتظار میرود تا برای ما فراهم سازد را به طور مختصر ذکر می کنیم.

ذخیره سازی داده، بازیابی و بهنگام سازی

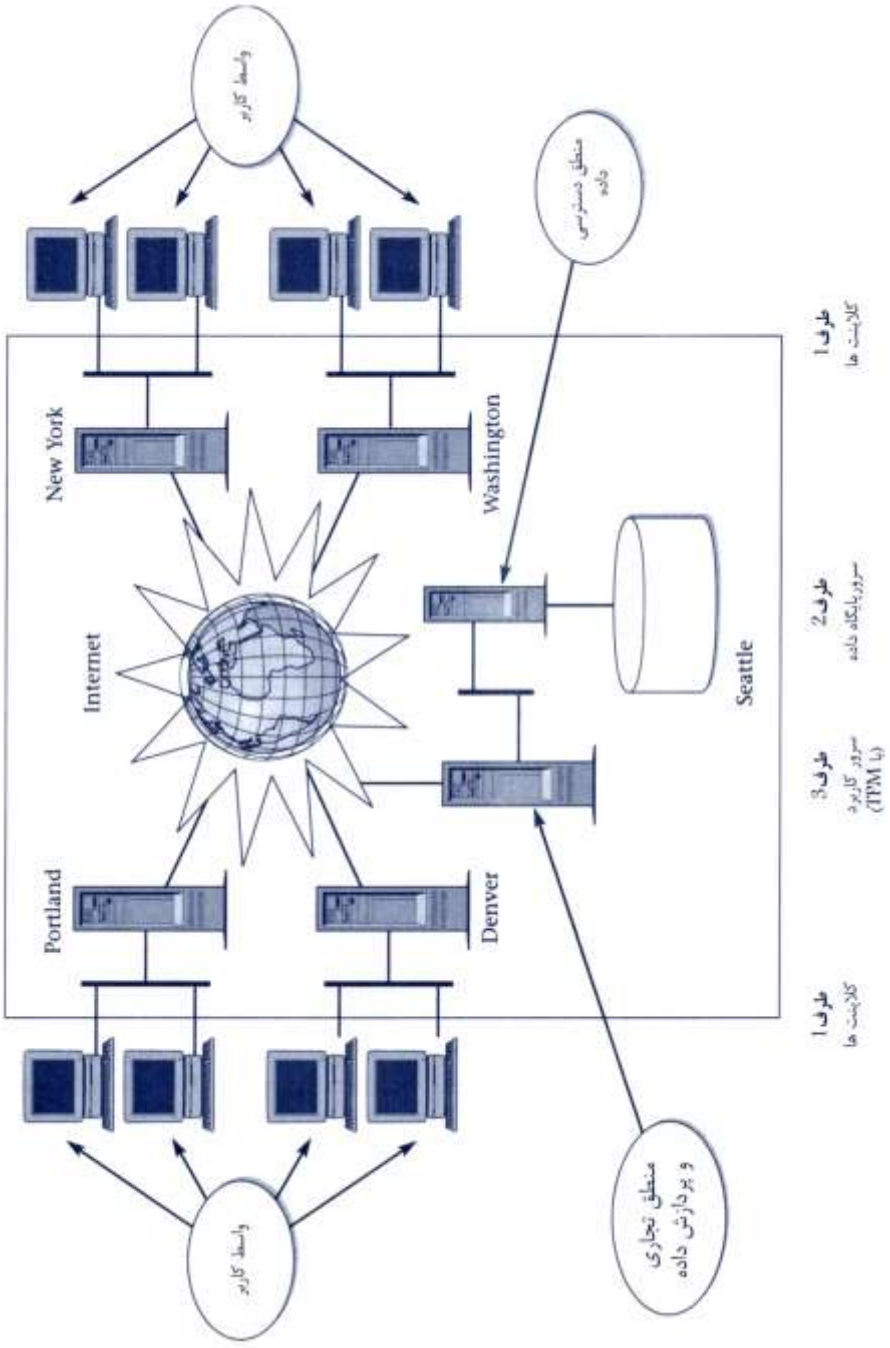
این وظیفه بنیادی یک DBMS است. با توجه به بحث قبلی مان، مشخصا در برآورد کردن این توانایی، DBMS بایستی جزئیات پیاده سازی فیزیکی داخلی (همچون سازمان فایل و ساختار ذخیره سازی) را از دید کاربر پنهان کند.

---

[۱] Business logic
[۲] Application server

شکل ۳-۱

پیکربندی ساده شده سه طرفه برای پایگاه داده *StayHome* .



فهرست دسترس پذیر برای کاربر

خصیصه کلیدی یک DBMS تدارک فهرست سیستمی مجتمع برای نگهداری داده درباره ساختار پایگاه داده، کاربران، برنامه های کاربردی، و غیره است. انتظار می رود که فهرست برای کاربران همچون DBMS قابل دسترس باشد. میزان اطلاعات و روشی که مورد استفاده قرار می گیرد در DBMS ها متفاوت است. به طور نمونه کاتالوگ سیستم موارد زیر را ذخیره می کند:

- نامها، انواع، و اندازه های اقلام داده ای ;
- محدودیتهای جامعیتی بر روی داده ها ;
- نام کاربران مجاز که به داده دسترسی دارند ;

پشتیبانی از تراکنش

چند تراکنش ساده که میتوان برای شرکت اجاره فیلم ***StayHome*** در نظر گرفت ممکن است افزودن عضوی از پرسنل به پایگاه داده، بهنگام سازی حقوق عضو خاصی از پرسنل، یا حذف عضوی از لیست ثبت نام باشد. مثال پیچیده تر میتواند حذف مدیر از پایگاه داده و انتساب دوباره شعبه ای که او مدیریت میکرد به عضو دیگری از پرسنل باشد. در این مورد، بیش از یک تغییر در پایگاه داده وجود دارد. اگر تراکنش هنگام اجرا و یا شاید به علت خرابی کامپیوتر با شکست مواجه شود، پایگاه داده در یک وضعیت *ناسازگار* قرار میگیرد: بعضی تغییرات اعمال میشوند و برخی دیگر اعمال نمیشوند. برای مثال، به شعبه هیچ مدیری اختصاص نمیابد. در نتیجه، تغییرات اعمال شده به پایگاه داده جهت برگشت به حالت سازگار ناتمام خواهد ماند.
برای غلبه بر این، جهت اطمینان از اینکه همه بهنگام سازی بر پایگاه داده اعمال شده است یا هیچ یک اعمال نشده DBMS بایستی مکانیزمی را فراهم سازد.

سرویس های کنترل همزمانی

یکی از اهداف مهم DBMS این است که چندین کاربر بطور همزمان بتوانند به داده های مشترک دسترسی پیدا کنند، که این عمل**کنترل همزمانی**[1] نام دارد. دستیابی همزمان برای همه کاربران فقط جهت خواندن داده ها به طوریکه هیچ کاربری نتواند با دیگری برخورد داشته باشد نسبتا ساده است. با این وجود، وقتیکه دو یا چند کاربر به طور همزمان یا حداقل یکی از آنها داده ای را بهنگام کند، ممکن است تداخل باعث ناسازگاری شود. برای مثال، دو تراکنش T1 و T2 را در نظر بگیرید که به طور همزمان مانند شکل ۴-۱ اجرا می شوند.

[1] Concurrency control

شکل ۴-۱

مشکل از دست رفتن بهنگام سازی.

Time	T_1	T_2	bal_x
t_1		read(bal_x)	50
t_2	read(bal_x)	$bal_x = bal_x - 20$	50
t_3	$bal_x = bal_x + 5$	write(bal_x)	30
t_4	write(bal_x)		55
t_5			

T2 مبلغ ۲۰$ را از حساب عضو *StayHome* (bal_x، به میزان ۵۰$) گرفته و T1 سود ۵$ را به همان حساب داده است. اگر این تراکنشها یکی پس از دیگری بدون هیچ فاصله ای بین عملیات اجرا شوند، موجودی نهایی صرفنظر از اینکه کدامیک اول انجام گرفته باشد ۳۵$ خواهد شد. تراکنش T1 و T2 در یک زمان مشابه شروع شده و هر دو موجودی را در ۵۰$ میخوانند. T2 ، bal_x را از ۲۰$ به ۳۰$ کاهش داده و مقدار آپدیت شده را در پایگاه داده ذخیره میکند. ضمنا، تراکنش T1 کپی bal_x را از ۵$ تا ۵۵$ افزایش داده و نتایج را در پایگاه داده با نوشتن دوباره روی بهنگام سازی قبلی و البته با از دست دادن ۲۰$ ذخیره کرده است. وقتیکه چندین کاربر به پایگاه داده دسترسی دارند، DBMS بایستی مطمئن شود که برخوردهای این چنینی دیگر رخ نخواهند داد.

سرویسها ی ترمیم

از بحث درباره پشتیبانی تراکنش، بیاد داریم که اگر تراکنشی به شکست بیانجامد پایگاه داده به حالت سازگار برگردانده میشود، که این عمل **کنترل ترمیم**[۱] نام دارد. این شکست در نتیجه خرابی سیستم، شکست رسانه ای، یا اینکه سخت افزار یا نرم افزار است که DBMS متوقف شود، یا اینکه کاربر قبل از اینکه تراکنشی کامل شود به علت مواجه شدن با خطایی تراکنش را لغو کرده است. در تمامی این موارد، DBMS بایستی مکانیزمی را جهت بازگرداندن پایگاه داده به حالت سازگار فراهم کند.

سرویس های مجوز دهی

هنگام برخورد با مواردی که می خواهیم داده های ذخیره شده در پایگاه داده را از دید دیگر کاربران محافظت کنیم مشکلی وجود ندارد. برای مثال، می خواهیم تنها مدیران شعبه و بخش حقوق اطلاعات مربوط به حقوق پرسنل را مشاهده کنند و دیگر کاربران قادر به دیدن این داده ها نباشند. به علاوه، می خواهیم پایگاه داده را از دستیابی غیرمجاز حفاظت کنیم. عبارت **امنیت** به حفاظت پایگاه داده از دسترسی غیر مجاز که چه داخلی باشد و چه تصادفی اشاره دارد. ما از DBMS انتظار داریم تا مکانیزمی را جهت اطمینان از امن بودن داده ها فراهم سازد.

پشتیبانی برای ارتباط داده ای

بیشتر کاربران از طریق پایانه ها به پایگاه داده دسترسی دارند. بعضی اوقات، این پایانه ها مستقیما به کامپیوتری که پایگاه داده در آن وجود دارد متصل است. در دیگر موارد، پایانه ها در محل های دور دست بوده و در شبکه با کامپیوتر میزبان پایگاه داده ارتباط دارند. در هر دو مورد، DBMS بایستی توانایی یکی شدن با نرم افزار شبکه/ ارتباط را دارا باشد. حتی DBMS های پی سی ها نیز بایستی توانایی اجرا شدن در شبکه های محلی (LAN) را دارا باشند و بنابراین به جای اینکه برای هرکاربر چندین پایگاه داده داشته باشیم درعوض میتواند یک پایگاه داده متمرکز شده بین کاربران به اشتراک گذارده شود .

[۱] Recovery control

سرویسهای جامعیتی

جامعیت داده[1] به بی عیبی و سازگاری داده های ذخیره شده اشاره دارد. به عبارت دیگر نوع دیگری از حفاظت پایگاه داده به شمار میرود و وقتی به امنیت ارتباط میابد، معانی وسیعتری پیدا میکند. جامعیت مربوط به کیفیت داده های درونی است. و معمولا در رابطه با محدودیتها بیان میشود، که این محدودیتها در واقع قواعد سازگاری هستند که به پایگاه داده اجازه ناسازگاری نمیدهند. برای مثال، ممکن است بخواهیم محدودیتی مشخص کنیم که درآن هیچ عضوی از ***StayHome*** در هر بار نتواند بیش از ۱۰ فیلم اجاره کند. اینجا، میخواهیم DBMS بررسی کند تا کاربر از این محدودیت تجاوز نکرده باشد و اگر تخطی رخ دهد از دادن اجازه به کرایه بیش از ۱۰ فیلم جلوگیری بعمل آورد.

سرویس هایی جهت بکارگیری استقلال داده ای

همانطور که در بخش ۳-۲-۱ بحث شد، استقلال داده ای به طور معمولی از طریق مکانیزم دید فراهم میشود. معمولا چندین نوع از تغییرات میتوانند بدون اینکه به دیدها تاثیر بگذارند به خصوصیات فیزیکی پایگاه داده اعمال شوند، همچون استفاده از سازمان متفاوتی از فایل یا اصلاح ایندکسها، که این بنام *استقلال فیزیکی* داده خوانده میشود. با این وجود، رسیدن به *استقلال داده ای منطقی* کامل کمی مشکل است. افزودن ستون یا جدول جدید معمولا میتواند انجام گیرد، اما حذف آنها اینطور نیست. در بعضی سیستمها، هر نوعی از تغییر در جدول کنونی ممنوع میباشد.

سرویس های سودمندی

برنامه های سودمند به DBA کمک میکنند تا پایگاه داده را به طور موثر مدیریت کنند. بعضی از برنامه های سودمند به صورت زیر هستند:

- وارد کردن امکاناتی جهت لود کردن از فایلهای مسطح[2]، و صادرکردن امکاناتی جهت خارج کردن پایگاه داده به صورت فایلهای مسطح (فایل با پسوند txt).
- امکانات مونیتور کردن، جهت نظارت بر عمل و استفاده از پایگاه داده .

---

[1] Data Integrity
[2] Flat file

۱-۴ طراحی پایگاه داده

تا حالا، آنچه که دیدیم این بود که در پایگاه داده ساختاری برای داده ها وجود دارد. اما سوال اینجاست که چگونه به این ساختار دست یابیم؟ جواب کاملا ساده است: ساختار پایگاه داده هنگام **طراحی پایگاه داده** مشخص میشود. با این وجود، عمل طراحی پایگاه داده میتواند فرآیندی پیچیده باشد. برای ارائه سیستمی که نیازهای اطلاعاتی شرکت را برآورد سازد به یک رهیافت مبتنی بر داده نیاز است، و این بدین معنی است که ما به داده قبل از کاربردشان میاندیشیم. برای رسیدن به سیستمی که در نظر کاربران نهایی قابل قبول باشد، طراحی پایگاه داده یک امر سخت تلقی میشود. طراحی ضعیف پایگاه داده اشتباهاتی را تولید خواهد کرد که ممکن است منجر به گرفتن تصمیمات اشتباه شده و انعکاس جدی برای شرکت داشته باشد. از طرف دیگر یک طراحی خوب سیستمی را ایجاد میکند که اطلاعات صحیحی را برای پروسه تصمیم گیری در یک مسیر موثر فراهم میسازد.

در این کتاب چندین بخش را جهت ارائه متدلوژی کامل طراحی پایگاه داده اختصاص داده ایم (بخش های ۱۶-۷). که در این بخش ها یک سری گام به گام جهت دنبال کردن مطالب معرفی میکنیم. در این بخش ها، از بررسی موردی یک شرکت اجاره فیلم های ویدیویی بنام *StayHome* استفاده میکنیم. برای تقویت خود در متدولوژی، در فصل های ۱۸ و ۱۹ دومین بررسی موردی کلینک دامپزشکی با نام *Perfect Pets* را مورد مطالعه قرار میدهیم. به علاوه، در ضمیمه (د) تعدادی از مدلهای داده ای تجاری عمومی را فراهم کرده ایم که میتوانید آنها را یکی پس از دیگری مطالعه کنید.

متاسفانه، متدلوژیهای طراحی پایگاه داده خیلی محبوب نیستند، که دلیل اصلی شکست در توسعه سیستمهای پایگاه داده به شمار میروند. به علت کمبود روشهای ساخت یافته در طراحی پایگاه داده، زمان و منابع مورد نیاز برای پروژه های پایگاه داده معمولا ناچیز بوده، و توسعه های پایگاه داده ناکافی هستند یا در مواجه با تقاضا ها ناموثر میباشند. همچنین مستندات محدود بوده، و نگهداری مشکل میباشد. امیدواریم که متدولوژیهای ارائه شده در این کتاب این رهیافت را تغییر دهد.

۱-۵ مزایا و معایب DBMS ها

از آنجائیکه شما در حال مطالعه این کتاب هستید ممکن است با بعضی از امتیازات DBMS ها همچون موارد زیر آشنا شده باشید:

- *کنترل **افزونگی داده*** رهیافت پایگاه داده افزونگی را تا جائیکه ممکن باشد حذف میکند. با این وجود، افزونگی داخلی را از بین نمیبرد، بلکه میزان افزونگی اصلی پایگاه داده را کنترل میکند. برای مثال، معمولا بعضی ارقام داده ای که برای مدل کردن ارتباطات نیاز است ممکن است تکراری باشد تا کارایی بهبود یابد. با مطالعه فصلهای مربوط به طراحی پایگاه داده علت ایجاد تکرارها در رابطه ها برایتان آشکار خواهد شد.
- ***سازگاری داده*** با حذف یا کنترل افزونگی، ریسکهای ناشی از ناسازگاری های رخداده را کاهش میدهیم. اگر اقلام داده ای فقط یکبار در پایگاه داده ذخیره شده باشند، بهنگام سازی مقادیر فقط یکبار انجام گرفته و مقدار جدید فورا در دسترس تمام کاربران قرار میگیرد. و اگر اقلام داده ای بیش از یکبار در پایگاه داده ذخیره شوند و سیستم از این آگاه باشد، سیستم میتواند مطمئن شود که تمامی کپی های اقلام به صورت سازگار نگه داشته شده اند. متاسفانه، تعدادی از DBMS های امروزی به طور اتوماتیک از این نوع همزمانی مطمئن نمیشوند.
- ***اشتراک داده ای*** در رهیافت مبتنی بر فایل، فایلها توسط افراد یا بخشهایی که آنها را استفاده میکنند مالکیت میشوند. از سوی دیگر، پایگاه داده متعلق به تمام شرکت بوده و احتمال دارد توسط تمامی کاربران مجاز به اشتراک گذارده شود. در اینجا، بیشتر کاربران داده های زیادی را به اشتراک می گذارند. علاوه بر این، کاربردهای جدیدی میتواند بر روی داده های موجود در پایگاه داده ساخته شود و به جای اینکه همه داده ها دوباره تعریف شوند داده هایی که در حال حاضر در

پایگاه داده وجود ندارند فقط افزوده و ذخیره شود. همچنین کاربردهای جدید میتواند به جای اینکه این عملیات را خودشان برای خود فراهم کنند بر عملیاتی همچون تعریف و دستکاری داده، کنترل و ترمیم و همزمانی که DBMS فراهم کرده است، تکیه کنند.

- *جامعیت بهبود یافته داده* همانطور که میدانیم، جامعیت پایگاه داده معمولا در معنی **محدودیت ها**[1] بیان میشود،که این محدودیتها بصورت قواعد سازگاری هستند که به پایگاه داده اجازه اختلال در سازگاری را نمیدهند. محدودیتها میتوانند به رکورد منفرد یا به روابط میان رکوردها اعمال شوند. بعلاوه، یکپارچگی به ما اجازه تعریف، و به DBMS اجازه اجرای محدودیتهای جامعیتی را میدهد.
- *نگهداری بهبودیافته از طریق استقلال داده ای* از آنجائیکه DBMS توصیفهای داده را از کاربرد آن جدا میکند، این عمل کمک میکند تا کاربردها را در تغییر توصیفهای داده ای ایمن کنیم. این به عنوان *استقلال داده ای* مشهور بوده و قوانین آن نگهداری کاربرد پایگاه داده را ساده میکند.

دیگر امتیازات شامل: امنیت بهبودیافته، تاثیرپذیری و دسترس پذیر بودن داده، همزمانی توسعه یافته، و بهبود یافتن سرویس های ترمیم و پشتیبان گیری میباشد.

با این وجود، معایبی هم در رهیافت پایگاه داده وجود دارد که به این شرح است:

- ***پیچیدگی*** همانطور که به یاد داریم، DBMS یک تکه به شدت پیچیده ای از نرم افزار است، و تمام کاربران (اعم از طراحان پایگاه داده و توسعه دهندگان، DBA ها، و کاربران نهایی) بایستی با عملکرد آن کاملا آشنا باشند تا بتوانند از تمام امتیازات آن بهره ببرند.
- ***هزینه DBMS*** هزینه یک DBMS به طور عمده با تکیه بر محیط و قابل استفاده بودن آن متغیر است. برای مثال، یک DBMS تک کاربره ممکن است تنها ۱۰۰$ هزینه داشته باشد در حالیکه DBMS هایی که به چندین صد کاربر سرویس میدهند خیلی گران بوده و هزینه هایی، شاید از ۱۰۰۰۰۰$ تا ۱۰۰۰۰۰۰ $ داشته باشند. همچنین هزینه نگهداری سالیانه نیز میتواند وجود داشته باشد.
- *هزینه تبدیل* در بعضی مواقع، هزینه DBMS و هر سخت افزار اضافی ممکن است در مقایسه با هزینه تبدیل DBMS کنونی جهت اجرا شدن در سخت افزار جدید بسیار ناچیز باشد. این هزینه شامل آموزش پرسنل برای استفاده از سیستم جدید، یا استخدام پرسنل ویژه ای جهت کمک در تغییر و اجرای سیستم جدید باشد. این هزینه ها دلیل اصلی اینست که چرا بعضی شرکتها به استفاده از سیستم کنونی اکتفا کرده و مایل به تبدیل به سیستم جدید نیستند. عبارت 'سیستم موروثی[2] ' بعضی اوقات اشاره به سیستمهای نامرغوب و قدیمی دارد، (همچون سیستمهای مبتنی بر فایل، سلسله مراتبی، یا سیستمهای شبکه ای).
- *کارایی* به طورخاص، سیستمهای مبتنی بر فایل برای کاربرد به خصوصی همچون صدور صورت حساب نوشته میشوند. در نتیجه، کارایی عموما خیلی خوب است. اما، DBMS بیشتر برای فراهم ساختن سرویسهایی جهت چندین کاربرد به جای یک کاربرد به صورت عمومی نوشته میشود. به این علت ممکن است که بعضی از برنامه ها به صورت سریع اجرا نشوند.
- *بالاترین اثر شکست* متمرکزسازی منابع، آسیب پذیری سیستم را بالا میبرد. زمانیکه تمام کاربران و برنامه های کاربردی تکیه بر دسترس پذیری DBMS میکنند، خرابی بعضی قسمتها ممکن است بر دیگر قسمتها تا زمان بهبودی قسمت آسیب دیده اثر بگذارد.

---

[1] Constraint
[2] Legacy system

خلاصه فصل

- ✓ **پایگاه داده** مجموعه مشترکی از داده های منطقا به هم مرتبط (و توصیف این داده ها) است، که جهت برآوردکردن نیازهای اطلاعاتی شرکت طراحی میشود. **DBMS** سیستم نرم افزاری است که کاربران را قادر میسازد تا پایگاه داده را تعریف، ایجاد و نگهداری کرده و دسترسی کنترل شده ای را به پایگاه داده فراهم می آورد.
- ✓ تمام دسترسی به پایگاه داده از طریق DBMS است. DBMS امکاناتی را فراهم میاورد و به کاربران اجازه میدهد تا پایگاه داده را تعریف و داده ها را، درج، حذف، بهنگام و بازیابی کنند.
- ✓ محیط DBMS شامل سخت افزار(کامپیوتر)، نرم افزار(DBMS، سیستم عامل و برنامه های کاربردی)، داده، رویه ها، و افراد میباشد. افراد شامل مدیران پایگاه داده (DBA ها)، طراحان پایگاه داده، برنامه نویسان کاربردی، و کاربران نهایی میباشند.
- ✓ DBMS دسترسی کنترل شده ای را برای پایگاه داده فراهم میسازد. همچنین امنیت، کنترل ترمیم و همزمانی و فهرست دسترس پذیر توسط کاربر را نیز فراهم میکند. همچنین مکانیزم دید را نیز جهت مواجه شدن ساده کاربران با پایگاه داده فراهم میکند.
- ✓ بعضی امتیازات پایگاه داده شامل کنترل افزونگی داده ها، اشتراک داده ای، و جامعیت و امنیت بهبودیافته میباشد. بعضی معایب شامل پیچیدگی، هزینه، کاهش کارایی، و تاثیر زیاد شکست است.

فصل ۲

مدل رابطه ای

آنچه در این فصل خواهید آموخت :

- مدل داده ای چیست و کاربردهایش چیست .
- اصطلاحات فنی مربوط به مدل رابطه ای .
- چگونه جدولها جهت نمایش داده استفاده میشوند .
- خصوصیات ارتباطات پایگاه داده .
- چگونه کلیدهای کاندید، اصلی، فرعی و خارجی را مشخص کنیم.
- معانی جامعیت موجودیّت و جامعیت ارجاع .
- درباره SQL و QBE، دو زبان رابطه ای فراگیر .

سیستم مدیریت پایگاه داده رابطه ای (که اغلب مختصرا RDBMS خوانده میشود) ، با فروش تخمینی ۱۰- ۸ میلیون دلار در سال (۲۵ میلیون دلار با ابزار جانبی)، و با رشد نرخ تقریبی ۲۵ درصد در سال امروزه بعنوان یکی از نرم افزارهای اصلی پردازش داده بشمار میرود. این نرم افزار نسل دوم DBMS بوده و بر اساس مدل داده رابطه ای که توسط دکتر کاد پیشنهاد شده در مقاله اصلی وی با عنوان 'مدل رابطه ای داده برای بانکهای بزرگ داده اشتراکی' در سال ۱۹۷۰ ارائه شده است. در مدل رابطه ای، همه داده به طور منطقی به صورت ساختاری متشکل از *رابطه ها*(جدولها) میباشد. نکته قوت مدل رابطه ای در همین ساختار ساده منطقی است. با این حال، پشت این ساختار ساده مفاهیم نظری بنیادی وجود داردکه کمبود آن در ساختارهای نسل اول DBMS (DBMS های شبکه ای و سلسله مراتبی) احساس میشد.

متدلوژی طراحی که ما در این کتاب ارائه کرده ایم مبتنی بر مدل داده ای رابطه ای بوده، و درحقیقت یکی از مهمترین مدلهایی است که بکار میبریم. در این بخش، درباره اصول و قواعد اساسی مدل داده رابطه ای بحث خواهیم کرد. اجازه دهید بحث را با این مطلب شروع کنیم که مدل داده ای چیست.

۲-۱ مدل داده چیست؟

مدلی است که بیانگر رویدادها و اشیا و رابطه های'جهان واقعی' میباشد. و روی جنبه های اصلی و ضروری یک شرکت تمرکز کرده و از خصوصیات تصادفی شرکت چشمپوشی میکند. **مدل داده ای** قصد دارد که شرکت یا قسمتی از آن را که میخواهید مدل کنید نشان دهد. مدل داده ای نمادها و مفاهیم اصلی را که به کاربران و طراحان پایگاه داده اجازه میدهد تا ارتباط

دانسته های خود را به صورت بدون ابهام و دقیق با داده های شرکت برقرار سازند را فراهم میاورد. مدل داده ای از ترکیب سه مولفه زیر تشکیل شده است :

(۱) *قسمت ساختاری*[۱]، که شامل مجموعه ای از قواعدی است که بیان میکند چگونه یک پایگاه داده ساخته میشود;
(۲) *قسمت دستکاری*[۲] ، که انواع عملیاتی که بر روی داده ها مجاز هستند را تعریف می کنند. (مانند عملیاتی که برای بهنگام سازی یا بازیابی داده و برای تغییر ساختمان پایگاه داده بکار میرود).
(۳) مجموعه *قواعد جامعیتی*، که اطمینان میدهد که داده دقیق است.

هدف مدل داده ای نمایش داده و قابل فهم ساختن آن است. که اگر انجام گیرد، به سادگی جهت طراحی پایگاه داده استفاده میشود. در ادامه این بخش، یک چنین مدل داده ای را مورد بررسی قرار میدهیم *: مدل داده رابطه ای*[۳] .

۲-۲ اصطلاحات فنی

مدل رابطه ای مبتنی بر مفهوم ریاضی رابطه است، که به صورت فیزیکی بصورت **جدول** نمایش داده میشود. آقای کاد، ریاضیدان فرهیخته، از اصطلاحات ریاضی، و اکثرا از تئوری مجموعه ها و منطق گزاره ای در مقاله خود استفاده کرده است. در این بخش، اصطلاحات فنی و مفاهیم ساختاری مدل رابطه ای را توضیح میدهیم. در بخش۳-۲، قواعد جامعیتی در مورد مدلها و در بخش ۴-۲ قسمت دستکاری نمودن داده ها را مورد بررسی قرار خواهیم داد .

۲-۲-۱ ساختار داده رابطه ای

DBMS رابطه ای تنها از پایگاه داده ای که به صورت جدول از کاربر دریافت کرده است استفاده میکند.

توجه کنید که، این درک تنها به طریقی که پایگاه داده را میبینیم اعمال میشود; و به ساختار فیزیکی پایگاه داده روی دیسک، که میتوان توسط انواع ساختارهای ذخیره سازی (همچون فایلهای پشته ای یا فایلهای درهم) پیاده سازی کرد اعمال نمیشود.

در مدل رابطه ای، ما از رابطه ها به عنوان نگهدارنده اطلاعات درباره اشیایی که میخواهیم در پایگاه داده نشان دهیم استفاده میکنیم. ما رابطه را بعنوان جدولی که سطرهای جدول با رکوردهای منحصر بفرد مرتبط بوده و ستونهای جدول برابر با **صفت های خاصه** جدول میباشد در نظر میگیریم. صفات خاصه میتواند با حفظ همان رابطه در هر ترتیبی پدیدار شود و بنابراین همان معنی را میدهد.

برای مثال، در شرکت اجاره فیلم *StayHome* ، اطلاعات مربوط به شعبه ها با ارتباط Branch، بهمراه ستونهایی به عنوان صفات خاصه branchNo (شماره شعبه) ،street، city، state، zipCode ، و mngStaffNo (شماره پرسنلی در ارتباط با مدیر شعبه) نشان داده میشود. به طور مشابه، اطلاعات مربوط به پرسنل با رابطه Staff، بهمراه ستونهایی برای صفات خاصه staffNo (شماره پرسنلی)، name، position، salary، و branchNo (شماره شعبه ای که پرسنل عضو در

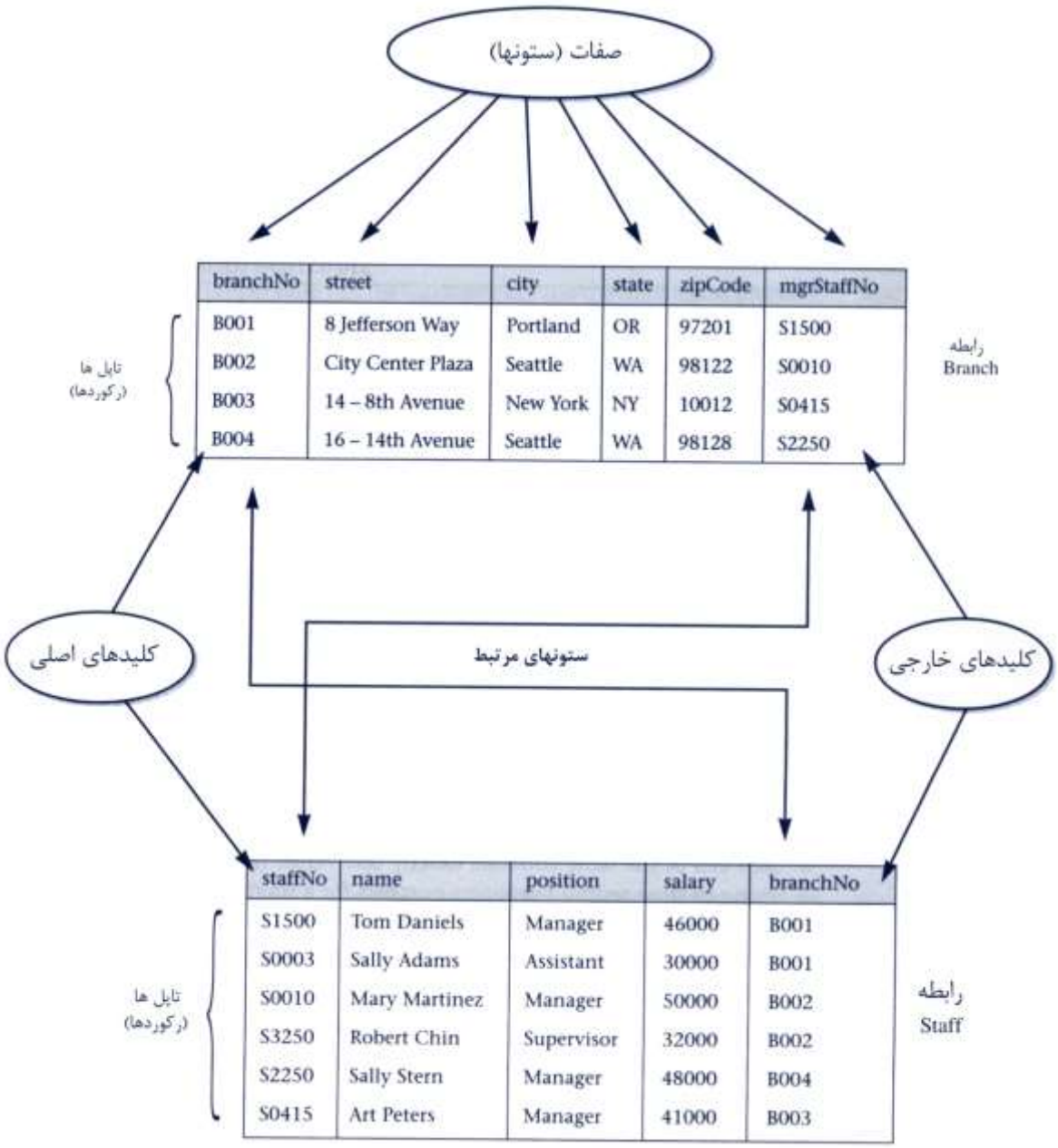
---

[۱] Structural part
[۲] Manipulative part
[۳] Relational data model

آن کار میکند) نشان داده میشود. شکل۱-۲ نمونه هایی از رابطه های Branch و Staff را نشان میدهد. همانطور که در این مثال میبینید، ستون برای هر صفت خاصه مقادیر مشخصی دارد; برای مثال ستون branchNo فقط شامل تعداد شعبه ها است.

شکل ۱-۲

مثالهایی از رابطه های Branch و Staff .



branchNo	street	city	state	zipCode	mgrStaffNo
B001	8 Jefferson Way	Portland	OR	97201	S1500
B002	City Center Plaza	Seattle	WA	98122	S0010
B003	14 – 8th Avenue	New York	NY	10012	S0415
B004	16 – 14th Avenue	Seattle	WA	98128	S2250

staffNo	name	position	salary	branchNo
S1500	Tom Daniels	Manager	46000	B001
S0003	Sally Adams	Assistant	30000	B001
S0010	Mary Martinez	Manager	50000	B002
S3250	Robert Chin	Supervisor	32000	B002
S2250	Sally Stern	Manager	48000	B004
S0415	Art Peters	Manager	41000	B003

شکل ۲-۲

دامنه های بعضی از صفات رابطه های Branch و Staff.

صفت	نام دامنه	معنی	تعریف دامنه
branchNo	Branch_Numbers	Set of all possible branch numbers.	Character: size 4, range B001–B999
street	Street_Names	Set of all possible street names.	Character: size 60
staffNo	Staff_Numbers	Set of all possible staff numbers.	Character: size 5, range S0001–S9999
position	Staff_Positions	Set of all possible staff positions.	One of Director, Manager, Supervisor, Assistant, Buyer
salary	Staff_Salaries	Possible values of staff salaries.	Monetary: 8 digits, range $10 000.00–$100 000.00

دامنه ها مشخصه مهمی از مدل رابطه ای میباشند. هر صفت موجود در پایگاه داده رابطه ای، دارای یک دامنه است. دامنه ها ممکن است برای هر صفت مجزا باشد، یا دو یا چند صفت ممکن است به دامنه مشابهی مربوط شوند. شکل۲-۲ دامنه های بعضی از صفات رابطه های Branch و Staff را نشان می دهد.

مفهوم دامنه به این علت مهم است که به ما اجازه میدهد تا معنی و منبع مقادیری که صفات میتواند نگه دارد را تعریف کنیم. در نتیجه، اکثر اطلاعات در دسترس سیستم بوده و سیستم میتواند عملیات غیر منطقی را رد کند. برای مثال، اگرچه دامنه صفات شماره پرسنلی و شماره شعبه رشته های کاراکتری هستند اما مقایسه این دو با هم کار منطقی نیست. متاسفانه، تعدادی از RDBMS هایی را خواهید یافت که دامنه ها را پشتیبانی نمیکنند.

عناصر رابطه در جدول **تاپل ها** یا **رکوردها** هستند. در رابطه Staff هر رکورد شامل پنج مقدار است که، هرکدام برای یک صفت است. همچون صفات، تاپل ها می توانند در هر ترتیبی قرار گیرند بطوریکه رابطه هنوز همان رابطه مشابه باشد، و بنابراین معنی مشابهی را می رسانند.

پایگاه داده رابطه ای[۱] شامل جداولی است که به طور مناسبی ساخت یافته می باشد. ما به این تناسب تحت عنوان *نرمال*[۲] *سازی* رجوع خواهیم کرد. همچنین بحث نرمال سازی را تا فصل ۶ به تعویق خواهیم انداخت .

چند اصطلاح فنی دیگر

اصطلاحات فنی مدل رابطه ای شاید کمی گیج کننده باشد. در این بخش، می خواهیم دو مجموعه از اصطلاحات را معرفی کنیم: (رابطه، صفت، تاپل) و (جدول، ستون، رکورد). عبارات دیگری که ممکن است برخورد کنید یکی **فایل** برای جدول، سطر برای رکورد، و **فیلد** برای ستون می باشد. همچنین ممکن است با ترکیبهای متفاوتی از این اصطلاحات، همچون جدول، سطر و فیلد نیز مواجه شوید.

از این به بعد، به جای اصطلاحات رسمی همچون ارتباط، تاپل، و صفت اصطلاحاتی همچون جدول، رکورد و ستون را که بیشتر استفاده می شوند به کار میبریم.

---

۱ Relational database
۲ Normalization

۲-۲-۲ ویژگیهای جداول رابطه ای

جدول رابطه ای دارای مشخصات زیر است:

- هر جدول دارای نامی است که آن را از دیگر جدولهای موجود در پایگاه داده مشخص می کند.
- هر خانه جدول دقیقا یک مقدار دارد. (برای مثال، اشتباه است که چندین شماره تلفن را برای یک شعبه در یک خانه ذخیره کنیم. به عبارت دیگر، جدول نباید شامل گروههای تکراری[۱] باشد. جدول رابطه ای که این خصیصه را برآورد کند **نرمال شده** یا در **فرم نرمال اول** گفته می شود.)
- هر ستون نام مجزایی دارد.
- مقادیر ستونها همگی از دامنه مشابهی هستند.
- ترتیب ستونها مهم نیستند. به عبارت دیگر، میتوانیم جای ستونها را با هم عوض کنیم.
- هر رکورد مجزا میباشد; هیچ رکورد تکراری وجود ندارد.
- ترتیب رکوردها در عمل مهم نیست. (با این وجود، درتمرین، همانگونه که در فصل ۱۳ خواهیم دید ترتیب، در نحوه دستیابی رکوردها وکارایی آنها موثر است.)

۲-۲-۳ کلیدهای رابطه ای

همانطور که قبلا ذکر شد، هر رکورد موجود در جدول بایستی یکتا باشد. این بدین معنی است که باید بتوانیم ستون یا ترکیبی از ستونهایی (که کلیدهای رابطه ای نامیده می شوند) را مشخص کنیم که یکتایی را میسر سازند. در این بخش، اصطلاحاتی که در کلیدهای رابطه ای استفاده می شوند را توضیح می دهیم.

از آنجاییکه **ابرکلید**[۲] ممکن است دارای ستونهای اضافی باشد که برای مشخص کردن یکتایی لازم نیست، بنابراین علاقه داریم ابرکلیدهایی را مشخص کنیم که فقط *کمترین تعداد* ستونهای لازم برای شناساندن یکتایی داشته باشد.

کلید کاندیدای[۳] جدول دارای دو ویژگی است:

- *یکتایی*[۴] در هر رکورد، مقدار کلید کاندید منحصرا آن رکورد را مشخص می کند.
- *غیر قابل کاهش بودن*[۵] هیچ زیر مجموعه مناسبی از کلید کاندید ویژگی یکتایی ندارد.

یک جدول ممکن است چندین کلید کاندید داشته باشد. وقتی که کلید دو یا چند ستون را در برگیرد، آن را **کلید ترکیبی**[۶] می نامیم. جدول Branch را که در شکل ۱-۲ نشان داده شده در نظر بگیرید. برای مقدار مفروض city، انتظار داریم که بتوانیم چندین شعبه (branch) را مشخص کنیم. (برای مثال Seattle دو شعبه دارد). این ستون نباید بعنوان کلید کاندید مشخص شود. در طرف دیگر، از آنجاییکه *StayHome* برای هر شعبه شماره شعبه یکتایی را اختصاص میدهد، پس برای مقدار مفروض شماره شعبه، *branchNo*، میتوانیم حداکثر یک رکورد مشخص کنیم، بنابراین *branchNo* کلید کاندید است. بطور مشابه، از آنجاییکه هیچ دو شعبه ای کد پستی مشابهی ندارند، پس zipCode نیز کلید کاندید برای *branch* میشود.

---

۱ Repeating group
۲ Super key
۳ Candidate key
۴ Uniqueness
۵ Irreducibility
۶ Composite key

حال جدول *Role* را در نظر بگیرید، که نقش هایی را که توسط بازیگر در فیلم بازی شده را نشان میدهد. جدول شامل شماره بازیگر (actorNo)، شماره کاتالوگ (catalogNo)، و نام کاراکترهای بازی شده در فیلم (character)، میباشد که در شکل ۳-۲ نشان داده شده است. برای هر شماره بازیگر، چندین فیلم متفاوت وجود دارد که آن بازیگر در آن درخشیده است. به طور مشابه، برای هر شماره کاتالوگ، چندین بازیگر وجود دارد که در این فیلم درخشیده است. از اینرو، اگرچه ترکیب actorNo یا catalogNo حداکثر یک رکورد را مشخص میسازند ولی به خودی خود به عنوان کلید کاندید انتخاب نمیشوند..

شکل ۳-۲

مثالی از جدول Role.

actorNo	catalogNo	character
A1002	207132	James Bond
A3006	330553	Sean Archer
A3006	902355	Jack Stranton
A2019	330553	Castor Troy
A2019	445624	Stanley Goodspeed
A7525	634817	Captain Steve Hiller
A4343	781132	Cruella De Vil

نکته: مراقب باشید که کلید کاندید را با استنتاج از اطلاعات مثال بالا نتیجه گیری نکنید. مگر اینکه نمونه نشان دهنده داده ای باشد که شما در جدول ذخیره خواهید کرد. به طورکلی، نمونه جدول برای اثبات آنکه ستون یا ترکیب ستونها کلید کاندید است نمیتواند استفاده شود. این حقیقت که هیچ تکراری برای مقداری که در لحظه معینی از زمان آشکار میشود متضمن این نیست که تکرار امکان ندارد. با این حال، وجود تکرار در بعضی از موارد نشان دهنده اینست که ترکیب بعضی از ستونها کلید کاندید نیست. شناختن کلید کاندید نیاز به دانستن معنی 'جهان واقعی' ستونهای در برگیرنده است تا درباره امکان وجود تکرار تصمیم بگیریم. تنها با استفاده از این اطلاعات معنایی است که میتوانیم مشخص کنیم که ترکیب ستون کلید کاندید میباشد.

برای مثال، از داده نشان داده شده در شکل ۱-۲، احتمال دارد که فکر کنیم کلید کاندید مناسب برای جدول Staff بایستی name، که نام کارمند است باشد. اما، از آنجاییکه تک مقدار Tom Daniels در جدول وجود دارد، بنابراین اگر عضو جدیدی از پرسنل با نام مشابه به شرکت ملحق شود، انتخاب name بعنوان کلید کاندید را نامعتبر میکند.

وقتی جدول رکورد تکراری نداشته باشد، اکثر اوقات مشخص کردن هر رکورد به طور یکتا امکان پذیر است. این بدین معنی است که جدول همیشه دارای **کلید اصلی** است. در بدترین مورد، مجموعه تمام ستونها بایستی به عنوان کلید اصلی بکار برده شوند، اما معمولا زیرمجموعه کوچکی نیز برای تمییز رکوردها کافی است. کلیدهای کاندیدی که به عنوان کلید اصلی انتخاب نشوند **کلیدهای فرعی** نامیده میشوند. در جدول Branch، اگر branchNo را به عنوان کلید اصلی انتخاب کنیم، zipCode بایستی به عنوان کلید فرعی انتخاب شود. در جدول Role، تنها یک کلید کاندید وجود دارد، که شامل actorNo و catalogNo است، بنابراین این ستونها به طور اتوماتیک کلید اصلی را تشکیل میدهند.

وقتی ستونی در بیش از یک جدول ظاهر میشود، این ستون معمولا بیانگر ارتباط میان رکورد های دو جدول است. برای مثال، در شکل ۱-۲ به کار رفتن branchNo در هر دو جدول Branch و Staff کاملا عمدی بوده و شعبه ها را به جزئیات پرسنلی که آنجا کار میکنند متصل میکند. در جدول Branch، branchNo کلید اصلی است. با این وجود، در جدول Staff ستون branchNo وجود دارد تا پرسنل را به شعبه هایی که در آن کار میکنند نظیر کند. در جدول Staff، branchNo

کلید فرعی است. و این طور بیان می کنیم که ستون branchNo جدول Staff *به* ستون *کلید اصلی* branchNo در جدول خانگی Branch نشانه می رود.

همانطور که در فصل ۱ دیدیم یکی از مزیتهای DBMS کنترل افزونگی داده بود. این یک نمونه مثال از افزونگی کنترل شده بود- همانطور که در بخشهای بعدی خواهیم دید ستونهای مشترک نقش مهمی را در مدل کردن رابطه ها بازی میکنند.

۲-۲-۴ نمایش پایگاه داده رابطه ای

پایگاه داده رابطه ای از یک یا چند جدول تشکیل شده است. قرارداد کلی که برای توصیف پایگاه داده رابطه ای بکار می رود، دادن نام به هر جدول، بدنبال آن نام ستونها در پرانتز است. معمولا، کلید اصلی بصورت زیرخط دار نشان داده می شود. توصیف پایگاه داده رابطه ای برای شرکت اجاره فیلم *StayHome* به صورت زیر است:

Branch	(branchNo, street, city, state, zipCode, mgrStaffNo)
Staff	(staffNo, name, position, salary, branchNo)
Video	(catalogNo, title, category, dailyRental, price, directorNo)
Director	(directorNo, directorName)
Actor	(actorNo, actorName)
Role	(actorNo, catalogNo, character)
Member	(memberNo, fName, lName, address)
Registration	(branchNo, memberNo, staffNo, dateJoined)
RentalAgreement	(rentalNo, dateOut, dateReturn, memberNo, videoNo)
VideoForRent	(videoNo, available, catalogNo, branchNo)

شکل ۴-۲ نمونه ای از پایگاه داده *StayHome* را نشان می دهد.

شکل ۴-۲

نمونه ای از پایگاه داده کرایه فیلم *StayHome*

Branch

branchNo	street	city	state	zipCode	mgrStaffNo
B001	8 Jefferson Way	Portland	OR	97201	S1500
B002	City Center Plaza	Seattle	WA	98122	S0010
B003	14 – 8th Avenue	New York	NY	10012	S0415
B004	16 – 14th Avenue	Seattle	WA	98128	S2250

Staff

staffNo	name	position	salary	branchNo
S1500	Tom Daniels	Manager	46000	B001
S0003	Sally Adams	Assistant	30000	B001
S0010	Mary Martinez	Manager	50000	B002
S3250	Robert Chin	Supervisor	32000	B002
S2250	Sally Stern	Manager	48000	B004
S0415	Art Peters	Manager	41000	B003

شکل ۴-۲

ادامه . . .

Video

catalogNo	title	category	dailyRental	price	directorNo
207132	Tomorrow Never Dies	Action	5.00	21.99	D1001
902355	Primary Colors	Comedy	4.50	14.50	D7834
330553	Face/Off	Thriller	5.00	31.99	D4576
781132	101 Dalmatians	Children	4.00	18.50	D0078
445624	The Rock	Action	4.00	29.99	D5743
634817	Independence Day	Sci-Fi	4.50	32.99	D3765

Director

directorNo	directorName
D1001	Roger Spottiswoode
D7834	Mike Nichols
D4576	John Woo
D0078	Stephen Herek
D5743	Michael Bay
D3765	Roland Emmerick

Actor

actorNo	actorName
A1002	Pierce Brosnan
A3006	John Travolta
A2019	Nicolas Cage
A7525	Will Smith
A4343	Glenn Close

Role

actorNo	catalogNo	character
A1002	207132	James Bond
A3006	330553	Sean Archer
A3006	902355	Jack Stranton
A2019	330553	Castor Troy
A2019	445624	Stanley Goodspeed
A7525	634817	Captain Steve Hiller
A4343	781132	Cruella De Vil

Member

memberNo	fName	lName	address
M250178	Bob	Adams	57 – 11th Avenue, Seattle, WA, 98105
M166884	Art	Peters	89 Redmond Rd, Portland, OR, 97117
M115656	Serena	Parker	22 W. Capital Way, Portland, OR, 97201
M284354	Don	Nelson	123 Suffolk Lane, Seattle, WA, 98117

شکل ۴-۲

(ادامه)

Registration

branchNo	memberNo	staffNo	dateJoined
B002	M250178	S3250	1-Jul-98
B001	M166884	S0003	4-Sep-99
B001	M115656	S0003	12-May-97
B002	M284354	S3250	9-Oct-98

RentalAgreement

rentalNo	dateOut	dateReturn	memberNo	videoNo
R753461	4-Feb-00	6-Feb-00	M284354	245456
R753462	4-Feb-00	6-Feb-00	M284354	243431
R668256	5-Feb-00	7-Feb-00	M115656	199004
R668189	2-Feb-00	4-Feb-00	M115656	178643

VideoForRent

videoNo	available	catalogNo	branchNo
199004	N	207132	B001
245456	Y	207132	B002
178643	N	634817	B001
243431	N	634817	B002

۲-۳ جامعیت رابطه ای

در بخش قبلی، درباره بخش ساختاری مدل داده رابطه ای بحث کردیم. همانگونه که از بخش ۱-۲ بیاد داریم، مدل داده دو بخش دارد: قسمت دستکاری، که عملیات مجاز بر روی داده ها را تعریف میکند، و مجموعه قواعد جامعیت، که از دقیق بودن داده ها اطمینان حاصل میکند. در این بخش، درباره قواعد جامعیت رابطه ای بحث کرده، و در بخش بعدی درباره زبانهای اصلی دستکاری روی داده ها مطالبی می آموزیم.

از آنجائیکه هر ستون دامنه مربوط به خود را دارا می باشد، بنابراین محدودیتهایی (که *محدودیتهای دامنه* نام دارد) بصورت محدودیت بر روی مجموعه مقادیر مجاز برای ستونهای جداول وجود خواهد داشت. بعلاوه، دو قاعده جامعیت رابطه ای مهم وجود دارد، که محدودیتهایی هستند که به تمام نمونه های پایگاه داده اعمال میشوند. این دو قاعده عمده روی مدل رابطه ای به نام *جامعیت موجودیت*[۱] و *جامعیت ارجاع*[۲] خوانده میشوند. قبل از اینکه به تعریف این عبارات بپردازیم، اول از همه باید با مفهوم Null (تهی) آشنا شویم.

۲-۳-۱ Nulls

تهی میتواند به معنی' نامشخص' بکار رود. همچنین میتواند به معنی داده ای بکار رود که قابل اجرا بر روی رکورد مشخصی نباشد، یا می تواند به این معنی باشد که فعلا هیچ داده ای به رکورد فوق تخصیص داده نشده است. تهی ها راهی برای سرو کار داشتن با داده های غیرکامل یا استثنایی هستند. البته قابل توجه است که، تهی برابر مقادیر داده ای صفر یا مقادیر رشته ای که با خط فاصله پر شده نیست. صفرها و خط فاصله ها خودشان مقدار هستند، اما تهی بیانگر عدم وجود مقدار است. بنابراین، با تهی بایستی بعنوان یک داده متفاوت برخورد شود.

---

۱ Entity integrity
۲ Referential integrity

برای مثال، موقتا فرض کنید شعبه بدون مدیر امکان داشته باشد، شاید بدین علت که مدیرکنونی شعبه را ترک کرده و هنوز مدیر دیگری به جای وی منصوب نشده باشد. در این مورد، مقدار ستون متناظر mgrStaffNo نامشخص خواهد بود. بدون وجود تهی، شاید لازم باشد که داده اشتباهی را در ستون فوق جهت نشان دادن این موقعیت وارد کنیم یا ستون دیگری را که برای کاربر قابل فهم نیست اضافه کنیم. در این مورد، سعی میکنیم تا غیبت مدیر را با مقدار، 'هنوز هیچ[1]' که نشان دهنده اینست که هنوز هیچ مدیری انتصاب نشده است، نشان دهیم. متناوبا، ستون جدیدی بنام' currentManager? ' به جدول Branch اضافه می کنیم، که شامل مقدار Y(Yes)، جهت وجود مدیر، وN(No) برای عدم وجود مدیر، باشد. هر دو این روشها می توانند برای کسانی که از پایگاه داده استفاده می کنند گیج کننده باشد.

حال ما در موقعیتی هستیم که دو قاعده جامعیت رابطه ای را تعریف کنیم.

۲-۳-۲ جامعیت موجودیت

اولین قاعده جامعیت به کلیدهای اصلی جدول پایه اعمال میشود.

> **جدول پایه** یک جدول دارای نامی است که رکوردهای آن به طور فیزیکی در پایگاه داده ذخیره شده است. که این در مغایرت با دید میباشد، که از فصل ۳-۲-۱ بیاد داریم. بطوریکه یک دید 'جدول مجازی' است که در پایگاه داده به طور فیزیکی وجود نداشته اما توسط DBMS به خاطر پوشش جداول پایه تولید شده است.

بنا به تعریف قبلی، ما می دانیم که کلید اصلی شناسه کمینه ای است که جهت شناسایی رکوردها به طور یکتا استفاده می شود. بدین معنا که هیچ زیر مجموعه ای از کلید اصلی برای شناسایی رکودها به طور یکتا کافی نیست. اگر ما مجاز به استفاده از تهی در هر قسمتی از کلید اصلی باشیم، بدین معنی است که هیچکدام از ستونها جهت تمییز بین رکوردها نیاز نمی باشد، که با تعریف کلید اصلی در تناقض است.

برای مثال، branchNo کلید اصلی جدول Branch است، که ما نباید قادر به وارد کردن مقدار دارای تهی برای ستون branchNo باشیم.

۲-۳-۳ جامعیت ارجاع

دومین قاعده جامعیت به کلیدهای فرعی اعمال می شود.

در شکل ۱-۲، branchNo در جدول Staff کلید فرعی است که به سوی ستون branchNo در جدول خانگی Branch اشاره می کند. نباید امکان ایجاد رکورد Staff، برای مثال، با شماره شعبه B300 وجود داشته باشد، مگر اینکه در جدول Branch رکوردی با شماره شعبه B300 وجود داشته باشد. با این وجود، باید بتوانیم رکورد پرسنلی جدیدی را با شماره شعبه Null برای مواقعی که در آنها عضو جدیدی از پرسنل به شرکت ملحق شده و هنوز به شعبه مشخصی تخصیص داده نشده است، ایجاد کنیم.

---

[1] None at present

۲-۳-۴ قواعد تجاری (**business rules**)

برای کاربران همچنین این امکان وجود دارد که محدودیتهای دیگری را نیز مشخص کنند تا نیازهای داده ای شان برآورده شود. برای مثال، اگر *StayHome* قاعده ای دارد که در آن هر عضو می تواند تنها حداکثر ۱۰ فیلم در هر بار اجاره کند، پس کاربر بایستی بتواند این قاعده را مشخص کرده و از DBMS انتظار دارد که آن را اجرا کند. در این مورد، امکان اجاره بیش از ۱۰ فیلم برای عضو نباید وجود داشته باشد. متاسفانه، سطح پشتیبانی برای جامعیت رابطه ای از سیستم به سیستم متفاوت است. پیاده سازی جامعیت رابطه ای را در فصل های ۱۲ و ۱۹ بحث می کنیم.

۲-۴ زبانهای رابطه ای

در بخش ۱-۲، خواندیم که یکی از بخشهای مدل داده ای قسمت دستکاری آن است، که انواع عملیات مجاز بر روی داده ها را تعریف می کند. این عملیات شامل عملیاتی جهت بهنگام کردن، یا بازیابی داده از پایگاه داده، و نیز جهت تغییر ساختار پایگاه داده است. دو زبان اصلی که برای DBMS های رابطه ای بوجود آمده است عبارتند از:

- **SQL** (Structured Query Language)
- **QBE** (Query-by-example)

SQL توسط سازمان استاندارد بین المللی (ISO) استاندارد شده است، و این زبان را بعنوان یک زبان رسمی جهت تعریف و دستکاری پایگاه داده رابطه ای کرده است. خصوصیات عمده SQL بدین صورت است :

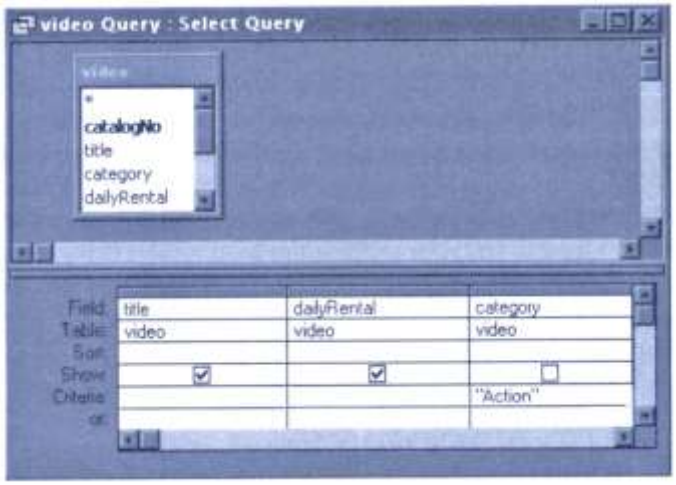
- یادگیری آن نسبتا آسان است .
- زبان غیر رویه ای است، بدین صورت که مشخص می کنید که **چه** اطلاعاتی نیاز دارید، به جای اینکه **چگونه** آنرا بدست آورید.
- شبیه دیگر زبانهای مدرن، SQL ذاتا به صورت قالب آزاد است، بدین معنی که قسمتهای دستورات لازم نیست که در محل مشخصی تایپ شوند.
- ساختار دستورات به زبان محاوره ای نزدیک است، همچون CREATE TABLE، INSERT،SELECT. بعنوان مثال با استفاده از جداول تعریف شده در شکل ۲-۴ میتوانیم لیست عناوین و نرخ اجاره روزانه را برای تمامی فیلم ها در طبقه فیلم های حادثه ای، که بر اساس عنوان فیلم مرتب شده است را، با استفاده از دستورات SQL به صورت زیر بنویسیم :

```
SELECT title, dailyrental
FROM video
WHERE category = 'Action'
ORDER BY title ASC;
```

- SQL می تواند در بین محدوده ای از کاربران اعم از مدیران پایگاه داده (DBA) ، مدیریت کارکنان، برنامه نویسان کاربردی، و همچنین بین کاربران نهایی مورد استفاده قرار گیرد.

با این وجود، SQL تنها شامل فرمانهایی برای تعریف و دستکاری پایگاه داده است. و در حال حاضر دستورات شرطی همچون IF…THEN…ELSE ، GO TO، DO…WHILE دراین زبان وجود ندارد. SQL در دو مورد استفاده می شود. اولین راه

استفاده از آن به صورت پرس و جو با وارد کردن جملات به صورت پایانه است. راه دوم این است که دستورات SQL را در بین دیگر دستورات زبانهای رویه ای همچون ویژوال بیسک، دلفی، زبان سی، جاوا ،کوبول، و فرترن و پاسکال یا ادا جاسازی کنیم.
QBE زبان دیگری است که گرافیکی بوده و با روش ' اشاره و کلیک' جهت پرس و جو از پایگاه داده بکار می رود و برای پایگاه داده هایی مناسب است که پیچیده نبوده و در چندین جدول بیان شوند. QBE برای کاربران غیر فنی مناسب است که بخواهند اطلاعات مناسبی را از پایگاه داده استخراج کنند. صفحه نمایش برای پرس و جو ها به جای تایپ، در نامهای ستونها و قالب استفاده میشود. با این وجود، بایستی ستونهایی را که میخواهیم ببینیم و همچنین مقدار داده ای که برای محدود کردن پرس و جو میخواهیم را مشخص کنیم. زبانهایی همچون QBE به شدت میتوانند جهت بهنگام رسانی و پرس و جو پر بازده باشند. شکل ۵-۲ نشان میدهد که چگونه پرس و جوی SQL بالایی را توسط QBE اکسس مایکروسافت ایجاد کرده ایم.
متاسفانه، برخلاف SQL، استاندارد رسمی برای این زبان وجود ندارد. با این حال، کارایی ای که توسط فروشندگان مختلف ارائه شده عموما مشابه بوده و زبان شهودی تر از SQL میباشد. بحث SQL و QBE را در فصل ۱۷ پی میگیریم.



شکل ۵-۲

پرس وجوی نمونه QBE.

خلاصه فصل

- ✓ سیستم مدیریت پایگاه داده رابطه ای (RDBMS) ، با فروش تخمینی ۱۰- ۸ میلیون دلار در سال (۲۵ میلیون دلار با فروش ابزار جانبی)، و رشدی در حدود ۲۵ درصد در سال امروزه یکی از عمده ترین نرم افزارهای پردازش داده میباشد. این نرم افزار دومین تولید DBMS به شمار میرود و بر اساس مدل داده ای رابطه ای است که توسط دکتر کاد بیان شده است.
- ✓ رابطه ها به طور فیزیکی به صورت **جداولی** با رکوردهای متناظر با تاپلهای منفرد و ستونها به عنوان صفات خاصه نشان داده میشوند.
- ✓ خصوصیات جدولهای رابطه ای چنین است: هر خانه شامل دقیقا یک مقدار است، نامهای ستون مجزا است، مقادیر ستون از دامنه مشابه میباشد، ترتیب ستون و رکورد بی اهمیت است و هیچ رکورد تکراری وجود ندارد.
- ✓ **ابرکلید**، ستون یا مجموعه ای از ستونهاست که رکوردهای جدول را به طور منحصر مشخص میکند، در حالیکه **کلید کاندید** ابرکلید کمینه شده است. **کلید اصلی** کلید کاندید انتخابی برای استفاده در شناسایی رکوردهاست. جدول

بایستی همیشه کلید اصلی داشته باشد. **کلید خارجی** ستون یا مجموعه ای از ستونهاست با یک جدول که کلید کاندید دیگری است.

- ✓ تهی بیانگر مقدار برای ستونی است که در حال حاضر نامشخص بوده یا برای این رکورد تعریف نشده است.
- ✓ **جامعیت موجودیت** محدودیتی است که در جدول اصلی واقع شده بدین صورت که هیچ ستونی از کلید اصلی نمیتواند تهی باشد. **جامعیت ارجاع** بیان میکند که مقادیر کلید فرعی بایستی با مقدار کلید کاندید با بعضی رکوردها در جدول مبنا تطبیق داشته یا تماما تهی باشند.
- ✓ دو زبان اصلی جهت دستیابی به پایگاه داده رابطه ای SQL (Structured Query Languages) و QBE(Query-by-Example) میباشند.

فصل ۳

چرخه حیات پایگاه داده

آنچه در این فصل خواهید آموخت :

- مراحل اصلی چرخه حیات سیستم های اطلاعاتی (IS).
- ارتباط میان برنامه پایگاه داده و چرخه حیات سیتمهای اطلاعاتی.
- مراحل اصلی چرخه حیات برنامه پایگاه داده.
- فعالیتهای مرتبط با هر مرحله از چرخه حیات.

این فصل در ابتدا، با توضیح اینکه چرا به رهیافت ساخت یافته ای جهت توسعه برنامه های کاربردی نیاز است شروع میشود. در این فصل مثالی از چنین رهیافتی را که چرخه حیات سیستم های اطلاعاتی[۱] نامیده میشود، همچنین ارتباط بین سیستم اطلاعاتی و پایگاه داده پشتیبانی کننده آنرا بررسی کرده و درباره آنها بحث میکنیم. سپس روی پایگاه داده تمرکز کرده و نمونه ای از رهیافت ساخت یافته جهت توسعه برنامه های پایگاه داده که چرخه حیات برنامه پایگاه داده نامیده میشود را معرفی خواهیم کرد. بالاخره، مراحلی را جهت ساختن چرخه حیات پایگاه داده ارائه خواهیم کرد.

۳-۱ بحران نرم افزار

احتمالا از چندین دهه قبل تا به امروز از توسعه چشمگیر نرم افزارهای کاربردی آگاه هستید، برنامه هایی که تنها شامل چندین سطر کد بوده که به برنامه هایی تا میلیونها کد توسعه یافته اند. پس از این توسعه چشمگیر، نگهداری تعدادی از این برنامه ها امری طاقت فرسا شده، و نیاز به نگهداری پایدار پیدا کردند. که این نگهداری شامل اصلاح عیبها، پیاده سازی نیازهای جدید کاربر، و اصلاح برنامه هایی است که روی ساختارهای جدید و بهبود یافته اجرا شوند.

جهت پشتیبانی از این تعداد نرم افزار، تلاش های انجام گرفته جهت پشتیبانی به منابعی بیشتری نیاز پیدا کرد. در نتیجه، تعدادی از پروژه های بزرگ نرم افزاری تاخیر کردند، و از بودجه موردنظر فراتر رفتند، همچنین نرم افزارهای تولیدی غیرقابل اعتماد شده، و نگهداری آن سخت گردیده، و کارکرد آنها کاهش یافت. که این منجر به آنچه که امروزه آنرا 'بحران نرم افزار[۲] ' می نامیم،گشته است. اگرچه این عبارت اولین بار در اواخر سال ۱۹۶۰ به کار رفت، اما بیش از سی سال است که این بحران هنوز با ماست. در نتیجه، بعضی از افراد بحران نرم افزار را ' انحطاط نرم افزار' می نامند. با مشاهده بحران نرم افزار، مطالعاتی در انگلستان توسط OASIG انجام گرفت، و نتایج زیر بدست آمد (OASIG,1996) :

---

[۱] Information system lifecycle

[۲] Software crises

- ۹۰-۸۰ درصد سیستمها به اهداف کارایی شان نمی رسند.
- حدود ۸۰ درصد دیر تحویل داده میشوند و یا هزینه ای بیش از بودجه موردنظر صرف میکنند.
- تقریبا ۴۰ درصد پیشرفتها شکست خورده یا رها گشته اند.
- کمتر از ۴۰ درصد کاملا با نیازهای مهارتی و آموزشی مطابقت دارند.
- کمتر از ۲۵ درصد احتمالا اهداف تکنولوژی و تجاری را یکی کرده اند.
- فقط ۱۰ تا ۲۰ درصد تمامی ضوابط موفقیت را رعایت کرده اند.

دلایل عمده شکست پروژه های نرم افزاری عبارتند از:

- کمبود خصوصیات نیازمندی های تکمیل شده;
- کمبود متدلوژی توسعه مناسب;
- تجزیه ضعیف طراحی به اجزا قابل مدیریت.

بعنوان راه حلی برای این مشکلات، رهیافت ساخت یافته ای را که عموما بنام *چرخه حیات سیتمهای اطلاعاتی* یا *چرخه حیات نرم افزار*[1] (*SDLC*) نامیده میشود جهت توسعه نرم افزار پیشنهاد می کنیم.

۳-۲ چرخه حیات سیستمهای اطلاعاتی

سیستم اطلاعاتی[2] نه تنها به مدیریت، جمع آوری، و کنترل داده های تولید و استفاده شده توسط شرکت می پردازد بلکه داده ها را نیز به اطلاعات تبدیل می کند. سیستم اطلاعاتی همچنین زیربنایی را جهت کمک به پخش اطلاعات به افرادی می کند که تصمیمات بحرانی را برای موفقیت شرکت می گیرند. جزء ضروری در مرکز سیستم اطلاعاتی، پایگاه داده ای است که آنرا پشتیبانی می کند.

معمولا، مراحل چرخه حیات سیستمهای اطلاعاتی شامل: برنامه ریزی، تحلیل و جمع آوری نیازمندیها، طراحی (شامل طراحی پایگاه داده)، نمونه سازی، پیاده سازی، تست، تبدیل، و نگهداری عملکردی می باشد. البته، در این کتاب به جنبه توسعه جزء پایگاه داده سیستم اطلاعاتی می پردازیم. از آنجائیکه پایگاه داده یکی از اجزاء اساسی سیستم اطلاعاتی بزرگتر در سطح شرکت به شمار می آید، چرخه حیات برنامه پایگاه داده در اصل به چرخه حیات سیستمهای اطلاعاتی پیوند خورده است.

۳-۳ چرخه حیات برنامه پایگاه داده

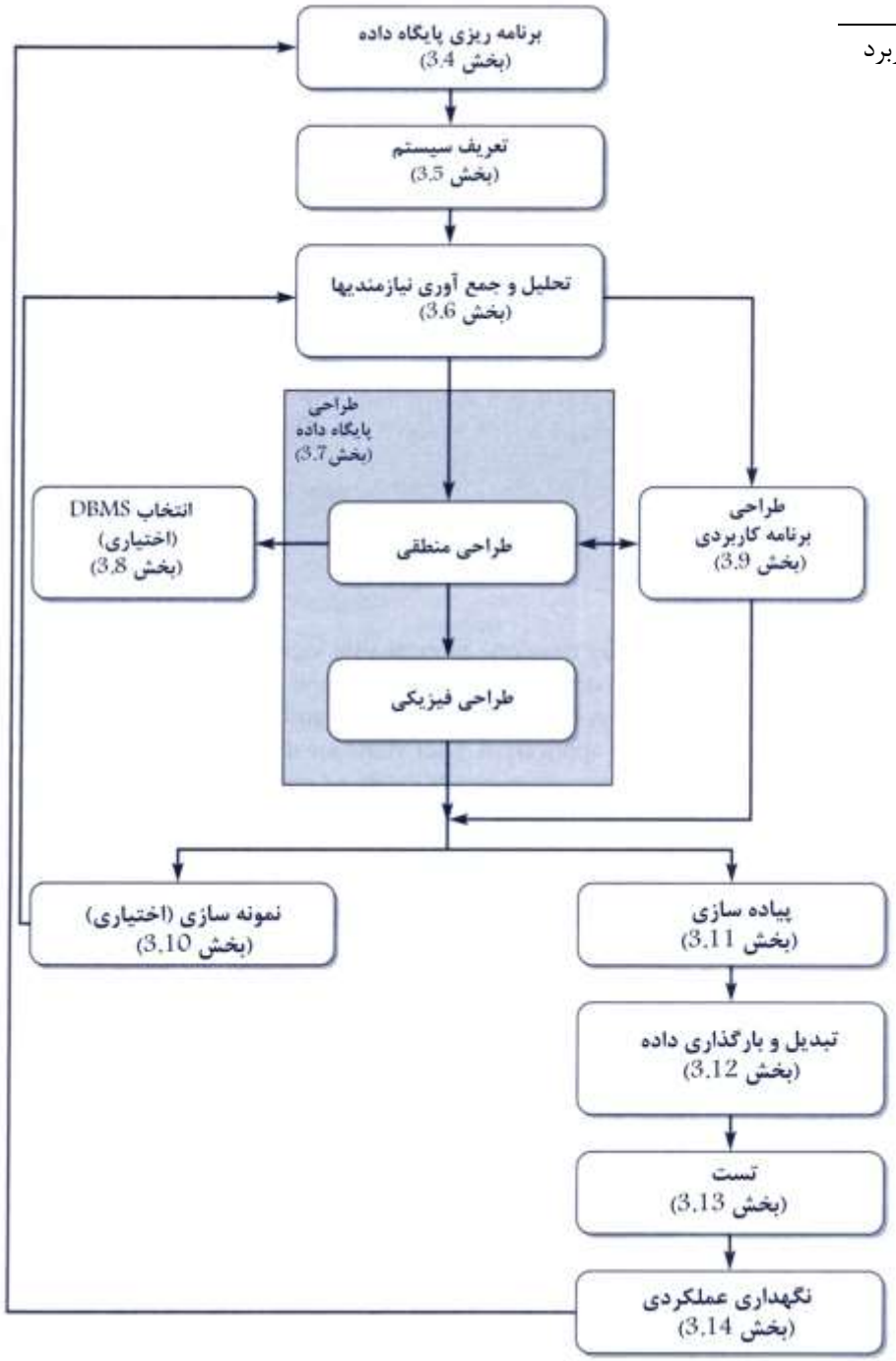
در این بخش، چرخه حیات برنامه پایگاه داده DBMS های رابطه ای را توصیف می کنیم. خلاصه مراحل چرخه حیات برنامه پایگاه داده در شکل ۱-۳ نشان داده شده است. زیر نام هر مرحله شماره بخشی از این کتاب که آن مرحله در آن شرح داده شده آمده است. به این نکته توجه کنید که مراحل چرخه حیات برنامه پایگاه داده به صورت ترتیبی نبوده بلکه بصورت تکرار مراحل قبلی از طریق حلقه فیدبک (بازخورد) حاصل میشود. برای مثال، مشکلات ایجاد شده به هنگام طراحی پایگاه داده احتمال دارد به مجموعه تحلیل ها و نیازهای دیگری احتیاج داشته باشد. از آنجاییکه حلقه های بازخوردی زیادی وجود دارد، تنها آنهایی را که بیش از همه واضح هست را در شکل ۱-۳ نشان می دهیم.

---

[1] Software Development Lifecycle
[2] Information System

شکل ۱-۳

مراحل اصلی چرخه حیات کاربرد پایگاه داده .



برای برنامه های پایگاه داده کوچکی که از تعداد کاربران کمی پشتیبانی می کند، چرخه حیات موردنیاز خیلی پیچیده نیست. با این وجود، به هنگام طراحی برنامه های پایگاه داده متوسط به بزرگ با ده ها تا هزاران کاربر، و استفاده از صدها پرس و جو و برنامه های کاربردی، چرخه حیات می تواند بشدت پیچیده باشد.

۳-۴ برنامه ریزی پایگاه داده

نقطه آغازین ساخت پروژه پایگاه داده ایجاد *طرح اجمالی ماموریت*[۱] *و اهداف ماموریت*[۲] برنامه کاربردی پایگاه داده می باشد. *طرح اجمالی ماموریت* اهداف اصلی برنامه پایگاه داده را بیان می کند، درحالیکه اهداف ماموریت آن کار خاصی را که پایگاه داده باید پشتیبانی کند را تعریف می کند. البته، مانند هر پروژه دیگری، بخشی از فرایند برنامه ریزی پایگاه داده باید شامل تخمینی از کارهایی که قرار است انجام گیرد، منابعی که مورد استفاده قرار گیرد، و هزینه ای که برای تمام آنها باید پرداخت شود باشد. همانطور که ملاحظه کردیم، پایگاه داده اغلب بخش بزرگی از سیستمهای اطلاعاتی موجود در سطح شرکت را تشکیل می دهد و بنابراین هر پروژه پایگاه داده ای بایستی با استراتژی سیستمهای اطلاعاتی شرکت ترکیب شود.

۳-۵ تعریف سیستم

قبل از اینکه به طراحی پایگاه داده بپردازیم، ابتدا اول از همه باید محدوده سیستم تحت بررسی، و همچنین ارتباط آن با دیگر قسمتهای سیستم اطلاعاتی شرکت را مشخص کنیم. شکل ۲-۳ محدوده سیستم برنامه پایگاه داده شرکت اجاره فیلم ویدیویی ***StayHome*** را نشان می دهد. وقتی محدوده سیستم برنامه پایگاه داده را نمایش می دهیم نه تنها دید کاربر کنونی را مشخص می کنیم بلکه تمامی دیدهای کاربران آتی را نیز معین می کنیم.

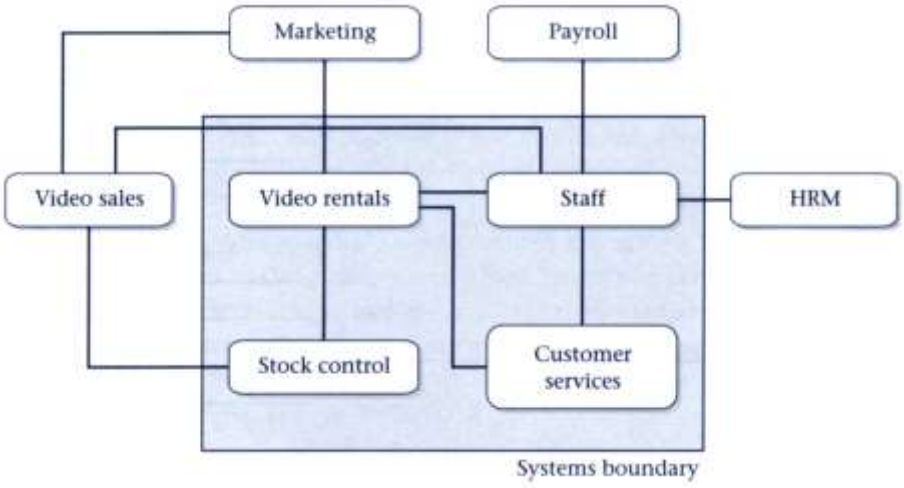
توجه کنید که این نوع نمودارها میتواند در هر سطح جزئیاتی رسم شود. دومین مثال از این نوع نمودار (در سطح پایین تر) در شکل ۹-۴ فصل بعدی نشان داده شده است.

---

[۱] Mission statement
[۲] Mission objective

شکل ۲-۳

محدوده سیستمی پایگاه داده شرکت اجاره فیلم *StayHome* .



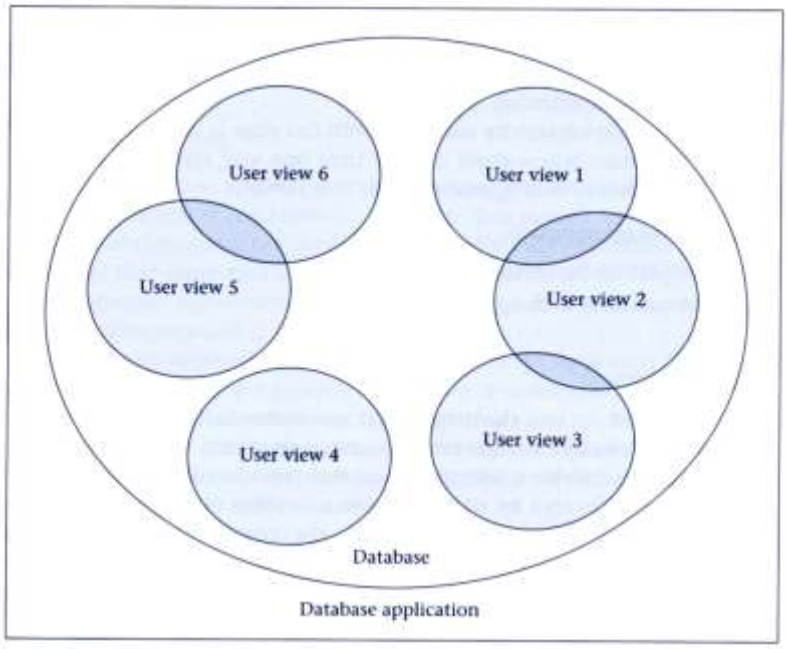
۳-۵-۱ دیدهای کاربر

برنامه پایگاه داده شاید یک یا چند دید کاربر داشته باشد. شناسایی صحیح دیدهای کاربر از مهمترین جنبه های توسعه برنامه پایگاه داده است زیرا به ما کمک می کند هنگام توسعه نیازهای جدید، از عدم فراموشی کاربران اصلی پایگاه داده اطمینان حاصل کنیم. دیدهای کاربر خصوصا هنگام توسعه برنامه پایگاه داده های نسبتا پیچیده مفید می باشند، زیرا باعث می شوند که نیازمندیها به قسمتهای قابل مدیریت شکسته شوند.

دید کاربر آنچه که برنامه پایگاه داده نیاز دارد را تعریف می کند که در رابطه با داده ای که باید بگیرد و تراکنشی که باید روی داده انجام بگیرد می باشد، (به عبارت دیگر، آنچه که کاربران می خواهند با داده انجام دهند). نیازمندی های دید کاربر ممکن است مجزا بوده و یا با دیگر دیدها همپوشانی داشته باشد. شکل ۳-۳ نمایش هندسی برنامه پایگاه داده با چندین دید کاربر موجود را نشان می دهد (که با دید کاربر ۱ تا ۶ مشخص شده است). توجه کنید درحالیکه دیدهای کاربر (۱ و۲ و۳) و (۵ و۶) در نیازمندیها همپوشانی دارند (که بصورت نواحی پررنگ مشخص هستند)، دید کاربر ۴ دارای نیازمندیهای مجزا است.

شکل ۳-۳

نمودار پایگاه داده ای با چندین دید کاربر را نشان می دهد : دید۴ مجزا است; دیدهای دیگر دارای چند همپوشانی هستند.



۳-۶ تحلیل و جمع آوری نیازمندیها

در این مرحله، به تحلیل و جمع آوری نیازمندیهای شرکت، یا قسمتی از شرکت، که توسط پایگاه داده به کار گرفته می شوند می پردازیم. چندین تکنیک از جمله تکنیک حقیقت یابی[۱] جهت جمع آوری این اطلاعات، وجود دارد که آن را با جزئیات در فصل ۴ بررسی خواهیم کرد.

ما همچنین اطلاعاتی را درباره دیدهای اصلی کاربر (به عبارت دیگر، نقش شغل[۲] یا محدوده کاربرد تجاری[۳]) جمع آوری می کنیم که شامل موارد زیر می باشند :

- توصیف داده های استفاده شده یا تولید شده،
- جزئیات اینکه داده ها چگونه تولید یا استفاده می شوند،
- هر گونه نیازمندیهای اضافی دیگری برای برنامه پایگاه داده جدید .

سپس این اطلاعات را جهت شناسایی نیازمندی هایی که در برنامه پایگاه داده جدید بایستی لحاظ شوند تحلیل می کنیم. این نیازمندیها در مستندهای مجتمعی به نام *خصوصیات نیازمندیها*[۴] برای برنامه پایگاه داده جدید مستند و جمع آوری می شوند. فعالیت مهم دیگر مربوط به این مرحله این است که چگونه با موقعیتهایی که در آنها بیش از یک دید کاربر وجود دارد برخورد کنیم. سه روش اصلی در مواجه با چندین دید کاربر وجود دارد:

- روش متمرکز سازی،
- روش یکپارچگی دید، و
- ترکیبی از دو روش فوق.

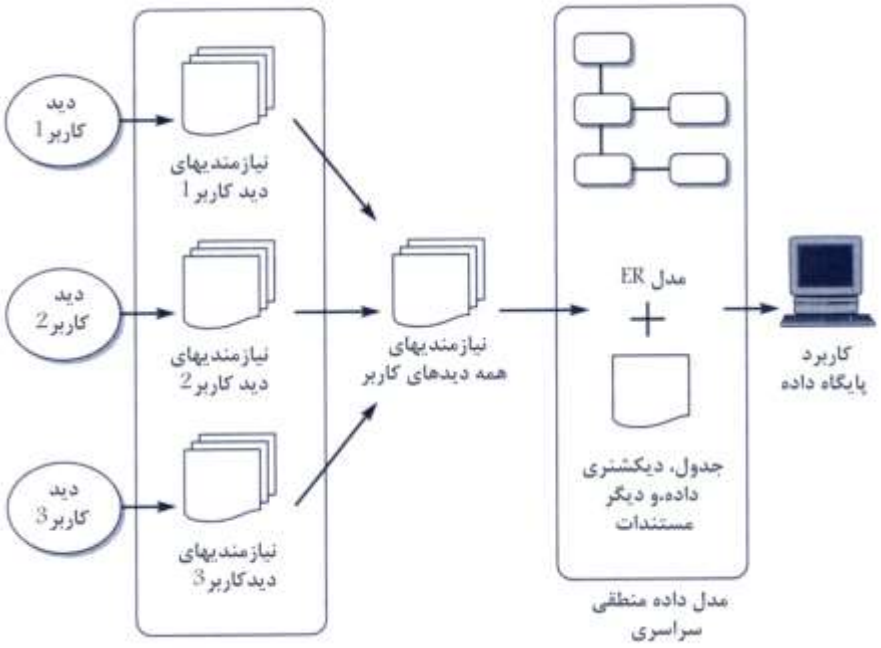
---

۱ Fact-finding
۲ Job role
۳ Business application area
٤ Requirement specification

روش متمرکزسازی (یکباره) در برگیرنده نیازمندیهای دیدهای مختلف در لیست واحدی از نیازها است، که ما به طور خلاصه به آن **دید** می گوییم. **مدل داده ای منطقی سراسری،** کل دیدی را که (به عبارت دیگر، شرکت، یا بخشی از شرکتی را که در حال مدل کردن آن هستیم) در مرحله بعد ایجاد شده است را نشان می دهد. شکل ۴-۳ مدیریت دیدهای کاربر ۱ تا ۳ را با استفاده از روش متمرکزسازی نشان می دهد. به طور کلی، استفاده از این روش زمانی است که همپوشانی عمده بین نیازمندیها و برنامه پایگاه داده بیش از اندازه پیچیده نباشد.

شکل ۴-۳

روش متمرکزسازی جهت مدیریت دیدهای کاربر ۱ تا ۳ .



روش یکپارچگی دید

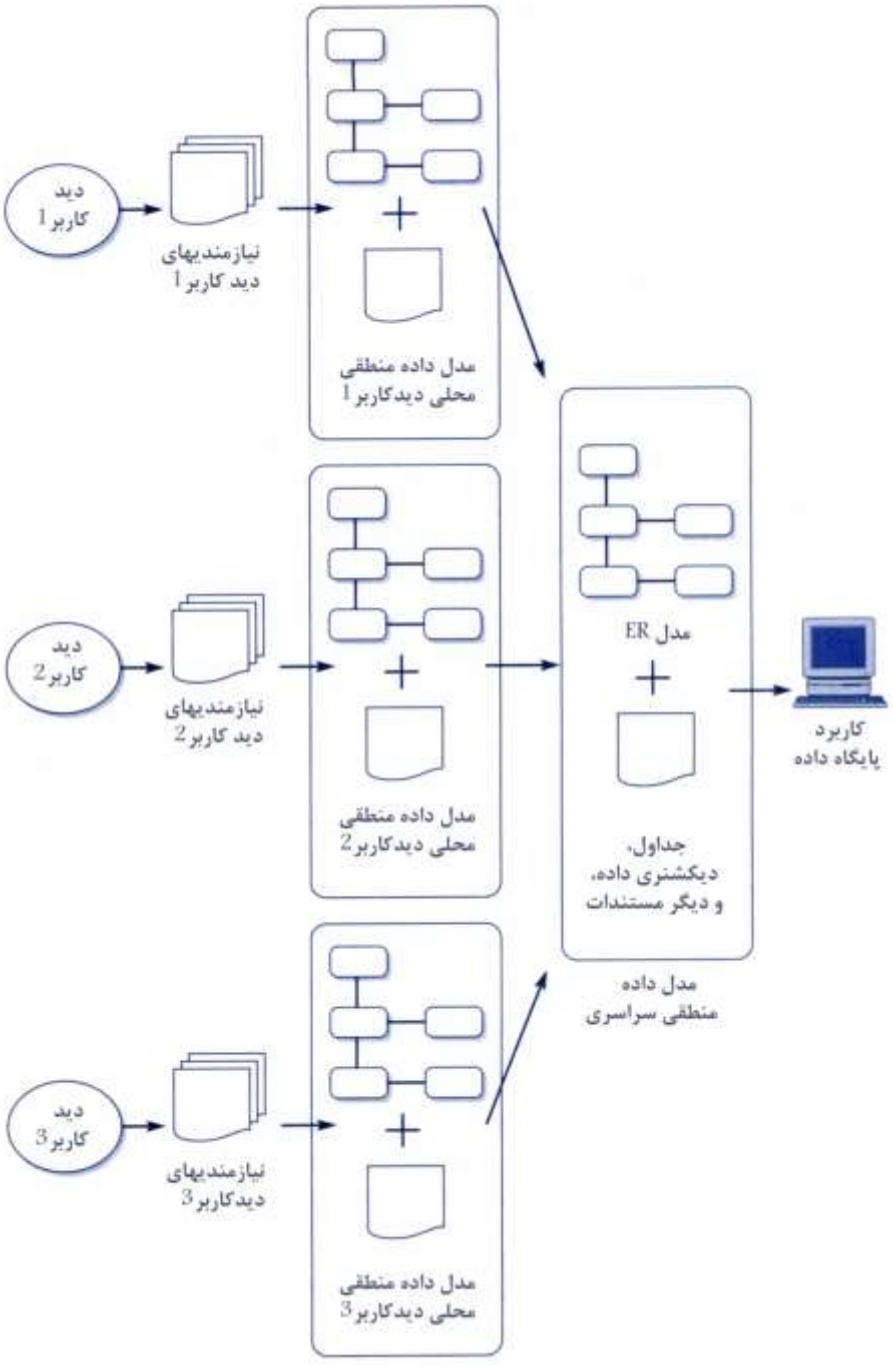
رهیافت یکپارچگی دید[1] شامل بیان نیازمندیهای هر دید به صورت لیست جدایی از نیازمندی هاست. در این روش مدلهای داده ای را که هر دید کاربر را نشان می دهد ایجاد کرده و سپس این مدلها را در مرحله بعدی طراحی پایگاه داده ترکیب می کنیم. مدل داده ایی که دید کاربر واحدی را نشان می دهد بنام **مدل داده ای منطقی محلی** نامیده می شود. نموداری که مدیریت دیدهای کاربر ۱ تا ۳ را با روش یکپارچگی دید نشان می دهد، در شکل ۵-۳ نمایش داده شده است.

عموما، این روش زمانیکه تفاوتهای عمده ای بین دید کاربر و برنامه پایگاه داده وجود داشته باشد و تقسیم کار به قسمتهای قابل مدیریت را توجیه کند، به کار برده می شود.

در بخش ۴-۴-۴ درباره چگونگی مدیریت دیدهای چندین کاربر با استفاده از شرکت اجاره فیلم ***StayHome*** با جزئیات بحث کرده و در فصل ۱۰ نشان می دهیم که چگونه از روش یکپارچگی دید استفاده کنیم.

[1] View integration approach

نکته: تحلیل و جمع آوری نیازمندیها مرحله مقدماتی طراحی پایگاه داده است. مقدار داده جمع آوری شده بستگی به ماهیت مساله و سیاستهای شرکت دارد. شناسایی کارکرد مورد نیاز برای برنامه پایگاه داده یک فعالیت بحرانی می باشد، آن هم به این علت که سیستم هایی که کارکرد ناکافی یا ناقص دارند کاربران را اذیت کرده، و منجر به عدم پذیرش توسط آنان خواهند شد. با این وجود، کارکرد بیش از اندازه هم مشکل ساز خواهد بود زیرا سیستم را بیش از حد پیچیده کرده، و سیستم را در مراحل پیاده سازی، نگهداری، استفاده، و یادگیری دچار مشکل خواهد کرد.



شکل ۵-۳

روش جامعیت دید جهت مدیریت دیدهای کاربر ۱ تا ۳ .

۳-۷ طراحی پایگاه داده

در فصل۷، درباره اهداف اصلی مرحله طراحی پایگاه داده و همچنین درباره خلاصه ای از متدلوژی طراحی پایگاه داده بحث خواهیم کرد. **طراحی پایگاه داده** از دو فاز اصلی تشکیل شده است که طراحی منطقی و فیزیکی نامیده می شود. هنگام طراحی منطقی پایگاه داده، سعی می کنیم تا اشیاء مهم مورد نیاز در پایگاه داده و همچنین ارتباط بین این اشیاء را نمایش دهیم. همچنین هنگام طراحی فیزیکی پایگاه داده، در این باره که چگونه بایستی طراحی منطقی را به طور فیزیکی (به صورت جدول) بر روی DBMS هدف پیاده سازی کرد تصمیم می گیریم. متدلوژی گام به گام طراحی منطقی پایگاه داده را در فصل های ۸ تا ۱۱ و طراحی فیزیکی را در فصل های ۱۲ تا ۱۶ شرح خواهیم داد.

۳-۸ انتخاب DBMS

اگر در حال حاضر هیچگونه DBMS رابطه ای در شرکت وجود نداشته باشد، بخش اصلی چرخه حیات انتخاب میان مراحل طراحی پایگاه داده فیزیکی و منطقی می باشد. بنابراین، انتخاب DBMS مناسب می تواند هر زمانی قبل از طراحی منطقی انجام بگیرد. اگرچه ممکن است انتخاب DBMS نادر باشد، ولی چنانکه حرفه نیاز به گسترش داشته باشد و یا سیستمهای موجود جایگزین شوند، احتمالا نیازمند بررسی محصولات DBMS جدید خواهیم شد. در چنین مواردی، هدف تعیین سیستمی است که نیازمندی های کنونی و آینده شرکت را برآورده کرده، و به علت خرید DBMS جدید تعادلی را بین هزینه ها، خرید هرگونه سخت افزار یا نرم افزار موردنیاز برای پشتیبانی از سیستم پایگاه داده، و هزینه های مربوط به تحول و آموزش پرسنل برقرار سازد.

روش ساده جهت انتخاب DBMS، بررسی DBMS جهت برآورد نیازمندی ها می باشد. هنگام انتخاب محصول DBMS جدید، فرصتی است تا مطمئن شویم که انتخاب ما بخوبی برنا مه ریزی شده باشد و از رسیدن سود به شرکت مطمئن شویم.

۳-۹ طراحی برنامه کاربردی

همانطور که قبلا در شکل ۱-۳ نشان داده شد، دیدیم که طراحی پایگاه داده و طراحی برنامه کاربردی از فعالیت های موازی چرخه حیات برنامه کاربردی پایگاه داده به شمار می روند. اغلب اوقات، نمی توانیم طراحی برنامه کاربردی را قبل از طراحی پایگاه داده انجام دهیم. از طرف دیگر، وجود پایگاه داده جهت پشتیبانی برنامه کاربردی بوده، و بنابراین بایستی جریانی از اطلاعات بین طراحی برنامه کاربردی و طراحی پایگاه داده وجود داشته باشد.

ما باید مطمئن شویم که تمامی کارایی ذکرشده در خصوصیات نیازمندیهای برنامه پایگاه داده در برنامه کاربردی موجود است. که این شامل تراکنش های موجود بین کاربر و داده بوده، که آنرا *طراحی تراکنش* می نامیم. علاوه بر طراحی اینکه چگونه کارایی مورد نیاز بدست میاید، مجبور هستیم تا *واسط* کاربری مختص برنامه کاربردی پایگاه داده را نیز طراحی کنیم.

۳-۹-۱ طراحی تراکنش

تراکنش بیانگر رویدادهای 'جهان واقعی' است که از آن جمله می توان به ثبت عضو جدیدی در شرکت اجاره فیلم، ایجاد توافق نامه اجاره برای کاربر جهت اجاره فیلم، یا افزودن عضو جدیدی از پرسنل اشاره نمود. این تراکنش ها باید به پایگاه داده اضافه شده تا از بهنگام بودن بهمراه 'جهان واقعی' جهت پشتیبانی از اطلاعات موردنیاز کاربران اطمینان حاصل کنند (بخش ۳-۱ را ببینید).

هدف طراحی تراکنش تعریف و مستند سازی مشخصات تراکنش مورد نیاز سیستم پایگاه داده است، که شامل موارد زیر می باشد:

- داده استفاده شده توسط تراکنش ;
- مشخصات عملی تراکنش (تراکنش باید چه چیزی انجام دهد) ;
- خروجی تراکنش;
- اهمیت برای کاربران;
- میزان کاربرد مورد انتظار;

سه نوع اصلی تراکنش داریم که به صورت زیر است :

- تراکنش های بازیابی;
- تراکنشهای بهنگام سازی;
- تراکنشهای مختلط.

تراکنش های بازیابی[1] برای بازیابی داده ها جهت نمایش در صفحه نمایش (یا به صورت گزارش) یا به صورت ورودی به دیگر تراکنشها مورد استفاده قرار می گیرند. برای مثال، عملیات جستجو و نمایش جزئیات فیلم (با دادن شماره فیلم) یک تراکنش بازیابی است. *تراکنش بهنگام سازی*[2] جهت درج رکورد جدید، حذف رکوردهای قدیمی، یا تغییر رکوردهای موجود در پایگاه داده استفاده می شود. برای مثال، عمل درج جزئیات فیلم جدید در پایگاه داده یک تراکنش بهنگام سازی است. *تراکنش مختلط*[3] شامل هر دو تراکنش بازیابی و بهنگام سازی داده است. برای مثال، عمل جستجو و نمایش جزئیات فیلم (توسط شماره فیلم) و سپس بهنگام سازی مقدار میزان اجاره روزانه، یک تراکنش مخلوط می باشد.

۲-۹-۳ طراحی واسط کاربر

علاوه بر طراحی اینکه چگونه کارایی مورد نیاز بدست می آید، ما بایستی واسط کاربر مناسبی را برای برنامه کاربردی پایگاه داده طراحی کنیم. این واسط باید نشان دهنده اطلاعات موردنیاز به شیوه ای کاربر پسند باشد. بعضی اوقات اهمیت طراحی واسط کاربر نادیده گرفته شده و یا تا آخرین مراحل طراحی به تعویق انداخته می شود. با این وجود، طراحی واسط کاربر بایستی یکی از اجزاء مهم سیستم در نظر گرفته شود. اگر طراحی واسط کاربر از لحاظ یادگیری آسان، و طریقه استفاده، آسان و سر راست باشد، کاربران از اطلاعات نشان داده شده به خوبی استفاده خواهند کرد. در غیر این صورت، اگر واسط هیچ یک از این خصوصیات را نداشته باشد، سیستم به طور ناخواسته دچار ایراد میشود. برای مثال، قبل از پیاده سازی فرم و گزارش، ضروری است که اول از همه طرح بندی مان را طراحی کنیم. در جدول ۱-۳ زیر راهنمایی کلی درباره طراحی فرم ها و گزارش ها آمده است (Shneiderman,1992).

---

[1] Retrieval transaction
[2] Update transaction
[3] Mixed transaction

جدول ۱-۳ راهنماییهایی جهت طراحی فرم/ گزارش

عنوان معنی دار
دستورالعمل های قابل درک
گروه بندی و ترتیب گذاری منطقی فیلدها
طراحی بصری جذاب فرمها و گزارشها
برچسبهای آشنا برای فیلدها
واژگان و اختصارات سازگار
استفاده سازگار از رنگها
فضاگذاری و مرزبندی نمایان برای فیلدهای ورود داده
حرکت آسان مکان نما
اصلاح خطا برای فیلدهای داخلی
پیغام های خطا برای مقادیر غیر قابل قبول
فیلدهای اختیاری به طور واضح علامت گذاری شده
پیغامهای توضیحی برای فیلدها
علامت اتمام

۳-۱۰ نمونه سازی

در قسمت های مختلفی از مرحله طراحی، دو گزینه مختلف پیاده سازی کامل برنامه کاربردی پایگاه داده یا ساخت نمونه را داریم. نمونه اصلی یک مدل کاری است که به طور معمول همه ترکیبات موردنیاز و کارائی های یک سیستم نهایی را ندارد. هدف از توسعه برنامه کاربردی پایگاه داده نمونه این است که به کاربران اجازه دهد تا از نمونه ها جهت شرح ترکیب سیستمی استفاده کند، که بهتر عمل کرده و در صورت ناکارامد بودن، خصوصیات جدید و اصلاح شده ای را به برنامه کاربردی پایگاه داده پیشنهاد کند. بدین طریق ما می توانیم به طور کامل نیازمندی ها را مشخص کرده و امکان طراحی سیستم خاصی را بررسی کنیم. نمونه ها بایستی امتیازات عمده ای داشته باشندکه ساختن آن نسبتا ارزان و سریع باشد.

دو نوع استراتژی نمونه سازی درکاربرد روزمره وجود دارد: نمونه سازی نیازمندی ها و نمونه سازی تکاملی.

نمونه سازی نیازمندی ها[1] از نمونه سازی جهت مشخص کردن نیازمندی های برنامه کاربردی پایگاه داده پیشنهادی استفاده می کند و بعد از تکمیل نیازمندیها نمونه دور انداخته می شود. درحالیکه *نمونه سازی تکاملی* برای هدف مشابهی استفاده شده، و تنها تفاوت مهم آن این است که نمونه دیگر دورانداخته نشده و برای توسعه بیشتر پایگاه داده مورد استفاده قرار می گیرد.

۳-۱۱ پیاده سازی

پس از تکمیل مرحله طراحی (که ممکن است شامل نمونه سازی هم باشد)، حال در موقعیتی هستیم که به پیاده سازی پایگاه داده و برنامه کاربردی آن بپردازیم. پیاده سازی پایگاه داده با استفاده از *زبان تعریف داده (DDL)*,DBMS انتخابی یا واسط کاربر گرافیکی *(GUI)* صورت می گیرد. که عملکرد مشابهی را فراهم کرده به طوریکه دستورات DDL سطح پایین را پنهان میکند. جملات DDL برای ایجاد ساختار پایگاه داده و فایلهای خالی آن استفاده میشود. همچنین هرگونه دیدکاربر مشخصی در این مرحله پیاده سازی میشود.

[1] Requirement prototyping

برنامه های کاربردی ترجیحا توسط **زبانهای نسل سوم و چهارم (3GL,4GL)** پیاده سازی می شوند. بخشهایی از این برنامه های کاربردی تراکنش های پایگاه داده بوده، که آنها را توسط زبان دستکاری داده (DML)، DBMS هدف پیاده سازی کرده، و داخل زبانهای برنامه سازی میزبان همچون ویژوال بیسیک، دلفی، سی، جاوا، کوبول، فرترن، ادا، و پاسکال جاسازی می کنیم. همچنین بایستی اجزاء دیگر طراحی کاربردی مانند منوها، فرمهای ورودی داده، و گزارشها را نیز پیاده سازی کنیم. بعلاوه، DBMS هدف شاید دارای ابزار نسل چهارم مختص به خود باشد که بتوان برنامه کاربردی را از طریق تدارک زبان پرس و جوی غیررویه ای، تولید کننده گزارشها، تولید کننده فرمها، و تولید کننده برنامه های کاربردی فراهم آورد.

همچنین کنترلهای امنیت و جامعیت برنامه کاربردی نیز پیاده سازی می شوند. بعضی از این کنترلها توسط DDL پیاده سازی شده، اما برخی دیگر ممکن است نیاز به تعریف و استفاده در خارج از DDL داشته باشند، که از آن جمله میتوان کنترل های سیستم عامل و برنامه های سودمند DBMS را نام برد.

SQL (زبان پرس و جوی ساخت یافته) هم DDL و هم DML می باشد.

۳-۱۲ تبدیل و بارگذاری داده

این مرحله زمانی نیاز است که بخواهیم سیستم پایگاه داده قدیمی را با سیستمی جدید جایگزین کنیم. امروزه، برای DBMS های کنونی کاملا عادی است که برنامه های سودمندی[۱] جهت بارکردن داده های موجود بر روی پایگاه داده جدید داشته باشند. این برنامه ها معمولا نیاز به خصوصیات فایل مبدا و پایگاه داده مقصد داشته، و به طور اتوماتیک داده ها را به فرمت فایل پایگاه داده ای جدید تبدیل می کنند. برای توسعه دهنده هر جا قابل اجرا باشد، می تواند برنامه های کاربردی را از سیستم قدیمی جهت استفاده در سیستم جدید تبدیل کند. هنگامیکه تبدیل و بارگذاری لازم باشد، مرحله بایستی برنامه ریزی شود تا انتقال بدون نقص به سیستم جدید صورت گیرد.

۳-۱۳ تست

قبل از استفاده از پایگاه داده، برنامه پایگاه داده جدید بایستی کاملا تست شود. و این کار توسط استراتژی تست بدقت برنامه ریزی می شود و عمل تست با داده های واقعی انجام می گیرد. تمام مرحله تست با قاعده و با دقت زیاد انجام می شود. تست پایگاه داده عدم وجود خطا را نشان نمی دهد بلکه می تواند تنها وجود خطا های نرم افزاری را نشان دهد. اگر تست به طور کامل انجام شود، خطاهای موجود در برنامه کاربردی و همچنین ساختار پایگاه داده را آشکار خواهد ساخت. بعنوان دومین سود عمل تست، انجام تست بیانگر این نکته است که پایگاه داده و برنامه کاربردی بر طبق خصوصیات تعریف شده و نیازمندی های کارائی اش بخوبی عمل می کند. بعلاوه مقادیر بدست آمده از مرحله تست، کیفیت نرم افزار و قابلیت اطمینان آن را مشخص می کند.

همچون طراحی پایگاه داده، در مرحله تست کاربران سیستم جدید نیز بایستی بکار گرفته شوند. موقعیت ایده ال برای تست سیستم، داشتن پایگاه داده تست در سخت افزار مجزا است، اما اغلب این سخت افزار در دسترس نیست. اگر داده های واقعی مورد استفاده قرار گیرند، از آنجائیکه ممکن است خطایی رخ دهد بنابراین لازم است که از داده ها پشتیبان[۲] گرفته شود. بعد از اینکه تست پایان یافت، سیستم برنامه کاربردی آماده تحویل به کاربران می باشد.

---

۱ Utility

۲ Backup

۳-۱۴ نگهداری عملکردی

در این مرحله، برنامه کاربردی پایگاه داده به سوی مرحله نگهداری منتقل می شود، که فعالیت های زیرین را در بر می گیرد:

- نظارت بر کارایی برنامه کاربردی پایگاه داده. که اگر کارایی از حد قابل قبول پایین بیاید، پایگاه داده نیاز به تنظیم یا سازماندهی مجدد دارد.
- نگهداری و ارتقاء برنامه کاربردی پایگاه داده (در صورت نیاز). نیازمندیهای جدید از طریق مراحل قبلی چرخه حیات داخل برنامه کاربردی پایگاه داده جا داده می شوند.

ما این مراحل را با جزئیات در فصل ۱۶ بحث خواهیم کرد.

خلاصه فصل

- ✓ **سیستم اطلاعاتی** منبعی است که جمع آوری، مدیریت، کنترل و پخش داده/ اطلاعات کلی شرکت را فراهم می سازد.
- ✓ **پایگاه داده** جزء مفهومی **سیستم اطلاعاتی** است. چرخه حیات سیستم اطلاعاتی به طور ذاتی به چرخه حیات پایگاه داده پشتیبان کننده آن مرتبط شده است.
- ✓ مراحل اصلی چرخه حیات برنامه پایگاه داده شامل: برنامه ریزی پایگاه داده، تعریف سیستم، تحلیل و جمع آوری نیازمندیها، طراحی پایگاه داده، انتخاب DBMS (اختیاری)، طراحی برنامه کاربردی، نمونه سازی (اختیاری)، پیاده سازی ، تبدیل و بارکردن داده، تست، و نگهداری عملکردی می باشد.
- ✓ **برنامه ریزی پایگاه داده** یک فعالیت مدیریتی است که باعث کارایی و بازده هر چه بیشتر برنامه کاربردی پایگاه داده می شود.
- ✓ **تعریف سیستم** شامل شناسایی قلمرو مرزهای برنامه کاربردی پایگاه داده می باشد، که دیدهای عمده کاربر را نیز شامل می شود. دید کاربر قادر به نمایش ناحیه وظایف کاری یا برنامه کاربردی تجاری می باشد.
- ✓ **تحلیل و جمع آوری نیازمندیها** عمل جمع آوری و تحلیل نیازمندیهای شرکت (یا قسمتی از شرکت) می باشد که توسط برنامه کاربردی پایگاه داده پشتیبانی می شود، و از این اطلاعات جهت شناسایی نیازمندیهای سیستم جدید استفاده می گردد.
- ✓ سه رهیافت عمده جهت برخورد با دیدهای چندگانه وجود دارد که بنام های روش متمرکزسازی، روش جامعیت دید، و ترکیب این دو روش می باشد.
- ✓ **طراحی پایگاه داده** فرایند ایجاد طراحی پایگاه داده رابطه ای است که از اهداف و عملیات شرکت پشتیبانی می کند. این مرحله شامل طراحی فیزیکی و منطقی پایگاه داده است .
- ✓ هدف از **انتخاب DBMS** انتخاب سیستمی است که نیازهای جاری و آینده شرکت را برآورد ساخته، و تعادلی در هزینه ها که شامل خرید محصولات DBMS و هرگونه سخت افزار یا نرم افزار، و انتقال و آموزش می باشد برقرار می سازد.
- ✓ **طراحی برنامه کاربردی** شامل طراحی برنامه کاربردی پایگاه داده و واسط کاربری است که پایگاه داده را پردازش کرده و مورد استفاده قرار می دهد. این مرحله شامل دو فعالیت عمده است: طراحی تراکنش و طراحی واسط کاربر.
- ✓ **نمونه سازی** شامل ساخت مدل کاری برنامه کاربردی پایگاه داده است، که به کاربران و طراحان اجازه می دهد تا به ارزیابی و مجسم کردن پایگاه داده بپردازند.
- ✓ **پیاده سازی** به واقعیت تبدیل کردن پایگاه داده و طراحی های برنامه کاربردی است.

- ✓ **تبدیل و بارکردن داده** شامل انتقال داده های موجود به پایگاه داده جدید و تبدیل برنامه های کاربردی موجود جهت اجرا شدن در پایگاه داده جدید می باشد.
- ✓ **تست کردن** فرایند اجرای برنامه های کاربردی به منظور پیداکردن خطاهاست.
- ✓ **نگهداری عملکردی** فرایند نظارت و نگهداری از سیستم نصب شده می باشد.

فصل ۴

حقیقت یابی

آنچه در این فصل خواهید آموخت :

- تکنیکهای حقیقت یابی چرخه حیات برنامه پایگاه داده چه مواقعی استفاده می شوند.
- انواع حقایقی که از چرخه حیات برنامه پایگاه داده جمع آوری شده است.
- انواع مستنداتی که از چرخه حیات برنامه پایگاه داده تولید شده است.
- پرکاربردترین تکنیک های حقیقت یابی که استفاده می شوند.
- چگونه از هر تکنیک حقیقت یابی استفاده کنیم و مزایا و معایب هر کدام چیست.
- درباره شرکت اجاره فیلم *StayHome* .
- چگونه تکنیک های حقیقت یابی را به مراحل ابتدایی چرخه حیات برنامه پایگاه داده اعمال کنیم.

در فصل ۳، درباره مراحل چرخه حیات برنامه پایگاه داده مطالبی آموختیم. در این مراحل موقعیت های حساسی وجود دارد که طراح پایگاه داده حقایق مورد نیاز را جهت ساخت برنامه کاربردی پایگاه داده بکار می گیرد. حقایق لازم، حرفه و همچنین کاربران برنامه کاربردی پایگاه داده را که شامل، واژگان، مسائل، فرصتها، محدودیتها، نیازمندیها و اولویتها می باشد را می پوشاند. این حقایق توسط تکنیکهای **حقیقت یابی**[۱] بدست میاید.

در این بخش، درباره موقعیت هائی که توسعه دهنده پایگاه داده قصد استفاده از تکنیکهای حقیقت یابی را دارد و همچنین انواع حقایقی که بایستی بدست آیند را بحث می کنیم. و همچنین خلاصه ای از اینکه چگونه این حقایق جهت تولید انواع عمده مستنداتی که در تمام چرخه حیات برنامه پایگاه داده استفاده می شوند، ارائه می کنیم. به طور خلاصه تکنیک هایی را که عموما مورد استفاده قرار می گیرند را شرح داده و امتیازات و معایب هر یک را مشخص می سازیم. بالاخره نشان می دهیم که چگونه بعضی از این تکنیکها می توانند در مراحل ابتدایی چرخه حیات برنامه پایگاه داده با استفاده از شرکت اجاره فیلم *StayHome* استفاده شوند. در فصل های ۸ تا ۱۶، از بررسی موردی *StayHome* جهت نشان دادن متدلوژی طراحی پایگاه داده استفاده خواهیم کرد.

در سرتاسر این فصل ما از عبارت 'توسعه دهنده پایگاه داده' برای شخص یا گروهی استفاده خواهیم کرد که مسئول تحلیل، طراحی، و پیاده سازی کاربرد پایگاه داده می باشد.

---

[۱] Fact-finding

۴-۱ تکنیکهای حقیقت یابی چه زمانی استفاده می شوند؟

برای حقیقت یابی در چرخه حیات پایگاه داده چندین موقعیت وجود دارد. از آنجائیکه حقیقت یابی در چرخه حیات بسیار مهم است، که شامل برنامه ریزی پایگاه داده، تعریف سیستم، و جمع آوری نیازمندی ها، و مراحل تحلیل می باشد. بنابراین باید در مراحل ابتدایی صورت گیرد و اینجا هست که توسعه دهنده پایگاه داده مطالبی را در مورد واژگان[۱]، مسئله ها، فرصتها، محدودیت ها، نیازمندی ها، و اولویتهای تجاری و کاربران سیستم یاد می گیرد. همچنین زمان طراحی پایگاه داده حقیقت یابی و مراحل بعدی چرخه حیات با وسعت کمتر استفاده می شود. برای مثال، هنگام طراحی فیزیکی پایگاه داده، حقیقت یابی یک امر فنی می شود زیرا توسعه دهنده تلاش بیشتری را در یادگیری هر چه بیشتر DBMS انتخابی برنامه پایگاه داده انجام می دهد. همچنین، از حقیقت یابی هنگام مرحله نهایی، نگهداری عملکردی، استفاده می کنیم تا مشخص کنیم که آیا لازم است که جهت بهبود کارایی سیستم تنظیم صورت گیرد و اینکه آیا سیستم جهت پشتیبانی از نیازمندیهای جدید توسعه کافی یافته است یا نه.

توجه داشته باشید که برای ما بسیار مهم است که از مدت زمان و تلاشی که در حقیقت یابی پروژه پایگاه داده صرف شده است تخمینی داشته باشیم. بررسی خیلی زیاد بزودی منجر به *از کارافتادگی توسط تحلیل*[۲] می شود. همچنین، بررسی خیلی کم نیز منجر به هدر دادن زمان و هزینه ناشی از کار روی راه حل اشتباه یک مساله نادرست می شود.

۴-۲ کدام حقایق باید جمع آوری شوند؟

در سرتاسر چرخه حیات کاربرد پایگاه داده، لازم است توسعه دهنده پایگاه داده حقایقی را درباره حال یا آینده سیستم بدست آورد. جدول ۱-۴ نمونه ای از انواع داده های بدست آمده و مستندات تولید شده برای هر مرحله چرخه حیات فراهم کرده است. همانطور که از فصل ۳ بیاد داریم، مراحل چرخه حیات پایگاه داده بصورت ترتیبی نمی باشند، بلکه در برگیرنده تکرار مراحل قبلی از طریق حلقه بازخورد می باشند. این عمل همچنین برای داده های بدست آمده و مستندات تولید شده در هر مرحله نیز صدق می کند. برای مثال، هنگام طراحی پایگاه داده روبرو شدن با خطایی احتمالا برای سیستم جدید نیاز به بدست آوردن داده اضافی در نیازمندیها خواهد بود.

در بخش ۴-۴، سه مرحله اول چرخه حیات برنامه کاربردی پایگاه داده، که بنام های برنامه ریزی پایگاه داده، تعریف سیستم، و جمع آوری نیازمندیها و تحلیل می باشند را مورد بررسی قرار دهیم. در هر مرحله، فرایند جمع آوری داده با استفاده از تکنیکهای حقیقت یابی را نشان داده و سپس مستنداتی را برای شرکت اجاره فیلم ***StayHome*** تولید می کنیم. با این وجود، قبل از این بخش، اول از همه خلاصه ای از تکنیکهای حقیقت یابی پرکاربرد را ارائه می کنیم.

---

[۱] Terminology

[۲] Paralysis by analysis

جدول ۱-۴ نمونه هایی از داده اخذ شده و مستندات تولید شده برای هر مرحله از چرخه حیات پایگاه داده .

مرحله چرخه حیات برنامه پایگاه داده	نمونه هایی از داده گرفته شده	نمونه هایی از مستندات تولید شده
برنامه ریزی پایگاه داده	اهداف و مقصود پروژه پایگاه داده	شرح وظایف و اهداف برنامه کاربردی پایگاه داده
تعریف سیستم	توصیف دیدهای اصلی کاربر (شامل وظایف شغلی و/ یا نواحی کاربرد تجاری)	تعریف حوزه و محدوده کاربرد پایگاه داده؛ تعریف دیدهای کاربر پشتیبانی شده
تحلیل و جمع آوری نیازمندیها	نیازمندیها برای دیدهای کاربر	خصوصیات نیازمندیهای سیستم و کاربران
طراحی پایگاه داده	واکنشهای کاربران جهت بررسی طراحی منطقی پایگاه داده؛ کارکرد فراهم شده توسط DBMS هدف	طراحی منطقی پایگاه داده؛ (شامل مدل(های) ER، دیکشنری داده، و جداول)؛ طراحی فیزیکی پایگاه داده
طراحی برنامه کاربردی	واکنشهای کاربران جهت بررسی طراحی واسط	طراحی برنامه کاربردی (شامل توصیف برنامه و واسط کاربر)
انتخاب DBMS	کارکرد فراهم شده توسط DBMS هدف	ارزیابی DBMS و پیشنهادات
نمونه سازی	واکنشهای کاربران به نمونه اولیه	خصوصیات سیستم و نیازمندیهای اصلاح شده کاربران
پیاده سازی	کارکرد فراهم شده توسط DBMS هدف	
تبدیل و بارگذاری داده	فرمت داده کنونی؛ امکانات واردکردن داده DBMS هدف	
تست کردن	نتایج تست	استراتژیهای تست استفاده شده؛ تحلیل نتایج تست
نگهداری عملکردی	نتایج تست کارایی؛ جدید کردن یا تغییر دادن کاربر و نیازمندیهای سیستم	راهنمای کاربر؛ تحلیل نتایج کارایی، خصوصیات سیستم و نیازمندیهای اصلاح شده کاربر

۴-۳ تکنیکهای حقیقت یابی

توسعه دهنده پایگاه داده عموما در پروژه پایگاه داده منحصرا از چندین تکنیک حقیقت یابی استفاده می کند. پنج تکنیک عمومی حقیقت یابی وجود دارد :

- بررسی مستندات
- مصاحبه
- مشاهده حرفه در عمل
- تحقیق و پژوهش
- پرسش نامه

۴-۳-۱ بررسی مستندات

زمانیکه می خواهید دیدی از آنچه برای پایگاه داده لازم است داشته باشید، بررسی مستندات می تواند مفید باشد. همچنین ممکن است مستنداتی که میتواند اطلاعات مفید مرتبط با حرفه(یا قسمتی از حرفه) برای شما ارائه دهد را پیدا کنید. اگر مسئله به سیستم کنونی مرتبط است بایستی مستندات مربوط با آن سیستم وجود داشته باشد. بررسی مستندات، فرمها، گزارشها، و فایلهای مرتبط با سیستم کنونی راه خوبی برای دستیابی سریع و درک سیستم است. نمونه هایی از انوع مستنداتی که باید بررسی کنید در جدول ۲-۴ لیست شده است.

۴-۳-۲ مصاحبه

مصاحبه یکی از مهمترین و در عین حال، پرکاربرد ترین تکنیکهای حقیقت یابی است. با مصاحبه شما می توانید رودررو اطلاعاتی را جمع آوری کنید. اهدافی که مصاحبه می تواند در برگیرد عبارت است از پیداکردن حقایق، مشخص ساختن حقایق، واضح سازی حقایق، ایجاد اشتیاق، بدست آوردن موارد مورد بحث کاربران نهایی، شناسایی نیازمندیها، و جمع آوری ایده ها و عقاید. با این وجود، استفاده از تکنیک مصاحبه مستلزم داشتن مهارتهای ارتباطی خوب برای برخورد موثر با افرادی که دارای ارزش متفاوت، نظرات، برتریها، انگیزه ها، و شخصیتهای گوناگون هستند، می باشد. همچون دیگر تکنیکهای حقیقت یابی، مصاحبه همیشه بهترین روش برای تمام موقعیتها نیست. مزایا و معایب استفاده از مصاحبه به عنوان یکی از تکنیک های حقیقت یابی در جدول ۳-۴ لیست شده است.

جدول ۲-۴ نمونه هایی از انواع مستنداتی که بایستی بررسی شوند.

هدف مستند سازی	مثالهایی از منابع مفید
موضوع مساله و نیاز به پایگاه داده را توصیف می کند	یادداشتهای داخلی، ایمیلها، و مدت ملاقاتها شکایات مشتری/کارمند، و مستنداتی که مساله را توصیف کنند گزارشها/ مرور کارایی
حرفه (یا قسمتی از آن) که تحت تاثیر مساله قرار گرفته است	چارت سازمانی، شرح وظایف، برنامه ریزی استراتژیک حرفه اهداف قسمتهایی از حرفه که مطالعه شده است نمونه هایی از گزارشها و فرمهای راهنما نمونه هایی از گزارشها و فرمهای کامپیوتری گزارشها و فرمهای تکمیل شده
سیستم کنونی را شرح میدهد	انواع مختلفی از نمودارها و فلوچارتها دیکشنری داده طراحی برنامه پایگاه داده مستندات برنامه دفترچه راهنمای کاربر/ آموزش

جدول ۳-۴ مزایا و معایب استفاده از مصاحبه بعنوان تکنیک حقیقت یابی.

مزیتها	معایب
به مصاحبه شونده اجازه می دهد تا آزادانه و آشکارا به سوالات جواب دهد	خیلی هزینه بر، و وقت گیر بوده بنابراین غیر عملی است
به مصاحبه شونده اجازه می دهد تا بخشی از پروژه باشد	موفقیت وابسته بر مهارتهای ارتباطی مصاحبه کننده می باشد
به مصاحبه کننده اجازه می دهد تا اظهارات جالب توجه مصاحبه شونده را پیگیری کند	
به مصاحبه شونده اجازه می دهد سوالات را وفق دهد یا با واژه های دیگری بیان کند	
به مصاحبه شونده اجازه می دهد تا زبان بدن مصاحبه شونده را مشاهده کند	

دو نوع مصاحبه وجود دارد، غیرساخت یافته و ساخت یافته. *مصاحبه غیرساخت یافته*[1] تنها با اهداف کلی موجود در ذهن و با کمی، یا به هر میزان، سوالات خاصی انجام می شود. مصاحبه کننده روی مصاحبه شونده از نظر فراهم آوردن چهارچوب و

[1] Unstructured

جهت مصاحبه حساب می کند. در این نوع مصاحبه معمولا تمرکز از دست می رود، و به این علت است که، خواهید دید این نوع مصاحبه معمولا در طراحی و تحلیل پایگاه داده بکار برده نمی شود.

در *مصاحبه ساخت یافته*[1]، مصاحبه کننده مجموعه مشخصی از سوالاتی را از مصاحبه شونده می پرسد. بر اساس پاسخ مصاحبه شونده، مصاحبه کننده سوالات دیگری را جهت آشکار ساختن و بسط هر چه بیشتر مطلب می پرسد. *سوالات نامحدود*[2] به مصاحبه شونده اجازه می دهد تا هر آنطور که مناسب است به سوالات جواب بدهد. نمونه ای از سوال *تشریحی* به این شکل می تواند باشد: 'چرا شما از گزارش های ثبت نام عضو ناراضی هستید؟' *سوالات تک جوابی* جوابها را به انتخاب کوتاه بیان خاصی محدود می کنند. نمونه ای از این نوع سوال بدین صورت می تواند باشد: ' آیا شما گزارش را در زمان مشخصی دریافت کردید؟' یا ' آیا گزارش موجود در ثبت نام عضو دارای اطلاعات درستی است؟' که پاسخ به هر دو سوال بصورت بلی و خیر است.

برای اطمینان از مصاحبه موفق باید افراد مناسبی را برای مصاحبه انتخاب کرده، و برای مصاحبه آماده کنید و مصاحبه را در یک مسیر موثر و کارا هدایت کنید.

۴-۳-۳ مشاهده حرفه در عمل

مشاهده یکی از کاراترین تکنیکهای حقیقت یابی است که می توانید برای درک هر چه بهتر سیستم از آن استفاده کنید. با این تکنیک، هم می توانید در آن مشارکت داشته باشید، و یا افرادی را که فعالیتها را انجام می دهند مشاهده کرده و درباره سیستم یاد بگیرید. این تکنیک مخصوصا زمانی که اعتبار داده های جمع آوری شده توسط روشهای دیگری همچون سوال کم باشد یا اینکه پیچیدگی قسمت خاصی از سیستم مانع از توصیف آشکار توسط کاربران نهایی باشد مفید است.

همچون دیگر روشهای حقیقت یابی، مشاهده موفق به آماده سازی بیشتری نیاز دارد. برای اطمینان از اینکه مشاهده موفق بوده است، نیازمند آن خواهید بود که درباره افراد و عملی که آنها انجام می دهند هرچه بیشتر بدانید. برای نمونه، وقتیکه فعالیتهای کم، متوسط و حداکثر افراد مشاهده شوند این سوال پیش می آید که آیا افراد از اینکه کس دیگری فعالیت شان را مشاهده و ثبت می کنند ناراحت می شوند؟ مزایا و معایب استفاده از مشاهده بعنوان یکی دیگر از تکنیکهای حقیقت یابی در جدول ۴-۴ لیست شده است.

۴-۳-۴ تحقیق و پژوهش

یکی دیگر از تکنیکهای مفید حقیقت یابی انجام تحقیق در مساله و برنامه کاربردی است. مجله های تجاری کامپیوتر، کتابهای مرجع، و اینترنت منابع مفیدی از اطلاعات هستند. آنها درباره اینکه چگونه فرد دیگری همان کار را قبلا انجام داده است اطلاعاتی را فراهم میکنند. بعلاوه شما می دانید که آیا بسته نرم افزاری که مساله شما را حل کند وجود دارد یا نه. مزایا و معایب استفاده از تحقیق در جدول ۵-۴ آمده است.

۴-۳-۵ پرسشنامه

تکنیک حقیقت یابی دیگری جهت مطالعه استفاده از پرسشنامه است. پرسشنامه مستندی با هدف خاصی است که به شما اجازه می دهد در حین اینکه کنترلی روی جوابها داشته باشید حقایق را از تعدادی کثیری از افراد جمع آوری کنید. وقتی با شنوندگان زیادی مواجه هستید، هیچ تکنیک حقیقت یابی دیگری به خودی خود کارایی لازم را در بر ندارد. مزایا و معایب استفاده از پرسشنامه در جدول ۶-۴ آمده است.

---

[1] Structured

[2] Open-ended question

دو فرمت برای پرسشنامه وجود دارد: که یکی قالب آزاد و دیگری قالب ثابت است. *پرسشنامه قالب آزاد* پاسخهای با آزادی بیشتری پیشنهاد می کند. بدین صورت که سوال پرسیده می شود و جواب در فضای خالی که در ادامه سوال میاید ثبت می شود. نمونه هایی از سوالات قالب آزاد بدین صورت است: ' شما چه گزارشهایی دریافت کرده اید و آنها چگونه استفاده می شوند؟ '

جدول ۴-۴ مزایا و معایب استفاده از مشاهده بعنوان یکی از تکنیکهای حقیقت یابی.

مزایا	معایب
اجازه میدهد که اعتبار حقایق و داده ها بررسی شود	اشخاص ممکن است خواسته یا ناخواسته زمانیکه مورد مشاهده قرار گیرند متفاوت عمل کنند
مشاهده کننده هر آنچه که می بیند انجام می دهد	ممکن است عدم مشاهده برخی کارها باعث سطوح متفاوتی از مشکلات یا حجم معمول کارها در آن دوره زمانی شود
مشاهده کننده می تواند داده هایی را که محیط فیزیکی کار را توصیف می کند بدست آورد	بعضی کارها ممکن است بصورتیکه مشاهده می شوند انجام نگیرند
نسبتا ارزان است	احتمالا غیرعملی باشد
مشاهده کننده میتواند کار را ارزیابی کند	

جدول ۵-۴ مزایا و معایب استفاده از تحقیق بعنوان یکی از تکنیکهای حقیقت یابی.

مزایا	معایب
اگر راه حلی موجود باشد میتواند در وقت صرفه جویی کند	میتواند وقت گیر باشد
محقق می تواند ببیند دیگران مسائل مشابه را چگونه انجام داده اند و چه نیازمندی هایی لازم است	نیاز به منابع مناسبی از اطلاعات دارد
محققان را با توسعه کنونی بهنگام نگه می دارد	ممکن است اگر مساله در جای دیگری مستند نشده باشد در نهایت کمکی به حل مسئله نکند

جدول ۶-۴ مزایا و معایب استفاده از پرسشنامه بعنوان یکی از تکنیکهای حقیقت یابی.

مزایا	معایب
افراد می توانند پرسشنامه را براحتی تکمیل کرده و تحویل دهند	تعداد پاسخها شاید کم باشد، احتمالا تنها ٪ ۱۰-۵
نسبتا راه ارزانی جهت جمع آوری اطلاعات از تعداد کثیری از افراد می باشد	پرسشنامه ها ممکن است ناقص تکمیل شوند
افراد احتمالش بیشتر است که حقایق واقعی را به عنوان جواب هایی که میتواند محرمانه نگه داشته شود ارائه کنند	فرصتی در تبدیل و تغییر سوالاتی که ممکن است اشتباه تعبیر شود وجود ندارد
پاسخها میتواند باسانی جدول بندی و تحلیل شود	نمی توان اشارات بدنی افراد را مشاهده و تحلیل کرد
	شاید مهیا کردن پرسشنامه وقت گیر باشد

و ' آیا مشکلی با این گزارشها وجود دارد؟ اگر هست لطفا شرح دهید.' مسائل با سوالات قالب آزاد جوابهایی هستند که ثابت شده فهرست بندی آنها مشکل است، و در بعضی موارد، با سوال پرسیده شده مطابقت نمی کنند.

نوع دوم، پرسشنامه با فرمت ثابت است. *پرسشنامه با فرمت ثابت*[1] شامل سوالاتی است که جواب مشخصی را از افراد می خواهد. در این نوع پرسشنامه، پاسخ دهنده بایستی جواب سوالات را از میان یکی از پاسخها انتخاب کند. که این باعث جدول بندی هرچه سریعتر پاسخها می شود. نمونه ای از سوالات فرمت ثابت بدین صورت است : ' فرمت کنونی گزارش در اجاره های فیلم ایده ال بوده و نبایستی تغییر پیدا کند.' کسی که جواب می دهد ممکن است به این سوال با گزینه های بلی یا خیر جواب دهد، یا اینکه با استفاده از گزینه هایی همچون، کاملا موافقم، موافقم، نظری ندارم، مخالف، و کاملا مخالفم پاسخ دهد.

۴-۴ بررسی موردی *StayHome*

در این بخش، ابتدا بررسی موردی *StayHome* را توضیح می دهیم. سپس از بررسی موردی *StayHome* استفاده کرده تا نشان دهیم که چگونه می توانید پروژه پایگاه داده ای را در مراحل ابتدایی چرخه حیات برنامه پایگاه داده از طریق برنامه ریزی پایگاه داده، تعریف سیستم، و جمع آوری نیازمندیها و مراحل تحلیل، ایجاد کنید.

۴-۴-۱ بررسی موردی *StayHome*- مرور

بررسی موردی فوق شرکتی با نام *StayHome* را توصیف میکند که به اجاره فیلم به عضوهایش می پردازد. اولین شعبه *StayHome* در سال ۱۹۸۲ در سیاتل برپا شد اما هم اکنون شرکت رشد کرده و دارای چندین شعبه شده است. موفقیت شرکت ناشی از سرویس درجه یک و فیلم های زیادی است که شرکت جهت اجاره فراهم کرده است.

StayHome درحال حاضر دارای ۲۰۰۰ پرسنل است که در ۱۰۰ شعبه مشغول کارند. وقتی عضوی از پرسنل به شرکت ملحق میشود، فرم ثبت نام *StayHome/استفاده* میشود. این فرم برای Mary Martinez در شکل ۱-۴ نشان داده شده است.

هر شعبه مدیری دارد که دارای چند سرپرست است. مدیر، مسئول برپایی روزانه شعبه داده شده بوده و هر ناظر سرپرستی گروهی از پرسنل را بر عهده دارد. نمونه ای از صفحه اول گزارشی که عضوهایش در شعبه سیاتل کار می کنند در شکل ۲-۴ نشان داده شده است.

[1] Fixed-format questionnaires

شکل ۱-۴

فرم ثبت نام پرسنل *StayHome* برای Mary Martinez.

StayHome
Staff Registration Form

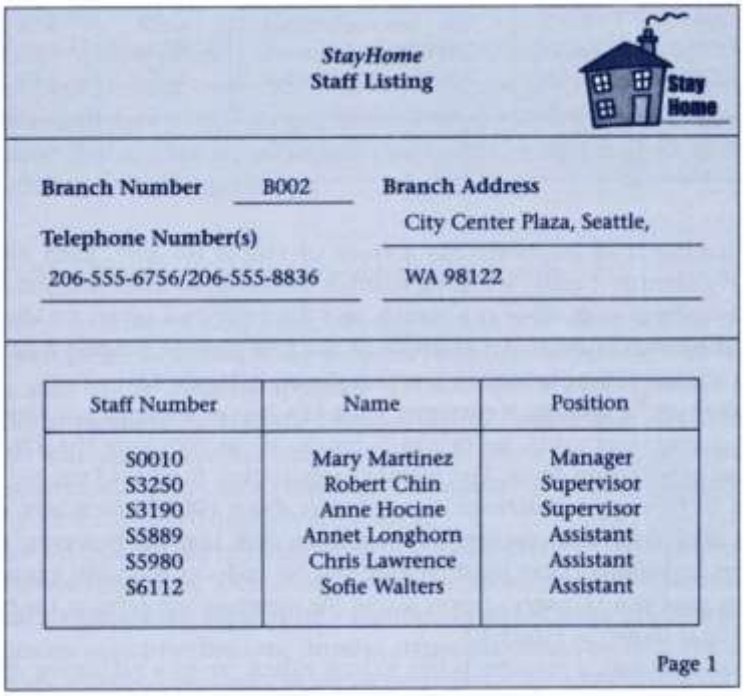
Stay Home

Staff Number S0010
Full Name Mary Martinez
Position Manager
Salary 50000

Branch Number B002
Branch Address City Center Plaza, Seattle, WA 98122
Telephone Number(s) 205-555-6756/206-555-8836

شکل ۲-۴

نمونه ای از صفحه اول گزارش پرسنلی که در شعبه *StayHome* در سیاتل قرار دارد.



StayHome
Staff Listing

Stay Home

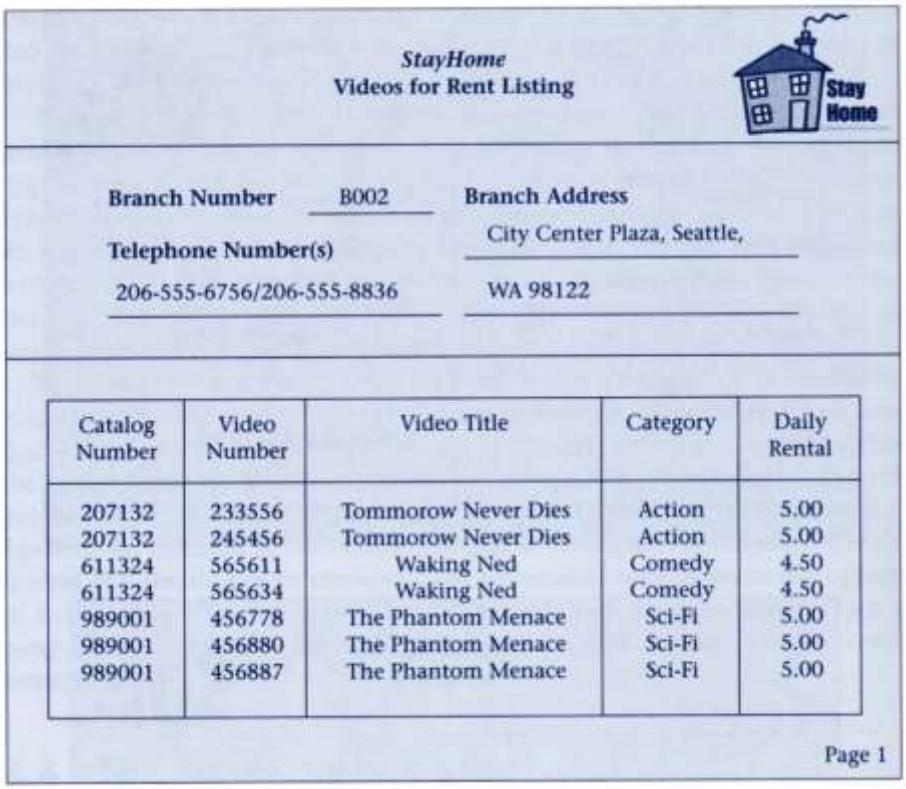
Branch Number B002
Branch Address City Center Plaza, Seattle, WA 98122
Telephone Number(s) 206-555-6756/206-555-8836

Staff Number	Name	Position
S0010	Mary Martinez	Manager
S3250	Robert Chin	Supervisor
S3190	Anne Hocine	Supervisor
S5889	Annet Longhorn	Assistant
S5980	Chris Lawrence	Assistant
S6112	Sofie Walters	Assistant

Page 1

شکل ۳-۴

نمونه ای از صفحه اول گزارش که فیلم های موجود در شعبه سیاتل *StayHome* را لیست کرده است.



StayHome
Videos for Rent Listing

Branch Number B002 **Branch Address** City Center Plaza, Seattle,

Telephone Number(s) 206-555-6756/206-555-8836 WA 98122

Catalog Number	Video Number	Video Title	Category	Daily Rental
207132	233556	Tommorow Never Dies	Action	5.00
207132	245456	Tommorow Never Dies	Action	5.00
611324	565611	Waking Ned	Comedy	4.50
611324	565634	Waking Ned	Comedy	4.50
989001	456778	The Phantom Menace	Sci-Fi	5.00
989001	456880	The Phantom Menace	Sci-Fi	5.00
989001	456887	The Phantom Menace	Sci-Fi	5.00

Page 1

هر شعبه *StayHome* تعدادی فیلم برای اجاره دارد. هر فیلم به طور یکتا توسط شماره کاتالوگ شناسایی می شود. با این وجود، در بیشتر موارد، چندین کپی از هر فیلم در شعبه موجود است، و کپی های مجزا توسط شماره فیلم شناسایی می شوند. نمونه صفحه اول گزارش فیلم های موجود در شعبه سیاتل در شکل ۳-۴ نشان داده شده است.

قبل از اجاره کردن فیلم، مشتری باید بعنوان عضوی از *StayHome* ملحق شود. سپس، از او خواسته می شود تا فرم ثبت نام عضو *StayHome* را پرکند. فرم ثبت نام عضو برای Don Nelson در شکل ۴-۴ نشان داده شده است. *StayHome* هم اکنون در حدود ۱۰۰۰۰۰ عضو دارد. مشتری می تواند در بیش از یک شعبه ثبت نام کند; و در هر موقعیت باید فرم ثبت نام جدیدی پر شود. نمونه ای از اولین صفحه گزارش مدیران که عضوهای ثبت نام کرده در شعبه سیاتل را لیست کرده است در شکل ۵-۴ نشان داده شده است.

شکل۴-۴

فرم ثبت نام عضو *StayHome* برای Don Nelson.

StayHome
Member Registration Form

Member Number (Enter if known) M284354	**Branch Number** B002
Full Name Don Nelson	**Branch Address** City Center Plaza, Seattle, WA 98122
Member Address 123 Suffolk Lane, Seattle, WA 98117	**Registered By** Robert Chin
Date Registered 09-Oct-98	

بعد از اینکه ثبت نام شد، عضو می تواند در هر بار مراجعه ۱۰ عدد فیلم، اجاره کند. اگر عضو بخواهد یک یا چند فیلم را انتخاب کند، فرم اجاره فیلم *StayHome* تکمیل میشود. نمونه ای از فرم تکمیل شده برای Claire Sinclair که دو فیلم *Waking Ned* و *The Phantom Menace* را اجاره کرده در شکل ۶-۴ آمده است.

شرکت *StayHome* رشد کرده است، و مشکلاتی در زمینه مدیریت داده بکار رفته و تولید شده توسط شرکت بوجود آمده است. برای اطمینان از موفقیت پایدار شرکت، رئیس شرکت فورا برنامه پایگاه داده ای درخواست کرده تا مشکلات مدیریت داده ها را حل کند.

شکل ۵-۴

نمونه اولین صفحه از گزارشی که عضوهای ثبت نام کرده در شعبه سیاتل *StayHome* را نشان میدهد.

StayHome
Members Listing

Branch Number B002 **Branch Address** City Center Plaza, Seattle, WA 98122

Telephone Number(s) 206-555-6756/206-555-8836

Member Number	Name	Address	Date Joined
M129906	Karen Homer	634–12th Avenue, Seattle, WA 98123	10-Jan-94
M189976	John Hood	4/4 Rosie Lane, Seattle	21-May-95
M220045	Jamie Peters	5A–22nd Street, Seattle, WA 98451	20-May-96
M228877	Claire Sinclair	44B–16th Street, Seattle, WA 98123	28-Aug-96
M265432	Janet McDonald	1 Lincoln Way, Seattle, WA 98234	19-Aug-97
M284354	Don Nelson	123 Suffolk Lane, Seattle, WA 98117	09-Oct-98
M284666	William Carring	1 Sparrowhill Way, Seattle, WA 98111	10-Oct-99

Page 1

شکل ۶-۴

نمونه فرم اجاره فیلم برای Claire Sinclair شرکت *StayHome*

StayHome
Video Rental

Member Number M228877 Branch Number B002

Member Name Claire Sinclair

Branch Address City Center Plaza, Seattle, WA 98122

Video Number	Video Title	Daily Rental	Date Out	Date In	Total Rental
565611	Walking Ned	4.50	12-Dec-99	14-Dec-99	4.50
476667	The Phantom Manace	5.00	13-Dec-99	15-Dec-99	5.00

۴-۴-۲ بررسی موردی ***StayHome*** – برنامه ریزی پایگاه داده

گام اول در توسعه برنامه کاربردی پایگاه داده به طور واضح تعریف طرح *اجمالی ماموریت*[۱] برای پروژه پایگاه داده است. طرح اجمالی ماموریت اهداف اصلی برنامه پایگاه داده را تعریف کرده و کمک می کند که هدف پروژه پایگاه داده آشکار شده و مسیر مشخصی را بسوی برنامه کاربردی پایگاه داده موثر و کارا فراهم آورد.

بعد از تعریف طرح اجمالی ماموریت، فعالیت بعدی شامل شناسایی **اهداف ماموریت**[۲] است. هر هدف ماموریت باید وظیفه مشخصی را که پایگاه داده باید پشتیبانی کند را بشناساند. فرض بر این است که اگر پایگاه داده اهداف ماموریت را پشتیبانی کند در نتیجه شرح وظایف نیز بایستی پشتیبانی شود. شرح وظایف و اهداف ماموریت شاید باتفاق هم با کمی اطلاعات اضافی که مشخص می شوند باشند، به عبارت کلی، کاری که باید انجام گیرد، منابعی که با آن انجام شود، و هزینه ای که باید برای همه آنها پرداخت شود.

ایجاد طرح اجمالی ماموریت برای کاربرد پایگاه داده ***StayHome***

شما باید ایجاد طرح اجمالی ماموریت کاربرد پایگاه داده ***StayHome*** را با مصاحبه ای که با رئیس شرکت و پرسنل های دیگری که توسط رئیس معرفی شده است انجام خواهید داد، آغاز کنید. معمولا در این مرحله سوالات تشریحی می توانند بسیار مفید باشند. برای نمونه، شما (توسعه دهنده پایگاه داده) ممکن است مصاحبه را با پرسیدن سوالات زیر از رئیس شرکت آغاز کنید:

---

[۱] Mission statement
[۲] Mission objective

توسعه دهنده پایگاه داده ′هدف شرکت شما چیست؟′

رئیس ′ما دامنه وسیعی از فیلم های ویدیویی را جهت اجاره به عضوهایی که در شعبه های ما در سرتاسر ایالات متحده ثبت نام کرده اند فراهم می کنیم.′

توسعه دهنده پایگاه داده 'چرا شما فکر می کنید به پایگاه داده نیاز دارید؟

رئیس ' راستش را بگویم اینکه ما نمی توانیم حریف موفقیت خود شویم. در سالیان گذشته، ما چندین شعبه جدید را باز کردیم، و در هر شعبه هم اکنون تعداد زیادی فیلم برای اجاره به اعضایی که هر روزه در حال افزایش است ارائه می کنیم. بنابراین، این موفقیت با مشکلات مدیریت داده همراه شده، که به این معنی است که سطح خدماتی که ارائه می کنیم پایین آمده است. همچنین، کمبود همکاری بین شعبه ها در اشتراک اطلاعات بوجود آمده که میتواند نگران کننده باشد.'

توسعه دهنده پایگاه داده ' از کجا می دانید که پایگاه داده می تواند مشکلات شما را حل کند؟'

رئیس ' کلا میدانم که ما در کاغذ بازی غرق شده ایم. ما به چیزی نیاز داریم که کارمان را سریع کند، یعنی، چیزی که کارهایی را که بطور مداوم انجام می گیرد را اتوماتیک کند. همچنین، می خواهم که شعبه ها با همدیگر کار کنند. پایگاه داده این را انجام می دهد، مگر انجام نمی دهد؟'

پاسخ به این قبیل سوالات به شما کمک می کند تا شرح وظایف را فرموله کنید. برای نمونه، شرح وظایف برای پایگاه داده ***StayHome*** در شکل ۷-۴ آمده است. وقتی احساس کنید شرح وظایف واضح و مشخصی دارید و پرسنل ***StayHome*** با آن موافقند، می توانید به تعریف اهداف ماموریت بپردازید.

'هدف کاربرد پایگاه داده ***StayHome*** نگهداشتن داده ای است، که ما جهت پشتیبانی از حرفه اجاره فیلمی عضوهای ما تولید کرده ایم، و همچنین آسان کردن کارهای شرکت و اشتراک اطلاعات بین شعبه ها می باشد.'

ایجاد اهداف ماموریت برای کاربرد پایگاه داده ***StayHome***

فرآیند ایجاد اهداف ماموریت انجام مصاحبه با اعضای خاصی از پرسنل است. معمولا در این مرحله دوباره سوالات تشریحی یکی از مفیدترین کارها است. برای بدست آوردن محدوده کاملی از اهداف ماموریت باید با اعضای مختلفی از پرسنل شرکت ***StayHome*** *مصاحبه* کنید. نمونه هایی از سوالات نوعی که شما باید بپرسید بدین صورت است:

'تعریف شغل شما چیست؟'
' در یک روز معمولا چه نوع کارهایی انجام می دهید؟'
'با چه نوع داده هایی کار می کنید؟'
'چه نوع گزارشهایی را استفاده می کنید؟'
'نیاز به ثبت چه چیزهایی دارید؟'
' شرکت شما چه سرویسهایی را برای اعضاء فراهم کرده است؟'

این سوالات (یا مشابه اینها) همچنین از رئیس و اعضا پرسنل در نقشهای مدیر، سرپرست، معاون، و خریدار شرکت ***StayHome*** نیز پرسیده می شود. توجه داشته باشید، شاید لازم باشد تا سوالاتتان را با شرایط کسی که می خواهید مصاحبه کنید سازگار کنید.

رئیس

توسعه دهنده پایگاه داده ′چه نقشی را در شرکت بر عهده دارید ؟′

رئیس ′من سرپرستی شرکت را بعهده دارم تا مطمئن شوم که بهترین فیلم ها را جهت اجاره به اعضایمان فراهم می کنیم'

توسعه دهنده پایگاه داده ' در یک روز معمولا چه نوع کارهایی را انجام می دهید؟'

رئیس ′ من بر دایر بودن هر شعبه توسط مدیرانمان نظارت می کنم. همچنین تلاش می کنم تا مطمئن شوم که شعبه ها با همدیگر به خوبی کار می کنند و اطلاعات فیلمها و اعضاء را به خوبی به اشتراک گذاشته اند. من همچنین از انجام گرفتن عمل خرید برای شرکت توسط خریدار آگاه می شوم; منظورم آن شخصی است که عهده دار خرید فیلمها برای تمام شعبه هایمان است. من معمولا سعی می کنم با یک یا دو بار تشکیل جلسه در ماه از کارکرد مدیران شعبه آگاه شوم′

توسعه دهنده پایگاه داده 'با چه نوع داده هایی کار می کنید؟'

رئیس ′من باید به هر آنچه که در شرکتمان تولید یا استفاده می شود دسترسی داشته باشم. آن شامل داده هایی درباره پرسنل، فیلمها، اجاره ها، اعضاء، تامین کنندگان فیلم، و سفارش دهندگان فیلم است. که این بمعنی همه چیز است!′

توسعه دهنده پایگاه داده 'از چه نوع گزارشهایی استفاده می کنید؟'

رئیس ' من باید بدانم که چه چیزهایی به شعبه ها می رود.که اطلاعاتم را ازگزارشهای مختلف پرسنل، فیلمهای موجود در انبار، فیلمهای اجاره ای، اعضاء، تامین کنندگان فیلم، و سفارشها بدست می آورم.'

توسعه دهنده پایگاه داده 'نیاز به ثبت چه چیزهایی دارید؟'

رئیس ' همانطور که قبلا گفتم، من نیاز دارم تا همه چیز را ثبت کنم، نیاز دارم تمام تصاویر را ببینم.
اینطور هست یا نه؟'

توسعه دهنده پایگاه داده 'چه سرویس هایی را شرکت شما برای اعضاء فراهم کرده است؟'

رئیس 'ما سعی می کنیم که بهترین و رقابتی ترین فیلم ها از نظر قیمت را سرویس دهی کنیم.'

مدیر

توسعه دهنده پایگاه داده 'تعریف شغل شما چیست؟'

مدیر 'عنوان شغل من مدیر است. من هر روزه به شعبه ام سرکشی می کنم تا از فراهم کردن بهترین سرویس به اعضاء مطلع شوم.'

توسعه دهنده پایگاه داده 'چه نوع کارهایی در یک روز معمول انجام می دهید؟'

مدیر 'من بایستی از اینکه شعبه از لحاظ تعداد و نوع پرسنل نقصی نداشته باشد اطمینان حاصل کنم. همچنین بر اجاره فیلمها نظارت می کنم تا مطمئن شوم که فیلمهای مناسب انتخابی برای عضوهایمان داشته باشیم، اگرچه، من شخصا فیلمها را نمیخرم-که آن توسط خریدار شرکت صورت میگیرد. من بر ثبت نام عضوهای جدید و فعالیت اجاره کردن اعضاء کنونی نظارت می کنم.'

توسعه دهنده پایگاه داده 'با چه نوع داده هایی کار می کنید؟'

مدیر 'من به داده هایی درباره پرسنل، فیلمها، اجاره ها، و اعضاء نیاز دارم.'

توسعه دهنده پایگاه داده ' چه نوع گزارشهایی را استفاده می کنید؟ '

مدیر 'گزارشهای مختلفی از، پرسنل، فیلمهای موجود در انبار، اجاره فیلم، و عضوها.'

توسعه دهنده پایگاه داده 'نیاز به ثبت چه چیزهایی دارید؟'

مدیر 'پرسنل، فیلمهای موجود در انبار، اجاره های فیلم، و عضوها.'

توسعه دهنده پایگاه داده 'چه سرویس هایی را شرکت شما برای اعضاء فراهم کرده است؟'

مدیر 'ما سعی می کنیم تا بهترین سرویس های اجاره فیلم را در منطقه فراهم کنیم.'

ناظر

توسعه دهنده پایگاه داده ' تعریف شغل شما چیست؟'

ناظر 'عنوان شغل من نظارت است. من سرپرستی گروه کوچکی از پرسنل را بر عهده دارم و مستقیما با عضوها در ارتباط هستم تا سرویس اجاره فیلم را فراهم کنم. '

توسعه دهنده پایگاه داده 'چه نوع کارهایی در یک روز معمول انجام می دهید؟'

ناظر 'من پرسنل را به وظایف مشخصی اختصاص می دهم، همچون سر و کار داشتن با اعضاء، تامین دوباره قفسه ها، و بایگانی اسناد مبتنی بر کاغذ. من به پرسشهای عضوها درباره فیلم های اجاره ای پاسخ می دهم. همچنین فیلمهای برگردانده شده را از اعضاء باز پس می گیرم. من جزئیات مشخصات اعضاء را بروز نگه میدارم همچنین وقتی کسی بخواهد به شرکت ما ملحق شوم آنها را ثبت نام می کنم.'

توسعه دهنده پایگاه داده 'با چه نوع داده هایی کار می کنید؟'

ناظر 'داده های درباره پرسنل، فیلمها، اجاره ها و اعضاء.'

توسعه دهنده پایگاه داده 'چه نوع گزارشهایی را استفاده می کنید؟'

ناظر 'گزارشهایی درباره پرسنل و فیلمهای موجود در انبار.'

توسعه دهنده پایگاه داده 'چه چیزهایی را نیاز دارید تا ثبت کنید؟'

ناظر 'درباره اینکه آیا فیلم مشخصی برای اجاره موجود است یا آیا جزئیات عضوهای ما بهنگام است یا نه.'

معاون

توسعه دهنده پایگاه داده 'تعریف شغل شما چیست؟'

معاون ′من معاون هستم. من مستقیما با اعضاء در تهیه سرویس اجاره فیلم سروکار دارم′

توسعه دهنده پایگاه داده 'چه نوع کارهایی در یک روز معمول انجام می دهید؟'

معاون ′به سوالات اعضا درباره فیلمهای اجاره ای پاسخ می دهم. مثل این نوع سوال که:' آیا چنین و چنان فیلمی را دارید؟' همچنین فیلمهای برگردانده شده را بازپس گرفته و در قفسه ها قرار می دهم، و وقتی سرم مشغول نباشد به بایگانی کاغذها نیز می پردازم′

توسعه دهنده پایگاه داده 'با چه نوع داده هایی کار می کنید؟'

معاون ′داده هایی درباره فیلم، اجاره ها، و اعضا′

توسعه دهنده پایگاه داده 'چه نوع گزارشهایی را استفاده می کنید؟'

معاون ′هیچ گزارشی′

توسعه دهنده پایگاه داده 'چه چیزهایی را نیاز دارید تا ثبت کنید؟'

معاون ′آیا فیلم مشخصی برای اجاره موجود است′

توسعه دهنده پایگاه داده 'چه سرویسهایی را شرکت شما برای اعضاء فراهم کرده است؟'

معاون ′سعی می کنیم به سوالاتی درباره فیلمهای موجود در انبار پاسخ دهیم مانند:' آیا شما فیلم Ewan MacGregor را دارید ؟' شما نمی توانید باور کنید که مشتری از شما چه سوالی می کند خوشبختانه اگر یکی از ما هم ندانیم دیگری هست که جواب بدهد′

خریدار

توسعه دهنده پایگاه داده 'تعریف شغل شما چیست؟'

خریدار 'من خریدار هستم. من مسئول خرید فیلمهای تمامی شعبه ها می باشم که جهت اجاره لازم است.'

توسعه دهنده پایگاه داده 'چه نوع کارهایی در یک روز معمول انجام می دهید؟'

خریدار 'من مستقیما با مدیر شعبه و تامین کننده فیلم در ارتباط ام. من به تقاضاهای مدیران در رابطه با تامین فیلم های مشخصی پاسخ می دهم. این شغل من است که وقتی با تامین کننده فیلم سر وکار دارم مطمئن شوم که معامله خوبی را برای شرکت فراهم کرده ام. البته، من وابسته به مدیران هستم که کارشان را انجام دهند – نمی خواهم که فیلمی را تهیه کنم که آنها نیاز ندارند یا در انبار کپی های بیشتری از آن وجود دارد. وقتی فرصت کردم، بررسی خودم را از لحاظ آگاهی از اینکه هر شعبه انتخابهای مناسبی از فیلمها را داشته باشد را انجام می دهم.'

توسعه دهنده پایگاه داده 'با چه نوع داده هایی کار می کنید؟'

خریدار 'من باید به داده هایی درباره شعبه ها، فیلمها، فیلمهای آماده اجاره، اعضاء، سفارش دهندگان فیلم، و تامین کنندگان دسترسی داشته باشم.'

توسعه دهنده پایگاه داده 'چه نوع گزارشهایی را استفاده میکنید؟'

خریدار 'من به گزارشهایی درباره سفارشهایی که برای فیلمها داده ام نیاز دارم. همچنین به گزارشهای مختلفی که به من فیلمهای موجود را نشان دهد، و فیلم های اجاره، و عضوهای موجود در هر شعبه و در تمام شعب را نیاز دارم.'

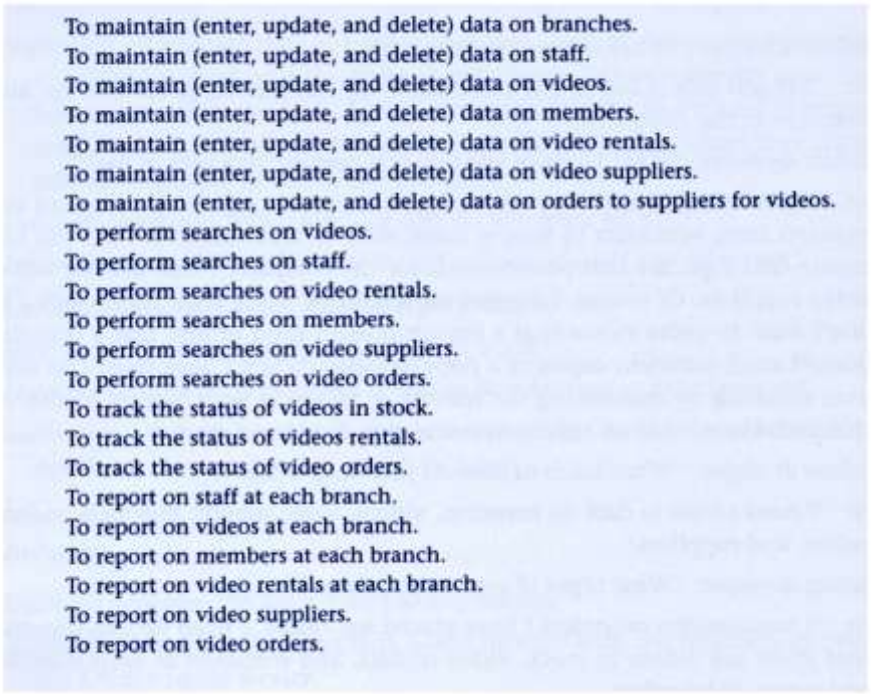
توسعه دهنده پایگاه داده 'نیاز به ثبت چه داده هایی دارید؟'

خریدار 'من به اطلاعات بهنگام شده ای درباره سفارش فیلمها نیاز دارم. همچنین باید بدانم چه چیزهایی به شعبه در مورد موجودی فیلمها و فیلمهای اجاره رد و بدل میشود. و همانطور که قبلا گفتم، نمیخواهم فیلمی را سفارش دهم که آنها نیاز ندارند.'

توسعه دهنده پایگاه داده 'چه سرویسهایی را شرکت شما برای اعضاء فراهم کرده است؟'

خریدار 'ما سعی می کنیم تا بهترین انتخاب فیلمها را برای اعضاء فراهم کنیم.'

پاسخگویی به اینگونه سوالات به شما کمک می کند تا اهداف ماموریت را فرموله کنید. برای نمونه، اهداف ماموریت برای شرکت ***StayHome*** در شکل ۸-۴ آمده است.



شکل ۸-۴

اهداف ماموریت برای کاربرد پایگاه داده *StayHome*

۴-۴-۳ بررسی موردی *StayHome* – تعریف سیستم

هدف از مرحله تعریف سیستم تعریف محدوده و مرز کاربرد پایگاه داده و دیدهای اصلی کاربر است. دید کاربر ارائه کننده نیازمندی هایی است که بایستی توسط کاربرد پایگاه داده پشتیبانی شده و توسط نقش شغل[۱] مشخصی (همچون مدیر یا معاون) یا ناحیه کاربرد تجاری (همچون کنترل وجودی یا اجاره فیلم) تعریف شوند.

تعریف محدوده سیستم برای کاربرد پایگاه داده StayHome

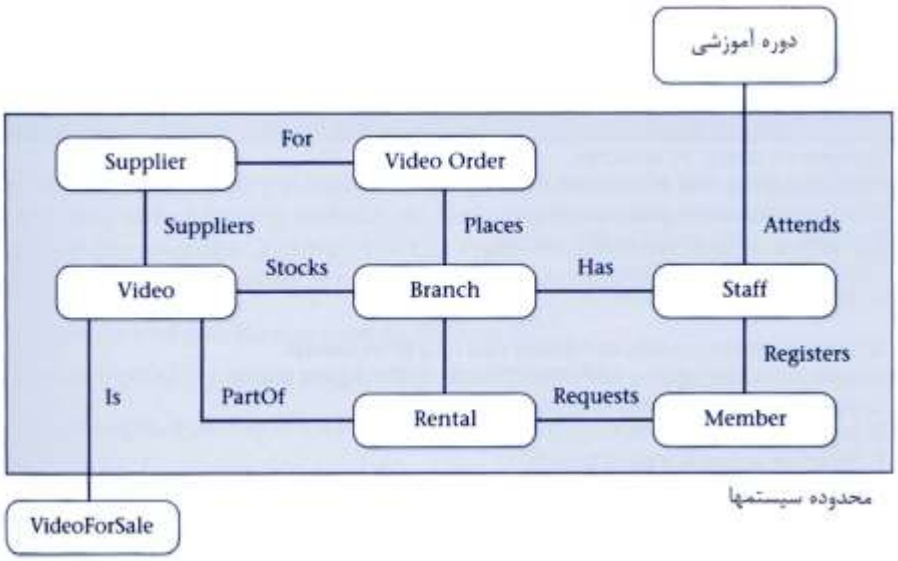
طی این مرحله از چرخه حیات کاربرد پایگاه داده، شما باید جهت وضوح و توسعه هر چه بیشتر داده های بدست آمده در مرحله قبل، از مصاحبه استفاده کنید. با این وجود، ممکن است از دیگر تکنیکهای حقیقت یابی همچون بررسی مستندات نمونه که در بخش۱-۴-۴ نشان داده شده اند استفاده کنید. هم اکنون باید داده هایی را که تا اینجا جمع آوری شده اند را تحلیل کنید تا محدوده کاربرد پایگاه داده را تعریف کنید. محدوده سیستمها برای برنامه کاربرد پایگاه داده ***StayHome*** در شکل ۹-۴ آمده است. موارد موجود در محدوده بیانگر انواع اصلی داده های ذکر شده در مصاحبه هستند و راهنمای نه چندان واضحی درباره اینکه چگونه این داده ها به هم مربوطند.

شناسایی دیدهای اصلی کاربر برای کاربرد پایگاه داده StayHome

شما حالا باید داده هایی را که تا کنون جمع آوری شده است را تحلیل کنید تا دیدهای عمده کاربر کاربرد پایگاه داده را تعریف کنید. اکثریت داده درباره دیدهای کاربر طی مصاحبه با رئیس و پرسنل در نقش مدیر، سرپرست، معاون، و خریدار جمع آوری شده اند. دیدهای اصلی کاربر برای کاربرد پایگاه داده در شکل ۱۰-۴ نشان داده شده است.

---

[۱] Job role



شکل ۹-۴

محدوده سیستمهای کاربرد پایگاه داده *StayHome*.

۴-۴-۴ بررسی موردی ***StayHome*** – تحلیل و جمع آوری نیازمندیها

طی این مرحله، شما باید سعی کنید تا جزئیات بیشتری از دیدهای کاربری که در مرحله قبل شناسایی کرده اید را جمع آوری کنید، تا برای ایجاد *خصوصیات نیازمندیهای کاربر* داده هایی که باید در پایگاه داده نگهداری شوند و اینکه چگونه داده ها باید استفاده شوند، با جزئیات شرح داده شوند. درحالیکه اطلاعات بیشتری را درباره دیدهای کاربر جمع آوری می کنیم، بایستی همچنین هرگونه نیازمندیهای کلی سیستم را جمع آوری کنیم. هدف از جمع کردن این اطلاعات ایجاد *خصوصیات سیستم* است، که هرگونه ویژگی دیگری که باید در کاربرد پایگاه داده لحاظ شود از جمله شبکه و نیازمندیهای دستیابی مشترک، نیازمندی های کارایی، و سطح امنیت مورد نیاز را توصیف می کند.

درحالیکه داده های مربوط به نیازمندیهای دیدهای کاربر و به طور کلی سیستم را جمع آوری می کنید، درباره اینکه سیستم چگونه کار می کند مطالبی یاد می گیرید. البته، هنگام ساخت پایگاه داده جدید باید چیزهای خوب سیستم قدیمی را و سودهایی را که سیستم جدید در پی خواهد داشت را حفظ کنید.

فعالیت مهم مرتبط با این مرحله تصمیم گیری درباره این است که چگونه با موقعیتهایی که بیش از یک دید کاربر دارید برخورد کنید. همانطور که در بخش ۶-۳ بحث کردیم، سه نوع روش عمده برخورد با چندین دید کاربر موجود است، روش متمرکزسازی، روش یکپارچگی دید، و ترکیبی از این دو روش وجود دارد. دراینجا مختصرا نشان می دهیم که چگونه می توانید از این روشها استفاده کنید.

شکل ۱۰-۴

دیدهای اصلی کاربر برای کاربرد پایگاه داده *StayHome* .

دید کاربر	نیازمندیهای کاربر
Director	To report on staff at all branches. To report on videos at all branches. To report on members at all branches. To report on video rentals at all branches. To report on video suppliers. To report on video orders.
Manager	To maintain (enter, update, and delete) data on a given branch. To maintain (enter, update, and delete) data on staff at a given branch. To perform searches on staff at all branches. To report on staff at a given branch. To report on videos at a given branch. To report on members at a given branch. To report on video rentals at a given branch.
Supervisor	To maintain (enter, update, and delete) data on videos at a given branch. To maintain (enter, update, and delete) data on members at a given branch. To maintain (enter, update, and delete) data on video rentals at a given branch. To perform searches on videos at all branches. To perform searches on video rentals at a given branch. To perform searches on members at a given branch. To track the status of videos in stock at a given branch. To track the status of video rentals at a given branch. To report on staff at a given branch.
Assistant	To maintain (enter, update, and delete) data on video rentals at a given branch. To maintain (enter, update, and delete) data on members at a given branch. To perform searches on videos at all branches. To perform searches on video rentals at a given branch. To perform searches on members at a given branch. To track the status of videos in stock at a given branch. To track the status of video rentals at a given branch.
Buyer	To maintain (enter, update, and delete) data on videos. To maintain (enter, update, and delete) data on video suppliers. To maintain (enter, update, and delete) data on video orders. To perform searches on videos at all branches. To perform searches on video suppliers. To perform searches on video orders. To track the status of video orders. To report on videos at all branches. To report on video rentals at all branches. To report on members at all branches. To report on video suppliers. To report on video orders.

جمع آوری اطلاعات بیشتر در دیدهای کاربر پایگاه داده StayHome

برای درک هر چه بیشتر نیازمندیهای هر دید کاربر، شما احتمالا دوباره از تکنیکهای حقیقت یابی همچون مصاحبه و مشاهده سیستم درعمل استفاده خواهید کرد. نمونه هایی از سوالاتی که بایستی درباره داده مورد نیاز با دید کاربر است بپرسید (با X نشان داده شده است) بدین صورت است :

'چه نوع داده هایی نیاز دارید تا در X نگه دارید؟'

'چه نوع داده هایی روی X انجام می دهید؟'

برای نمونه، شاید از مدیر شعبه این نوع سوالات را بپرسید :

توسعه دهنده پایگاه داده ' چه نوع داده هایی نیاز دارید درباره پرسنل نگه دارید؟'
مدیر ' داده ای که درباره اعضا پرسنل نگه می داریم نام، سمت، و حقوق است. هر عضو پرسنل شماره پرسنلی دارند که در سرتاسر شرکت منحصر بفرد است.'
توسعه دهنده پایگاه داده 'چه نوع داده هایی را روی پرسنل انجام می دهید؟'
مدیر 'من باید بتوانم جزئیات عضو جدید را درج کنم و هنگام خروج آنها را حذف کنم. همچنین نیاز دارم جزئیات پرسنل را بروز نگه داشته و گزارشی از لیست نامها، شغلها، و حقوق هر عضوی از پرسنل شعبه ام را چاپ کنم. من باید بتوانم تا سرپرست ها را اختصاص دهم تا مراقب پرسنل باشند. بعضی اوقات وقتی میخواهم با دیگر شعب ارتباط برقرار کنم، نیاز دارم تا نام دیگر مدیران دیگر شعبه ها را بدانم.'

شما بایستی سوالات مشابهی را در مورد داده های مهم ذخیره شده در پایگاه داده بپرسید. پاسخ به این سوالات باید به شما کمک کند که جزئیات لازم برای خصوصیات نیازمندی های کاربر را شناسایی کنید.

جمع آوری اطلاعات نیازمندیهای سیستم کاربرد پایگاه داده StayHome

زمانیکه مصاحبه را درباره دیدهای کاربر پیش می برید، بهمراه آن باید اطلاعات کلی بیشتری درباره نیازمندی های سیستم جمع آوری کنید. نمونه هایی از انواع سوالاتی که باید درباره سیستم بپرسید شامل موارد زیر است :

'چه تراکنشهایی مکررا در پایگاه داده اجرا می شود؟'
'چه تراکنشهایی در نحوه انجام کارتان حیاتی است؟'
' تراکنشهای حیاتی چه موقع اجرا می شود؟ '
'دوره های حجم کاری پایین، متوسط، و زیاد در تراکنشهای حیاتی چه زمانیست؟'
'چه نوعی از امنیت را برای کاربرد پایگاه داده می خواهید؟'
'آیا داده به شدت حساسی دارید که می خواهد فقط عضو خاصی از پرسنل به آن دسترسی داشته باشد؟'
'چه داده تاریخی را میخواهید نگه دارید؟'
'نیازمندیهای دسترسی مشترک و شبکه ای برای سیستم پایگاه داده چیست؟'
'چه نوعی از حفاظت در برابر خرابی و از دست دادن داده را برای پایگاه داده تان می خواهید؟'

برای مثال، ممکن است از مدیر سوالات زیر را بپرسید:

توسعه دهنده پایگاه داده 'چه تراکنشهایی مکررا در پایگاه داده اجرا می شود؟'

مدیر ′ما درخواستهایی را مکررا چه از طریق تلفن یا با عضوهایی که به شعبه ما می آیند تا فیلمی را جستجو کرده و ببینند که آیا برای اجاره موجود است، دریافت می کنیم. البته، همچنین فیلم هایی را که به شعبه برمی گردانند نیز بازپس می گیریم′

توسعه دهنده پایگاه داده 'چه تراکنشهایی در نحوه انجام کارتان حیاتی است؟'

مدیر ′تراکنشهای حیاتی باید بتواند جستجو ی فیلمی مشخص و همچنین بازپس گیری آنها را انجام دهد. که اگر این نوع تراکنشها را در نظر نگیریم اعضاء ما به جای دیگری خواهند رفت′

توسعه دهنده پایگاه داده ' تراکنشهای حیاتی چه موقع اجرا می شود؟ '

مدیر 'هر روز.'

توسعه دهنده پایگاه داده 'دوره های حجم کاری کم، متوسط، و زیاد در تراکنشهای حیاتی چه زمانهایی است؟ '

مدیر ′ما هنگام صبح آرام هستیم و در طول روز سرمان شلوغ می شود. شلوغ ترین زمان هر روز بین ساعت ۶ تا ۹ عصر است. ما حتی می خواهیم که پرسنل را در ماموریت خود در روزهای جمعه ها و شنبه ها دو برابر کنیم′

مدیریت دیدهای کاربر برنامه پایگاه داده StayHome

شما چه تصمیمی را جهت استفاده از روش های متمرکز سازی دید یا یکپارچگی دید جهت مدیریت دیدهایی که دارای چندین کاربر باشند گرفته اید؟ یک راه برای کمک به شما این است که خودتان تصمیم بگیرید همپوشانی را به معنی داده های استفاده شده بین دیدهای کاربری که طی مرحله تعریف سیستم شناسایی کرده اید در نظر بگیرید. جدول ۷-۴ ارجاع متقابل میان رئیس، مدیر، سرپرست، معاون، و خریدار با انواع اصلی داده های استفاده شده توسط پایگاه داده *StayHome* (بنامهای تامین کننده، اجاره فیلم، سفارش فیلم، فیلم، شعبه، پرسنل، اجاره ای، و عضو) را نشان می دهد.

جدول۷-۴ ارجاع متقابل دیدهای کاربر با داده های اصلی استفاده شده در پایگاه داده *StayHome*

	Supplier	Video Order	Video	Branch	Staff	Rentals	Member
Director	X	X	X	X	X	X	X
Manager			X	X	X	X	X
Supervisor			X	X	X	X	X
Assistant			X	X		X	X
Buyer	X	X	X	X		X	X

در این جدول مشاهده می کنید که میان داده های استفاده شده تمامی دیدهای کاربر همپوشانی وجود دارد. با این وجود، دید های رئیس و خریدار در مورد نیاز به داده اضافی که توسط دیگر دیدها استفاده شده (بنامهای، تامین کننده و سفارش فیلم) متمایز هستند. بر اساس این تحلیل، شما ابتدا باید از روش متمرکز سازی استفاده کرده تا نیازمندی های دیدهای کاربر رئیس و خریدار(با دادن نام کلی دید *تجاری*) و نیازمندیها برای دیدهای کاربر مدیر، سرپرست، و معاون (با دادن نام کلی دید *شعبه*) را ترکیب کنید. سپس می توانید مدلهای داده ای را توسعه بدهید که نمایش دهنده دیدهای حرفه و شعبه می باشند و بعد از آن با استفاده از روش ترکیب دید دو مدل داده ای را ادغام کنید.

البته، برای تمرین ساده ای همچون ***StayHome***، بسادگی می توانید از روش متمرکزسازی برای همه دیدهای کاربر استفاده کنید اما ما روی تصمیم خود مانده و دو دید را ایجاد می کنیم. بنابراین می توانیم نشان دهیم که چگونه روش جامعیت دید در تمرین فصل ۱۰ کار می کند.

مشکل است که قواعد جامعی را درباره اینکه از کدام روش ترکیب استفاده باید کنیم ارائه دهیم. به عنوان توسعه دهنده پایگاه داده، شما باید بر اساس ارزیابی تان از پیچیدگی پایگاه داده و درجه همپوشانی بین دیدهای مختلف کاربر تصمیم بگیرید. با این وجود، اینکه شما از روش متمرکز سازی یا روش ترکیب دید استفاده کنید و یا از ترکیب این دو روش برای ساختن زیربنای پایگاه داده استفاده کنید، نهایتا نیاز خواهید داشت دیدهای اصلی کاربر را برای پایگاه داده اصلی ایجاد کنید. در فصل ۱۵ نحوه ایجاد دیدهای کاربر برای پایگاه داده را بررسی خواهیم کرد.

در بقیه این فصل، خصوصیات نیازمندی های کاربر را برای دید شعبه ارائه می کنیم و دید تجاری را نیز در فصل ۱۰ بحث خواهیم کرد.

ایجاد خصوصیات نیازمندیهای کاربر برای دید شعبه برنامه کاربرد پایگاه داده StayHome

خصوصیات نیازمندیهای کاربر برای دید شعبه به دو بخش شده است:

- بخش ' نیازمندیهای داده ' داده های استفاده شده توسط دید شعبه را توصیف می کند.
- بخش ' نیازمندیهای تراکنش' نمونه هایی از اینکه داده چگونه توسط دید شعبه مورد استفاده قرار می گیرد (یعنی، تراکنشهایی که پرسنل میخواهند روی داده انجام دهند) را فراهم میکند.

نیازمندیهای داده

داده هایی که درباره شعبه ای از ***StayHome*** نگه داشته می شود آدرس شعبه متشکل از خیابان، شهر، منطقه، و کدپستی، و شماره های تلفن (حداکثر سه خط) می باشد. به هر شعبه شماره شعبه ای داده شده، که در سرتاسر شرکت منحصر است.

هر شعبه شرکت پرسنلی دارد، که شامل مدیر، یک یا چند سرپرست، و تعدادی از دیگر پرسنل می باشد. مدیر، مسئول کارکرد روزانه شعبه مذکور است. هر شعبه چندین سرپرست دارد و هر سرپرست مسئول نظارت بر گروهی از پرسنل است. داده های نگه داشته شده درباره عضوی از پرسنل نام او، موقعیت شغلی، و حقوق است. هر عضو پرسنل شماره پرسنلی دارد، که در سراسر شرکت منحصر است.

به هر شعبه ***StayHome*** انباری از فیلمها اختصاص داده شده است. داده هایی که درباره فیلم نگه داشته می شود عبارتند از شماره فهرست، شماره فیلم، عنوان، فهرست، دسته، نرخ اجاره روزانه، قیمت خرید، وضعیت، و نام بازیگر اصلی (و هنرپیشه ای که در فیلم درخشیده است) و کارگردان فیلم می باشد. شماره فهرست بطور مجزا هر فیلم را مشخص می کند. در بیشتر موارد، بیش از یک کپی از هر فیلم در هر شعبه موجود است، که هر کپی منحصرا توسط شماره فیلم شناسایی می شود. به هر

فیلم فهرستی همچون حادثه ای، کودکان، بزرگسالان، هیجان انگیز، ترسناک، علمی- تخیلی داده شده است. وضعیت بیانگر این است که آیا کپی بخصوصی برای اجاره در دسترس است.

مشتری قبل از اجاره فیلم از شرکت، اول باید بعنوان عضو محلی شعبه ***StayHome*** ثبت نام کند. داده هایی که درباره عضو نگه داشته می شود نام و نام خانوادگی او، آدرس، و تاریخی که در شرکت عضو شده است می باشد. هر عضو شماره عضویت دارد، که او را در تمام شعبه ها منحصر می کند و وقتیکه عضو در بیش از یک شعبه ثبت نام کند استفاده می شود. نام عضوی از پرسنل که مسئول ثبت نام از اعضا در شعبه است نیز مورد توجه قرار می گیرد.

بعد از ثبت نام، عضو آزاد است که فیلم هایی اجاره کند، که حداکثر در هر بار مراجعه ۱۰ فیلم می تواند اجاره کند. داده هایی که درباره هر فیلم اجاره داده شده نگه داشته می شود شماره اجاره، نام کامل عضو و شماره عضویت، شماره فیلم، عنوان، هزینه اجاره روزانه، و تاریخی که فیلم برگردانده شده است می باشد. شماره اجاره در سرتاسر شرکت منحصر است.

نیازمندیهای تراکنش

ثبت داده

(a) جزئیات شعبه جدید را وارد کنید.

(b) جزئیات عضو جدید پرسنل شرکت را وارد کنید. (همچون کارمند Tom Daniels در شعبه B001).

(c) جزئیات فیلم تازه منتشر شده را وارد کنید. (همچون جزئیات فیلم با نام ***Independence Day***).

(d) جزئیات کپی های فیلم جدید را در شعبه معینی واردکنید (همچون سه کپی از ***Independence Day*** در شعبه B001).

(e) جزئیات عضو تازه ثبت نام کرده را در شعبه معینی وارد کنید (همچون عضو Bob Adams که در شعبه B002 ثبت نام کرده است).

(f) جزئیات قرارداد اجاره برای عضوی که فیلم اجاره کرده است را وارد کنید (همچون عضو Don Nelson فیلم ***Tomorrow Never Dies*** را در تاریخ ۴-Feb-۲۰۰۷ اجاره کرده است).

بهنگام سازی/ حذف جزئیات داده

(g) جزئیات شعبه را بهنگام/ حذف کنید.

(h) جزئیات عضو پرسنل شعبه را بهنگام/ حذف کنید .

(i) جزئیات فیلم معینی را بهنگام/ حذف کنید.

(j) جزئیات کپی فیلم را بهنگام/ حذف کنید.

(k) جزئیات عضو معینی را بهنگام/ حذف کنید.

(l) جزئیات قرارداد اجاره معینی برای عضوی که فیلم را اجاره کرده است را بهنگام/ حذف کنید.

پرس و جوی داده

پایگاه داده بایستی قادر باشد تا از پرس وجو های زیر پشتیبانی کند :

(m) جزئیات شعبه ها در شهرهای معینی را لیست کنید.

(n) نام، موقعیت شغلی، و حقوق پرسنل در شعبه معینی را، که با نام پرسنلی مرتب شده است لیست کنید.

(o) نام مدیران هر شعبه را، که با شماره شعبه مرتب شده است لیست کنید.

(p) عنوان، فهرست، دردسترس بودن همه فیلمها برای نام بازیگر معینی در شعبه بخصوصی، که با عنوان مرتب شده است را لیست کنید.

(q) عنوان، فهرست، دردسترس بودن تمامی فیلمها برای نام هنرپیشه معینی در شعبه بخصوصی، که با فهرست مرتب شده است را لیست کنید.

(r) عنوان، فهرست، در دسترس بودن تمامی فیلمها برای نام کارگردان معینی در شعبه بخصوصی، که با عنوان مرتب شده است را لیست کنید.

(s) جزئیات تمام فیلمهایی که عضو بخصوصی اجاره کرده است را لیست کنید.

(t) جزئیات کپی های فیلم معینی در شعبه بخصوصی را لیست کنید.

(u) عناوین تمامی فیلمهای موجود در دسته خاصی، مرتب شده با عنوان را لیست کنید.

(v) مجموع تعداد فیلمهای موجود در هر دسته فیلم هر شعبه را لیست کنید.

(w) هزینه تمام فیلمهای کل شعبه ها را لیست کنید.

(x) مجموع تعداد فیلمهای هر هنرپیشه، که با نام هنرپیشه مرتب شده است را لیست کنید.

(y) مجموع تعداد اعضاء هر شعبه که در ۲۰۰۶ ملحق شده اند را، که با شماره شعبه مرتب شده است را لیست کنید.

(z) مجموع تعداد اجاره های ممکن روزانه برای فیلمها در هر شعبه، که با شماره شعبه مرتب شده است را لیست کنید.

ایجاد خصوصیات سیستم برای برنامه پایگاه داده ***StayHome***

خصوصیات سیستم بایستی تمام جوانب مهم کاربرد پایگاه داده ***StayHome*** را لیست کند. نمونه هایی از مواردی که باید در خصوصیات سیستم ذکر شوند بدین صورت است:

- سایز اولیه پایگاه داده
- نرخ رشد پایگاه داده
- انواع و متوسط تعدادی از رکوردها که جستجو میشوند
- نیازمندیهای دستیابی مشترک و شبکه ای
- کارایی
- امنیت
- پشتیبان گیری و ترمیم
- انتشار قانونی.

سایز اولیه پایگاه داده

(a) تقریبا ۲۰۰۰۰ عنوان فیلم و ۴۰۰۰۰ فیلم برای اجاره در ۱۰۰ شعبه وجود دارد. و به طور متوسط ۴۰۰۰ و کلا ۱۰۰۰۰ فیلم برای اجاره در هر شعبه موجود است.

(b) تقریبا ۲۰۰۰ پرسنل در تمام شعبه ها کار می کنند. و تعداد متوسط ۱۵ تا حداکثر ۲۵ پرسنل در هر شعبه کار می کنند.

(c) تقریبا ۱۰۰۰۰۰ عضو در سراسر شعبه ها وجود دارند که ثبت نام کرده اند. به طور متوسط ۱۰۰۰ تا حداکثر ۱۵۰۰ عضو در هر شعبه ثبت نام کرده اند.

(d) تقریبا ۴۰۰۰۰۰ فیلم اجاره ای در تمام شعبه ها موجود است. و حدودا ۴۰۰۰ و حداکثر ۱۰۰۰۰ فیلم اجاره ای در هر شعبه وجود دارد.

(e) تقریبا ۱۰۰۰ کارگردان و ۳۰۰۰۰ بازیگر اصلی در ۶۰۰۰۰ نقشهای برجسته وجود دارد.

(f) تقریبا ۵۰ تهیه کننده فیلم و ۱۰۰۰ سفارش دهنده فیلم وجود دارد.

میزان رشد پایگاه داده

(a) تقریبا ۱۰۰ عنوان فیلم جدید و ۲۰ کپی از هر یک هر ماه به پایگاه داده افزوده می شوند.

(b) هر بار که کپی پس داده شده فیلم قابل استفاده نبود (که شامل آنهاییکه کیفیت تصویری کم، گم شده، یا دزدیده شده هستند)، رکورد متناظر آن از پایگاه داده حذف می شود. تقریبا ۱۰۰ رکورد فیلمهای اجاره ای هر ماهه از پایگاه داده حذف می شوند.

(c) تقریبا بتعداد ۲۰ عضو پرسنل هر ماه به/ از شرکت ملحق یا ترک می شوند. رکورد پرسنلی که شرکت را ترک کرده اند بعد از یک سال از پایگاه داده حذف می شوند. تقریبا ۲۰ رکورد پرسنلی هر ماهه از پایگاه داده حذف می شوند.

(d) تقریبا ۱۰۰۰ عضو جدید هر ماهه در شعبه ها ثبت نام می کنند. اگر عضوی فیلمی را در مدت ۲ سال تحویل ندهد رکورد او حذف می شود. تقریبا بتعداد ۱۰۰ رکورد هر ماه حذف می شود.

(e) تقریبا ۵۰۰۰ فیلم اجاره ای جدید هر روزه در سراسر ۱۰۰ شعبه ثبت می شود. جزئیات فیلمهای اجاره ای بعد از دو سال از تاریخ ایجاد حذف می شوند.

(f) تقریبا ۵۰ سفارش جدید فیلم هر هفته جای داده شده است. جزئیات سفارشات فیلم دو سال بعد از تاریخ ایجاد رکورد حذف می شود.

انواع و متوسط تعدادی از رکوردها که جستجو می شوند

(a) جستجو برای جزئیات شعبه- تقریبا در هر روز ۱۰ بار.

(b) جستجو برای جزئیات عضو پرسنل شعبه- تقریبا در هر روز ۲۰ بار.

(c) جستجو برای جزئیات فیلم معینی- تقریبا هر روز ۵۰۰۰ (یکشنبه تا پنجشنبه)، تقریبا در هر روز ۱۰۰۰۰ بار (جمعه و شنبه) حجم کاری نهایی ۹-۶ عصر روزانه .

(d) جستجو برای جزئیات کپی فیلم- تقریبا ۱۰۰۰۰ هر روز (یکشنبه تا پنجشنبه)، تقریبا ۲۰۰۰۰ هر روز (جمعه و شنبه) اوج حجم کاری ۹-۶ عصر روزانه.

(e) جستجو برای جزئیات عضو مشخصی- تقریبا ۱۰۰ بار در هر روز.

(f) جستجو برای جزئیات قرارداد اجاره ای عضوی که فیلم اجاره میکند- تقریبا ۱۰۰۰۰ بار در هر روز (یکشنبه تا پنجشنبه)، تقریبا ۲۰۰۰۰ هر روز (جمعه و شنبه) اوج حجم کاری ۹-۶ عصر هر روز.

نیازمندیهای دستیابی مشترک و شبکه ای

(a) تمام شعبه ها بایستی به طور امن به پایگاه داده مرکزی در مرکز عملیات شرکت به شبکه وصل شوند.

(b) سیستم بایستی به حداقل سه نفر در هر شعبه اجازه دهد تا به طور همزمان به آن دسترسی داشته باشند. مراعاتی نیاز است تا مجوز لازم برای این تعداد دسترسی همزمان فراهم شود.

کارایی

(a) طی ساعات کاری اما نه در دوره های اوج ۱ثانیه و کمتر از آن برای جستجوی تک رکورد زمان باید لحاظ شود. طی ساعات اوج کاری (۹-۶ روزانه) که کمتر از ۵ ثانیه زمان لحاظ باید شود.

(b) طی ساعات کاری اما نه در دوره های اوج ۵ ثانیه و کمتر از آن برای هر جستجو چند رکورده زمان باید لحاظ شود. طی ساعات اوج کاری (۹-۶ روزانه) که کمتر از ۱۰ ثانیه زمان لحاظ باید شود.

(c) طی ساعات کاری اما نه در دوره های اوج ۱ثانیه و کمتر از آن برای هر ذخیره/ بهنگام سازی رکورد زمان باید لحاظ شود. طی ساعات اوج کاری (۹-۶ روزانه) که کمتر از ۵ ثانیه زمان لحاظ باید شود.

امنیت

(a) پایگاه داده بایستی توسط رمز حفاظت شود.

(b) به هر عضو پرسنل باید امتیاز مناسبی برای دید کاربر ویژه همچون رئیس، مدیر، سرپرست، معاون، و خریدار اختصاص داده شود.

(c) پرسنل تنها باید به داده هایی که مربوط به آنها است دسترسی داشته باشند.

پشتیبان گیری و ترمیم

پایگاه داده باید هر روز ساعت ۱۲ شب پشتیبان گیری شود.

انتشار قانونی

هر کشوری قانون خود را دارد که معین می کند ذخیره سازی کامپیوتری داده های شخصی چگونه انجام شود. از آنجائیکه پایگاه داده *StayHome* داده های پرسنل و عضوها را نگه می دارد، هر گونه نشر قانونی که موافقت شده بایستی بررسی و پیاده سازی شود.

٥-٤-٤ بررسی موردی *StayHome* – طراحی پایگاه داده

در این بخش، ایجاد خصوصیات نیازمندیهای کاربر را برای دید شعبه و خصوصیات سیستم برای کاربرد پایگاه داده *StayHome* را نشان دادیم. این مستندات منبع اطلاعاتی هستند که در مرحله بعد چرخه حیات که **طراحی پایگاه داده** نام دارد از آن استفاده خواهیم کرد. در فصل ۸ تا ۱۶، متدلوژی گام به گامی برای طراحی پایگاه داده ارائه می کنیم، و از بررسی موردی *StayHome* استفاده خواهیم کرد و مستندات پایگاه داده *StayHome* را که در این بخش ایجاد شد، در فصل بعد در عمل نشان داده خواهند شد.

خلاصه فصل

- ✓ **حقیقت یابی** مرحله رسمی استفاده از تکنیکهایی همچون مصاحبه و پرسش بوده که درباره جمع آوری حقایق درباره سیستم، نیازمندیها، و اولویتها بکار میرود.
- ✓ حقیقت یابی به طور ویژه ای در مراحل ابتدایی چرخه حیات برنامه پایگاه داده که شامل برنامه ریزی پایگاه داده، تعریف سیستم، و مراحل تحلیل و جمع آوری نیازمندیها است بسیار مهم می باشد.
- ✓ پنج تکنیک مهم حقیقت یابی عبارتند از بررسی مستندات، مصاحبه، مشاهده حرفه در عمل، تحقیق، و پرسش .
- ✓ مرحله اول در توسعه کاربرد پایگاه داده به طور مشخص تعریف **طرح اجمالی ماموریت** و **اهداف ماموریت** پروژه پایگاه داده می باشد. طرح اجمالی ماموریت اهداف عمده کاربرد پایگاه داده را نشان می دهد. اهداف ماموریت بایستی وظیفه مشخصی را که پایگاه داده پشتیبانی کند را شناسایی کند.
- ✓ هدف مرحله تعریف سیستم، تعریف محدوده و دیدهای کاربر کاربرد پایگاه داده است.
- ✓ دید کاربر بیانگر نیازمندیهایی است که باید توسط کاربرد پایگاه داده تعریف شده با نقش وظیفه مشخصی (همچون مدیر و معاون) یا ناحیه کاربرد تجاری (همچون اجاره فیلم و کنترل انبار) پشتیبانی شود.
- ✓ دو مستند اصلی ایجاد شده طی مرحله تحلیل و جمع آوری نیازمندیها، بنامهای مشخصات نیازمندیها و مشخصات سیستم می باشد.
- ✓ مشخصات نیازمندیهای کاربر بهمراه جزئیات، داده نگه داشته شده در پایگاه داده و نحوه استفاده از آن را شرح می دهد.
- ✓ مشخصات سیستم هر ترکیبی که شامل کاربرد پایگاه داده از جمله کارایی مورد نیاز و سطح امنیت باشد را توصیف می کند.

فصل ۵

مدل سازی موجودیت- رابطه ای

آنچه در این فصل خواهید آموخت :

- چگونه از مدلسازی ER در طراحی پایگاه داده استفاده کنیم.
- مفاهیم اساسی مدل ER، که موجودیتها، رابطه ها، و صفتها نام دارند.
- تکنیک نموداری جهت نمایش مدل ER.
- چگونه مشکلاتی را که تله های ارتباطی نام دارند و در مدل ER رخ می دهند را شناسایی و برطرف کنیم.

در فصل ۴، درباره تکنیکهای جمع آوری و ضبط اطلاعات درباره آنچه که کاربران پایگاه داده نیاز دارند مطالبی آموختید. بعد از تکمیل مرحله تحلیل و جمع آوری نیازمندیهای چرخه حیات پایگاه داده، شما نیازمندی های کاربرد پایگاه داده را مستند کردید، حالا آماده هستید تا طراحی پایگاه داده را شروع کنید.

یکی از مشکل ترین جنبه های طراحی پایگاه داده این است که طراحان، برنامه نویسان، و کاربران نهایی میل دارند داده ها را ببینند و از آنها به طرق مختلف استفاده کنند. متاسفانه جز اینکه ما تنها می توانیم فهم کلی از نحوه عمل شرکت داشته باشیم، طراحی تولید شده توسط ما در تامین نیازمندیهای کاربر شکست خواهد خورد. برای اطمینان از اینکه درک جامع و دقیقی از ماهیت داده ها داشته باشیم و اینکه این داده ها چگونه توسط شرکت مورد استفاده قرار می گیرند، جهت ارتباط به مدلی غیر فنی و واضح نیاز داریم. مدل **ارتباط- موجودیت**[۱] این چنین مدلی است. مدل ER در سال ۱۹۷۶ معرفی شد، این مدل اکنون گسترش یافته تا در برگیرنده مفاهیم پیشرفته مدلسازی نیز باشد. در این فصل مفاهیم اساسی ER را بحث کرده و بعضی از جوانب پیشرفته متداول را نیز در فصل ۱۱ معرفی می کنیم.

مدل سازی ارتباط – موجودیت یک روش بالا به پایین جهت طراحی پایگاه داده است. مدل سازی ER را با شناسایی داده های مهم (که موجودیت نامیده می شوند) و ارتباط بین داده ها که بایستی در مدل نشان داده شوند شروع می کنیم. سپس جزئیات بیشتری از آنچه می خواهیم درباره موجودیتها و رابطه ها (صفات نامیده می شوند) و هرگونه محدودیت روی موجودیتها، ارتباطات، و صفات داشته باشیم به مدل اضافه می کنیم.

در تمام این فصل، شما با مفاهیم پایه جهت ساختن مدل ER آشنا می شوید. اگر چه توافق کلی درباره معنای این مفاهیم وجود دارد، چندین راه دیگری نیز وجود دارد که شما می توانید هر مفهوم را به صورت نمودار نشان دهید. ما شیوه نمایش نموداری را انتخاب کرده ایم که به طور فزاینده ای در زبان مدل سازی شی گرای UML[۲] (زبان مدل سازی مجتمع) مورد استفاده قرار می گیرد. نمونه هایی از نمایش مدل ER در ضمیمه (الف) آمده است.

---

[۱] Entity-Relationship
[۲] Unified Modeling Languages

UML جانشینی برای تعدادی از روشهای طراحی و تحلیل شی گرای سال ۱۹۸۰ و ۱۹۹۰ است. گروه مدیریت شی (OMG) درحال حاضر بدنبال استاندارد سازی UML هستند، و پیش بینی کرده اند که UML در آینده نزدیک زبان استاندارد مدل سازی برای سیستمهای شی گرا خواهد بود.

از آنجاییکه مدل ER پایه متدلوژی را که در فصلهای ۷ تا ۱۶ ارائه خواهیم کرد تشکیل می دهد، اثبات شده است که فصول مذکور یکی از مهم ترین فصلهای این کتاب هستند. اگر مفاهیم را فورا درک نکردید، خیلی نگران نشوید. سعی کنید که فصل را دوباره خوانده، و جهت راهنمایی بیشتر، به نمونه هایی که در متدلوژی ارائه کرده ایم رجوع کنید. اجازه دهید بحث مان را با مفاهیم پایه مدل ER که، موجودیتها، روابط، و صفتها هستند شروع کنیم.

۵-۱ موجودیتها

مفهوم پایه مدل ER **موجودیت** است، که مجموعه شی هایی را که واقع 'در جهان واقعی' بوده و مشخصات مشابهی دارند را نشان میدهد. هر شی، که بایستی در مجموعه ای منحصرا قابل شناسایی باشد، **رخداد موجودیت** نام دارد. موجودیت وجود مستقلی داشته و میتواند مجموعه اشیایی با وجود فیزیکی (واقعی) یا مجموعه اشیایی با وجود مفهومی (انتزاعی) را که در شکل ۵-۱ نشان داده شده است را نمایش دهد.

ما هر موجودیت را با نام مجزا و لیستی از خصوصیات، که **صفات** نامیده میشوند شناسایی می کنیم. اگرچه، موجودیت مجموعه مجزایی از صفات را داراست، هر موجودیت مقدار صفت خاص خود را دارد. پایگاه داده عموما شامل چندین موجودیت مختلف است.

شکل ۵-۱

نمونه های موجودیت با وجود فیزیکی و مفهومی.

وجود فیزیکی	وجود مفهومی
Member	Role
Video	Rental
Branch	Registration

نمایش هندسی موجودیتها

هر موجودیت با مستطیل با برچسب نام موجودیت، که به طور معمول اسم مفردی است نشان داده می شود. در UML، اولین حرف هرکلمه نام موجودیت با حرف بزرگ نوشته می شود (برای نمونه,Video , Role, Actor, VideoForRent). شکل ۵-۲ نمایش هندسی موجودیتهای Video، Role، وActor را نشان می دهد.

شکل۵-۲

نمایش هندسی موجودیتهای Role ,Video, Actor.

۵-۲ رابطه ها

رابطه مجموعه ارتباطات مرتبط بین موجودیتهای شرکت کننده است. مانند موجودیتها، هر ارتباط در مجموعه بایستی به طور یکتا قابل شناسایی باشد. ارتباطی که منحصرا قابل شناسایی باشد **رخداد ارتباط**[۱] نامیده میشود.

هر ارتباط نامی داردکه عملش را توصیف می کند. برای نمونه، موجودیت Actor که با موجودیت Role رابطه دارد از طریق ارتباطی با نام *Plays* بهم مربوط شده اند، و موجودیت Role با موجودیت Video از طریق رابطه ای بنام *Features* بهم مربوط شده اند.

نمایش هندسی رابطه ها

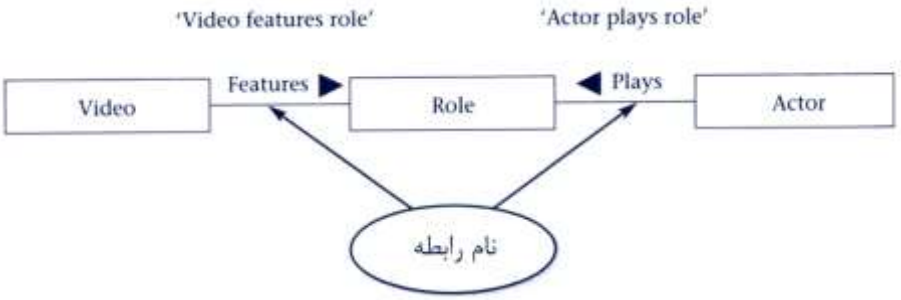
هر رابطه با خطی که موجودیت های شرکت کننده را به هم وصل می کند و با نام رابطه برچسب گذاری می شود نشان داده می شود. عموما، رابطه با استفاده از برچسب فعل (مثل، *Plays* یا *Features*) یا عبارت کوتاه شامل فعل (مثل، *IsPartOf* یا *WorksAt*) نامگذاری می شود. بار دیگر، در اینجا نیز حرف اول هر کلمه در نام روابط با حرف بزرگ نشان داده می شود. در صورت امکان، نام رابطه بایستی در مدل ER داده شده منحصر بفرد باشد.

رابطه فقط در یک جهت برچسب گذاری می شود، که معمولا بدین معنی است که رابطه فقط در یک جهت قابل فهم است (برای نمونه Actor *Plays* Role معنی دارد ولی Actor *Plays* Role بی معنی است).

بنابراین وقتیکه نام رابطه انتخاب شد، علامت پیکانی درجلوی نام گذاشته می شود تا جهت درست را به خواننده نشان دهد تا نام رابطه ها را درست تفسیر کند (برای نمونه، Actor *Plays* ◀ Role). شکل ۳-۵ نمایش هندسی ارتباط Video Role *Features* و Actor *Plays* Role، را نشان می دهد.

شکل ۳-۵

نمایش نموداری رابطه های Video *Features* Role و Actor *Plays* Role .



۵-۲-۱ درجه ارتباط

به موجودیتهای درگیر در رابطه خاصی بعنوان موجودیت شرکت کننده رجوع می شود. تعداد موجودیتهای شرکت کننده در یک رابطه **درجه** نامیده می شود. رابطه درجه دو رابطه **باینری** نام دارد. دو رابطه نشان داده شده در شکل ۳-۵ **رابطه های باینری** هستند.

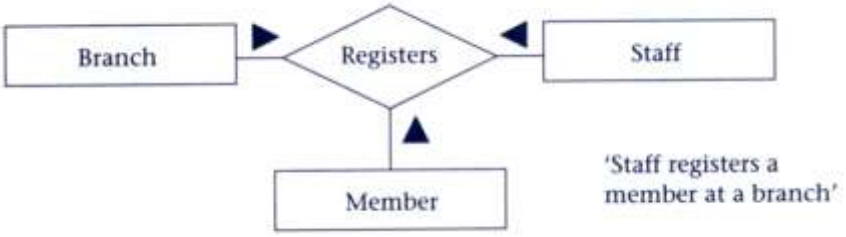
رابطه درجه سه، **سه گانه**[۲] نامیده می شود. نمونه ای از رابطه سه گانه، *Registers* با سه موجودیت شرکت کننده بنامهای , Branch ، Staff، وMember هستند که در شکل ۴-۵ نشان داده شده اند. هدف این رابطه نشان دادن موقعیتی است که عضوی از پرسنل، عضوی را در شعبه مشخصی ثبت نام می کند، و اجازه می دهد تا اعضاء در بیش از یک شعبه ثبت نام کنند، و برای اعضاء پرسنل که بین شعبه ها تغییر مکان کنند.

---

[۱] Relationship Occurrence

[۲] Ternary

رابطه درجه چهار، چهارگانه نام دارد، و رابطه با درجه بالاتر رابطه n- گانه نامیده می شود. رابطه های با درجه بالاتر از باینری رابطه **پیچیده** (complex) نام دارد. عمده ترین نوع رابطه ای که با آن برخورد می کنید رابطه باینری است، اما بعضی اوقات نیز با روابط سه گانه و بعضا چهارگانه نیز برخورد خواهید کرد.



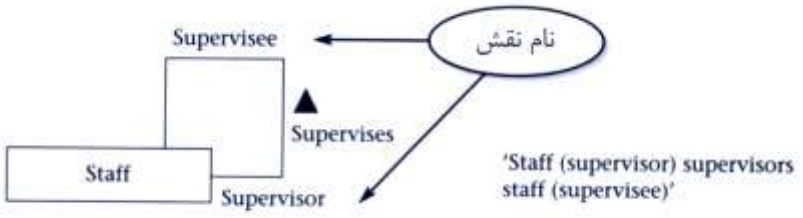
شکل ۴-۵

نمونه رابطه سه گانه با نام *Registers* .

۵-۲-۲ رابطه های بازگشتی

اجازه بدهید رابطه ای که ***Supervises*** نامیده می شود را در نظر بگیریم، که رابطه پرسنل با ناظر وقتی که ناظر در عین حال عضوی از پرسنل باشد را نشان می دهد. بعبارت دیگر، موجودیت پرسنل دو بار در رابطه ***Supervises*** شرکت کرده است : اولین شرکت وقتی است که ناظر باشد، و دومین شرکت وقتی است که بعنوان عضوی از پرسنل در نظر گرفته شود و بعنوان کسی که نظارت می شود (Supervisee)، که در شکل ۵-۵ نشان داده شده است. رابطه های بازگشتی بعضی اوقات رابطه های *یگانی*[1] نیز نامیده می شوند.

به رابطه ها شاید نامهای نقش داده شده باشد که هدف هر موجودیت شرکت کننده که در رابطه بازی می کند را نشان می دهد. نامهای نقش رابطه های بازگشتی برای تعیین عمل هر موجودیت شرکت کننده مهم هستند. شکل ۵-۵ استفاده از نامهای نقش برای توصیف رابطه بازگشتی ***Supervisee*** را نشان می دهد. اولین شرکت موجودیت پرسنل در رابطه ***Supervises*** داده شده با نام نقش Supervisor و دومین شرکت با نام داده شده Supervisee را نشان می دهد.



شکل ۵-۵

نمونه رابطه بازگشتی با نام Supervises.

۵-۳ صفتها

خصوصیات خاص موجودیتها صفات نامیده می شوند. صفتها هر آنچه را که ما می خواهیم درباره موجودیتها بدانیم را نشان می دهند. برای نمونه، موجودیت فیلم شاید با صفتهای catalogNo، title، category ، dailyRental ، و price نشان داده شود. این صفتها مقادیری را نگه می دارد که رخداد هر فیلم را شرح می دهد، و منبع اصلی داده های ذخیره شده در پایگاه داده را ارائه می کند.

[1] Unary

رابطه میان موجودیتها همچنین می تواند صفتهایی مشابه با آن موجودیتها داشته باشد، اما ما بحث رابطه های دارای صفت را تا بخش۶-۵ به تعویق خواهیم انداخت.

همانگونه که هم اکنون بحث می کنیم، می توانیم صفتها را به صورت زیر درجه بندی کنیم : ساده یا مرکب ; تک مقداره یا چند مقداره; یا مشتق.

۵-۳-۱ صفتهای ساده و ترکیبی

صفتهای ساده نمی توانند مجددا تقسیم بندی شوند. نمونه هایی از صفتهای ساده شامل صفتهای category و price برای فیلم است. صفات ساده بعضی اوقات *صفات تجزیه ناپذیر*[1] نامیده می شود.

صفات ترکیبی می تواند مجددا جهت رسیدن به اجزا کوچکتری با وجود مستقل تقسیم شوند. برای نمونه، صفت نام موجودیت عضوی با مقدار 'Don Nelson' می تواند مجددا به نام ('Don') و نام خانوادگی ('Nelson') تقسیم شود.

تصمیم گیری درباره اینکه صفت **نام** را به صورت صفت ساده یا صفت ترکیبی **نام** و **نام خانوادگی** مدل کنیم بستگی به این دارد که آیا تراکنش های کاربر به صفت **نام** به صورت واحد منفرد یا اجزا منحصر بفرد دسترسی دارد.

۵-۳-۲ صفات تک مقداره و چند مقداره

بیشتر صفتهای موجودیت بصورت **تک مقداره** هستند. برای نمونه، هر رخداد موجودیت فیلم یک مقدار صفت catalogNo دارد (برای نمونه، ۲۰۷۱۳۲)، بنابراین به صفت catalogNo بعنوان تک مقداره رجوع می شود.

بعضی صفتهای موجودیت خاصی چندین مقدار دارند. برای نمونه، هر رخداد موجودیت فیلم شاید برای صفت category چندین مقدار داشته باشد (برای نمونه، 'کودکان' و 'کمدی')، و بنابراین صفت category در این مورد **چندین مقداره** خواهد بود. صفت چند مقداره شاید مجموعه ای از مقادیر با حد بالا و پایین داشته باشد. برای نمونه، صفتcategory شاید بین یک تا سه مقدار داشته باشد.

رده بندی ساده یا ترکیبی، و رده بندی تک مقداره و چند مقداره، دو به دو ناسازگار نیستند. به عبارت دیگر، شما می توانید صفتهای تک مقداره ساده، تک مقداره ترکیبی، چند مقداره ساده، و چند مقداره ترکیبی داشته باشید.

۵-۳-۳ صفات مشتق

بعضی صفات شاید به موجودیت خاصی مربوط باشند. برای نمونه، سن عضوی از پرسنل (age) قابل اشتقاق از صفت تاریخ تولد (DOB) می باشد، و بنابراین صفات سن و تاریخ تولد بهم مربوط هستند. صفت سن را صفت مشتق شده می گوییم، که مقدار صفت از صفت تاریخ تولد مشتق شده است.

در بعضی موارد، مقدار صفت از مقادیر موجودیت واحد مثل سن، مشتق می شود. اما در دیگر موارد، مقدار صفت احتمالا از مقادیر بیش از یک موجودیت مشتق شود.

---

[1] Atomic attribute

سن معمولا در پایگاه داده ذخیره نمی شود چون به طور مرتب بهنگام می شود. در طرف دیگر، از آنجائیکه تاریخ تولد هرگز تغییر نمی کند و سن می تواند از تاریخ تولد مشتق شود، تاریخ تولد به جای سن ذخیره می شود، و سن در صورت نیاز از صفت **DOB** مشتق می شود.

۴-۳-۵ کلیدها

در بخش ۳-۲-۲، مفهوم کلیدهای مرتبط با جداول را معرفی کردیم. این مفاهیم همچنین به موجودیتها نیز اعمال می شوند. برای نمونه، branchNo (شماره شعبه) و zipCode (کدپستی شعبه) کلیدهای کاندید موجودیت Branch هستند، بعلت اینکه هرکدام مقدار مجزائی برای هر رخداد شعبه دارند. اگر ما branchNo را بعنوان کلید اصلی موجودیت Branch انتخاب کنیم، در این صورت zipCode کلید فرعی خواهد شد.

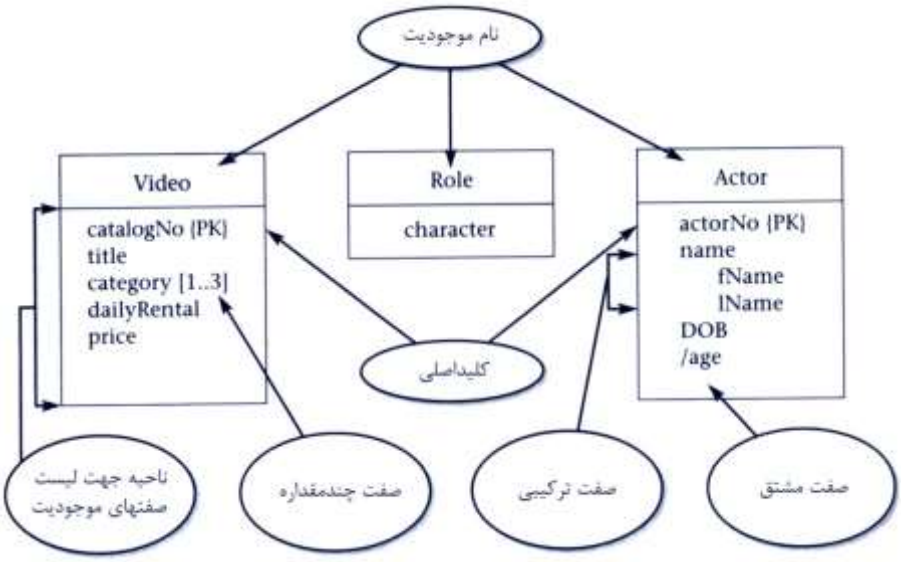
نمایش هندسی صفات

اگر موجودیت با صفاتش نمایش داده شود، آن را به صورت مستطیلی که به دو قسمت تقسیم شده نشان می دهیم. که قسمت بالایی مستطیل نام موجودیت و قسمت پائینی آن لیست صفتهای آن را نشان می دهد. برای نمونه، شکل ۶-۵ مدل ER موجودیتهای Video، Role، وActor و صفات مرتبط را نشان می دهد.

اولین صفت (ها)، در صورت شناخته شدن کلید اصلی موجودیت را، نشان می دهد. نام کلید اصلی صفات با برچسب {PK} مشخص میشود. در UML، نام صفت با حرف اول کوچک نشان داده شده و اگر نام بیش از یک کلمه داشته باشد، نام اول کلمات بعد از کلمه اول با حرف بزرگ نشان داده می شود. (برای نمونه، catalogNo ،actorNo، character). برچسبهای اضافی دیگری که می تواند استفاده شود عبارتند از کلید اصلی جزئی {PPK}، وقتیکه صفت تنها از قسمتی از کلید اصلی ترکیبی تشکیل شده باشد، و همچنین کلید فرعی {AK}.

برای صفات تک مقداره ساده، نیازی به استفاده از برچسب نیست و بنابراین بسادگی نام صفات را در لیستی زیر نام موجودیت نشان می دهیم. برای صفات ترکیبی نیز، نام صفات ترکیبی را در زیر و به صورت تو رفته بسمت راست بعلاوه نامهای اجزای آنها نشان می دهیم. برای نمونه، در شکل ۶-۵ صفت ترکیبی name بدنبال نام مولفه های صفات، fName و lName آمده است.

برای صفات چند مقداره، نام صفات را با اشاره به محدوده مقادیر موجود برای صفات برچسب می زنیم. برای نمونه، اگر صفت category را با محدوده [*..1] برچسب بزنیم، بدین معنی است که یک یا چند مقدار برای صفت category وجود دارد. که اگر مقدار دقیق را بدانیم در محدوده مقدار دقیق را می نویسیم. برای نمونه، اگر صفت category یک تا حداکثر سه مقدار را نگه دارد، صفت را با [1..3] برچسب خواهیم زد.



شکل ۶-۵

نمـایش هندسـی صـفات موجودیتهـای Video، Role ، و Actor .

برای صفات مشتق شده، در ابتدای نام صفت یک '/' قرارمی دهیم. برای نمونه، صفت مشتق شده age در شکل ۶-۵ بصورت /age نشان داده شده است.

توجه کنید که برای موجودیت Role هیچ کلید اصلی تعیین نشده است. وجود یا عدم وجود کلید اصلی به ما اجازه می دهد تا بدانیم موجودیت ضعیف است یا قوی. مفهوم ضعیف یا قوی بودن موجودیت را در ادامه بررسی می کنیم.

برای بعضی از پایگاه داده های ساده، نشان دادن تمام صفات هر موجودیت در مدل داده ای ممکن است. با این وجود، برای بیشتر کاربردهای پایگاه داده های پیچیده، شما تنها صفاتی را نشان دهید که کلید اصلی هر موجودیت را تشکیل می دهند. و زمانیکه تنها صفات شامل کلید اصلی در مدل ER نشان داده شوند، دراین صورت میتوانید برچسب {PK} را حذف کنید.

۵-۴ موجودیتهای قوی و ضعیف

ما می توانیم موجودیتها را بر اساس قوی یا ضعیف بودنشان دسته بندی کنیم.

برای نمونه، از آنجائیکه می توانیم یک بازیگر را از بازیگر دیگر و یک فیلم را از دیگر فیلمها بدون وجود هیچ موجودیت دیگر تشخیص دهیم، بنابراین موجودیتهای Actor و Video موجودیت **قوی** هستند. بعبارت دیگر، دو موجودیت فوق قوی هستند به خاطر اینکه کلید اصلی مختص به خود را دارند، همانطور که در شکل ۶-۵ نشان داده شده است.

شکل ۶-۵ همچنین دارای **موجودیت ضعیف** بنام Role نیز هست، که شخصیت بازی شده توسط بازیگر در فیلم را نشان می دهد. اگر ما نتوانیم به طور مجزا یک رخداد موجودیت Role را از دیگری بدون وجود دو موجودیت Actor و Video مشخص کنیم، پس Role یک موجودیت ضعیف می شود. بعبارت دیگر، موجودیت فوق ضعیف است چون کلید اصلی مختص بخود را ندارد.

موجودیتهای قوی بعضا موجودیت والد، مالک، یا غالب نیز نامیده می شود و همچنین به موجودیت ضعیف نیز بعنوان فرزند، وابسته، یا تابع نیز گفته می شود.

۵-۵ محدودیتهای کثرت رابطه ها

حال به بررسی محدودیتهای موجود بر روی موجودیتهای شرکت کننده در رابطه می پردازیم. نمونه هایی از چنین محدودیتها نیازمندیهایی که در آن شعبه باید عضوهایی داشته باشد و هر شعبه باید پرسنل داشته باشد، می باشد. نوع اصلی محدودیت روی رابطه ها **کثرت**[۱] نامیده می شود.

کثرت تعداد رخداد های موجودیتی که به رخداد دیگر موجودیتها از طریق رابطه مشخصی مربوط هستند را محدود میکند. کثرت نمایش سیاستهای برقرار شده توسط کاربر یا شرکت است، و بعنوان **قواعد تجاری** [۲]بدان رجوع می شود. اطمینان از اینکه تمام قواعد تجاری مناسب شناسایی و ارائه شده باشد یکی از مهم ترین قسمت مدل سازی شرکت است.

همانطور که قبلا یادآور شدیم، عمومی ترین درجه رابطه باینری است. کثرت رابطه های باینری عموما یک به یک (۱:۱)، یک به چند(*:۱)، یا چند به چند(*:*) می باشد. این سه نوع ارتباط را با استفاده از قواعد تجاری زیر بررسی می کنیم :

- **عضوی از پرسنل شعبه را مدیریت می کند.**
- **شعبه دارای اعضاء پرسنلی است.**
- **هنرپیشه ها در فیلمها بازی می کنند.**

برای هر قاعده تجاری، نشان می دهیم که چگونه اگر، در بعضی موارد، کثرت را به طور واضح در قاعده مشخص نشده باشد حل کنیم، و چگونه آنرا در مدل ER نشان دهیم. در بخش ۴-۵-۵، کثرت را برای رابطه های با درجه بیشتر از دو نیز بررسی خواهیم کرد.

قابل توجه است که تمام قواعد تجاری به سهولت و آشکار در مدل ER نشان داده نمیشوند. برای نمونه، ممکن است مشکل باشد که نوع نیازمندی که عضوی از پرسنل تعطیلی روزانه اضافی هر سال کاری با شرکت را دریافت میکند را طور واضح در مدل ER نشان بدهیم.

۱-۵-۵ رابطه های یک- به- یک (۱:۱)

اجازه بدهید رابطه *Manages* را در نظر بگیریم، که موجودیتهای Staff و Branch را به هم مربوط می کند. شکل ۷-۵ الف نمونه یکتایی از رابطه *Manages* را با استفاده از مقادیری برای صفات کلید اصلی موجودیتهای Staff و Branch نشان می دهد.

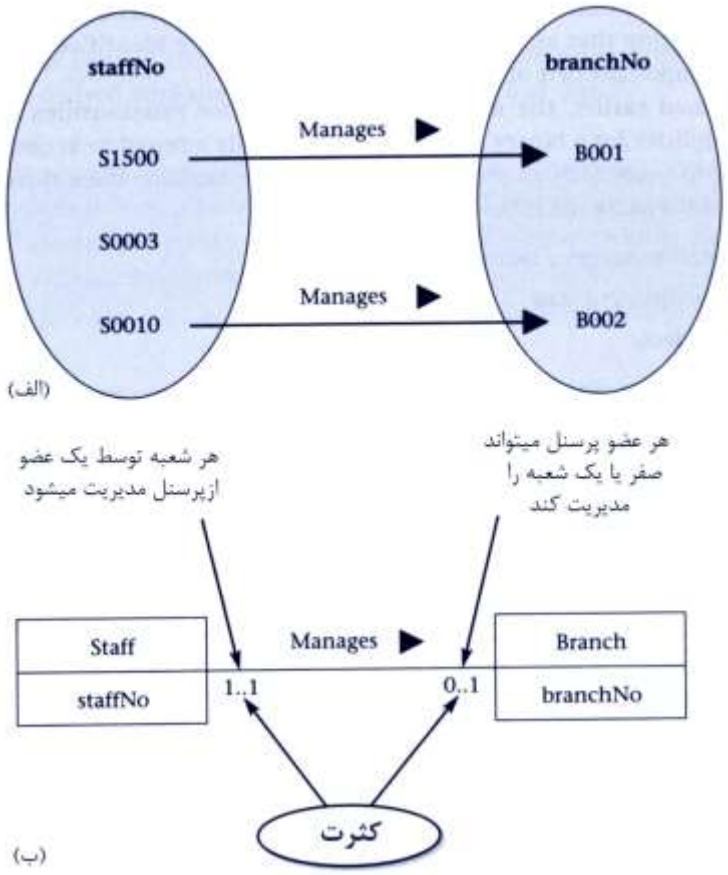
به کار بردن کثرت در عمل

جهت تمرین کثرت عموما نیاز به بررسی مختصر و مفیدی از رابطه های موجود بین داده مشخصی در قواعد تجاری با استفاده از داده نمونه ای داریم. داده نمونه شاید از طریق فرمهای تکمیل شده یا گزارش یا، در صورت امکان، از گفتگوی بیشتر با کاربر بدست بیاید. به هر حال، باید تاکید کنیم که برای بدست آوردن استنتاج درست درباره قواعد تجاری داده های نمونه بایستی بررسی شده یا درباره آنها بحثهایی انجام گیرد تا نمایش صحیحی از تمام داده ها داشته باشیم.

---

۱ Multiplicity
۲ Business Rule

در شکل ۷-۵ الف، می توانیم مشاهده کنیم که staffNo با شماره S1500 مدیریت می کند B001 BranchNo را و S0010 staffNo مدیریت می کند branchNo B002 را، اما staffNo S0003 هیچ شعبه ای را مدیریت نمی کند. به عبارت دیگر، هر عضو پرسنل می تواند صفر یا یک شعبه را مدیریت کند و هر شعبه توسط یکی از اعضاء پرسنلی مدیریت می شود. از آنجائیکه حداکثر یک شعبه برای هر عضو پرسنل و حداکثر یک عضو از پرسنل برای هر شعبه در رابطه در برگرفته می شود، ما به این نوع از ارتباط یک به یک می گوئیم، که به طور خلاصه معمولا با (۱:۱) نشان می دهیم.



شکل ۷-۵

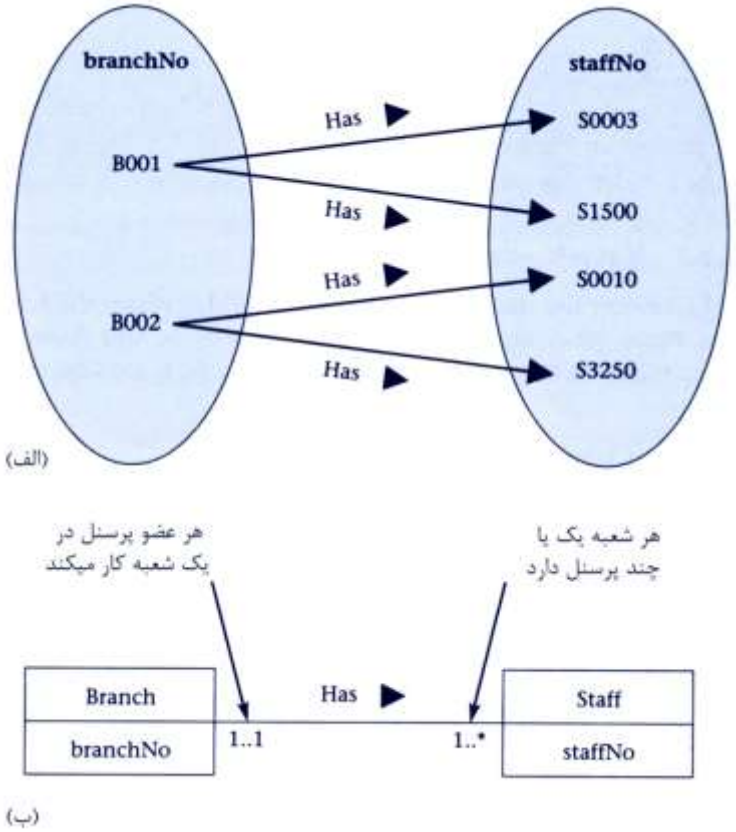
رابطه Staff *Manages* Branch : (الف) مثال نمونه؛ (ب) کثرت .

نمایش هندسی رابطه های ۱:۱

مدل ER رابطه Staff *Manages* Branch در شکل ۷- ۵(ب) نشان داده شده است. برای نشان دادن اینکه عضو پرسنل میتواند صفر یا یک شعبه را مدیریت کند، ما علامت'۱..۰' را در انتهای مقابل رابطه از موجودیت Staff قرار می دهیم. برای نشان دادن اینکه شعبه همیشه یک مدیر دارد، علامت '۱..۱' در طرف مقابل انتهای رابطه از موجودیت Branch قرار می دهیم. (توجه کنید که برای رابطه ۱:۱، نامی قابل فهم برای رابطه بین دو طرف انتخاب کرده ایم.)

۲-۵-۵ رابطه های یک به چند (*:۱)

اجازه بدهید رابطه ای بنام *Has* را در نظر بگیریم، که دوباره موجودیتهای Staff و Branch را بهم مربوط می کند. شکل ۸-۵ (الف) نمونه های منحصر بفرد رابطه Branch *Has* Staff را با استفاده از مقادیر نمونه ای برای صفات کلید اصلی موجودیتهای Branch و Staff نشان می دهد.



شکل ۸-۵

رابطه Branch *Has* Staff:

(الف) مثال نمونه؛ (ب) کثرت .

به کاربردن کثرت در عمل

در شکل ۸-۵ (الف)، دیدیم که B001 branchNo دارد S0003 staffNo و S1500 را، و B002 branchNo دارد S0010 staffNo وS3250 را. بنابراین، هر شعبه یک یا چندین عضو دارد و هر عضو پرسنل در یک شعبه کار می کنند. از آنجائیکه یک شعبه می تواند چندین پرسنل داشته باشد، ما به این نوع از ارتباط یک به چند می گوییم، که به طور خلاصه (*:۱) می نویسیم.

*نمایش هندسی رابطه های *:۱*

مدل ER رابطه Branch *Has* Staff در شکل ۸-۵ (ب) نشان داده شده است. برای نمایش اینکه هر شعبه می تواند یک یا چند پرسنل داشته باشد، نماد '*..۱' در طرف مخالف پایانی رابطه از موجودیت شعبه قرار می دهیم. برای نشان دادن اینکه هر عضو پرسنل در یک شعبه کار می کنند، نماد '۱..۱' را در طرف انتهای مقابل رابطه آمده از موجودیت پرسنل قرار می دهیم. (در رابطه ۱:* توجه شود که، ما نامی قابل فهم در طرفهای ۱:* انتخاب کرده ایم.)

نکته اگر حداکثر و حداقل مقادیر واقعی کثرت را میدانید، میتوانید آنها را به جای نمادها قرار دهید. برای نمونه، اگر شعبه بین دو و ده پرسنل داشته باشد، میتوانیم بجای '*..۱' نماد '۱۰..۲' را قرار دهیم.

۳-۵-۵ رابطه های چند-به-چند (*:*)

اجازه بدهید رابطه *PlaysIn* را در نظر بگیریم، که موجودیتهای Actor و Video را بهم مربوط میکند. شکل ۹-۵(الف) نمونه منحصربفرد رابطه Actor *PlaysIn* Video را با استفاده از مقادیری برای صفات کلید اصلی موجودیتهای Actor و Video را نشان میدهد.

بکار بردن کثرت در عمل

در شکل ۹-۵ (الف)، می بینیم که A3006 actorNo در فیلمهای با catalogNos ۹۰۲۳۵۵ و ۳۳۰۵۵۳ بازی می کند . بعبارت دیگر، یک بازیگر می تواند در یک یا چند فیلم بازی کند. همچنین می بینیم که catalogNo فیلم ۳۳۰۵۵۳ دو بازیگر نقش اول دارد اما catalogNo ۳۴۸۹۸۹ هیچ بازیگر ای که در آن بازی کند ندارد، و بنابراین نتیجه می گیریم که یک فیلم می تواند یک یا چند بازیگر نقش اول داشته باشد.

به طورخلاصه، رابطه *PlaysIn* از دید دو موجودیت بازیگر و فیلم رابطه *:۱ است. ما این رابطه را به صورت دو رابطه *:۱ در هر دو طرف نمایش می دهیم، که به طورکلی بنام رابطه چند به چند، که مختصرا با (*:*) نشان داده می شود، نامیده می شود.

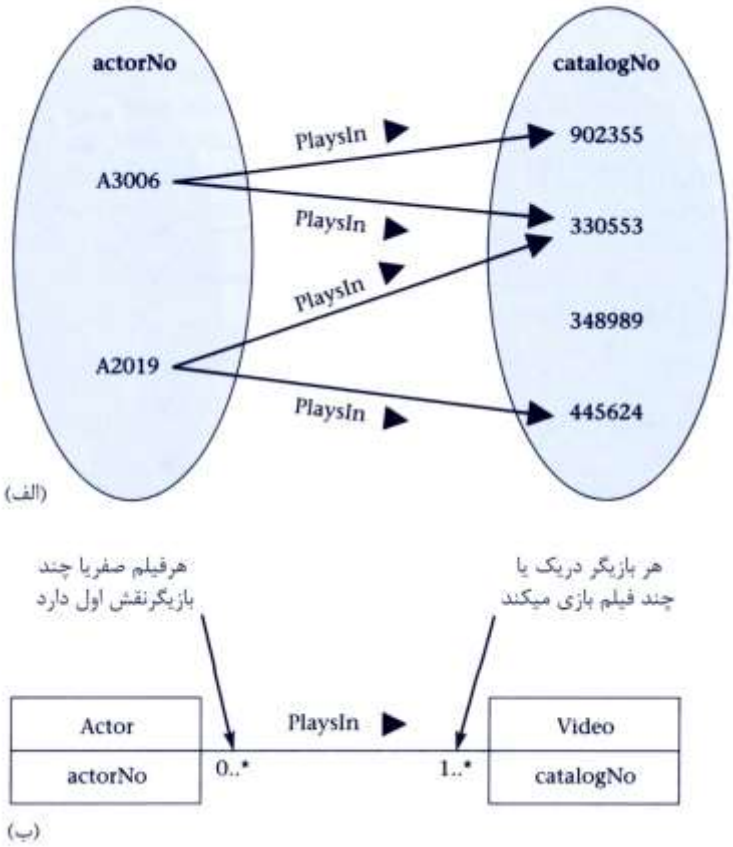
*نمایش هندسی رابطه های *:**

مدل ER رابطه Actor *PlaysIn* Video در شکل ۹-۵(ب) نشان داده شده است. برای نمایش اینکه هر بازیگر میتواند در یک یا چند فیلم بدرخشد، ما نماد '*..۱' را در انتهای مخالف رابطه آمده از موجودیت بازیگر قرار می دهیم. برای نشان دادن اینکه، هر فیلم می تواند یک یا چند بازیگر ستاره داشته باشد، نماد '*..۰' را در طرف مقابل رابطه آمده از موجودیت فیلم قرار می دهیم.

۴-۵-۵ کثرت برای رابطه های غیر دودویی

کثرت برای رابطه های فراتر از دو اندکی پیچیده است. برای نمونه، کثرت برای رابطه سه گانه تعداد بالقوه ای از رویدادها را در موجودیت نشان می دهد آنهم وقیتکه دیگر دو مقدار ثابت هستند. اجازه دهید دوباره رابطه سه گانه ***Registers*** بین شعبه، پرسنل، و عضو نشان داده شده در شکل ۴-۵ را در نظر بگیریم.

شکل ۱۰-۵ (الف) نمونه واحدی از رابطه ***Registers*** را وقتی مقادیر موجودیتهای عضو و پرسنل ثابت هستند را نشان می دهد.



شکل ۹-۵

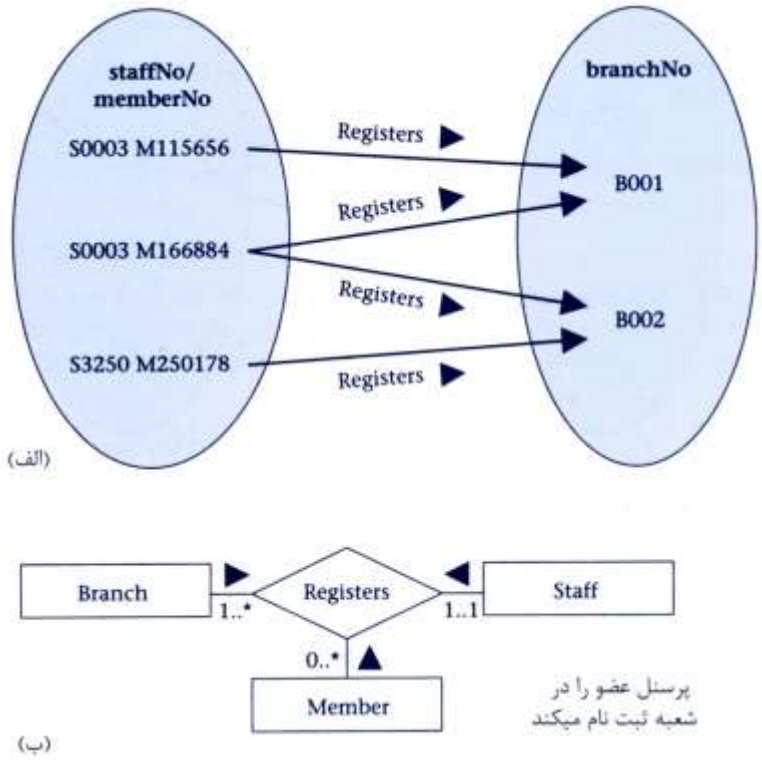
رابطه Actor *PlaysIn* Video: (الف) مثال نمونه ؛ (ب) کثرت .

به کاربردن کثرت در عمل

در شکل ۱۰-۵(الف)، برای هر ترکیبی از مقادیر staffNo/memberNo دیدیم که همیشه حداقل یک مقدار متناظر branchNo وجود دارد. خصوصا، S0003 staffNo ثبت نام می کند M166884 memberNo را در branchNo B001 وB002. این بیانگر موقعیتی است که عضو M166884 در شعبه B001 توسط پرسنل S003 ثبت نام شده است، و متعاقبا در B002 دوباره توسط همان شخصی که به شعبه B002 در بین دو ثبت نام انتقال داده، ثبت نام شده است. بعبارت دیگر، از نقطه نظر شعبه کثرت *..۱ است.

اگر این تست را از نقطه نظر پرسنل تکرار کنیم، مشاهده می کنیم که رابطه برای پرسنل ۱..۱ است، و اگر همان را از نقطه نظر عضو بررسی کنیم می بینیم *..۰ است. مدل ER رابطه سه گانه ***Registers*** نشان دهنده کثرت در شکل ۱۰-۵(ب) آمده است.

به طور کلی، کثرت برای رابطه های n- گانه وقتیکه دیگر مقادیر (n-1) ثابت هستند بیانگر تعداد بالقوه ای از رخداد موجودیت در رابطه است.



شکل۱۰-۵ (الف)

رابطه سه گانه *Registers* از دید Branch با مقادیری برای Staff و Member : (الف) مثال نمونه؛ (ب) کثرت .

خلاصه ای از راههایی که میتوانید محدودیتهای کثرت را نمایش دهید با توضیح در جدول ۱-۵ آمده است.

جدول ۱-۵ خلاصه راههای نمایش محدودیتهای کثرت.

راههای پیشنهادی برای نمایش محدودیتهای کثرت	*معنی*
۰..۱	وقوع صفر یا یک موجودیت
۱..۱ (یا فقط ۱)	وقوع دقیقا یک موجودیت
*..۰ (یا فقط *)	وقوع صفر یا چندین موجودیت
*..۱	وقوع یک یا چندین موجودیت
۵..۱۰	وقوع حداقل ۵ و حداکثر ۱۰ موجودیت
۰، ۳ ، ۸- ۶	وقوع صفر یا سه یا شش، هفت، یا هشت موجودیت

۵-۵-۵ کاردینالیتی و محدودیتهای مشارکت

کثرت در حقیقت شامل دو محدودیت جدا از هم بنام کاردینالیتی و شرکت[۱] است.

کاردینالیتی[۲] رابطه دودویی آن چیزی است که ما به آن بعنوان یک به یک، یک به چند، و چند به چند رجوع می کنیم.

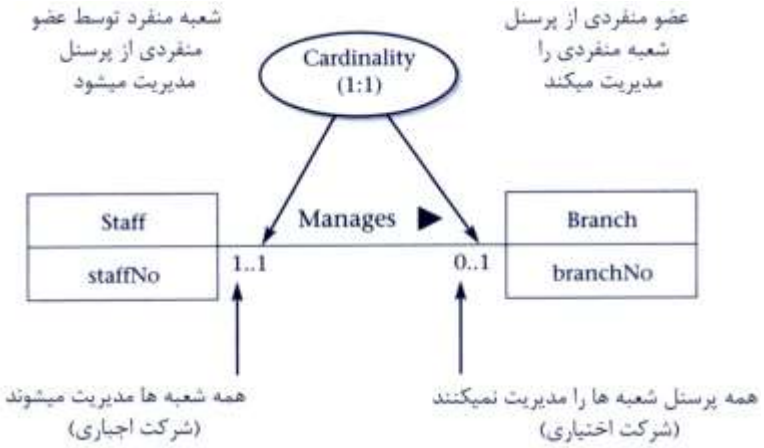
محدودیت شرکت بیان می کند که آیا همه رخدادهای موجودیت در رابطه مشخصی وارد شده اند (شرکت اجباری) یا فقط بعضی (شرکت اختیاری) وارد شده اند. در شکل ۱۱- ۵، کاردینالیتی و محدودیت شرکت را برای رابطه Staff *Manages* Branch شکل ۷-۵ (ب) را نشان می دهیم. همچنین طی متدلوژی طراحی منطقی پایگاه داده از محدودیت شرکت استفاده خواهیم کرد تا مشخص کنیم :

(الف) چگونه جداولی برای رابطه های یک به یک ایجاد کنیم (در مراحل ۱-۲ و ۴-۲ پوشش داده شده اند);
(ب) آیا کلید خارجی میتواند null داشته باشد (در مرحله ۴-۲).

۵-۶ صفات در رابطه ها

همانطور که از بخش ۳-۵ بیاد دارید، صفات می توانند به رابطه ها نیز نسبت داده شوند. برای نمونه، اجازه بدهید رابطه *PlaysIn* را در نظر بگیریم، که موجودیتهای Actor و Video را بهم مربوط می سازد. ممکن است بخواهیم شخصیت بازی شده توسط بازیگر در فیلم مزبور را ثبت کنیم. اطلاعات بیشتر به جای اینکه با موجودیتهای Actor یا video مربوط شده باشند با رابطه *PlaysIn* مربوط شده اند. همانگونه که در شکل ۱۲-۵ نشان داده شده است صفتی را بنام character برای ذخیره این اطلاعات ایجاد کرده و به رابطه *PlaysIn* نسبت می دهیم. توجه کنید، در این شکل صفت character با استفاده از نماد برای موجودیت نشان داده شده است; با این وجود، برای تمییز دادن میان رابطه با صفات و موجودیت، مستطیلی که بیانگر صفت است با خطوط تیره بهم مربوط شده اند.

وجود یک یا چند صفت انتساب داده شده به رابطه احتمالا نشانگر این است که رابطه موجودیت ناشناسی را پنهان ساخته است. برای نمونه، صفت character مربوط با موجودیتی بنام Role شکل ۶-۵ می باشد که قبلا نشان داده شده بود.



شکل ۱۱-۵

کثرت نشان داده شده بعنوان کاردینالیتی و محدودیتهای جامعیت برای رابطه ۱:۱ Staff Manages Branch نشان داده شده در شکل ۷-۵(ب) .

---

[۱] Participation
[۲] Cardinality

شکل ۱۲-۵

رابطه با نام *PlaysIn* با صفتی بنام character.

۵-۷ مشکلات طراحی با مدل ER

در این بخش، دو مورد از مشکلاتی که ممکن است هنگام طراحی مدل ER بوجود آیند را بررسی می کنیم. این مشکلات عموما بنام *تله های ارتباطی*[1] نامیده می شوند، و معمولا هنگام تفسیر اشتباه معنی روابط خاصی، رخ می دهند. ما دو نوع مهم این تله های ارتباطی با نامهای fan trap و Interrupt / chasm trap را بررسی می کنیم، و نشان می دهیم که چگونه اینگونه مشکلات را در مدل ER یافته و حل کنیم.

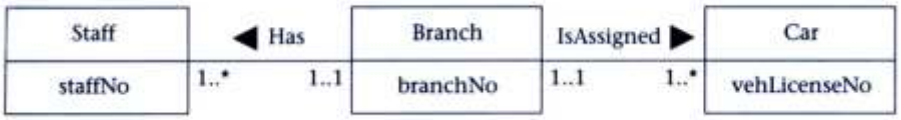
به طورکلی، برای شناسایی تله های ارتباطی بایستی اطمینان حاصل کنیم (و قواعد تجاری که آنها بیان می کنند) که مفهوم رابطه را کاملا فهمیده و به طور واضح تعریف کرده ایم. در غیر اینصورت مدلی را ایجاد خواهیم کرد که نمایش صحیحی از 'جهان واقعی' نخواهد بود.

۵-۷-۱ Fan trap

Fan trap زمانیکه دو یا چندین ارتباط یک به چند (*:۱) از یک موجودیت خارج شده باشند بوجود میاید. Fan trap در شکل ۱۳- ۵(الف) نشان داده شده است، که دو رابطه *:۱ (*Has* و *IsAssigned*) از موجودیت Branch خارج شده است. این مدل به ما می گوید که یک شعبه چندین پرسنل دارد و چندین اتومبیل به آن اختصاص داده شده است . بنابراین، در این رابطه مشکل هنگامی ایجاد می شود که بخواهیم بدانیم که کدام عضو پرسنل از اتومبیل بخصوصی استفاده می کند. برای درک مساله، اجازه دهید چندین نمونه از روابط *Has* و *IsAssigned* را با استفاده از مقادیر داده شده برای صفات کلید اصلی موجودیتهای پرسنل، شعبه، و اتومبیل بررسی کنیم، همانگونه که در شکل ۱۳-۵(ب) نشان داده شده است.

اگر سعی کنیم به این سوال جواب دهیم که : ' کدام عضو پرسنل از اتومبیل SH34 استفاده می کند؟' ، غیر ممکن است که بتوان با ساختار فعلی به این سوال جواب داد. ما تنها می توانیم مشخص کنیم که اتومبیل SH34 به شعبه B001 اختصاص دارد اما نمی توان گفت که پرسنل S0003 یا S1500 از این اتومبیل استفاده می کنند یا نه. که علت این عدم توانایی ناشی از fan trap است.

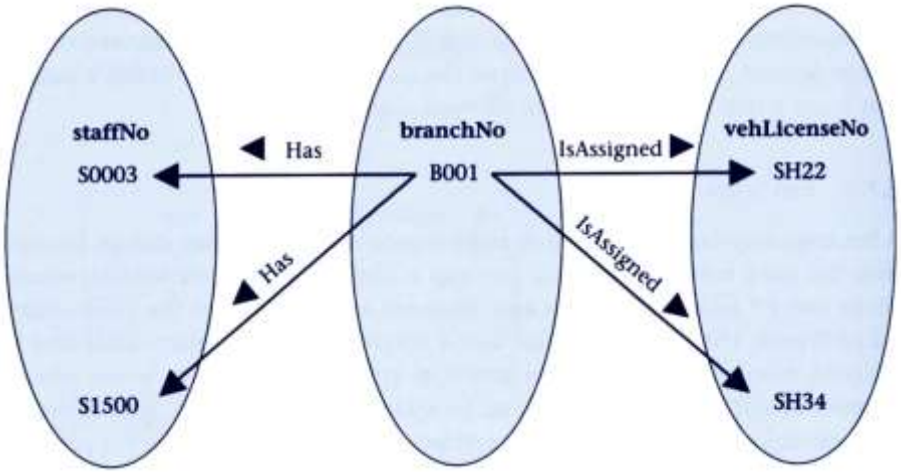
همانگونه که در شکل ۱۳-۵(ج) نشان داده شده است، ما این تله را با افزودن رابطه ای بنام *Uses* Car Staff به مدل ER اصلی برطرف می کنیم. حال اگر نمونه رابطه های *Has* ، *IsAssigned*، و *Uses* نشان داده شده در شکل ۱۳-۵ (ج) را بررسی کنیم، میتوان دید که پرسنل S1500 از اتومبیل SH34 استفاده می کند.



شکل ۱۳-۵(الف)

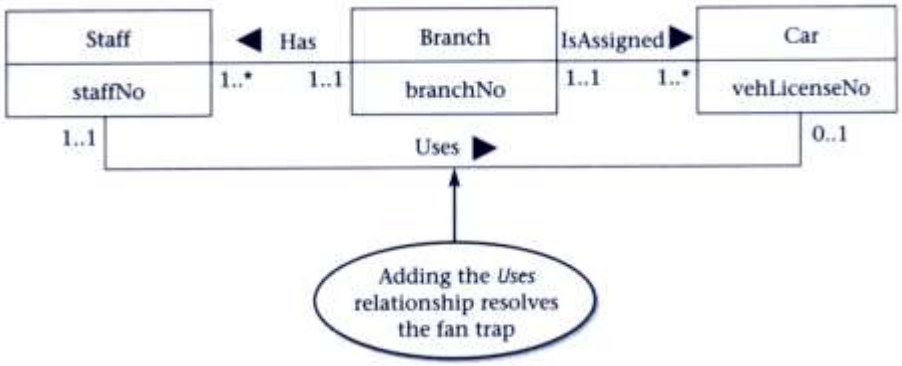
نمونه ای از fan trap .

[1] Connection traps



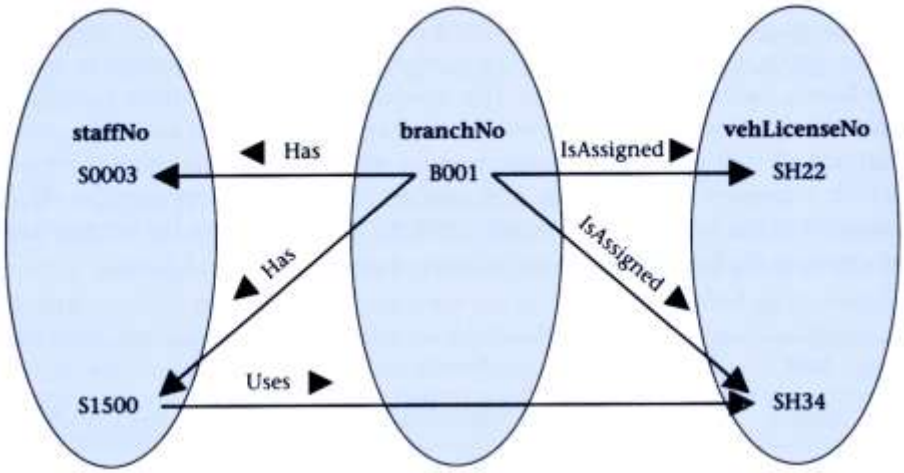
شکل ۱۳-۵(ب)

نمونه هایی از رابطه های Staff *Has* Branch و Car *IsAssigned* Branch. نمیتواند بگوید کدام عضو پرسنل از اتومبیل SH34 استفاده می کند.



شکل ۱۳-۵(ج)

برطرف سازی fan trap.



شکل ۱۳-۵(د)

نمونه هایی از رابطه های Staff *Has* Branch و Car *IsAssigned* Branch. حال میتواند بگوید پرسنل کدام اتومبیل را استفاده میکنند.

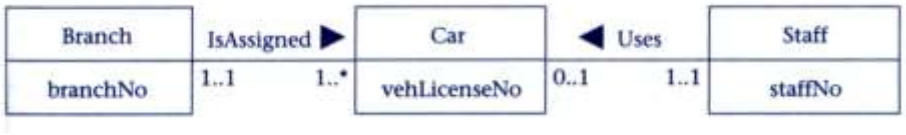
۵-۷-۲ Chasm trap

chasm trap جائی رخ می دهد که رابطه ای با شرکت اختیاری وجود داشته باشد که مسیر بین موجودیتهای مربوط بهم را شکل داده است. chasm trap در شکل ۱۴-۵(الف) نشان داده شده است، که ارتباط میان موجودیتهای Branch، Car، و Staff را نشان می دهد. این مدل به ما می گوید که یک شعبه چندین اتومبیل را در اختیار دارد و عضو پرسنل ممکن است از یک اتومبیل استفاده کنند. خصوصا، توجه کنید که تمام پرسنل از اتومبیل استفاده نمی کنند. مشکل زمانی ایجاد می شود که بخواهیم بدانیم که در کدام شعبه عضوی از پرسنل کار می کند. برای درک این مساله، اجازه دهید چندین نمونه از رابطه

های *Uses* و*IsAssigned* را با استفاده از مقادیر برای کلید اصلی صفات موجودیتهای Branch، Car، و Staff که در شکل ۱۴-۵(ب) نشان داده شده را بررسی کنیم.

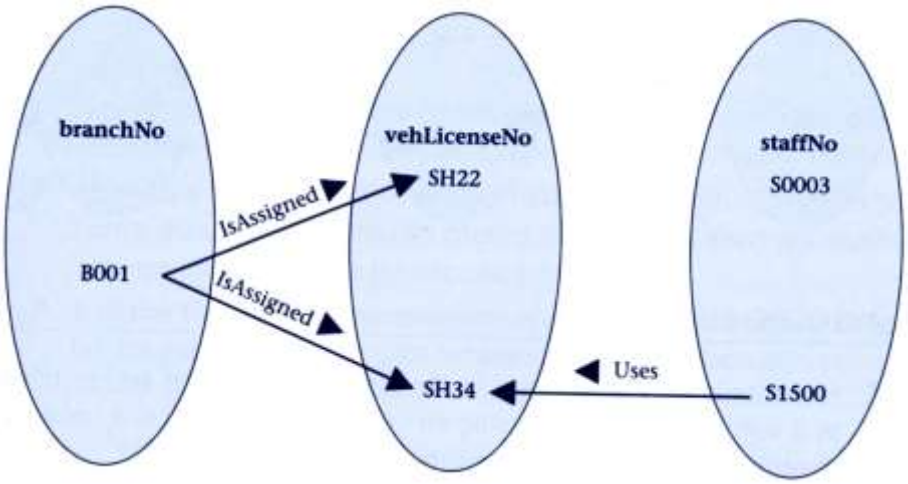
شکل ۱۴-۵(الف)

نمونه chasm trap .



شکل ۱۴-۵(ب)

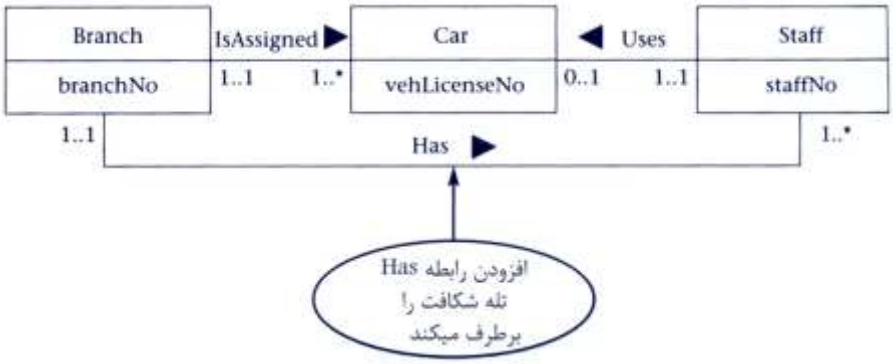
نمونه های رابطه های Branch *IsAssigned* Car و Staff *Uses* Car. نمیتواند بگوید درکدام شعبه پرسنل S003 کار میکند.



اگر به این سوال بخواهیم جواب دهیم که : 'در کدام شعبه پرسنل S0003 کار می کند؟'، از آنجائیکه تمام پرسنل از اتومبیل استفاده نمی کنند ما نمی توانیم با ساختار کنونی به این سوال جواب دهیم. عدم توانایی در جواب دادن به این سوال بعلت فقدان اطلاعات (همانطور که می دانیم عضو پرسنل باید در شعبه کار کنند) است، که باعث ایجاد **chasm trap** می شود. شرکت اختیاری پرسنل در رابطه Staff *Uses* Car بدین معنی است که بعضی از اعضاء پرسنل با شعبه از طریق استفاده از اتومبیل مربوط نیستند.

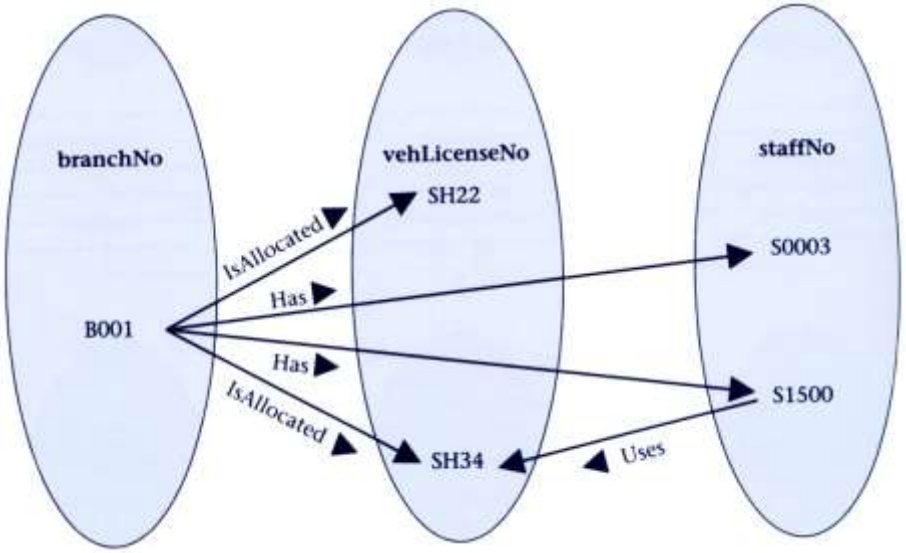
بنابراین، برای حل این مساله و برطرف ساختن **chasm trap**، ارتباط جدیدی بنام *Has* بین موجودیتهای پرسنل و شعبه اضافه می کنیم، همانطور که در شکل ۱۴-۵(ج) نشان داده شده است. حال اگر نمونه های *Has*، *Use* ، و *IsAssigned* را بررسی کنیم می توان مشاهده کرد که پرسنل S0003 در شعبه B001 کار می کند، که در شکل ۱۴-۵(د) آمده است.

بعضا ثابت شده است که مفاهیم ER شرح داده شده در این فصل برای مدل سازی کاربردهای پیچیده پایگاه داده ناکافی است. در فصل ۱۱، بعضی از مفاهیم پیشرفته مربوط به مدل ER را که در مدلسازی داده های پیچیده مفید است را معرفی خواهیم کرد.



شکل ۱۴-۵(ج)

برطرف سازی chasm trap .



شکل ۱۴-۵(د)

نمونه های رابطه های Branch Staff، *Has* Branch Car و *IsAssigned* Staff Car. *Uses* حال میتواند بگوید در هر شعبه کدام عضو پرسنل در آن کار می کنند.

خلاصه فصل

- ✓ **موجودیت** مجموعه ای از اشیاء با خصوصیات مشابه است که توسط کاربر یا شرکت شناخته می شوند و دارای وجود مستقل هستند. شی بطور مجزا قابل شناسایی **رخداد موجودیت** نامیده می شود.
- ✓ **رابطه** ارتباط با معنی بین موجودیتهاست. رابطه ای که به طور مجزا قابل شناسایی باشد **رخداد رابطه** نامیده می شود.
- ✓ **درجه رابطه** تعداد موجودیتهای شرکت کننده در رابطه است.
- ✓ **رابطه بازگشتی** رابطه ای است که موجودیت مشابه در بیش از یک نقش مختلف شرکت کند.
- ✓ **صفت** خصوصیت موجودیت در رابطه است.
- ✓ **صفت ترکیبی** ترکیب شده از چندین جزء واحد است.
- ✓ **صفت تک-مقداره** یک مقدار را برای رخداد موجودیت نگه می دارد.
- ✓ **صفت چندمقداره** چندین مقدار را برای رخداد موجودیت نگه می دارد.
- ✓ **صفت مشتق شده** بیانگر مقداری است که از مقدار صفت (یا مجموعه ای از صفات) وابسته به آن قابل اشتقاق است، که لزوما در موجودیت مشابه نیستند.
- ✓ **موجودیت قوی** وابسته به وجود موجودیت دیگر از نظر کلید اصلی نیست. **موجودیت ضعیف** بخشی از آن یا کاملا به وجود موجودیت دیگر (یا موجودیتهای دیگر) به کلید اصلی آن وابسته است.
- ✓ **کثرت** تعداد رخدادهای یک موجودیت را که به یک رخداد موجودیت مربوطه وابسته است را تعریف می کند.

- ✓ **کاردینالیتی** تعداد رابطه های ممکن برای هر موجودیت شرکت کننده را تعریف می کند.
- ✓ **محدودیت مشارکت** مشخص می کند که آیا تمام یا فقط بعضی از رخداد موجودیت در رابطه شرکت دارند.
- ✓ **Fan trap** وقتی رخ می دهد که دو موجودیت رابطه *:۱ داشته باشند که از موجودیت سوم خارج شده باشد، اما دو موجودیت بایستی رابطه مستقیمی بین هم داشته باشند تا اطلاعات لازم را فراهم سازد.
- ✓ **chasm trap** وجود رابطه بین موجودیتها را تاکید می کند، ولی مسیری بین رخدادهای موجودیت مشخص وجود ندارد.

نرمال سازی

آنچه در این فصل خواهید آموخت :

- تکنیکهای نرمالسازی چگونه در طراحی پایگاه داده مورد استفاده قرار می گیرند.
- چگونه جداولی را که شامل داده افزونه هستند میتوانند از بهنگام سازی غیر متعارف زیان ببینند، که میتوانند نشان دهنده ناسازگاری داخل پایگاه داده باشند.
- قواعد مرتبط با فرمهای نرمال متداول، بنام های فرمهای نرمال اول (1NF) ، دوم (2NF) ، و سوم (3NF).
- چگونه جداولی که قواعد 1NF، 2NF، و3NF نقض می کنند و شامل داده افزونه می باشند، از بهنگام سازی غیر متعارف زیان می بینند.
- چگونه جداولی که دوباره ساخته شده اند قواعد 1NF، 2NF، یا 3NF را نقض می کنند.

در فصل قبل، درباره مدلسازی ER که یک روش متدوال بالا به پایین طراحی پایگاه داده است مطالبی آموختیم. در این فصل، روش معروف دیگری بنام نرمالسازی که در طراحی پایگاه داده استفاده می شود را بررسی می کنیم. در طراحی پایگاه داده نرمالسازی به دو طریق استفاده می شود: اولین طریق استفاده از آن روش طراحی پایگاه داده پایین به بالا بوده و دومی استفاده از آن در اتصال با مدلسازی ER است.

استفاده از نرمال سازی در **روش پایین به بالا**[1] شامل تحلیل ارتباط میان صفتها بوده، و بر اساس این تحلیل است که گروه بندی صفتها با همدیگر جهت تشکیل جداولی که بیانگر موجودیتها و رابطه ها باشند صورت می گیرد. بنابراین، این روش در صورتیکه تعداد موجودیتها زیاد باشد، بعلت اینکه ساختن تمام ارتباطات مهم بین صفتها دشوار است عملا کار دشواری است.

به این جهت، در این کتاب متدلوژی ارائه می کنیم که به شما پیشنهاد می کند که باید برای طراحی پایگاه داده اول داده را با استفاده از روش پایین به بالا درک کنید. در این روش، برای ایجاد مدل داده ای که روابط و موجودیتهای مهم را نشان دهد از مدلسازی ER استفاده می کنیم. سپس مدل ER را به صورت مجموعه ای از جدولهایی که این داده ها را نشان دهد تبدیل می کنیم. آنچه در اینجا مطرح است این است که می خواهیم از نرمالسازی برای بررسی اینکه آیا جداول خوب طراحی شده اند استفاده کنیم.

هدف این فصل این است که بررسی کنیم که چرا نرمالسازی تکنیک مفیدی در طراحی پایگاه داده است و خصوصا، اینکه چگونه نرمالسازی می تواند در بررسی ساختار جداولی که از مدل ER ایجاد شده اند استفاده شود.

---

[1] Bottom-up approach

۶-۱ مقدمه

در سال ۱۹۷۲، دکتر کاد تکنیک **نرمالسازی** را جهت پشتیبانی از طراحی پایگاه داده بر اساس *مدل رابطه ای* توسعه داد. نرمالسازی اغلب بصورت یک سری تستهایی روی جدول انجام می گیرد تا مشخص شود که آیا قواعد بر روی **فرم نرمال** معینی برآورد می شود یا نه. چندین فرم نرمال وجود دارد، اگر چه پرکاربرد ترین آنها فرم نرمال اول (1NF)، فرم نرمال دوم (2NF)، و فرم نرمال سوم (3NF) می باشند. همه این فرمهای نرمال مبتنی بر قواعدی درباره روابط بین ستونهای جدول است.

در این بخش، ما ابتدا نشان می دهیم که چگونه جدولهایی که بصورت بد ساخته شده و یا دارای داده های افزونه ای هستند می توانند باعث مشکلاتی همچون *بهنگام سازی غیر متعارف* و خلاف قاعده شوند. احتمالا علت آن که جداول بد ساخته شده اند این باشد که اشتباهاتی در مدل ER اصلی و یا در مرحله برگرداندن مدل ER به جداول رخ داده است. ما سپس تعریفی را برای فرم نرمال اول (1NF)، فرم نرمال دوم (2NF)، و فرم نرمال سوم (3NF) ارائه می دهیم، و همچنین نشان می دهیم که هر فرم نرمال چگونه می تواند در شناسایی و برطرف ساختن مشکلات متفاوت موجود در جداولمان به کار رود.

۶-۲ افزونگی داده و ناهنجاریهای بهنگام سازی

هدف عمده طراحی پایگاه داده رابطه ای این است که ستونهای جدولها را جهت کاستن از افزونگی داده و کاهش فضای ذخیره سازی فایل مورد نیاز توسط جدولهای پیاده سازی شده پایه گروه بندی کنیم. برای نشان دادن مشکلات مربوط با افزونگی داده، اجازه دهید جدولهای Branch و Staff را که در شکل ۱-۶ نشان داده شده است را با جدول StaffBranch شکل ۲-۶ مقایسه کنیم.

جدول StaffBranch شکل دیگر جدولهای Branch و Staff است. جدول دارای ساختار زیرین است، با کلید اصلی برای هر جدول که زیرش خط کشیده شده است:

Staff (staffNo, name, position, salary, branchNo)

Branch (branchNo, branchAddress, telNo)

StaffBranch (staffNo, name, position, salary, branchNo, branchAddress, telNo)

در جدول StaffBranch **داده افزونه** وجود دارد; جزئیات شعبه برای هر عضو پرسنل در همان شعبه تکرار شده است. در مقابل، جزئیات هر شعبه فقط یکبار در جدول شعبه پدیدار شده و برای نشان دادن اینکه هر عضو پرسنل کجا قرار دارد فقط شماره شعبه (branchNo) در جدول پرسنل تکرار شده است. جداولی که داده افزونه دارند ممکن است مشکلاتی که **بهنگام سازی غیر متعارف** نامیده می شوند را باعث شوند، که بصورت درج، حذف، و اصلاح های غیر متعارف درجه بندی می شوند.

شکل ۱-۶

جدولهای Staff و Branch.

Staff

staffNo	name	position	salary	branchNo
S1500	Tom Daniels	Manager	46000	B001
S0003	Sally Adams	Assistant	30000	B001
S0010	Mary Martinez	Manager	50000	B002
S3250	Robert Chin	Supervisor	32000	B002
S2250	Sally Stern	Manager	48000	B004
S0415	Art Peters	Manager	41000	B003

Branch

branchNo	branchAddress	telNo
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618
B002	City Center Plaza, Seattle, WA 98122	206-555-6756
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131

شکل ۲-۶

جدول StaffBranch.

staffNo	name	position	salary	branchNo	branchAddress	telNo
S1500	Tom Daniels	Manager	46000	B001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0003	Sally Adams	Assistant	30000	B001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0010	Mary Martinez	Manager	50000	B002	City Center Plaza, Seattle, WA 98122	206-555-6756
S3250	Robert Chin	Supervisor	32000	B002	City Center Plaza, Seattle, WA 98122	206-555-6756
S2250	Sally Stern	Manager	48000	B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131
S0415	Art Peters	Manager	41000	B003	14 – 8th Avenue, New York, NY 10012	212-371-3000

۶-۲-۱ ناهنجاری های درج

دو نوع اصلی ناهنجاری درج وجود دارد، که ما با استفاده از جدول StaffBranch شکل ۲-۶ نشان می دهیم.

(۱) برای درج جزئیات عضو جدید در جدول StaffBranch، باید جزئیات شعبه ای را که پرسنل در آن قرار دارد را نیز در نظر بگیریم. برای مثال، برای درج جزئیات پرسنل جدید واقع در شعبه B003، باید جزئیات شعبه B003 را نیز وارد کنیم تا جزئیات شعبه با مقادیر شعبه B003 در دیگر رکوردهای جدول StaffBranch سازگار باشند. جدولهای نشان داده شده در شکل ۱-۶ تحت تاثیر این ناسازگاری ایجاد شده قرار نمی گیرند، زیرا برای هر عضو پرسنل تنها شماره شعبه مناسب را در جدول پرسنل وارد می کنیم. بعلاوه، جزئیات شعبه B003 فقط یکبار در پایگاه داده بصورت رکورد واحدی در جدول شعبه ذخیره شده اند.

(۲) برای وارد کردن جزئیات شعبه جدید که در حال حاضر هیچ عضوی از پرسنل را ندارند در جدول StaffBranch، لازم است که در ستونهایی که به پرسنل وابسته است همچون staffNo، null وارد کنیم. بنابراین، از آنجائیکه staffNo کلید اصلی جدول StaffBranch است، هر نوع تلاشی جهت وارد کردن Null برای staffNo جامعیت موجودیت را نقض کرده، و مجاز نمی باشد. طراحی جدولهای شکل ۱-۶ از اینگونه مشکلات اجتناب می کند زیرا جزئیات شعبه ای که در جدول شعبه درج می شود جدا از جزئیات پرسنل است.

۶-۲-۲ ناهنجاریهای حذف

اگر رکوردی را از جدول StaffBranch که آخرین عضو پرسنل موجود در آن شعبه را نشان می دهد را حذف کنیم، جزئیات شعبه مذکور از پایگاه داده گم میشود. برای نمونه، اگر رکورد مربوط به Staff S0415 با نام ('Art Peters') را از جدول StaffBranch حذف کنیم، جزئیات مرتبط با شعبه B003 از پایگاه داده گم میشود. طراحی جداول شکل ۱-۶ از این مشکل اجتناب می کند زیرا رکوردهای شعبه به طور جداگانه از رکوردهای پرسنل ذخیره شده اند و تنها ستون branchNo دو جدول را به هم ربط می دهد. اگر ما رکورد Staff S0415 را از جدول Staff حذف کنیم، جزئیات مربوط به شعبه B003 در جدول شعبه بی تاثیر باقی می ماند.

۶-۲-۳ ناهنجاریهای اصلاح

اگر بخواهیم مقدار یکی از ستونهای شعبه معینی را در جدول StaffBranch اصلاح کنیم، برای نمونه شماره تلفن شعبه B001 را، بایستی در اینصورت رکوردهای تمام پرسنل واقع در آن شعبه را نیز بروز کنیم. اگر این اصلاح در تمام رکوردهای مناسب جدول StaffBranch انجام نگیرد، پایگاه داده ناسازگار خواهد شد. در این مثال، شعبه B001 شماره تلفن های متفاوتی را در رکوردهای پرسنلی دیگر خواهد داشت.

مثالهای فوق نشان می دهد که جدولهای Staff و Branch در شکل ۱-۶ خصوصیات به مراتب بهتری را نسبت به جدول StaffBranch شکل ۲-۶ دارد. در بخش بعدی، بررسی می کنیم که چگونه شکلهای نرمال می تواند برای رسمی کردن هویت جدولهایی که خصوصیات بهتری نسبت به آنهایی که ممکن است به طور بالقوه از ناسازگاریهای بهنگام سازی تاثیر بپذیرند، به کار روند.

۶-۳ فرم نرمال اول (1NF)

فقط فرم نرمال اول (1NF) در ایجاد جدولهای مناسب برای پایگاه داده رابطه ای اهمیت دارد. بقیه شکلهای نرمال اختیاری هستند. با وجود این، برای اجتناب از ناسازگاریهای بهنگام سازی که در بخش ۲-۶ بحث شد، معمولا توصیه می شود که شما تا فرم نرمال سوم (3NF) پیش بروید.

اجازه دهید جدول Branch نشان داده شده در شکل ۳-۶ را، با کلید اصلی شماره شعبه (branchNo) در نظر بگیریم. میتوان مشاهده کرد که تمام ستونهای این نسخه از جدول Branch تعریف ما از1NF را برآورده می سازد بجز ستون مربوط به شماره تلفن (telNos). همانطور که می بینیم برای هر رکورد ستون (telNos) چندین مقدار وجود دارد. برای نمونه، شماره شعبه B001 سه شماره تلفن دارد، 503-555-3618 ، 503-555-2727 ، و503-555-6534. در نتیجه، جدول شعبه در 1NF نیست.

توجه کنید اگر چه ستون **branchAddress** چندین مقدار را می گیرد، اینگونه نمایش آدرس 1NF را نقض نمی کند. در این مثال، ما بسادگی می توانیم گزینه ای را انتخاب کنیم که تمام داده های آدرس را بصورت یک مقدار واحد نگهداری کند.

شکل ۳-۶

این نسخه از جدول Branch در 1NF نیست.

تبدیل به 1NF

برای تبدیل این نسخه از جدول شعبه به 1NF، جدول دیگری بنام BranchTelephone جهت نگه داشتن شماره تلفن های شعبه ها ایجاد می کنیم، که اینکار بوسیله برداشتن ستون telNos از جدول شعبه بهمراه کپی کلید اصلی جدول شعبه (branchNo) انجام می گیرد. کلید اصلی جدول جدید BranchTelephone ستون telNo است. ساختار اصلاح شده جدول شعبه و جدول جدید BranchTelephone در شکل ۴-۶ نشان داده شده است. جدولهای Branch و BranchTelephone بعلت اینکه در تقاطع رکورد با ستون هر جدول تنها یک مفدار وجود دارد بنابراین در 1NF می باشد.

۶-۴ فرم نرمال دوم (2NF)

فرم نرمال دوم تنها به جدولهایی با کلیدهای اصلی ترکیبی، یعنی، جدولهایی که کلید اصلی آنها ترکیبی از دو یا چند ستون است اعمال می شود. جدولی که کلید اصلی آن از یک ستون تشکیل شده به طور اتوماتیک حداقل در 2NF است. جدولی که در 2NF نیست احتمالا از ناسازگاریهای بهنگام سازی بحث شده در بخش ۲-۶ تاثیر یافته است.

اجازه بدهید جدول TempStaffAllocation را که در شکل ۵-۶ نشان داده شده است را بررسی کنیم. این جدول ساعات کاری هفتگی پرسنل موقتی موجود در هر شعبه را نشان می دهد.کلید اصلی جدول TempStaffAllocation متشکل از دو ستون staffNo و branchNo است. توجه کنید که ما عبارت ستونهای 'غیر کلید اصلی' را برای ارجاع به ستونهایی استفاده می کنیم که بخشی از کلید اصلی نباشند. برای مثال، ستونهای غیر کلید اصلی جدول TempStaffAllocation، name ، branchAddress ، position، و hoursPerWeek هستند.

پیکانهای نشان داده شده در زیر جدول TempStaffAllocation رابطه های خاص بین ستونهای کلید اصلی و غیر کلید اصلی را نشان می دهد.

رابطه های خاصی را که ما بین ستونهای جدول **TempStaffAllocation** در شکل ۵-۶ نشان می دهیم به طور رسمی **وابستگی های تابعی** نامیده می شوند. وابستگی تابعی[۱] خصیصه ای از معنی ستونها در جدول است که نشان می دهد که چگونه یک ستون به ستون دیگری مربوط می باشد.

برای مثال، جدولی را با ستونهای **A** و **B** در نظر بگیرید، که ستون B به طور تابعی به ستون A وابسته است (A→B) . اگر ما مقدار A را بدانیم، در هر لحظه تنها یک مقدار از B در تمام رکوردهایی که این مقدار را برای A دارند پیدا می کنیم. بنابراین، وقتی دو رکورد مقدار مشابهی از A را دارند، همچنین آنها مقدار مشابهی از B را نیز دارند. بنابراین، برای مقدار مذکور B ممکن است چندین مقدار متفاوت از A موجود باشد.

می توانیم ببینیم که جدول TempStaffAllocation شامل داده های افزونه است و احتمالا از ناسازگاریهای بهنگام سازی شرح داده شده در بخش ۲-۶ تاثیر پذیرفته است. برای مثال، جهت تغییر نام 'Ellen Layman'، مجبور هستیم تا دو رکورد موجود در جدول TempStaffAllocation را بروز کنیم. اگر تنها یک رکورد بروز شود، پایگاه داده ناسازگار خواهد شد. دلیل وجود داده افزونه در جدول TempStaffAllocation این است که این جدول با تعریف ما درباره 2NF سازگار نیست.

ستون غیر کلید اصلی branchAddress جدول TempStaffAllocation را در نظر بگیرید. مقدارهای موجود در ستون branchAddress بایستی بتواند از مقدارهای ستون branchNo (بخشی از کلیداصلی) استخراج شود. به عبارت دیگر، هر مقدار یکتا در ستون branchNo مقدار مشابهی در ستون branchAddress دارد. برای مثال، هر وقت که مقدار B002 در ستون branchNo ظاهرشود، آدرس متناظر'City Center Plaza, Seatel, WA 98122' در ستون branchAddress ظاهر می شود.

حال ستونهای غیر کلید اصلی name و position را در نظر بگیرید. مقادیر موجود در ستونهای name و position بایستی بتواند از مقادیر موجود درستون staffNo (بخشی از کلید اصلی) استخراج شود.

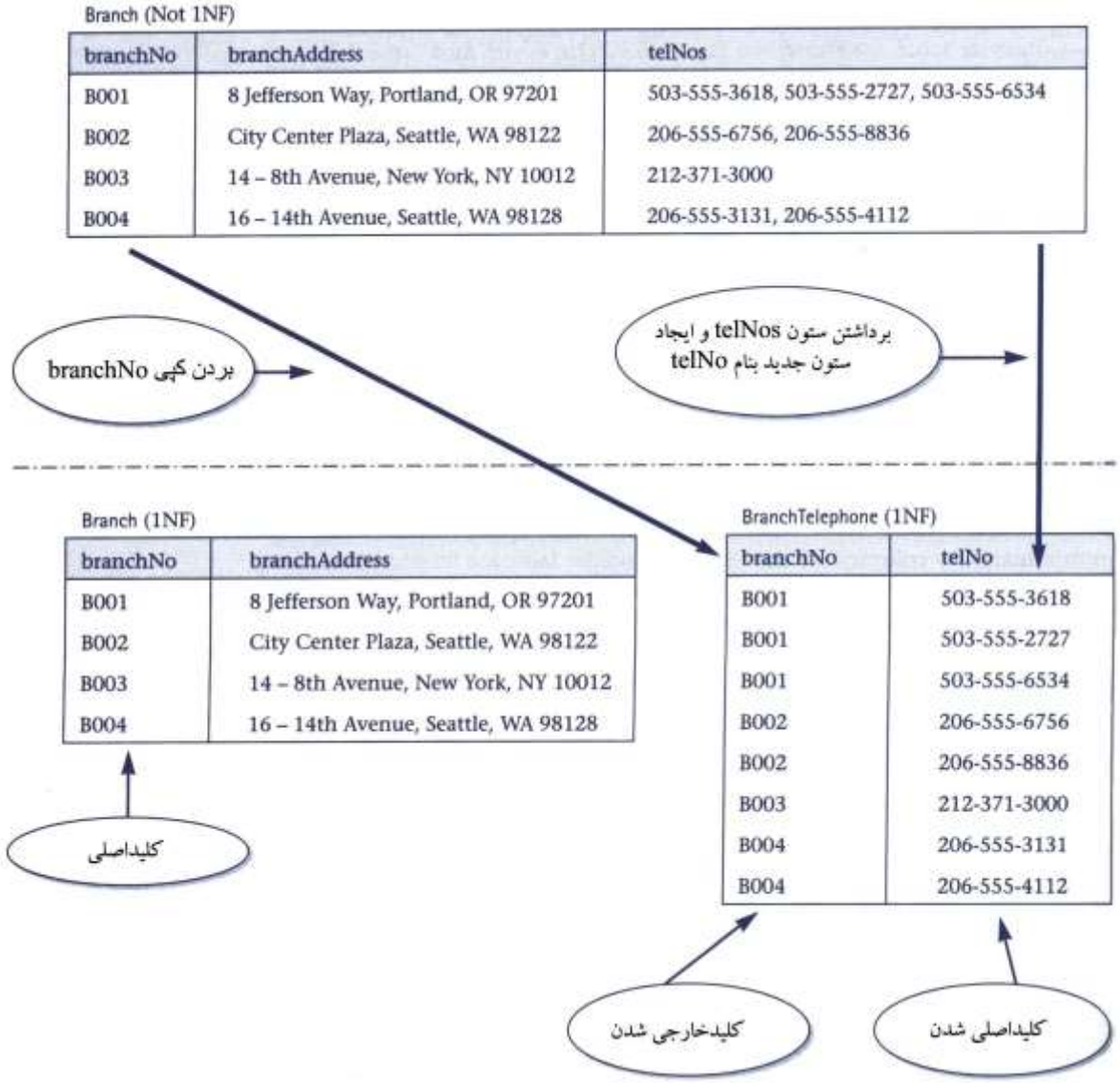
---

[۱] Functional dependency

شکل ۴-۶

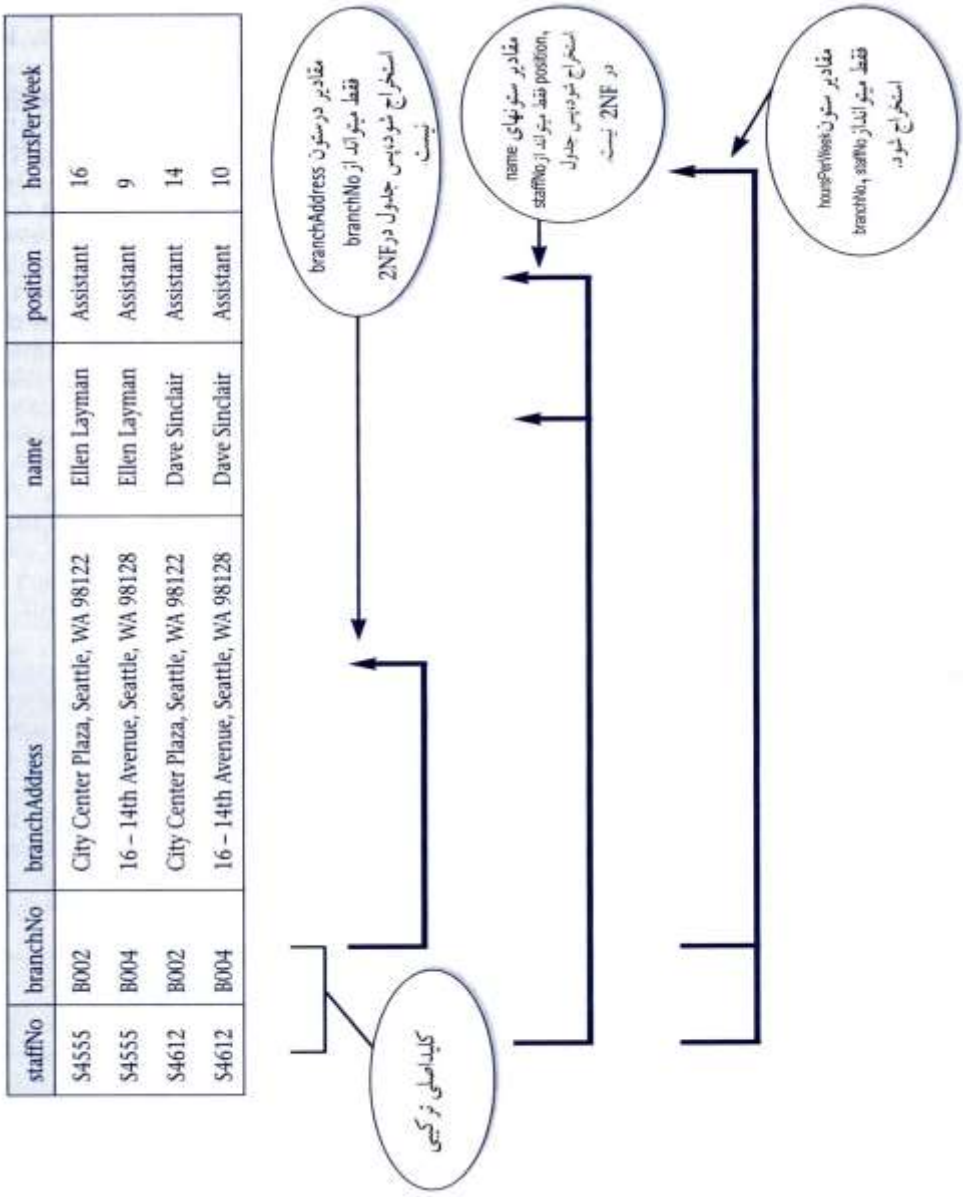
جدول اصلاح شده در 1NF با برداشتن ستونtelNo و ایجاد جدول جدیدی بنام BranchTelephone .

تبدیل به 1NF



شکل ۵-۶

جدول TempStaffAllocation در 2NF نیست.



staffNo	branchNo	branchAddress	name	position	hoursPerWeek
S4555	B002	City Center Plaza, Seattle, WA 98122	Ellen Layman	Assistant	16
S4555	B004	16 – 14th Avenue, Seattle, WA 98128	Ellen Layman	Assistant	9
S4612	B002	City Center Plaza, Seattle, WA 98122	Dave Sinclair	Assistant	14
S4612	B004	16 – 14th Avenue, Seattle, WA 98128	Dave Sinclair	Assistant	10

برای مثال، هر وقت S4555 در ستون staffNo ظاهر شد، نام 'Ellen Layman' و موقعیت شغلی 'Assistant' در ستونهای name و position باید ظاهر گردد.

بالاخره، ستون غیر کلید اصلی hoursPerWeek را در نظر بگیرید. مقادیر موجود در ستون hoursPerWeek می تواند فقط از مقادیر ستونهای staffNo و branchNo (همه کلید اصلی) استخراج شود. برای مثال، هر وقت S4555 در ستون staffNo ظاهر شود در زمان مشابه B002 در ستون branchNo ظاهر می شود، سپس مقدار '۱۶' در ستون hoursPerWeek ظاهر می شود.

تعریف رسمی فرم نرمال دوم (2NF) جدولی است که در فرم نرمال اول باشد و هر ستون غیر کلید اصلی کاملا بصورت تابعی به کلید اصلی وابسته باشد. **وابستگی تماما تابعی** نشان می دهد که اگر A و B ستونهای جدول باشند، B تماما از لحاظ تابعی به A وابسته است، اگر B به هر زیر مجموعه ای از A وابسته نباشد. اگر B به زیر مجموعه ای از A وابسته باشد، بنام **وابستگی جزئی**[۱] خوانده می شود. اگر وابستگی جزئی در کلید اصلی موجود باشد، جدول در 2NF نیست. وابستگی جزئی بایستی برای رسیدن به 2NF حذف شود.

تبدیل به 2NF

برای تبدیل جدول TempStaffAllocation نشان داده شده در شکل ۵-۶ به 2NF، لازم است تا ستونهای غیر کلید اصلی که با استفاده از تنها قسمتی از کلید اصلی استخراج شوند را حذف کنیم. بعبارت دیگر، باید ستونهایی را حذف کنیم که می توانند از ستون staffNo یا branchNo بدست آیند ولی به هر دو نیاز نیست. در جدول TempStaffAllocation، این بدین معنی است که باید ستونهای branchAddress، name، و position را برداشته و در جدول جدیدی قرار دهیم.

برای انجام این، دو جدول بنامهای Branch و TempStaff ایجاد می کنیم. جدول Branch شامل ستونهایی است که جزئیات شعبه ها را نگه می دارد و جدول TempStaff جزئیات پرسنل موقت را نگه می دارد.

(۱) جدول Branch با برداشتن ستون branchAddress از جدول TempStaffAllocation بهمراه کپی بخشی از کلید اصلی که ستون مرتبط به آن است، که در این حالت ستون branchNo است، ایجاد شده است.

(۲) به روش مشابه، جدول TempStaff با برداشتن ستونهای name و position از جدول TempStaffAllocation بهمراه کپی بخشی از کلید اصلی که ستونهایش به آن مربوط است، و در این مورد ستون staffNo است ایجاد شده است.

لازم نیست که ستون hoursPerWeek را حذف کنیم زیرا وجود این ستون قواعد 2NF را نقض نمی کند. ساختار جدول اصلاح شده TempStaffAllocation و جدولهای Branch و TempStaff در شکل ۶-۶ نشان داده شده است. کلید اصلی جدول جدید Branch، branchNo و جدول TempStaff ، StaffNo است.

جدولهای TempStaff و Branch بایستی در 2NF باشند چون کلید اصلی هر جدول ستون یکتا است. جدول اصلاح شده TempStaffAllocation همچنین در 2NF است چون ستون غیر کلید اصلی hoursPerWeek به ستونهای staffNo و branchNo وابسته است.

---

[۱] Partial dependency

۶-۵ فرم نرمال سوم (3NF)

اگر چه جدولهای 2NF افزونگی کمتری نسبت به 1NF دارند، هنوز با این حال در معرض ناسازگاریهای بهنگام سازی قرار دارند. اجازه دهید جدول BranchManager نشان داده شده در شکل ۶-۷، با کلید اصلی branchNo را بررسی کنیم. این جدول در 3NF نیست آنهم بعلت وجود ستون name است. اگرچه ما می توانیم نام مدیران را در شعبه مفروض از کلید اصلی، branchNo، بدست آوریم همچنین می توانیم نام را از دیگر ستون غیر کلید اصلی، mgrStaffNo نیز بدست بیاوریم. برای مثال، وقتیکه B001 در ستون branchNo ظاهر می شود، در ستون نام 'Tom Daniels' ظاهر می شود. بنابراین، وقتیکه S1500 در mgrStaffNo ظاهر می شود همچنین 'Tom Daniels' در ستون نام ظاهر می شود. بعبارت دیگر، نام عضو پرسنل می تواند با دانستن مقدار mgrStaffNo بدست آید. این در 3NF مجاز نیست زیرا مقادیر همه ستونهای غیرکلید اصلی بایستی فقط از مقادیر ستونهای کلید اصلی بدست بیایند.

تعریف رسمی فرم نرمال سوم (3NF) جدولی است در فرم نرمال اول و دوم که در آن هیچ ستون غیر کلید اصلی به طور انتقالی وابسته به کلید اصلی نباشند. **وابستگی انتقالی** نوعی از وابستگی تابعی است و وقتی رخ می دهد که نوع خاصی از رابطه بین ستونهای جدول نگه داشته شود.

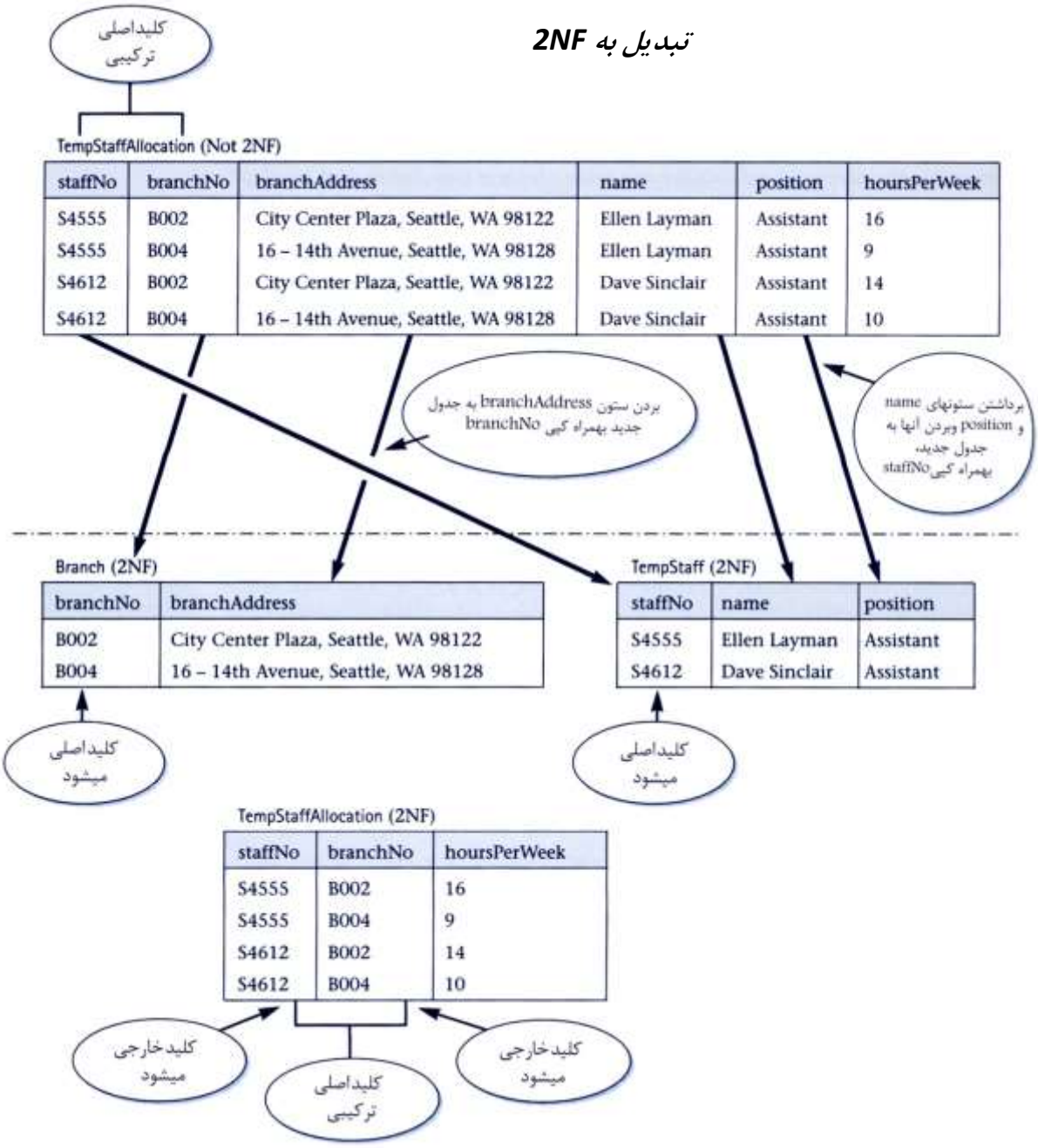
برای مثال، جدولی با ستونهای A، B، C را در نظر بگیرید. اگر B به طور تابعی وابسته به A باشد (A→B) و C وابسته به B باشد (C→B)، بنابراین C به طور انتقالی به A از طریق B وابسته است. اگر وابستگی انتقالی روی کلید اصلی وجود داشته باشد، جدول در 3NF نیست. برای رسیدن به 3NF وابستگی انتقالی باید برداشته شود.

به طور قابل توجه، جدول BranchManager که در شکل ۶-۷ نشان داده شده است، اینکه دارای داده افزونه ای باشد چنین نیست. و به خاطر این است که رابطه بین شعبه ها و مدیران رابطه یک به یک است. بعبارت دیگر، شعبه منفردی دارای مدیر منحصری بوده و یک مدیر نیز یک شعبه را مدیریت می کند. برای مثال، مدیر شماره شعبه B001 ، mgrStaffNo S1500 است و S1500 مدیر شماره شعبه B001 است نه شعبه دیگری.

بنابراین، ساختار کنونی جدولBranchManager احتمالا نوع افزونگی دارد که در برگیرنده بیش از یک جدول است. برای مثال، رابطه میان ستونهای mgrStaffNo و name در جدول BranchManager احتمالا در جدولی در پایگاه داده که تمام جزئیات پرسنل شامل مدیران را نگه می دارد وجود دارد. برای مثال، ستونهای جدولstaffNo و name شکل ۶-۱ را ببینید.

شکل ۶-۶

جدول اصلاح شده TempStaffAllocation در 2NF بواسطه برداشتن ستونهای name, branchAddress وposition و ایجاد جدولهای جدید Branch و TempStaff .



تبدیل به 3NF

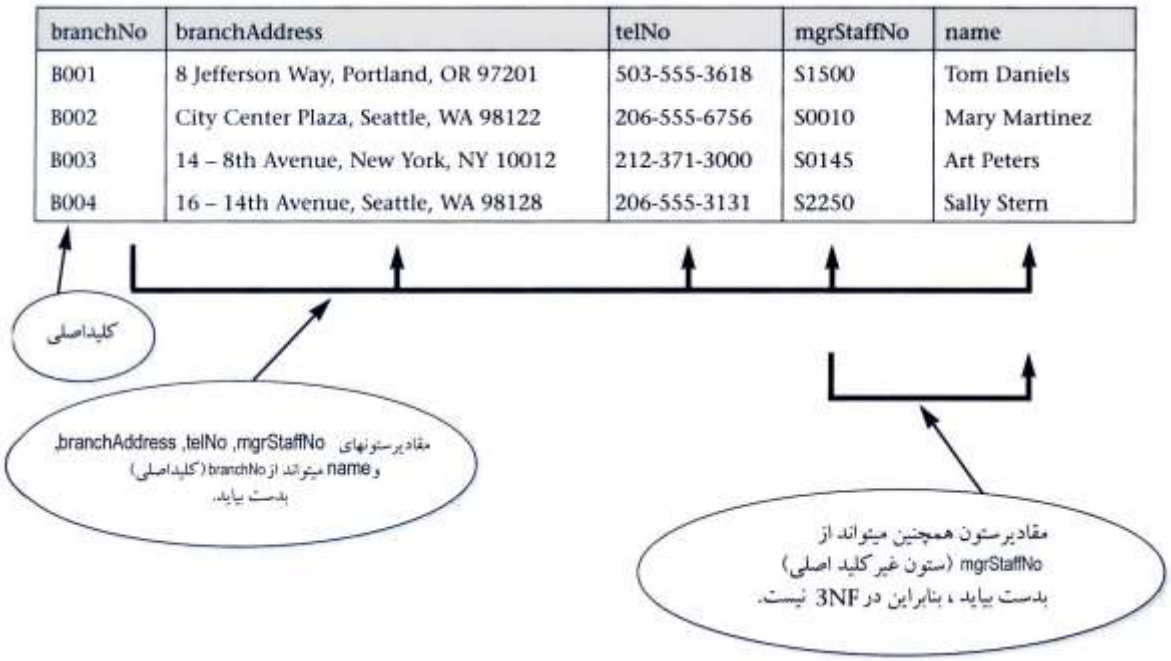
برای تبدیل جدول BranchManager شکل ۷-۶ به 3NF، باید ستون name را برداریم. سپس جدول دیگری بنام ManagerStaff ایجاد کرده تا ارتباط بین ستونهای mgrStaffNo و name را نشان دهیم. (البته، ایجاد جدول جدید اگر رابطه بین ستونهای mgrStaffNo(StaffNo) و name درحال حاضر در جدولی در پایگاه داده موجود باشد لزومی ندارد.) در ادامه ساختن دوباره جداول، نام جدول BranchManager را به Branch تغییر می دهیم. ساختار جداول جدید Branch و ManagerStaff در شکل ۸-۶ نشان داده شده است.

شکلهای نرمال دیگری همچون (4NF) و (5NF) بالاتر از این فرمهای نرمال نیز وجود دارد. با این همه، شکلهای نرمال بالاتر از 3NF عموما در برطرف کردن مشکلات جدولها استفاده نمی شوند.

شکل ۷-۶

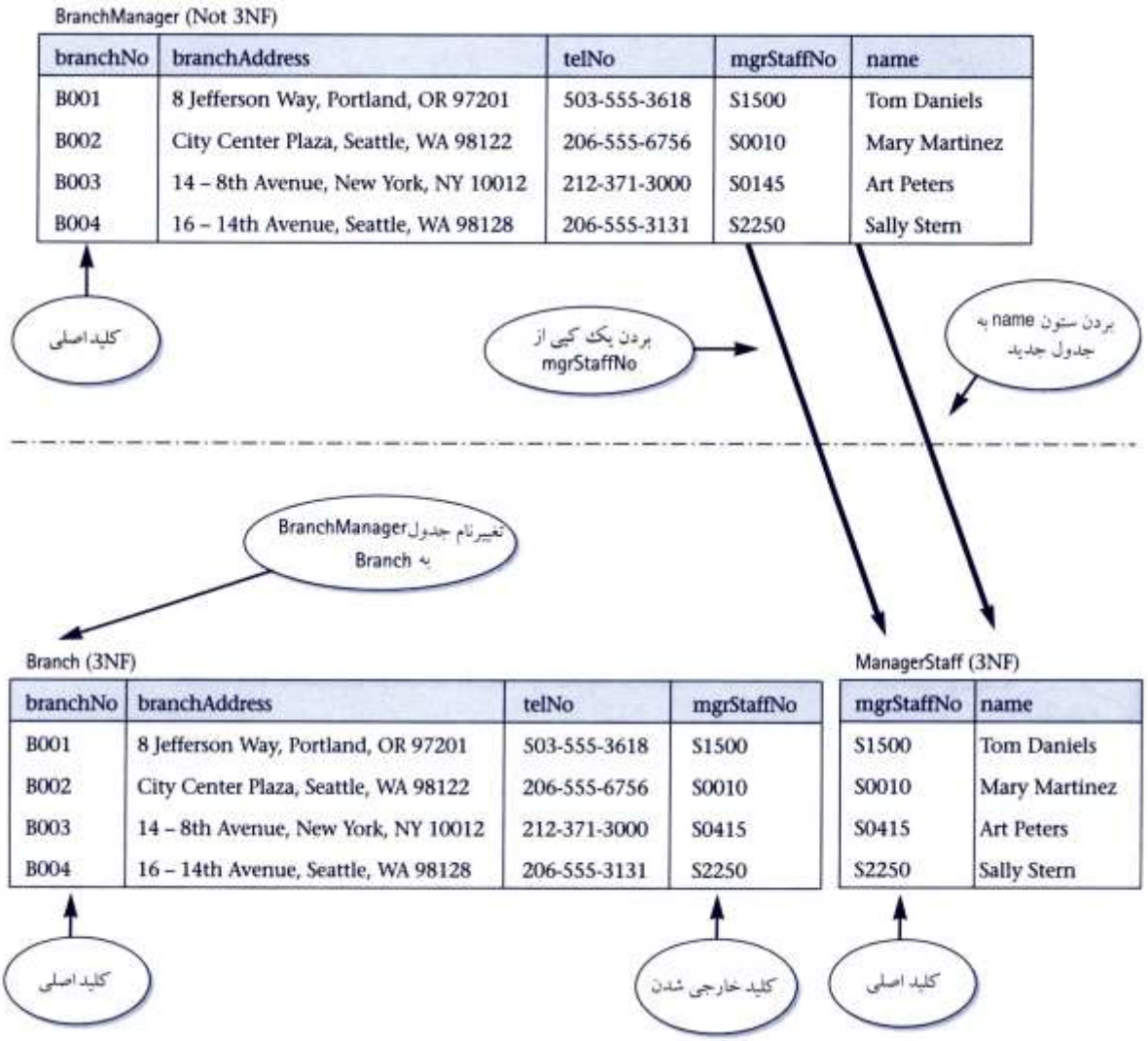
جدول BranchManager که در 3NF نیست.

branchNo	branchAddress	telNo	mgrStaffNo	name
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618	S1500	Tom Daniels
B002	City Center Plaza, Seattle, WA 98122	206-555-6756	S0010	Mary Martinez
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000	S0145	Art Peters
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131	S2250	Sally Stern



شکل ۸-۶

جدول Branch که در 3NF است که با بردن ستون نام بوجود می آید.



خلاصه فصل

- ✓ **نرمالسازی** تکنیکی برای تولید مجموعه ای از جداول با خصوصیات مطلوبی است که نیازمندیهای کاربران یا شرکت را پشتیبانی میکند.
- ✓ جدولهایی که داده افزونه دارند ممکن است مشکلاتی بنام ناسازگاریهای بهنگام سازی داشته باشند، که بصورت ناسازگاریهای درج، حذف، یا اصلاح درجه بندی میشوند.
- ✓ تعریف **فرم نرمال اول (1NF)** جدولی است که اشتراک هر ستون و رکورد تنها و تنها دارای یک مقدار باشد.
- ✓ تعریف **فرم نرمال دوم (2NF)** جدولی است که در حال حاضر در 1NF است و مقادیر هر ستون غیرکلید اصلی میتواند فقط از مقادیر همه ستونهای تشکیل دهنده کلید اصلی بدست آیند.
- ✓ تعریف **فرم نرمال سوم (3NF)** جدولی است که در 1NF و 2NF است، و مقادیر در همه ستونهای غیرکلید اصلی تنها از ستونهای کلید اصلی بدست آیند نه از ستونهای دیگری.

فصل ۷

مروری بر متدلوژی

آنچه در این فصل خواهید آموخت :

- متدلوژی طراحی چیست.
- اهداف متدلوژی طراحی پایگاه داده.
- هدف مدل سازی داده ای.
- طراحی پایگاه داده دو مرحله اصلی دارد : طراحی منطقی و فیزیکی.
- گامهای مربوط به متدلوژی طراحی منطقی و فیزیکی پایگاه داده.

در فصل ۳، ما مراحل اصلی چرخه حیات کاربرد پایگاه داده، که یکی از آن طراحی پایگاه داده است را شرح دادیم. این مراحل تنها بعد از اینکه تحلیل کاملی از نیازمندیهای شرکتی که تقبل شده است شروع می شود، که در فصل ۴ بحث کرده ایم. در فصلهای ۱۶-۸، متدلوژی برای پایگاه داده منطقی و فیزیکی ارائه خواهیم کرد. در این فصل، درباره اهداف متدلوژی طراحی پایگاه داده بحث می کنیم و مرور مختصری بر متدلوژی ارائه شده در این کتاب خواهیم داشت.

۷-۱ مقدمه ای بر متدلوژی طراحی پایگاه داده

هدف این کتاب این است که به شما کمک کند تا پایگاه داده را طراحی کرده و آن را بسازید. با این وجود، زمانیکه پایگاه داده ای که می خواهید شروع کنید پیچیده باشد، در اینصورت به روش سیستماتیکی نیاز خواهید داشت تا طراحی پایگاه داده ای را داشته باشید که نیازمندیهای کاربران و کارائی سیستم (همچون زمانهای پاسخ سیستم) را برآورده سازد. این روش سیتماتیک **متدلوژی طراحی پایگاه داده** نامیده می شود. همانطور که به طور مختصر در فصل ۷-۳ بحث کردیم، متدلوژی بحث شده در این کتاب شامل دو بخش عمده، طراحی منطقی و فیزیکی است، که ما مدلهای داده ای را برای نمایش شرکت می سازیم. قبل از اینکه به مرور متدلوژی بپردازیم، پاسخ به سوالات زیر ممکن است بعضی از نقاطی را که برای شما جالب است را آشکار کند:

(۱) متدلوژی طراحی پایگاه داده چیست ؟
(۲) هدفهای متدلوژی طراحی پایگاه داده چیست ؟
(۳) چرا مدلهای داده ای می سازیم ؟

۷-۱-۱ متدلوژی طراحی پایگاه داده چیست ؟

متدلوژی طراحی پایگاه داده شامل مراحل گام به گامی است، که طراح را راهنمایی می کند تا تکنیکهای مربوط به هر مرحله از پروژه را اجرا کند. این مراحل همچنین به طراح کمک می کنند تا پروژه توسعه پایگاه داده را برنامه ریزی، مدیریت، کنترل، و ارزیابی کند. بعلاوه، ساختار ساخت یافته ای جهت تحلیل و مدلسازی مجموعه نیازمندیهای پایگاه داده به روش استاندارد و سازمان یافته ارائه می کند.

۷-۱-۲ هدفهای متدلوژی طراحی پایگاه داده چیست ؟

اهداف عمده متدلوژی طراحی پایگاه داده بدین صورت است:

- داده و روابط بین داده های موردنیاز تمام نواحی و گروههای کاربر اصلی شرکت را نمایش می دهد (برای مثال، تولید، بازاریابی، کنترل موجودی)؛
- یک مدل داده ای فراهم می کند تا تراکنشهای موردنیاز کاربران بر روی داده ها را پشتیبانی کند؛
- طراحی کمینه ای را مشخص می کند که به خوبی ساخت یافته باشد تا نیازمندیهای کارایی کیفیت سیستم (همچون زمانهای پاسخ) را برآورده سازد.

متاسفانه، خواهید دید که این اهداف معمولا به آسانی قابل دستیابی نیستند، و بعضی اوقات بخصوص برای رسیدن به کارایی قابل قبول سیستم لازم است توافقهایی صورت پذیرد.

روشی که ما برای طراحی پایگاه داده استفاده می کنیم به روش بالا به پایین معروف است. این روش با توسعه مدلهای داده ای آغاز می شود که شامل تعداد کمی موجودیت و رابطه می باشد سپس پالایش پی در پی بالا به پایینی را جهت شناسایی صفات و شاید موجودیتها و رابطه های اضافی اعمال می کنیم. برای مثال، ممکن است:

(۱) موجودیتهای Branch و Staff را شناسایی کنید.

(۲) سپس رابطه بین این موجودیتها را شناسایی کنید، Branch *Has* Staff ;

(۳) بالاخره، صفات مربوط به آنها را شناسایی کنید، مثل:

Branch (branchNo, Street, city, state, zipcode) و Staff (staffNo, name, salary, position) .

روش بالا به پایین استراتژی مناسبی برای طراحی پایگاه داده های ساده و مشکل می باشد، و اساس روشی است که ما در متدلوژی خود استفاده خواهیم کرد.

۷-۱-۳ چرا مدلهای داده ای می سازیم ؟

دو هدف اصلی ساختن مدلهای داده ای عبارت است از:

(۱) کمک به درک معنی داده ها ;

(۲) تسهیل ارتباط درباره نیازمندیهای اطلاعات ;

ساختن مدل داده ای نیازمند پاسخ به سوالاتی درباره موجودیتها، ارتباطات، و صفات می باشد. در رابطه با این، شما معنی داده های موجود در شرکت یا آنهایی که اتفاق نیافتاده اند تا به صورت مدل داده رسمی ثبت شوند را کشف می کنید. موجودیتها، رابطه ها، و صفات برای همه شرکتها بنیادی هستند. با این وجود، تا زمانیکه این داده ها به طور صحیح مستند

نشده باشند معنی آنها به طور ضعیف درک خواهد شد. مدل داده ای امکانی را برای شما فراهم می کند تا معنی داده را درک کنید. بنابراین، شما داده را مدل می کنید تا مطمئن شوید که موارد زیر را درک کرده اید:

- دید کاربران از داده ها؛
- ماهیت خود داده، مستقل از نمایش فیزیکی؛
- استفاده از داده در داخل و بهمراه کاربردها؛

مدلهای داده ای می تواند در رساندن مفهوم نیازمندیهای داده شرکت به شما استفاده شود. دو بخش ارائه شده با نمادهای استفاده شده در مدل آشنا هستند، و ارتباط با کاربران را پشتیبانی خواهند کرد. ما طراحی پایگاه داده را به دو بخش اصلی تقسیم می کنیم :

- **طراحی منطقی پایگاه داده** – برای ساختن نمایش منطقی پایگاه داده، که شامل شناسایی موجودیتها و روابط مهم، و تبدیل این نمایش به مجموعه جدولها می باشد به کار می رود.
- **طراحی فیزیکی پایگاه داده** – تصمیم گیری درباره اینکه چگونه طراحی منطقی به طور فیزیکی (بصورت جدول) در DBMS هدف پیاده سازی می شود.

اگر بخواهیم دقیقتر بیان کنیم، مرحله ای قبل از طراحی منطقی پایگاه داده بنام *طراحی مفهومی پایگاه داده* وجود دارد. طراحی مفهومی پایگاه داده دید کلی از داده هایی را که تنها توسط مدیران سطح بالا قابل دیدن است را ارائه می دهد. مدل مفهومی مبنایی جهت شناسایی و توصیف اشیا اصلی داده بوده و از جزئیات غیر لازم اجتناب می کند. طراحی مفهومی پایگاه داده مستقل از تمام ملاحظات فیزیکی بوده، و شامل مدل زیرین داده ای می باشد. با این وجود، از آنجائیکه پایگاه داده را خصوصا در DBMS های رابطه ای طراحی می کنیم، دو مرحله را با هم ترکیب کرده و عبارت کلی 'طراحی منطقی پایگاه داده' را استفاده می کنیم.

طراحی منطقی پایگاه داده

مرحله اول از طراحی پایگاه داده طراحی منطقی پایگاه داده نامیده می شود و نتیجه آن ایجاد مدل منطقی داده برای شرکت است. مدل منطقی مشتق شده از شناختن مدل داده ای در زیر قرار گرفته شده DBMS هدف است، که آن را فقط در این کتاب مدل رابطه ای می گوییم. بنابراین، شما از هر نوع جنبه دیگر DBMS انتخابی، بخصوص، هر گونه جزئیات فیزیکی، همچون سازمان فایل ها و اندیسها چشم پوشی کنید.

به طور روزافزون، شرکتها در حال استاندارد کردن اینکه چگونه داده را با انتخاب روش مشخصی مدل کنند و از آن در سراسر پروژه های توسعه پایگاه داده خودشان استفاده کنند، می باشند. مدل داده متداول سطح بالای استفاده شده در طراحی پایگاه داده، و همان که ما در این کتاب از آن استفاده می کنیم، بر اساس مفاهیم مدل موجودیت- رابطه (ER) می باشد.

در سراسر مرحله توسعه مدل منطقی داده، دائما سعی کنید مدل را از لحاظ نیازمندیهای کاربر تست و معتبر سازید:

(۱) استفاده از نرمال سازی. نرمالسازی اطمینان می دهد که جدولهای مشتق شده از مدل داده ای افزونگی داده را که ممکن است در موقع پیاده سازی باعث ناسازگاریهای بهنگام سازی شوند نشان نمی دهند.

(۲) مطمئن شوید که مدل منطقی داده ای تراکنشهای مشخص شده توسط کاربر را پشتیبانی می کند.

مدل منطقی داده منبع اطلاعاتی مرحله طراحی فیزیکی است، که با فراهم ساختن وسیله ای جهت سبک سنگین کردن طراحی موثر پایگاه داده اهمیت فراوانی دارد. برای مثال، در مدل منطقی داده ای، ممکن است تشخیص دهید که مقدار یک صفت شاید از مقدار صفت دیگری مشتق شده است. هنگام طراحی فیزیکی، شاید تصمیم بگیرید صفت دوم را ذخیره کرده و

مقدار صفت دیگر را هر وقت که لازم داشتید محاسبه کنید. از طرف دیگر، ممکن است تصمیم بگیرید که هر دو صفت را ذخیره کنید، که نتیجه آن افزایش اندازه پایگاه داده بوده اما دسترسی به صفت موثرتر خواهد بود.

مدل منطقی همچنین هنگام مرحله نگهداری عملیاتی چرخه حیات پایگاه داده نقش مهمی را بر عهده دارد. با نگهداری خوب و بهنگام، خواهید فهمید که مدل داده ای اجازه می دهد تغییرات در آینده بر روی داده یا تراکنشهای موردنیاز بدرستی و موثر روی پایگاه داده صورت پذیرند.

طراحی منطقی پایگاه داده مرحله تکراری است که نقطه شروع داشته و تقریبا می تواند هر چه قدر که لازم باشد روی آن پالایش صورت گیرد. بعد از اینکه عملکرد شرکت و معنی داده های آنرا فهمیدید، و آنچه را که فهمیدید در غالب مدل داده ای انتخابی شرح دادید، اطلاعات بدست آمده ممکن است ایجاب کند که تغییراتی در دیگر قسمتهای طراحی صورت دهید. برای مثال، با شناختن موجودیتهای Branch و Staff و رابطه Branch *Has* Staff ممکن است موجودیت Member و همچنین رابطه Branch *Registers* Member را شناسایی کنید. با این وجود، بعد از کمی تحقیق، ممکن است تشخیص دهید که اطلاعاتی وجود دارد که باید درباره ثبت نام نگهداری شود، همچون تاریخ ثبت نام و اینکه کدام عضو پرسنل عضو جدید را ثبت نام کرده است، و ممکن است رابطه *Registers* موجودیت Registration را تغییر دهید.

> طراحی منطقی پایگاه داده مرحله بحرانی در موفقیت کلی سیستم است. اگر طراحی ارائه صحیحی از شرکت ارائه ندهد، مشکل خواهد بود، اگر چه غیر ممکن نیست، تا تمام **دیدهای موردنیاز کاربر** یا جامعیت نگهداری پایگاه داده را تعریف کنیم. حتی ثابت شده است که تعریف پیاده سازی فیزیکی و یا کارایی قابل قبول سیستم دشوار می شود. اگر طراحی ارائه صحیحی از شرکت را داشته باشد، قادر خواهیم بود سیستم را پیاده سازی کرده و در همان زمان به کارایی قابل قبول برسیم. نشانه دیگری از طراحی خوب پایگاه داده توانایی اعمال تغییر است. بنابراین، ارزش تلاش و صرف وقت را دارد که بهترین طراحی ممکن را تولید کنیم.

طراحی فیزیکی پایگاه داده

طراحی فیزیکی پایگاه داده دومین مرحله از فرایند طراحی پایگاه داده است، که در این مرحله شما تصمیم می گیرید که چگونه پایگاه داده پیاده سازی شود. مرحله قبلی طراحی پایگاه داده شامل توسعه ساختار منطقی پایگاه داده بود (یعنی، شناسایی موجودیتها، روابط، و صفاتی که موردنیاز است). اگرچه این ساختار مستقل از DBMS است، در این کتاب جهت مدل رابطه ای توسعه یافته است، و بنابراین چنین ساختار مشخصی که می دانیم نمی تواند در پایگاه داده رابطه ای نشان داده شود را حذف کرده ایم. با این وجود، قبل از توسعه طراحی فیزیکی پایگاه داده، شما اول باید DBMS هدف را بشناسید. بنابراین، طراحی فیزیکی مناسب سیستم DBMS خاصی همچون، مایکروسافت اکسس ، اوراکل ، یا SQL server می باشد. بین طراحی فیزیکی و منطقی بازخورد وجود دارد، زیرا تصمیمات مطرح شده در مرحله طراحی فیزیکی جهت بهبود کارایی ممکن است به ساختار مدل منطقی داده تاثیر بگذارد.

به طورکلی، هدف اصلی طراحی فیزیکی پایگاه داده این است که توصیف کند چگونه شما قصد دارید به طور فیزیکی طراحی منطقی پایگاه داده را پیاده سازی کنید. برای مدل رابطه ای، این شامل موارد زیر است:

- ایجاد جدول در DBMS رابطه ای هدف و پیاده سازی محدودیتها بر روی این جداول از طریق داده های ارائه شده در مدل داده منطقی سراسری.
- شناسایی سازمان فایل و اندیس های معین برای داده ها جهت رسیدن به کارایی مطلوب سیستم پایگاه داده .
- در نظر گرفتن ساختار جداول مبنا برای شناسایی اینکه آیا کم کردن قواعد نرمال سازی تاثیری در بهبود کارایی می تواند داشته باشد.
- طراحی حفاظ امنیتی برای سیستم.

۷-۱-۴ فاکتورهای مهم موفقیت در طراحی پایگاه داده

راهنمایی هایی که در زیر آمده است در موفقیت طراحی پایگاه داده اهمیت زیادی دارند:

- تا آنجائیکه امکان دارد بصورت محاوره ای با کاربران در ارتباط باشید.
- از متدلوژی ساخت یافته در سراسر مرحله مدلسازی داده پیروی کنید.
- روش مبتنی بر داده را به کار گیرید.
- ملاحظات جامعیتی و ساختاری را در داخل مدلهای داده با هم ترکیب کنید.
- تکنیکهای نرمالسازی و معتبرسازی تراکنش را داخل متدلوژی مدلسازی داده با هم ترکیب کنید.
- تا آنجائیکه امکان دارد از نمودارهای بیشتری برای نمایش مدلهای داده ای استفاده کنید.
- از زبان طراحی پایگاه داده (DBDL) برای نمایش اطلاعات اضافی درباره داده ها که توسط مدل قابل نمایش نیست استفاده کنید.
- دیکشنری داده را بعنوان مکمل نمودارهای مدل داده ای بسازید.
- حاضر باشید که مراحل را تکرار کنید.

تمام این راهنمایی ها در متدلوژی که ما در بخش بعد و فصلهای بعدی ارائه خواهیم داد پیاده سازی خواهند شد. زمانیکه شما این فصلها را بخوانید، ممکن است بخواهید که به عقب برگردید و با مرور این فاکتورها خواهید دید که چگونه در متدلوژی طراحی پایگاه داده مفید می باشند .

۷-۲ مروری بر متدلوژی طراحی پایگاه داده

در این بخش، متدلوژی طراحی پایگاه داده را مرور می کنیم. گامهای متدلوژی در شکل ۱-۷ نشان داده شده اند، و فصلهایی که هر مرحله در آن فصل بحث شده در ستونهای مجاور آمده است.

طراحی منطقی پایگاه داده به سه قسمت عمده تقسیم شده است :

- ***مرحله ۱*** طراحی را به چندین کار قابل مدیریت تجزیه می کند، که بوسیله بررسی دیدهای مختلف کاربران شرکت صورت می پذیرد. خروجی این مرحله ایجاد **مدلهای داده ای منطقی محلی** است، که کامل هستند و نمایش صحیحی از شرکتی که توسط گروهای مختلف کاربری دیده می شود را ارائه می کنند.
- ***مرحله ۲*** مدلهای منطقی داده را به مجموعه ای از جداول نگاشت می کند. بعلاوه، مدلهای داده ای با استفاده تکنیکهای نرمالسازی معتبر می شوند. نرمالسازی معنای موثر اطمینان از این است که مدلها از لحاظ ساختار سازگار و منطقی بوده و کمترین افزونگی را دارند. مدلهای داده همچنین در برابر تراکنشهایی که نیاز به پشتیبانی دارند معتبر می شوند. معتبر سازی فرآیند اطمینان از این است که شما مدل 'صحیحی' را ایجاد می کنید.
- ***مرحله ۳*** شامل یکپارچگی مدلهای منطقی داده محلی (که دیدهای کاربر متفاوتی را نشان می دهد) برای فراهم آوردن **مدل داده منطقی سراسری** واحد (که همه دیدهای کاربر را نمایش می دهد) می باشد. این مدل سراسری مانند قبل معتبر می شود تا اطمینان دهد که از لحاظ ساختمان صحیح بوده و تراکنشهای موردنیاز را پشتیبانی می کند.

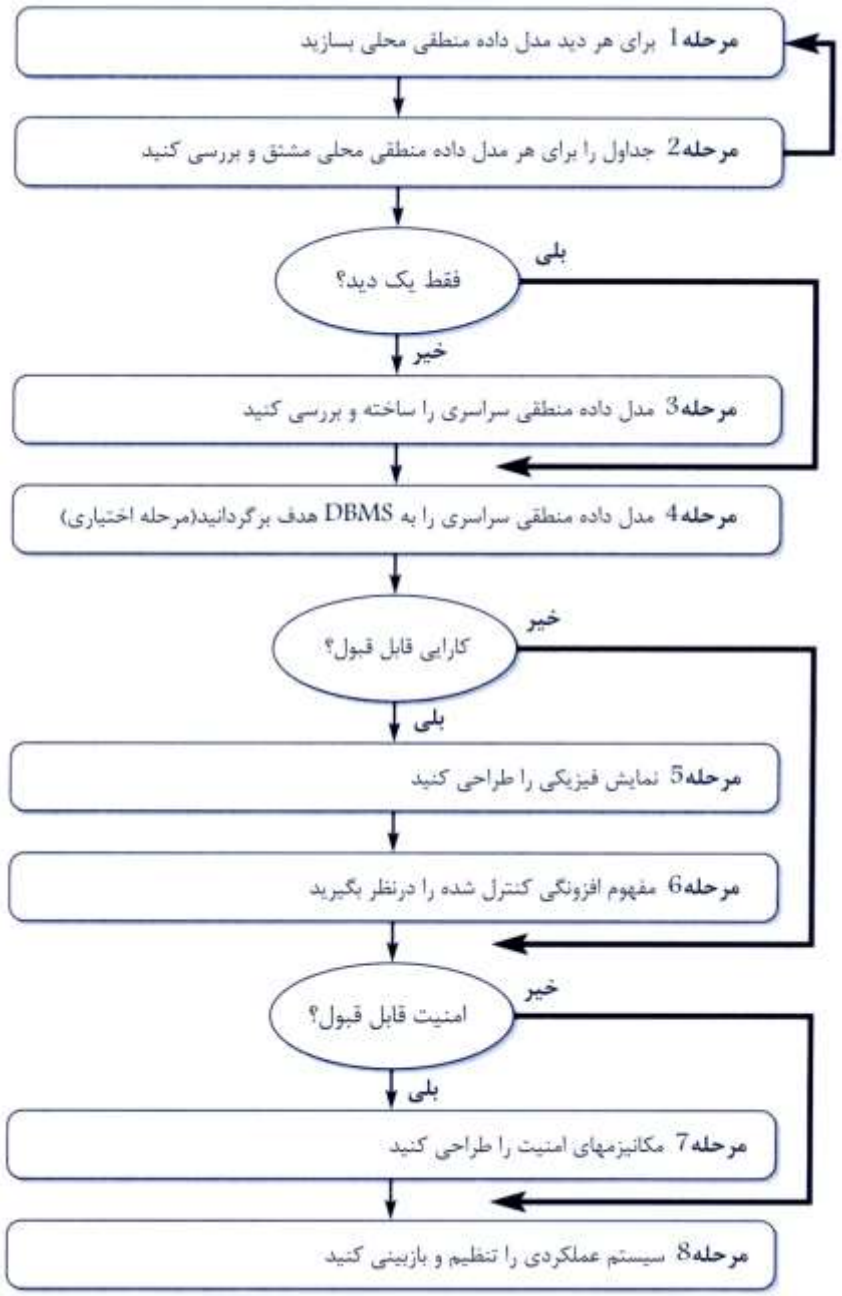
طراحی فیزیکی پایگاه داده به چهار مرحله اصلی تقسیم می شود:

- *مرحله ۴* شامل طراحی جداول پایه و محدودیتهای جامعیت با استفاده از کارایی موجود DBMS هدف می باشد.
- *مرحله ۵* شامل انتخاب سازمان فایل و اندیسها برای جداول پایه است. به طور نمونه، DBMS ها انواع متعددی از سازمان فایل پیشنهادی برای داده ارائه می دهند، به استثناء DBMS های کامپیوتر شخصی، که میل دارند بیشتر ساختار ذخیره سازی ثابتی داشته باشند.
- *مرحله ۶* کم کردن محدودیتهای نرمالسازی تحمیل شده بر روی جدولها را در نظر می گیرد که به خاطر بهبود کارایی کلی سیستم می باشد. این مرحله ای است که شما باید در صورت لزوم آن را برعهده بگیرید.
- *مرحله ۷* شامل طراحی مقیاسهای امنیتی برای حفاظت از دستیابی غیر مجاز می باشد. این بدین معنی است که تصمیم می گیرید که چگونه هر دید کاربر بایستی پیاده سازی شوند، و چه کنترلهای دسترسی روی جداول پایه مورد نیاز است.
- *مرحله ۸* فرایند مداومی از نظارت روی سیستم عملکردی است که برای شناسایی و برطرف کردن هر گونه مشکلات کارایی که در نتیجه طراحی بوجود میاید صورت می پذیرد .

ضمیمه (ب) خلاصه ای از متدلوژی را برای آندسته از خوانندگانی که در حال حاضر با طراحی پایگاه داده آشنا بوده و صرفا به خلاصه ای از مراحل اصلی نیاز دارند ارائه می کند. در سرتاسر متدلوژی، کاربران نقش حساسی را در بازبینی مداوم و معتبر سازی مدل داده ای و همچنین پشتیبانی از مستندات بر عهده دارند.

بعضی مراحل با توجه به پیچیدگی شرکتی که تحلیل می کنید مورد نیاز نمی باشند. شکل ۲-۷ نشان میدهد که کدام مراحل ممکن است با وابسته بودن به نیازمندیهای ویژه ای حذف شود .

نکته طراحی پایگاه داده فرایند تکراری است که اغلب دارای نقطه شروع و پیشرفت بی پایانی از پالایش می باشد. اگرچه مادر اینجا آنرا بصورت فرایند رویه مانند نشان می دهیم، باید تاکید شود که دلیلی بر این نیست که طراحی حتما باید بدین شیوه انجام گیرد. آن احتمالا معلوماتی است که که شما در یک مرحله بدست آورده اید و ممکن است تصمیماتی را که در مرحله قبلی گرفته اید را تغییر دهد. متدلوژی باید بعنوان چارچوبی باشد که به شما کمک کند که فعالیت طراحی پایگاه داده را به طور موثر انجام دهید.



شکل ۲-۷

پیشرفت از طریق متدلوژی.

خلاصه فصل

- ✓ **متدلوژی طراحی** روش ساخت یافته ای است که از رویه ها، تکنیکها، ابزار ، و مستندات کمکی استفاده می کند تا فرایند طراحی را پشتیبانی و تسهیل کند.
- ✓ هدف اصلی متدلوژی طراحی پایگاه داده نمایش تمام داده های موردنیاز و روابط بین داده هاست; برای فراهم کردن مدل داده که تراکنشهای موردنیاز را پشتیبانی کند. و معین کردن طراحی کمینه ای که به طور مناسب ساخت یافته بوده تا به نیازمندی های کارایی دست یابد.
- ✓ دو هدف اصلی مدلسازی داده کمک به فهمیدن معانی داده ها و تسهیل ارتباط درباره نیازمندیهای اطلاعاتی است.
متدلوژی طراحی پایگاه استفاده شده در این کتاب دو مرحله اصلی دارد: طراحی منطقی و فیزیکی پایگاه داده.

- ✓ **طراحی منطقی پایگاه داده** فرایند ساخت مدلی اطلاعات استفاده شده در شرکت بر اساس مدل داده خاصی است، اما مستقل از DBMS خاص و دیگر ملاحظات فیزیکی می باشد. در مورد ما، طراحی منطقی همان مدل رابطه ای است.
- ✓ **طراحی فیزیکی پایگاه داده** فرایند تولید توصیف پیاده سازی پایگاه داده در محل ذخیره سازی جانبی است. که سازمان فایل و اندیسهایی که برای دستیابی موثر به داده ها استفاده می شود را توصیف می کند، و همچنین هر گونه محدودیتهای جامعیت و امنیت مربوطه را نیز توصیف می کند. طراحی فیزیکی همان طراحی است که روی DBMS خاصی انجام می گیرد.
- ✓ بازخوردی میان طراحی منطقی و فیزیکی وجود دارد، زیرا تصمیمات گرفته شده جهت بهبود کارایی در طراحی فیزیکی ممکن است به ساختار مدل داده منطقی تاثیر بگذارد.
- ✓ فاکتورهای مهمی برای موفقیت مرحله طراحی پایگاه داده وجود دارند، برای مثال، به طور محاوره ای کارکردن با کاربران و تمایل داشتن به اینکه مراحل تکرار شود.

فصل ۸

طراحی منطقی پایگاه داده – مرحله ۱

آنچه در این فصل خواهیم آموخت :

- چگونه محدوده طراحی را به دیدهای مشخصی از شرکت تجزیه کنیم.
- چگونه از مدل ER برای ساختن مدل داده منطقی برای هر دید استفاده کنیم.
- چگونه مرحله طراحی منطقی پایگاه داده را مستند کنیم.
- کاربران در سرتاسر مرحله طراحی منطقی پایگاه داده نقش کاملی را بازی می کنند.

این فصل مرحله اول متدلوژی طراحی منطقی پایگاه داده را می پوشاند. در این مرحله، شما برای دیدهایی که در مراحل قبلی مشخص شده است مدل داده منطقی محلی می سازید.

شما هنگام تحلیل، تعدادی از دیدهای کاربر را شناسایی خواهید کرد، و بر اساس تعداد همپوشانی بین این دیدها، ممکن است چندین دید را با هم ترکیب کنید. در مرحله تحلیل و جمع آوری نیازمندیها که در بخش ۴-۴-۴ بحث شد، ما دو دید ادغام شده را برای *StayHome* شناسایی کردیم :

- **Branch**، که شامل دیدهای کاربر مدیر، ناظر، و معاون بود;
- **Business**، که شامل دیدهای کاربر کارگردان و خریدار بود.

در این فصل، برای دید شعبه *StayHome* مدل داده منطقی محلی می سازید، و در فصل ۱۰ نشان خواهیم داد که چگونه مدلهای داده منطقی محلی را برای دیدهای Branch و Business با هم ادغام کنیم تا یک مدل داده منطقی سراسری برای شرکت تولید کنیم.

مرحله ۱ برای هر دید مدل داده منطقی محلی بسازید

هدف

ساختن مدل داده منطقی محلی شرکت برای هر دید.

ما مدل داده منطقی هر دید را **مدل داده منطقی محلی** آن دید می نامیم. هر مدل داده منطقی محلی شامل موارد زیر است:

- موجودیتها،
- رابطه ها،
- صفات و دامنه صفات،
- کلیدهای کاندید، کلیدهای اصلی، و کلیدهای فرعی ،
- محدودیتهای جامعیت .

مدل داده منطقی توسط مستندات پشتیبانی می شود، که این مستندات شامل دیکشنری داده است، که در سرتاسر توسعه مدل آنرا تولید خواهید کرد. در اینجا جزئیات مستندات پشتیبانی کننده ای را شما قصد دارید تولید کنید از طریق مراحلی توصیف می کنیم. کارهایی که به مرحله ۱ مربوط می شوند به صورت زیر هستند:

- مرحله ۱,۱ موجودیتها را شناسایی کنید
- مرحله ۱,۲ روابط را شناسایی کنید
- مرحله ۱,۳ صفات مرتبط با موجودیتها یا روابط را شناسایی کنید
- مرحله ۱,۴ دامنه های صفات را تعیین کنید
- مرحله ۱,۵ صفات کلید کاندید و اصلی را تعیین کنید
- مرحله ۱,۶ موجودیتها را عمومی/ اختصاصی کنید (مرحله اختیاری)
- مرحله ۱,۷ خصیصه هایی را که با مدل رابطه ای سازگار نیستند حذف کنید
- مرحله ۱,۸ مدل را از نظر پشتیبانی تراکنشهای کاربر بررسی کنید.

هم اکنون، اجازه دهید ساختن مدل دید شعبه را شروع کنیم.

مرحله ۱,۱ موجودیتها را شناسایی کنید

هدف

شناسایی موجودیتهای اصلی که توسط دید موردنیاز است.

مرحله اول ساختن مدل داده منطقی محلی تعریف اشیا اصلی است که کاربران به آن علاقمند هستند. این اشیا موجودیتهای مدل هستند. یک روش شناسایی موجودیتها بررسی خصوصیات نیازمندیهای کاربران است. از این خصوصیات، میتوانید اسمها و عبارت اسمی را در بیاورید (برای مثال، شماره پرسنل، نام پرسنل، شماره کاتالوگ، عنوان، نرخ اجاره روزانه، قیمت خرید). شما همچنین باید به اشیا اصلی مثل افراد، مکانها، یا چیزهای مورد علاقه نگاه کنید، بجز آن اسمهایی که صرفا کیفیت دیگر اشیا را نشان می دهند.

برای مثال، می توانید نام پرسنل و شماره پرسنلی را با موجودیت بنام پرسنل گروه بندی کنید، و شماره کاتالوگ، عنوان، و نرخ اجاره روزانه، و قیمت خرید را با موجودیتی بنام فیلم گروه بندی کنید.

راه دیگر شناسایی موجودیتها نگریستن به آن اشیایی است که در اصلیت خود وجود دارند. برای مثال، پرسنل موجودیت است زیرا پرسنل وجود دارد یا نه، و شما نام، آدرس، و حقوق آن را می دانید. اگر لازم باشد، شما باید از کاربران برای کمک در این امر یاری بگیرید.

بعضی اوقات شناسایی موجودیتها بخاطر نحوه نشان دادنشان در خصوصیات نیازمندیهای کاربر مشکل است. کاربران اغلب در عبارات مثال و مقایسه حرف می زنند. بجای صحبت درباره پرسنل در حالت کلی، کاربران ممکن است نام آنها را بکار برند. در بعضی موارد، کاربران در رابطه با وظایف شغلی صحبت می کنند، خصوصا جائیکه شامل مردم یا شرکتها باشند. این وظایف ممکن است عناوین شغل یا مسئولیتها، همچون مدیر، جانشین مدیر، ناظر، یا معاون باشد. در موارد گیج کننده تر نیز ممکن است کاربران مکررا از کلمات مشابه و مترادف استفاده کنند.

دو کلمه *مترداف* هستند وقتی معانی مشابهی داشته باشد، برای مثال 'شعبه' و 'فروشگاه' مترادف هستند. ولی کلمات متشابه می تواند با توجه به متن معانی مختلف داشته باشد. برای مثال، کلمه 'برنامه' چندین معنی مختلف دارد مانند یک سری رخداد، برنامه کاری، قسمتی از نرم افزار، و برنامه درسی.

همیشه واضح نیست که آیا شی مشخصی موجودیت است، رابطه است، یا صفت است. برای مثال، چگونه می توانید ازدواج را مدل کنید؟ در حقیقت، با تکیه بر نیازمندیهای واقعی می توانید ازدواج را بعنوان هیچ یا تمام اینها در نظر بگیرید. متوجه خواهید شد که تحلیل یک امر فردی است، و طراحان متفاوت ممکن است تفسیرهای متفاوت، اما بطور مشابه صحیح، تولید کنند. بنابراین عمل تحلیل تکیه بر تجربه شخصی طراح دارد. طراحان پایگاه داده باید بهترین دید را از جهان داشته باشند و چیزهایی را که مشاهده می کنند در زمینه شرکت بخوبی طبقه بندی کنند. بنابراین، هیچ مجموعه مشخصی از موجودیتهایی که از خصوصیات نیازمندیهای کاربر بتوانند استنباط شوند وجود ندارد. با این وجود، تکرار موفق مرحله تحلیل باید منجر به انتخاب موجودیتهایی شود که حداقل برای سیستم کافی باشد.

نکته این حقیقت که طراحی پایگاه داده یک امر ذهنی است در ابتدا می تواند نامفهوم باشد. با این وجود، با پیروی از متدلوژی ارائه شده در این کتاب، خواهید یافت که این کار دست یافتنی بوده و با کمی تمرین و تجربه آسانتر نیز میشود. جهت کمک، در فصلهای ۱۸ و ۱۹ ما به دومین بررسی موردی می پردازیم، و در ضمیمه (د) مدلهای داده تجاری متداول را فراهم کرده ایم که با توجه به سلیقه تان میتوانید آنها را بررسی کنید.

موجودیتهای StayHome

در بررسی موردی *StayHome*، احتمالا می توانید موجودیتهای زیرین را شناسایی کنید:

Branch	Staff
Video	VideoForRent
Member	RentalAgreement
Actor	Director

موجودیتها را مستند کنید

بعد از اینکه موجودیتها را مشخص کردید، به آنها نامهایی که برای کاربر با معنا و واضح باشد اختصاص دهید. نامها و توصیفات موجودیتها را در دیکشنری داده ثبت کنید. در صورت امکان، تعداد رخداد مورد انتظار هر موجودیت را نیز مستند کنید. اگر

موجودیت با نامهای متفاوتی شناخته می شود، شما باید مترادف و **نام مستعار**[1] آنرا نیز در دیکشنری داده ثبت کنید. شکل ۱-۸ خلاصه ای از دیکشنری داده که موجودیتهای *StayHome* را مستند کرده است را نشان می دهد.

شکل ۱-۸

خلاصه ای از دیکشنری داده برای *StayHome* که توصیف موجودیتها را نشان می دهد.

نام موجودیت	توصیف	نام مستعار	رخداد
Branch	محل کار	بازار فروش و بازار فروش شعبه	یک یا چند شعبه *StayHome* که در شهرهای اصلی ایالات متحده قرار دارند.
Staff	عبارت کلی که پرسنل به کار گرفته شده توسط *StayHome* را توصیف می کند.	کارمند	هر عضو پرسنل در شعبه بخوصی کار می کند.

مرحله ۲-۱ رابطه ها را شناسایی کنید.

هدف

شناسایی رابطه های مهم موجود بین موجودیتهایی که شناسایی کرده اید.

پس از شناسایی موجودیتها، مرحله بعدی مشخص کردن تمام رابطه هایی است که بین این موجودیتها وجود دارد. زمانیکه موجودیتها را شناسایی می کنید، یک روش، جستجو اسمهایی است که در خصوصیات نیازمندیهای کاربر موجود می باشد. دوباره، می توانید از نحو خصوصیات نیازمندیها استفاده کنید تا رابطه ها را نیز شناسایی کنید. به طور نمونه، رابطه ها توسط فعلها یا عبارات فعلی مشخص می شوند. برای مثال:

- Branch *Has* Staff
- Branch *IsAllocated* VideoForRent
- VideoForRent *IsPartOf* RentalAgreement

این حقیقت که خصوصیات نیازمندیهای کاربر این رابطه ها را ثبت میکند اشاره بر این دارد که آنها برای کاربران مهم هستند، و بایستی در مدل ذکر شوند.

[1] Aliases

نکته ما تنها به رابطه های مورد نیاز بین موجودیتها علاقمند هستیم. در مثال قبلی، شما رابطه های **Branch** *IsAllocated* **VideoForRent** و **RentalAgreement** *IsPartOf* **VideoForRent** را شناسایی کردید. ممکن بود مایل بودید رابطه بین موجودیتهای Branch و RentalAgreement را نیز در نظر بگیرید. (برای مثال، **Branch** *Handles* **RentalAgreement**). بنابراین، اگرچه این رابطه نیز امکان دارد، ولی با توجه به نیازمندیها، رابطه ای است که ما علاقه ای به مدل کردن آن نداریم. این جنبه را در مرحله ۶-۱ بحث می کنیم.

مواظب باشید و اطمینان حاصل کنید که تمام رابطه های چه آشکار و چه پنهان موجود در خصوصیات نیازمندیهای کاربر را مورد توجه قرار دهید. در اصل، بایستی ممکن باشد که هر جفت موجودیت را بتوان از لحاظ رابطه بین آنها بررسی کرد، اما این خود می تواند برای سیستمهای بزرگ که از صدها موجودیت تشکیل شده اند کاری خطرناک باشد. از طرف دیگر، غیر عاقلانه است که چنین بررسی انجام نشود. اگرچه، رابطه گم شده زمانیکه مدل را از لحاظ پشتیبانی تراکنشهای کاربر بررسی می کنید آشکار می شود.

در بیشتر موارد، رابطه شما باینری خواهد بود; بعبارت دیگر، رابطه ها دقیقا بین دو موجودیت وجود دارند. به هر حال، بایستی به رابطه های پیچیده ای که ممکن است شامل بیش از دو موجودیت باشند و همچنین دارای رابطه های بازگشتی با یک موجودیت باشند نیز دقت داشته باشید. برای *StayHome* ، باید رابطه های غیر باینری زیر را شناسایی کنید:

Registers رابطه سه گانه میان Branch، Member، و Staff.

Supervises رابطه بازگشتی بین Staff.

شکل ۲-۸

اولین پیش نویس روابط *StayHome*.

موجودیت	رابطه	موجودیت
Branch	Has	Staff
	IsAllocated	VideoForRent
Branch, Staff†	Registers	Member
Staff	Manages	Branch
	Supervises	Staff
Video	Is	VideoForRent
VideoForRent	IsPartOf	RentalAgreement
Member	Requests	RentalAgreement
Actor	PlaysIn	Video
Director	Directs	Video

† رابطه سه گانه را نشان میدهد.

رابطه های StayHome

برای بررسی موردی *StayHome* ، ممکن است رابطه های نشان داده شده در شکل ۲-۸ را تشخیص دهید.

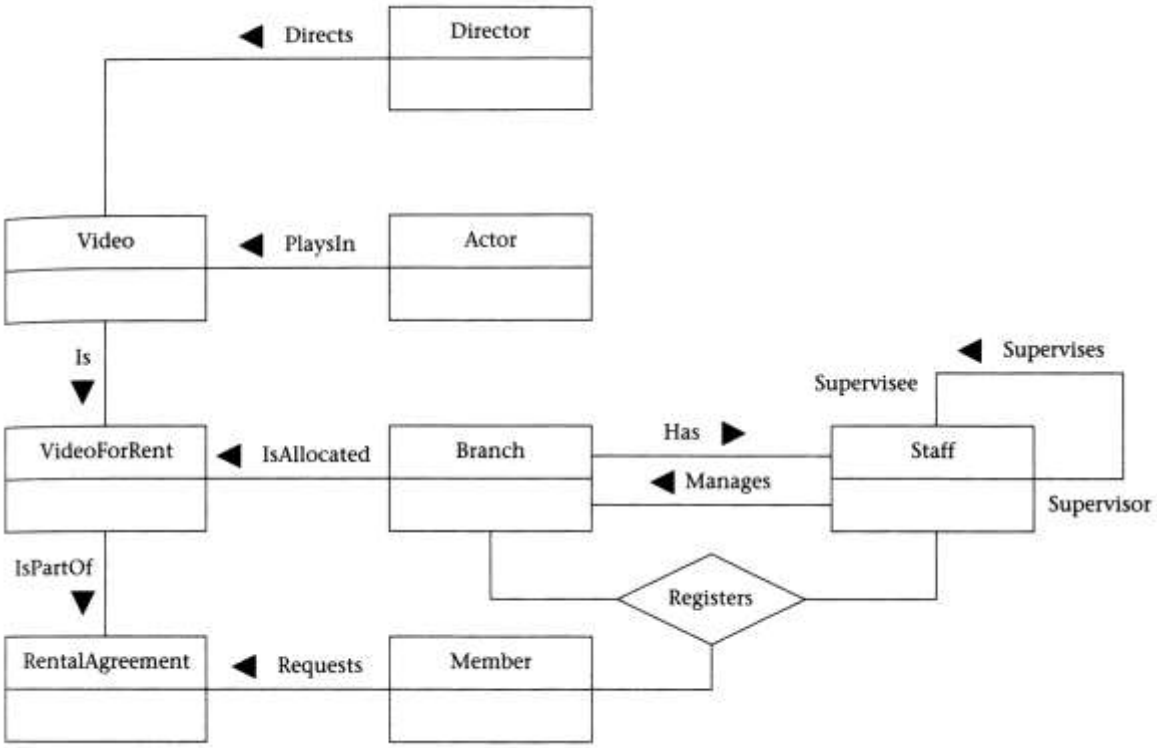
از مدلسازی رابطه- موجودیت (ER) استفاده کنید

اغلب اوقات مصور ساختن سیستم پیچیده آسانتر از نشان دادن بصورت توصیف متنی چنین سیستمی می باشد. استفاده از نمودارهای رابطه- موجودیت (ER) به شما کمک می کند که به آسانی موجودیتها و اینکه چگونه به هم مربوط شده اند را نشان دهید. شما می توانید چنین موجودیتها را در طرح اول همانطور که در شکل ۳-۸ نشان داده شده نشان دهید.

در سراسر مرحله طراحی پایگاه داده، پیشنهاد می کنیم که مدلسازی ER وقتی که لازم باشد استفاده شود، تا به شما کمک کند بتوانید تصویری از آنچه می خواهید داشته باشید را مدل کنید. افراد متفاوت از نمادهای متفاوت در مدلسازی ER استفاده می کنند. در این کتاب، ما از آخرین نماد گذاری شی گرایی که UML خوانده می شود استفاده می کنیم، اما دیگر نمادگذاریها نیز همان عمل مشابه را انجام می دهند.

شکل ۳-۸

اولین طرح مدل ER برای *StayHome* که موجودیتها و رابطه ها را نشان می دهد.



محدودیتهای کثرت رابطه ها را مشخص کنید.

بعد از شناسایی رابطه هایی که می خواهید مدل کنید، هم اکنون باید کثرت هر رابطه را مشخص کنید. اگر مقدار مشخصی هر کثرت را می دانید، یا حتی محدوده بالا یا پایین را، بخوبی این مقادیر را مستند کنید. مدلی که شامل محدودیتهای کثرت است آشکارتر معنای رابطه ها را نشان می دهد و در نتیجه نمایش بهتری از آنچه میخواهید مدل کنید را بیان می کند. محدودیتهای کثرت جهت بررسی و نگهداری کیفیت داده ها استفاده می شوند. این محدودیتها می توانند وقتی اعمال شوند که پایگاه داده بروز شده باشد و یا اینکه بهنگام سازی قواعد مقرر شده را نقض نکند.

نکته اگر رابطه های بازگشتی را با کمترین کثرت برای هر نقش شناسایی کردید، محدودیتهای کثرت را دوباره بررسی کنید، چون این می تواند یک موقعیت غیر معمولی ایجاد کند.

محدودیتهای کثرت *StayHome*

در بررسی موردی *StayHome* ، باید محدودیت های کثرت نشان داده شده در شکل ۴-۸ را شناسایی کنید.
شکل ۵-۸ مدل ER بهنگام شده با اطلاعات افزوده شده را نشان می دهد.

fan trap و chasm trap را بررسی کنید

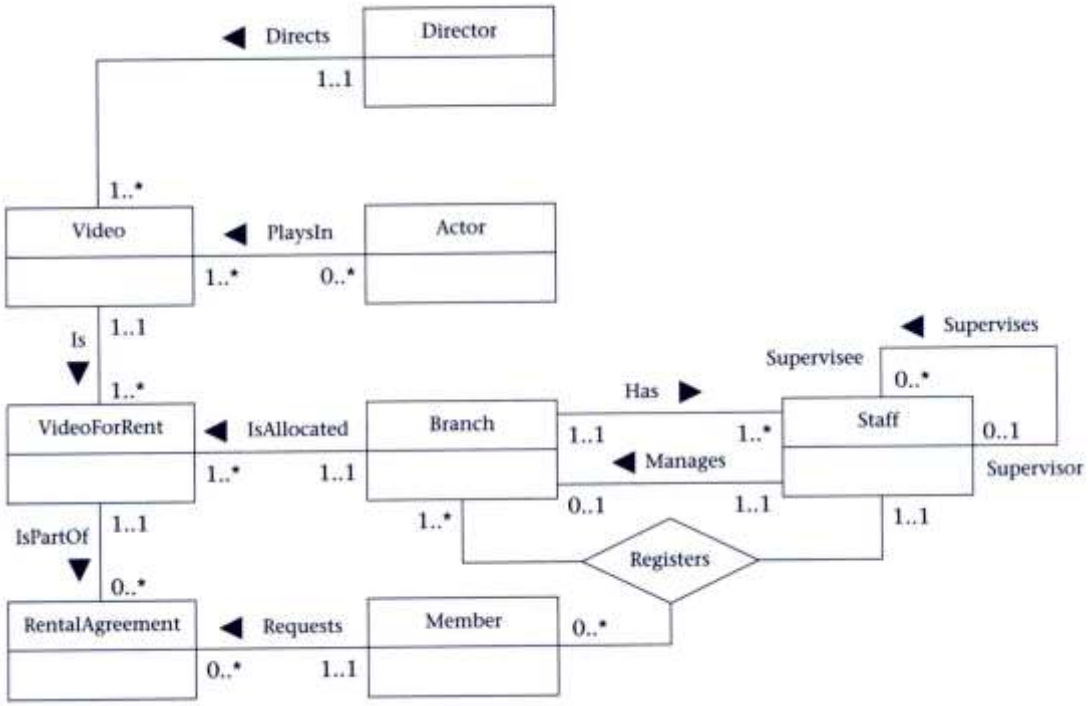
بعد از شناسایی رابطه ها، حال باید هر یک از رابطه ها را به طور صحیح بررسی کنید تا ببینید که آیا آنچه را که می خواهید نمایش می دهند و آیا به طور غیر عمدی تله های fan و chasm را ایجاد نکرده باشید.

بررسی کنید تا هر موجودیت حداقل در یک رابطه شرکت کرده باشند

عموما، موجودیت نمی تواند بدون وابستگی به دیگر موجودیتها مدل شود، در غیر اینصورت زمانیکه موجودیت را به جدولی در مرحله ۱-۲ نگاشت می کنید، راهی برای رجوع به آن جدول وجود ندارد. استثنا عمومی این قاعده، پایگاه داده ای با یک جدول است. اگر چنین موجودیتی داشتید، بررسی کنید تا موجودیت دیگری با نام متفاوت در جای دیگری موجود نباشد. در غیر اینصورت، دوباره به نیازمندیها رجوع کرده تا مشخص کنید که رابطه ای را گم نکرده باشید. اگر رابطه گم شده را شناسایی نکردید، به کاربران رجوع کنید تا درباره آن موجودیت ویژه با جزئیات بیشتر بحث کنید.

شکل۵-۸

افزودن محدودیتهای کثرت به مدل ER برای *StayHome*



رابطه ها را مستند کنید

بعد از اینکه رابطه ها را شناسایی کردید، به آنها نامهایی که به کاربر با معنی و واضح باشد اختصاص دهید. همچنین توصیفات رابطه ها و محدودیتهای کثرت را در دیکشنری داده ثبت کنید. شکل ۶- ۸ خلاصه ای از دیکشنری داده که رابطه ها برای *StayHome* مستند شده است را نشان می دهد.

مرحله ۳-۱ صفات مرتبط با هر موجودیت یا رابطه را شناسایی کنید

هدف

مربوط ساختن صفات با موجودیتها یا رابطه های مناسب.

مرحله بعدی متدلوژی شناسایی حقیقتها درباره موجودیتها و رابطه هایی است که برای نمایش در پایگاه داده انتخاب کرده اید. به طور مشابه جهت شناسایی موجودیتها، به اسمها و عبارات اسمی در ویژگی های خصوصیات کاربر نگاه کنید. صفات می تواند بدین صورت شناسایی شود که اسم یا عبارت اسمی مشخصه، کیفیت، شناسه، یا خصوصیاتی از موجودیت یا رابطه ای باشد که قبلا پیدا شده است.

شکل ۶-۸

خلاصه ای از دیکشنری داده StayHome که توصیف داده ها را نشان می دهد.

موجودیت	کثرت	رابطه	موجودیت	کثرت
Branch	۱..*	Has	Staff	۱..۱
Branch	۱..*	IsAllocated	VideoForRent	۱..۱
Staff	۰..۱	Manages	Branch	۱..۱
Staff	۰..*	Supervises	Staff	۰..۱

صفات ساده / ترکیبی

مهم است که توجه داشته باشید که صفت ساده است یا ترکیبی. صفت ترکیبی از صفات ساده تشکیل شده است. برای مثال، صفت address را هم می توان صفت ساده در نظر گرفت که جزئیات آدرس را به صورت مقدار واحد در نظر می گیرد
همچون '8 Jefferson Way, Portland, OR, 97201'. در عین حال، صفت آدرس را میتوان صفت ترکیبی نیز در نظر گرفت که از صفات ساده خیابان ('8 Jefferson Way')، شهر('Portland') ، منطقه ('OR')، و کدپستی ('97201') تشکیل شده اند.

گزینه ای که آیا جزئیات آدرس را صفت ترکیبی یا ساده در نظر بگیریم توسط نیازمندیهای کاربر مشخص می شود. اگر کاربران به اجزا مجزای آدرس نیازی نداشته باشند، شما باید آدرس را بصورت ساده نشان دهید. از سوی دیگر، اگر کاربران نیاز دارند تا به هر قسمت صفت **address** دسترسی داشته باشند بایستی صفت **address** را بصورت صفات ترکیبی تشکیل شده از صفات ساده نشان دهید.

صفات تک / چند- مقداره

علاوه بر ساده یا ترکیبی بودن، صفت همچنین میتواند تک مقداره یا چند مقداره باشد. بیشتر صفاتی که مواجه می شوید تک مقداره خواهند بود، اما بعضی اوقات ممکن است با صفت چند مقداره نیز مواجه شوید: یعنی، صفتی که چندین مقدار را برای رخداد موجودیت واحد نگه می دارد. برای مثال، ممکن است صفت telNo(شماره تلفن)، Branch را بصورت صفت چند مقداره در نظر بگیرید.

ممکن است شماره تلفن شعبه را بصورت موجودیت مجزا شناسایی کنید. این راه دیگر، یا معتبر مشابهی، برای مدل کردن می باشد. همانطور که مختصرا در مرحله ۷-۱ خواهید دید، صفات چند مقداره به هرحال به موجودیتها نگاشت شده اند، بنابراین هر دو روش نتیجه نهایی مشابهی تولید می کند.

صفات مشتق

صفاتی که مقادیرشان می تواند از دیگر صفات بدست آید صفات مشتق نامیده می شوند. اغلب، این صفتها در مدل داده منطقی نشان داده نمی شوند. با این وجود، بعضی اوقات مقدار صفت با صفتهایی که صفت مشتق بر مبنای آن است ممکن است حذف شده یا تغییر یابد. در این مورد، صفت مشتق شده، بایستی در مدل داده ای برای اجتناب از گم شدن اطلاعات نشان داده شوند. اگر صفت مشتق در مدل داده ای نشان داده شوند، شما باید صفت مشتق را با پسوند '/' نشان دهید.

ما نمایش صفات مشتق را بهنگام طراحی فیزیکی پایگاه داده در نظر می گیریم. بسته بر اینکه صفات چگونه استفاده می شوند، مقادیر جدید برای صفت مشتق ممکن است هر وقت که مورد دستیابی قرار گرفت یا وقتیکه مقدار آن از تغییر مشتق شد ممکن است محاسبه شود. با این وجود، این عمل ربطی به طراحی منطقی پایگاه داده ندارد، و اینکه چگونه به بهترین نحو صفات مشتق شده را نشان دهیم در مرحله ۱- ۶ فصل ۱۴ بحث خواهیم کرد.

مشکلات بالقوه

وقتی صفات را مشخص می کنید، گم کردن یک یا چند موجودیت از مدل اصلی امر غیرعادی نیست و هر زمانی می تواند اتفاق بیافتد. در این صورت، باید به مرحله قبلی برگشته، موجودیتهای جدیدی را مستند کنید و رابطه های مربوطه را دوباره بررسی کنید.

نکته بسیار مفید خواهد بود که لیستی از تمام صفاتی که در خصوصیات نیازمندیهای کاربر مشخص کرده اید را تولید کنید. و هر وقت که صفاتی را به موجودیت یا رابطه خاصی مربوط می کنید، می توانید آن عضو را از لیست حذف کنید. در این صورت، می توانید مطمئن شوید که صفت تنها به یک موجودیت یا رابطه مربوط شده است، و وقتی لیست خالی شد، بدین معنی است که تمام صفات به موجودیت یا رابطه ها مربوط شده اند.

شما همچنین باید از مواردی که در آن صفات با بیش از یک موجودیت ارتباط دارند آگاه باشید و می تواند بیانگر این باشد که:

(۱) چندین موجودیت را مشخص کرده اید که در واقع بیانگر یک موجودیت است. برای مثال، ممکن است موجودیتهای Manager وSupervisor را که هر دو دارای صفات staffNo ، Name، و salary باشند را مشخص کرده باشید که در واقع می توان به جای آن دو، یک موجودیت بنام Staff را با صفات staffNo , name , salary, و position در نظر گرفت.

در طرف دیگر، ممکن است که این موجودیتها چندین صفت مشترک داشته باشند، اما همچنین صفاتی وجود داشته باشد که برای هر موجودیت مجزا باشد. در فصل ۱۱، به بعضی از مفاهیم پیشرفته تر مدلسازی ER بنام خصوصی سازی و عمومی سازی خواهیم پرداخت، و راهبردی را برای استفاده از آنها ذکر می کنیم. این مفاهیم پیشرفته به شما اجازه می دهد تا این نوع از موقعیتها را دقیقتر نشان دهید. در اینجا این مفاهیم پیشرفته را به خاطر هر چه ساده تر بودن متدلوژی اصلی مان تا مرحله ۶-۱ ذکر نمی کنیم.

(۲) رابطه های بین موجودیتها را شناسایی کرده اید. در این مورد، شما باید صفت را تنها با یک موجودیت مربوط کنید، یعنی موجودیت والد، و اطمینان حاصل کنید که رابطه قبلا در مرحله ۲-۱ شناسایی شده باشد. و گرنه، مستندات بایستی با جزئیات رابطه های جدیدا شناسایی شده بروز شده باشند. برای مثال، ممکن است موجودیتهای Branch و Staff با این صفات را شناسایی کرده باشید:

Branch	branchNo, Street, city, state, zipCode, managerName
Staff	staffNo, name, position, salary

حضور صفت managerName در Branch بخاطر نمایش رابطه *Manages* Branch Staff است. در این مورد، بنابراین، صفت managerName بایستی از Branch حذف شود و رابطه Manages به مدل اضافه شود.

صفات *StayHome* برای موجودیتها

برای بررسی موردی *StayHome* ، باید صفات موجودیتها را به صورت زیر شناسایی و مربوط کنید:

Branch	branchNo, address (composite: street, city, state, zipcode), telNo (multi-valued)
Staff	staffNo, name, position, salary
Video	catalogNo, title, category, dailyRental, price
Director	directorName
Actor	actorName
Member	memberNo, name (composite: fName, lName), address
RentalAgreement	rentalNo, dateOut, dateReturn
VideoForRent	videoNo, available

توجه کنید که صفت **address** در **Branch** و صفت **name** در **Member** به صورت صفت ترکیبی مشخص شده اند، در حالیکه صفت **address** در **Member** و صفات **name** در **Staff**, **Director** و **Actor** بصورت ساده در نظر گرفته شده اند. که این نیازمندی های کاربر را برای این صفات منعکس می کند.

صفات *StayHome* برای رابطه ها

شما ممکن است در مرتبط ساختن صفتی که نشان دهنده تاریخی است که عضو در شعبه ثبت نام کرده ،dateJoined، با موجودیت معینی مشکل داشته باشید. از آنجائیکه عضو می تواند در بیش از یک شعبه ثبت نام کند، بنابراین نمی تواند با Member مربوط شود. در حقیقت، این صفت نباید با هر یک از موجودیتهای بالایی مربوط شود اما به جای آن می تواند با رابطه سه گانه بین Member, Branch, Staff مربوط شود. به طور مشابه، صفتی که نمایش دهنده نام شخصیت بازیگری که در فیلم بازی می کند، character ، باید با رابطه چند به چند ***PlaysIn*** بین Actor و Video مربوط شود.

صفات را مستند کنید

بعد از شناسایی صفات، به آنها نامی اختصاص دهید که به کاربر قابل فهم و واضح باشد. اطلاعات زیرین را برای هر صفت ثبت کنید:

- نام و توصیف صفت;
- طول و نوع داده;
- هرگونه نام مترادف یا مستعاری که صفت با آن شناخته می شود;
- اینکه آیا صفت همیشه باید مشخص شود (بعبارت دیگر، آیا صفت میتواند تهی باشد یا نه);
- آیا صفت چند مقداره است;
- آیا صفت ترکیبی است و اگر چنین است، صفات ساده تشکیل دهنده آن چیست;
- آیا صفت مشتق است و اگر چنین است، چگونه می تواند محاسبه شود;
- مقادیر پیش فرض برای صفات (اگر مشخص شده باشد).

شکل ۷-۸ خلاصه ای از دیکشنری داده را که صفات را برای *StayHome* مستند کرده را نشان می دهد.

موجودیت	صفات	توصیف	اندازه و نوع داده	تهی Nulls	چند مقداره	...
Branch	branchNo	Uniquely identifies a branch	4 fixed characters	No	No	
	address:					
	street	Street of branch address	30 variable characters	No	No	
	city	City of branch address	20 variable characters	No	No	
	state	State of branch address	2 fixed characters	No	No	
	zipCode	Zip code of branch address	5 variable characters	No	No	
	telNo	Telephone numbers of branch	10 variable characters	No	Yes	
Staff	staffNo	Uniquely identifies a member of staff	5 fixed characters	No	No	
	name	Name of staff member	30 variable characters	No	No	

شکل ۷-۸

خلاصه ای از دیکشنری داده *StayHome* که توصیف داده ها را نشان می دهد.

مرحله ۴-۱ دامنه های صفات را مشخص کنید

هدف

مشخص کردن دامنه های صفات در مدل داده منطقی محلی.

هدف این مرحله مشخص کردن دامنه های صفات در مدل داده منطقی محلی است. **دامنه**[1] مخزنی از مقادیری است که یک یا چند صفت مقدارشان را از آن می گیرند. نمونه هایی از دامنه های صفات برای *StayHome* شامل موارد زیر است:

- دامنه صفت شماره شعبه معتبر شامل رشته چهار کاراکتری با طول ثابت است، با کاراکتر اول بصورت یک حرف و سه کاراکتر بعدی رقم هایی از محدوده 999-000.
- دامنه صفت شماره تلفن معتبر یک رشته ۱۰ رقمی است.
- مقادیر ممکن برای صفت available موجودیت VideoForRent یا 'Y' است و یا 'N'. دامنه این صفت رشته تک کاراکتری شامل مقادیر 'Y' یا 'N' است.

مدل داده ای کاملا توسعه یافته دامنه های هر صفت مدل را مشخص میکند که شامل:

- مجموعه مقادیر مجاز برای صفت;
- اندازه و فرمت صفت.

[1] Domain

دامنه های صفت را مستند کنید

بعد از شناسایی دامنه های صفت، نامها و مشخصات آنها را در دیکشنری داده ثبت کنید. مدخل دیکشنری داده صفات را برای ثبت دامنه آنها در مکان نوع داده و اطلاعات طول بروز کنید.

مرحله ۵-۱ صفات کلید اصلی و کاندید را مشخص کنید

هدف

مشخص کردن کلید (های) کاندید هر موجودیت و، اگر بیش از یک کلید کاندید وجود داشته باشد، انتخاب یکی بعنوان کلید اصلی .

این مرحله مربوط به مشخص کردن کلید (های) کاندید موجودیت و سپس انتخاب یکی از آنها بعنوان کلید اصلی می باشد. مواظب باشید کلید کاندیدی را انتخاب کرده باشید که هرگز تهی نباشد (اگر کلید کاندید در برگیرنده بیش از یک صفت است، پس این به هر صفت اعمال می شود). اگر بیش از یک کلید کاندید مشخص کرده اید، باید یکی را بعنوان کلید اصلی انتخاب کنید; کلیدهای کاندید باقیمانده کلیدهای فرعی نامیده می شوند.

نکته همانطور که در بخش ۳-۲-۲ اشاره کردیم نام افراد عموما کلیدهای کاندید خوبی نیستند. برای مثال، ممکن است فکر کنید که کلید کاندید مناسب برای موجودیت Staff صفت name (نام پرسنل) باشد. با این وجود، دو نفر با نام مشابه ممکن است به *StayHome* ملحق شوند، که مشخصا انتخاب name بعنوان کلید کاندید نا معتبر خواهد بود. ما می توانیم آرگومان مشابهی را برای عضوهای *StayHome* بوجود آوریم. در چنین مواردی، بجای ترکیبی از صفات که یکتا باشند، بهتر است صفت جدیدی که همیشه یکتا باشد را تعریف کنیم، همچون صفت staffNo برای موجودیت Staff و صفت memberNo برای موجودیت Member .

زمانیکه کلید اصلی را از بین کلیدهای کاندید انتخاب می کنید، می توانید از راهنمای زیر کمک بگیرید تا انتخاب خود را انجام دهید :

- کلید کاندید با مجموعه حداقلی از صفات؛
- کلید کاندیدی که احتمالش کمتر است که تغییر کند؛
- کلید کاندیدی که به احتمال کمتر یکتایی خود را در آینده از دست بدهند؛
- کلید کاندیدی با کاراکترهای اندک (آنهایی که دارای صفات متنی هستند)؛
- کلید کاندید با کوچکترین مقدار بیشینه (برای صفات عددی)؛
- کلید کاندیدی که از لحاظ نقطه نظر کاربر برای استفاده آسان است؛

در مرحله شناسایی کلیدهای اصلی، توجه کنید آیا موجودیت ضعیف است یا قوی. اگر بتوانید کلید اصلی برای آن موجودیت اختصاص دهید، آن موجودیت **قوی** نامیده میشود. در سوی دیگر، اگر نتوانید کلید اصلی برای آن موجودیت مشخص کنید، موجودیت **ضعیف** نامیده می شود.

تنها زمانی می توانید کلید اصلی موجودیت ضعیف را مشخص کنید که ابتدا موجودیت ضعیف را به جدول نگاشت کنید، که در مرحله ۱-۲ فصل ۹ توضیح خواهیم داد.

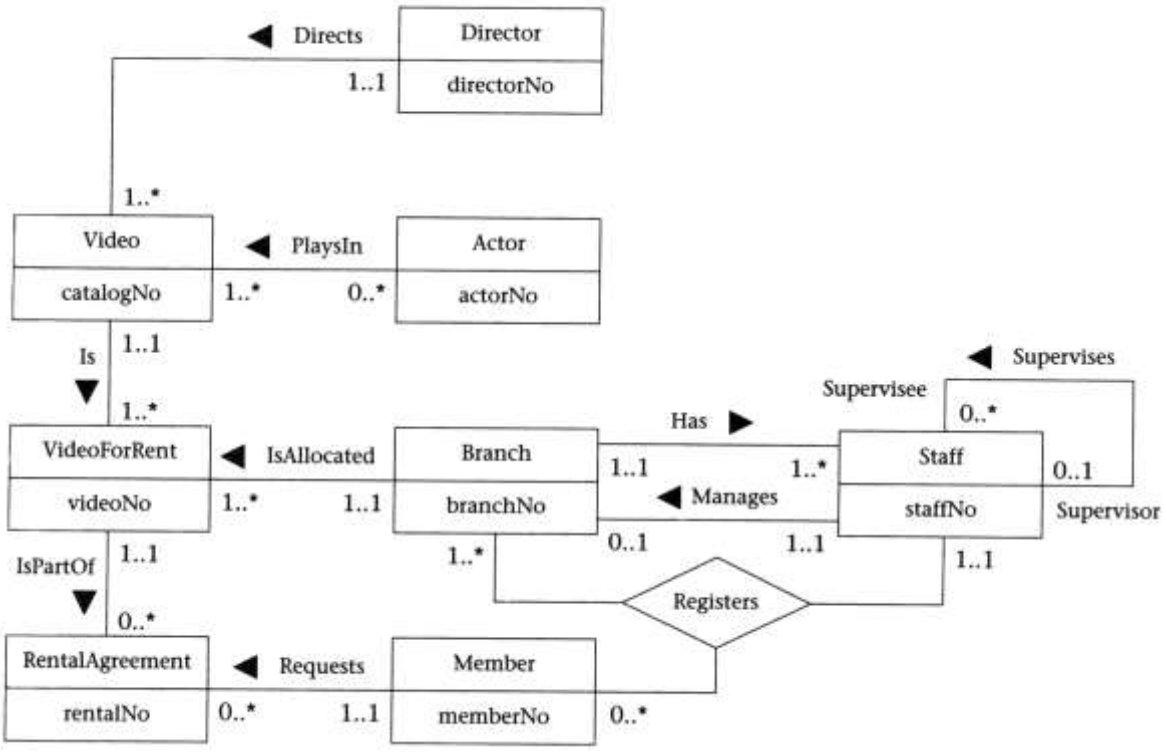
کلیدهای اصلی *StayHome*

برای بررسی موردی *StayHome* ، باید کلیدهای اصلی نشان داده شده در شکل ۸- ۸ را مشخص کنید.

از نیازمندی های کاربر *StayHome* مشاهده می شود که کلیدهای مشخصی برای موجودیتهای Director و Actor وجود ندارد. درحقیقت، تنها صفتی که برای موجودیتها مشخص شده نام کارگردان برای موجودیت Director و نام بازیگر برای موجودیت Actor هستند. و همانطور که گفتیم، اینها بعنوان کلید اصلی مناسب نیستند، بنابراین ما کلید اصلی برای هریک از این موجودیتها ساخته ایم که بترتیب directorNo و actorNo نام دارند.

شکل ۸-۸

مدل ER برای *StayHome* که کلیدهای اصلی را نشان می دهد.



کلیدهای کاندید، اصلی، و فرعی را مستند کنید.

مشخصات کلیدهای کاندید، اصلی، و فرعی (اگر موجود باشند) را در دیکشنری داده ثبت کنید. شکل ۹-۸ خلاصه ای از دیکشنری داده را که در آن صفات بهمراه کلیدها برای *StayHome* مستند شده است را نشان می دهد.

مرحله ۶-۱ موجودیتها را اختصاصی/ عمومی کنید (مرحله اختیاری)

هدف

مشخص کردن سوپرکلاس و زیرکلاس موجودیتها، جائیکه ممکن باشد.

در این مرحله، شما گزینه ای برای ادامه توسعه مدل داده منطقی دارید که آن مرحله استفاده از خصوصی سازی یا عمومی سازی است. مدلسازی ابرکلاس و زیر کلاس اطلاعات بیشتری را به مدل داده ای می افزاید، اما به همان میزان پیچیدگی را نیز اضافه می کند. در نتیجه، از آنجاییکه این یک مرحله اختیاری است، در اینجا به جزئیات وارد نمی شویم و به آنهایی که به این مبحث علاقه دارند در فصل ۱۱ این مبحث را بحث می کنیم.

موجودیت	صفات	توصیف	کلید	تهی	...
Branch	branchNo	Uniquely identifies a branch	Primary key	No	
	address:				
	street	Street of branch address		No	
	city	City of branch address		No	
	state	State of branch address		No	
	zipCode	Zip code of branch address	Alternate key	No	
	telNo	Telephone numbers of branch		No	
Staff	staffNo	Uniquely identifies a member of staff	Primary key	No	
	name	Name of staff member		No	

شکل ۹-۸

خلاصه ای از دیکشنری داده *StayHome* که صفات بهمراه کلید اصلی و فرعی را نشان می دهد.

مرحله ۷-۱ خصوصیاتی را که با مدل رابطه ای سازگار نیستند حذف کنید.

هدف

اصلاح مدل داده ای منطقی محلی با برداشتن خصیصه هایی که با مدل رابطه ای سازگار نیستند.

تا اینجا، شما دارای یک مدل داده محلی برای دید شعبه شرکت هستید. با این وجود، مدل داده ای ممکن است شامل بعضی ساختارهایی باشد که به آسانی توسط DBMS رابطه ای نتوان مدل کرد. در این مرحله، چنین ساختمان داده ای را به صورت فرم منتقل می کنید که توسط چنین سیستمهایی به آسانی بکار برده شوند. اهداف این مرحله چنین هستند:

(۱) حذف رابطه های باینری چند به چند (*:*).

(۲) حذف رابطه های بازگشتی چند به چند (*:*).

(۳) حذف رابطه های ترکیبی.

(۴) حذف صفات چند مقداره.

(۵) بررسی دوباره رابطه های یک به یک (۱:۱)

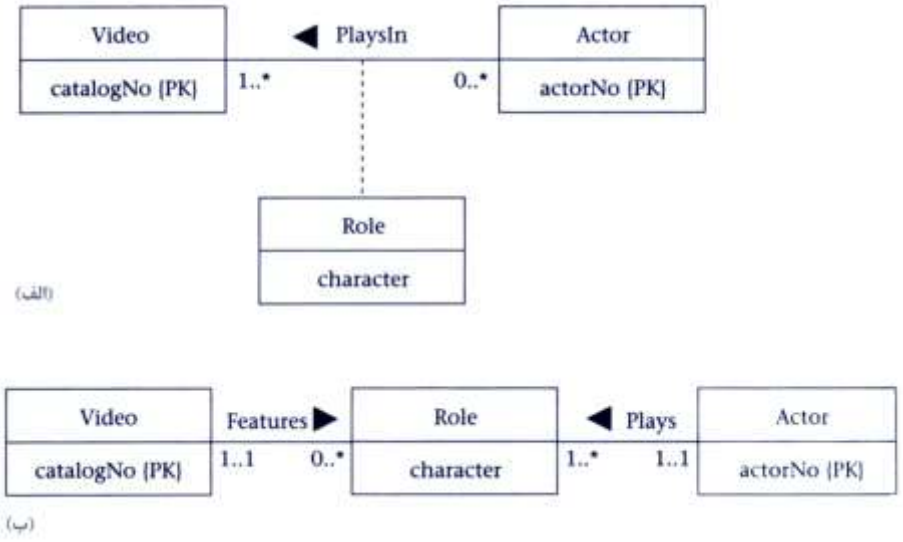
(۶) حذف رابطه های زاید.

حذف رابطه های باینری چند به چند (*: *)

اگر رابطه چند به چند در مدل داده ای منطقی دیده شود، باید این رابطه را تجزیه کنید تا موجودیت میانی مشخص شود. سپس باید رابطه *:* را برای موجودیتی که جدیدا شناسایی شده با دو رابطه یک به چند (*:۱) جایگزین کنید.

برای مثال، رابطه *:* Actor ***PlaysIn*** Video را در شکل ۱۰-۸ (الف) در نظر بگیرید. اگر ما رابطه ***PlaysIn*** را تجزیه کنیم، موجودیت Role و دو رابطه ۱:* جدید را شناسایی می کنیم (***Plays*** و ***Features***). رابطه *:* ***PlaysIn*** حالا به صورت Actor ***Plays*** Role و Video ***Features*** Role ، که در شکل ۱۰-۸ (ب) آمده نشان داده می شود.

این مثال همچنین موجودیت میانی جدیدا ایجاد شده را نشان می دهد. با این وجود، این تنها به صفات مرتبط با رابطه های *:* اعمال می شود; صفات مربوط شده با رابطه های ۱:۱ یا *:۱ می تواند بدون ایجاد موجودیت جدید بکار برده شود، همانطور که در مرحله ۱-۲ فصل بعدی خواهیم دید.



شکل ۱۰-۸

(الف) رابطه چند به چند Actor *PlaysIn* Video؛ (ب) تجزیه رابطه *:* به دو رابطه مجزای یک به چند (۱:*) (Plays و Features) با نشان دادن موجودیت ضعیف Role.

توجه کنید که موجودیت Role نشان داده شده در شکل موجودیت ضعیف است (کلید اصلی ندارد) که بعلت وابستگی وجودی به موجودیتهای مالک Actor و Video می باشد.

رابطه های بازگشتی چند به چند (*:*) را حذف کنید

رابطه بازگشتی به رابطه ای گفته می شود که در آن موجودیت با خودش رابطه دارد. سه نوع رابطه بازگشتی وجود دارد:

- بازگشتی یک به یک (۱:۱)
- بازگشتی یک به چند (۱:*)
- بازگشتی چند به چند (*:*)

دو رابطه اولی می توانند در مدل رابطه ای بصورت یک جدول بدون نیاز به ساختن دوباره رابطه نشان داده شوند، با این وجود اگر رابطه بازگشتی *:۱ اشتراک اختیاری در طرف * رابطه داشته باشد، بهتر است جدول دومی ایجاد کنیم تا تعداد تهی هایی که احتمالا ذخیره می شود را کاهش دهد. ما در مرحله ۱-۲ فصل بعد اینکه چگونه رابطه های بازگشتی ۱:۱ و *:۱ را نشان دهیم را بررسی خواهیم کرد. با این وجود، اگر شما رابطه بازگشتی *:* را شناسایی کردید، باید این رابطه را برای مشخص کردن موجودیت میانی تجزیه کنید. ما در بررسی موردی *StayHome* هیچ رابطه بازگشتی *:* نداریم، اما رابطه بازگشتی *:۱ Staff *Supervises* Staff را داریم. به هرحال، برای نشان دادن اینکه چگونه می توانیم چنین رابطه هایی را دوباره بازسازی کنیم تا با مدل رابطه ای سازگار باشند، اجازه دهید اینطور فرض کنیم که این رابطه بازگشتی *:* نشان دهنده موقعیتی است که پرسنل می تواند توسط چند ناظر نظارت شود، همانطور که در شکل ۱۱-۸(الف) نشان داده شده است.

شما رابطه بازگشتی *:* را به همان صورت که با رابطه *:* بین موجودیتهای مختلف استفاده می کنید نیز ساده کنید. راه مستقیم حل این مساله اول نمایش Staff بصورت دو موجودیت است، که به ما رابطه باینری آشنای *:* را می دهد، که در شکل ۱۱-۸(ب) نشان داده شده است. همانند قبل، این رابطه *:* را با در نظر گرفتن موجودیت میانی ضعیف (SuperviseStaff) نیز می توانید حل کنید، که در شکل ۱۱-۸ (ج) نشان داده شده است. سپس می توانید دو قسمت موجودیت جدا از هم Staff را با هم ترکیب کنید، فقط مواظب باشید که رابطه ای را گم نکنید، که مدل نهایی بصورت شکل ۱۱-۸(د) خواهد شد.

رابطه های مرکب را حذف کنید

رابطه مرکب رابطه ای بین سه یا چند موجودیت می باشد. اگر رابطه مرکب در مدل داده منطقی دیده شد، باید این رابطه را جهت مشخص کردن موجودیت میانی تجزیه کنید. رابطه مرکب با تعدادی رابطه باینری *:۱ موجودیت جدیدا مشخص شده جایگزین می شود.

برای مثال، رابطه سه گانه ***Registers*** وابستگی عضو پرسنلی که عضو جدید برای شرکت را ثبت نام می کند (عضو می تواند در شعب مختلفی ثبت نام کند) را نشان می دهد، که در شکل ۱۲-۸ (الف) آمده است. حال می توانید این رابطه را با در نظر گرفتن موجودیت جدید و تعریف رابطه های (باینری) بین موجودیتهای اصلی و موجودیت جدید ساده کنید. در این مثال، رابطه ***Registers*** را جهت شناسایی موجودیت ضعیفی با نام Registiration تجزیه کنید. موجودیت جدید با موجودیتهای اصلی از طریق سه رابطه باینری جدید:

Member ***Agrees*** Registration , Branch ***Registers*** Registration, Staff ***Processes*** Registration, مربوط شده اند، که در شکل ۱۲-۸ (ب) آمده است.

صفتهای چند مقداره را حذف کنید

صفات چند مقداره چندین مقدار را برای موجودیت واحد نگه می دارند. اگر صفت چند مقداره در مدل داده منطقی وجود داشته باشد، باید این صفت را تجزیه کنید تا موجودیت شناسایی شود.

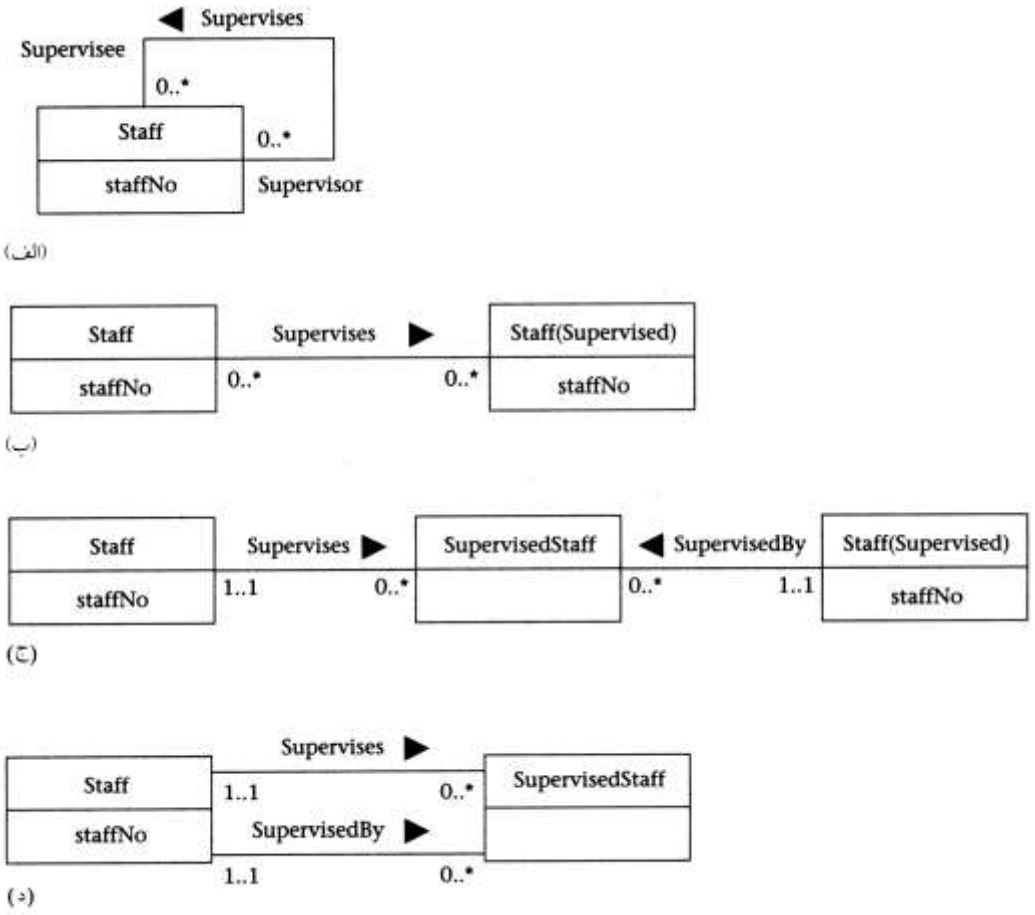
برای مثال، جهت نشان دادن موقعیتی که یک شعبه سه شماره تلفن دارد، صفت telNo موجودیت Branch را بعنوان صفت چند مقداره مشخص می کنیم، که در شکل ۱۳-۸ (الف) آمده است. شما باید این صفت چند مقداره را برداشته و موجودیت جدیدی بنام Telephone ایجاد کنید، که با صفت چند مقداره telNo که هم اکنون بعنوان صفت ساده (کلید اصلی) نشان داده می شود، و رابطه جدید ***Provides*** نامیده می شود، که در شکل ۱۳-۸(ب) آمده است.

رابطه های یک به یک (۱:۱) را دوباره بررسی کنید

در شناسایی موجودیتها، ممکن است دو موجودیت را که نشان دهنده شی مشابهی در شرکت هستند را مشخص کرده باشید. برای مثال، ممکن است دو موجودیت با نامهای Branch و Outlet که دقیقا یکسان هستند را مشخص کنید; بعبارت دیگر، Branch مترادف Outlet است. در این مورد، دو موجودیت بایستی با هم ادغام شوند. اگر کلیدهای اصلی شان متفاوت است، یکی از آنها را بعنوان کلید اصلی و دیگری را بعنوان کلید فرعی مشخص کنید.

شکل ۱۱-۸

(الف) رابطه بازگشتی چند به چند (*:*) *Supervises* .(ب) نمایش رابطه بازگشتی *:* *Supervises* بعنوان رابطه آشنای باینری. (ج) برطرف کردن رابطه*:* *Supervises* جهت ایجاد موجودیت (ضعیف) جدید بنام *SupervisedStaff* و رابطه اضافی *SupervisedBy* . (د) ترکیب دوباره دو موجودیت *Staff* جهت رفع رابطه بازگشتی *:*.



رابطه های زاید را حذف کنید

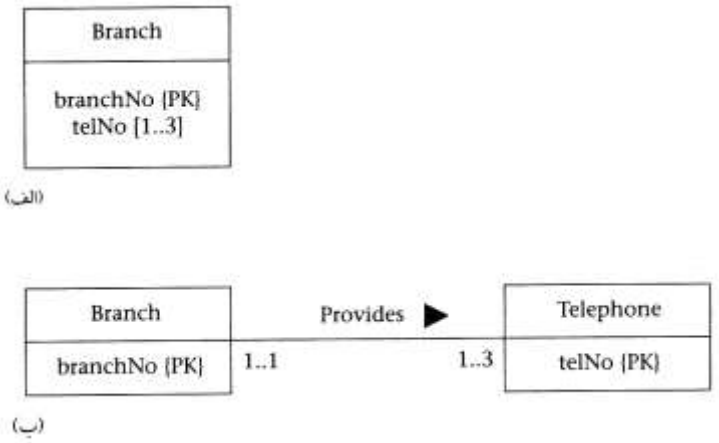
رابطه ای زاید است که اطلاعات مشابهی بتواند از رابطه ای دیگر بدست آید. شما می خواهید مدل داده ای را بصورت کمینه توسعه دهید، و از آنجائیکه رابطه های زاید غیر لازم هستند، بنابراین بایستی حذف شوند. شناسایی وجود بیش از یک مسیر بین دو موجودیت نسبتا ساده است. با این وجود، لزوما اشاره بر زاید بودن رابطه ها ندارد، زیرا ممکن است نشان دهنده رابطه های دیگر شرکت باشد.

برای مثال، بعد از تجزیه رابطه سه گانه *Registers* به سه رابطه باینری جدا از هم، اجازه بدهید رابطه های بین Branch ، Staff ، و Registration نشان داده شده در شکل ۱۴-۸ را بررسی کنیم. در این مدل، دو مسیر برای دستیابی به Registration از Branch وجود دارد. این نوع ساختار بازگشتی را باید شما همیشه در نظر بگیرید که آیا افزونگی وجود دارد. اجازه دهید رابطه های *Has* و *Registers* را از نظر وجود افزونگی بررسی کنیم.

(الف) *Has* آیا می توانیم تنها با رابطه های *Registers* و *Processes* محل کار پرسنل را مشخص کنیم؟ برای هر ثبت نام، می توانیم ارتباط شعبه با ثبت نام را با استفاده از رابطه ۱:* *Registers* مشخص کنیم. بعلاوه، برای هر ثبت نام، می توانیم با رابطه Processes عضوی از پرسنل را که ثبت نام را انجام داده شناسایی کنیم، و از آن محل شعبه ای را که عضوی از پرسنل کار می کند را نتیجه بگیریم. با این وجود، شرکت Staff در رابطه *Processes* اختیاری است.
بعبارت دیگر، هر عضو پرسنل ثبت نام را انجام نمی دهند، و ما نمی توانیم شعبه را برای تمام اعضا پرسنل نتیجه گیری کنیم.
بنابراین، می توانید نتیجه گیری کنید که رابطه *Has* زاید نیست.

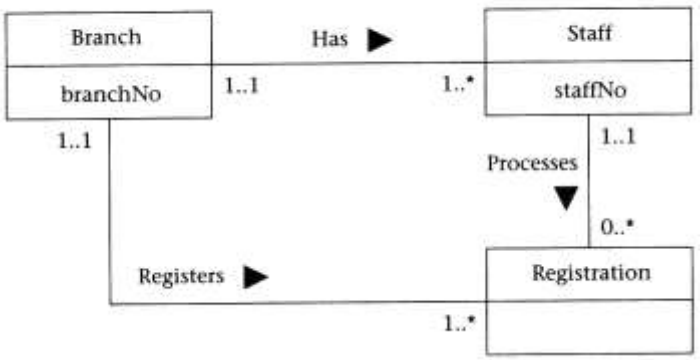
شکل ۱۳-۸

(الف) موجودیت Branch با صفت چند مقداره بنام telNo. (ب) تجزیه telNo به رابطه ۱:۳ بنام *Provides* بین Branch و موجودیت جدید Telephone با صفت ساده (کلید اصلی) telNo.



شکل ۱۴-۸

موجودیتهای Branch، Staff ،و Registration با سه رابطه .



(ب) *Registers* آیا می توانیم مشخص کنیم که کدام شعبه ثبت نام فقط برای استفاده رابطه های *Has* و*Processes* است؟ برای هر عضو پرسنل، می توانیم شعبه عضو را با استفاده از رابطه ۱:* *Has* مشخص کنیم. بعلاوه، برای هر عضو پرسنل، می توانیم ثبت نام هایی را مشخص کنیم که این پرسنل با استفاده از رابطه ۱:* *Processes* انجام داده است، و بنابراین شعبه ای را نتیجه بگیریم که ثبت نام از طریق شعبه پرسنل صورت گرفته است. بنابراین، شرکت موجودیت Registration در رابطه Processes الزامی بوده، و باید عضوی از پرسنل برای هر ثبت نام وجود داشته باشد. بر اساس این آزمون، شما باید نتیجه گرفته باشید که رابطه *Registers* زاید است. با این وجود، وقتی چگونگی عملکرد شرکت را در نظر می گیرید، امکان اینکه پرسنل در زمان وجود خود در *StayHome* در بیش از یک شعبه فعالیت خواهد کرد. وقتی این را مورد توجه قرار می دهید، می توانید نتیجه بگیرید که تمام رابطه های مورد نیاز را به طور صحیح برای مدل این موقعیت در نظر گرفته اید.

بعد از تشخیص افزونگی بُعد زمان را بررسی کنید

بُعد زمان رابطه ها هنگام تشخیص افزونگی همچنین دارای اهمیت است. برای مثال، موقعیتی را در نظر بگیرید که می خواهید رابطه های بین موجودیتهای Man، Woman، و Child را که در شکل ۱۵- ۸ نشان داده شده است را مدل کنید. آشکار است که، دو مسیر بین Man و Child وجود دارد: یکی از طریق رابطه مستقیم FatherOf و دیگری از طریق رابطه های MarriedTo و MotherOf است. بنابراین، ممکن است چنین فکر کنید که رابطه FatherOf غیر لازم است. اما، این طرز فکر به دو دلیل صحیح نمی باشد:

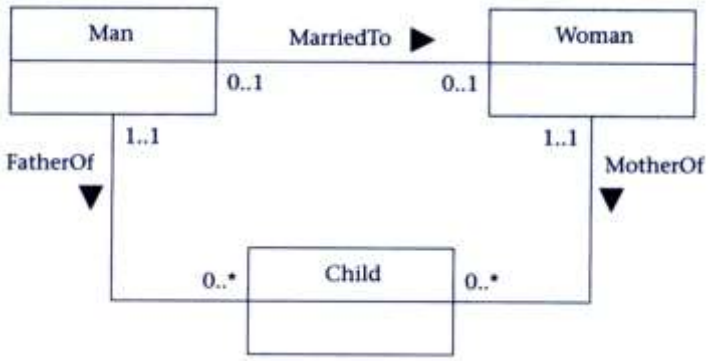
(۱) پدر ممکن است از ازدواج قبلی فرزندی داشته باشد، و شما تنها ازدواج کنونی پدر را از طریق رابطه ۱:۱ مدل کرده باشید.

(۲) پدر و مادر ممکن است ازدواج نکرده باشند، یا پدر ممکن است با کس دیگری به غیر از مادر ازدواج کرده باشد (یا مادر ممکن است با شخص دیگری به غیر پدر ازدواج کرده باشد).

در هر دو مورد، رابطه مورد نیاز نمی تواند بدون رابطه FatherOf مدل شود.

شکل ۱۵-۸

رابطه های غیر زاید.



نکته پیامی که اینجا وجود دارد این است که، پس از تشخیص افزونگی، بررسی معانی روابط بین موجودیتها از اهمیت فراوانی برخوردار است.

مدل داده منطقی محلی اصلاح شده را رسم کنید

شما در پایان این مرحله، مدل داده منطقی محلی را با حذف ساختمان داده هایی که پیاده سازی آن در پایگاه داده رابطه ای مشکل است ساده خواهید کرد. شما همچنین باید مدل داده منطقی محلی و دیکشنری داده را بهنگام سازید تا هرگونه تغییر انجام گرفته در این مرحله را نیز منعکس کند. با به بهم پیوستن تغییرات مشخص شده فوق، مدل داده منطقی محلی اصلاح شده ای را که در شکل ۱۶-۸ نشان داده شده است را بدست خواهید آورد.

مرحله ۸-۱ مدل را جهت پشتیبانی از تراکنش ها بررسی کنید

هدف

اطمینان از اینکه مدل داده منطقی محلی تراکنش های مورد نیاز کاربر را پشتیبانی می کند.

شما هم اکنون مدل داده محلی دارید که دید مشخصی از شرکت را ارائه می کند. هدف این مرحله بررسی مدل داده منطقی محلی جهت اطمینان از این است که مدل از تراکنشهای موردنیاز این دید پشتیبانی می کند. در مورد ما، نیازمندی های تراکنش برای دید شعبه *StayHome* در بخش ۴-۴-۴ لیست شده است.

با استفاده از مدل داده منطقی و دیکشنری داده، سعی کنید که عملیات را بصورت دستی انجام دهید. اگر بتوانید تمام تراکنشها را بدین طریق حل کنید، نشان داده اید که مدل داده منطقی شما تراکنشهای مورد نیاز را پشتیبانی می کند. بنابراین، اگر قادر نبودید تراکنشها را به طور دستی بررسی کنید، بایستی مشکلی در مدل داده ای وجود داشته باشد، که باید برطرف شود. مشکل ممکن است این باشد که شما، موجودیت، رابطه و یا صفتی را در مدل داده ای از قلم انداخته باشید.

ما در اینجا دو روش ممکن جهت اطمینان از اینکه مدل منطقی محلی تراکنشهای موردنیاز را پشتیبانی می کند را بررسی میکنیم.

شرح تراکنش

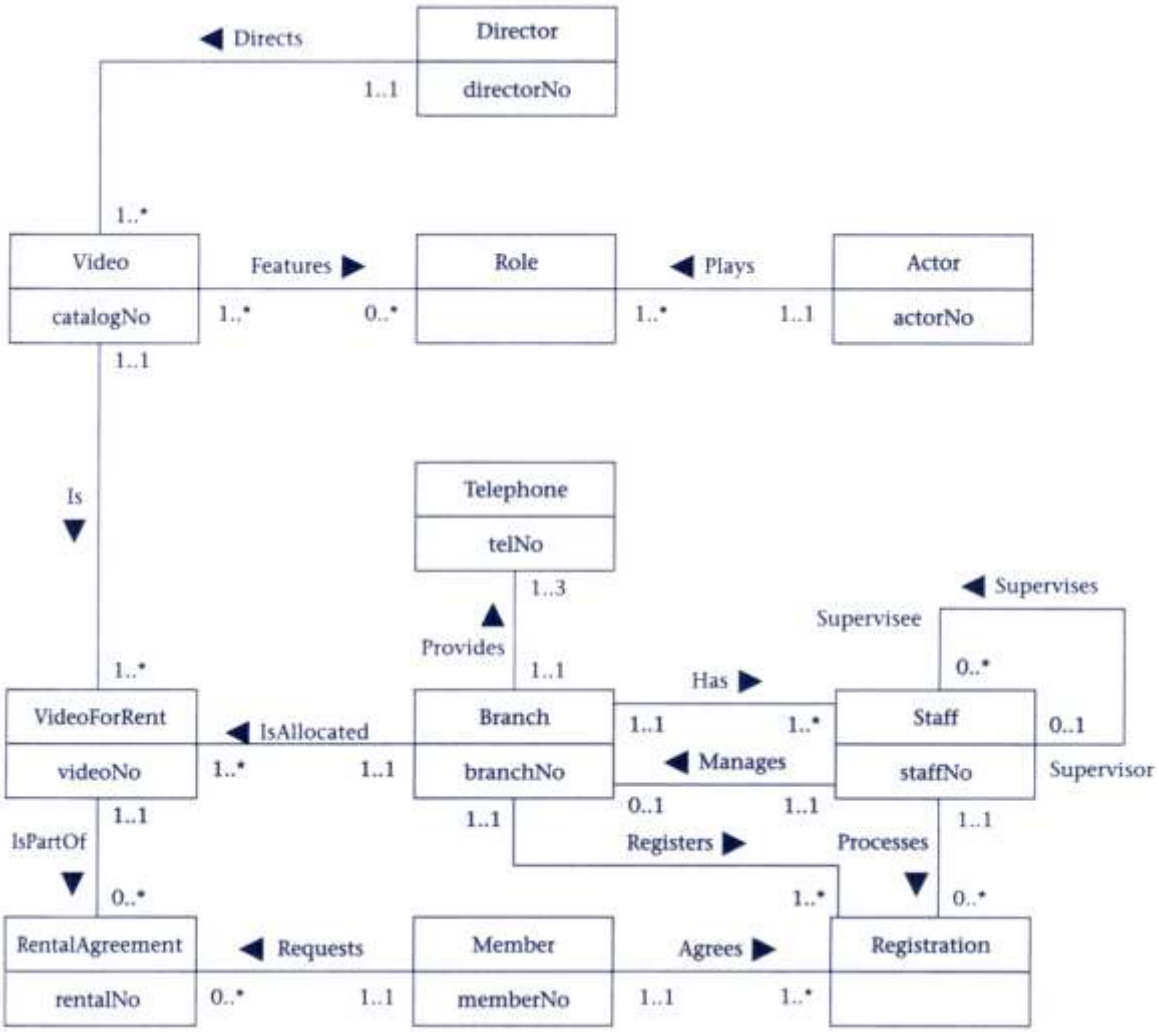
با استفاده از روش اول، بوسیله مستند سازی شرح هر نیازمندی تراکنش، تمام اطلاعات مورد نیاز توسط هر تراکنش را که توسط مدل فراهم گشته است را بررسی کنید. اجازه دهید چند نیازمندی تراکنش نمونه *StayHome* را بررسی کنیم:

تراکنش (o) نام هر مدیر موجود در هر شعبه را لیست می کند، که با شماره شعبه مرتب گشته اند

نام هر مدیر در موجودیت Staff و جزئیات شعبه در موجودیت شعبه نگه داشته می شوند. در این مورد، شما می توانید از رابطه Staff *Manages* Branch برای پیداکردن نام هر مدیر هر شعبه استفاده کنید.

شکل ۱۶-۸

مدل داده منطقی محلی اصلاح شده دید شعبه *StayHome* .



استفاده از خط سیر تراکنش

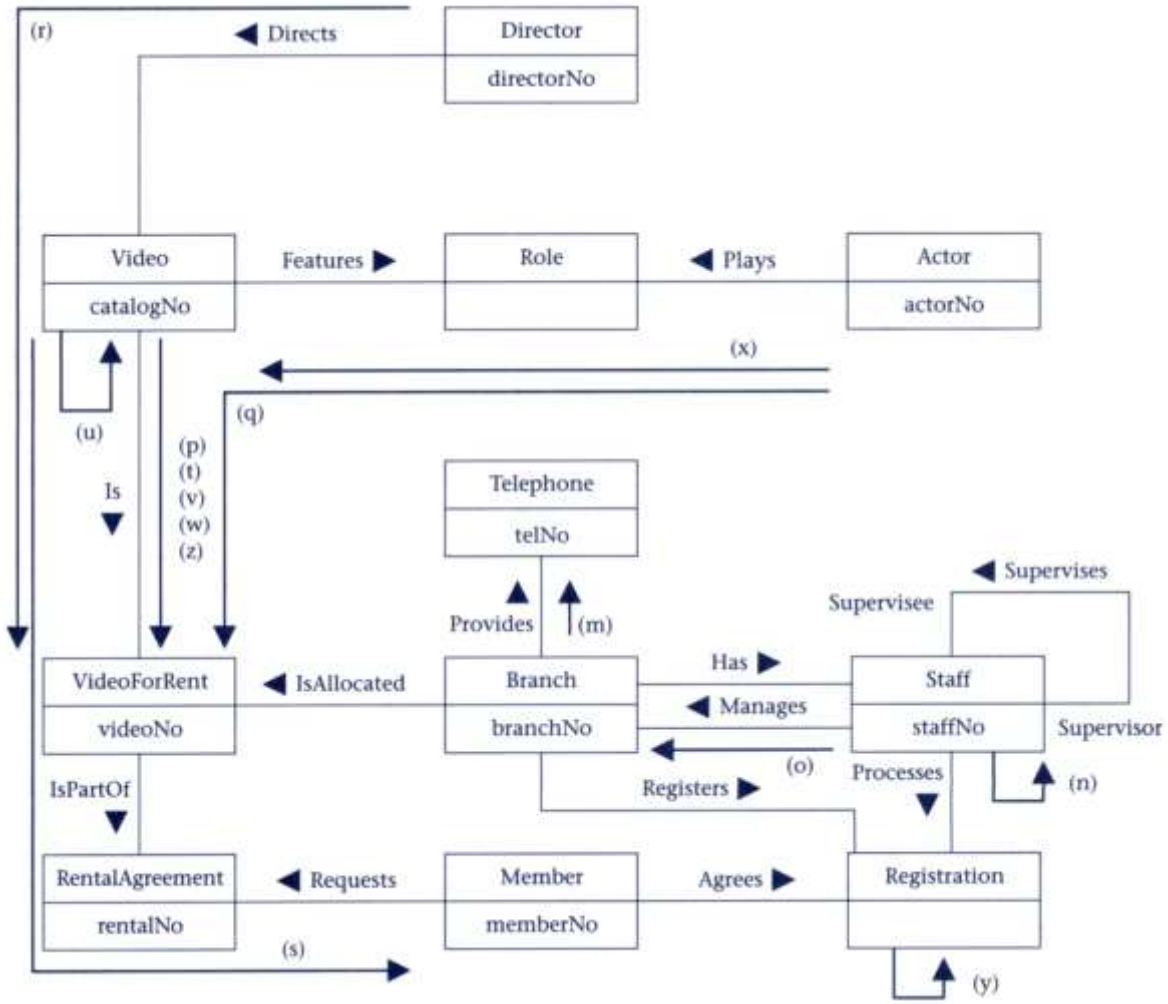
روش دوم برای معتبرسازی مدل داده در برابر تراکنشهای مورد نیاز شامل نمایش نموداری خط سیر تراکنش بطور مستقیم روی مدل ER است. مثالی از این روش استفاده از پرس و جوهای داده لیست شده در بخش ۴-۴-۴ است که در شکل ۱۷-۸ نشان داده شده است. واضح است که، تراکنش های بیشتری وجود دارد، نمودار پیچیده ای از این بوجود خواهد آمد، بنابراین برای خوانایی ممکن است به چندین نمودار از این نوع جهت پوشش تمام تراکنش ها نیاز داشته باشید.

نکته این ممکن است کمی کار مشکلی به نظر برسد و مطمئنا نیز چنین خواهد بود. در نتیجه، ممکن است بخواهید این مرحله را موقتا از قلم بیاندازید. با این وجود، مهم است که این بررسی را حتما بجای آینده هم اکنون انجام دهید زیرا ممکن است در آینده مدل از این هم پیچیده تر شده و برطرف کردن هر اشتباه رخ داده ای زمانبر و هزینه بر خواهد بود.

در پایان این مرحله، شما مدل داده منطقی محلی را بررسی خواهید کرد تا از پشتیبانی مدل از تراکنشهای مورد نیاز آگاه شوید. حال آماده هستید که به مرحله اصلی بعدی بپردازید، که مدل داده ای را به صورت مجموعه ای از جدولها نگاشت کرده و مدل را معتبر خواهید کرد (طراحی فیزیکی).

شکل ۱۷-۸

استفاده از خط سیر جهت بررسی اینکه مدل داده منطقی محلی تراکنشهای کاربر را پشتیبانی می کند.



خلاصه فصل

- ✓ هدف اصلی مرحله ۱ متدلوژی ساختن مدل داده ای منطقی محلی هر دید شرکت است.
- ✓ هر مدل داده ای منطقی محلی متشکل از: موجودیتها، رابطه ها، صفات، دامنه های صفت، کلیدهای کاندید، کلیدهای اصلی، و محدودیتهای جامعیت است.
- ✓ هر مدل داده ای منطقی محلی توسط مستندات پشتیبانی می شوند، همچون دیکشنری داده، که از طریق توسعه مدل تولید می شوند.
- ✓ هر مدل منطقی بایستی بدین صورت تصحیح شود: حذف رابطه های باینری *:*، حذف رابطه های بازگشتی *:*، حذف رابطه های ترکیبی، حذف صفات چند مقداره، بررسی دوباره رابطه های ۱:۱ ، و حذف رابطه های زاید.
- ✓ هر مدل منطقی بایستی از لحاظ پشتیبانی از تراکنشهای موردنیاز توسط کاربران بررسی شوند.

فصل ۹

طراحی منطقی پایگاه داده - مرحله ۲

آنچه در این فصل خواهید آموخت:

- چگونه جدولهای مدل داده منطقی را ایجاد کنید.
- چگونه با استفاده از نرمالسازی ساخت یافته بودن جداول را بررسی کنید.
- چگونه بررسی کنید که جداول تراکنشهای کاربر را پشتیبانی می کنند.
- چگونه محدودیتهای جامعیت را تعریف و مستند کنید.

این فصل دومین مرحله از متدلوژی طراحی منطقی پایگاه داده ما را می پوشاند. در این مرحله، شما مجموعه ای از جدولها را برای هر مدل داده منطقی محلی ایجاد می کنید. سپس با استفاده از نرمالسازی بررسی می کنید تا ببینید که جدولها دارای ساختار مناسبی هستند، و از تراکنشهای کاربر پشتیبانی می کنند. بالاخره، محدودیتهای جامعیت را برای مدل داده منطقی محلی تعریف و مستند می کنید.

مرحله ۲ جدولها را برای هر مدل داده منطقی محلی ایجاد و بررسی کنید

هدف

برای هر مدل داده منطقی محلی جدولها را ایجاد کرده و ساختار جدولها را بررسی کنید.

هدف عمده این مرحله تولید توصیف جدولها برای هر مدل داده منطقی محلی ایجاد شده در مرحله ۱ متدلوژی است. مجموعه جداول تولید شده باید نشان دهنده موجودیتها، رابطه ها، صفات، و محدودیتهای موجود در مدل داده منطقی باشند. سپس ساختار هر جدول بررسی می شود تا از عدم وجود اشتباه به هنگام ایجاد جدولها اطمینان حاصل شود. اگر اشتباهاتی در جداول موجود باشد احتمالا نشان دهنده این باشد که این اشتباهات هنگام فرایند ایجاد جدول تولید شده باشد یا نشانگر این باشد که در مدل ER هنوز اشتباهاتی است که شناسایی نشده اند. مرحله ۲ شامل کارهای زیر است:

- مرحله ۲-۱ جداولی را برای مدل داده منطقی محلی ایجاد کنید
- مرحله ۲-۲ ساختارهای جدول را با استفاده از نرمال سازی بررسی کنید
- مرحله ۲-۳ جدولها را از نظر پشتیبانی از تراکنش های کاربر بررسی کنید
- مرحله ۲-۴ محدودیتهای جامعیت را تعریف کنید
- مرحله ۲-۵ مدل داده منطقی محلی را با کاربران بازبینی کنید

ما مرحله ۲ متدلوژی را با استفاده از مدل داده منطقی محلی ایجاد شده در مرحله ۱ فصل قبل برای دید شعبه کاربرد پایگاه داده *StayHome* نشان می دهیم. این دید نشان دهنده دیدهای کاربر Manager، Supervisor، و Assistant می باشد.

مرحله ۱-۲ جداول را برای مدل داده منطقی محلی ایجاد کنید.

هدف

ایجاد جداول برای مدل داده منطقی محلی.

در این مرحله، جهت نشان دادن موجودیتها، رابطه ها، صفات، و محدودیتهای توصیف شده در این دید از شرکت جداولی را برای مدل داده منطقی محلی ایجاد می کنید. ساختار جدولها از اطلاعاتی مشتق شده است که در مدل داده منطقی توصیف شده اند، که این اطلاعات شامل مدل ER، دیکشنری داده، و هر گونه مستند پشتیبانی کننده دیگر می باشد. برای شرح ترکیب هر جدول شما برای پایگاه داده رابطه ای از زبان تعریف پایگاه داده (DBDL) استفاده می کنید. با استفاده از DBDL، ابتدا نام جدول را مشخص می کنید، و بدنبال آن لیستی از نامهای صفات ساده جدول را داخل کروشه ذکر می کنید. سپس کلیدهای اصلی و فرعی/ یا خارجی جدول را مشخص می کنید. برای هر کلید خارجی، جدول شامل کلید اصلی ارجاع شده نیز نشان داده می شود.

ما این مرحله را با استفاده از مدل ER برای دید شعبه کاربرد پایگاه داده *StayHome* نشان می دهیم که در شکل ۱۶-۸ نشان داده شده است. با این حال، در بعضی موارد لازم است تا نمونه هایی را که نقاط خاصی را نشان می دهد و در این مدل نشان داده نشده است را نیز اضافه کنیم.

چگونه موجودیتها را نشان بدهیم

برای هر موجودیت موجود در مدل ER، جدولی ایجاد کنید که شامل تمام صفات ساده موجودیت است. برای صفات ترکیبی، فقط صفات ساده تشکیل دهنده صفات ترکیبی را ذکر کنید. برای مثال، برای صفت ترکیبی address، باید صفات ساده شهر، خیابان، منطقه، و کدپستی را ذکر کنید. هر جا امکان دارد، ستونهایی را که کلید اصلی هر جدول را تشکیل داده مشخص کنید. برای موجودیتهای نشان داده شده در شکل ۱۶-۸ باید ساختار ابتدایی جدول نشان داده در شکل ۱-۹ را ایجاد کنید.

شکل ۱-۹

ساختارهای جدول اولیه نشان دهنده موجودیتهای نشان داده شده در شکل ۱۶-۸.

Actor (actorNo, actorName) **Primary Key** actorNo	**Branch** (branchNo, street, city, state, zipCode) **Primary Key** branchNo **Alternate Key** zipCode
Director (directorNo, directorName) **Primary Key** directorNo	**Member** (memberNo, fName, lName, address) **Primary Key** memberNo
Registration (dateJoined) **Primary Key** unknown	**RentalAgreement** (rentalNo, dateOut, dateReturn) **Primary Key** rentalNo
Role (character) **Primary Key** unknown	**Staff** (staffNo, name, position, salary) **Primary Key** staffNo
Telephone (telNo) **Primary Key** telNo	**Video** (catalogNo, title, category, dailyRental, price) **Primary Key** catalogNo
VideoForRent (videoNo, available) **Primary Key** videoNo	

در بعضی موارد، مجموعه تمام ستونهای تشکیل دهنده جدول را شناسایی نمی کنید. آن هم به این دلیل است که شما هنوز دارید رابطه های بین موجودیتها را نشان می دهید. خصوصا، این بدین معنی است که شما نمی توانید ستونهای تشکیل دهنده کلید اصلی موجودیتهای ضعیفی همچون Role و Registration را تا وقتیکه رابطه های آن را در مدل ER نشان نداده اید تعیین کنید. شناسایی ستونهای کلید اصلی برای موجودیتهای ضعیف را در پایان این مرحله بحث می کنیم.

چگونه رابطه ها را نشان دهیم

رابطه ای که یک موجودیت با موجودیت دیگر دارد توسط مکانیزم کلید اصلی/ خارجی نشان داده می شود. گرفتن تصمیم درباره اینکه کجا صفات کلید خارجی را *بفرستیم* (یا قرار دهیم)، شما ابتدا باید موجودیتهای 'والد' و ' فرزند' در برگرفته شده در رابطه را مشخص کنید. موجودیت والد به موجودیتی اشاره دارد که کپی کلید اصلی خود را به جدولی که بعنوان موجودیت فرزند است می فرستد، تا بعنوان کلید خارجی عمل کند.

شناسایی موجودیتهای والد/ فرزند را برای انواع رابطه های زیر در نظر می گیریم:

(الف) رابطه های باینری یک به چند (*:۱)
(ب) رابطه های بازگشتی یک به چند (*:۱)
(ج) رابطه های باینری یک به یک (۱:۱)
(د) رابطه های بازگشتی یک به یک (۱:۱)

توجه کنید که شما رابطه های ترکیبی و چند به چند (*:*) را در مرحله ۷-۱ متدلوژی فصل قبلی حذف کرده اید.

رابطه های باینری یک به چند (*:۱)

برای هر رابطه باینری *:۱، موجودیت در 'طرف یک' از رابطه بعنوان موجودیت والد و موجودیت در 'طرف چند' رابطه بعنوان موجودیت فرزند معین شده است. برای نمایش این رابطه، کپی کلید اصلی موجودیت والد در جدولی که نشان دهنده موجودیت فرزند است قرار می گیرد، تا بعنوان کلید خارجی عمل کند.

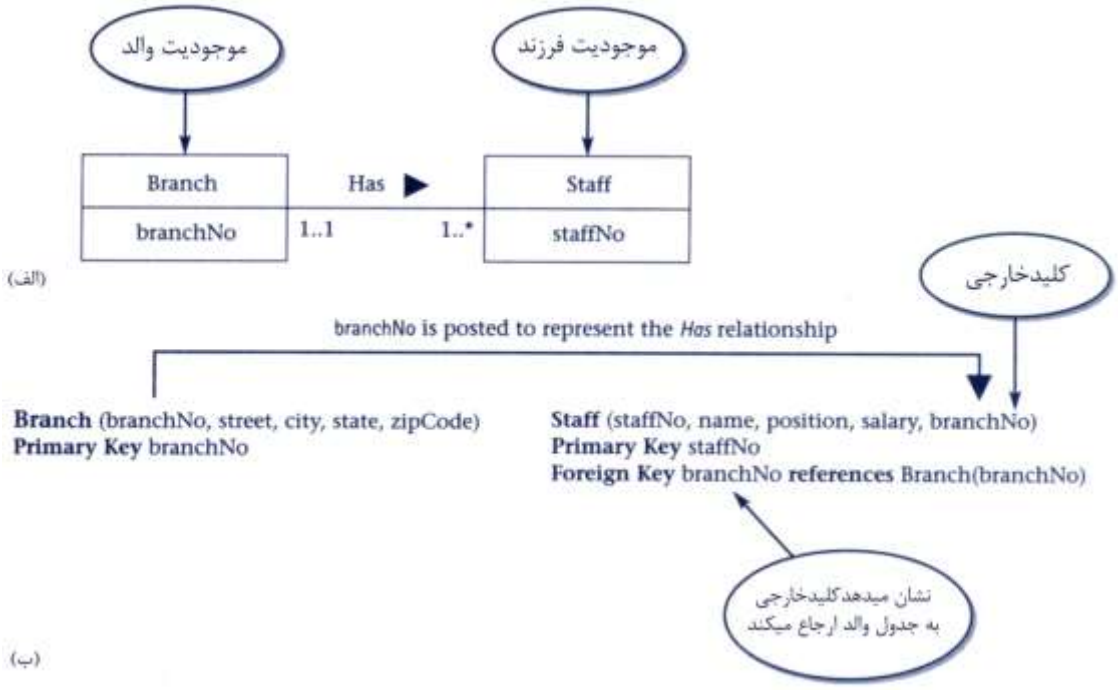
اجازه بدهید رابطه Branch *Has* Staff نشان داده شده در شکل ۱۶-۸ را برای نشان دادن اینکه چگونه رابطه *:۱ را به صورت جدول نشان دهیم در نظر بگیریم. در این مثال، Branch در 'طرف یک' قرار دارد و موجودیت والد را نشان می دهد، و Staff در 'طرف چند' قرار دارد و موجودیت فرزند را نشان می دهد. رابطه بین این موجودیتها با قرار دادن کپی کلید اصلی موجودیت (والد) Branch ، بنام branchNo ، در داخل جدول (فرزند) Staff برقرار می شود. شکل ۲-۹ (الف) مدل ER، Branch *Has* Staff و شکل ۲-۹ (ب) جدولهای متناظر را نشان می دهند.

چندین مثال دیگر از رابطه های *:۱ همچون Director *Directs* Video و Member *Requests* RentalAgreement در شکل ۱۶-۸ وجود دارند. شما باید قواعد معین شده در بالا را برای هر رابطه *:۱ مدل داده منطقی محلی تکرار کنید.

در مواردی که رابطه *:۱ یک یا چند صفت دارد، این صفتها باید از روش فرستادن کلید اصلی به جدول فرزند پیروی کنند. برای مثال، اگر رابطه Branch *Has* Staff صفتی بنام dateStart داشت که بیانگر موقعیکه عضوی از پرسنل در شعبه شروع به کار می کند می باشد، این صفت همچنین بایستی به جدول Staff همراه با کپی کلید اصلی جدول Branch، بنام branchNo فرستاده شود.

شکل ۲-۹

رابطه ۱:* Branch *Has* Staff (الف) مدل ER ; (ب) نمایش به صورت جدول با استفاده از DBDL.



رابطه های بازگشتی یک به چند (۱:*)

نمایش رابطه بازگشتی ۱:* مشابه آنچه که در بالا شرح داده شد می باشد. در شکل ۱۶-۸، رابطه بازگشتی ۱:* ای که وجود دارد رابطه Staff *Supervises* Staff می باشد. در این مورد، موجودیت Staff هم والد و هم فرزند است. بدنبال قواعد مذکور، رابطه Supervises را با فرستادن کپی کلید اصلی موجودیت (والد) staffNo، Staff، به جدول (فرزند) Staff ، و ایجاد دومین کپی این ستون، به خاطر عمل بعنوان کلید خارجی نشان بدهید. این کپی ستون بنام supervisorStaffNo تغییر نام یافته است تا نمایش بهتری از عملکردش داشته باشد. شکل ۳-۹ (الف) مدل ER رابطه Staff *Supervises* Staff و شکل ۳-۹ (ب) جدول متناظر را نشان می دهد.

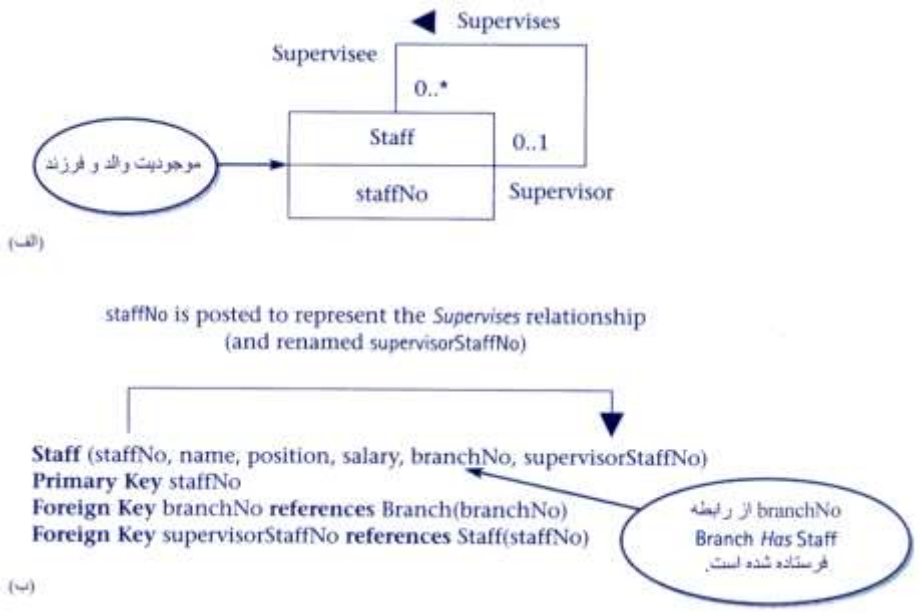
رابطه های باینری یک به یک (۱:۱)

ایجاد جدول برای نشان دادن رابطه ای ۱:۱ کمی مشکل تر است زیرا نمی توانید از کاردینالیتی برای کمک در شناسایی موجودیتهای والد و فرزند در رابطه استفاده کنید. به جای آن، شما نیاز به استفاده از مشارکت دارید تا شما را در اخذ تصمیم درباره اینکه آیا بهتر است رابطه ها را با ادغام موجودیتهای در برگیرنده در یک جدول نشان دهید یا دو جدول ایجاد کنید و کپی کلید اصلی را از یکی به دیگری بفرستید یاری دهد. ما چگونگی ایجاد جداول برای نشان دادن محدودیتهای مشارکت فوق را در نظر می گیریم:

(۱) مشارکت *اجباری* در *دو* طرف رابطه ۱:۱
(۲) مشارکت *اجباری* در یک طرف رابطه ۱:۱
(۳) مشارکت *اختیاری* در *دو* طرف رابطه ۱:۱.

شکل ۳-۹

رابطه *:۱ Staff *Supervises* Staff: (الف) مدل ER ; (ب) نمایش بصورت جدول با استفاده از DBDL.



مشارکت اجباری در دو طرف رابطه ۱:۱

در این مورد، شما باید موجودیتهای درگیر را در یک جدول ادغام کنید و یکی از کلیدهای اصلی موجودیتهای اصلی را به عنوان کلید اصلی جدول جدید انتخاب کنید، در حالیکه دیگری را بعنوان کلید فرعی انتخاب کرده اید.

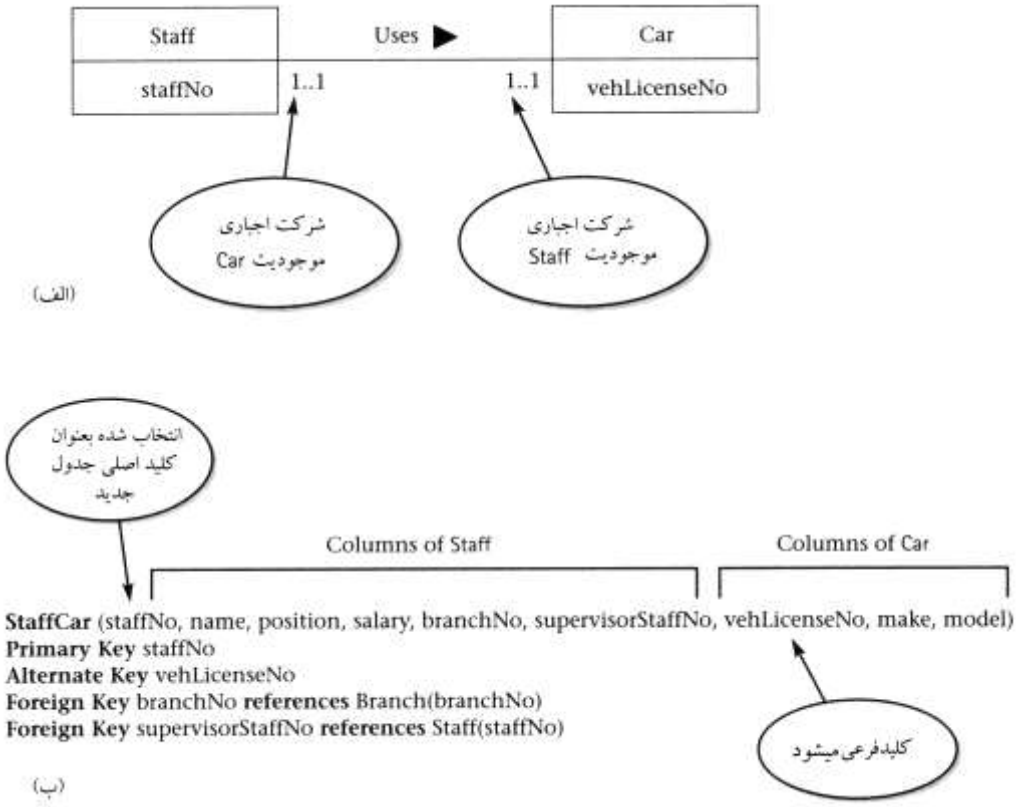
ما نمونه ای از چنین رابطه ای را در شکل ۱۶-۸ نداریم. با این وجود، اجازه دهید اینکه چگونه رابطه ۱:۱ با نام Staff *Uses* Car را با مشارکت اجباری برای هر دو موجودیت نشان دهیم را در نظر بگیریم، که در شکل ۴-۹ (الف) نشان داده شده است. کلید اصلی برای موجودیت Car شماره مجوز وسیله نقلیه (vehLicenseNo) است، و صفات دیگر make و model است. در این مثال، همه صفات موجودیتهای Car و Staff را در داخل یک جدول قرار دهید. سپس یکی از کلیدهای اصلی را، که همان staffNo بوده، بعنوان کلید اصلی جدول جدید انتخاب کنید، و دیگری نیز بعنوان کلید فرعی انتخاب می شود، که در شکل ۴-۹ (ب) نشان داده شده است.

در موردی که رابطه ۱:۱ با شرکت اجباری در هر دو طرف یک یا چند صفت دارد، این صفتها نیز باید در جدول گنجانده شوند تا رابطه را نشان دهند. برای مثال، اگر رابطه **Staff** *Uses* **Car** صفتی بنام **dateStart** داشت، این صفت همچنین بصورت ستونی در جدول **StaffCar** پدیدار می شود.

توجه کنید که تنها زمانی امکان ادغام دو موجودیت در یک جدول وجود دارد که رابطه دیگری همچون رابطه *:۱ بین این موجودیتها وجود نداشته باشد که مانع از ادغام دو موجودیت بشود. اگر این مورد وجود داشته باشد، معمولا لازم خواهد بود که رابطه Staff *Uses* Car را با استفاده از مکانیزم کلید اصلی/ خارجی نشان دهید. در بخش (۳) درباره اینکه چگونه موجودیتهای والد و فرزند را در این نوعی از موقعیتها طراحی کنیم به طور مختصر بحث می کنیم.

شکل ۴-۹

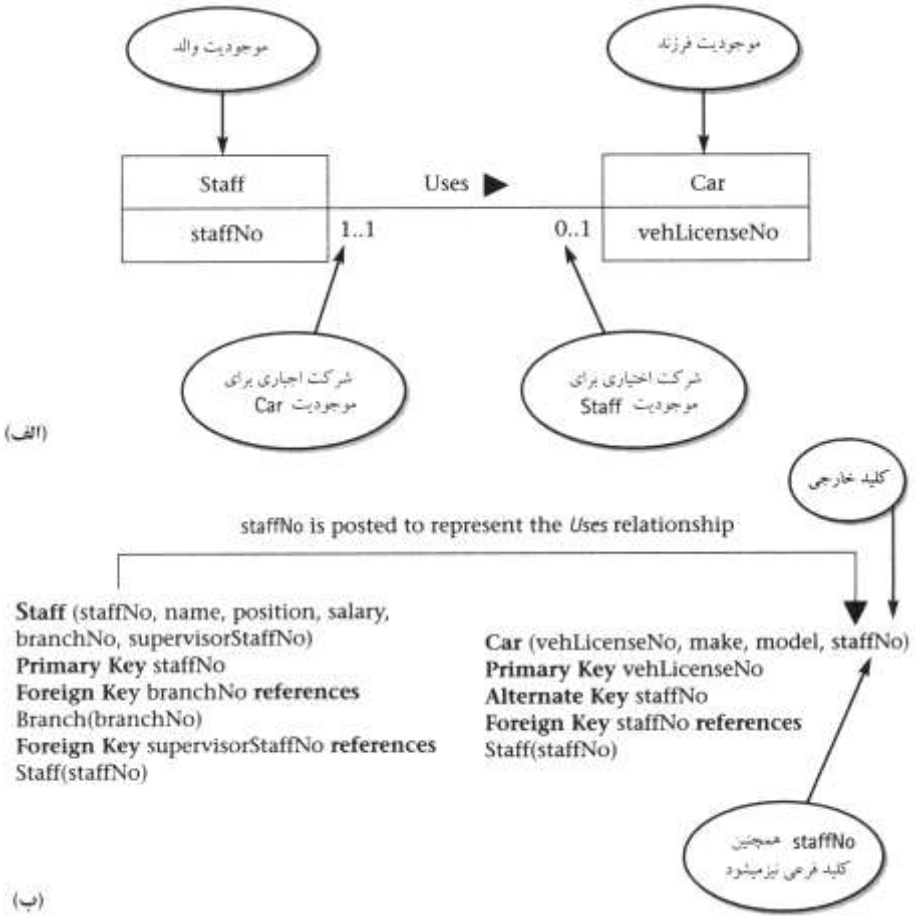
رابطه ۱:۱ Staff *Uses* car با شرکت اجباری برای هر دو موجودیت (الف) مدل ER ; (ب) نمایش بصورت جدول با استفاده از DBDL.



شرکت اجباری در یک طرف رابطه ۱:۱

در این مورد، شما می توانید تا موجودیتهای والد و فرزند را برای رابطه ۱:۱ با استفاده از محدودیتهای مشارکت مشخص کنید. موجودیتی که شرکت اختیاری در رابطه دارد بعنوان موجودیت والد در نظر می گیریم، و موجودیتی که شرکت اجباری در رابطه دارد به عنوان موجودیت فرزند در نظر گرفته می شود. همانطور که در بالا شرح داده شد، کپی کلید اصلی موجودیت والد در جدول موجودیت فرزند قرار می گیرد.

حال اجازه بدهید ببینیم که چگونه شما رابطه ۱:۱ Staff *Uses* Car با شرکت اجباری را فقط برای موجودیت Car نشان خواهید داد، همانطور که در شکل ۵-۹(الف) نشان داده شده است. موجودیتی که شرکت اختیاری در رابطه (Staff) دارد به عنوان موجودیت والد درنظر گرفته می شود، و موجودیت با شرکت اجباری در رابطه (Car) به عنوان موجودیت فرزند مشخص می شود. بنابراین، کپی کلید اصلی موجودیت (والد) Staff، staffNo، در جدول (فرزند) Car قرار داده می شود، همانطور که در شکل ۵-۹(ب) نشان داده شده است. در این مورد، staffNo همچنین بعنوان کلید فرعی جدول Car نیز می شود.



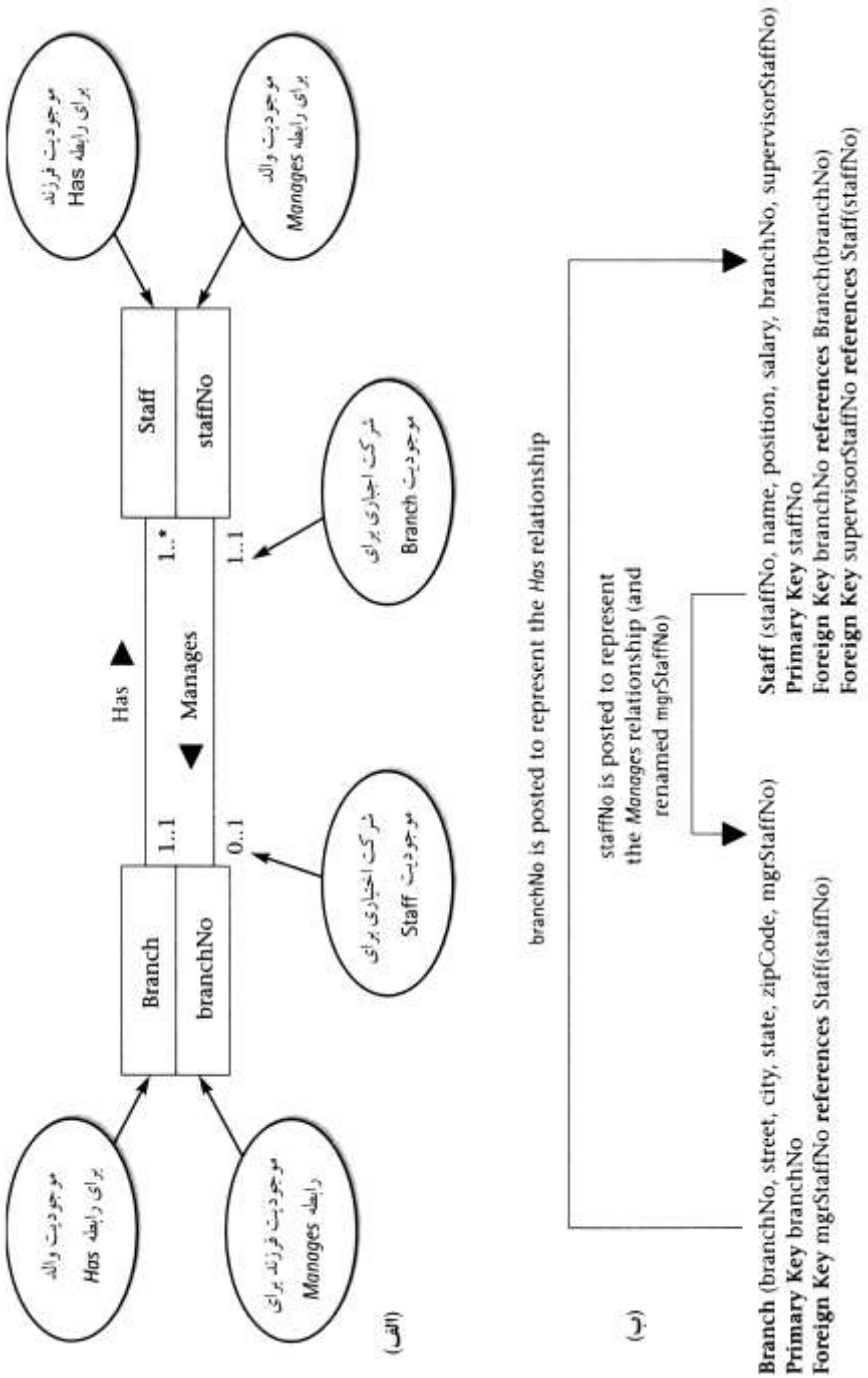
شکل ۵-۹

رابطه ۱:۱ Staff *Uses* Car با مشارکت اجباری برای موجودیت Car و مشارکت اختیاری برای موجودیت Staff: (الف) مدل ER؛ (ب) نمایش بصورت جدول با استفاده از DBDL.

شکل ۱۶-۸ مثال دومی از رابطه ۱:۱ با شرکت اجباری فقط در یک طرف دارد، بنام Staff *Manages* Branch با شرکت اجباری تنها برای موجودیت Branch است. با پیروی از قواعد داده شده بالا، موجودیت Staff بعنوان موجودیت والد و موجودیت Branch بعنوان موجودیت فرزند تعیین شده است. بنابراین، کپی کلید اصلی موجودیت (والد) Staff، staffNo ، در جدول (فرزند) Branch قرار داده می شود و به علت نشان دادن وضوح هر چه بیشتر هدف کلید خارجی در جدول Branch به نام mgrStaffNo تغییر نام می یابد. شکل ۶-۹ (الف) مدل ER رابطه Staff *Manages* Branch و شکل ۶-۹ (ب) جداول متناظر را نشان می دهند.

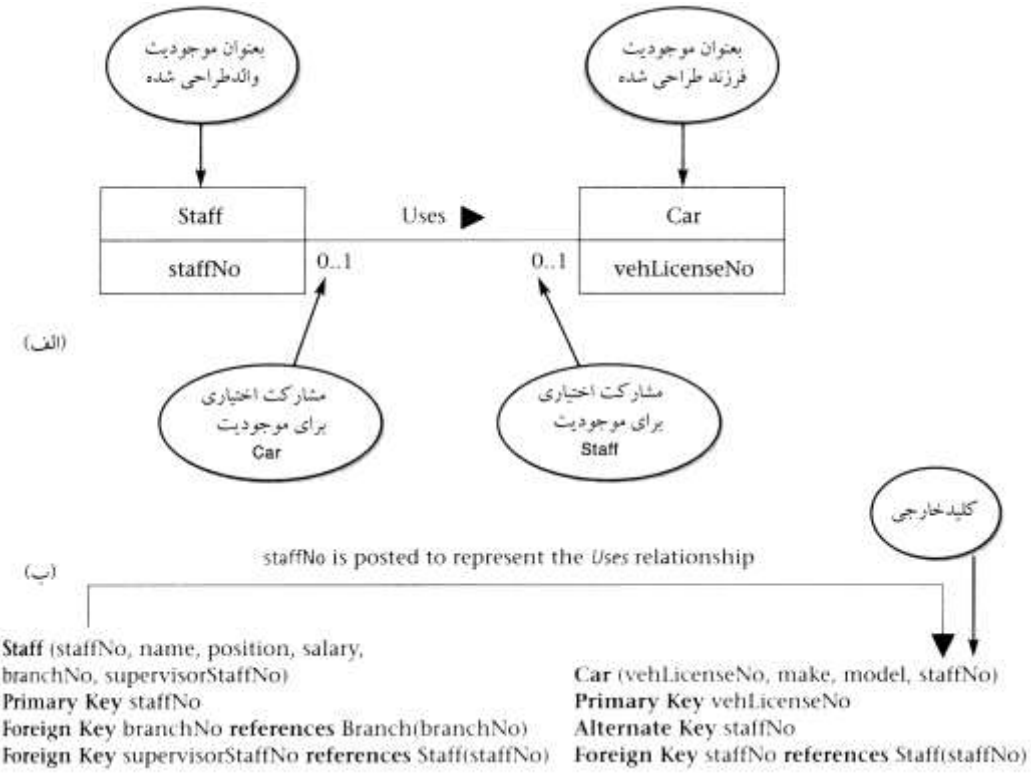
شکل ۶-۹

رابطه ۱:۱ Staff *Manages* Branch با شرکت اجباری برای موجودیت Branch و شرکت اختیاری برای موجودیت Staff : (الف) مدل ER ;(ب) نمایش بصورت جدول با استفاده از DBDL.



شکل ۷-۹

رابطه ۱:۱ **Staff** *Uses* **Car** با شرکت اختیاری برای هر دو موجودیت: (الف) مدل ER ; (ب) نمایش بصورت جداولی با استفاده از DBDL .



در موردی که رابطه یک به یک با شرکت اجباری برای یک موجودیت در رابطه یک یا چند صفت دارد، این صفات بایستی کلید اصلی را به جدول فرزند بفرستند. برای مثال، اگر رابطه Staff *Manages* Branch صفتی بنام dateStart داشت، این صفت باید بصورتی ستونی در جدول Branch بهمراه کپی staffNo نیز وجود داشته باشد.

شرکت اختیاری در دو طرف رابطه ۱:۱

در این مورد، طراحی موجودیتهای والد و فرزند دلخواه است مگر اینکه بتوانید درباره رابطه بیشتر بدانید که به شما کمک کند به تصمیمی در یک طرف برسید.

اجازه دهید در نظر بگیریم که چگونه باید رابطه ۱:۱ Staff *Uses* Car را نشان بدهید، با شرکت اختیاری در دو طرف رابطه، که در شکل ۷-۹ (الف) نشان داده شده است. (توجه کنید که بحثی که مطرح می شود در عین حال مربوط به رابطه های ۱:۱ با شرکت اجباری برای هر دو موجودیت نیز می باشد جائیکه شما نمی توانید گزینه ای را انتخاب کنید تا همه چیز را در یک جدول قرار دهید.) اگر شما اطلاعات اضافی دیگری بجهت انتخاب موجودیتهای والد و فرزند ندارید، انتخاب شما اختیاری است. بعبارت دیگر، گزینه ای که دارید فرستادن کپی کلید اصلی موجودیت Staff به موجودیت Car است، و یا بلعکس.

با این وجود، اجازه دهید فرض کنیم شما فهمیدید که اکثر ماشین ها، نه تمامی آنها، توسط پرسنل استفاده می شوند و تنها بعضی از پرسنل از ماشین استفاده می کنند. حالا شما می توانید بگویید که موجودیت car، اگر چه اختیاری است، اجباری تر از موجودیت Staff می باشد. بنابراین شما می توانید Staff را بعنوان موجودیت والد و Car را بعنوان موجودیت

فرزند در نظر بگیرید، و کپی کلید اصلی موجودیت Staff (staffNo) را به داخل جدول Car بفرستید، که در شکل ۷-۹ (ب) نشان داده شده است. (ترکیب جداول Car و Staff مشابه مثالی است که در بالا مطرح شد که رابطه ۱:۱ با شرکت اجباری تنها در یک طرف را نشان می داد.)

رابطه های بازگشتی یک به یک (۱:۱)

برای رابطه بازگشتی ۱:۱، شما باید از قواعد مربوط به اشتراک رابطه که در بالا برای رابطه ۱:۱ توضیح داده شد پیروی کنید. با این وجود، در این مورد خاص رابطه ۱:۱، موجودیت در دو طرف رابطه مشابه هستند. برای رابطه بازگشتی ۱:۱ که شرکت اجباری در دو طرف دارد، شما باید رابطه بازگشتی را به صورت تک جدول با دو کپی از کلید اصلی نشان دهید. همانطور که قبلا گفتیم، یک کپی کلید اصلی کلید خارجی را نمایش میدهد و بایستی جهت نمایش رابطه تغییر نام بیابد.

برای رابطه بازگشتی ۱:۱ با شرکت اجباری تنها در یک طرف، می توانید جدولی با دو کپی کلید اصلی همچون بالا ایجاد کنید، یا جدول جدیدی ایجاد کنید که رابطه را نشان دهد. جدول جدید تنها دو ستون خواهد داشت، که شامل دو کپی کلید اصلی است. مانند قبل کپی های کلید های اصلی بصورت کلیدهای خارجی عمل کرده و جهت نشان دادن هدف هر یک در جدول تغییر نام می یابند.

برای رابطه بازگشتی ۱:۱ با شرکت اختیاری در هر دو طرف، شما باید مانند آنچه در بالا شرح داده شد جدول جدیدی ایجاد کنید.

خلاصه ای از اینکه چگونه موجودیتهای والد و فرزند را در رابطه شناسایی کنیم

جدول ۱-۹ خلاصه ای از چگونگی شناسایی موجودیتهای والد و فرزند در رابطه را نشان می دهد.

جداول و صفات کلید خارجی را مستند کنید

در پایان مرحله ۱-۲، ترکیب کامل جداول ایجاد شده از مدل داده منطقی را مستند کنید. جدولهای مربوط به دید شعبه کاربرد پایگاه داده *StayHome* در شکل ۸-۹ نشان داده شده است. حال که هر جدول مجموعه کاملی از ستونها را دارد، شما در موقعیتی هستید که کلیدهای اصلی و یا فرعی جدید را مشخص کنید. این عمل یقینا برای موجودیتهای ضعیفی که به فرستادن کلید اصلی از موجودیت(های) والد جهت تشکیل کلید اصلی مربوط به خود تکیه دارند اهمیت دارند. برای مثال، موجودیت ضعیف Role حالا دارای کلید اصلی ترکیبی است که از کپی کلید اصلی موجودیت Video (catalogNo) و کپی کلید اصلی موجودیت Actor (actorNo) تشکیل شده است. به طور مشابه، موجودیت ضعیف Registration دارای کلید اصلی ترکیبی است که از دو کلید خارجی (branchNo و memberNo) تشکیل شده است.

جدول ۱-۹ خلاصه ای از اینکه چگونه موجودیتهای والد و فرزند رابطه را شناسایی کنیم.

رابطه	شناسایی
رابطه باینری *:۱	**والد**: طرف ۱; **فرزند**: طرف *
رابطه بازگشتی *:۱	**والد**: طرف ۱; **فرزند**: طرف *
رابطه باینری ۱:۱:	
(الف) شرکت اجباری در دو طرف	جدولها را در یک جدول ادغام کنید
(ب) شرکت اجباری در یک طرف	**والد**: طرف اختیاری; **فرزند**: طرف اجباری
(ج) شرکت اختیاری در دو طرف	دلخواه بدون اطلاعات اضافی
رابطه بازگشتی ۱:۱:	
(الف) شرکت اجباری در دو طرف	جدولها را داخل یکی با دو کپی کلید اصلی ادغام کنید
(ب) شرکت اجباری در یک طرف	مانند (الف), یا جدول جدیدی را برای نشان دادن رابطه ایجاد کنید
(ج) شرکت اختیاری در دو طرف	جدول جدیدی ایجاد کنید تا رابطه را نشان دهد

شکل ۸-۹

جدولهای دید Branch کاربرد پایگاه داده *StayHome*.

Actor (actorNo, actorName) **Primary Key** actorNo	**Branch** (branchNo, street, city, state, zipCode, mgrStaffNo) **Primary Key** branchNo **Alternate Key** zipCode **Foreign Key** mgrStaffNo **references** Staff(staffNo)
Director (directorNo, directorName) **Primary Key** directorNo	**Member** (memberNo, fName, lName, address) **Primary Key** memberNo
Registration (branchNo, memberNo, staffNo, dateJoined) **Primary Key** branchNo, memberNo **Foreign Key** branchNo **references** Branch(branchNo) **Foreign Key** memberNo **references** Member(memberNo) **Foreign Key** staffNo **references** Staff(staffNo)	**RentalAgreement** (rentalNo, dateOut, dateReturn, memberNo, videoNo) **Primary Key** rentalNo **Alternate Key** memberNo, videoNo, dateOut **Foreign Key** memberNo **references** Member(memberNo) **Foreign Key** videoNo **references** VideoForRent(videoNo)
Role (catalogNo, actorNo, character) **Primary Key** catalogNo, actorNo **Foreign Key** catalogNo **references** Video(catalogNo) **Foreign Key** actorNo **references** Actor(actorNo)	**Staff** (staffNo, name, position, salary, branchNo, supervisorStaffNo) **Primary Key** staffNo **Foreign Key** branchNo **references** Branch(branchNo) **Foreign Key** supervisorStaffNo **references** Staff(staffNo)
Telephone (telNo, branchNo) **Primary Key** telNo **Foreign Key** branchNo **references** Branch(branchNo)	**Video** (catalogNo, title, category, dailyRental, price, directorNo) **Primary Key** catalogNo **Foreign Key** directorNo **references** Director(directorNo)
VideoForRent (videoNo, available, catalogNo, branchNo) **Primary Key** VideoNo **Foreign Key** catalogNo **references** Video(catalogNo) **Foreign Key** branchNo **references** Branch(branchNo)	

گرامر DBDL می تواند برای نشان دادن محدودیتهای جامعیت بر روی کلیدهای خارجی، همانگونه که در بخش ۴-۲ خواهید دید تعمیم یابد. دیکشنری داده همچنین باید جهت نشان دادن وجود هر کلید اصلی و فرعی که در این مرحله جدیدا شناسایی شده است بروز شود. برای مثال، بدنبال ارسال کلیدهای اصلی، جدول RantalAgreement کلید اصلی جدیدی بدست آورده، که ترکیبی از memberNo ، videoNo ، و dateOut است.

مرحله ۲-۲ ساختارهای جدول را با استفاده از نرمال سازی بررسی کنید.

هدف

بررسی، با استفاده از نرمالسازی، تا هر جدول ساختار مناسبی داشته باشد.

هدف این مرحله بررسی گروه بندی ستونها در جداول ایجاد شده در مرحله ۱-۲ است. در این مرحله شما ترکیب هر جدول را با استفاده از قواعد نرمالسازی، جهت اجتناب از تکرار داده های غیر لازم بررسی می کنید.

شما باید مطمئن شوید که هر جدولی که در مرحله ۱-۲ ایجاد شده است حداقل در فرم نرمال سوم (3NF) است. اگر جدولهایی را شناسائی کنید که در 3NF نباشد، شاید بیانگر این باشد که قسمتی از مدل ER صحیح نبوده، یا هنگام ایجاد جدولها برای مدل مرتکب اشتباه شده اید. اگر لازم باشد، احتمالا نیاز داشته باشید جدولها یا مدل داده ای را دوباره بسازید.

مرحله ۳-۲ جدولها را جهت پشتیبانی تراکنشهای کاربر بررسی کنید

هدف

اطمینان از اینکه جدولها از تراکنشهای مورد نیاز دید پشتیبانی می کنند.

هدف این مرحله بررسی این است که آیا جدولهای ایجاد شده در مرحله ۱-۲ از تراکنش های مورد نیاز دید، همان طور که در مشخصات نیازمندی های کاربر مستند شده پشتیبانی می کنند. این نوع از بررسی قبلا در مرحله ۸-۱ برای اینکه مطمئن شویم مدل داده منطقی محلی مان از تراکنشهای مورد نیاز پشتیبانی می کند انجام گرفته است. در این مرحله، بررسی می کنید که جدولهای ایجاد شده در مراحل قبلی همچنین این تراکنشها را نیز پشتیبانی کنند، و بنابراین مطمئن می شوید که هیچ خطائی هنگام ایجاد جدولها بوجود نیامده است.

یک روش برای بررسی اینکه جدولها تراکنش را پشتیبانی می کنند بررسی نیازمندیهای داده تراکنش است تا مطمئن شویم که داده در یک یا چند جدول وجود دارد. همچنین، اگر تراکنش به داده در بیش از یک جدول نیاز داشته باشد شما باید این جدولها را بررسی کنید تا اطمینان حاصل کنید که از طریق مکانیزم کلید اصلی/ فرعی به هم مربوط شده باشند. ما این روش را با بررسی تراکنشهای تعیین شده در بخش ۴-۴-۴ نشان می دهیم. جدول ۲-۹ (الف) ثبت داده و تراکنشهای بهنگام سازی/ حذف را ارائه میکند و جدول ۲-۹ (ب) تراکنشهای پرس و جو برای دید شعبه *StayHome* ، بهمراه جدولهای مورد نیاز هر یک را ارائه می کند. در هر مورد، ستونهای مورد نیاز توسط تراکنشهای در برگرفته شده را، هرجا لازم باشد، بویژه آنهایی که در الحاق جداول به کار می رود مشخص کرده ایم.

جدول ۲-۹(الف) جداول موردنیاز توسط تراکنشهای ثبت داده و بهنگام سازی/ حذف دید شعبه *StayHome*.

تراکنش	جدول(های) موردنیاز
(a) جزئیات شعبه جدید را وارد کنید. (g) جزئیات شعبه را بهنگام/ حذف کنید.	**Branch (branchNo, street, city, zipCode, mgrStaffNo)** **Telephone (telNo, branchNo)** Foreign Key branchNo references Branch(branchNo)
(b) جزئیات عضو جدید پرسنل شعبه را وارد کنید. (h) جزئیات عضو پرسنل شعبه را بهنگام/ حذف کنید.	**Staff (staffNo, name, position, salary, branchNo, supervisorStaffNo)**
(c) جزئیات فیلمی که بتازگی منتشر شده است را وارد کنید. (i) جزئیات فیلم مشخصی را بهنگام/ حذف کنید.	**Video (catalogNo, title, category, dailyRental, price, directorNo)** Foreign Key directorNo references Director(directorNo) **Director (directorNo, directorName)** **Role (catalogNo, actorNo, character)** Foreign Key catalogNo references Video(catalogNo) Foreign Key actorNo references Actor(actorNo) **Actor (actorNo, actorName)**
(d) جزئیات کپی های فیلم جدید در شعبه معینی را وارد کنید. (j) جزئیات کپی فیلم معینی را بهنگام/ حذف کنید.	**VideoForRent (videoNo, available, catalogNo, branchNo)**
(e) جزئیات عضو جدیدی را که در شعبه معینی ثبت نام کرده را وارد کنید. (k) جزئیات عضو معینی را بهنگام/ حذف کنید.	**Member (memberNo, fName, lName, address)** **Registration (branchNo, memberNo, staffNo, dateJoined)** Foreign Key memberNo references Member(memberNo)
(f) جزئیات توافقنامه اجاره را برای عضوی که فیلمی را اجاره می کند وارد کنید. (l) جزئیات توافقنامه اجاره برای عضوی که فیلمی اجاره می کند را بهنگام/ حذف کنید.	**RentalAgreement (rentalNo, dateOut, dateReturn, memberNo, VideoNo)**

جدول ۲-۹ (ب) جداول موردنیاز توسط تراکنش های پرس وجوی دید شعبه *StayHome*.

تراکنش	جدول(های) موردنیاز
(m) جزئیات شعبه ها در شهر معینی را لیست کنید.	**Branch (branchNo, street, city, zipCode, mgrStaffNo)** **Telephone (telNo, branchNo)** Foreign Key branchNo references Branch(branchNo)
(n) نام، موقعیت شغلی، و حقوق پرسنل در شعبه معینی، بترتیب نام پرسنل لیست کنید.	**Staff (**staffNo**, name, position, salary, branchNo,** supervisorStaffNo**)**
(o) نام مدیر هر شعبه را، بترتیب نام شعبه لیست کنید.	**Branch (branchNo,** street**,** city**,** zipCode**, mgrStaffNo)** Foreign Key mgrStaffNo references Staff (staffNo) **Staff** (**staffNo**, **name**, position, salary, branchNo, supervisorStaffNo)
(p) عنوان، فهرست، و در دسترس بودن همه فیلمهای شعبه معینی را، بترتیب فهرست لیست کنید.	**Video** (**catalogNo**, **title**, **category**, dailyRental, price, directorNo) **VideoForRent** (videoNo, **available**, **catalogNo**, **branchNo**) Foreign Key catalogNo references Video (catalogNo)
(q) عنوان، فهرست، و در دسترس بودن همه فیلمها برای نام هنرپیشه معینی در شعبه مشخصی را، بترتیب عنوان لیست کنید.	**Actor (actorNo, actorName)** **Role (catalogNo, actorNo,** character**)** Foreign Key catalogNo references Video(catalogNo) Foreign Key actorNo references Actor(actorNo) **Video (catalogNo, title, category,** dailyRental**,** price**,** directorNo**)** **VideoForRent(**videoNo**, available, catalogNo, branchNo)** Foreign Key catalogNo references Video (catalogNo)
(r) عنوان، فهرست، و در دسترس بودن همه فیلمها برای نام کارگردان معین در شعبه مشخصی، را بترتیب عنوان لیست کنید.	**Director** (**directorNo**, **directorName**) **Video** (**catalogNo**, **title**, **category**, dailyRental, price, **directorNo**) Foreign Key directorNo references Director (directorNo) **VideoForRent** (videoNo, **available**, **catalogNo**, **branchNo**) **Foreign Key catalogNo** references Video (catalogNo)
(s) جزئیات تمام فیلمهایی که عضو مشخصی اجاره کرده است را لیست کنید.	**Video** (**catalogNo**, **title**, category, **dailyRental**, **price**, directorNo) **VideoForRent** (**videoNo**, **available**, **catalogNo**, branchNo) Foreign Key catalogNo references Video (catalogNo) **RentalAgreement** (rentalNo, **dateOut**, **dateReturn**, **memberNo**, **videoNo**) Foreign Key videoNo references VideoForRent (videoNo) Foreign Key memberNo references Member(memberNo) **Member** (**memberNo**, **fName**, **lName**, address)
(t) جزئیات همه کپی های فیلم معینی در شعبه مشخصی را لیست کنید.	**Video (catalogNo, title, category, dailyrental, price, directorNo)** **VideoForRent(videoNo,available, catalogNo, branchNo)** Foreign Key videoNo references VideoForRent (videoNo)

جدول ۲-۹ (ب) ادامه

تراکنش	جدول(های) موردنیاز
(u) عناوین تمام فیلمهای فهرست مشخصی را، بترتیب عنوان . لیست کنید.	**Video** (catalogNo, **title**, **category**, dailyRental, price, directorNo)
(v) مجموع تعداد فیلمها در هر فهرست فیلم در هر شعبه را، بترتیب شماره شعبه لیست کنید.	**Video** (**catalogNo**, title, **category**, dailyRental, price, directorNo) **videoForRent** (videoNo, available, **catalogNo**, **branchNo**) Foreign Key catalogNo references Video(catalogNo)
(w) مجموع هزینه فیلمهای تمام شعبه ها را لیست کنید.	**Video** (**catalogNo**, title, category, dailyRental, **price**, directorNo) **videoForRent** (videoNo, available, **catalogNo**, branchNo) Foreign Key catalogNo references Video(catalogNo)
(x) مجموع تمام فیلمهایی با ستارگی هر بازیگر بترتیب نام بازیگر را لیست کنید.	**Video** (**catalogNo**, title, category, dailyRental, price, directorNo) **Role** (**catalogNo**, **actorNo**, character) Foreign Key catalogNo references Video(catalogNo) Foreign Key actorNo references actor (actorNo) **Actor** (**actorNo, actorName**)
(y) مجموع کل عضوهای هر شعبه را که در 2006 ملحق شده اند، بترتیب شماره شعبه لیست کنید.	**Registration** (**branchNo**, **memberNo**, staffNo, **dateJoined**)
(z) مجموع اجاره روزانه ممکن برای فیلمهای هر شعبه را بترتیب شماره شعبه را لیست کنید.	**Video** (**catalogNo**, title, category, **dailyRental**, price, directorNo) **videoForRent** (videoNo, available, **catalogNo**, **branchNo**) Foreign Key catalogNo references Video (catalogNo)

با این تحلیل، شما نتیجه می گیرید که جدولهای نشان داده شده در شکل ۸-۹ از تمام تراکنشهای مربوط به دید شعبه *StayHome* پشتیبانی می کنند. در فصل ۱۷، پیاده سازی بعضی از تراکنشهای پرس و جورا که در جدول ۲-۹ (ب) نشان داده شده است را با استفاده از نرم افزار DBMS مایکروسافت اکسس نشان خواهیم داد.

نکته: مانند مرحله ۸-۱ که در فصل قبلی بحث شد، این عمل ممکن است کمی کار مشکلی به نظر برسد و یقینا هم همینطور است. در نتیجه، ممکن است موقتا بخواهید این مرحله را رد شوید. اما بهتر است که به جای آینده که هم مدل پیچیده میشود و برطرف کردن هر خطایی در مدل داده ای تان هزینه برخواهد بود، هم اکنون این مرحله را انجام دهید.

مرحله ۴-۲ محدودیتهای جامعیتی را تعریف کنید

هدف

تعریف محدودیتهای جامعیت تعیین شده در دید موجود در شرکت.

محدودیتهای جامعیت محدودیتهایی هستند که میخواهید وضع کنید تا از ناسازگار شدن پایگاه داده تان محافظت کنید. اگر چه کنترلهای DBMS بر جامعیت ممکن است وجود داشته باشد یا نه، به هر حال در این جا سوال مورد نظر ما نیست. در این

مرحله، برای شما فقط طراحی سطح بالا اهمیت دارد، یعنی، مشخص کردن محدودیتهای جامعیت اینکه چه *چیزی* مورد نیاز است، صرف نظر از اینکه *چگونه* این ممکن است بدست آید. بعد از شناسایی محدودیتهای جامعیت، مدل داده منطقی محلی خواهید داشت که کامل بوده و نمایش درستی از دید می باشد. اگر لازم باشد، می توانید طراحی فیزیکی پایگاه داده را از مدل داده منطقی محلی ایجاد کنید، همچون، نمایش نمونه اولیه سیستم برای کاربر.
ما پنج نوع از محدودیتهای جامعیت زیر را در نظر می گیریم:

- داده مورد نیاز،
- محدودیتهای دامنه ستون،
- جامعیت موجودیت ،
- جامعیت ارجاع ،
- قواعد تجاری .

داده مورد نیاز

بعضی از ستونها همیشه باید دارای مقدار باشند; بعبارت دیگر، نباید مجاز به نگه داشتن مقادیر Null باشند. برای مثال، هر عضو پرسنل باید موقعیت شغلی داشته باشند (همچون مدیر یا سرپرست). این محدودیتها باید زمانیکه ستونها (صفات) را در دیکشنری داده در مرحله ۳-۱ مستند می کنید شناسایی و مشخص شوند.

محدودیتهای دامنه ستون

هر ستونی دامنه ای (مجموعه مقادیر مجاز برای آن) دارد. برای مثال، موقعیت شغلی عضو پرسنل معاون، مدیر، سرپرست، دستیار، یا خریدار است بنابراین دامنه ستون position فقط و فقط شامل این مقادیر است. این محدودیتها بایستی زمانیکه دامنه های ستونها (صفتها) را در مرحله ۴-۱ برای داده انتخاب کردید شناسایی شده باشند.

جامعیت موجودیت

کلید اصلی موجودیت نمی تواند مقادیر Null را نگه دارد. برای مثال، هر رکورد از جدول Staff باید مقداری برای ستون کلید اصلی، staffNo داشته باشد. این محدودیتها بایستی زمانیکه کلیدهای اصلی را برای هر موجودیت در مرحله ۵-۱ شناسایی کردید در نظر گرفته شوند.

جامعیت ارجاع

کلید خارجی، هر رکورد جدول فرزند را به رکوردی در جدول والد که شامل مقدار کلید اصلی همتا است پیوند میدهد. جامعیت ارجاع به این معنی است که، اگر کلید اصلی شامل مقداری باشد، آن مقدار باید به رکورد موجود در جدول والد رجوع کند. برای مثال، ستون branchNo در جدول Staff عضوی از پرسنل را به رکورد موجود در جدول Branch یعنی جائیکه او کار میکند پیوند می دهد. اگر ستون branchNo تهی نباشد، باید شامل مقداری باشد که در ستون branchNo جدول Branch وجود دارد، یا عضو پرسنل به شعبه ای که وجود ندارد اختصاص یابد.
دو موضوع درباره کلید خارجی وجود دارد که باید به آن توجه کرد:

(۱) آیا مقادیر Null برای کلید خارجی مجاز است؟

برای مثال، آیا شما می توانید جزئیات عضو پرسنلی را بدون داشتن شماره شعبه کارمند ذخیره کنید؟ این موضوع به این معنی نیست که آیا شماره شعبه وجود دارد، بلکه بدین معنی است که آیا شماره شعبه بایستی مشخص شود. به طور کلی، اگر

مشارکت جدول فرزند در رابطه اجباری باشد، پس وجود Null مجاز نیست. از طرف دیگر، اگر شرکت جدول فرزند اختیاری باشد، پس مقادیر Null بایستی مجاز باشد.

(۲) چگونه از جامعیت ارجاع مطمئن باشیم

برای این کار، **محدودیتهای وجود**[۱] را مشخص کنید، که شرایطی را تحت اینکه کدام کلید اصلی یا خارجی احتمالا درج، بهنگام، یا حذف شده اند را تعریف می کنند. رابطه *:۱ Branch *Has* Staff را در نظر بگیرید. کلید اصلی جدول Branch، (branchNo) کلید خارجی در جدول Staff است. اجازه دهید موارد شش گانه زیر را در نظر بگیریم.

مورد ۱: رکوردی را داخل جدول فرزند (Staff) درج کنید
جهت اطمینان از جامعیت ارجاع، بررسی کنید ببینید که آیا ستون کلید خارجی (branchNo) رکورد جدید Staff بصورت null است یا به مقدار رکورد موجود Branch ست شده است.

مورد ۲: رکوردی را از جدول فرزند (Staff) حذف کنید
اگر رکورد جدول فرزند حذف شود، جامعیت ارجاع بی تاثیر باقی می ماند.

مورد ۳: کلید خارجی رکورد فرزند (Staff) را بروز کنید
این شبیه مورد ۱ است. برای اطمینان از جامعیت ارجاع، بررسی کنید ببینید که آیا کلید خارجی (branchNo) رکورد بروز شده Staff تهی است یا به مقداری رکورد Branch موجود ست شده است.

مورد ۴: رکوردی را داخل جدول والد (Branch) درج کنید
درج کردن رکوردی داخل جدول والد (Branch) جامعیت ارجاع را تحت تاثیر قرار نمی دهد; که بدون هیچ فرزندی بسادگی والد می باشد- بعبارت دیگر، شعبه ای بدون عضوی از پرسنل.

مورد ۵: رکوردی را از جدول والد (Branch) حذف کنید
اگر رکوردی از جدول والد حذف شود، اگر رکورد فرزندی وجود داشته باشد که به رکورد والد حذف شده اشاره کند در این صورت جامعیت ارجاع گم می شود. بعبارت دیگر، جامعیت ارجاع گم می شود اگر شعبه حذف شده در حال حاضر یک یا چند عضو داشته باشد که در آن کار می کنند. در اینجا چندین کار وجود دارد که می توانید در نظر بگیرید :

- NO ACTION اگر رکورد فرزند ارجاع شده دیگری هنوز وجود دارد، از انجام عمل حذف از جدول والد جلوگیری می کند. در مثال ما، شما نمی توانید شعبه را حذف کنید اگر هنوز عضوی از پرسنل مشغول به کار در آنجا هستند .
- CASCADE وقتیکه رکورد والد حذف شد، به طور اتوماتیک هر رکورد فرزند ارجاعی به آن حذف می شود. اگر رکورد فرزند حذف شده همچنین در رابطه دیگری به صورت رکورد والد عمل کند در اینصورت عمل حذف باید به رکوردهای موجود در این جدول فرزند نیز اعمال شود، و بنابراین به صورت آبشاری ادامه می یابد. بعبارت دیگر، حذف از جدول والد بصورت آبشاری به جدول فرزند ادامه می یابد. در مثال ما، 'حذف شعبه به طور اتوماتیک تمام عضوهایی که در آنجا کار می کنند را حذف می کند.' آشکار است که، در این موقعیت، این راهبرد معقول به نظر نمی رسد.
- SET NULL وقتی رکورد والد حذف شود، مقادیر کلید خارجی در تمام رکوردهای فرزند وابسته به طور اتوماتیک null می شوند. در مثال ما، ' اگر شعبه حذف شود، مشخص می کند که شعبه جاری برای آن عضوهایی از پرسنل که قبلا در

[۱] Existence Constraint

آنجا کار می کردند نامشخص است.' شما می توانید این راهبرد را وقتی در نظر بگیرید که ستونهای تشکیل دهنده کلید خارجی null را قبول کند، همانطور که در مرحله ۳-۱ تعریف شد.

- SET DEFAULT وقتی رکورد والد حذف شود، مقادیر کلید خارجی در تمام رکوردهای فرزند وابسته به طور اتوماتیک به مقادیر پیش فرض ست می شوند. در مثال ما، ' اگر شعبه حذف شود، نشان می دهد که انتصاب کنونی عضو پرسنلی که قبلا آنجا کار می کردند به شعبه (پیش فرض) دیگری انتصاب یافته اند.' زمانی می توانید این راهبرد را در نظر بگیرید که ستونهای تشکیل دهنده کلید خارجی مقادیر پیش فرض داشته باشند، همانطورکه در مرحله ۳-۱ تعریف شده است.
- NO CHECK وقتی رکورد والد حذف شود، هیچ کاری انجام نمی گیرد تا مطمئن شود که جامعیت ارجاع حفظ شده است. این راهبرد باید تنها در شرایط نهایی در نظر گرفته شود.

مورد ۶: کلید اصلی رکورد والد (Branch) *را بهنگام کنید*

اگر مقدار کلید اصلی رکورد جدول والد بهنگام شود، جامعیت ارجاع گم می شود در صورتیکه رکورد فرزند هنوز به مقدار کلید اصلی قدیمی رجوع می کند: یعنی، اگر شعبه بهنگام شده در حال حاضر پرسنلی داشته باشد که در آنجا کار می کند. جهت اطمینان از جامعیت ارجاع، راهبردهای شرح داده شده در بالا می توانند استفاده شوند. در مورد CASCADE، بهنگام سازی کلید اصلی رکورد والد در هر رکورد فرزند ارجاع شده ای منعکس میشود، و بنابراین به صورت آبشاری ادامه میابد.

محدودیتهای جامعیت ارجاع برای جدولهایی که برای دید شعبه *StayHome* ایجاد شده است در شکل ۹-۹ نشان داده شده است.

شکل۹-۹

محدودیتهای جامعیت ارجاع برای جدولهای دید شعبه *StayHome* .

Branch **Foreign Key** mgrStaffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION
Registration **Foreign Key** branchNo **references** Branch(branchNo) ON UPDATE CASCADE ON DELETE NO ACTION **Foreign Key** memberNo **references** Member(memberNo) ON UPDATE CASCADE ON DELETE NO ACTION **Foreign Key** staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION
RentalAgreement **Foreign Key** memberNo **references** Member(memberNo) ON UPDATE CASCADE ON DELETE NO ACTION **Foreign Key** videoNo **references** VideoForRent(videoNo) ON UPDATE CASCADE ON DELETE NO ACTION
Role **Foreign Key** catalogNo **references** Video(catalogNo) ON UPDATE CASCADE ON DELETE CASCADE **Foreign Key** actorNo **references** Actor(actorNo) ON UPDATE CASCADE ON DELETE NO ACTION
Staff **Foreign Key** branchNo **references** Branch(branchNo) ON UPDATE CASCADE ON DELETE NO ACTION **Foreign Key** supervisorStaffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL
Telephone **Foreign Key** branchNo **references** Branch(branchNo) ON UPDATE CASCADE ON DELETE CASCADE
Video **Foreign Key** directorNo **references** Director(directorNo) ON UPDATE CASCADE ON DELETE NO ACTION
VideoForRent **Foreign Key** catalogNo **references** Video(catalogNo) ON UPDATE CASCADE ON DELETE NO ACTION **Foreign Key** branchNo **references** Branch(branchNo) ON UPDATE CASCADE ON DELETE NO ACTION

Business rules

در پایان، محدودیتها را بنام قواعد تجاری در نظر بگیرید. بهنگام سازی روی موجودیتها احتمالا توسط قواعد تجاری که پوشاننده تراکنشهای جهان واقعی هستند محدود می شوند. برای مثال، *StayHome* قاعده تجاری دارد که از گرفتن بیش از ده فیلم توسط عضو جلوگیری بعمل می آورد.

تمام محدودیتهای جامعیت را مستند کنید
تمام محدودیتهای جامعیت را در دیکشنری داده جهت لحاظ در مرحله طراحی فیزیکی پایگاه داده مستند کنید.

مرحله ۵-۲ مدل داده منطقی محلی را با کاربران بازبینی کنید

هدف
اطمینان حاصل کنید که مدل داده منطقی محلی و مستنداتی که مدل را توصیف می کند نمایش صحیحی از دید می باشند.

مدل داده منطقی محلی دید، حالا دیگر باید کامل و کاملا مستند شده باشد. با این وجود، برای به پایان رساندن این مرحله باید مدل داده ای و مستندات مربوطه را بهمراه کاربران سیستم بازبینی کنید.

اگر شما پایگاه داده ای را طراحی می کنید که تنها یک دید دارد، حالا آماده اید که به مرحله بعدی که طراحی فیزیکی پایگاه داده است بروید، که در فصول ۱۲ تا ۱۶ شرح داده شده اند. اگر، با این وجود پایگاه داده ای طراحی می کنید که چندین دید دارند و از روش جامعیت دید (view integration) استفاده می کنید پس باید به مرحله ۳ متدلوژی بروید، که آنرا در فصل آینده شرح می دهیم.

خلاصه فصل

- ✓ هدف اصلی این مرحله تولید توصیف جدولها برای هر مدل داده منطقی محلی ایجاد شده در مرحله ۱ متدلوژی است. مجموعه جدولهای تولید شده باید بیانگر، موجودیتها، رابطه ها، صفات، و محدودیتها در مدل داده ای باشند.
- ✓ جدولها برای مدل داده منطقی محلی ایجاد، و ساختار آنها با استفاده از نرمالسازی بررسی می شوند.
- ✓ ساختار جدولها همچنین بررسی می شوند تا اطمینان حاصل شود که از تراکنش های تعریف شده توسط دید پشتیبانی می کنند.
- ✓ **محدودیتهای جامعیت** محدودیتهایی هستند که می خواهید اعمال کنید تا از ناسازگار شدن پایگاه داده محافظت کنند. پنج نوع از محدودیتهای جامعیت وجود دارد : داده مورد نیاز، محدودیتهای دامنه ستون، جامعیت موجودیت، جامعیت ارجاع، و قواعد تجاری.
- ✓ جهت اطمینان از جامعیت ارجاع، محدودیتهای وجود را مشخص کنید، که شرایطی را تعریف می کنند که کلید اصلی یا کلید خارجی احتمالا درج، بهنگام، یا حذف شده است.
- ✓ چندین راهبرد جهت درنظرگرفتن موقعیتی که رکورد فرزند رجوع به رکورد والدی دارد که سعی می کنید حذف یا بروز سازید عبارتند از: SET DEFAULT, SET NULL, CASCADE, NO ACTION، و NO CHECK.

فصل ۱۰

طراحی منطقی پایگاه داده – مرحله ۳

آنچه در این فصل خواهید آموخت :

- چگونه مدلهای داده منطقی محلی را داخل مدل داده منطقی سراسری شرکت ادغام کنیم.
- چگونه مطمئن شویم که مدل سراسری حاصل صحیح بوده و نمایش مناسبی از شرکت (یا بخشی از شرکت) در دست مدل است.

این فصل سومین مرحله از متدولوژی طراحی پایگاه داده منطقی ما را می پوشاند. این مرحله اختیاری بوده و تنها زمانی مورد نیاز است که قصد دارید پایگاه داده پیچیده با چندین دید را ایجاد کنید و این دیدها (تماما و یا بخشی از آن) را با استفاده از روش **جامعیت دید** مدیریت کنید. بنابراین فرض کنیم شما از این روش استفاده کرده اید، پس بایستی مدلهای داده ای را که هر دید را نشان می دهد را در مرحله ۱ و ۲ متدلوژی ایجاد کرده باشید. حال باید آماده باشید این مرحله را با دو یا چند مدل داده منطقی محلی آغاز کنید.

در فصل ۴، ما چندین دید کاربر کاربرد پایگاه داده *StayHome* را، بنامهای Director، Manager، Supervisor، Assistant و Buyer مشخص کردیم. بدنبال تحلیل نیازمندی های هر دید کاربر، تصمیم گرفتیم که این دیدها را با استفاده از ترکیب روشهای جامعیت دید و متمرکز شده مدیریت کنیم. ما از روش متمرکز شده استفاده کردیم تا نیازمندیها را برای دیدهای کاربر Manager، Supervisor و Assistant در دیدی بنام *Branch* ادغام کنیم و همچنین نیازمندیها را برای دیدهای کاربر Director و Buyer در داخل دیدی بنام *Business* ادغام کنیم. در فصل ۸ و ۹، از دید شعبه جهت نشان دادن ساختن مدل داده منطقی محلی با استفاده از مراحل ۱ و ۲ متدلوژی استفاده کردیم. مدل ER در شکل ۱۶-۸ و توصیف جداول در شکل ۸-۹ نشان داده شده است.

در این فصل ، ابتدا مشخصات نیازمندیهای کاربر را برای دید تجاری *StayHome* ارائه می کنیم. در اینجا ساختن مدل داده منطقی محلی را برای این دید نشان نمی دهیم ولی به جای آن اجزای مهم مدل منطقی را که مدل ER و توصیف جدولهای مبتنی بر این مدل می باشند را ارائه می کنیم. سپس جهت نشان دادن مرحله ۳ متدلوژی، از مدل داده منطقی محلی برای دید های Business و Branch استفاده می کنیم.

۱۰-۱ دید تجاری *StayHome*

در این بخش، مشخصات نیازمندیهای کاربر و همچنین مدل داده منطقی محلی متناظر را برای دید تجاری *StayHome* ارائه می دهیم.

نکته: اگر نیازمندیهای مطرح شده در این فصل را خوانده و سپس سعی کنید که مراحل ۱ و ۲ متدلوژی را برای خودتان تمرین کنید مطالب گفته شده سودمندتر خواهد بود. سپس می توانید راه حل خود را با راه حل های نمونه مقایسه و بررسی کنید.

۱-۱-۱۰ مشخصات نیازمندیهای کاربر

مشخصات نیازمندی های دید تجاری به دو بخش تقسیم می شود. اولین بخش داده استفاده شده توسط دید تجاری و دومی نمونه هایی را درباره اینکه داده چگونه استفاده شود را فراهم می کند (یعنی، تراکنشهایی که پرسنل با دید تجاری مرتبط شده تا بر روی داده انجام بگیرند).

نیازمندیهای داده

جزئیاتی که در مورد شعبه *StayHome* نگه داری می شود آدرس شعبه و شماره تلفن می باشد. هر شعبه دارای شماره شعبه ای است که، در سراسر شرکت منحصر بفرد است.

هر شعبه *StayHome* پرسنلی دارد، که دارای مدیر است. جزئیاتی که درباره عضو پرسنل باید نگه داشته شود یکی نام، موقعیت شغلی، و حقوق است. هر عضو پرسنل دارای شماره پرسنلی است، که در سراسر شرکت منحصر بفرد است.

به هر شعبه *StayHome* انباری از فیلمها اختصاص یافته است. جزئیاتی که باید درباره فیلمها نگه داری شوند شماره فهرست، شماره فیلم، عنوان، فهرست، نرخ اجاره روزانه، و قیمت خرید است. شماره فهرست هر فیلم را منحصرا شناسایی می کند. با این وجود، در بیشتر موارد بیش از چندین کپی از فیلم ها در شعبه وجود دارد، که در این صورت کپی ها منحصرا با استفاده از شماره فیلم مشخص می شوند.

هر شعبه *StayHome* فیلمها را از تامین کنندگان فیلم تهیه می کند. جزئیاتی که درباره تامین کنندگان فیلم نگهداری می شود شماره تامین کننده، نام، آدرس، شماره تلفن، و وضعیت می باشد. سفارشات برای فیلمها با تامین کنندگان می باشد و جزئیات برای سفارش فیلم عبارت از شماره سفارش، شماره تامین کننده، آدرس تامین کننده، شماره فهرست فیلم، عنوان فیلم، قیمت خرید فیلم، تعداد، تاریخ سفارش، تاریخ دریافت، و آدرس شعبه ای که سفارش را دریافت کرده است می باشد.

مشتری *StayHome* ابتدا باید بصورت عضوی از شعبه محلی *StayHome* ثبت نام کند. جزئیاتی که درباره عضو نگه داری می شوند نام، آدرس، و تاریخی که عضو در شعبه ثبت نام کرده است می باشد. هر عضو دارای شماره عضویت است، که در همه شعبه های شرکت منحصر بوده و حتی وقتی که عضو در بیش از یک شعبه ثبت نام می کند نیز استفاده می شود.

جزئیاتی که درباره فیلم اجاره شده نگهداری می شود عبارتند از شماره اجاره، شماره و نام کامل عضو، شماره فیلم، عنوان، و نرخ اجاره روزانه، و تاریخی که فیلم اجاره و بازگردانده شده است. شماره اجاره در تمام شرکت منحصر است.

نیازمندیهای تراکنش

ثبت داده ها

(a) جزئیات مربوط به فیلمهایی که جدیدا اکران شده را وارد کنید (مانند جزئیات فیلم بنام *Independence Day*).

(b) جزئیات فروشنده فیلم را وارد کنید (مانند فروشنده ای بنام *Worldview Videos*) .

(c) جزئیات سفارش فیلم را وارد کنید (همچون سفارش ۱۰ کپی از *Saving Private Ryan* برای شعبه B002).

بهنگام سازی/ حذف داده

(d) جزئیات فیلم معینی را بهنگام/حذف کنید .

(e) جزئیات فروشنده فیلم معینی را بهنگام/حذف کنید .

(f) جزئیات سفارش فیلم معینی را بهنگام/حذف کنید .

پرس و جو های داده

(g) نام، موقعیت، و حقوق پرسنل تمام شعبه ها را، بترتیب شماره شعبه لیست کنید.

(h) نام و شماره تلفن مدیر شعبه معینی را لیست کنید.

(i) شماره فهرست و عنوان همه فیلمهای موجود در شعبه معینی را، بترتیب عنوان لیست کنید.

(j) تعداد کپی های فیلم معینی را در شعبه بخصوصی لیست کنید.

(k) تعداد عضوهای هر شعبه را، بترتیب شماره شعبه لیست کنید.

(l) تعداد عضوهایی که این سال به هر شعبه ملحق شده اند، را بترتیب شماره شعبه لیست کنید.

(m) تعداد اجاره های فیلم در هر شعبه بین تاریخ مشخصی، بترتیب شماره شعبه لیست کنید.

(n) تعداد فیلمهای موجود در هر فهرست در شعبه معینی را، بترتیب فهرست لیست کنید.

(o) نام، آدرس، و شماره تلفن همه فروشندگان فیلم را، بترتیب شماره فروشنده لیست کنید.

(p) نام و شماره تلفن فروشندگان فیلم را لیست کنید.

(q) جزئیات همه سفارشات فیلم بهمراه فروشنده معین، بترتیب تاریخ سفارش لیست کنید.

(r) جزئیات همه سفارشات فیلم واقع در تاریخ مشخصی را لیست کنید.

(s) مجموع اجاره های روزانه برای فیلمها برای هر شعبه بین تاریخ های مشخصی، بترتیب شماره شعبه لیست کنید.

۲-۱-۱۰ مدل داده منطقی محلی

همانطور که بیاد داریم، به جای انجام مرحله ساختن مدل داده منطقی محلی دید تجاری ***StayHome***، فرض کردیم که این مدل با استفاده از مراحل ۱ و ۲ متدلوژی ایجاد می شود و اجزاء مهم مدل منطقی را ارائه می کند، که بنامهای:

- مدل ER ، که در شکل ۱-۱۰ نشان داده شده است
- جداول، که در شکل ۲-۱۰ نشان داده شده اند .

حال اجازه دهید از مدلهای داده منطقی محلی Branch و Business برای ساختن مدل داده منطقی سراسری برای کاربرد پایگاه داده ***StayHome*** استفاده کنیم .

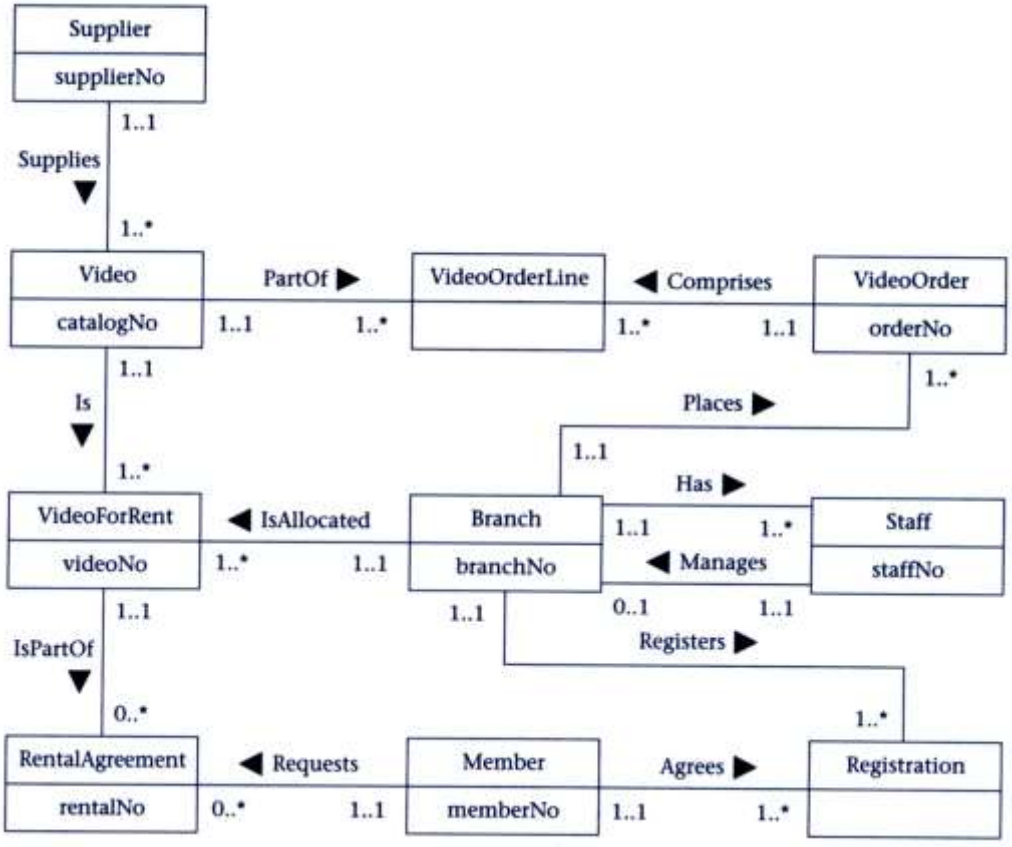
مرحله ۳ مدل داده منطقی سراسری را ساخته و آنرا بررسی کنید

هدف

ترکیب کردن مدلهای داده منطقی محلی منحصر داخل مدل داده منطقی سراسری که بیانگر شرکت (یا بخشی از شرکت) در حال مدل می باشد.

شکل ۱-۱۰

مدل ER دید تجاری *StayHome*



در این مرحله، مدل داده منطقی سراسری را با ادغام کردن مدلهای داده منطقی محلی منحصر هر دید تولید می کنید. بیاد بیاورید که مدل داده منطقی محلی می تواند بیانگر دید کاربر منحصر یا بیش از یک دید کاربر باشد، در حالیکه مدل داده منطقی سراسری بیانگر همه دید های کاربر می باشد. با ترکیب مدلها با همدیگر، شاید لازم باشد بررسی کنیم تا از نرمال بودن مدل سراسری و پشتیبانی آن از تراکنشهای موردنیاز اطمینان یابیم، همانگونه که در مراحل ۲-۲ و ۳-۲ فصل قبل مشاهده کردید. بنابراین، فقط نیاز دارید تا قسمتهایی از مدل را که طی فرآیند ادغام تغییر کرده است را بررسی کنید. در سیستمهای بزرگ، این بررسی به طور قابل توجه میزان بررسی دوباره ای که موردنیاز است را کاهش می دهد.

با وجود اینکه هر مدل داده منطقی محلی بایستی صحیح، جامع، و غیر مبهم باشد، هر مدل تنها نمایشی از دید شرکت می باشد. بعبارت دیگر، مدل دقیقا بیانگر عملکرد شرکت نیست، اما مدلی از یک یا چند دید میباشد و از اینرو نمی تواند کامل باشد.

بنابراین، وقتیکه شما مدل داده منطقی محلی را داخل مدل سراسری منحصری ادغام می کنید، باید برخوردهای بین دیدها و هرگونه همپوشانی موجود را برطرف سازید.

کارهایی که در مرحله ۳ موردنیاز است بدین صورت هستند:

- مرحله ۳-۱ مدلهای داده منطقی محلی را داخل مدل سراسری ادغام کنید
- مرحله ۳-۲ مدل داده منطقی محلی را بررسی کنید
- مرحله ۳-۳ مدل را از لحاظ رشد بیشتر بررسی کنید
- مرحله ۳-۴ مدل داده منطقی سراسری را با کاربران بازبینی نمائید

شکل ۲-۱۰

جدولهای دید تجاری *StayHome*.

Branch (branchNo, address, telNo, mgrStaffNo) **Primary Key** branchNo **Alternate Key** telNo **Foreign Key** mgrStaffNo **references** Staff(staffNo)	**Member** (memberNo, name, address) **Primary Key** memberNo
Registration (branchNo, memberNo, dateJoined) **Primary Key** branchNo, memberNo **Foreign Key** branchNo **references** Branch(branchNo) **Foreign Key** memberNo **references** Member(memberNo)	**RentalAgreement** (rentalNo, dateOut, dateReturn, memberNo, videoNo) **Primary Key** rentalNo **Alternate Key** memberNo, videoNo, dateOut **Foreign Key** memberNo **references** Member(memberNo) **Foreign Key** videoNo **references** Video(videoNo)
Staff (staffNo, name, position, salary, branchNo) **Primary Key** staffNo **Foreign Key** branchNo **references** Branch(branchNo)	**Supplier** (supplierNo, sName, sAddress, sTelNo, status) **Primary Key** supplierNo **Alternate Key** sTelNo
Video (catalogNo,title, category, dailyRental, price, supplierNo) **Primary Key** catalogNo **Foreign Key** supplierNo **references** Supplier(supplierNo)	**VideoForRent** (videoNo, available, catalogNo, branchNo) **Primary Key** videoNo **Foreign Key** catalogNo **references** Video(catalogNo) **Foreign Key** branchNo **references** Branch(branchNo)
VideoOrder (orderNo, dateOrdered, dateReceived, branchNo) **Primary Key** orderNo **Foreign Key** branchNo **references** Branch(branchNo)	**VideoOrderLine** (orderNo, catalogNo, quantity) **Primary Key** orderNo, catalogNo **Foreign Key** orderNo **references** VideoOrder(orderNo) **Foreign Key** catalogNo **references** Video(catalogNo)

مرحله ۱-۳ مدلهای داده منطقی محلی را داخل مدل سراسری ادغام کنید

هدف

ادغام مدلهای داده منطقی محلی منفرد داخل یک مدل داده منطقی سراسری.

تا به اینجا، شما برای هر مدل داده مدل ER، مجموعه جدولها، دیکشنری داده، و مستندات مربوطه را که محدودیتهای مدل را توصیف می کنند را ایجاد کرده اید. در این مرحله، از این عناصر جهت شناسایی تشابهات و تفاوتهای بین این مدلها، کمک به ادغام آنها استفاده می کنید.

برای کاربردهای پایگاه داده های ساده با تعداد نسبتا کوچکی از جداول/ موجودیتها، مقایسه مدلهای محلی، ادغام آنها با همدیگر، و برطرف کردن هرگونه تفاوت موجود، کار ساده ای است. با این وجود، در سیستمهای بزرگ، روش سیستماتیکی لازم است. ما روشی را که می تواند جهت ادغام مدلهای محلی با همدیگر و برطرف سازی هرگونه ناسازگاری یافت شده استفاده شود را ارائه می کنیم. بعضی از کارهایی که در اینجا باید انجام گیرند به صورت زیر هستند:

(۱) نامهای موجودیتها/ جدولها و کلیدهای اصلی شان را مرور کنید.
(۲) نامهای رابطه ها را مرور کنید.
(۳) موجودیتها/ جدولها را از مدل های داده منطقی ادغام کنید.
(۴) موجودیتها/ جدولهای منفرد را (بدون ادغام) برای هر مدل داده محلی شامل کنید.
(۵) رابطه ها را از مدلهای داده محلی ادغام کنید.
(۶) رابطه های منفرد را (بدون ادغام) برای هر مدل داده محلی شامل کنید.
(۷) موجودیتها/ جدولها و رابطه های گم شده را بررسی کنید.
(۸) کلیدهای خارجی را بررسی کنید.

(۹) محدودیتهای جامعیت را بررسی کنید.
(۱۰) مدل داده منطقی محلی را ترسیم کنید.
(۱۱) مستندات را بهنگام کنید.

در بعضی از کارهای بالا، ما از عبارت 'موجودیتها / جدولها' استفاده کرده ایم. این کار را انجام داده ایم زیرا ممکن است ترجیح بدهید که مدلهای ER و مستندات پشتیبان شان را بررسی کنید یا ترجیح دهید جدولهایی را که از مدل ER ایجاد شده اند را بررسی کنید، یا حتی از ترکیب هر دو روش استفاده کنید.

نکته شاید آسانترین راه در ترکیب چندین مدل داده محلی با همدیگر این باشد که ابتدا دو مدل را برای ایجاد مدل جدید با هم ترکیب کنیم، و سپس دیگر مدلهای داده محلی را تا زمانیکه تمام مدلهای محلی در مدل داده سراسری نهایی ارائه شوند با هم ترکیب کنیم. این شاید روش ساده تری نسبت به اینکه سعی کنیم همه مدلهای داده محلی را در یک زمان با هم ترکیب کنیم.

جهت اطمینان از اینکه مشابه را با مشابه مقایسه می کنید، بهتر است که هر مدل محلی بر اساس مراحل ۱و۲ متدلوژی ایجاد شده باشند. برای مثال، شاید مقایسه دو مدل که در یکی رابطه های افزونه و پیچیده را حذف کرده اید و در دیگری این روابط هنوز باقی است مشکل باشد.

نامهای موجودیتها/ جدولها و کلیدهای اصلی شان را مرور کنید

مرور نامهای موجودیتها / جدولهای موجود در مدلهای داده محلی می تواند ارزنده باشد. مشکلات زمانی ایجاد می شوند که دو یا چند موجودیت/ جدول:

- نام مشابه داشته باشند، ولی معنی متفاوت داشته باشند;
- مشابه باشند ولی دارای نام متفاوتی باشند.

شاید لازم باشد جهت برطرف ساختن این مشکلات محتوای داده هر موجودیت/ جدول را با هم مقایسه کنیم. بخصوص، ممکن است از کلیدهای اصلی جهت کمک به شناسایی موجودیتها/ جدولهای معادل که دارای نامهای متفاوتی در سراسر دیدها هستند استفاده کنید. مقایسه موجودیتها/ جدولها و کلیدهای اصلی دیدهای تجاری و شعبه *StayHome* در جدول ۱-۱۰ نشان داده شده است. موجودیتها/ جدولهایی که در هر دید مجزا هستند پر رنگ نشان داده شده اند.

نامهای رابطه ها را بازبینی کنید

این فعالیت مشابه آنچه که برای موجودیتها / جدولها شرح داده شد می باشد. مقایسه رابطه های موجود در دیدهای Business و Branch شرکت *StayHome* در جدول ۲-۱۰ نشان داده شده است. رابطه هایی که در هر دید مجزا هستند پر رنگ نشان داده شده اند.

جدول ۱-۱۰ مقایسه موجودیتها / جدولها و کلیدهای اصلی دیدهای شعبه و تجاری *StayHome*

دید Branch		دید Business	
جدول	کلیداصلی	جدول	کلیداصلی
Branch	branchNo	Branch	branchNo
Staff	staffNo	Staff	staffNo
Telephone	**telNo**		
Video	catalogNo	Video	catalogNo
VideoForRent	videoNo	VideoForRent	videoNo
		Supplier	**supplierNo**
		VideoOrder	**orderNo**
		VideoOrderLine	**orderNo, catalogNo**
RentalAgreement	rentalNo	RentalAgreement	rentalNo
Member	memberNo	Member	memberNo
Registration	branchNo, memberNo	Registration	branchNo, memberNo
Actor	**actorNo**		
Role	**catalogNo, actorNo**		
Director	**directorNo**		

ادغام موجودیتها / جدولها مدلهای داده محلی

شما باید نام و محتوای هر موجودیت/ جدول مدلهای ادغام شده را جهت مشخص کردن اینکه آیا موجودیتها/ جدولها بیانگر چیز مشابه بوده و باید ادغام شوند را بررسی کنید. فعالیتهای نوعی که باید انجام گیرند عبارتند از:

- موجودیتها / جدولهای با نام و کلید اصلی مشابه را با هم ادغام کنید.
- موجودیتها / جدولهای با نام مشابه و کلیدهای اصلی متفاوت را ادغام کنید.
- موجودیتها / جدولهای با نامها و کلیدهای اصلی مشابه یا متفاوت را ادغام کنید.

موجودیتها/ جدولهای با نام و کلید اصلی مشابه را با هم ادغام کنید عموما، موجودیتها / جدولهای با کلید اصلی مشابه بیانگر شیء 'جهان واقعی' مشابهی هستند و بایستی ادغام شوند. موجودیت/ جدول ادغام شده در برگیرنده صفات/ستونهایی از موجودیتها / جدولهای با حذف های تکراری هستند. برای مثال، شکل ۳-۱۰ ستونهای مرتبط با دو جدول Member که در دیدهای Branch و Business تعریف شده است را لیست کرده است. از آنجائیکه کلید اصلی هر دو جدول memberNo است، بنابراین باید این دو جدول را با ترکیب ستونهایشان ادغام کنید، از اینرو جدول ادغام شده حالا دارای تمام ستونهای مرتبط با جدول های اصلی است.

توجه کنید که بین دیدها درباره اینکه چگونه باید نام عضو را نشان دهید تضاد وجود دارد. در این موقعیت، شما باید (در صورت امکان) با کاربران هر دید درباره مشخص کردن نمایش نهایی مشورت کنید. در این مثال، دید شعبه با ستونهای fName و lName در مدل داده منطقی سراسری استفاده شده است.

جدول ۲-۱۰ مقایسه رابطه های دیدهای تجاری و شعبه *StayHome*

Branch view			Business view		
Entity	Relationship	Entity	Entity	Relationship	Entity
Branch	Has	Staff	Branch	Has	Staff
Branch	IsAllocated	VideoForRent	Branch	IsAllocated	VideoForRent
Branch	**Provides**	**Telephone**			
Branch	Registers	Member	Branch	Registers	Member
			Branch	**Places**	**VideoOrder**
			VideoOrder	**Comprises**	**VideoOrderLine**
			Supplier	**Supplies**	**Video**
Staff	Manages	Branch	Staff	Manages	Branch
Staff	**Supervises**	**Staff**			
Staff	**Processes**	**Registration**			
Video	Is	VideoForRent	Video	Is	VideoForRent
Video	**Features**	**Role**	**Video**	**PartOf**	**VideoOrderLine**
VideoForRent	IsPartOf	RentalAgreement	VideoForRent	IsPartOf	RentalAgreement
Member	Agrees	Registration	Member	Agrees	Registration
Member	Requests	RentalAgreement	Member	Requests	RentalAgreement
Actor	**Plays**	**Role**			
Director	**Directs**	**Video**			

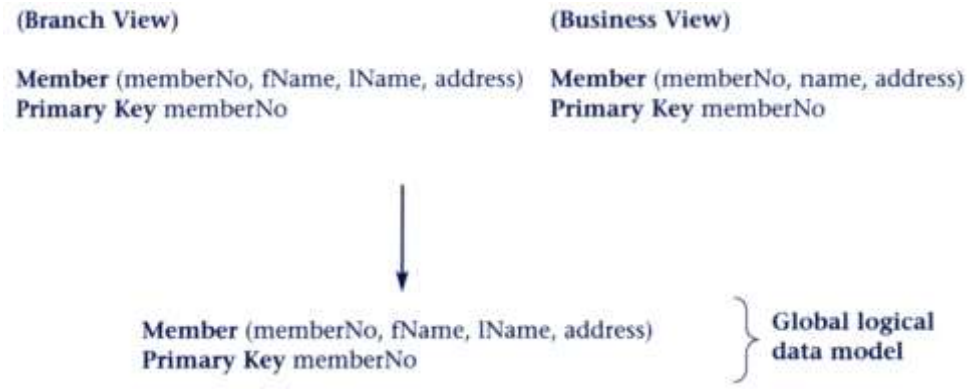
موجودیتها/ جدولهای با نام مشابه با استفاده از کلیدهای اصلی متفاوت را ادغام کنید در بعضی موقعیتها، ممکن است دو موجودیت/ جدول با یک نام و کلیدهای کاندید مشابه، ولی با کلیدهای اصلی متفاوت بیابید. در این مورد، موجودیتها / جدولها بایستی همانگونه که در بالا شرح داده شد با یکدیگر ادغام شوند. با این حال، لازم است یک کلید را بعنوان کلید اصلی انتخاب کنید، و دیگری بعنوان کلیدهای فرعی باشند.

موجودیتها / جدولهای با نامهای متفاوت با استفاده از کلیدهای اصلی متفاوت یا مشابه را ادغام کنید. در بعضی موارد، ممکن است موجودیتها / جدولهایی را مشخص کنید که دارای نام متفاوت ولی هدف مشابهی هستند. این موجودیتها /جدولهای همسان ممکن است بسادگی توسط موارد زیر شناخته شوند :

- نامشان، که هدف مشابهشان را نشان میدهد;
- محتوای آنها، بخصوص، کلید اصلی آنها;
- ارتباط آنها با روابط مشخصی.

شکل ۳-۱۰

ادغام جدولهای Member دیدهای Branch و Business.



موجودیتها / جدولهای منفرد به هر مدل داده محلی را نیز شامل کنید

کارهای قبلی باید همه موجودیتها/ جدولهای مشابه را شناسایی کند. بقیه موجودیتها/ جدولهای موجود در مدل سراسری بدون تغییر باقی مانده اند. از جدول ۱-۱۰، موجودیتها/ جدولهای منفرد دید شعبه عبارتند از Telephone، Actor ،Role ، و Director ، و موجودیتها/ جدولهای دید تجاری عبارتند از Supplier، VideoOrder ، و VideoOrderLine .

رابطه های مدلهای داده محلی را ادغام کنید

در این مرحله، باید نام و هدف هر رابطه را در همه مدل داده ای بررسی کنید. قبل از ادغام رابطه ها، بایستی هر نوع برخورد و تضادی را بین رابطه ها مانند تفاوت موجود بین محدودیتهای کثرت گرایی برطرف سازید. فعالیتهای مرتبط به این مرحله شامل ادغام رابطه های با نام مشابه و هدف مشابه می باشد، و بعد از آن ادغام رابطه های با نامهای متفاوت ولی هدف مشابه می باشد. برای مثال، رابطه Branch *Has* Staff در هر دو دید Branch و Business رخ داده است. ما فرایند ادغام رابطه هایی مشابه بین دیدها را در جدول ۲-۱۰ نشان دادیم.

رابطه های منفرد هر مدل داده محلی را (بدون ادغام) شامل کنید

دوباره، کار قبلی باید همه رابطه های مشابه را مشخص سازد (با این تعریف که، بایستی بین موجودیتها/ جدولهای مشابهی باشند، که بزودی می خواهند با هم ادغام شوند). بقیه رابطه های باقیمانده بدون تغییر در مدل داده سراسری شامل می شوند. از جدول ۲-۱۰، رابطه های منفرد دید شعبه شامل Staff ، Staff *Supervises* Staff، Branch *Provides* Telephone، Registration *Processes*، Video *Features* Role، Actor *Plays* Role، و Director *Directs* Video میباشد. رابطه های منفرد دید Business عبارتند از Branch *Places* VideoOrder، VideoOrder *Comprises* VideoOrderLine، Supplier *Supplies* Video، و Video *partOf* VideoOrderLine.

موجودیتها / جدولها و رابطه های گم شده را مشخص کنید

شاید یکی از مشکل ترین کارها در تولید مدل سراسری شناسایی موجودیتها / جدولها و رابطه های گم شده بین مدلهای داده محلی متفاوت باشد. اگر مدل داده حقوقی برای شرکت وجود داشته باشد، این ممکن است بیانگر این باشد که رابطه ها و موجودیتها / جدولها در هیچ مدل داده محلی ظاهر نشده اند. بنابراین، از لحاظ مقیاس پیشگیرانه، وقتی با کاربران دید خاصی مصاحبه می کنید، از آنها بخواهید تا توجه خاصی را نسبت به موجودیتها/ جدولها و رابطه های موجود در دیگر دیدها داشته باشند. در غیر این صورت، صفات/ ستونهای هر موجودیت/ جدول را بررسی کنید و ارجاعات موجودیتها/ جدولها در دیگر

مدلهای داده محلی را نیز مشاهده کنید. ممکن است بفهمید که در یک مدل داده محلی صفت/ ستون مرتبط با موجودیت/ جدولی دارید که متناظر با کلید اصلی، کلیدفرعی، یا حتی **صفت/ستون غیر کلید** در مدل داده محلی دیگر می باشد.

کلیدهای خارجی را بررسی کنید

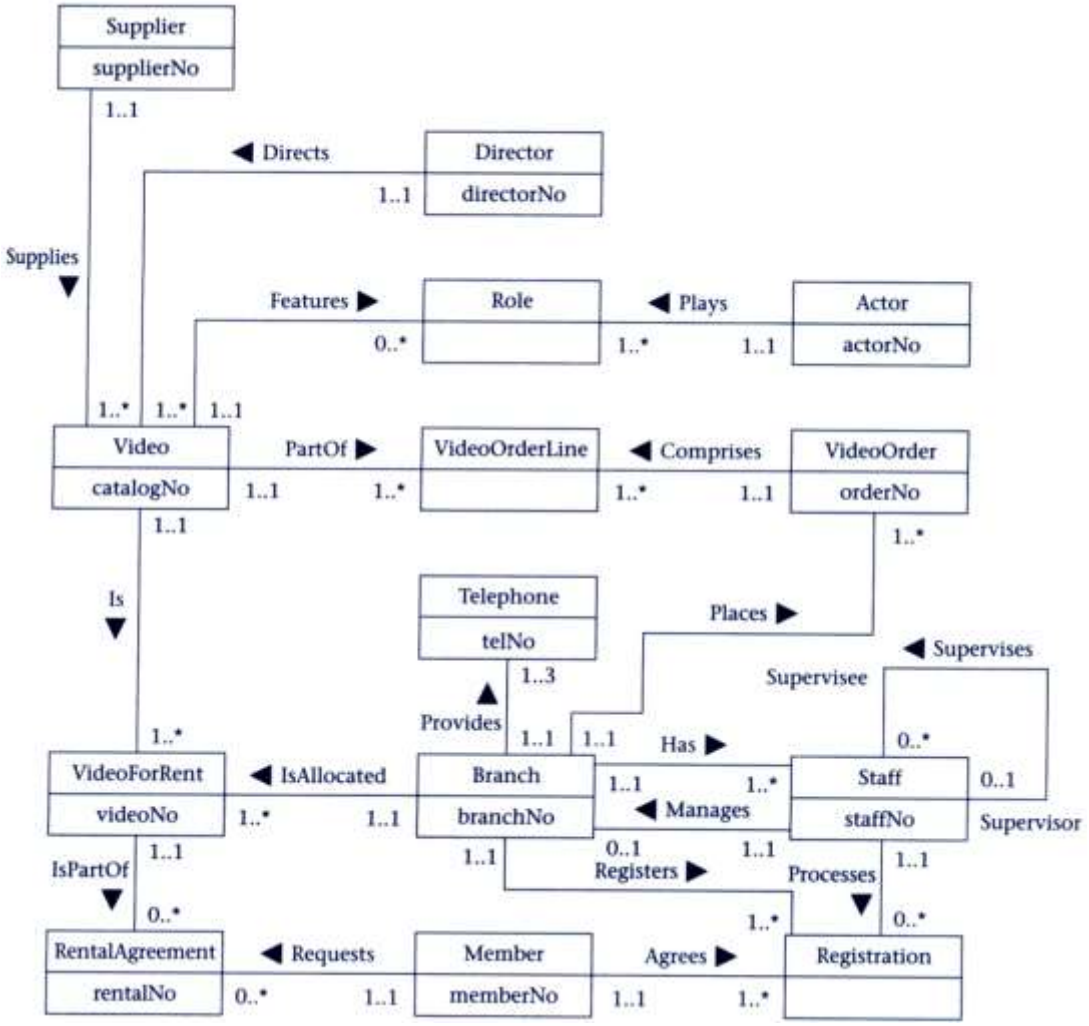
طی این مرحله، موجودیتها/ جدولها و رابطه ممکن است ادغام شوند، کلیدهای اصلی تغییر یابند، و رابطه های جدید شناسایی شوند. بررسی کنید که کلیدهای خارجی جدولهای فرزند هنوز صحیح باشند، و هر نوع تغییرات لازم را اعمال کنید.

محدودیتهای جامعیت را اعمال کنید

بررسی کنید که محدودیتهای جامعیت مدل داده منطقی سراسری با آنچه که در اصل برای هر دید کاربر مشخص شده ناسازگاری نداشته باشند.

شکل ۴-۱۰

مدل داده منطقی سراسری کاربرد پایگاه داده *StayHome* .



مدل داده منطقی محلی را ترسیم کنید

حال می توانید مدل داده سراسری را که نشان دهنده همه مدلهای داده محلی ادغام شده میباشد را ترسیم کنید. مدل ER سراسری پایگاه داده *StayHome* در شکل ۴-۱۰ نشان داده شده است.

مستندات را بروز کنید

مستندات را بروز کنید تا تغییراتی را که طی توسعه مدل داده سراسری اعمال کرده اید را منعکس کند. اهمیت دارد که مستندات بروز باشند و مدل داده جاری را منعکس کند. اگر تغییرات متعاقبا طی طراحی پایگاه داده و یا نگهداری آن به مدل اعمال میشود، بنابراین مستندات بایستی در زمان مشابه بروز شوند. اطلاعات قدیمی در آینده باعث ناسازگاری در آینده خواهد شد. جدولهایی که مدل داده منطقی سراسری پایگاه داده *StayHome* را نشان میدهد در شکل ۵-۱۰ نشان داده شده است.

شکل ۵-۱۰

ساختار جدول مدل داده منطقی سراسری *StayHome* .

Actor (actorNo, actorName) **Primary Key** actorNo	**Branch** (branchNo, street, city, state, zipCode, mgrStaffNo) **Primary Key** branchNo **Alternate Key** zipCode **Foreign Key** mgrStaffNo **references** Staff(staffNo)
Director (directorNo, directorName) **Primary Key** directorNo	**Member** (memberNo, fName, lName, address) **Primary Key** memberNo
Registration (branchNo, memberNo, staffNo, dateJoined) **Primary Key** branchNo, memberNo **Foreign Key** branchNo **references** Branch(branchNo) **Foreign Key** memberNo **references** Member(memberNo) **Foreign Key** staffNo **references** Staff(staffNo)	**RentalAgreement** (rentalNo, dateOut, dateReturn, memberNo, videoNo) **Primary Key** rentalNo **Alternate Key** memberNo, videoNo, dateOut **Foreign Key** memberNo **references** Member(memberNo) **Foreign Key** videoNo **references** Video(videoNo)
Role (catalogNo, actorNo, character) **Primary Key** catalogNo, actorNo **Foreign Key** catalogNo **references** Video(catalogNo) **Foreign Key** actorNo **references** Actor(actorNo)	**Staff** (staffNo, name, position, salary, branchNo, supervisorStaffNo) **Primary Key** staffNo **Foreign Key** branchNo **references** Branch(branchNo) **Foreign Key** supervisorStaffNo **references** Staff(staffNo)
Supplier (supplierNo, name, address, telNo, status) **Primary Key** supplierNo **Alternate Key** telNo	**Telephone** (telNo, branchNo) **Primary Key** telNo **Foreign Key** branchNo **references** Branch(branchNo)
Video (catalogNo, title, category, dailyRental, price, directorNo, supplierNo) **Primary Key** catalogNo **Foreign Key** directorNo **references** Director(directorNo) **Foreign Key** supplierNo **references** Supplier(supplierNo)	**VideoForRent**(videoNo, available, catalogNo, branchNo) **Primary Key** videoNo **Foreign Key** catalogNo **references** Video(catalogNo) **Foreign Key** branchNo **references** Branch(branchNo)
VideoOrder (orderNo, dateOrdered, dateReceived, branchNo) **Primary Key** orderNo **Foreign Key** branchNo **references** Branch(branchNo)	**VideoOrderLine** (orderNo, catalogNo, quantity) **Primary Key** orderNo, catalogNo **Foreign Key** orderNo **references** VideoOrder(orderNo) **Foreign Key** catalogNo **references** Video(catalogNo)

مرحله ۲-۳ مدل داده منطقی سراسری را بررسی کنید

هدف

جدولهای ایجاد شده از مدل داده منطقی سراسری را بررسی کنید که با استفاده از نرمال سازی ساخت یافته باشند و در صورت لزوم، تراکنشهای مورد نیاز را پشتیبانی کنند.

در این مرحله بررسی کنید که آیا با استفاده از نرمال سازی ساختار جدولهای ایجاد شده مدل داده سراسری ساخت یافته هستند و آیا این جدولها از تراکنشهای همه کاربران پشتیبانی می کنند. همانگونه که در مراحل ۲-۲ و ۲-۳ فصل قبلی این

بررسی را انجام دادید. با این وجود، نیاز دارید فقط قسمتهایی از مدل را که تحت فرایند ادغام قرار دارد را بررسی کنید. در سیستمهای بزرگ، این مرحله میزان بررسی دوباره موردنیاز در مراحل بعدی را کاهش خواهد داد.

مرحله ۳-۳ مدل را از جهت رشدهای آتی بررسی کنید

هدف

مشخص کردن اینکه آیا در آینده تغییرات مهم قابل پیش بینی وجود دارد که مدل داده منطقی سراسری باید با این تغییرات مطابقت پیدا کند.

مهم است که مدل داده منطقی سراسری بتواند به آسانی گسترش یابد. اگر مدل بتواند تنها نیازمندیهای کنونی را برآورده کند، در اینصورت مدت حیات مدل نسبتا کوتاه شده و ممکن است دوباره کاری قابل توجهی نیاز باشد تا با نیازمندیهای جدید مطابقت کند. بنابراین مدلی را توسعه دهید که *قابل توسعه* باشد، و توانائی پشتیبانی از نیازمندیهای جدید با حداقل تاثیر بر کاربران موجود را داشته باشد. البته، از آنجایی که شرکت ممکن است نداند در آینده چه چیزی را میخواهد انجام دهد، دستیابی به این میتواند کاری مشکل باشد. حتی اگر صورت پذیرد، ممکن است از لحاظ زمان و هزینه بعلت تطبیق با تسهیلات آینده گران باشد.

مرحله ۴-۳ مدل داده منطقی سراسری را با کاربران باز بینی کنید

هدف

اطمینان از اینکه مدل داده منطقی سراسری نمایش صحیحی از شرکت است.

مدل داده منطقی سراسری حالا بایستی کامل و صحیح باشد. مدل و مستنداتی که مدل را توصیف می کند بایستی به همراه کاربران مجددا بازبینی شود تا مطمئن شویم که مدل نمایش صحیحی از شرکت است.

شما حالا آماده اید طراحی منطقی را به طراحی فیزیکی برگردانید، که در مراحل ۴ تا ۸ متدولوژی ذکر شده است که ما طراحی فیزیکی را در فصلهای ۱۲ تا ۱۶ بحث می کنیم.

خلاصه فصل

- ✓ مرحله ۳ متدلوژی طراحی منطقی پایگاه داده اختیاری بوده و تنها وقتی که کاربرد پایگاه داده نسبتا پیچیده ای با چندین دید که با استفاده از روش **جامعیت دید** مدیریت میشود ایجاد می کنید .

فصل ۱۱

تکنیکهای پیشرفته مدل سازی

آنچه در این فصل خواهید آموخت :

- محدودیتهای مفاهیم ساده مدل سازی ER و نیازمندیها جهت مدل سازی هر چه پیشرفته تر برنامه های کاربردی با استفاده از مفاهیم مدل ساز ی بهبود یافته داده.
- مفاهیم اصلی مرتبط با مدل رابطه- موجودیت بهبودیافته (EER) بنام تخصص/ تعمیم.
- تکنیک نموداری جهت نمایش تخصص/ تعمیم در مدل EER .
- چگونه جدولهایی ایجاد کنیم که نشان دهنده تخصص/ تعمیم در مدل EER باشند.

مفاهیم اساسی مرتبط با مدلسازی ER در فصل ۵ ذکر شد، و از این مفاهیم جهت ساختن مدلهای ER متدلوژی طراحی منطقی پایگاه داده فصلهای ۸ تا ۱۰ استفاده کردیم. این مفاهیم معمولا جهت نمایش اکثر مدلهای داده ای همچون مدلهای سنتی، کاربردهای پایگاه داده مبتنی بر مجری[۱] کافی می باشد. با این وجود، برای بیشتر کاربرد پایگاه داده های پیچیده مفاهیم پایه ای ER میتواند محدود کننده باشد. این موضوع باعث می شود تا نیاز به توسعه مفاهیم مدلسازی 'معنایی' پیشرفته تر داشته باشیم. مدل اصلی ER به همراه مفاهیم معنایی اضافی، **مدل EER** [۲] نامیده می شود. در این فصل، یکی از مفیدترین مفاهیم مرتبط با مدل EER را که تخصص/ تعمیم نامیده می شود را شرح داده و طریقه استفاده از آن را نشان می دهیم.

متدلوژی طراحی پایگاه داده ارائه شده در این کتاب گزینه ای جهت استفاده از مفاهیم اضافی مدل EER در مرحله ۶-۱ را فراهم می کند. انتخاب استفاده از این گزینه به پیچیدگی شرکت (یا بخشی از شرکت) در دست مدل و اینکه آیا این مفاهیم مدلسازی پیشرفته تر ما را در فرایند طراحی پایگاه داده کمک میکند بستگی دارد.

۱-۱۱ تخصص/ تعمیم

مفهوم تخصص / تعمیم مرتبط با انواع بخصوصی از موجودیتهای با نام سوپرکلاس و زیرکلاس، و فرایند **وراثت صفت** می باشد. ما این بخش را با تعریفی از اینکه سوپرکلاس و زیرکلاس چیست و همچنین بررسی رابطه های سوپرکلاس/ زیرکلاس شروع می کنیم. و فرایند وراثت صفت و مقایسه فرایند تخصص با تعمیم را شرح می دهیم.

همچنین نشان می دهیم که چگونه تخصص/ تعمیم را به طور نموداری با استفاده از نمایش UML نشان دهیم.

---

[۱] Administrative-base

[۲] Enhanced Entity-Relationship

۱۱-۱-۱ سوپرکلاسها و زیرکلاسها

موجودیت عمومی سوپرکلاس نامیده می شود که شامل انواع بخصوصی از موجودیتها است که زیرکلاس نامیده می شود. برای مثال، موجودیتی که ممکن است چند زیر کلاس مجزا داشته باشد موجودیت Staff باشد. موجودیتهایی که عضوهای موجودیت Staff هستند شاید بنامهای Secretary،Manager ، و SalesPersonnel طبقه بندی شوند. بعبارت دیگر، موجودیت Staff سوپرکلاس زیرکلاسهای Secretary، Manager ، و SalesPersonnel است.

۱۱-۱-۲ رابطه های سوپرکلاس/ زیرکلاس

رابطه بین سوپرکلاس و هر یک از زیرکلاسهای آن یک به یک (۱:۱) است و رابطه سوپرکلاس/ زیرکلاس نامیده می شود. برای مثال، Staff / Manager تشکیل رابطه سوپرکلاس/ زیرکلاس را می دهد. هر عضو زیرکلاس همچنین عضوی از سوپرکلاس نیز هست اما نقش مجزایی دارد.

ما می توانیم از سوپرکلاسها و زیرکلاسها برای اجتناب از توصیف انواع متفاوتی از موجودیت با صفات ممکن مختلف در یک موجودیت واحد استفاده کنیم. برای مثال، SalesPersonnel ممکن است صفات بخصوصی از قبیل SalesArea و CarAllowance، و همچون اینها داشته باشد. اگر تمام صفتهای Staff و آنهایی که مربوط به کارهای بخصوصی هستند توسط موجودیت واحد Staff نشان دهیم، این عمل ممکن است باعث انبوهی از null ها برای صفتهای مربوط به شغل شود. واضح است که، Sales Personnel دارای صفات مشترک با دیگر پرسنل همچون staffNo، name ، position، و salary می باشد، اما زمانیکه سعی می کنیم تمام عضوهای پرسنل را با موجودیت واحد نمایش دهیم، صفات غیر مشترک است که باعث مشکل می شود. تعیین سوپرکلاسها / زیرکلاسها می تواند به ما اجازه دهد تا رابطه هایی را که به طورکلی مرتبط با زیرکلاس خاصی از پرسنل و نه با تمام آن است را نشان دهیم. برای مثال، Sales Personnel ممکن است رابطه مجزایی همچون SalesPersonnel *Requires* Car داشته باشد که مناسب تمام پرسنل نیست.

برای نشان دادن نقاط فوق، اجازه دهید جدولی بنام AllStaff شکل ۱-۱۱ را در نظر بگیریم. این جدول جزئیات تمام عضوهای پرسنل را نگه می دارد و اهمیتی ندارد چه شغلی را نگه می دارد. نتیجه نگه داشتن جزئیات همه اعضا پرسنل در یک جدول آن است که در حالیکه ستونهای مربوط به همه پرسنل (بنامهای staffNo ، name ، position ، salary ،و branchNo)پر شده است، فقط بعضی از آنهایی که تنها قابل اعمال به بعضی قواعد شغلی هستند پر شوند. برای مثال، ستونهای مربوط با زیر کلاس Sales Personnel (بنام salesArea ، vehLicenseNo ، و carAllowance) مقداری برای آن عضوهایی که در زیرکلاس Staff نیستند وجود ندارد.

> دو دلیل مهم برای معرفی مفاهیم سوپرکلاسها و زیرکلاسها در مدل ER وجود دارد. اولین دلیل اجتناب از توصیف بیش از یک بار مفاهیم مشابه می باشد، که باعث ذخیره وقت و خوانایی هر چه بیشتر مدل ER می گردد. دلیل دوم این است که باعث افزودن اطلاعات معنایی بیشتر به طراحی به شکلی که بیشتر افراد با آن آشنایی دارند میشود.
> برای مثال، اثباتهای ' Manager IS-A member of staff' و 'van IS-A type of vehicle' محتوای ارتباط معنایی به شکلی که بسادگی بتوان آن را ادامه داد ایجاد می کند.

شکل ۱-۱۱

جدول AllStaff که جزئیات همه اعضا پرسنل را نگه میدارد .

staffNo	name	position	salary	branchNo	salesArea	vehLicenseNo	carAllowance
S1500	Tom Daniels	Manager	46000	B001			
S0003	Sally Adams	Assistant	30000	B001			
S0010	Mary Martinez	Manager	50000	B002			
S0099	Joe Hope	Sales Personnel	35000	B002	WA 1A	SH22	5000
S3250	Robert Chin	Supervisor	32000	B002			
S2250	Sally Stern	Manager	48000	B004			
S2345	Linda Haven	Sales Personnel	37500	B002	WA 2B	SH34	5000
S0415	Art Peters	Manager	41000	B003			

۱۱-۱-۳ وراثت صفت

همانطور که بالا ذکر شد، رخداد موجودیت در زیرکلاس بیانگر شی مشابه'جهان واقعی' همچون سوپرکلاس است. بنابراین، اعضاء زیرکلاس صفات مرتبط با سوپرکلاس را به ارث میبرد، اما ممکن است صفات مختص به زیر کلاس را داشته باشد. برای مثال، عضو زیرکلاس Sales Personnel صفات مختص به زیرکلاس SalesArea, vehLicenseNo , و carAllowance و همه صفات سوپرکلاس Staff بنامهای staffNo، name ، position ، salary ، و branchNo را دارد.

زیرکلاس موجودیتی است که حق مخصوص به خود را دارد، بنابراین میتواند یک یا چند زیرکلاس داشته باشد. زیرکلاسی که بیش از یک سوپر کلاس دارد *زیرکلاس مشترک*[1] نامیده میشود. بعبارت دیگر، عضو زیرکلاس مشترک بایستی عضوی از سوپرکلاس مرتبط باشد. در نتیجه، صفات سوپرکلاس توسط زیرکلاس مشترک به ارث رسیده است، که ممکن است همچنین صفات مربوط به خود را نیز داشته باشد. این فرایند به نام *وراثت چندگانه* معروف است.

۱۱-۱-۴ فرایند تخصص

تخصص یک روش بالا به پایین جهت تعریف مجموعه سوپرکلاس ها و زیرکلاس های مربوطه است. مجموعه زیرکلاسها در پایه بعضی از خصوصیات متمایزکننده موجودیتها در سوپرکلاس تعریف شده است. وقتیکه ما زیرکلاس موجودیتی را شناسایی می کنیم، در این صورت صفتهای ویژه ای را (جایی که لازم باشد) به زیرکلاس آن مربوط می سازیم، و همچنین رابطه های دیگر بین زیرکلاس و دیگر موجودیتها یا زیرکلاسها (جایی که لازم باشد) را شناسایی می کنیم.

۱۱-۱-۵ فرایند تعمیم

فرایند تعمیم روش پایین به بالا است، که نتیجه شناسایی سوپرکلاس تعمیم شده از زیرکلاسهای اصلی است. فرایند تعمیم می تواند بعنوان معکوس فرایند تخصص بحساب آید. برای مثال، مدلی را در نظر بگیرید که در آن Manager ، Secretary ، و SalesPersonnel بصورت موجودیتهای مجزا نشان داده شده اند. اگر ما فرایند تعمیم را به این موجودیتها اعمال کنیم،

[1] Shared subclass

سعی میکنیم هر گونه مشابهتی را بین اینها اعم از صفات مشترک و رابطه ها شناسایی کنیم. همانطور که قبلا گفتیم، این موجودیتها صفات عمومی را به همه پرسنل در اشتراک دارند، و بنابراین میخواهیم Manager ،Secretary ، و SalesPersonnel را بصورت زیرکلاسهای سوپرکلاس تعمیم یافته پرسنل شناسایی کنیم.

نمایش بصورت نمودار

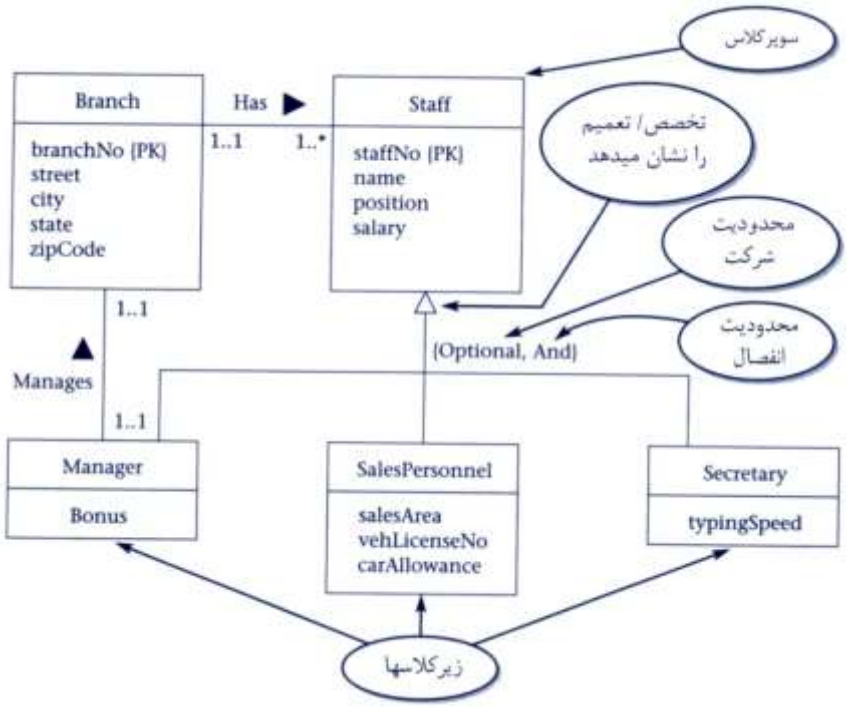
UML نماد گذاری ویژه ای جهت نمایش زیرکلاسها و سوپرکلاسها دارد. برای مثال، تخصص/ تعمیم را برای موجودیت پرسنل داخل زیرکلاسهایی که بیانگر وضایف شغلی می باشند در نظر بگیرید. سوپرکلاس پرسنل و زیرکلاسهای Manager ،Secretary ، و SalesPersonnel می توانند با نموداری در مدل EER مانند شکل ۲-۱۱ نشان داده شوند. توجه کنید که سوپرکلاس Staff و زیرکلاسهایش، که موجودیت هستند، با مستطیل نشان داده شده اند. زیرکلاسهای تخصص / تعمیم توسط خطوطی به مثلثی وصل شده اند که به پایین سوپرکلاس اشاره دارد. برچسب زیر مثلث، که به صورت { Optional , And} نشان داده شده اند محدودیتهای روی رابطه تخصص/ تعمیم را توصیف می کند. این محدودیتها با جزئیات در ادامه مبحث مطرح شده اند.

صفاتی که متعلق به زیرکلاس معینی هستند در بخش پایینی مستطیل زیرکلاس لیست شده اند. برای مثال، صفات salesArea، vehLicenseNo ، و carAllowance تنها با زیرکلاس SalesPersonnel مرتبط هستند، و قابل اعمال به زیرکلاسهای Manager یا Secretary نیستند. به طورمشابه، ما صفاتی را که مخصوص زیرکلاسهای Manager (bonus) و Secretary (typingSpeed) هستند را نشان می دهیم.

شکل ۲-۱۱ همچنین رابطه هایی را نشان می دهد که قابل اعمال به زیرکلاسهای خاصی یا تنها به سوپرکلاس می باشند. برای مثال، زیرکلاس Manager به موجودیت Branch از طریق رابطه Manages مربوط شده است، درحالیکه موجودیت Staff به موجودیت Branch از طریق رابطه Has مربوط شده است.

شکل ۲-۱۱

تخصص/ تعمیم موجودیت Staff داخل زیرکلاسهایی که وظایف شغلی را نشان می دهند.



توجه کنید که کثرت Manager در رابطه *Manages* ۱..۱ است، درحالیکه قبلا کثرت Staff در رابطه *Manges* ۱..۰ بود (بعبارت دیگر ، *Manager* شرکت اجباری دارد در حالیکه Staff شرکت اختیاری داشت).

در شکل ۳-۱۱، تخصص/ تعمیم Staff توسعه یافته است تا زیرکلاس مشترک با نام SalesManager را نشان دهد و زیرکلاس با نام Secretary با زیرکلاس خودش AssistantSecretary نامیده می شود. بعبارت دیگر، عضو زیرکلاس مشترک SalesManager بایستی عضو زیرکلاسهای SalesPersonnel و Manager و سوپرکلاس Staff باشد. در نتیجه، صفتهای زیرکلاسهای SalesPersonnel (salesArea,vehLicenseNo, carAllowance) و Manager (bonus) توسط زیرکلاس SalesManager ارث برده شده اند، که همچنین صفت اضافی خود بنام salesTarget را دارد. AssistantSecretary زیرکلاس Secretary است، که خود زیرکلاس Staff نیز می باشد. این بدین معنی است که عضو زیرکلاس AssitantSecretary بایستی عضو زیرکلاس Secretary و سوپر کلاس Staff باشد. درنتیجه، صفتهای سوپرکلاس Staff (staffNo, name, position, salary) و صفت زیرکلاس Secretary (typingSpeed) توسط زیرکلاس AssistantSecretary به ارث برده شده اند، که همچنین صفت اضافی خود بنام startDate را نیز دارد.

۱۱-۱-۶ محدودیتهای روی رابطه های سوپرکلاس/ زیرکلاس

دو محدودیتی که می تواند روی رابطه سوپرکلاس/ زیرکلاس اعمال شوند محدودیتهای شرکت[۱] و محدودیتهای انفصال[۲] نام دارند.

محدودیتهای شرکت

محدودیتهای شرکت ممکن است *اجباری* یا *اختیاری* باشند. رابطه زیرکلاس/ سوپرکلاس با شرکت *اجباری* مشخص میکند که هر رخداد موجودیت در سوپرکلاس بایستی همچنین عضو زیرکلاس باشد. برای نشان دادن شرکت اجباری، علامت ' اجباری' داخل کروشه مجعد زیر مثلث که بطرف سوپرکلاس اشاره می کند قرار می گیرد. برای مثال، در شکل ۴-۱۱ تخصص/ تعمیم Vehicle (Van, Bus , Car) شرکت اجباری دارد، که بدین معنی است که هر وسیله نقلیه بایستی van، bus، یا car باشد.

رابطه زیرکلاس/ سوپرکلاس با شرکت *اختیاری* مشخص می کند که عضو سوپرکلاس لازم نیست متعلق به زیرکلاسهایش باشد. برای نشان دادن شرکت اختیاری، علامت 'اختیاری' در داخل کروشه مجعد زیر مثلث قرار داده می شود که به سوپرکلاس اشاره می کند. برای مثال، در شکل ۲-۱۱ تخصص/ تعمیم وظیفه شغلی شرکت اختیاری دارد، که بدین معنی است که عضو پرسنل لازم نیست وظیف شغلی اضافی همچون Manager، Secretary ، یا Sales Personnel داشته باشد.

محدودیتهای انفصال

محدودیت انفصال تنها وقتی اعمال می شود که سوپرکلاس بیش از یک زیرکلاس داشته باشد. اگر زیرکلاسها گسسته باشند، در این صورت رخداد موجودیت می تواند عضو تنها یکی از زیرکلاسها باشد. برای نشان دادن رابطه زیرکلاس/

---

[۱] Participation constraint
[۲] Disjoint constraint

سوپرکلاس گسسته، علامت 'Or' بعد از محدودیت شرکت در کروشه مجعد قرار داده می شود. برای مثال، در شکل ۴-۱۱ زیرکلاسهای تخصص/ تعمیم Vehicle (Van, Bus , Car) گسسته هستند، که بدین معنی است که وسیله نقلیه van، bus، یا car است.

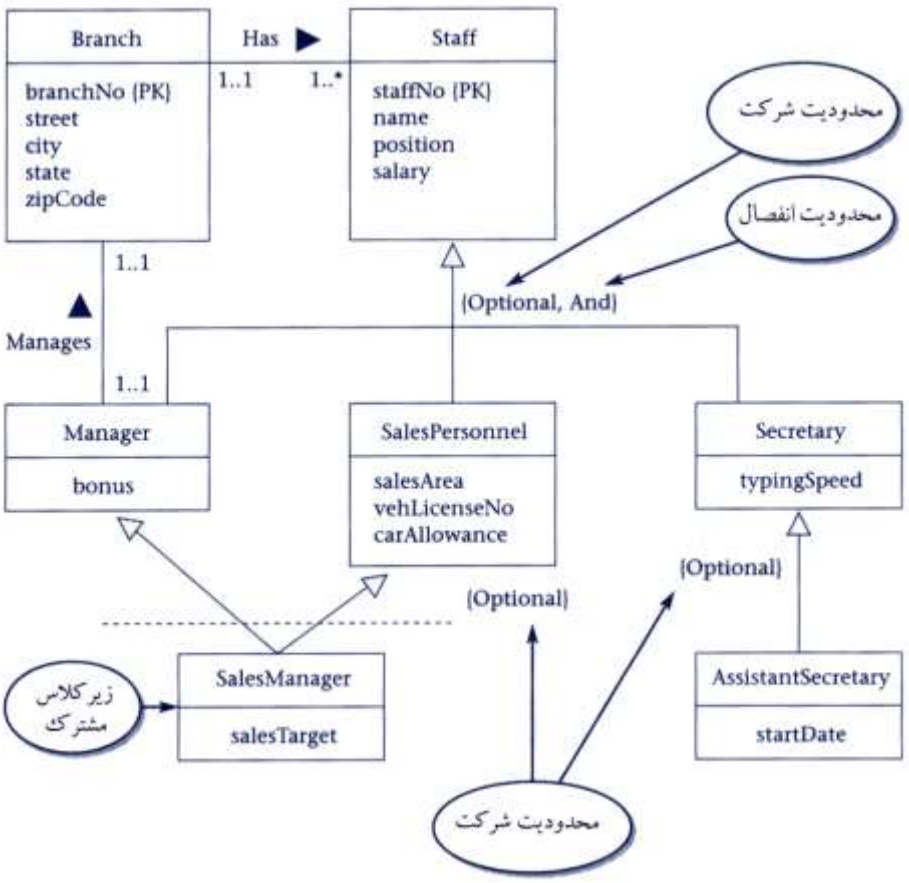
اگر زیرکلاس های تخصص/ تعمیم گسسته نباشند (غیرگسسته می گوییم)، دراینصورت رخداد موجودیت ممکن است عضو بیش از یک زیرکلاس باشد. برای نشان دادن رابطه تخصص/ تعمیم غیرگسسته، یک 'And' بعد از محدودیت شرکت در کروشه مجعد قرار می دهیم. برای مثال، در شکل ۲-۱۱ (و شکل ۳-۱۱) زیرکلاسهای وظیفه شغلی تخصص/ تعمیم (Manager, Secretary, SalesPersonnel) غیرگسسته هستند، که بدین معنی است که رخداد موجودیت میتواند عضو هر دو زیرکلاس Manager و SalesPersonnel باشد. این همچنین بوسیله وجود زیرکلاس مشترک بنام SalesManager نیز تصدیق می شود.

محدودیتهای شرکت و انفصال تخصص/ تعمیم مجزا بوده و به چهار دسته طبقه بندی می شوند :

اجباری و غیرگسسته بودن، اختیاری و غیرگسسته بودن، اجباری و گسسته بودن، و اختیاری و گسسته بودن.

۲-۱۱ ایجاد جدولهایی برای نشان دادن تخصص/تعمیم

در فصل ۹، شرح دادیم که چگونه جدولهایی از مدل داده ای که با استفاده از مفاهیم اساسی مدل ER ساخته شده باشد ایجاد کنیم. در این بخش، نشان می دهیم که چگونه جدولهایی برای سلسله مراتب تخصص/ تعمیم ایجاد کنیم. ما این فرایند را برای مدلهای EER شکلهای ۲-۱۱ و ۴-۱۱ نشان می دهیم. قبل از آن، هرجدول را با استفاده از زبان تعریف پایگاه داده (DBDL) برای پایگاه داده رابطه ای توضیح می دهیم.



شکل ۳-۱۱

تخصص/ تعمیم موجودیت Staff شامل زیرکلاس مشترک با نام SalesManager و زیرکلاس با نام Secretary با زیرکلاس مختص به خود با نام AssistantSecretary.

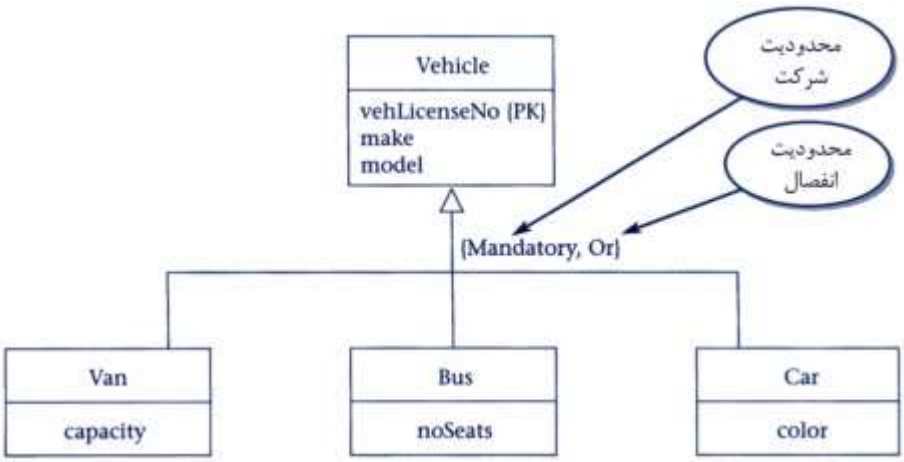
برای هر رابطه سوپرکلاس/ زیرکلاس در مدل EER، سوپرکلاس را بعنوان موجودیت والد و زیرکلاس را بعنوان فرزند مشخص کنید. چندین گزینه مختلف درباره اینکه چگونه چنین رابطه ای را بصورت یک یا چند جدول نشان دهید وجود دارد. انتخاب مناسبترین گزینه همانگونه که در جدول ۱-۱۱ نشان داده شده است بستگی به اشتراک و محدودیتهای انفصال موجود بر روی رابطه سوپرکلاس/ زیرکلاس دارد.

جدول ۱-۱۱ گزینه های موجود برای نشان دادن رابطه سوپرکلاس/ زیرکلاس بر اساس محدودیتهای اشتراک و انفصال.

محدودیت شرکت	محدودیت انفصال	جدولهای مورد نیاز
اجباری	غیرگسسته {And}	یک جدول
اختیاری	غیرگسسته {And}	دو جدول: یکی برای سوپرکلاس و دیگری برای همه زیرکلاسها
اجباری	گسسته {Or}	چندین جدول: یک جدول برای هر سوپرکلاس/ زیرکلاس ترکیب شده
اختیاری	گسسته {Or}	چندین جدول: یک جدول برای سوپرکلاس و دیگری برای هر زیرکلاس

شکل ۴-۱۱

تخصص/ تعمیم موجودیت Vehicle به انواع وسایل نقلیه .



ما از تخصص/ تعمیم Staff در شکل ۲-۱۱ بعنوان اولین مثال استفاده می کنیم. رابطه ای که در آن سوپرکلاس Staff با زیرکلاسهای (Manager ,SalesPersonnel , Secretary) دارد **اختیاری** است، اگر عضوی از پرسنل متعلق به زیرکلاسهای دیگری نباشد، و **غیرگسسته** است، اگر عضوی از پرسنل متعلق به بیش از یک زیرکلاس باشد. بر اساس گزینه های داده شده در جدول ۱-۱۱ ، شما باید رابطه سوپرکلاس/ زیرکلاس Staff را بوسیله ایجاد جدول برای سوپرکلاس و جدولی دیگر برای همه زیرکلاسهای آن ، همانند شکل ۵-۱۱ نشان دهید. برای وضوح هر چه بیشتر، همچنین جدولی برای نشان دادن موجودیت Branch و رابطه اش با Staff را نیز ایجاد کرده ایم.

شکل ۵-۱۱

جداولی که تخصص/ تعمیم Staff و موجودیت Branch شکل ۲-۱۱ را نشان می دهد.

Staff Superclass
staff (staffNo, name, position, salary, branchNo)
Primary Key staffNo
Foreign Key branchNo **references** Branch(branchNo)

Staff Subclasses
AllStaffSubclasses (subclassStaffNo, bonus, salesArea, vehLicenseNo, carAllowance, typingSpeed)
Primary Key subclassStaffNo
Foreign Key subclassStaffNo **references** Staff(staffNo)

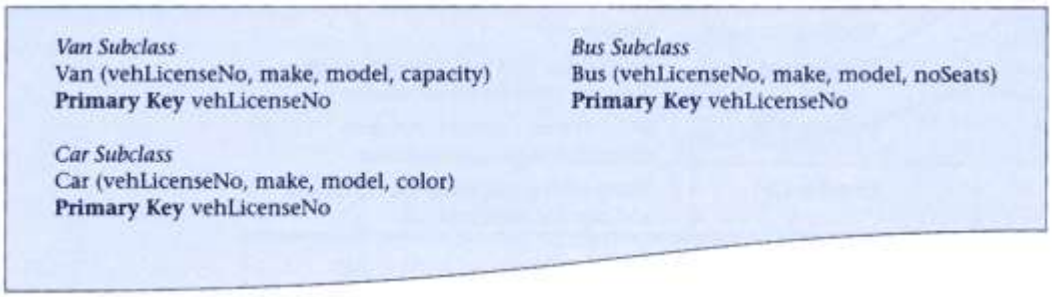
Branch
Branch (branchNo, street, city, state, zipCode, mgrStaffNo)
Primary Key branchNo
Foreign Key mgrStaffNo **references** AllStaffSubclasses(subclassStaffNo)

ما از تخصص/ تعمیم Vehicle در شکل ۴-۱۱ بعنوان مثال دوم استفاده میکنیم. رابطه ای که سوپرکلاس Vehicle با زیرکلاس هایش دارد (Van, Busیا ،Car) *اجباری* است اگر همه عضوهای سوپرکلاس Vehicle بایستی متعلق به یکی از زیرکلاسهایش باشد، و گسسته است اگر عضوی از سوپرکلاس Vehicle بتواند متعلق به تنها یک زیرکلاس باشد. بر اساس

گزینه های دیگر جدول ۱-۱۱ ، شما باید رابطه سوپرکلاس/ زیرکلاس Vehicle را با ایجاد جدولی برای هر سوپرکلاس/ زیرکلاس ترکیبی ، همانند شکل ۶-۱۱ نشان دهید.

شکل ۶-۱۱

جدولهایی برای نشان دادن تخصص/ تعمیم Vehicle نشان داده شده در شکل ۴-۱۱ .



اگر چه گزینه های توضیح داده شده در جدول ۱-۱۱ راهنمایی هایی برای چگونه بهتر نمایش دادن رابطه سوپرکلاس/ زیرکلاس فراهم می کند، چندین فاکتور دیگر نیز ممکن است انتخاب نهایی را تحت تاثیر قرار دهند :

- اینکه آیا زیرکلاسها در رابطه های مجزا درنظر گرفته می شوند;
- تعداد صفاتی که در هر زیرکلاس مجزا هستند;
- تعداد نسبی رخدادهای موجودیت ارائه شده توسط سوپرکلاس و توسط هر زیر کلاس.

خلاصه فصل

- ✓ **سوپرکلاس** موجودیتی است که رابطه ها و صفتهای مشترک برای همه رخدادها در موجودیت را نگه میدارد.
- ✓ **زیرکلاس** موجودیتی است که نقش مجزایی دارد و صفتها و رابطه های مشخصی برای بعضی رخدادها در موجودیت (سوپرکلاس) را نگه میدارد.
- ✓ **وراثت صفت** فرایندی است که عضوی از زیرکلاس ممکن است صفتهای مختص به زیرکلاس را داشته باشد، و صفتهای مربوط به آنها با زیرکلاس را به ارث ببرد.
- ✓ **تخصص** فرایند بزرگ کردن تفاوتهای بین عضوهای موجودیت بوسیله شناسایی خصیصه های مجزا کننده آنها میباشد.
- ✓ **تعمیم** فرایند به حداقل رساندن تفاوتها بین موجودیتها بوسیله شناسایی خصیصه های عمومیشان میباشد.
- ✓ محدودیتهایی که ممکن است به رابطه سوپرکلاس/زیرکلاس اعمال شود محدودیتهای انفصال و شرکت نامیده میشود.
- ✓ **محدودیت شرکت** مشخص میکند که آیا هر رخداد در سوپرکلاس باید بعنوان عضوی از زیرکلاس سهیم شود.
- ✓ **محدودیت انفصال** رابطه بین عضوهای زیرکلاسها را توصیف میکند و مشخص میکند که آیا برای عضوی از زیرکلاس ممکن است که عضوی از دیگری، یا بیش از یک زیرکلاس باشد .

بخش چهارم

طراحی فیزیکی پایگاه داده

فصل ۱۲

طراحی فیزیکی پایگاه داده – مرحله ۴

آنچه در این فصل خواهید آموخت :

- هدف از طراحی فیزیکی پایگاه داده.
- چگونه طراحی منطقی پایگاه داده را به طراحی فیزیکی پایگاه داده تبدیل کنیم.
- چگونه جدولهای پایه را در DBMS هدف طراحی کنیم.
- چگونه قواعد تجاری را در DBMS هدف طراحی کنیم.

در این فصل و چهار فصل بعدی، متدلوژی طراحی فیزیکی پایگاه داده را برای پایگاه داده های رابطه ای نشان داده و شرح خواهیم داد. نقطه شروع برای این فصل مدل داده منطقی سراسری و همچنین مستنداتی است که مدلی را که در مراحل ۱ تا ۳ متدلوژی ایجاد شده است را شرح می دهد. در مرحله ۱ متدلوژی با ایجاد مدلهای داده منطقی محلی شروع شده و سپس در مرحله ۲ از مدلهای منطقی جهت مشتق کردن مجموعه جدولها استفاده گردید. مدلهای منطقی و جدولهای مشتق شده بررسی شدند تا از خوب ساخته شدن جداول با استفاده از تکنیکهای نرمال سازی مطمئن شده باشیم، و همچنین اطمینان حاصل شود که تراکنشهای موردنیاز کاربر را پشتیبانی می کنند. مرحله طراحی منطقی پایگاه داده در مرحله ۳ با ترکیب مدلهای داده محلی (که هر دید شرکت را نشان می دهد) با همدیگر برای ایجاد مدل داده منطقی(که همه شرکت را نشان میدهد) به پایان رسید.

در دومین مرحله متدلوژی طراحی پایگاه داده، بنام **طراحی فیزیکی پایگاه داده**، شما باید تصمیم بگیرید که چگونه ساختار منطقی پایگاه داده (اعم از موجودیتها، صفتها، رابطه ها، و محدودیتها) را به طراحی فیزیکی پایگاه داده برگردانید به طوریکه بتواند با استفاده از DBMS هدف پیاده سازی شود. از آنجائیکه بیشتر قسمتهای طراحی فیزیکی پایگاه داده به شدت به DBMS هدف وابسته اند، ممکن است دریابید که بیش از یک راه برای پیاده سازی هر قسمت معینی از پایگاه داده وجود دارد. بنابراین برای انجام این کار، احتمالا نیاز دارید که از کارکرد DBMS هدف مطلع باشید، و همچنین مزایا و معایب پیشنهاد های دیگر را برای پیاده سازی خاصی درک کنید. همچنین برای بعضی سیستمها، احتمال دارد استراتژی ذخیره سازی مناسبی را انتخاب کنید. RDBMS های PC، همچون مایکروسافت اکسس، عموما ساختار ذخیره سازی ثابتی دارند و بنابراین، اگر از چنین سیستمهایی استفاده کنید، احتمالا درباره این مرحله نگرانی نخواهید داشت.

در این فصل، نشان می دهیم که چگونه جدولهای مشتق شده از مدل داده منطقی سراسری را به پیاده سازی پایگاه داده مشخصی تبدیل کنید. در فصل ۱۳، برای شما زمانیکه می خواهید سازمان فایلهای جدولهای پایه و ایندکسها را ایجاد کنید راهنمایی را ارائه خواهیم کرد. همچنین در فصل ۱۴، راهنمایی دیگری را وقتی می خواهید تصمیم بگیرید که مدل داده فیزیکی را دوباره نرمال سازی کرده و همچنین افزونگی را نشان دهید ارائه خواهیم کرد. در فصل ۱۵، خواهیم دید که چگونه می توانید مطمئن شوید که پایگاه داده امن است، و بالاخره در فصل ۱۶، درباره فرایند مداوم نظارت و بهبود سازی سیستم عملکردی بحث خواهیم کرد.

در جای خود، جزئیات پیاده سازی فیزیکی را برای وضوح هرچه بیشتر بحث نشان خواهیم داد. برای نشان دادن تفاوت میان DBMS ها، برای بررسی موردی *StayHome* که تا به اینجا در این کتاب درباره آن بحث کرده ایم، از مایکروسافت اکسس جهت نشان دادن موارد پیاده سازی استفاده خواهیم کرد. در مقابل، از DBMS اوراکل در دومین بررسی موردی که در فصلهای ۱۸ و ۱۹ ارایه کرده ایم استفاده می کنیم.

همانگونه که بیاد دارید قبلا متدولوژی ای را که برای طراحی منطقی پایگاه داده ارائه کرده ایم، در زیر به طور مختصر فرایند طراحی را مرور می کنیم.

۱۲-۱ مقایسه طراحی منطقی و فیزیکی پایگاه داده

طراحی منطقی پایگاه داده بسیار مستقل از جزئیات پیاده سازی می باشد، که میتوان به قابل استفاده بودن DBMS هدف، برنامه های کاربردی، زبانهای برنامه نویسی، یا هر نوع دیگری از ملاحظات فیزیکی اشاره کرد. خروجی این مرحله مدل داده منطقی سراسری و مستنداتی است که این مدل را توصیف می کند که از این مستندات میتوان به دیکشنری داده و مجموعه جدولهای رابطه ای اشاره کرد. مجموعه این دو، منابع اطلاعات مرحله طراحی فیزیکی را ارائه می کند، و برای شما ابزاری را جهت سبک سنگین کردن فراهم می آورند که برای طراحی موثر پایگاه داده اهمیت فراوان دارند.

در حالیکه طراحی منطقی پایگاه داده مربوط به *چه چیزی* است، طراحی فیزیکی پایگاه داده مربوط به *چگونگی* می باشد. مشخصا، بعنوان طراح فیزیکی پایگاه داده باید بدانید سیستم کامپیوتری چگونه فعالیتهای *DBMS* را میزبانی می کند، و کاملا باید از کارکرد *DBMS* هدف مطلع باشید. از آنجائیکه سیستمهای کنونی کارایی های متفاوتی دارند، طراحی فیزیکی بایستی درخور سیستم DBMS خاصی باشد. با این وجود، طراحی فیزیکی پایگاه داده فعالیت ایزوله شده ای نیست- بنابراین اغلب بازخوردی بین طراحی فیزیکی، منطقی، و کاربردی وجود دارد. برای مثال، تصمیمات اخذ شده هنگام طراحی فیزیکی برای بهبود کارایی، همچون ترکیب جدولها با همدیگر، ممکن است به مدل داده منطقی اثر کنند.

۱۲-۲ خلاصه ای از متدلوژی طراحی فیزیکی پایگاه داده

مراحل مربوط به طراحی فیزیکی پایگاه داده در شکل ۱۲-۱ نشان داده شده است. ما متدلوژی طراحی فیزیکی پایگاه داده را به چهار مرحله اصلی تقسیم کرده ایم، که از شماره ۴ به طور متوالی جهت سازگاری با سه مرحله متدلوژی طراحی منطقی پایگاه داده آغاز می شود. فصلی که مرحله در آن بحث خواهد شد در ستون مجاور نشان شده است.

مرحله ۴ طراحی فیزیکی پایگاه داده شامل طراحی جدولهای پایه و محدودیتهای جامعیت با استفاده از کارکرد در دسترس DBMS هدف می باشد.

مرحله ۵ شامل انتخاب سازمان فایلها و ایندکس ها برای جدولهای پایه است. به طور خاص، DBMS های کامپیوتر های شخصی ساختار ذخیره سازی ثابتی دارند اما دیگر DBMS ها میل دارند تا تعداد متعددی سازمان فایل برای داده در نظر بگیرند. از نقطه نظر کاربر، نمایش داخلی ذخیره سازی برای جدولها که بایستی کاربران را پنهان کند بایستی قادر باشد به جدولها و رکوردها بدون مشخص کردن اینکه کجا و چگونه رکوردها ذخیره شده اند دستیابی داشته باشد. بعنوان طراح فیزیکی پایگاه داده، باید جزئیات طراحی فیزیکی را هم به DBMS و هم به سیستم عامل فراهم کنید. برای DBMS، باید سازمان فایلهایی که برای نمایش هر جدول استفاده می شود را مشخص کنید; و برای سیستم عامل نیز، جزئیاتی همچون محل و امنیت هر فایل را مشخص کنید.

شکل ۱-۱۲

مراحل متدلوژی طراحی فیزیکی پایگاه داده .

	فصل
مرحله ۴ مدل داده منطقی سراسری را برای DBMS هدف ترجمه کنید	۱۲
مرحله ۱-۴ جدولهای پایه را برای DBMS هدف طراحی کنید	
مرحله۲-۴ قواعد تجاری DBMS هدف را طراحی کنید	
مرحله ۵ نمایش فیزیکی را طراحی کنید	۱۳
مرحله ۱-۵ تراکنشها را تحلیل کنید	
مرحله ۲-۵ سازمان فایلها را انتخاب کنید	
مرحله ۳-۵ ایندکسها را انتخاب کنید	
مرحله ۶ معرفی افزونگی کنترل شده را در نظر بگیرید	۱۴
مرحله ۱-۶ داده مشتق شده را در نظر بگیرید	
مرحله ۲-۶ ستونهای تکراری یا جدولهای ملحق شده بهم را در نظر بگیرید	
مرحله ۷ مکانیزمهای امنیت را طراحی کنید	۱۵
مرحله ۱-۷ دیدهای کاربر را طراحی کنید	
مرحله۲-۷ قواعد دسترسی را طراحی کنید	
مرحله ۸ سیستم عملکردی را نظارت و بازبینی کنید	۱۶

مرحله ۶ کم کردن محدودیتهای نرمالسازی تحمیل شده بر مدل داده منطقی جهت بهبود کلی کارایی سیستم را در نظر می گیرد. این مرحله ای است که باید تنها در صورت لزوم تقبل کنید.

مرحله ۷ شامل طراحی مقیاسهای امنیتی برای حفاظت از داده ها در برابر دستیابی غیرمجاز است. که در برگیرنده این تصمیم است که چگونه هر دید کاربر بایستی پیاده سازی شود، و چه کنترلهای دسترسی در جدولهای پایه مورد نیاز است.

مرحله ۸ فرایند مداوم نظارت بر سیستم عملکردی جهت شناسایی و برطرف کردن هرگونه مشکل کارایی ایجاد شده از طراحی، و همچنین پیاده سازی جدید یا تغییر نیازمندیها است.

ضمیمه ب خلاصه متدلوژی را برای کسانی که در حال حاضر با طراحی پایگاه داده آشنا بوده و تنها به خلاصه ای از مراحل اصلی نیاز دارند ارائه می کند. در بقیه این فصل، مرحله ۴ متدلوژی طراحی پایگاه داده را مورد بررسی قرار می دهیم. در این و چهار فصل بعدی، ارتباط نزدیک بین طراحی فیزیکی پایگاه داده و پیاده سازی را با شرح اینکه چگونه طراحی های دیگر بایستی پیاده سازی شوند را نشان می دهیم.

مرحله ۴ مدل داده منطقی سراسری را برای DBMS هدف ترجمه کنید

هدف

تولید پایگاه داده رابطه ای کاری ساده از مدل منطقی داده.

اولین فعالیت طراحی فیزیکی پایگاه داده شامل ترجمه جدولهایی است که از مدل داده منطقی سراسری مشتق کرده اید به صورتی که بتوانید در DBMS رابطه ای هدف پیاده سازی کنید. قسمت اول این مرحله مستلزم تلفیق اطلاعاتی است که هنگام طراحی منطقی پایگاه داده جمع آوری و در دیکشنری داده مستند کرده اید. قسمت دوم فرایند استفاده از این اطلاعات

برای تولید طراحی جداول پایه می باشد. این فرایند نیازمند دانش درونی کارکرد پیشنهاد شده توسط DBMS هدف است. برای مثال، شما نیاز خواهید داشت تا بدانید:

- چگونه جدولهای پایه را ایجاد کنید;
- آیا سیستم از تعریف کلیدهای اصلی، کلیدهای خارجی، و کلیدهای فرعی پشتیبانی می کند;
- آیا سیستم از تعریف داده موردنیاز پشتیبانی می کند (یعنی، آیا سیستم اجازه میدهد ستونها بصورت NOT NULL تعریف شوند);
- آیا سیستم تعریف دامنه ها را پشتیبانی می کند;
- آیا سیستم از قواعد جامعیت رابطه ای پشتیبانی می کند;
- آیا سیستم تعریف قواعد تجاری را پشتیبانی می کند;

دو فعالیت عمده مرحله ۴ بدین صورت است:

- مرحله ۱-۴ جدولهای پایه DBMS هدف را طراحی کنید;
- مرحله ۲-۴ قواعد تجاری DBMS هدف را طراحی کنید;

مرحله ۱-۴ جدولهای پایه DBMS هدف را طراحی کنید

هدف

تصمیم گیری درباره اینکه چگونه جدولهای پایه ای را که در مدل داده منطقی سراسری مشخص شده است را در DBMS هدف نشان دهیم.

برای شروع مرحله طراحی فیزیکی، ابتدا نیاز به تطبیق و یکسان سازی اطلاعات جدولهایی دارید که طی طراحی منطقی پایگاه داده تولید کرده اید. اطلاعات مورد نیاز می تواند از دیکشنری داده و تعریف جدولهایی که با استفاده از زبان طراحی پایگاه داده (DBDL) تعریف کرده اید به دست آید. برای هر جدولی که در مدل داده منطقی سراسری مشخص کرده اید، باید تعریفی از موارد زیر را داشته باشید:

- نام جدول;
- لیستی از ستون های ساده در کروشه;
- کلید اصلی، و جایی که مناسب است، کلیدهای فرعی و کلیدهای خارجی;
- محدودیتهای جامعیت ارجاع برای هر کلید اصلی مشخص شده.

از دیکشنری داده، شما همچنین باید برای هر ستون موارد زیر را در نظر داشته باشید:

- دامنه های آنها، شامل نوع داده، طول، و هر نوع محدودیت بر روی دامنه;
- مقدار انتخابی پیش فرض برای ستون;
- آیا ستون میتواند null ها را نگه دارد;
- آیا ستون مشتق شده است، و اگر چنین است، چگونه باید محاسبه شود.

شکل ۲-۱۲

طراحی فیزیکی جدول Branch با استفاده از DBDL توسعه یافته.

```
domain Branch_Numbers       fixed length character string length 4
domain Street_Names         variable length character string maximum length 30
domain City_Names           variable length character string maximum length 20
domain State_Codes          fixed length character string length 2
domain Zip_Codes            fixed length character string length 5
domain Staff_Numbers        fixed length character string length 5

branch( branchNo      Branch_Numbers      NOT NULL,
        street        Street_Names        NOT NULL,
        city          City_Names          NOT NULL,
        state         State_Names         NOT NULL,
        zipCode       Zip_Codes           NOT NULL,
        mgrStaffNo    Staff_Numbers       NOT NULL)
        Primary Key branchNo
        Alternate Key zipCode
        Foreign Key mgrStaffNo References Staff(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION
```

برای نشان دادن طراحی جدولهای پایه ما از شکل توسعه یافته DBDL برای تعریف دامنه ها، مقادیر پیش فرض، و نشانگر null استفاده می کنیم. برای مثال، برای جدول Branch کاربرد پایگاه داده *StayHome* که در شکل ۵-۱۰ تعریف شده، ممکن است طراحی را که در شکل ۲-۱۲ نشان داده شده است را ایجاد کنید.

پیاده سازی جداول پایه

مرحله بعد تصمیم گیری درباره این است که چگونه جدولهای پایه را پیاده سازی کنیم. همانطور که پیشترگفتیم، این تصمیم به DBMS هدف بستگی دارد; بعضی از سیستمها امکانات بیشتری را نسبت به دیگری جهت تعریف جدولهای پایه و محدودیتهای جامعیت فراهم می کنند. برای نشان دادن این مرحله، ما دو راه خاص برای ایجاد جدولها و محدودیتهای جامعیت با استفاده از موارد زیر نشان می دهیم:

(۱) SQL ۱۹۹۲ استاندارد (SQL2)

(۲) مایکروسافت اکسس ۹۷.

ابتدا در اینجا SOL2 را که بعنوان یک زبان رسمی برای DBMS های رابطه ای بشمار می رود و میتواند بعنوان خط مبنایی که بسیاری از سیستمها آن را قبول دارند دیده شود را مطرح می کنیم، اگرچه امروزه بیشتر سیستمها امکانات اضافی دیگری را نیز به آن افزوده اند.

SQL ۱۹۹۲ استاندارد (SQL2)

اگر DBMS هدف از SQL ISO استاندارد ۱۹۹۲ پشتیبانی کند، در این صورت طراحی پیاده سازی مبنا نسبتا آسان است. برای مثال، برای ایجاد جدول Branch از طراحی مذکور، بایستی از دستورات SQL2 نشان داده شده در شکل ۳-۱۲ استفاده کنید.

شکل ۳-۱۲

SQL2 جهت ایجاد جدول Branch.

```
CREATE DOMAIN Branch_Numbers AS CHAR(4)
CREATE DOMAIN Street_Names AS VARCHAR(30)
CREATE DOMAIN City_Names AS VARCHAR(20)
CREATE DOMAIN State_Codes AS CHAR(2)
CREATE DOMAIN Zip_Codes AS CHAR(5)
CREATE DOMAIN Staff_Numbers AS CHAR(5)
      CHECK(VALUE IN (SELECT staffNo FROM staff)

CREATE TABLE branch( branchNo                    Branch_Numbers     NOT NULL,
                     street                      Street_Names       NOT NULL,
                     city                        City_Names         NOT NULL,
                     state                       State_Names        NOT NULL,
                     zipCode                     Zip_Codes          NOT NULL UNIQUE,
                     mgrStaffNo                  Staff_Numbers      NOT NULL,
                     PRIMARY KEY (branchNo),
                     FOREIGN KEY (mgrStaffNo) REFERENCES Staff ON UPDATE CASCADE
                                                          ON DELETE NO ACTION
                   )
```

جدول دارای ستونهای مشابه و نوعهای دامنه مشخص شده در DBDL شکل ۲-۱۲ می باشد. کلید اصلی جدول شماره شعبه، branchNo، است که با استفاده از شرط PRIMARY KEY تعریف شده است. SQL2 به طور اتوماتیک یکتایی را به این ستون تحمیل می کند. کلید فرعی، zipCode، با استفاده از ترکیب موارد زیر تعریف شده است:

- محدودیت NOT NULL، برای اطمینان از اینکه مقدار همیشه موجود است، و
- محدودیت UNIQUE، برای اطمینان از اینکه هیچ دو رکوردی در جدول مقدار مشابه نداشته باشند.

یک کلید خارجی با استفاده از عبارت FOREIGN KEY با محدودیتهای جامعیت ارجاع مناسب مشخص شده است، که بصورت زیر هستند :

- قاعده بهنگام سازی (ON UPDATE CASCADE)، که این اطمینان را بوجود می آورد که هر وقت مقداری در ستون staffNo در جدول Staff بهنگام شود، در این صورت هر نوع مقدار(های) متناظر در ستون mgrStaffNo در جدول Branch به مقدار جدید ست شوند (یعنی، 'the update 'cascade');
- قاعده حذف (ON DELETE NO ACTION)، که از حذف شدن رکوردها از جدول Staff در صورتیکه عضو متناظر پرسنل مدیر شعبه باشد جلوگیری می کند.

مایکروسافت اکسس ۹۷

در بعضی سیستمها که با SQL2 استاندارد به طور کامل سازگار نیستند، هیچگونه پشتیبانی برای یک یا چندین عبارت PRIMARY KEY، FOREIGN KEY، UNIQUE، NOT NULL، DEFAULT، و CHECK وجود ندارد. به طور مشابه، بعضی سیستمها دامنه ها را نیز پشتیبانی نمی کنند. برای مثال، در مایکروسافت اکسس ۹۷ (و نسخه های بالاتر) می توانید موارد زیر را تعریف کنید:

- کلیدهای اصلی،
- مقادیر پیش فرض،
- ناتهی (not null) که فیلدهای required نامیده می شود. (توجه کنیدکه، مایکروسافت از عبارت فیلد به جای ستون استفاده می کند.)

با وجود اینکه، هیچ عبارت FOREIGN KEY مشخصی در داخل امکان ایجاد جدول وجود ندارد، اما همچنان که مختصرا خواهید دید این امکان وجود دارد که عبارات کلید خارجی را از طریق رابطه ها تعریف کنیم. بعلاوه، انواع داده ای کمی از SQL2 استاندارد تفاوت دارد، که در جدول ۱-۱۲ نشان داده شده است. حال نشان می دهیم که چگونه می توانیم جدول Branch با محدودیتهای گفته شده در بالا را در مایکروسافت اکسس ایجاد کنیم.

ایجاد جدول خالی در مایکروسافت اکسس

مایکروسافت اکسس عموما چهار راه مختلف برای ایجاد جدول خالی تدارک دیده است. شما می توانید:

- از ویزارد پایگاه داده در یک لحظه برای ایجاد همه جدولها، فرمها، و گزارشهایی که برای محتوای پایگاه داده مورد نیاز است استفاده کنید. ویزارد پایگاه داده پایگاه داده جدیدی ایجاد می کند، اگر چه برای اضافه کردن جداول جدید، فرمها، یا گزارشها به پایگاه داده فعلی استفاده نمی شود.
- از ویزارد جدول برای انتخاب فیلدهای جدولتان از جداول مختلف تعریف شده همچون تماسهای تجاری، کنترل موجودی خانه، یا رکوردهای پزشکی استفاده کنید.
- داده را مستقیما داخل جدول خالی وارد کنید (***Datasheet** نامیده می شود*). وقتی شما برگه داده جدیدی را ذخیره می کنید، اکسس داده شما را بررسی کرده و به طور اتوماتیک نوع داده مناسب و فرمتی را برای هر فیلد اختصاص خواهد داد.
- از Design view برای مشخص کردن تمام جزئیات جدول از چرکنویس استفاده کنید.

شکل ۴-۱۲ دید طراحی را جهت ایجاد جدول Branch نشان می دهد. صرف نظر از اینکه از کدام روش برای ایجاد جدول استفاده می کنید، شما می توانید از دید طراحی جدول در هر زمان برای سفارشی کردن هرچه بیشتر جدولتان همچون افزودن فیلدهای جدید، تنظیم مقادیر پیش فرض، یا ایجاد ماسکهای ورودی استفاده کنید.

ایجاد رابطه بین جدولها در اکسس

رابطه ها در پنجره Relationships ایجاد می شوند. برای ایجاد رابطه ها، شما باید ابتدا جدولهایی را که می خواهید بین آنها رابطه ایجاد کنید را نشان دهید، و سپس صفت کلید اصلی جدول والد را به صفت کلید خارجی جدول فرزند بکشید (drag کنید). در این صورت، اکسس پنجره ای را نشان خواهد داد که به شما اجازه می دهد محدودیتهای جامعیت ارجاع را مشخص کنید.

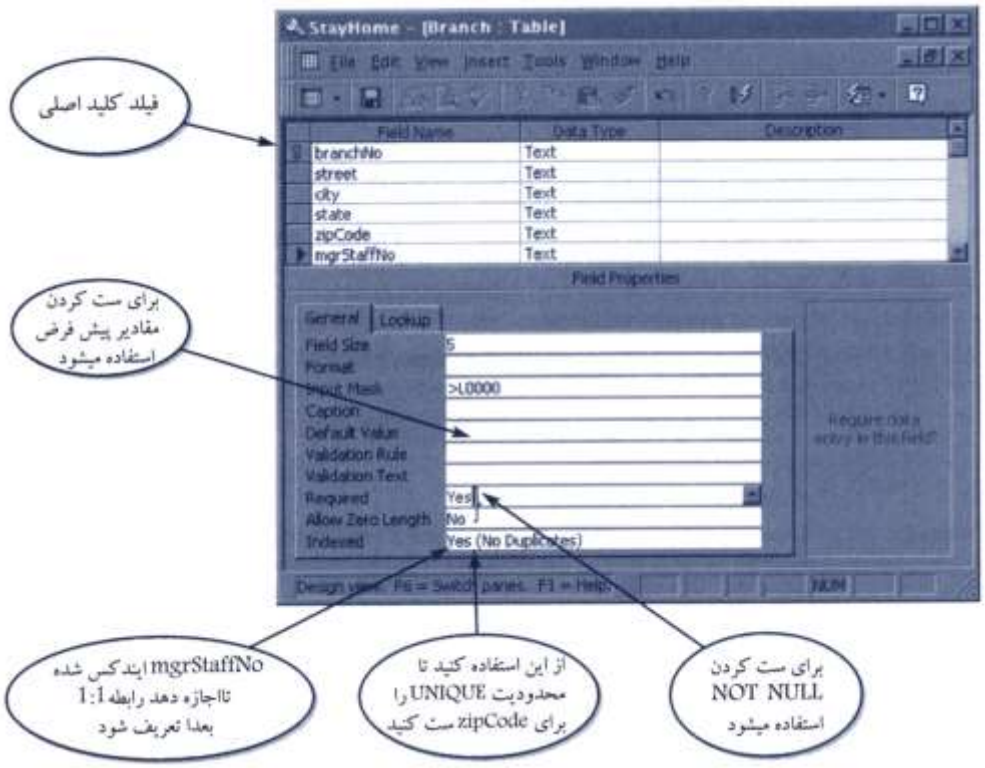
شکل ۵-۱۲(الف) کادر محاوره ای جامعیت ارجاعی را نشان می دهد زمانیکه رابطه یک به یک (۱:۱) Staff *Manages* Branch ایجاد شد نمایش داده می شود، و شکل ۵-۱۲(ب) پنجره Relationships را بعد از ایجاد رابطه نشان می دهد.

چندین مورد وجود دارد که باید درباره تنظیمات محدودیت جامعیت ارجاع مایکروسافت اکسس به آن توجه کنید:

(۱) رابطه یک به چند (*:۱) ایجاد می شود اگر فقط یکی از فیلدهای مربوطه کلید اصلی باشند یا شاخص یکتا داشته باشد; رابطه ۱:۱ زمانی ایجاد می شود که هر دو فیلد مربوطه کلید اصلی باشند یا شاخص های یکتا داشته باشند. بنابراین، برای اطمینان از اینکه رابطه *Manages* ۱:۱ است، باید نه تنها مطمئن شوید که فیلد staffNo جدول Staff به عنوان کلید اصلی ست شده است، در عین حال باید مطمئن شوید که فیلد mgrStaffNo در جدول Branch خاصیت شاخص بندی دارد که به Yes ست شده است (تکرار ندارد)، همچنانکه در شکل ۴-۱۲ نشان داده شده است.

شکل ۴-۱۲

دید طراحی که ایجاد جدول Branch را نشان میدهد.



(۲) تنها دو عمل جامعیت ارجاع برای بهنگام سازی و حذف وجود دارد که متناظر با NO ACTION و CASCADE است. بنابراین، اگر شما دیگر عملها را طی مرحله ۴-۲ تعریف محدودیتهای جامعیت مشخص کرده اید، بایستی در نظر بگیرید که آیا باید این محدودیتها را برای مناسب بودن در محدودیتهای در دسترس در اکسس اصلاح کنید، یا اینکه باید جستجو کنید که چگونه این محدودیتها را در کد برنامه کاربردی پیاده سازی کنید. مثالهایی از چگونگی پیاده سازی محدودیتهای جامعیت ارجاع را که مستقیما توسط DBMS هدف پشتیبانی نمی شوند را در فصل ۱۹ خواهید دید.

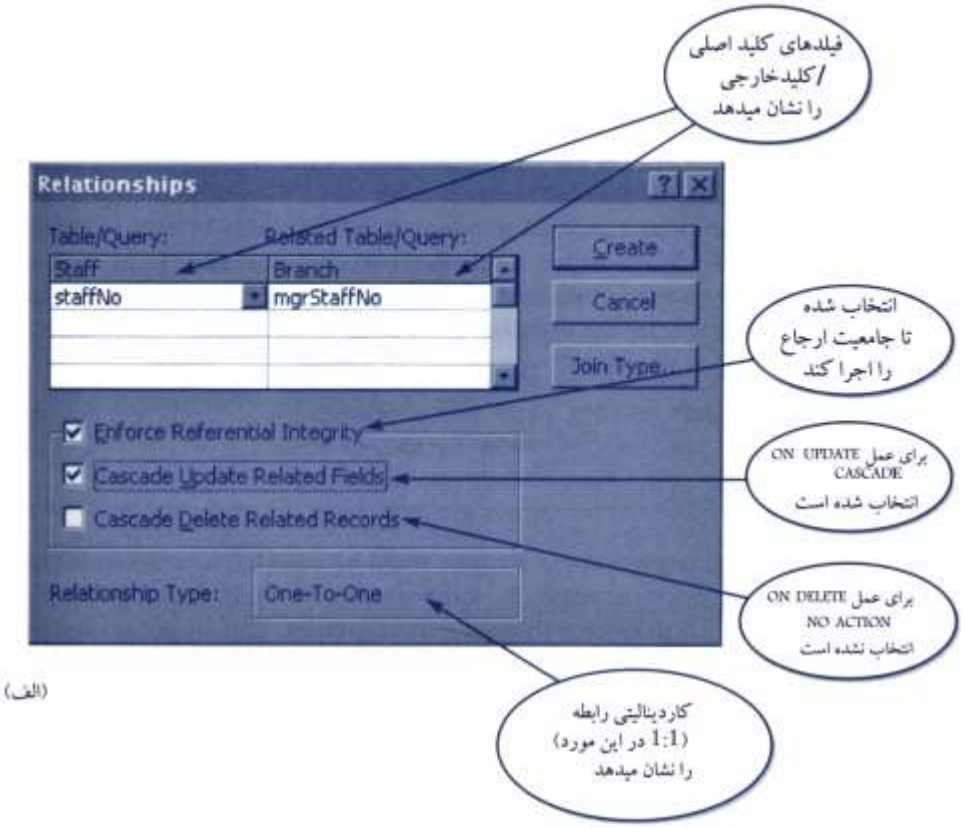
طراحی جداول پایه را مستند کنید

طراحی جدولهای پایه بایستی کاملا بهمراه دلایل انتخاب طراحی پیشنهادی مستند شوند. خصوصا، دلایل انتخاب یک روش را وقتیکه چندین روش دیگر موجود است را نیز مستند کنید.

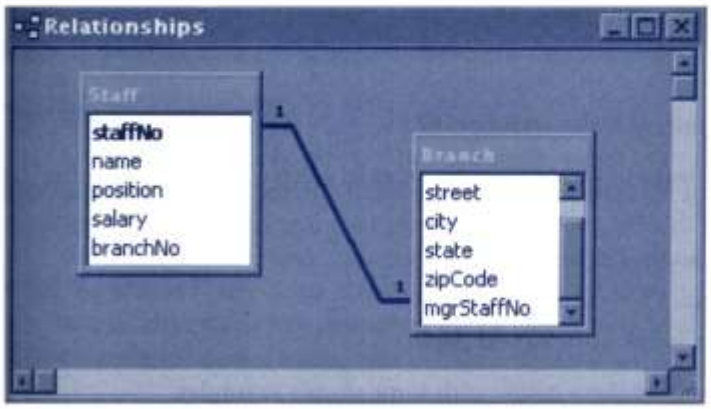
شکل ۵-۱۲

(الف) تنظیمات محدودیتهای جامعیت ارجاع برای رابطه ۱:۱ Staff *Manages* Branch .

(ب) پنجره Relationship با رابطه ۱:۱ نشان داده شده Staff *Manages* Branch .



(الف)



(ب)

مرحله ۲-۴ Business rules را برای DBMS هدف طراحی کنید.

هدف

طراحی قواعد تجاری برای DBMS هدف.

بهنگام سازی جدولها ممکن است توسط قواعد تجاری حاکم بر تراکنشهای 'جهان واقعی' که توسط عمل بهنگام سازی نشان داده شده اند محدود شوند. دوباره، طراحی چنین قواعدی وابسته بر انتخاب DBMS است; بعضی سیستمها امکانات بیشتری نسبت به دیگر سیستمها در تعریف قواعد تجاری فراهم می آورند. همانند مرحله قبل، اگر سیستم با SQL2 استاندارد سازگار باشد، ممکن است بعضی از قواعد پیاده سازی آسان باشد. برای مثال، *StayHome* قاعده ای دارد که از اجاره کردن بیش از ۱۰ فیلم در یک زمان ممانعت بعمل می آورد. شما می توانید این قاعده را در SQL2 با ایجاد دستور جدول برای جدول RentalAgreement ، با استفاده از عبارت زیرین طراحی کنید :

```
CONSTRAINT member_not_renting_too_many
  CHECK (NOT EXISTS   (SELECT memberno
                         FROM rentalagreement
                     GROUP BY memberno
                   HAVING COUNT (*) >= 10))
```

متناوبا، در بعضی سیستمها ممکن است از تریگر (*trigger*) برای تحمیل بعضی محدودیتها استفاده شود. برای مثال قبلی، در بعضی از سیستمها ما بایستی تریگر نشان داده شده در شکل ۶-۱۲ را برای تحمیل محدودیتهای جامعیت ایجاد کنیم. این تریگر قبل از اینکه رکورد داخل جدول RentalAgreement وارد شود و یا رکورد فعلی بهنگام شود فراخوانی می شود. اگر عضو در همان حال ۱۰ فیلم اجاره کند، سیستم پیامی را نشان می دهد و تراکنش را لغو می کند.

زیاد درباره جزئیات تریگر نگران نباشید. بحث تریگر را با جزئیاتش در مرحله ۲-۴ فصل ۱۹ بحث خواهیم کرد.

ایجاد قواعد تجاری در مایکروسافت اکسس

چندین روش جهت ایجاد قواعد تجاری در مایکروسافت اکسس استفاده می شود، برای مثال :

(۱) قواعد معتبرسازی برای فیلدها;
(۲) قواعد معتبرسازی برای رکوردها;
(۳) معتبرسازی برای فرمها با استفاده از بیسیک اکسس;

ما هر یک از اینها را در زیر بهمراه بعضی مثالهای ساده نشان می دهیم .

شکل ۶-۱۲

تریگر جهت اعمال این محدودیت که عضو نمی تواند بیش از ۱۰ فیلم در یک زمان اجاره کند.

```
CREATE TRIGGER member_not_renting_too_many
BEFORE INSERT OR UPDATE ON rentalagreemnet
                                    FOR EACH ROW
                                         DECLARE
                                       X  NUMBER;
                                          BEGIN
                                SELECT COUNT (*) INTO X
                                 FROM rentalagreement r
        WHERE r.memberno =: new. memberno;
                                              IF X >= 10 THEN
    Raise_application_error  (-20000,  ('Member'  ||  :new.memberno  ||
                                  'already renting 10 videos'));
                                              END IF;
                                                 END;
```

قواعد معتبرسازی برای فیلدها

شما می توانید با تعریف قاعده معتبرسازی فیلد مطمئن شوید که داده بصورت صحیح داخل فیلد وارد شده است. قاعده معتبرسازی فیلد برای بررسی مقدار وارد شده داخل فیلد زمانیکه کاربر فیلد را رها می سازد استفاده می شود. پیامی که شما تعریف می کنید در صورتیکه مقدار، قاعده معتبرساز ی را بشکند نشان داده می شود.

برای مثال، *StayHome* محدودیت ساده ای دارد که همه تاریخها برای فیلم اجاره ای نمی تواند زودتر از روز کنونی باشد، اگر چه تاریخ در ابتدا نامشخص باشد. شما می توانید این محدودیت را در سطح فیلد در جدول RentalAgreement با استفاده از تابع ()Date، که تاریخ کنونی را برمیگرداند، و در شکل ۷-۱۲ نشان داده شده است پیاده سازی کنید.

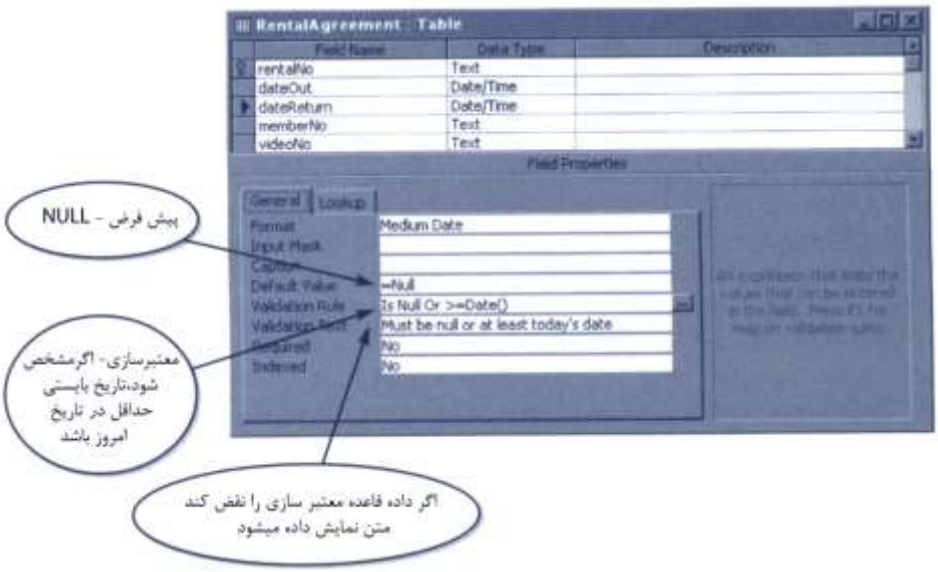
قواعد معتبرسازی برای رکوردها

قاعده معتبرسازی رکورد هنگام ذخیره داخلی رکورد کنترل می شود. برخلاف قواعد معتبرسازی فیلد، قواعد معتبرسازی رکورد می تواند به دیگر فیلدها رجوع کند. این عمل زمانی وقتیکه شما می خواهید مقادیری را از فیلدهای مختلف در جدول مقایسه کنید مفید می شود. برای مثال، *StayHome* ممکن است محدودیتی به این صورت داشته باشد که حداکثر زمان اجاره فیلم پنج روز باشد، اگر چه تاریخ ممکن است در ابندا نامشخص رها شود. شما می توانید این محدودیت را در سطح رکورد در جدول RentalAgreement با استفاده از قاعده معتبرسازی زیر پیاده سازی کنید:

```
[dateReturn]  IS NULL OR [DateReturn] <= [dateOut] + 5
```

شکل ۷-۱۲

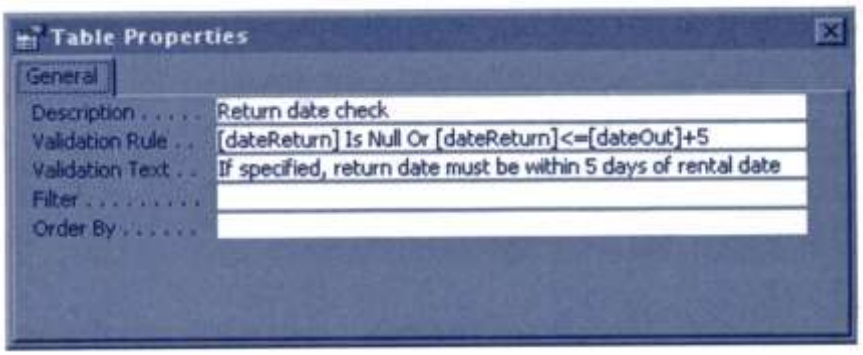
مثال معتبرسازی فیلد در مایکروسافت اکسس.



شکل ۸-۱۲ جعبه خاصیت قاعده معتبرسازی برای جدول داخل این مجموعه قاعده نشان می دهد.

شکل ۸-۱۲

نمونه معتبرسازی رکورد در مایکروسافت اکسس.



معتبرسازی فرمها با استفاده از بیسیک اکسس

همانطور که قبلا گفتیم، *StayHome* محدودیتی دارد که عضوها در یک زمان نمی توانند بیش از ۱۰ فیلم اجاره کنند. این یک محدودیت نسبتا پیچیده است، که نیاز دارد تا بررسی کنید که هر عضو در حال حاضر چند فیلم اجاره کرده است. یک راه برای پیاده سازی این محدودیت در اکسس استفاده از رویه رویداد[۱] (BeforeUpdate) است، که در شکل ۹-۱۲ نشان داده شده است. رویداد BeforeUpdate قبل از اینکه رکورد بهنگام شود فراخوانی می شود، و شما می توانید کدی را با این رویداد در فرم مرتبط سازید.

[۱] Event procedure

شکل ۹-۱۲

کد بیسیک اکسس برای بررسی اینکه عضو در حال حاضر بیش از ۱۰ فیلم اجاره نکرده باشد.

```
Private Sub Form_BeforeUpdate (Cancel As Integer)
Dim MyDB As Database
Dim MySet As RecordSet
Dim MyQuery As String


'Set up query to select all records for specified member'
MyQuery ="SELECT rentalno FROM rentalagreement WHERE memberno='"+memberNoField+"'"


'Open the database and run the query'
Set MyDB = DBEngine.Workspaces (0).Databases (0)
Set MySet = MyDB.OpenRecordset (MyQuery)

'Check if any records have been returned, then move to the end of the file to' 'allow
RecordCount property to be correctly set'
 If (NOT MySet.EOF) Then
         MySet.MoveLast
         If (MySet.RecordCount >= 10) Then 'If currently 10 - cannot rent any more'
               MsgBox "Member currently has 10 videos out"
               Me.Undo
         End If
End If

MySet.Close
MyDB.Close
End Sub
```

در بعضی از سیستمها، برای بعضی یا همه قواعد تجاری پشتیبانی وجود ندارد و لازم خواهد بود تا قواعد را در داخل برنامه کاربردی طراحی کنید، همانطور که در مثال گذشته نشان دادیم، که محدودیتها را در داخل کد ویژوال بیسیک قرار دادیم. پیاده سازی قاعده تجاری در کد برنامه در صورتیکه قواعد در جای خود پیاده سازی نشود، میتواند خطرناک باشد و منجر به مضاعف شدن تلاش، یا بدتر اینکه، باعث ناسازگاری پایگاه داده شود.

طراحی قواعد تجاری را مستند کنید

طراحی قواعد تجاری بایستی کاملا مستند شود. خصوصا، دلایل انتخاب یک روش جائیکه چندین روش وجود دارد را مستند کنید.

خلاصه فصل

- ✓ **طراحی فیزیکی پایگاه داده** فرایند تولید توصیف پیاده سازی پایگاه داده در حافظه جانبی است. که جدولهای پایه، سازمان فایلها و اندیسهای استفاده شده برای دسترسی به این داده ها به طور موثر، و محدودیتهای جامعیت مرتبط دیگر و محدودیتهای امنیت را توصیف می کند. طراحی جدولهای پایه را میتواند تنها یک مرتبه وقتیکه کاملا از امکانات پیشنهادی DBMS هدف آگاهی داشتید برعهده بگیرید.
- ✓ در مرحله مقدماتی (مرحله ۴) طراحی فیزیکی پایگاه داده، مدل داده منطقی سراسری را به فرمهایی که قابل پیاده سازی در DBMS هدف باشد برمی گردانید.
- ✓ در مرحله بعدی (مرحله ۵)، شما سازمان فایلها و شاخصهایی را که برای ذخیره سازی جدولهای پایه مورد استفاده قرار می دهید را طراحی می کنید. این شامل تحلیل تراکنش هایی خواهد بود که در پایگاه داده اجرا خواهد شد، و انتخاب سازمانهای فایل مناسب براساس آن تحلیل، و افزودن شاخصها خواهد بود.
- ✓ در مرحله ۶، شما معرفی افزونگی کنترل شده جهت بهبود دادن کارایی را در نظر می گیرید.
- ✓ پایگاه داده منبع ضروری شرکت را ارائه می کند، و بنابراین امنیت این منبع بینهایت مهم است. هدف مرحله ۷ طراحی این است که چگونه امنیت مشخص شده طی طراحی پایگاه داده منطقی به واقعیت تبدیل خواهد شد. و ممکن است شامل ایجاد دیدهای کاربر و استفاده از مکانیزمهای کنترل دستیابی، آنچه که SQL فراهم آورده می باشد.
- ✓ مرحله نهایی (مرحله ۸) فرایند مداوم بازبینی و بهبود سیستم عملکردی برای رسیدن به حداکثر کارایی می باشد.

فصل ۱۳

طراحی فیزیکی پایگاه داده – مرحله ۵

آنچه در این فصل خواهید آموخت :

- چگونه منابع سیستم کارائی را تحت تاثیر قرار می دهد.
- چگونه تراکنشهای کاربر را تحلیل کنیم تا خصوصیتهایی را که کارائی را تحت تاثیر قرار می دهند مشخص کنیم.
- چگونه سازمان فایل مناسب را بر مبنای تحلیلهای تراکنشها انتخاب کنیم.
- چه موقع شاخصها را برای بهبود کارائی انتخاب کنیم.

این فصل دومین مرحله از مرحله متدلوژی طراحی فیزیکی پایگاه داده مان را می پوشاند. در مرحله قبل، نشان دادیم چگونه طراحی منطقی را به مجموعه جدولهای پایه تبدیل کنیم. با این وجود، حتی برای پایگاه داده های ساده تر نیز، ملاحظات اضافی دیگری برای رسیدن به کارائی قابل قبول نیازمند است. در این فصل، مرحله بعد طراحی فیزیکی پایگاه داده، که جنبه های کارایی را که می توانید توسط انتخاب مناسب سازمان فایلها و شاخصها تحت تاثیر قرار دهید را در نظر می گیریم.

همانند طراحی منطقی، طراحی فیزیکی نیز باید توسط طبیعت داده و قصد استفاده آن هدایت شود. به طور مشخص، شما باید حجم کاری نوعی را که پایگاه داده باید پشتیبانی کند را درک کنید. طی مرحله تحلیل، ممکن است بفهمید که بعضی از کاربران خواسته هایی را درباره اینکه چگونه تراکنشهای مشخصی به طور سریع باید اجرا شده یا چند تراکنش بایستی در هر ثانیه پردازش شود را مطرح می کنند. این اطلاعات اساس تعداد تصمیماتی که باید طی این مرحله بگیرید را تشکیل می دهد.

همچنانکه قبلا گفتیم، برای متقبل شدن طراحی فیزیکی پایگاه داده، شما باید نحوه کارکرد DBMS هدف، از جمله سازمان فایلها، شاخص گذاری، و تکنیکهای پردازش پرس و جویی را که باید پشتیبانی کند را درک کنید. برای مثال، احتمالا شرایطی وجود دارد که DBMS زمانیکه شاخص دیگری وجود دارد نباید از شاخص بندی دومی استفاده کند. بنابراین، افزودن شاخص کارائی پرس وجو را بهبود نخواهد داد، و نتیجه کلی نامشخص خواهد بود.

از بخش ۴-۲ بیاد دارید که QBE و SQL زبانهای دستکاری داده غیر رویه ای (DMLs) می باشند. چنین زبانهایی جزئیات سطح پایین چگونگی دستیابی به داده در حافظه جانبی را پنهان می سازند. این برعهده DBMS است، یا دقیقتر، برعهده بهینه ساز پرس و جوی (*query optimizer*) DBMS است. به طور نمونه، بهینه ساز پرس وجو تعداد استراتژی های مختلفی را برای انجام دادن درخواست کاربر تحلیل خواهد کرد و یکی را که باور دارد کارایی بهینه را خواهد داد انتخاب خواهد کرد. این تحلیل بر اساس تخمینی از هزینه عملیات پایگاه داده با استفاده از آمار و ارقام پایگاه داده، همچون تعداد رکوردهای جدول، اندازه هر رکورد، و وجود شاخص خواهد بود.

این مطلب ممکن است بدین معنی باشد که شما هیچ تاثیری را بر روی استراتژی نهایی که DBMS انتخاب خواهد کرد ندارید. درحقیقت، شما خواهید دید که می توانید بعضی از ساختارهای ذخیره سازی را که در دسترس بهینه ساز پرس و جو می باشد را جهت انتخاب استراتژی بهینه تعریف کنید.

قبل از اینکه به بحث در مورد فعالیتهای این مرحله بپردازیم، اجازه دهید مختصرا بررسی کنیم که چگونه اجزاء مختلف سیستم با هم فعل و انفعال داشته و بر کارایی DBMS اثر می کنند.

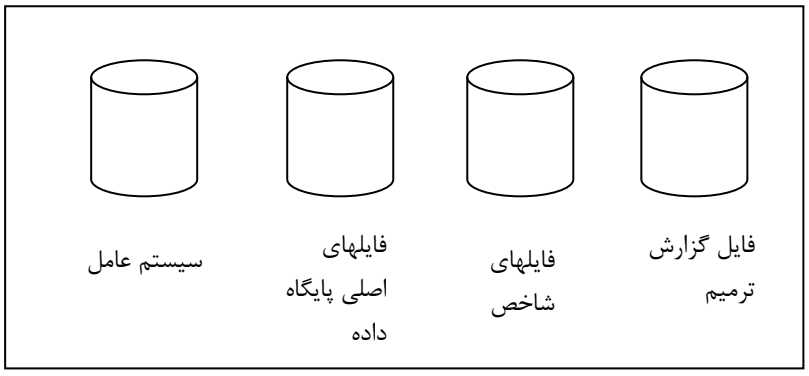
۱-۱۳ درک کردن منابع سیستم

برای بهبود کارائی، شما باید از چگونگی تعامل چهار جزء اصلی سخت افزاری با یکدیگر و اینکه چگونه بر کارائی سیستم اثر می گذارند آگاهی داشته باشید:

- **حافظه اصلی** دسترسی بر حافظه اصلی به طور عمده سریعتر از دسترسی بر حافظه جانبی است، بعضی اوقات دهها و یا حتی صدها هزار برابر سریعتر است. به طورکلی، حافظه اصلی هر چه بیشتر در دسترس DBMS و برنامه های کاربردی پایگاه داده باشد، باعث سریعتر شدن اجرای برنامه های کاربردی خواهد شد. با این وجود، همیشه داشتن حداقل ۵ درصد حافظه ای که در دسترس باشد محسوس می باشد. معادل آن، توصیه می شود که بیش از ۱۰ درصد حافظه در دسترس نباشد، که در غیر اینصورت حافظه اصلی به طور بهینه استفاده نمی شود. زمانیکه حافظه برای جا دادن همه فرایندها کافی نباشد، سیستم عامل جهت آزاد کردن حافظه صفحات فرایندها را بر روی دیسک می برد. چنانکه به یکی از این صفحات بعدا نیاز شد، سیستم عامل آن صفحه را از دیسک بر می گرداند. بعضی اوقات، لازم است برای آزادسازی حافظه تمام فرایندها از حافظه اصلی به دیسک و بر عکس منتقل شوند. در این صورت مشکلات هنگام صفحه بندی بیش از اندازه رخ می دهد.
- **CPU** CPU کارهای دیگر منابع سیستم را کنترل کرده و فرایندهای کاربر را اجرا می کند. هدف اصلی این جزء این است که از درگیری فرایندهایی که منتظر تخصیص CPU هستند ممانعت بعمل آورد. گلوگاه های CPU زمانی رخ می دهد که یا سیستم عامل و یا برنامه کاربردی تقاضاهای زیادی را به CPU داده اند. که این اغلب نتیجه صفحه بندی بیش از اندازه است.
- **Disk I/O** با هر DBMS بزرگی که باشد، مقدار عمده ای I/O دیسک صرف ذخیره و بازیابی داده ها می شود. روشی که داده ها بر روی دیسک سازماندهی می شود می تواند اثر بزرگی بر کارایی دیسک داشته باشد. پیشنهاد می شود حافظه جانبی بایستی به طور یکنواخت در سراسر درایوهای موجود توزیع شود تا احتمال مشکلات کارایی رخ داده تا حد امکان کم شود. شکل ۱-۱۳ مفاهیم اساسی توزیع داده بر روی دیسک را نشان می دهد :

- فایلهای سیستم عامل باید از فایلهای پایگاه داده جدا باشند.
- فایلهای اصلی پایگاه داده بایستی از فایلهای شاخص جدا شده باشند.
- فایلهای ثبت کننده گزارش ترمیم[1]، اگر وجود دارد و استفاده می شود، بایستی از خود پایگاه داده جدا باشد.

- شبکه وقتی مقدار داده هایی که در حال انتقال در شبکه هستند بزرگ باشد، گلوگاه های شبکه رخ می دهد.



شکل ۱-۱۳
پیکربندی معمول دیسک.

[1] RECOVERY LOG FILE

هر یک از این منابع ممکن است بر روی دیگر منابع سیستم اثر بگذارند. به طور مشابه، هرگونه بهبودی در یک منبع ممکن است بر بهبود کارایی دیگر منابع سیستم اثر داشته باشند. برای مثال:

- افزودن حافظه اصلی اضافی می تواند باعث کاهش صفحه بندی شود. که میتواند از رخ دادن گلوگاه های CPU جلوگیری کند.
- استفاده موثر از حافظه اصلی ممکن است منجر به کاهش I/O دیسک شود.

با داشتن این اهداف در ذهن، حال اجازه دهید فعالیتهای مرحله ۵ را بحث کنیم.

۱۳-۲ مرحله ۵ نمایش فیزیکی را طراحی کنید

هدف

مشخص کردن سازمانهای فایل بهینه برای ذخیره کردن جدولهای پایه، و شاخصها که جهت دستیابی به کارایی مورد قبول مورد نیاز می باشند.

ما در ضمیمه ج مقدمه مختصری از سازمانهای فایل و شاخصها را برای خوانندگانی که با این مفاهیم آشنا نیستند فراهم کرده ایم. جهت مرور، **سازمان فایل** روش مرتب کردن رکوردها در فایل بهنگام ذخیره فایل روی دیسک است; **شاخص** به DBMS اجازه می دهد تا رکورد مشخصی را در فایل به صورت سریع پیدا کنیم، و بدین وسیله پاسخ به پرس وجوهای کاربران سریع می شود.

انواع سازمان فایل موجود وابسته به DBMS هدف می باشد; بعضی از سیستمها نسبت به دیگری گزینه های بیشتری از سازمان فایل را فراهم می کنند. برای انجام طراحی فیزیکی پایگاه داده شما بایدکاملا ساختارهای موجود را شناخته، و اینکه چگونه سیستم هدف از این ساختار استفاده می کند را بخوبی درک کنید.

همچنین شما نمی توانید تصمیمهای معنی دار طراحی فیزیکی پایگاه داده را بگیرید مگر اینکه با جزئیات از تراکنشهایی که باید پشتیبانی شوند آگاه باشید. در تحلیل تراکنشها، سعی خواهید کرد تا معیارهای کارایی را مشخص کنید، که از آن جمله میتوان به موارد زیر اشاره کرد:

- تراکنشهایی که مکررا اجرا می شوند و اثر عمده ای بر کارایی خواهند داشت;
- تراکنشهایی که در عملکرد حرفه حیاتی هستند;
- زمانهایی بر اساس روز/ هفته زمانیکه تقاضاهای زیادی بر پایگاه داده اعمال می شود. (بحبوحه مصرف میگویند (PEAK LOAD)) .

شما از این اطلاعات استفاده خواهید کرد تا قسمتهایی از پایگاه داده را که ممکن است علت مشکلات کارایی باشند را مشخص کنید. در زمان مشابه، نیاز دارید تا کارکرد سطح بالایی از تراکنشها را، همچون ستونهایی که در تراکنش بهنگام سازی بروز شده اند یا ستونهایی که در پرس وجو بازیابی شده اند را مشخص کنید. شما از این اطلاعات برای انتخاب شاخصها و سازمان های فایل مناسب استفاده خواهید کرد.

بالاخره، در مرحله ۵ این کارها را به مراحل زیر می شکنیم:

- مرحله ۱-۵ تراکنشها را تحلیل کنید
- مرحله ۲-۵ سازمانهای فایل را انتخاب کنید
- مرحله ۳-۵ شاخصها را انتخاب کنید

مرحله ۱-۵ تراکنشها را تحلیل کنید

هدف

درک کردن عملکرد تراکنشهایی که بر روی پایگاه داده اجرا خواهند شد و تحلیل تراکنشهای مهم.

برای اینکه طراحی فیزیکی پایگاه داده به طور موثر انجام گیرد، نیاز دارید تا درک خوبی از تراکنشهایی که بر روی پایگاه داده اجرا خواهند شد داشته باشید.

در بیشتر مواقع بررسی همه تراکنشها زمانگیر می باشد، بنابراین شما حداقل باید تراکنشهای مهم را بررسی کنید. بررسی بدین صورت پیشنهاد شده است که ۲۰ درصد فعالترین پرس وجو های کاربر برای ۸۰ درصد مجموع دسترسی داده محاسبه شود. شما این قاعده ۸۰/۲۰ را زمانیکه تحلیل را انجام می دهید مفید خواهید یافت.

جهت کمک به مشخص کردن تراکنشهایی که جستجو می کنید، می توانید از *نقشه کاربرد تراکنش*[1] استفاده کنید، که بوسیله نمودار مشخص می کند که کدام جدولها به طور سنگین استفاده می شوند، و/ یا *ماتریس ارجاع متقابل تراکنش/ جدول*، که تراکنشهای مورد دستیابی توسط هر جدول را نشان می دهد. برای تمرکز بر قسمتهایی که ممکن است مشکل زا باشد، یک روش این است که موارد زیر را انجام دهید:

(۱) تمام مسیرهای تراکنشها به جدولها را ترسیم کنید.
(۲) مشخص کنید که کدام جدولها مکررا توسط تراکنشها مورد دستیابی قرار می گیرند.
(۳) تراکنشهای انتخابی را که این جدولها را در بر میگیرد تحلیل کنید.

تمام مسیرهای تراکنشها به جدولها را رسم کنید

در مراحل ۸-۱، و ۲-۳ متدلوژی طراحی منطقی پایگاه داده، مدلی را که در آن تراکنشهایی را که کاربران با نگاشت مسیرهای تراکنش به موجودیتها/ جدولها نیاز دارند پشتیبانی کنند را بررسی کردید. اگر شما از نمودار خط سیر تراکنش مشابه آنچه که در شکل ۱۷-۸ نشان داده شده است استفاده کرده اید، قادر خواهید بود از این نمودار برای مشخص کردن جدولهایی که بطور مکرر مورد دستیابی قرار گرفته اند استفاده کنید. از سوی دیگر، اگر تراکنشها را به طریق دیگر بررسی کرده اید، ایجاد ماتریس ارجاع متقابل تراکنش/ جدول میتواند مفید واقع شود. ماتریس، تراکنشهای موردنیاز و جدولهایی که دسترسی دارند را نشان می دهد. برای مثال، جدول ۱-۱۳ ماتریس ارجاع متقابل برای انتخابهای زیرین داده، بهنگام سازی/ حذف، و تراکنشهای پرس و جوی *StayHome* را نشان می دهد:

[1] *transaction usage map*

(e) جزئیات عضو جدید را که در شعبه معینی ثبت نام میکند را وارد کنید.
(k) جزئیات عضو معینی را بهنگام/ حذف کنید.
(p) عنوان، فهرست، و دسترس پذیری همه فیلمهای موجود در شعبه معینی را، به ترتیب فهرست لیست کنید.
(q) عنوان، فهرست، و دسترس پذیری همه فیلمها برای نام هنرپیشه معینی در شعبه مشخصی، به ترتیب عنوان لیست کنید.
(r) عنوان، فهرست، و دسترس پذیری همه فیلمها برای نام کارگردان معینی در شعبه مشخصی، به ترتیب عنوان لیست کنید.
(s) جزئیات همه فیلمها را که در اجاره عضو مشخصی است لیست کنید.

ماتریس، به صورت بصری، الگوهای دسترسی تراکنشهایی که در پایگاه داده اجرا خواهد شد را خلاصه می کند. برای مثال، ماتریس مشخص می کند که تراکنش (e) جدول Staff را می خواند و همچنین رکوردها را داخل جدولهای Member و Registration وارد می کند. جهت هر چه بیشتر مفید واقع شدن، باید تعداد دسترسی های طی چندین دوره زمانی (برای مثال، ساعته، روزانه، هفتگی) در هر سلول را مشخص کنید. با این وجود، جهت ساده نگه داشتن ماتریس، ما این اطلاعات را نشان نداده ایم. این ماتریس جدولهای Video و VideoForRent را که توسط چهار تراکنش پرس وجو مورد دستیابی قرار گرفته اند را نشان می دهد.

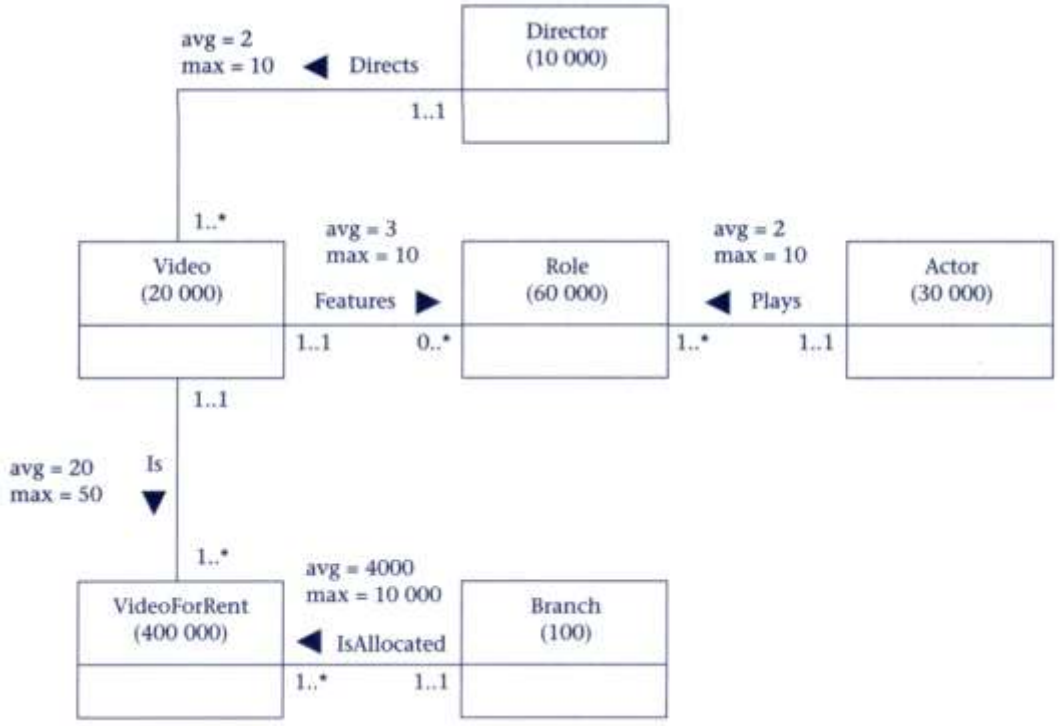
جدول ۱-۱۳ جدولها و تراکنشهای ارجاع متقابل

تراکنش/جدول	(e)				(k)				(p)				(q)				(r)				(s)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Branch																								
Staff		X																						
Video										X				X				X				X		
VideoForRent										X				X				X				X		
RentalAgree-ment																						X		
Member	X					X	X	X														X		
Registration	X																							
Actor														X										
Role														X										
Director																		X						

I = Insert; R = Read; U = Update; D = Delete

شکل ۲-۱۳

مدل منطقی داده ساده شده که رخدادهای مورد انتظار را نشان می دهد.



اطلاعات فراوانی را مشخص کنید

در بحث با مدیران شعبه *StayHome* تخمین زده شد که حدود ۲۰۰۰۰ عنوان فیلم و ۴۰۰۰۰۰ فیلم برای اجاره، توزیع شده در ۱۰۰ اداره شعبه ها، با متوسط ۴۰۰۰ و حداکثر ۱۰۰۰۰ فیلم برای اجاره در هر شعبه وجود دارد. بعلاوه، *StayHome* داده را برای ۱۰۰۰۰ کارگردان و ۳۰۰۰۰ بازیگر اصلی در ۶۰۰۰۰ نقش نگه میدارد. شکل ۲-۱۳ مدل داده منطقی کاهش یافته را با این تعداد اضافه شده نشان می دهد.

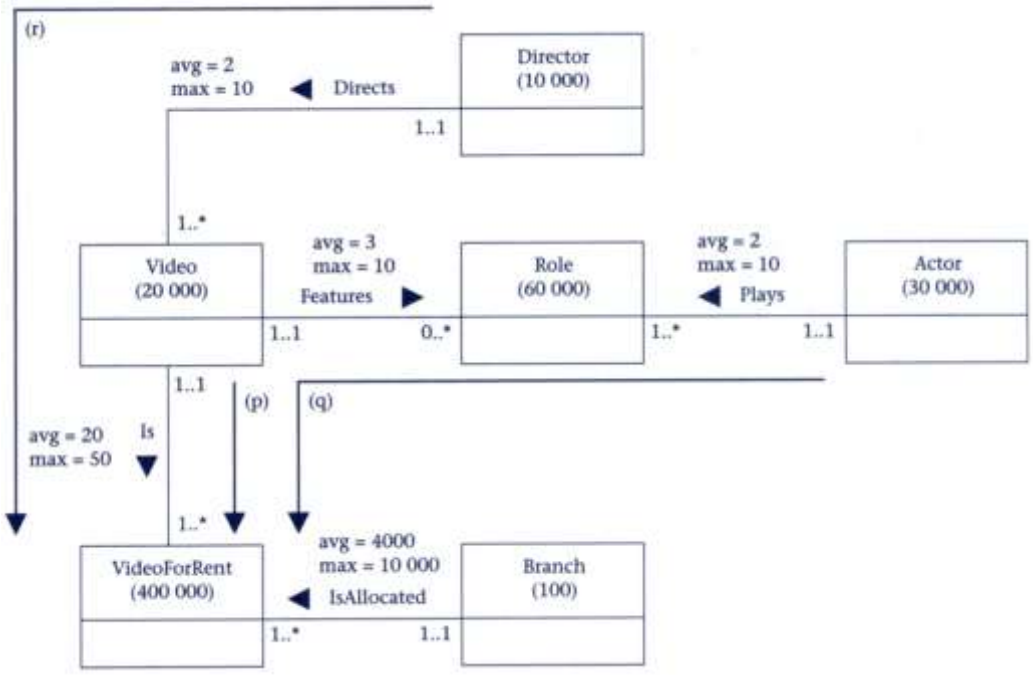
شکل ۳-۱۳ نقشه کارکرد تراکنش برای تراکنشهای (p)، (q) ، و (r)، که همه شان به جدولهای VideoForRent و Video دسترسی دارند را نشان می دهد. با توجه به سایز جدول VideoForRent، مهم است که دسترسی به این جدول به طور موثر صورت پذیرد. شما حال ممکن است بگویید که تحلیل نزدیکتر تراکنشهای در برگیرنده این جدولها می تواند مفید باشد.

در بررسی هر تراکنش، مهم است که شما نه تنها متوسط و حداکثر تعداد زمانهایی را که تراکنش در هر ساعت اجرا می شود را در نظر بگیرید، بلکه باید روز و ساعتی که تراکنش اجرا شده، و بحبوبه مصرف محتمل را نیز در نظر بگیرید. برای مثال، بعضی تراکنشها ممکن است بیشتر اوقات با نرخ متوسطی اجرا شوند، اما در ساعات ۱۴،۰۰ و ۱۶،۰۰ پنجشنبه قبل از روز جمعه صبح دارای بحبوحه مصرف باشند. دیگر تراکنشها ممکن است فقط در ساعات مشخصی اجرا شوند، برای مثال ۱۸،۰۰ تا ۲۱،۰۰ روزهای جمعه/ شنبه ، که در بحبوحه مصرف می باشند.

زمانیکه تراکنشها نیاز به دستیابی به جدولهای مشخصی دارند، در اینصورت الگوی عمل خیلی مهم است. اگر این تراکنشها به شیوه دو به دو ناسازگار عمل کنند، ریسک مشکلات محتمل کارایی پایین می آید. با این وجود، اگر الگوهای عملشان دارای ناسازگاری باشند، مشکلات نهانی احتمالا توسط بررسی ساختار جدولها برای بهبود کارایی کم می شود، همانطور که در مرحله ۶ فصل بعدی بحث خواهیم کرد.

شکل ۳-۱۳

نقشه کارکرد تراکنش برای تراکنشهای ساده.



تحلیل کاربرد داده

بعد از شناسایی تراکنشهای مهم، حالا باید هر یک را با جزئیات بیشتر تحلیل کنید. برای هر تراکنش، باید موارد زیر را مشخص کنید:

(۱) جدولها و ستونهایی که توسط تراکنش مورد دسترسی قرار می گیرند و همچنین نوع دسترسی; یعنی، آیا آن تراکنش از نوع درج، بهنگام سازی، حذف، یا بازیابی است.

- برای تراکنش بهنگام سازی، از آنجائیکه این ستونها احتمال دارد که برای اجتناب از ساختار دسترسی کاندید شوند (همچون شاخص ثانوی) ستونهایی که بهنگام شده اند را یادداشت کنید.

(۲) ستونهای استفاده شده در *شرایط جستجو* (*search conditions*) (در SQL، این شرایط در شرط WHERE مشخص می شوند). بررسی کنید ببینید که آیا شرایط شامل موارد زیر است:

i. تطبیق الگو ; برای مثال: (name LIKE '%Smith%') ;
ii. جستجوهای محدوده; برای مثال (salary BETWEEN 30000 AND 40000) ;
iii. بازیابی کلید تطبیق دقیق; برای مثال: (salary= 30000).

این نه تنها به پرس و جوها اعمال می شود بلکه به تراکنشهای بهنگام سازی و حذف، که می تواند رکوردها را به رکوردهای بهنگام شده/حذف شده در جدول محدود کند اعمال می شود.

(۳) برای پرس وجو، ستونهایی که در الحاق دو یا چند جدول در برگرفته می شوند.

- دوباره، شاید این ستونها برای ساختارهای دسترسی کاندید شوند.

(۴) تکرار مورد انتظار در تراکنشی که اجرا خواهد شد; برای مثال، تراکنش به طور تخمینی ۵۰ بار در روز اجرا خواهد شد.

(۵) اهداف کارائی برای تراکنشها; برای مثال، تراکنش باید در ۱ ثانیه کامل شود.

- ستونهای استفاده شده در هر نوع شرایط جستجو برای تراکنشهای بحرانی یا تکراری برای ساختارهای دسترسی باید اولویت بالاتری داشته باشد.

شکل ۴-۱۳

شکل تحلیل تراکنش نمونه .

Transaction Analysis Form **1-Jan-2000**

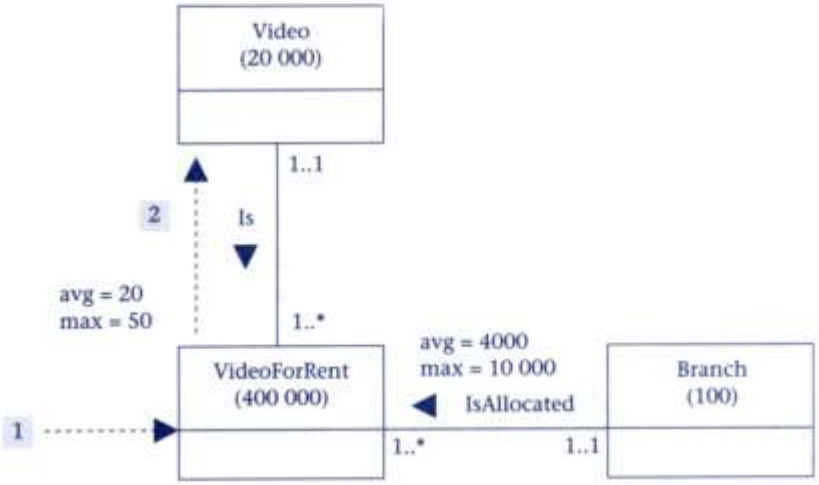
Transaction (p) List the title, category, and availability of all videos at at specified branch, ordered by category.

Transaction volume
Average: 50 per hour
Peak: 100 per hour (between 18.00 and 21.00 every day of the week)

```
SELECT title, category, availability
FROM video v INNER JOIN videoforrent vfr ON
        vfr.catalogno = v.catalogno
WHERE vfr.branchNo = 'B001'
ORDER BY title;
```

Search condition: branchno = 'B001'
Join columns: vfr.catalogno = v.catalogno
Ordering column: title
Grouping column: none
Built-in functions: none
Columns updated: none

Transaction usage map



Access	Entity	Type of access	No. of references		
			Per transaction	Avg per hour	Peak per hour
1	VideoForRent (entry)	R	4000–10 000	200 000–500 000	400 000–1 000 000
2	Video	R	4000–10 000	200 000–500 000	400 000–1 000 000
Total references			8000–20 000	400 000–1 000 000	800 000–2 000 000

شکل ۴-۱۳ مثالی از شکل تحلیل تراکنش برای تراکنش (p) را نشان می دهد. این شکل نشان می دهد که تکرار متوسط این تراکنش ۵۰ بار در ساعت با بحبوحه مصرف ۱۰۰ بار در ساعت بین ۱۸،۰۰ و ۲۱،۰۰ است. بعبارت دیگر، نصف شعبه ها این تراکنش را در هر ساعت اجرا خواهند کرد و در زمان اوج همه شعبه ها این تراکنش را یکبار در هر ساعت اجرا خواهند کرد.

شکل همچنین دستورات SQL موردنیاز و نقشه کاربرد تراکنش را نشان می دهد. در این مرحله، دستورات کامل SQL احتمالا با جزئیات بیشتری همراه خواهد بود، اما شما بایستی حداقل انواع جزئیاتی را که مجاور دستور SQL نشان داده شده اند را مشخص کنید، که بنامهای :

- شرط جستجوی دیگری که استفاده خواهد شد;
- ستون های دیگری که برای الحاق جدولها به همدیگر نیاز است (برای تراکنشهای بازیابی داده);
- ستونهایی که جهت مرتب سازی نتیجه استفاده می شوند (برای تراکنشهای بازیابی داده);
- ستونهایی که برای گروه بندی داده استفاده می شوند (برای تراکنشهای بازیابی داده);
- هر نوع تابع داخلی که ممکن است استفاده شود (همچون AVG ,SUM);
- ستونی که توسط تراکنش بهنگام خواهند شد.

شما از این اطلاعات برای مشخص کردن شاخصهای مورد نیاز استفاده می کنید، که مختصرا بحث خواهیم کرد. زیر نقشه کاربرد تراکنش جزئیات مستندات بصورت تفکیک شده وجود دارد:

- چگونه هر جدول دستیابی می شود (در این مورد منظور خواندن جدول);
- در هر بار اجرای تراکنش چه تعداد رکورد دستیابی خواهد شد;
- در هر ساعت و زمانهای بحبوحه مصرف چه تعداد رکورد به طور متوسط دستیابی خواهد شد.

توجه شود که ، در تراکنش بهنگام سازی، دو دسترسی روی جدول وجود خواهد داشت: یکی برای خواندن داده و دیگری بهنگام سازی داده.

اطلاعات تکرار جدولهایی را مشخص خواهد کرد که باید جهت اطمینان از اینکه ساختارهای دستیابی بطور مناسب استفاده شده اند، بررسی صورت پذیرد. همانطور که در بالا ذکر شد، شرایط جست و جویی که توسط تراکنش ها استفاده می شوند جهت اختصاص اولویت بالاتر به ساختارهای دستیابی محدودیت زمانی دارند.

مرحله ۲-۵ سازمان های فایل را انتخاب کنید

هدف

مشخص کردن سازمان فایل موثر برای هر جدول پایه.

یکی از اهداف عمده طراحی فیزیکی پایگاه داده ذخیره سازی داده به صورت موثر است. برای مثال، اگر شما می خواهید رکوردهای پرسنل را بترتیب حروف الفبایی نام بازیابی کنید، مرتب سازی فیلد با نام پرسنلی سازمان فایل خوبی خواهد بود. با این وجود، اگر می خواهید پرسنل هایی را بازیابی کنید که حقوقشان در محدوده مشخصی است، فایلی که با نام پرسنل مرتب شده باشد سازمان فایل مناسبی نمی باشد.

در موارد پیچیده، بعضی از سازمان فایلها جهت بارگذاری دسته ای[1] داده داخل پایگاه داده کارآمد هستند ولی بعد از آن غیر کارا می شوند. بعبارت دیگر، ممکن است بخواهید از ساختار ذخیره سازی کارا برای برپا ساختن پایگاه داده استفاده کنید و سپس برای استفاده معمول آن را تغییر دهید.

بنابراین هدف این مرحله انتخاب سازمان فایل بهینه برای هر جدول است، در صورتیکه DBMS هدف این را اجازه دهد. در برخی موارد، ممکن است متوجه شوید که DBMS رابطه ای به شما هیچ گزینه ای برای انتخاب سازمان فایلها در اختیارتان نمی گذارد، بنابراین شاید بعضی توسط شاخصها بوجود آیند. با این وجود، مفهوم سازمان فایلها و شاخصها را به طور کامل درک کنیم، در زیر راهنمایی را برای انتخاب سازمان فایل بر اساس انواع فایلهای زیرین فراهم کرده ایم:

- فایلهای پشته[2]
- فایلهای درهم[3]
- روش دستیابی ترتیبی شاخص دار[4]
- B^+-Tree.

اگر DBMS هدف شما اجازه انتخاب سازمان فایل را نمی دهد، می توانید این مرحله را رد شده و به مرحله بعدی، که مرحله ۳-۵ است بروید.

پشته (نامرتب)

پشته ساختار ذخیره سازی خوبی در موقعیتهای زیر است:

(۱) وقتی داده داخل جدول بصورت انباشته شده بارگذاری می شود. برای مثال، ممکن است بخواهید دسته ای از رکوردها را بعد از اینکه ایجاد شدند وارد کنید. اگر پشته را بعنوان سازمان فایل اولیه انتخاب کنید، شاید بازسازی فایل بعد از تکمیل درج داده موثرتر باشد.

(۲) جدول فقط به اندازه صفحات معدودی طول داشته باشد. در این مورد، زمان شناسایی هر رکورد کوتاه است، حتی اگر جدول داخلی به صورت ترتیبی جستجو شود.

(۳) وقتی هر رکورد در جدول زمانیکه مورد دستیابی قرار گیرد بازیابی شود. برای مثال، آدرسهای تمام عضوهای *StayHome* را بازیابی کنید.

(۴) وقتی جدول دارای ساختار دستیابی اضافی، همچون کلید شاخص، است. و ذخیره سازی پشته می تواند جهت حفاظت از فضا استفاده شود.

فایل درهم

فایل درهم زمانیکه رکوردها براساس تطبیق دقیق مقدار فیلد درهم بازیابی می شوند و ترتیب دستیابی تصادفی باشد ساختار ذخیره سازی خوبی است. برای مثال اگر جدول Member براساس memberNo درهم شود بازیابی رکورد با memberNo معادل 'M25178' موثر است. با این وجود درهم سازی ساختار ذخیره سازی مناسب در موقعیتهای زیر نیست:

---

[1] Bulk-loading
[2] Heap-file
[3] Hash-file
[4] Indexed Sequential Access Method (ISAM)

(۱) زمانیکه رکوردها بر اساس تطبیق الگوی مقدار فیلد درهم بازیابی شود. برای مثال، تمام عضوهایی که شماره عضویتشان (memberNo) با کاراکترهای 'M2' شروع می شوند را بازیابی کنید.

(۲) زمانیکه رکوردها براساس محدوده مقدار فیلد درهم بازیابی شود. برای مثال، تمام عضوهای با شماره عضویت بین 'M200000' و 'M200100' را بازیابی کنید.

(۳) وقتی رکوردها بر اساس ستونی به غیر از ستون درهم بازیابی شوند. برای مثال، اگر جدول Member براساس memberNo درهم شود در اینصورت درهم سازی نمی تواند برای جستجوی رکورد براساس ستون lName استفاده شود. در این مورد، جستجوی خطی برای پیدا کردن رکورد لازم است یا اینکه lname را بعنوان شاخص ثانویه بیافزاییم (مرحله ۳-۵ را نگاه کنید).

(۴) وقتی رکوردها بر اساس تنها بخشی از فیلد درهم بازیابی شوند. برای مثال، اگر جدول Role براساس catalogNo، actorNo درهم سازی شود. بنابراین درهم سازی نمی تواند جستجوی رکورد براساس تنها ستون catalogNo استفاده شود. در این صورت دوباره، برای پیدا کردن رکورد جستجوی خطی لازم خواهد بود.

(۵) زمانیکه ستون درهم مکررا بهنگام می شود، DBMS باید محتوای رکورد را حذف کند و در صورت امکان به آن آدرس جدیدی بدهد (اگر نتایج تابع درهم در آدرس جدید باشد). بنابراین، بهنگام سازی ستون درهم کارایی این را تحت تاثیر قرار می دهد.

روش دستیابی ترتیبی شاخص دار (ISAM)

ISAM ساختار ذخیره سازی سلیس تری نسبت به درهم سازی است; و بازیابی را بر اساس تطبیق کلید، تطبیق الگو، مقادیر محدوده، و خصوصیات بخشی کلید پشتیبانی می کند. با این وجود، شاخص ISAM ایستا بوده، و زمانیکه فایل ایجاد شود بوجود می آید. بنابراین شما مشاهده خواهید کرد زمانیکه جدول بهنگام شود کارایی فایل ISAM نیز کم می شود. بهنگام شدن همچنین باعث می شود فایل ISAM ترتیب کلید دستیابی را گم کند، بنابراین بازیابی با ترتیب کلید دستیابی کندتر خواهد شد. این دو مشکل بوسیله سازمان فایل B^+-Tree حل می شود.

B^+-Tree

دوباره B^+-Tree نیز ساختار ذخیره سازی سلیستری نسبت به درهم سازی است. و بازیابی ها را بر اساس تطبیق دقیق کلید، تطبیق الگو، دامنه مقادیر و خصوصیات بخشی کلید پشتیبانی می کند. شاخص B^+-Tree پویا بوده و هنگام رشد جدول به تبع آن رشد می کند. بنابراین، برخلاف ISAM کارایی فایل B^+-Tree موقع بهنگام شدن جدول بدتر نمی شود. B^+-Tree همچنین ترتیب کلید دستیابی را حتی زمانیکه فایل بهنگام شود حفظ می کند. بنابراین بازیابی رکوردها به ترتیب کلید دستیابی کاراتر از ISAM است. با این وجود اگر جدول بطور مکرر بهنگام نشود ساختار ISAM شاید کاراتر از زمانی باشد که یک سطح شاخص کمتر از B^+-Tree دارد و آن که گره های برگ شامل اشاره گرهایی به رکوردهای واقعی جدول نسبت به رکوردهای واقعی خودشان باشند.

سازمان فایل انتخابی را مستند کنید

انتخاب سازمان فایل بایستی کاملا بهمراه دلایل انتخاب آن مستند شود. خصوصا، دلایل انتخاب یک سازمان فایل نسبت به دیگری وقتی چندین سازمان فایل وجود دارد را نیز مستند کنید.

مرحله ۳-۵ شاخصها را انتخاب کنید

هدف

مشخص کردن اینکه آیا افزودن شاخص کارایی سیستم را افزایش میدهد یا نه .

یک روش برای اینکه سازمان فایل مناسبی برای جداول انتخاب کنیم این است که رکوردها را به صورت نامرتب نگه داریم و در صورت نیاز تعدادی **شاخصهای ثانوی** روی آنها ایجاد کنیم. روش دیگر مرتب سازی رکوردهای جدول این است که **شاخص خوشه** یا **اولیه** ای را مشخص کنیم. در این مورد، باید ستونی را جهت مرتب کردن یا خوشه سازی رکوردها با خصوصیات زیر در نظر بگیرید:

- ستونی که اکثر اوقات برای عمل الحاق استفاده می شود، که این باعث کاراتر شدن عمل می شود، یا
- ستونی که از آن اغلب اوقات برای دستیابی رکوردها بترتیب ستونها در جدول استفاده می شود.

اگر ستون مرتب سازی انتخاب شده کلید جدول باشد، شاخص شاخص اصلی خواهد بود; اگر ستون مرتب سازی کلید نباشد، شاخص شاخص خوشه خواهد بود. بیاد داشته باشید که شما می توانید تنها برای هر فایل یا شاخص اصلی و یا شاخص خوشه داشته باشید.

شاخصها را مشخص کنید

نسخه ابتدایی SQL استاندارد دستوراتی برای ایجاد و حذف شاخصها داشت. اما، این دستورات از نسخه استاندارد اصلی سال ۱۹۹۲ حذف شد آنهم به این علت که مفاهیم فیزیکی را به جای مفاهیم منطقی در نظرگرفته بودند. دستورات همچنین در پیشنویس تجدید نظرشده استاندارد، که عموما بنام SQL3 نامیده می شوند، نیز ظاهر نشدند و این روند در سالهای بعدی نیز ادامه یافت. همانطور که گفته شد، بیشتر DBMS های رابطه ای این دستورات را در یک شکل یا اشکال دیگر پشتیبانی می کنند. دستورات SQL ایکه در زیر استفاده می کنیم نمونه ای از آنچه که محصولات کنونی پشتیبانی میکنند می باشد.

برای ایجاد شاخص در SQL، به طور نمونه از دستور CREATE INDEX استفاده کنید. برای مثال، جهت ایجاد شاخص اصلی در جدول Video براساس ستون catalogNo، ممکن است از دستور SQL زیر استفاده کنید:

```
CREATE UNIQUE INDEX catalogno_index
ON video (catalogno);
```

اگر خواستید شاخص خوشه در جدول VideoForRent براساس ستون catalogNo ایجاد کنید، از دستور SQL زیر استفاده کنید:

```
CREATE INDEX catalogno_index
ON videoforrent (catalogno) CLUSTER;
```

همانطور که گفتیم، در بعضی از سیستمها سازمان فایل ثابت است. برای مثال، تا سالهای اخیر اوراکل تنها B^+-Tree را پشتیبانی می کرد، اما حالا خوشه های درهم سازی را نیز پشتیبانی می کند. در طرف دیگر، INGRES مجموعه گسترده ای از ساختارهای شاخص متفاوتی را پیشنهاد می کند که می توانید با استفاده از عبارت اختیاری دستور CREATE INDEX انتخاب کنید :

```
[STRUCTURE = BTREE | ISAM | HASH | HEAP|];
```

جهت حذف شاخص در SQL، به طور نمونه می توانید از دستور DROP INDEX استفاده کنید. برای مثال، جهت حذف شاخص خوشه catalogno_index، ممکن است از دستور SQL زیرین استفاده کنید:

```
DROP INDEX catalogno_index;
```

شاخصهای ثانوی را انتخاب کنید

شاخصهای ثانوی مکانیزمی است که می توانیم با مشخص کردن کلید اضافی برای جدول پایه داده ها را به طور موثر بازیابی کنیم. برای مثال، جدول Member احتمالا براساس شماره عضو ،memberNo ، *شاخص اصلی* درهم سازی شود.
بنابراین، دستیابی مکرر بر این جدول براساس ستونlName (last Name) وجود دارد. در این مورد، ممکن است تصمیم بگیرید که lName را بعنوان *شاخص ثانوی* در نظر بگیرید.

بنابراین، کلیاتی درباره نگهداری و استفاده از شاخصهای ثانوی بطوریکه تعادلی را بهنگام بازیابی داده در کارایی بوجود آورد وجود دارد. این کلیات شامل:

- افزودن رکورد شاخص به هر شاخص ثانوی زمانیکه رکوردی در جدول درج شود;
- بهنگام سازی شاخص ثانوی زمانیکه رکورد متناظر در جدول بهنگام شود;
- افزایش در فضای دیسک جهت ذخیره شاخص ثانوی مورد نیاز است;
- تنزل کارایی ممکن طی بهینه سازی پرس و جو، از آنجائیکه بهینه ساز پرس و جو باید همه شاخصهای ثانوی را قبل از انتخاب استراتژی اجرای بهینه در نظر بگیرد.

راهنمایی جهت انتخاب 'لیست خواسته ها' شاخصها

یک روش برای مشخص کردن اینکه کدام شاخصهای ثانوی را نیاز دارید، تولید 'لیست خواسته های' ستونهایی است که فکر می کنید برای شاخص بندی مناسب هستند، و روش دوم در نظرگرفتن اثر نگهداری هر یک از این شاخصها می باشد. ما به شما کمک می کنیم تا چنین 'لیست خواسته' ای را تولید کنید:

(۱) عموما، کلید اصلی جدول را اگر کلید سازمان فایل نباشد بعنوان شاخص در نظر بگیرید. اگرچه SQL2 استاندارد عبارتی را برای مشخص کردن کلیدهای اصلی چنانکه در مرحله ۱-۴ فصل قبل دیدیم مشخص کرده است، باید توجه شود که این بتنهایی شاخص بودن کلید اصلی را تضمین نمی کند.

(۲) جدولهای کوچک را شاخص بندی نکنید. جستجوی جدول های کوچک در حافظه کاراتر از این است که ساختار شاخص دیگری را ذخیره کنیم.

(۳) شاخص ثانوی را به ستونی که در بازیابی داده به طور سنگین استفاده می شود اضافه کنید (برای مثال، شاخص ثانوی را به جدول Member براساس ستون lName ، همانطور که دربالا بحث شد بیافزائید).

(۴) شاخص ثانوی را به کلید خارجی که دستیابی مکرر براساس آن صورت می پذیرد اضافه کنید. برای مثال، ممکن است مکررا جدولهای VideoForRent و Branch را روی ستون branchNo (شماره شعبه) با هم الحاق کنید. بنابراین، موثرتر خواهد بود که شاخص ثانوی را به جدول VideoForRent براساس branchNo اضافه کنید.

(۵) شاخص ثانوی را به ستونهایی که مکررا در موارد زیر درگیر هستند اضافه کنید:
- معیارهای الحاق یا انتخاب;
- ORDER BY;
- GROUP BY;
- عملیاتی که شامل مرتب سازی باشند (همچون UNION یا DISTINCT).

(۶) شاخص ثانوی را به ستونهایی که شامل توابع توکار (Built-in) هستند، بهمراه ستونی که جهت جمع کردن توابع توکار استفاده می شوند اضافه کنید. برای مثال، برای پیدا کردن متوسط حقوق پرسنل هر شعبه، می توانید از پرس و جوی SQL زیر استفاده کنید:

```
SELECT branchno, AVG (salary)
FROM staff
GROUP BY branchno;
```

از راهنمایی قبلی، می توانید افزودن شاخص به ستون branchNo را بوسیله خاصیت عبارت GROUP BY در نظر بگیرید. با این وجود، در نظرگرفتن شاخص در هر دو ستون branchNo و salary میتواند موثرتر باشد. این به DBMS اجازه می دهد بدون اینکه به فایل داده دستیابی داشته باشد پرس و جوی داخلی از داده ها را تنها در شاخص انجام دهد. این جستجو بعضی اوقات *طرح تنها-شاخص*[1] نامیده میشود، بدین صورت که پاسخ مورد نیاز می تواند تنها با استفاده از داده موجود در شاخص تولید شود.

(۷) بعنوان مورد کلی تر راهنمایی قبلی، شاخص ثانوی را به ستونهایی اضافه کنید که میتواند نتیجه طرح تنها-شاخص را داشته باشد.

(۸) از شاخص بندی جدول یا ستونهایی که مکررا بهنگام می شوند خودداری کنید.

(۹) از شاخص بندی ستونهایی که پرس وجو مقدار قابل توجهی از رکوردهای جدول را بازیابی می کند (برای مثال، ۲۵ درصد) خودداری کنید، حتی اگر جدول بزرگ باشد. در این مورد، بهتر است به جای جستجو بر اساس شاخص جستجوی داخلی جدول انجام گیرد.

(۱۰) از شاخص بندی ستونهایی که شامل رشته های کارکتری بلند باشد خودداری کنید.

نکته اگر معیار جستجو شامل بیش از یک موقعیت باشد، و یکی از عبارتها شامل عبارت OR باشد، و عبارت ترتیب index/order را نداشته باشد، بنابراین افزودن شاخصها به دیگر ستونها کمکی به بهبود سرعت پرس و جو نمی کند، به علت اینکه جستجوی خطی جدول هنوز مورد نیاز می باشد. برای مثال، فرض کنید تنها ستونهای **category** و **dailyRental** جدول **video** شاخص بندی شده باشند، و شما نیاز به استفاده از پرس و جوی زیر دارید:

```
SELECT  *
FROM video
WHERE (category = 'Action' OR dailyrental > 3 OR price >15);
```

اگرچه دو شاخص می توانست برای پیداکردن رکوردها جائیکه (category ='Action' یا dailyrental > 3) استفاده شود، این حقیقت که ستون **price** شاخص بندی نشده است بدین معنی خواهد بود که این شاخصها نمی تواند برای عبارت کامل WHERE استفاده شود.

از طرف دیگر، اگر موقعیتهای جستجو در عبارت WHERE با همدیگر AND شده باشند، دو شاخص موجود در ستونهای **category** و **dailyRental** می توانست برای بهبود پرس و جو استفاده شود.

---

[1] Index-only plan

حذف شاخصها از ' لیست خواسته '

بعد از اینکه 'لیست خواسته ' های شاخصها را ترسیم نمودید، حالا باید اثر هریک از اینها را بر روی تراکنشهای بهنگام سازی در نظر بگیرید. اگر نگهداری شاخصها باعث کاهش سرعت تراکنشهای بهنگام سازی شود، در اینصورت حذف شاخص از لیست را نیز در نظر بگیرید. توجه کنید که، با این وجود، وجود شاخصهای مشخصی ممکن است همچنین باعث شود عملیات بهنگام سازی موثرتر انجام گیرد. برای مثال، اگر بخواهید حقوق عضوی از پرسنل را با شماره پرسنلی عضو اش، staffNo، بهنگام کنید و شاخصی بر روی staffNo داشته باشید، در اینصورت رکورد بهنگام شده می تواند به سرعت پیدا شود.

نکته ایده خوبی که برای تجربه وجود دارد این است که در صورت امکان مشخص کنید که آیا شاخص کارایی را بهبود می دهد، بهبود خیلی کمی را ایجاد می کند، یا بالعکس کارایی را تحت تاثیر قرار می دهد. در مورد آخر، واضح است که شما باید این شاخص را از لیست خواسته ها حذف کنید. اگر با وجود شاخص اضافی بهبود کارایی کوچکی مشاهده شد، شاید لازم باشد بررسی کنید که در چه شرایطی شاخص می تواند مفید باشد، و آیا این شرایط بقدرکافی جهت تضمین پیاده سازی شاخص مهم هستند.

بعضی سیستمها به شما اجازه می دهند که استراتژی بهینه ساز را برای اجرای پرس وجوی مشخص یا بهنگام، که بعضی اوقات طرح اجرای پرس وجو[1] نامیده می شود بررسی کنید. بعنوان مثال، اوراکل برنامه تشخیصی سودمند EXPLAIN PLAN دارد، مایکروسافت اکسس تحلیل کننده کارایی را دارد، DB2 برنامه سودمند EXPLAIN را دارد، و INGRES امکان مشاهده آنلاین QEP را دارد. وقتیکه پرس و جو از حد انتظار کندتر اجرا می شود، استفاده از چنین تسهیلاتی برای پیداکردن دلایل کندی پرس و جو ارزنده است، و می توانیم راه دیگری را برای بهبود کارایی پرس وجو بیابیم.

نکته اگر تعداد زیادی از رکوردها بهمراه یک یا چند شاخص وارد جدول شده باشند، بهتر است اول از همه شاخصها را حذف کرده، بعد عمل درج را انجام دهیم، و سپس شاخصها را دوباره ایجاد کنیم. بعنوان یک قاعده سرانگشتی، اگر عمل درج اندازه جدول را حداقل ۱۰ درصد افزایش دهد، شاخصها را موقتا حذف کنید.

آمار پایگاه داده را بهنگام کنید

قبلا یادآوری کردیم که بهینه ساز پرس وجو تکیه بر آمار پایگاه داده دارد که جهت انتخاب راهبرد بهینه در کاتالوگ سیستم نگهداری می شود. هر موقع که شما شاخصی ایجاد می کنید، DBMS به طور اتوماتیک وجود شاخص را به کاتالوگ سیستم می افزاید. با این وجود، ممکن است بفهمید که DBMS شما را مجبور می کند که برنامه های سودمند را برای بهنگام سازی آمار موجود در کاتالوگ سیستم مربوط به جدولها و شاخصها اجرا کنید.

[1] Query Execution Plan

شاخصهای انتخاب شده بایستی کاملا، بهمراه دلایل انتخابشان مستند شوند. خصوصا، اگر دلایل کارایی وجود داشته باشد که چرا بعضی از ستونها نباید شاخص بندی شوند را نیز مستند کنید.

۱۳-۳ سازمان های فایل و شاخصها برای *StayHome* با مایکروسافت اکسس ۹۷

مانند بیشتر، اما نه همه، DBMS های PC ، مایکروسافت اکسس از سازمان فایل ثابت استفاده می کند، بنابراین اگر شما از اکسس استفاده می کنید، مرحله ۲-۵ را میتوانید رد شوید.

۱۳-۳-۱ راهنمایی هایی برای انتخاب شاخص

مایکروسافت اکسس، با وجود مواردی که در بالا ذکر شد از شاخص پشتیبانی می کند. در اکسس، کلید اصلی جدول به طور اتوماتیک شاخص بندی می شود، اما فیلدی که نوع داده اش Memo، Hyperlink، یا OLE Object باشد نمی تواند شاخص بندی شود. برای دیگر فیلدها، مایکروسافت توصیه می کند که تنها در صورتی باید شاخص بندی را را روی فیلدها در نظر بگیرید که موارد زیر اعمال شود:

- نوع داده ی فیلد، Text، Number، یا Date/Time باشد;
- شما جستجو را برای مقادیر ذخیره شده در فیلد پیش بینی کرده باشید;
- مرتب سازی در فیلد را پیش بینی کرده باشید;
- مرتب سازی چندین مقدار مختلف در فیلد را پیش بینی کرده باشید. اگر چندین مقدار موجود در فیلد مشابه باشند، شاخص به طور مشخص سرعت پرس وجو را افزایش نمی دهد.

بعلاوه، مایکروسافت توصیه می کند :

- شما باید فیلدهای شاخص بندی شده را در دو طرف الحاق در نظر بگیرید یا رابطه ای بین این فیلدها ایجاد کنید، در چنین مواردی اگر در صورتیکه شاخصی درحال حاضر موجود نباشد اکسس به طور خودکار شاخص را بر روی فیلد کلید خارجی ایجاد خواهد کرد;
- وقتی رکوردها را بوسیله مقادیرشان در فیلد الحاق گروه بندی می کنید، شما باید GROUP BY را برای فیلدی که در جدول مشابه بصورت فیلدی که محاسبه روی آن انجام میگیرد مشخص کنید.

مایکروسافت اکسس می تواند موقعیتهای جستجوی ساده و پیچیده را بهبود بخشد (در اکسس *عبارات*[1] نامیده میشود). برای انواع مشخصی از عبارات پیچیده، مایکروسافت اکسس از تکنولوژی دستیابی داده بنام Rushmore استفاده می کند، که برای رسیدن به سطح عمیقی از بهینه سازی صورت می گیرد. عبارت پیچیده از دو عبارت ساده با عملگرهای AND یا OR تشکیل شده است، همچون:

```
branchno = 'B001' AND available = Yes
category = 'Action' OR dailyrental > 3
```

[1] Expressions

در اکسس، عبارت پیچیده کاملا یا اندکی بسته بر اینکه آیا یک یا دو عبارت ساده قابل بهینه سازی باشد، و کدام عملگر برای ترکیب آنها استفاده شده باشد قابل بهینه سازی است. عبارت پیچیده، قابل بهینه سازی RushMore است اگر همه سه شرط زیر درست باشد:

- عبارت از AND و OR برای الحاق دو شرط استفاده کند.
- هر دو شرط از عبارات قابل بهینه سازی تشکیل شوند.
- هر دو عبارت شامل فیلدهای شاخص بندی باشند. فیلدها می تواند منفردا شاخص بندی شده یا قسمتی از شاخص فیلد چندگانه باشند.

ایجاد کردن شاخصها در اکسس

شاخص را در اکسس بوسیله تنظیم کردن خصیصه شاخص جدول در بخش خصوصیات فیلدها در Design View جدول ایجاد می کنید. خصیصه شاخص دار مقادیر زیرین را دارد:

NO	No Index (پیش فرض)
Yes (Duplicates OK)	شاخص تکرار را اجازه میدهد.
Yes(No Duplicates)	شاخص تکرار را اجازه نمیدهد

ما مثالی از تنظیم شاخص در شکل ۴-۱۲ را در فصل قبل دیدیم.

۲-۳-۱۳ شاخصهای *StayHome*

بر اساس راهنمایی که در بالا آوردیم، حال باید مطمئن شوید که برای هر جدول کلید اصلی ایجاد کرده باشید، که باعث می شود اکسس به طور خودکار این ستون را شاخص بندی کند. دوما آنکه، باید مطمئن شوید که به طور صحیح همه رابطه ها در پنجره Relationships را ایجاد کرده باشید، که باعث می شود اکسس به طور خودکار ستونهای کلید خارجی را شاخص بندی کند.

از تراکنشهای *StayHome* لیست شده در بخش ۴-۴-۴ ، ممکن است تصمیم بگیرید شاخصهای دیگری را که در شکل ۵-۱۳ نشان داده شده است را ایجاد کنید. این شکل ستونهای موجود هر جدول را که باید شاخص بندی شود نشان می دهد، و همچنین تراکنش(ها)یی که از ستونها استفاده می کند، و دلیل افزودن شاخص (یا بخاطر اینکه در موقعیت جستجو استفاده می شود، همچون ستون مرتب سازی، یا ستون گروه بندی) را نشان می دهد. ما پیاده سازی بعضی از این تراکنشها هم در اکسس QBE و SQl در فصل ۱۷ نشان می دهیم، بنابراین شما ممکن است بخواهید به این پیاده سازیها بعنوان یک بررسی متقابل نگاه کنید. شما می توانید تمرین مشابهی برای تراکنشهای موجود در دید تجاری را که در فصل ۱۰ مستند شده است را انجام دهید.

توجه کنید که ستون available جدول VideoForRent به عنوان موقعیت جستجو توسط تراکنش (ها) استفاده می شود. با این وجود، این ستون تنها می تواند بر دو مقدار انجام گیرد (Yیا N) و همچنین از راهنمایی (۹) بالا، شاخص بندی این ستون ارزش ندارد.

شکل ۵-۱۳

شاخصهای اضافی برای *StayHome*.

جدول	ستون	تراکنش	علت
Branch	city	(m)	search condition
Staff	name	(n)	ordering
Video	category	(p)	ordering
		(u)	search condition
		(v)	grouping
	title	(q), (r), (u)	ordering
		(t)	search condition
Actor	actorName	(q)	search condition
		(x)	grouping, ordering
Director	directorName	(r)	search condition
Member	fName/lName	(s)	search condition
RentalAgreement	dateReturn	(s)	search condition
Registration	dateJoined	(y)	search condition

خلاصه فصل

- ✓ در مرحله ۵، شما سازمانهای فایل مناسب برای ذخیره سازی جداول پایه را انتخاب کردید، و شاخص بندی برای رسیدن به کارایی قابل قبول نیاز می باشد. این شامل تحلیل تراکنشهایی که در پایگاه داده اجرا خواهد شد، و انتخاب سازمانهای فایل مناسب براساس این تحلیل، و افزودن شاخص ها می باشد.
- ✓ غیر ممکن است که قبل از اینکه تراکنشهایی که باید پشتیبانی شوند را با جزئیاتشان بررسی کنید تصمیمات طراحی فیزیکی با معنی بگیرید. این شامل تحلیل تراکنشهای مهم ، یعنی، تراکنشهایی که مکررا اجرا می شوند یا برای عمل حرفه حیاتی هستند می باشد.
- ✓ **فایلهای پشته** برای درج تعداد کثیری از رکوردها داخل فایل مناسب هستند. و زمانیکه تنها رکوردهای انتخابی بازیابی می شوند مناسب نیستند.
- ✓ **فایلهای درهم** زمانیکه بازیابی براساس تطبیق دقیق کلید باشد مناسب است. آنها زمانیکه بازیابی مبتنی بر تطبیق الگو، یا محدوده مقادیر ، کلیدهای قسمتی، یا وقتیکه بازیابی براساس ستونی غیر از فیلد درهم باشد مناسب نیستند.
- ✓ ISAM سلیس تر از درهم سازی است، بازیابی را براساس تطبیق دقیق کلید، تطبیق الگو، محدوده مقادیر، و خصوصیات بخشی کلید انجام می دهد. با این وجود، شاخص ISAM ثابت است و بنابراین کارایی زمان بهنگام شدن جدول کمتر می شود. بهنگام سازی همچنین باعث می شود فایل ISAM ترتیب کلید دستیابی را گم کند، بنابراین بازیابی ها در ترتیب کلید دستیابی آهسته تر است.
- ✓ این دو مشکل توسط سازمان فایل B^+-Tree برطرف می شود. که شاخص بندی پویا دارد. اگر جدول مکررا بهنگام نشود یا خیلی بزرگ نباشد، ساختار ISAM احتمالا موثرتر از زمانی باشد که یک سطح کمتر از شاخص B^+-Tree دارد، آنهایی که گره های برگ شان شامل اشاره گرهای رکورد می باشد.

- ✓ شاخص بندی ثانوی مکانیزمی جهت مشخص کردن کلید اضافی برای جدول پایه فراهم می آورد که می تواند برای بازیابی داده به صورت کاراتر استفاده شود. با این وجود، کلیاتی در نگهداری و استفاده از شاخصهای ثانوی که باید در برابر بهبودهای کارایی بدست آمده بهنگام بازیابی داده متعادل شوند.
- ✓ یک روش برای انتخاب سازمان فایل مناسب برای جدول نگه داشتن رکوردها بصورت نامرتب و ایجاد چندین شاخص ثانوی بهنگام نیاز است. روش دیگر مرتب کردن رکورد در جدول بوسیله مشخص کردن شاخص خوشه ساز یا اصلی می باشد.
- ✓ یک روش مشخص کردن اینکه کدام شاخص ثانوی را نیاز دارید تولید 'لیست خواسته ها' ستونهایی است که فکر می کنید برای شاخص بندی مناسب باشند، و بنابراین در نظرگرفتن اثر نگداری هر یک از این شاخصها می باشد.

طراحی فیزیکی پایگاه داده- مرحله ۶

آنچه در این فصل خواهید آموخت:

- چگونه داده مشتق شده را کنترل کنیم.
- در چه مواقعی برای بهبود کارایی نرمال سازی را دوباره انجام دهیم.

این فصل سومین مرحله از متدلوژی طراحی فیزیکی پایگاه داده ما را می پوشاند. در دو فصل قبلی، ما چگونگی تبدیل طراحی منطقی به مجموعه جداول پایه و انتخاب سازمانهای فایل و شاخصهای مناسب مبتنی بر تحلیل تراکنش هایی که پایگاه داده بایستی پشتیبانی کند را نشان دادیم. در بعضی از موارد، بهبودهای کارایی دیگری می تواند بوسیله کم کردن قواعد نرمال سازی بدست آید، آنچه که در این فصل مدنظر ما است.

مرحله ۶ مفهوم افزونگی کنترل شده را در نظر بگیرید

هدف

تصمیم گیری درباره اینکه چگونه داده مشتق شده را کنترل کرده و مشخص کنیم که آیا نمایش افزونگی به طریق کنترل شده توسط کم کردن قواعد نرمال سازی کارایی سیستم را بهبود خواهد بخشید.

نرمالسازی رویه ای برای تصمیم گیری درباره این است که کدام ستون ها در جدول متعلق به همدیگر هستند. یکی از اهداف اساسی طراحی پایگاه داده رابطه ای گروه بندی ستونها با یکدیگر در جدول است (**وابستگی تابعی** نامیده می شود) و آنهم بخاطر اینکه رابطه مستقیم بین آنها وجود دارد. نتیجه انجام نرمال سازی روی داده، طراحی منطقی پایگاه داده ای است که از لحاظ ساختار سازگار بوده و حداقل افزونگی را دارد.

با این وجود، طراحی پایگاه داده نرمال سازی شده بازده پردازش حداکثری را فراهم نمی آورد. در این شرایط، ممکن است بخواهید برای رسیدن به کارایی بهتر کمبود بعضی از منافع طراحی کاملا نرمال سازی شده را قبول کنید. شما این را تنها در صورتی در نظر می گیرید که تخمین زده باشید که سیستم قادر به مواجه شدن با نیازمندی های کارایی نخواهد بود.

ما طرفدار این نیستیم که نرمال سازی باید از طراحی منطقی پایگاه داده حذف شود: نرمال سازی شما را مجبور می کند تا هر ستون جدول موجود در پایگاه داده را کاملا شناخته و درک کنید. برعهده گرفتن این مرحله شاید مهم ترین فاکتور باشد تا باعث موفقیت کلی سیستم شود. اگر نرمال سازی دوباره را انجام دهید فاکتورهای زیرین را نیز باید در نظر بگیرید:

- نرمال سازی دوباره پیاده سازی را پیچیده تر می کند.
- نرمال سازی دوباره اغلب انعطاف پذیری را قربانی می کند.
- نرمال سازی دوباره بازیابی را سرعت می بخشد اما بهنگام سازی را کند می کند.

به طور رسمی، **نرمال سازی دوباره**[1] به تغییر ساختار جدول پایه، بدین صورت که جدول جدید در فرم نرمال پائین تر از جدول اصلی باشد اشاره دارد. با این وجود، ما همچنین از این عبارت آزادانه برای رجوع به موقعیت هایی که دو جدول را در یک جدول ادغام می کنیم، و جدول جدید در فرم نرمال مشابهی باشد اما شامل null های بیشتری نسبت به جدولهای اصلی باشد استفاده می کنیم.

نکته به عنوان یک قاعده سر انگشتی، اگر کارایی قابل قبول نباشد و جدول نرخ بهنگام سازی کمتر و نرخ پرس وجوی بالایی داشته باشد، نرمال سازی دوباره می تواند گزینه مناسبی باشد.

ماتریس ارجاع متقابل تراکنش/ جدول که در مرحله ۱-۵ تولید کرده اید اطلاعات مفیدی برای این مرحله فراهم می کند. این ماتریس، به شکل بصری، خلاصه شده است، که الگوهای دستیابی تراکنشهایی که در پایگاه داده اجرا خواهد شد را نشان می دهد. شما می توانید از آن ماتریس برای پر رنگ کردن کاندیدهای ممکن برای نرمالسازی دوباره، و همچنین جهت تعیین تاثیراتی که این بر سایر قسمتهای مدل خواهد داشت استفاده کنید.

دو فعالیت عمده این مرحله بدین صورت هستند:

- مرحله ۱-۶ داده مشتق شده را در نظر بگیرید
- مرحله ۲-۶ ستونهای تکراری یا الحاق جداول در همدیگر را در نظر بگیرید

مرحله ۱-۶ داده مشتق شده را در نظر بگیرید

هدف

تصمیم گیری درباره چگونگی اداره کردن داده مشتق شده.

ستونی که مقدارش را بتوان با بررسی مقادیر دیگر ستونها بدست آورد ستون *مشتق شده* یا *محاسباتی* نام دارد. برای مثال، همه موارد زیرین ستونهای مشتق شده هستند:

- تعداد پرسنلی که در شعبه مشخصی کار می کنند;
- مجموع حقوقهای ماهانه همه پرسنل شعبه مشخص;
- تعداد فیلمهایی که در حال حاضر در اجاره عضو معینی است;

[1] denormalization

همانطور که در مرحله ۳-۱ فصل ۸ یادآوری کردیم، ستونهای مشتق شده اغلب در مدل منطقی داده ظاهر نمی شوند، اما بجای آن در دیکشنری داده مستند می شوند. اگر ستون مشتق شده در مدل نشان داده شود، نام آن بهمراه علامت '/' جهت نشان دادن اینکه مشتق شده است نشان داده می شود. بنابراین، مرحله اول، بررسی مدل منطقی داده و دیکشنری داده، و تولید لیستی از تمام ستونهای مشتق می باشد.

از منظر طراحی فیزیکی پایگاه داده نیز، بدین صورت که آیا ستون مشتق شده در پایگاه داده ذخیره می شود یا هر زمانی که جهت بررسی لازم است محاسبه می شود یک امتیاز بحساب می آید. برای تصمیم گیری درباره نحوه محاسبه ستون محاسباتی، باید موارد زیر را در نظر بگیرید:

- هزینه اضافی جهت ذخیره داده مشتق شده و همزمان نگه داشتن آن با داده عملیاتی از اینکه کدام مشتق شده است، و
- هزینه جهت محاسبه آن هر زمان که مورد نیاز باشد،

و انتخاب گزینه کم هزینه مربوط به محدودیتهای کارایی می باشد. برای مثال آخر بالایی، شما می توانید ستون اضافی را در جدول Member ذخیره کنید که نشان دهنده تعداد اجاره هایی است که هر عضو در حال حاضر دارد. جدول RentalAgreement و جدول Member با ستون جدید مشتق شده در شکل ۱-۱۴ نشان داده شده است.

هزینه ذخیره سازی اضافی برای این ستون مشتق شده جدید مشخصا مهم نمی باشد. با این وجود، ستون noOfRentals هر زمان که فیلمی اجاره شود یا اینکه برگردانده شود بایستی بهنگام شود. شما عموما نیاز خواهید داشت مطمئن شوید که این تغییر به طور منظم برای نگهداری شمارش صحیح اعمال می شود، و بنابراین از جامعیت پایگاه داده مطمئن می شوید. بوسیله ذخیره داده به روش فوق، وقتیکه پرس و جو به این اطلاعات نیاز دارد، مقدار فورا در دسترس قرار می گیرد و نباید محاسبه شود.

از طرف دیگر، اگر ستون مستقیما در جدول Member ذخیره نشود، بایستی در صورت نیاز محاسبه شود. که این شامل الحاق جدولهای Member و RentalAgreement می باشد. برای مثال، برای محاسبه تعداد فیلمهایی که در حال حاضر در اجاره عضوی بنام 'Don Nelson' می باشد، شما بایستی از پرس وجوی SQL زیر استفاده کنید:

```
SELECT COUNT (*) AS noofrentals
FROM member m INNER JOIN rentalagreement ra
ON m.memberno = ra.memberno
WHERE m.fname = 'Don' AND m.lname = 'Nelson';
```

اگر این نوع از پرس وجو تکرار شونده باشد یا جهت هدفهای کارایی بحرانی در نظر گرفته شود، شاید مناسب تر باشد که ستون مشتق شده را بجای اینکه هر بار محاسبه شود ذخیره کنیم. در مثال ما، *StayHome* می خواهد که این نوع از پرس و جو ها را هر زمان که عضو می خواهد فیلم جدیدی اجاره کند اجرا کند. از طریق گفتگو با پرسنل *StayHome*، تخمین زده شد که اندازه جدول RentalAgreement در حدود ۴۰۰۰۰۰ رکورد است، با فرض اینکه رکوردها از RentalAgreement هر موقع که فیلمها برگردانده شده اند آرشیو شوند. بنابراین از آنجائیکه جدول RentalAgreement نسبتا بزرگ است و پرس و جو تکراری است، ممکن است تصمیم بگیرید که افزودن ستون مشتق شده به جدول Member موثرتر خواهد شد. پرس و جوی مشابه می تواند بصورت زیر نوشته شود:

```
SELECT noofrentals
FROM member m
WHERE m.fname = 'Don' AND m.lname = 'Nelson';
```

شکل ۱-۱۴

جدول RentalAgreement و جدول Member با ستون اضافی مشتق شده noOfRentals .

RentalAgreement

rentalNo	dateOut	dateReturn	memberNo	videoNo
R753461	4-Feb-00	6-Feb-00	M284354	245456
R753462	4-Feb-00	6-Feb-00	M284354	243431
R668256	5-Feb-00	7-Feb-00	M115656	199004
R668189	2-Feb-00	4-Feb-00	M115656	178643

Member

memberNo	fName	lName	address	noOfRentals
M250178	Bob	Adams	57 – 11th Avenue, Seattle, WA, 98105	0
M166884	Art	Peters	89 Redmond Rd, Portland, OR, 97117	0
M115656	Serena	Parker	22 W. Capital Way, Portland, OR, 97201	2
M284354	Don	Nelson	123 Suffolk Lane, Seattle, WA, 98117	2

نکته همچنین ممکن است مناسب تر باشد که ستونهای مشتق شده را در مواقعی که زبان پرس وجوی سیستم نتواند از عهده الگوریتم برای محاسبه ستون مشتق شده برآید ذخیره کنیم. برای مثال، SQL مجموعه محدودی از توابع تجمعی دارد و نمی تواند به آسانی پرس وجوهای بازگشتی را کنترل کند.

مرحله ۲-۶ ستونهای تکراری یا الحاق جداول در همدیگر را در نظر بگیرید

هدف

مشخص کردن اینکه آیا معرفی افزونگی در حالت کنترل شده بوسیله کم کردن قواعد نرمال سازی کارایی سیستم را بهبود خواهد بخشید.

مرحله بعد در نظرگرفتن تکرار ستونهای خاصی یا الحاق جدولها به یکدیگر جهت کاهش تعداد الحاقهای موردنیاز برای انجام پرس وجو است. به طور غیر مستقیم، شما زمانیکه با آدرسها سروکار دارید با مثال ضمنی نرمال سازی دوباره مواجه شده اید. برای مثال، تعریف جدول Branch را در نظر بگیرید:

Branch (branchNo, street, city, state, zipcode, mgrStaffNo)

اگر بخواهیم دقیق صحبت کنیم، این جدول در فرم نرمال سوم (3NF) نیست: zipCode بخوبی City و State را مشخص می کند; بعبارت دیگر اگر شما ZipCode را بدانید در اینصورت City و State را نیز می دانید. بنابراین برای نرمال سازی جدول ابتدا لازم است جدول را بدو قسمت مانند زیر تقسیم کنید:

Branch (branchNo, street, zipcode, mgrStaffNo)
ZipCode (zipCode, city, state)

بنابراین، شما بندرت می خواهید که به آدرس شعبه بدون ستونهای City و State دستیابی پیدا کنید. این بدین معنی است که می خواهید الحاق را تنها زمانیکه آدرس کامل شعبه را خواستید انجام دهید. در نتیجه، ما معمولا جدول اصلی Branch را پیاده سازی می کنیم و جهت فرم نرمال دوم (2NF) قرار می دهیم.

متاسفانه هیچ قاعده ثابتی برای مشخص کردن اینکه چه موقع جدولها را دوباره نرمال سازی کنیم وجود ندارد. با این وجود اجازه دهید برخی از عمومی ترین موقعیتهایی را که برای نرمال سازی دوباره جهت سرعت بخشیدن به تراکنشهای بحرانی و تکراری مورد استفاده قرار می گیرند را بحث کنیم:

مرحله ۱-۲-۶ رابطه های یک به یک (1:1) را ترکیب کنید
مرحله ۲-۲-۶ ستونهای غیر کلید را جهت کاهش الحاقها در رابطه های یک به چند (*:1) تکرار کنید
مرحله ۳-۲-۶ ستونهای کلید خارجی را جهت کاهش الحاقها در رابطه های یک به چند (*:1) تکرار کنید
مرحله ۴-۲-۶ ستونها را جهت کاهش الحاقها در رابطه های چند به چند (*:*) تکرار کنید
مرحله ۵-۲-۶ گروههای تکرار شده را نشان دهید
مرحله ۶-۲-۶ جدولهای look-up را با جدولهای پایه ادغام کنید
مرحله ۷-۲-۶ جدولهای خلاصه را ایجاد کنید

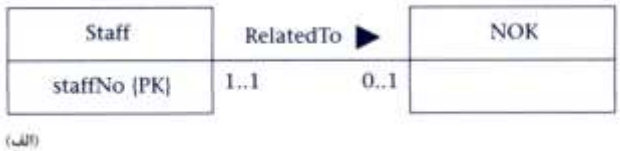
مرحله ۱-۲-٦ رابطه های یک به یک (1:1) را ادغام کنید

رابطه های یک به یک را دوباره بررسی کنید تا تاثیر ترکیب جدولها بصورت جدول واحد را مشخص سازید. شما باید این بررسی را زمانی در نظر بگیرید که جدولها مکررا یا بصورت مجزا بندرت بهم رجوع می کنند. اجازه دهید رابطه یک به یک بین Staff و NOK را در نظر بگیریم، که در شکل ۲-۱۴(الف) نشان داده شده است. موجودیت Staff شامل اطلاعات پرسنل و موجودیت NOK شامل اطلاعاتی درباره خویشاوندان نزدیک پرسنل می باشد. جدولهای نهایی در شکل ۲-۱۴(ب) نشان داده شده است.

رابطه بین Staff و NOK یک به یک است و اشتراکشان اختیاری است. زمانیکه اشتراک اختیاری باشد، در صورت ادغام دو جدول با همدیگر تعداد ستونهایی که دارای Null هستند و برای بعضی از رکوردها ظاهر می شوند، در شکل ۳-۱۴ نشان داده شده است. اگر جدول Staff بزرگ باشد و سهم رکوردهای شامل شده در اشتراک کوچک باشد، مقدار قابل توجهی فضای هرز وجود خواهد داشت. مقدار اتلاف در برابر هر بهبود کارایی حاصل شده نتیجه ادغام جدولها متعادل می شود.

شکل ۲-۱۴

Staff و NOK: (الف) مدل ER اصلی; (ب) جداول اصلی.



(الف)

Staff

staffNo	name	position	salary	branchNo
S1500	Tom Daniels	Manager	46000	B001
S0003	Sally Adams	Assistant	30000	B001
S0010	Mary Martinez	Manager	50000	B002
S3250	Robert Chin	Supervisor	32000	B002
S2250	Sally Stern	Manager	48000	B004
S0415	Art Peters	Snr Assistant	41000	B003

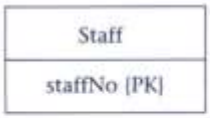
NOK

staffNo	nokName	nokTelNo
S1500	Jane Daniels	207-878-2751
S0003	John Adams	518-474-5355
S3250	Michelle Chin	206-655-9867
S0415	Amy Peters	718-507-7923

(ب)

شکل ۳-۱۴

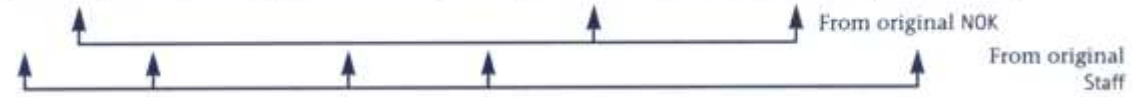
ادغام کردن Staff و NOK: (الف) مدل ER اصلاح شده; (ب) جداول ادغام شده .



(الف)

Staff

staffNo	name	position	salary	nokName	nokTelNo	branchNo
S1500	Tom Daniels	Manager	46000	Jane Daniels	207-878-2751	B001
S0003	Sally Adams	Assistant	30000	John Adams	518-474-5355	B001
S0010	Mary Martinez	Manager	50000			B002
S3250	Robert Chin	Supervisor	32000	Michelle Chin	206-655-9867	B002
S2250	Sally Stern	Manager	48000			B004
S0415	Art Peters	Snr Assistant	41000	Amy Peters	718-507-7923	B003



(ب)

مرحله ۲-۲-۶ ستونهای غیر کلید را در رابطه های یک به چند جهت کاهش الحاقها تکرار کنید

شما باید با هدف کاهش یا حذف الحاقها از پرس وجو های بحرانی یا تکراری، در رابطه های یک به چند امتیازاتی را که از تکرار یک یا چند ستون غیرکلید جدول والد در جدول فرزند حاصل می شود را در نظر بگیرید. برای مثال، وقتیکه جدول VideoForRent دستیابی می شود، بسیار متداول است که نرخ اجاره روزانه فیلم ها در زمان مشابه مورد دستیابی قرارگیرند. پرس وجوی SQL نمونه بصورت زیر خواهد بود:

```
SELECT vfr.*, v.dailyrental
FROM videoforrent vfr INNER JOIN video v
ON vfr.catalogno = v.catalogno
WHERE branchno = 'B001';
```

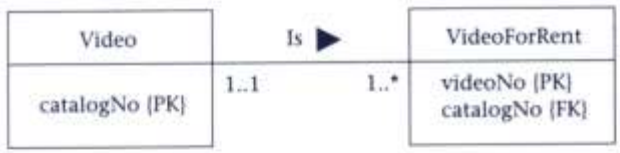
که بر اساس مدل اصلی ER و جدولهای نتیجه نشان داده شده در شکل ۴-۱۴ می باشد.
اگر شما ستون dailyRental را در جدول VideoForRent تکرار کنید، می توانید جدول Video را از پرس وجو بردارید که حالا در SQL بدین صورت خواهد شد:

```
SELECT vfr.*
FROM   videoforrent  vfr
WHERE  branchno = 'B001';
```

که بر اساس جدول اصلاح شده VideoForRent نشان داده شده در شکل ۵-۱۴ خواهد شد. سودی که از این تغییرحاصل می شود در برابر مشکلاتی که احتمال دارد بوجود آید متعادل می شود. برای مثال، اگر شما داده تکراری را در جدول والد تغییر دهید، آن داده را باید در جدول فرزند نیز بروز کنید. بعلاوه، برای رابطه یک به چند احتمالا برای هر قلم داده چندین رخداد در جدول فرزند وجود دارد. (برای مثال فیلم های ۲۰۷۱۳۲ و۶۳۴۸۱۷ با نرخهای اجاره ۵,۰۰ و۴,۵۰ بترتیب، هر دو در جدول اصلاح شده VideoForRent ظاهر می شوند). بنابراین، شما همچنین مجبور به حفظ کردن همزمانی چندین کپی

شکل ۴-۱۴

Video و VideoForRent: (الف) مدل ER اصلی; (ب) جداول اصلی.



(الف)

Video

catalogNo	title	category	dailyRental	price	directorNo
207132	Tomorrow Never Dies	Action	5.00	21.99	D1001
902355	Primary Colors	Comedy	4.50	14.50	D7834
330553	Face/Off	Thriller	5.00	31.99	D4576
781132	101 Dalmatians	Children	4.00	18.50	D0078
445624	The Rock	Action	4.00	29.99	D5743
634817	Independence Day	Sci-Fi	4.50	32.99	D3765

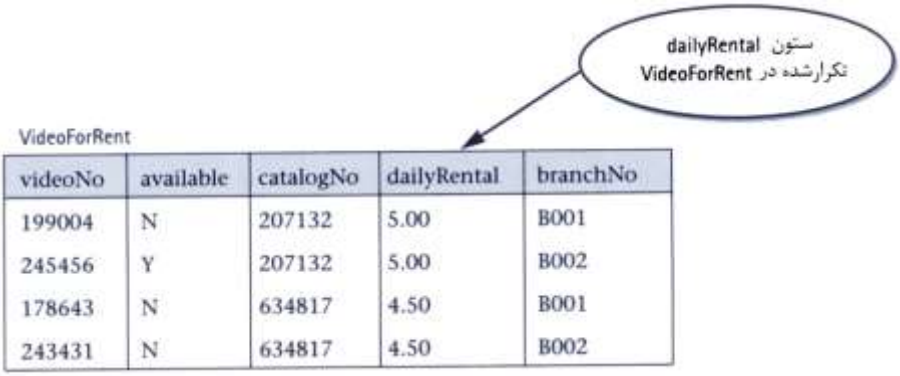
VideoForRent

videoNo	available	catalogNo	branchNo
199004	N	207132	B001
245456	Y	207132	B002
178643	N	634817	B001
243431	N	634817	B002

(ب)

شکل ۵-۱۴

تکرار ستون dailyRental در جدول VideoForRent.



VideoForRent

videoNo	available	catalogNo	dailyRental	branchNo
199004	N	207132	5.00	B001
245456	Y	207132	5.00	B002
178643	N	634817	4.50	B001
243431	N	634817	4.50	B002

موجود هستید. اگر بهنگام سازی ستون dailyRental در جدول Video و VideoForRent نتواند خودکار شود، عاملی برای اتلاف جامعیت قابل توجه است. حتی اگر این فرایند بتواند خودکار شود، هر زمان که رکوردی درج، بهنگام، یا حذف شود زمان اضافی برای حفظ همزمانی لازم خواهد بود. در مورد ما، احتمال دارد که نرخ اجاره روزانه زمانی که فیلم قدیمی تر شود کاهش یابد، بنابراین تکرار شاید یک امر تضمین نشده ای باشد.

نوعی دیگری از مشکلات که بایستی لحاظ شود افزایش در فضای ذخیره سازی ناشی از تکرار است. دوباره در این مورد نیز، با هزینه نسبتا کمی که وسایل ذخیره سازی جانبی امروزه دارند، شاید این مشکل قابل توجهی نباشد. با این وجود توجیهی برای تکرار اختیاری نیست.

مرحله ۳-۲-۶ تکرار ستونهای کلید خارجی در رابطه های یک به چند برای کاهش الحاقها

دوباره، با هدف ویژه ای از کاهش یا حذف الحاقها از پرس وجوهای تکراری یا بحرانی شما بایستی سودی که شاید نتیجه تکرار یک یا چند ستون کلید خارجی باشند را در نظر بگیرید. برای مثال، پرس وجوی مکرری که برای *StayHome* بکار می رود لیست کردن تمام توافقنامه های اجاره مربوط به شعبه است، که با استفاده از پرس وجوی SQL زیر انجام می گیرد:

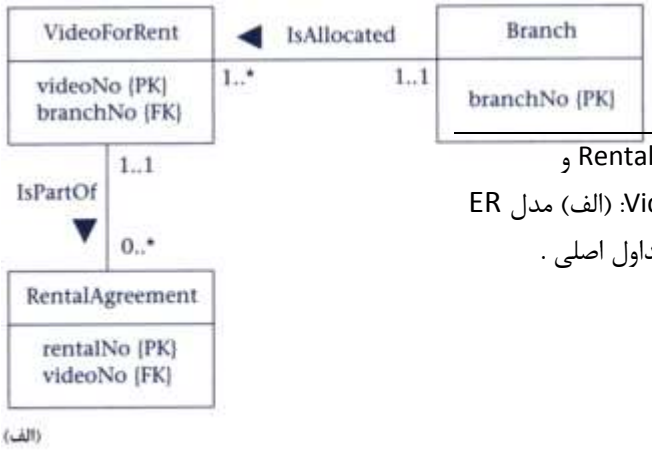
```
SELECT ra.*
FROM rentalagreement ra INNER JOIN videoforrent vfr
ON ra.videono = vfr.videono
WHERE vfr.branchno = 'B001';
```

مبتنی بر مدل ER اصلی و جدولهایی که در شکل ۶-۱۴ نشان داده شده است.

همانطورکه از این پرس وجو مشاهده می شود، برای بدست آوردن توافقنامه های اجاره مجبورید از جدول VideoForRent جهت دسترسی به شماره شعبه مورد نیاز ، branchNo، استفاده کنید. شما می توانید نیاز به این الحاق را با تکرار کلید خارجی branchNo در جدول RentalAgreement حذف کنید; یعنی، رابطه مستقیمی بین جدولهای Branch و RentalAgreement را نشان دهید. دراین مورد، شما می توانید پرس وجوی SQL زیر را بصورت ساده زیر بنویسید:

```
SELECT *
FROM RentalAgreement
WHERE branchno = 'B001';
```

مبتنی بر مدل ER اصلاح شده و جدول RentalAgreement که در شکل ۷-۱۴ نشان داده شده است . اگر این تغییر اعمال شود، لازم است محدودیتهای کلید خارجی اضافی را ، همچنان که در مرحله ۳-۲ بحث شد معرفی کنیم.



(الف)

RentalAgreement

rentalNo	dateOut	dateReturn	memberNo	videoNo
R753461	4-Feb-00	6-Feb-00	M284354	245456
R753462	4-Feb-00	6-Feb-00	M284354	243431
R668256	5-Feb-00	7-Feb-00	M115656	199004
R668189	2-Feb-00	4-Feb-00	M115656	178643

VideoForRent

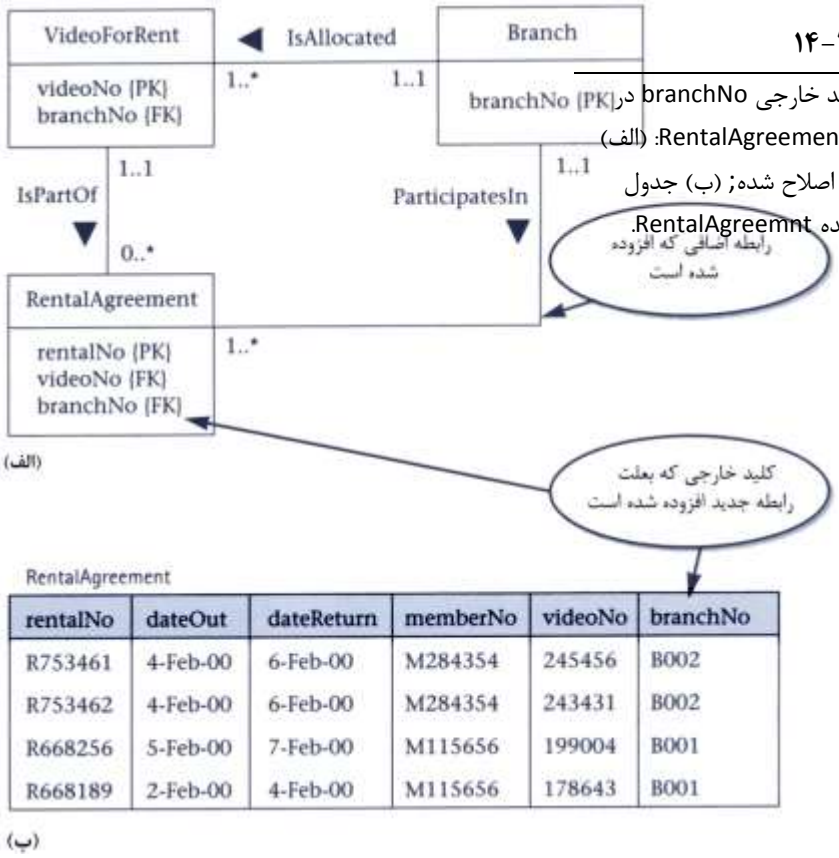
videoNo	available	catalogNo	branchNo
199004	N	207132	B001
245456	Y	207132	B002
178643	N	634817	B001
243431	N	634817	B002

شکل ۶-۱۴ VideoForRent: (الف) مدل ER و RentalAgreement اصلی; (ب) جداول اصلی .

توجه کنید که این به این علت درست است که رابطه جدید بین Branch و RentalAgreement ۱:* است. بعبارت دیگر، برای هر توافقنامه اجاره یک و تنها یک شعبه مربوطه وجود دارد. اگر رابطه *:* بود، تغییر بالایی کار نمی کرد. برای مثال، پرس وجوی تکراری دیگری ممکن است عناوین فیلمهای موجود در انبار شعبه را با استفاده از پرس و جوی SQL زیر لیست کند:

```
SELECT v.title
FROM video v INNER JOIN videoforrent vfr
ON v.catalogno = vfr.catalogno
WHERE vfr.branchno = 'B001';
```

از آنجائیکه رابطه بین Branch و Video *:* است این پرس وجو نمی تواند با افزودن ستون branchNo به جدول Video ساده شود; یعنی عنوان فیلم توسط تعدادی از شعبه ها موجود می باشد، و شعبه می تواند چندین عنوان فیلم داشته باشد. با این وجود، در این مورد می توانید تکرار ستون title جدول Video را در جدول VideoForRent در نظر بگیرید. اگرچه حافظه جانبی ذخیره سازی که افزایش یافته است در این مورد میتواند اهمیت بیشتری داشته باشد.



RentalAgreement

rentalNo	dateOut	dateReturn	memberNo	videoNo	branchNo
R753461	4-Feb-00	6-Feb-00	M284354	245456	B002
R753462	4-Feb-00	6-Feb-00	M284354	243431	B002
R668256	5-Feb-00	7-Feb-00	M115656	199004	B001
R668189	2-Feb-00	4-Feb-00	M115656	178643	B001

(ب)

شکل ۷-۱۴

تکرار کلید خارجی branchNo در جدول RentalAgreement: (الف) مدل ER اصلاح شده; (ب) جدول اصلاح شده RentalAgreemnt.

مرحله ٤-٢-٦ ستونهای تکراری در رابطه های چند به چند جهت کاهش الحاقها

در مرحله ۷-۱، شما همه رابطه های چند به چند را به دو رابطه یک به چند تبدیل کردید. این تبدیل با موجودیت میانی سومی نشان داده می شود. حال، اگر بخواهید اطلاعاتی را از رابطه چند به چند تولید کنید، مجبور هستید این سه جدول را با هم الحاق کنید: که دو جدول مشتق شده از موجودیت اصلی و جدول جدید بیانگر رابطه بین دو موجودیت می باشد. در بعضی از شرایط، ممکن است بتوانید تعداد جدولهای الحاقی بوسیله تکرار ستونهای یکی از موجودیتهای اصلی در جدول میانی را کاهش دهید.

برای مثال، بین Video و Actor رابطه چند به چند وجود دارد، که در این رابطه Role بعنوان موجودیت میانی عمل می کند. پرس وجویی را که عناوین فیلمها و نقشها را برای بازیگری که در آن نقش بازی کرده است لیست می کند را در نظر بگیرید:

```
SELECT   v.title, a.*, r.*
FROM video v INNER JOIN (role r INNER JOIN actor a
ON r.actorno = a. actorno) ON v.catalogno = r.catalogno;
```

بر اساس مدل ER و جدولهای نشان داده شده در شکل ۸-۱۴.

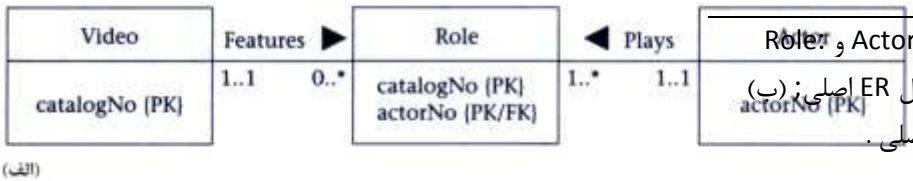
اگر ستون title را در جدول Role تکرار کنید می توانید جدول Video را از پرس وجو حذف کنید، که پرس وجوی SQL اصلاح شده زیرین آن را نتیجه می دهد:

```
SELECT  a.* , r.*
FROM  role r INNER JOIN actor a ON r.actorno = a.actorno;
```

بر اساس جدول Role اصلاح شده نشان داده شده در شکل ۹-۱۴.

شکل ۸-۱۴

Video، Actor و Role: (الف) مدل ER اصلی؛ (ب) جداول اصلی.



(الف)

Video

catalogNo	title	category	dailyRental	price	directorNo
207132	Tomorrow Never Dies	Action	5.00	21.99	D1001
902355	Primary Colors	Comedy	4.50	14.50	D7834
330553	Face/Off	Thriller	5.00	31.99	D4576
781132	101 Dalmatians	Children	4.00	18.50	D0078
445624	The Rock	Action	4.00	29.99	D5743
634817	Independence Day	Sci-Fi	4.50	32.99	D3765

Actor

actorNo	actorName
A1002	Pierce Brosnan
A3006	John Travolta
A2019	Nicolas Cage
A7525	Will Smith
A4343	Glenn Close

Role

actorNo	catalogNo	character
A1002	207132	James Bond
A3006	330553	Sean Archer
A3006	902355	Jack Stanton
A2019	330553	Castor Troy
A2019	445624	Stanley Goodspeed
A7525	634817	Captain Steve Hiller
A4343	781132	Cruella De Vil

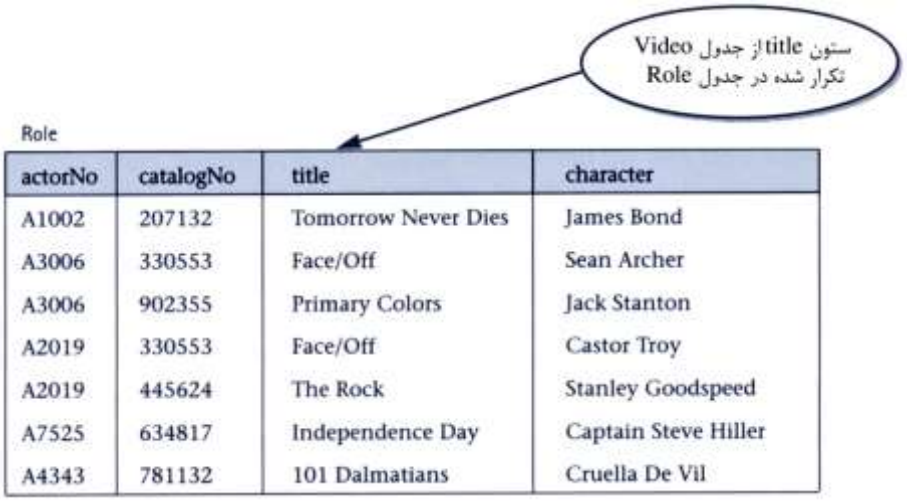
(ب)

مرحله ۵-۲-۶ معرفی کردن گروههای تکرار

گروههای تکرار که از مدل منطقی داده تخمین زده می شود نتیجه نیازمندی هایی است که در آن باید تمام موجودیتها در فرم نرمال اول (1NF) باشند. گروههای تکرار در جدول جدید جدا شده اند، که رابطه یک به چندی را با جدول اصلی (والد) تشکیل می دهند. اغلب اوقات معرفی دوباره گروهای تکرار راه موثری جهت بهبود کارایی سیستم است.

برای مثال، هر دفتر شعبه *StayHome* حداکثر سه شماره تلفن دارد، اگرچه الزاما همه دفاتر شماره خطهای مشابهی ندارند. در مدل منطقی داده، شما موجودیت Telephone با رابطه سه به یک با Branch ایجاد کردید. این دو موجودیت در دو جدول شکل ۱۰-۱۴ نشان داده شده است.

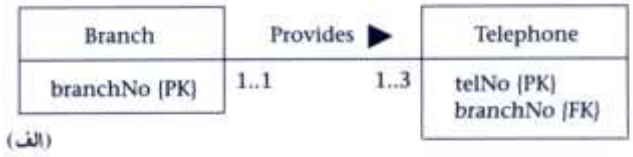
اگر دسترسی به این اطلاعات مهم و مکررا صورت گیرد شاید موثرتر باشد که این جدولها را با هم ادغام کرده و جزئیات تلفن را با یک ستون برای هر شماره تلفن، در جدول Branch ذخیره کنیم، که در شکل ۱۱-۱۴ نشان داده شده است.



actorNo	catalogNo	title	character
A1002	207132	Tomorrow Never Dies	James Bond
A3006	330553	Face/Off	Sean Archer
A3006	902355	Primary Colors	Jack Stanton
A2019	330553	Face/Off	Castor Troy
A2019	445624	The Rock	Stanley Goodspeed
A7525	634817	Independence Day	Captain Steve Hiller
A4343	781132	101 Dalmatians	Cruella De Vil

شکل ۹-۱۴

تکرار ستون title از جدول Video در جدول Role.



(الف)

Branch

branchNo	street	city	state	zipCode	mgrStaffNo
B001	8 Jefferson Way	Portland	OR	97201	S1500
B002	City Center Plaza	Seattle	WA	98122	S0010
B003	14 – 8th Avenue	New York	NY	10012	S0415
B004	16 – 14th Avenue	Seattle	WA	98128	S2250

Telephone

telNo	branchNo
503-555-3618	B001
503-555-2727	B001
503-555-6534	B001
206-555-6756	B002
206-555-8836	B002
212-371-3000	B003
206-555-3131	B004
206-555-4112	B004

(ب)

شکل ۱۰-۱۴

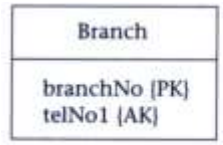
Branch و Telephone: (الف) مدل ER اصلی; (ب) جداول اصلی.

بطور کلی، شما باید این نوع از نرمال سازی دوباره را تنها تحت شرایط زیرین لحاظ کنید:

- تعداد مطلقی از اقلام در گروه های تکرار مشخص شده باشد. (در این مثال، حداکثر سه شماره تلفن وجود دارد).
- تعداد ثابت بوده و به مرور زمان تغییر نخواهد کرد (حداکثر تعداد خطوط تلفن شعبه ثابت بوده و انتظار تغییر در آن وجود ندارد).
- تعداد خیلی بزرگ نیستند، به طور نمونه بزرگتر از ۱۰ نیست، اگرچه این مانند دو شرط اول مهم نیست.

شکل ۱۱-۱۴

Branch گروه تکرار را ترکیب می کند: (الف) مدل ER اصلاح شده; (ب) جداول اصلاح شده.



(الف)

Branch

branchNo	street	...	telNo1	telNo2	telNo3	mgrStaffNo
B001	8 Jefferson Way	...	503-555-3618	503-555-2727	503-555-6534	S1500
B002	City Center Plaza	...	206-555-6756	206-555-8836		S0010
B003	14 – 8th Avenue	...	212-371-3000			S0415
B004	16 – 14th Avenue	...	206-555-3131	206-555-4112		S2250

from original Telephone table

(ب)

بعضی اوقات، شاید تنها مقدار جاری و اخیر در گروه تکرار باشد، یا تنها این حقیقت که گروه تکراری وجود دارد که مکررا مورد نیاز است. در مثال بالا، ممکن است که این انتخاب را داشته باشید که یک شماره تلفن را در جدول Branch ذخیره کرده و بقیه شماره ها را در جدول Telephone نگه دارید. از آنجاییکه هر شعبه بایستی حداقل یک شماره تلفن داشته باشد، این شرط وجود Null ها را از جدول Branch حذف می کند.

مرحله ۶-۲-۶ جدولهای جستجو را با جداول پایه ادغام کنید

جدولهای جستجوگر[۱]، بعضی وقتها جدولهای مرجع یا لیستهای انتخاب[۲] نامیده می شوند، که مورد ویژه ای از رابطه های یک به چند (*:۱) هستند. خصوصا برای مثال، جدولهای جستجوگر شامل کد و توصیف هستند. مثلا، ممکن است جدول جستجوگری برای فهرست فیلم تعریف کنید و جدول Video را تغییر دهید، چنانکه در شکل ۱۲-۱۴ نشان داده شده است.

امتیازات استفاده از جدولهای جستجوگر بدین صورت است:

- کاهش در اندازه جدول فرزند (در این مورد جدول Video); کد فهرست ۱ بایت را اشغال می کند و در برابر آن برای توصیف فهرست ۸ بایت داریم.
- اگر توصیف بتواند تغییر کند بجای اینکه در جدول فرزند (Video) آن را چندین بار تغییر دهیم ابتدا بهتر است که در جدول جستجوگر آن را تغییر دهیم (VideoCategory).
- جدول جستجوگر می تواند جهت معتبر سازی ورودی کاربر استفاده شود.

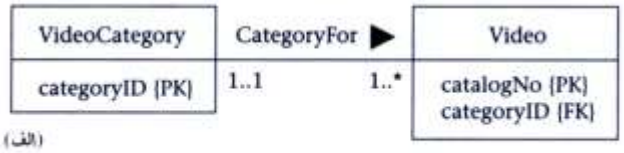
اگر جدولهای جستجوگر در پرس و جوهای بحرانی و تکراری استفاده شوند، و احتمال تغییر توصیفات وجود نداشته باشد شما باید تکرار ستون description جدول فرزند را، که در شکل ۱۳-۱۴ نشان داده شده است را در نظر بگیرید. جدول جستجوگر اصلی دارای افزونگی نیست – و هنوز می تواند برای معتبر سازی ورودی کاربر بکار رود.

---

[۱] Lookup tables
[۲] Pick list

شکل ۱۲-۱۴

جدول جستجوگر برای فهرست فیلم: (الف) مدل ER اصلی; (ب) جداول اصلی.



(الف)

VideoCategory

categoryID	description
1	Action
2	Comedy
3	Children
4	Sci-Fi
5	Thriller

Video

catalogNo	title	categoryID	dailyRental	price	directorNo
207132	Tomorrow Never Dies	1	5.00	21.99	D1001
902355	Primary Colors	2	4.50	14.50	D7834
330553	Face/Off	5	5.00	31.99	D4576
781132	101 Dalmatians	3	4.00	18.50	D0078
445624	The Rock	1	4.00	29.99	D5743
634817	Independence Day	4	4.50	32.99	D3765

lookup list

(ب)

شکل ۱۳-۱۴

جدول Video تغییریافته با ستون تکرارشده description.

Video

catalogNo	title	categoryID	description	dailyRental	price	directorNo
207132	Tomorrow Never Dies	1	Action	5.00	21.99	D1001
902355	Primary Colors	2	Comedy	4.50	14.50	D7834
330553	Face/Off	5	Thriller	5.00	31.99	D4576
781132	101 Dalmatians	3	Children	4.00	18.50	D0078
445624	The Rock	1	Action	4.00	29.99	D5743
634817	Independence Day	4	Sci-Fi	4.50	32.99	D3765

بنابراین، با تکرار ستون description در جدول فرزند، در واقع تخمین زده اید که نیاز به یک الحاق جدول فرزند با جدول lookup دارید.

مرحله ۷-۲-٦ ایجاد جدولهای خلاصه

در طول روز شاید موقعیتهایی بوجود آید که به گزارشهای مشخصی نیاز پیدا کنید. این گزارشها به داده مشتق شده دسترسی داشته و الحاقهای چند جدولی را بر روی مجموعه همسانی از جداول پایه انجام می دهد. بنابراین، داده و گزارش مبتنی بر آن شاید ایستا باشد یا، در بعضی موارد بصورت جاری نباشد (یعنی اگر داده مربوط به ساعات قبلی باشد، گزارش کاملا قابل قبول خواهد بود). در این مورد بهتر است جدول خلاصه منفردی را که دوباره نرمال سازی شده است را براساس جدولهای موردنیاز توسط گزارش ایجاد کنیم، و به کاربران بجای دسترسی مستقیم به جداول پایه اجازه دسترسی به جداول خلاصه را بدهیم. تکنیک رایج برای ایجاد جداول خلاصه، ایجاد و برقراری جداول پایه بهنگام ساعات آخر شب می باشد.

مفاهیم نرمالساز ی دوباره را در نظر بگیرید

شما هم اکنون باید مفاهیم نرمال سازی دوباره مراحل قبلی متدلوژی را در نظر بگیرید. برای مثال، ممکن است مجبور باشید که انتخاب شاخصها بر روی جدولهای دوباره نرمالسازی شده برای بررسی اینکه آیا شاخصهای موجود بایستی حذف شوند و یا اینکه شاخصهای دیگری نیز باید افزوده شوند را دوباره در نظر بگیرید.

مفهوم افزونگی را مستند کنید

افزونگی های موجود بایستی کاملا بهمراه دلایل معرفی آنها مستند شوند. خصوصا دلایل انتخاب یک روش را هنگامی که چندین روش دیگر نیز موجود هستند را نیز مستند کنید. مدل منطقی داده و مستندات پشتیبانی کننده را برای اعمال هر نوع تغییر انجام شده بر روی نتایج نرمال سازی دوباره را بهنگام سازید.

خلاصه فصل

- ✓ در مرحله ۶، شما مفهوم افزونگی کنترل شده را برای بهبود کارایی در نظر گرفتید.

- ✓ شاید شرایطی وجود داشته باشد که لازم باشد اتلاف بعضی از سودمندی های طراحی کاملا نرمال ساز ی شده را جهت توجه به کارایی قبول کنید. نرمال سازی دوباره تنها زمانیکه تخمین زده شده است که سیستم قادر به برآورده سازی نیازمندی های کارایی نیست باید در نظر گرفته شود. بعنوان یک قاعده سر انگشتی، اگر کارایی ناکافی باشد و جدول نرخ بهنگام شدن پایین و نرخ پرس وجوی بالایی داشته باشد نرمال سازی دوباره می تواند صورت گیرد.

- ✓ از منظر طراحی فیزیکی پایگاه داده، خواه ستون مشتق شده در پایگاه داده ذخیره شده باشد یا هر زمانی که موردنیاز باشد محاسبه شود بعنوان یک امتیاز محسوب می شود. هزینه اضافی را برای ذخیره داده مشتق شده در نظر بگیرید و آن را بصورت سازگار نگه دارید بهتر است که هر زمان که داده مورد نیاز باشد آن را محاسبه کنید و گزینه ارزان تری را انتخاب کنید.
- ✓ نرمال سازی دوباره را در موقعیتهای زیر در نظر بگیرید، خصوصا برای سرعت بخشیدن به تراکنشهای بحرانی یا تکراری: ادغام رابطه های یک به یک; تکراری کردن ستونهای غیر کلید در رابطه های یک به چند برای کاهش الحاقها; تکرار ستونهای کلید خارجی در رابطه های یک به چند برای کاهش الحاقها; تکرار ستونها در رابطه های چند به چند برای کاهش تکرارها ; معرفی گروهای تکرار ; ادغام جدولهای جستجوگر با جداول اصلی; ایجاد جداول خلاصه.

فصل ۱۵

طراحی فیزیکی پایگاه داده- مرحله ۷

آنچه در این فصل خواهید آموخت:

- پایگاه داده نشان دهنده منبع اساسی سازمان می باشد که باید امن ساخته شود.
- چگونه مکانیزمهای امنیت را جهت برآورده ساختن نیازمندیهای کاربر طراحی کنیم.

این فصل چهارمین مرحله از متدلوژی طراحی فیزیکی پایگاه داده ما را می پوشاند. در سه فصل قبلی، بر اساس تحلیل تراکنشهای مهم، طراحی منطقی را به مجموعه ساده ای از جدولها، سازمان فایلهای انتخابی مناسب و شاخصها برگرداندیم، و سپس مفهوم افزونگی کنترل شده را برای رسیدن به بهبودهای کارایی اضافی در نظرگرفتیم.

در این مرحله چگونگی امن ساختن پایگاه داده را بحث می کنیم. همچون دیگر مراحل طراحی فیزیکی پایگاه داده، پیاده سازی مکانیزمهای امنیت وابسته بر DBMS هدف خواهد بود.

مرحله ۷ مکانیزمهای امنیت را طراحی کنید

هدف

طراحی معیارهای امنیت برای پایگاه داده تعیین شده بوسیله کاربران.

پایگاه داده منابع ضروری شرکت را نشان می دهد، بنابراین امنیت این منبع بسیار مهم است. شاید نیازمندیهای امنیت بخصوصی طی مرحله تحلیل و جمع آوری نیازمندیهای چرخه حیات کاربرد پایگاه داده وجود داشته باشد. هدف این مرحله این است که تصمیم بگیریم چگونه این معیارهای امنیت تحقق خواهند یافت. بعضی سیستمها نسبت به دیگر سیستمها امکانات امنیت متفاوتی را پیشنهاد می کنند. شما بایستی دوباره از امکانات پیشنهادی DBMS هدف آگاه باشید. دو فعالیت عمده مربوط به این مرحله بصورت زیر هستند:

- مرحله ۱-۷ دیدهای کاربر را طراحی کنید
- مرحله ۲-۷ قواعد دستیابی را طراحی کنید

مرحله ۱-۷ دیدهای کاربر را طراحی کنید

هدف

طراحی دیدهای کاربری که در مرحله تحلیل چرخه حیات کاربرد پایگاه داده رابطه ای شناسایی شده است.

همان گونه که بیاد داریم مرحله اول متدلوژی طراحی پایگاه داده، ایجاد *مدل داده منطقی محلی* دیدهایی بود که طی مرحله تحلیل پایگاه داده شناسایی شده بودند. هر دید شامل یک یا چندین *دید کاربر* می باشد. در مرحله ۴-۴-۴ ما دو دید را برای *StayHome* مشخص کردیم:

- Branch، که شامل دیدهای کاربر Manager، Supervisor ، و Assistant است.
- Business، که شامل دیدهای کاربر Buyer و Director است.

در مرحله۳، این مدلهای منطقی محلی داخل یک مدل داده منطقی سراسری ادغام شدند. هدف این مرحله طراحی همه دیدهای کاربر مشخص شده قبلی می باشد. در DBMS مستقل کامپیوتر شخصی، دیدها معمولا راحت بوده، و برای پرس و جوی های ساده ای تعریف شده اند. با این وجود، در DBMS های چند کاربره دیدها نقش محوری را در تعریف ساختار پایگاه داده و اجرای امنیت ایفا می کنند. همانند طراحی جداول پایه که در فصل ۱۲ بحث شد، برای مشخص کردن این فرایند ما دو راه ویژه برای ایجاد دیدها را با استفاده از موارد زیر نشان می دهیم:

(۱) ISO SQL استاندارد ۱۹۹۲ (SQL2)،

(۲) مایکروسافت اکسس ۹۷.

ISO SQL استاندارد ۱۹۹۲ (SQL2)

معمولا، دیدها با استفاده از تسهیلاتی همچون SQL یا QBE ایجاد می شوند. برای مثال برای سرپرستها و معاونین شعبه B001 ممکن است بخواهید دیدی را برای جدول پایه Staff ایجاد کنید که اطلاعات حقوق پرسنل در آن وجو نداشته باشد. دستور SQL برای ایجاد چنین دیدی بدین صورت خواهد شد:

```
CREATE VIEW staff1_view
AS     SELECT staffno, name, position
                          FROM staff
              WHERE branchno = 'B001';
```

این دستور دیدی بنام Staff1_View با همه ستونهای جدول Staff بجز ستونهای branchNo وsalary ایجاد می کند. اگر اطلاعات این دید را لیست کنید بایستی داده های نشان داده شده در شکل ۱-۱۵ را بدست آورید.

شکل ۱-۱۵

فهرست دید Staff1_View.

Staff1_View

staffNo	name	position
S1500	Tom Daniels	Manager
S0003	Sally Adams	Assistant

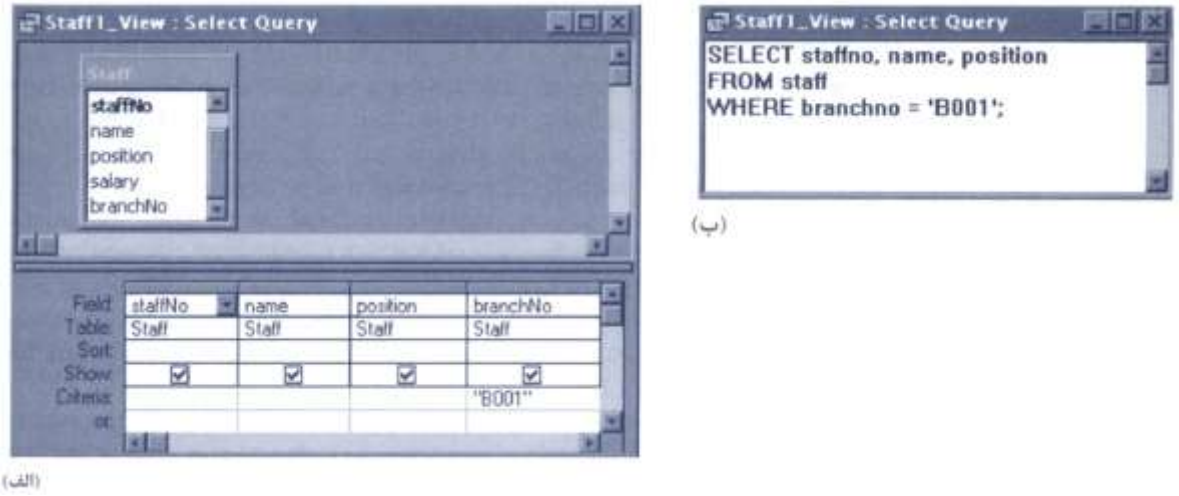
برای اطمینان از اینکه تنها مدیر شعبه می تواند ستون salary را ببیند، و سرپرست ها و معاونین نباید به جدول پایه Staff دسترسی داشته باشند بایستی چاره ای بیاندیشیم. یک راه حل این است که آنها بایستی امتیاز دسترسی معینی را برای دید Staff1_View بدهند، بنابراین از دسترسی آنها به داده حساس حقوق جلوگیری بعمل آورده شود. امتیازات دسترسی را در مرحله ۲-۷ بحث خواهیم کرد.

ایجاد دیدها در مایکروسافت اکسس

مایکروسافت اکسس دستور SQL، CREATE VIEW را پشتیبانی نمی کند. در عوض، می توانید پرس وجویی با استفاده از QBE یا SQL ایجاد کنید. برای مثال، می توانید دید Staff1_View را با استفاده از پرس وجوی QBE نشان داده شده در شکل ۲-۱۵ (الف) یا با استفاده از دستور SQL نشان داده شده در شکل ۲-۱۵ (ب) ایجاد کنید. این پرس وجو حالا می تواند برای ایجاد دیگر پرس وجوها یا رکوردهای درج/ بهنگام سازی/ حذف در جدول پایه Staff استفاده شود، و همچنین می تواند مبنایی برای ایجاد فرمها و گزارشها باشد.

شکل ۲-۱۵

ایجاد پرس وجو در مایکروسافت اکسس: (الف) با استفاده از QBE ; (ب) با استفاده از SQL.



مرحله ۲-۷ قواعد دسترسی را طراحی کنید

هدف

طراحی قواعد دسترسی جدولهای پایه و دیدهای کاربر.

DBMS های رابطه ای عموما دو نوع امنیت پایگاه داده را فراهم می کنند:

- امنیت سیستم،
- امنیت داده .

امنیت سیستم دسترسی و استفاده از پایگاه داده در سطح سیستم همچون نام کاربر و رمز عبور را می پوشاند. **امنیت داده** دسترسی و استفاده از اشیا پایگاه داده (همچون جدولها و دیدها) و اعمالی را که کاربران می توانند روی اشیا داشته باشند را می پوشاند.

همانطور که گفتیم، طراحی قواعد دسترسی وابسته بر DBMS هدف است; بعضی سیستمها امکانات بیشتری را نسبت به دیگری در طراحی قواعد دسترسی فراهم می کنند. مانند قبل، برای نشان دادن این مرحله ما دو راه ویژه طراحی قواعد دسترسی را با استفاده از موارد زیر انجام می دهیم:

(۱) ISO SQL استاندارد ۱۹۹۲ (SQL2) ،
(۲) مایکروسافت اکسس.

در فصل ۱۷، چگونگی طراحی قواعد دسترسی را در اوراکل ۸ با استفاده از مثال دیگری بررسی می کنیم.

ISO SQL استاندارد ۱۹۹۲ (SQL2)

یک راه تامین امنیت داده استفاده از تسهیلات کنترل دسترسی SQL است. همانطور که گفته شد، کاربران نباید به جدولهای پایه دسترسی مستقیم داشته باشند. در عوض، آنها باید به جدولهای پایه از طریق دیدهای کاربری که در مرحله ۱-۷ طراحی شده دسترسی پیدا کنند. این روش درجه بزرگی از استقلال داده را فراهم می کند و حفاظی را در برابر کاربران از جهت تغییرات در ساختار پایگاه داده ایجاد می کند. در ادامه مختصرا مکانیزمهای کنترل دسترسی SQL2 را مرور می کنیم.

به هرکاربر پایگاه داده یک *شناسه اختیار*[1] توسط مدیر پایگاه داده (DBA) اختصاص داده میشود; معمولا، شناسه همراه یک رمز میباشد، که برای دلایل امنیتی بکار میرود. هر دستور SQL ای که توسط DBMS اجرا می شود در حقیقت توسط کاربر مشخصی اجرا شده است. شناسه اختیار جهت مشخص کردن اینکه کاربر ممکن است به کدام شی پایگاه داده رجوع کند، و چه عملیاتی شاید بر روی آن اشیا انجام گرفته شود استفاده می شود. هر شی که در SQL ایجاد شده است مالکی دارد که توسط شناسه اختیار شناسایی می شود. به طور پیش فرض، مالک شخصی است که وجود شی را می داند و هر عملیاتی را روی آن شی ایجاد می کند.

Privilage ها امتیازاتی هستند که کاربران با استفاده از آن برای انجام دادن عملی روی دید یا جدول پایه مجاز شده اند. برای مثال ، SELECT امتیازی برای بازیابی داده و UPDATE امتیازی برای تغییر رکوردهای جدول است. وقتیکه کاربر جدولی را با استفاده از دستور SQL CREATE TABLE ایجاد می کند، به طور خودکار مالک جدول می شود و تمام امتیازات بر روی جدول را دریافت می کند. کاربران دیگر در ابتدا هیچ امتیازی را بر روی جدول تازه ایجاد شده ندارند. برای دادن دسترسی آنها به جدولها، مالک بایستی به طور واضح امتیازات لازم را با استفاده از دستور SQL GRANT به آنها بدهد. عبارت A WITH GRANT OPTION می تواند با دستور GRANT مشخص شود که به کاربر(های) دریافت کننده اجازه می دهد که امتیازات را به دیگر کاربران پاس دهند. امتیازات می توانند با استفاده از دستور SQL REVOKE پس گرفته شوند.

وقتیکه کاربر دیدی را با استفاده از دستور CREATE VIEW ایجاد می کند به طور خودکار مالک این دید می شود، اما لزوما امتیازات کامل آن دید را دریافت نمی کند. برای ایجاد دید کاربر بایستی امتیاز SELECT را برای تمام جدولها که تشکیل دهنده آن دید هستند داشته باشد. بنابراین، مالک تنها زمانی دیگر امتیازات را دریافت می کند که او آن امتیازات را برای هر جدول دید نگه دارد.

برای مثال، برای اینکه به کاربر MANAGER اجازه دهیم رکوردها را از جدول Staff بازیابی کنند و در جدول Staff درج، بهنگام سازی، و حذف داده انجام دهد شما می توانید از دستور SQL زیرین استفاده کنید:

```
GRANT ALL PRIVILEGES
ON staff
TO manager WITH GRANT OPTION;
```

در این مورد، MANAGER همچنین قادر به ارجاع جدول و تمام ستونهای موجود در جدولی که متناوبا ایجاد کرده است می باشد. عبارت WITH GRANT OPTION مشخص می کند که MANAGER می تواند این امتیازات را به هر کاربری که

[1] Authorization identifier

مناسب می بیند اعطا کند. به عنوان مثال دیگری، شما می توانید به کاربران با شناسه اختیار ADMIN امتیاز SELECT را بر روی جدول staff با استفاده از دستور SQL زیرین بدهید:

```
GRANT SELECT
    ON staff
    TO admin;
```

عبارت WITH GRANT OPTION در اینجا حذف شده بنابراین ADMIN قادر به پاس دادن این امتیازات به دیگر کاربران نخواهد بود.

امنیت در مایکروسافت اکسس

مایکروسافت اکسس ۹۷ دستورات SQL GRANT و REVOKE را پشتیبانی نمی کند. در عوض، اکسس دو روش زیرین را برای امن کردن پایگاه داده فراهم آورده است:

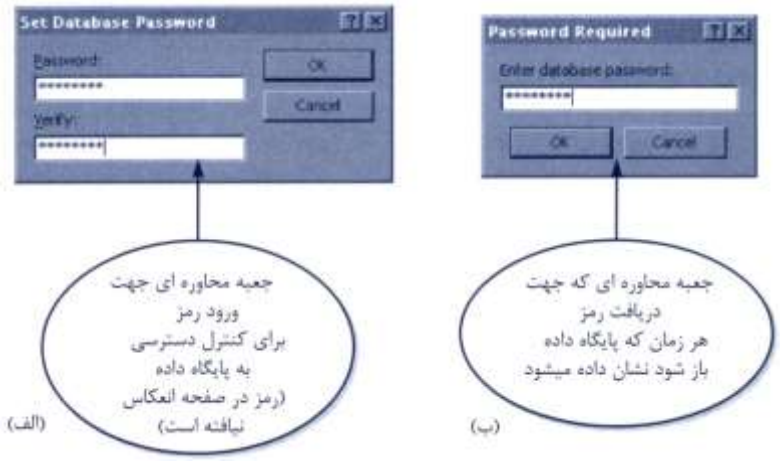
(الف) دادن رمز عبور بهنگام باز کردن پایگاه داده (امنیت سیستمی)
(ب) امنیت سطح کاربر، که میتواند برای محدود سازی قسمتهایی که کاربر می تواند خوانده یا بهنگام سازد استفاده شود (امنیت داده)

دادن رمز عبور

ساده ترین روش دادن رمز هنگام شروع پایگاه داده است. بار اول که رمز داده می شود
هر زمان که پایگاه داده باز شود جعبه متنی که رمز را درخواست می کند نشان داده خواهد شد. جعبه متنی برای دادن رمز عبور و درخواست رمز بهنگام بازشدن پایگاه داده در شکل ۳-۱۵ نشان داده شده است.

شکل ۳-۱۵

امن کردن پایگاه داده *StayHome* با استفاده از رمز عبور: (الف) جعبه محاوره ای Set Database Password ; (ب) جعبه محاوره ای Password نشان داده شده در آغاز برنامه .



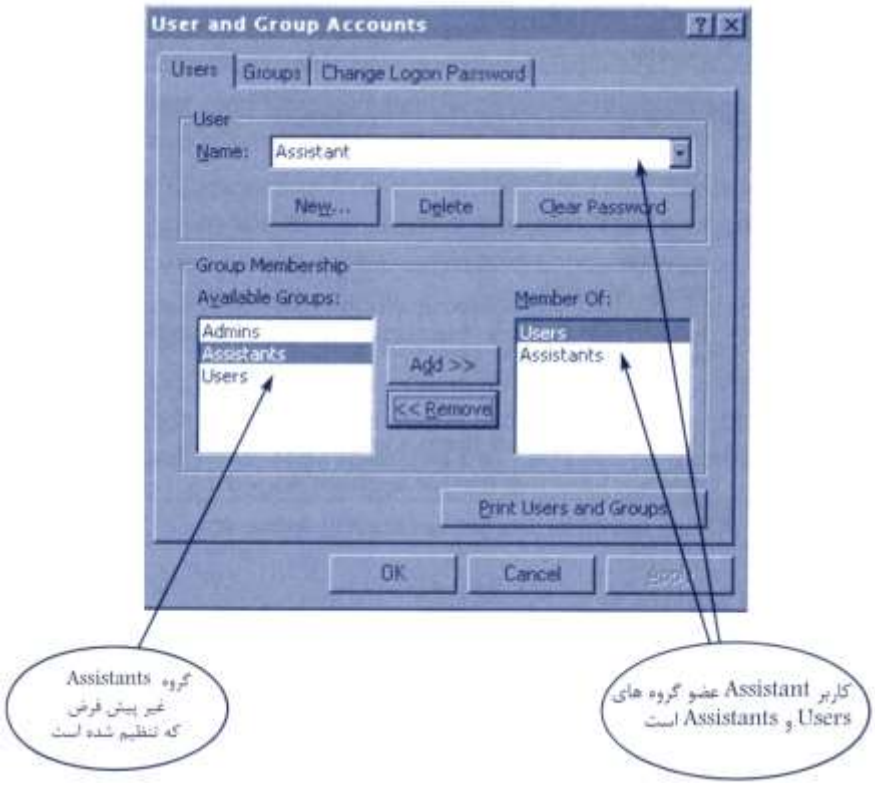
تنها کاربرانی که رمز صحیح را تایپ کنند مجاز به باز کردن پایگاه داده هستند. در این روش ازآنجاییکه مایکروسافت اکسس رمز وارد شده را بصورت کد شده در میاورد بنابراین به طور مستقیم توسط فایل پایگاه داده قابل خواندن نیست. بنابراین، وقتیکه پایگاه داده باز شد تمام اشیاء موجود در پایگاه داده در دسترس کاربر خواهد بود.

امنیت سطح کاربر

امنیت سطح کاربر در مایکروسافت اکسس مشابه روشهای استفاده شده در بیشتر سیستمهای شبکه می باشد. کاربران وقتیکه مایکرسافت اکسس را اجرا می کنند نیاز به شناساندن خودشان و تایپ کردن رمز دارند. در داخل فایل اطلاعات گروه کاری (Workgroup) ، کاربران بعنوان عضوی از *گروه* شناسایی می شوند. اکسس دو گروه پیش فرض را فراهم کرده است:
مدیران (گروه Admins) ،کاربران (گروه Users)، در عین حال گروهای دیگری نیز می توانند تعریف شوند. شکل ۴-۱۵ جعبه متنی استفاده شده برای تعریف سطح امنیت برای حسابهای group و user را نشان می دهد. شکل گروه غیر پیش فرضی بنام Assistants را نشان می دهد و کاربری که Assistant نامیده می شود عضوی از گروهای users و Assistants است.

شکل ۴-۱۵

کادر محاوره ای User and Group Accounts پایگاه داده *StayHome*



مجوز ها[1] به گروهها و کاربران عطا می شوند تا تنظیم کنند که چگونه آنها مجاز به استفاده از هر شی پایگاه داده بوسیله جعبه متنی گروه و کاربر می باشند. جدول ۱-۱۵ مجوز هایی را که بایستی در مایکروسافت اکسس ست شوند را نشان می دهد. برای مثال شکل ۵-۱۵ جعبه متنی برای کاربری که در *StayHome* معاون نامیده می شود را نشان می دهد که همان کاربری است که تنها دسترسی خواندن دید Staff1_view را که قبلا ایجاد شده است را دارد. بطور مشابه تمام دسترسی به جدول پایه Staff حذف خواهد شد بنابراین کاربر معاون فقط می تواند داده جدول Staff را با استفاده از این دید مشخص کند.

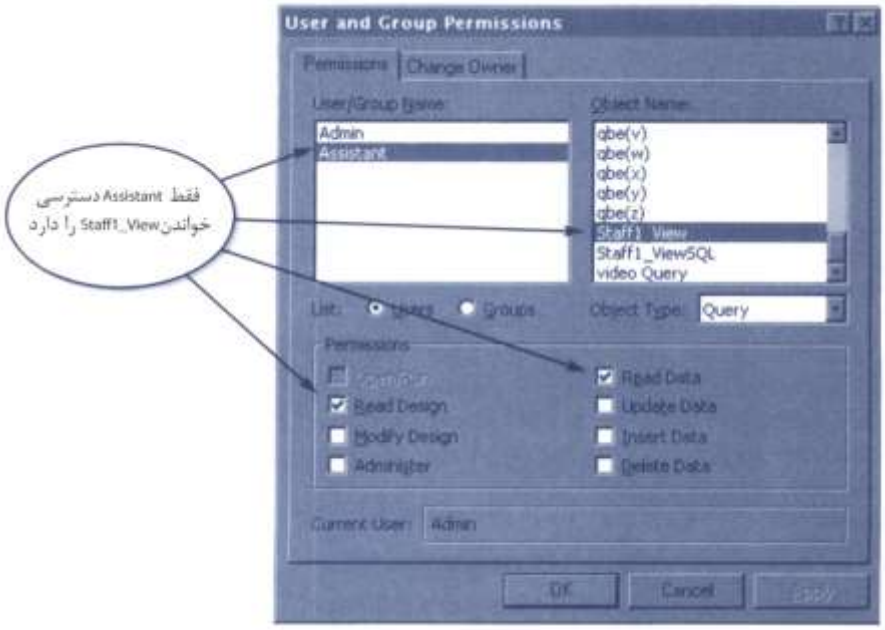
[1] Permission

جدول ۱-۱۵ اجازه های مایکروسافت اکسس

اجازه	توضیح
Open/Run	بازکردن پایگاه داده، گزارش، فرم، یا اجرای ماکرو.
Open Exclusive	بازکردن پایگاه داده با دسترسی انحصاری.
Read Design	مشاهده اشیاء در Design View.
Modify Design	مشاهده و تغییر اشیاء پایگاه داده، و حذف کردن آنها.
Administer	برای پایگاه داده، قراردادن رمز پایگاه داده، تکرار پایگاه داده، و تغییر خصوصیات آغازین. دسترسی کامل به اشیاء پایگاه داده شامل توانایی تخصیص مجوزها.
Read Data	مشاهده داده.
Update Data	مشاهده و تغییر داده (اما نه درج یا حذف داده).
Insert Data	مشاهده و درج داده (اما نه بهنگام سازی یا حذف داده).
Delete Data	مشاهده و حذف داده (اما نه درج یا بهنگام سازی داده).

شکل ۵-۱۵

جعبه محاوره ای User and Group Permissions که نشان دهنده آن است که تنها کاربر Assistant اجازه دسترسی خواندن پرس وجوی Staff1_View را دارد.



طراحی دیدهای کاربر و معیارهای امنیت را مستند کنید

طراحی دیدهای منفرد کاربر و مکانیزمهای امنیت مربوطه بایستی بطور کامل مستند شوند. اگر طراحی فیزیکی مدلهای داده منطقی محلی منفرد را تحت تاثیر قرار دهند این مدلها نیز بایستی بهنگام شوند.

خلاصه فصل

- ✓ پایگاه داده منبع ضروری شرکت را نشان میدهد، و بنابراین امنیت منبع بسیار مهم است.
- ✓ در مرحله ۷ شما طراحی کردید که چگونه معیارهای کارایی که طی مرحله تحلیل مشخص شده اند بایستی تحقق یابند. که این معیارهای کارایی احتمالا شامل ایجاد دیدهای کاربر و استفاده از مکانیزمهای کنترل دسترسی همچون آنهایی که توسط SQL فراهم گشته است می باشد.
- ✓ DBMS های رابطه ای عموما دو نوع امنیت پایگاه داده را فراهم می کنند: امنیت سیستم و امنیت داده. **امنیت سیستم** دسترسی و استفاده از پایگاه داده را در سطح سیستم، همچون نام کاربر و رمز عبور را می پوشاند. امنیت داده دسترسی و استفاده از اشیاء پایگاه داده (همچون جدولها و دیدها) و عملهایی که کاربران می توانند بر روی اشیا را داشته باشند را می پوشاند.

فصل ۱۶

طراحی فیزیکی پایگاه داده- مرحله ۸

آنچه در این فصل خواهید آموخت :

- اهمیت نظارت و تنظیم سیستم عملکردی .

این فصل پنجمین و آخرین مرحله از متدلوژی طراحی فیزیکی پایگاه داده ما را می پوشاند. در چهار فصل قبل، طراحی منطقی را به مجموعه ساده ای از جدولها، سازمانهای فایل مناسب انتخابی و شاخصهای مبتنی بر تحلیل تراکنشهای مهم تبدیل کردیم، و افزونگی کنترل شده را برای رسیدن به بهبودهای کارایی هرچه بیشتر در نظر گرفتیم و همچنین بررسی کردیم که چگونه پایگاه داده را امن سازیم.

نیازمندیهای کاربر بدست آمده، معمولا نیاز به بهبود، یا تنظیم دارند، تا پایگاه داده به کارایی مورد قبول برسد. بعلاوه، شما احتمالا دریافته اید که نیازمندیها یا بخاطر موفقیت سیستم و اینکه کاربران طالب کارایی بیشتری هستند و یا در نتیجه اینکه حرفه در حال توسعه است تغییر می کنند. در این فصل، ما مرحله آخر طراحی فیزیکی پایگاه داده را که این جوانب را می پوشاند در نظر می گیریم.

مرحله ۸ سیستم عملکردی را میزان کرده و بر آن نظارت کنید

هدف

نظارت بر سیستم عملکردی و بهبود کارایی سیستم جهت اطلاح تصمیمات طراحی نامناسب یا انعکاس نیازمندیهایی که تغییر یافته اند.

شما نباید طراحی فیزیکی پایگاه داده ابتدایی را بصورت ثابت در نظر بگیرید، بلکه بایستی بصورت تخمینی از اینکه چگونه سیستم عملکردی بایستی کار کند در نظر بگیرید. هر بار که طراحی ابتدایی پیاده سازی می شود، همواره بایستی بر سیستم نظارت کرده و آنطور که می خواهید سیستم در آینده عمل کند میزان کنید و نیازمندیها را برآن اساس تغییر دهید. بعضی از DBMS ها مدیر پایگاه داده ای (DBA) را همراه برنامه های سودمندی که برای نظارت بر عمل سیستم و همچنین میزان کردن آن بکار می رود فراهم می آورند.

میزان سازی پایگاه داده برای ما منافعی دارد که نمونه هایی از آن را در زیر ذکر می کنیم:

- میزان سازی می تواند از تهیه سخت افزارهای اضافی جلوگیری بعمل آورد.
- میزان سازی میتواند کمک کند که اندازه پیکربندی سخت افزاری را کم کنیم که این منتج به سخت افزار کم و ارزان شده و به طبع آن هزینه نگهداری نیز کم خواهد شد.
- سیستمی که بخوبی میزان شده است زمان پاسخگویی سریعتر و گذردهی بهتر را تولید میکند، که باعث برگرداندن کاربران و در نتیجه بوجود آمدن یک شرکت تولید کننده و پرحاصل می شود.
- زمان پاسخگویی بهبود یافته می تواند روحیه پرسنل را تقویت کند.
- زمان پاسخگویی بهبود یافته می تواند رضایتمندی مشتریان را افزایش دهد.

دو مورد آخر ممکن است از بقیه موارد غیر قابل لمس تر باشد. با این وجود، بدیهی است که زمان پاسخگویی آهسته سیستم باعث ناخشنودی پرسنل و در نتیجه از دست دادن مشتریان خواهد شد. میزان سازی[1] از جمله فعالیت هایی است که هرگز کامل نیست. در سرتاسر حیات سیستم، شما عموما نیاز خواهید داشت که بر کارایی سیستم نظارت کنید، و علت این نظارت به سبب تغییرات در محیط نیازمندیهای کاربر می باشد. با این وجود، انجام تغییرات در یک ناحیه از سیستم عملکردی جهت بهبود کارایی ممکن است تاثیر معکوسی بر دیگر نواحی بگذارد. برای مثال، افزودن شاخص به جدول ممکن است که کارایی را در یک برنامه کاربردی بهبود بخشد، اما در عین حال ممکن است برنامه کاربردی مهمتری را تحت تاثیر قرار دهد. بنابراین، زمانیکه تغییری را در سیستم عملکردی انجام می دهید بایستی مراقب باشید. به عنوان جلوگیری از تغییرات ناخواسته می توانید تغییرات را یا در پایگاه داده آزمایشی و یا اینکه هنگامیکه سیستم بیکار است تست کنید. (مثلا، هنگام خارج از ساعات کاری)

نیازمندیهای جدید *StayHome*

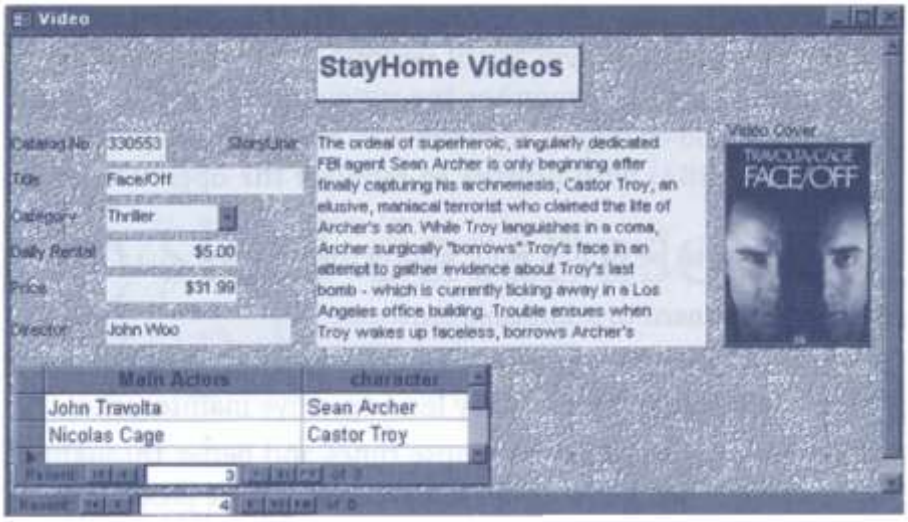
شرکت *StayHome* تصمیم گرفته است که جدول Video بایستی تصویر روی جلد فیلم بهمراه مختصری از داستان را نگه دارد، تا بتوان آن را برای دید عموم در وب قرار داد. شما می توانید این نیازمندی جدید را در مایکروسافت اکسس با استفاده از نوع داده OLE، که جهت ذخیره کردن داده هایی همچون مستندات مایکروسافت ورد یا اکسل، تصاویر، صداها، و انواع دیگری از داده باینری ایجاد شده در دیگر برنامه ها بکار می رود تطبیق دهید. اشیاء OLE میتواند بصورت فیلد در جدول مایکروسافت اکسس جاسازی یا لینک داده شده و سپس در فرم یا گزارش نشان داده شوند.

برای برآورد ساختن نیازمندی جدید، شما می توانید جدول Video را با افزودن تغییرات زیر دوباره بسازید:

(۱) ستونی بنام videoCover ایجاد کنید که بصورت نوع داده OLE Object باشد, و
(۲) ستونی بنام storyline با نوع داده Memo مشخص کنید که توانایی ذخیره متنهای طویل را دارد.

فرمی که از این ستونهای جدید استفاده می کند در شکل ۱-۱۶ نشان داده شده است. مشکل عمده ای که با افزودن این دو ستون اضافی وجود دارد این است که بطور بالقوه میزان زیادی از فضای دیسک جهت ذخیره فایلهای گرافیکی و میزان زیادی از متن برای ذخیره خط داستان نیاز داریم. جهت حل این مشکل باید عمل نظارت بر کارایی پایگاه داده را ادامه دهید تا مطمئن شوید که این نیازمندیهای جدید کارایی سیستم را تحت تاثیر قرار ندهد.

[1] Tuning



شکل ۱-۱۶

فرم براساس جدول Video اصلاح شده با ستونهای افزوده شده videoCover و storyline .

شما حالا بخوبی تمام مراحل موجود در متدلوژی را پشت سر گذاشتید، و شاید بخواهید به فصل ۴-۱-۷ برای مشاهده فاکتورهایی که گفتیم در طراحی موفق پایگاه داده اهمیت دارند نگاهی بیاندازید. شاید زمانیکه در اول این بخش را مطالعه می کردید با خود فکر می کردید که مشاهده ارتباط بعضی از این فاکتورها مشکل است اما امیدواریم هم اکنون به اهمیت همه آنها پی برده باشید.

خلاصه فصل

- ✓ مرحله ۸، آخرین مرحله از طراحی فیزیکی پایگاه داده است که شامل فرایند مداوم نظارت و تنظیم سیستم عملکردی برای رسیدن به حداکثر کارایی یا انعکاس نیازمندیهایی است که تغییر یافته اند.

فصل ۱۷

پرس و جوهای ساده *StayHome* با استفاده از SQL و QBE

آنچه در این فصل خواهید آموخت:

- چگونه پرس وجو های *StayHome* را با استفاده از SQL پیاده سازی کنیم.
- چگونه پرس وجو های *StayHome* را با استفاده از QBE پیاده سازی کنیم.

در این فصل نشان می دهیم که چگونه برخی از پرس و جوهای داده *StayHome* که در بخش ۴-۴-۴ ذکر شدند می توانند در مایکروسافت SQL و مایکروسافت QBE پیاده سازی شوند. با این وجود، در این فصل قصدمان مرور SQL یا QBE نیست، بلکه هدف این است که نشان دهیم چگونه طراحی قبلی می تواند منجر به پیاده سازی شود. برای بحث کامل درباره SQL و QBE خوانندگان علاقمند می توانند به Connolly و Begg (۱۹۹۲) رجوع کنند.

۱-۱۷ مقدمه ای بر مایکروسافت SQL و QBE

دو زبان اصلی که برای DBMS های رابطه ای ضروری هستند به صورت زیر میباشند:

- SQL (زبان ساختیافته پرس وجو)
- QBE (پرس وجو با مثال)

۱-۱-۱۷ SQL

SQL زبان پایگاه داده رابطه ای تجاری متداول است، که برای استفاده افراد حرفه ای یا غیرحرفه ای طراحی شده است. دستورات SQL شامل کلمات استاندارد زبان انگلیسی همچون SELECT، INSERT، UPDATE ، و DELETE است. این زبان در SEQUEL و پروژه های System-R موجود در لابراتوار تحقیق IBM در San Jose بین سالهای ۱۹۷۴ و ۱۹۷۷ توسعه داده شده است. امروز، بعضی از افراد هنوز SQL را بصورت 'See-Quel' تلفظ می کنند، اگرچه تلفظ تجاری آن 'S-Q-L' می باشد. با شروع کار اوراکل در اواخر سال۱۹۷۰، تعدادی از RDBMS های تجاری مبتنی بر SQL، همچنین با ANSI/ISO استاندارد موجود است، و امروزه SQL یک زبان استاندارد برای دستکاری پایگاه داده های رابطه ای می باشد.

متاسفانه، در حال حاضر هیچ RDBMS ای کاملا از استانداردی که در سال ۱۹۹۲ بود مطابقت نمی کند. برای مثال، بعضی سیستمها انواع داده اضافی مانند آنهایی که بالا و قبلا گفته شد فراهم می کنند (همچون Autonumber غیراستاندارد، OLE-OBJECT، و انواع داده ای Currency، که توسط مایکروسافت اکسس تدارک دیده شده است). در مثالهایی که در این فصل بررسی می کنیم، آن قسمتهایی از پرس وجوهای SQL ای را که استاندارد نیستند مشخص می کنیم. ابتدا اجازه دهید فرمت دستور SELECT را که در این فصل بیشتر بر روی آن متمرکز خواهیم شد بررسی کنیم.

دستور SELECT

هدف دستور SELECT بازیابی و نمایش داده از یک یا چند جدول می باشد. که یک دستور بسیار قوی بوده و همچنین بعنوان یکی از فرمانهایی که مکررا در SQL استفاده می شود بشمار می رود. شکل کلی دستور SELECT بدین صورت است:

```
SELECT   [DISTINCT | ALL]
{* | [column_expression [AS new_name]] [,...]}
FROM table_name  [alias] [,...]
[WHERE condition]
[GROUP BY column_list]      [HAVING condition]
[ORDER BY column_list]
```

کروشه [ ] عناصر اختیاری را نشان می دهد، و کروشه مجعد{}عنصری را که به تعداد صفر یا بیشتر تکرار می شود را نشان می دهد. علاوه بر آنها:

- *column_expression* نام ستون یا عبارت را نشان می دهد;
- *new_name* نامی را که ستون را میتوانید با آن نام نمایش دهید بیان می کند;
- *table_name* نام جدول موجود در پایگاه داده یا دیدی که به آن دسترسی دارید را مشخص می کند;
- *alias* خلاصه ای اختیاری برای *table_name* است;
- *Column_list* فهرستی از یک یا چند ستون را مشخص می کند.

ترتیب پردازش در دستور SELECT بدین صورت است:

FROM	جدول یا جداول استفاده شده را مشخص می کند.
WHERE	موضوع رکوردها را به شرایط خاصی فیلتر می کند.
GROUP BY	گروهی از رکوردهای با مقادیر مشابه فیلد را شکل می دهد.
HAVING	موضوع گروهها را به بعضی از شرایط فیلتر می کند.
SELECT	مشخص میکند که کدام ستونها در خروجی مشخص شوند.
ORDER BY	ترتیب خروجی را مشخص می کند.

ترتیب عبارات در دستور SELECT *نمی تواند* تغییر داده شود. و تنها دو عبارت اولی SELECT ، و FROM اجباری هستند. بقیه اختیاری می باشند.

۲-۱-۱۷ پرس وجو با مثال (QBE)[۱]

QBE روش پرس وجوی دیگری است که میتوان از پایگاه داده انجام داد، که یک روش مبتنی بر گرافیک، و بصورت ' اشاره و کلیک' می باشد. QBE بعنوان یکی از ساده ترین روشهایی که کاربران غیر فنی می توانند اطلاعات را از پایگاه داده استخراج کنند شهرت یافته است. QBE ابزار بصری برای پرس وجوی داده با استفاده از الگوها را فراهم میاورد. پرس وجوی پایگاه داده بوسیله نمایش پرس و جوی نتیجه در صفحه نمایش بدست می آید. بنابراین شما بایستی ستونهایی (که در اکسس فیلد نامیده میشوند) را که می خواهید مشاهده کنید و مقادیر داده ای را جهت محدودکردن پرس وجو استفاده کنید را مشخص کنید. زبانهایی مثل QBE می توانند در پرس و جوی محاوره ای بهنگام سازی پایگاه داده یک روش بسیار سودمند باشند.
همچون SQL، QBE نیز در IBM توسعه یافته است (در حقیقت QBE دارای علامت تجاری IBM است)، اما تعداد دیگر فروشنده ها، که مایکروسافت نیز جزو آنهاست، واسطهای شبیه QBE را می فروشند. اغلب فروشندگان هر دو امکان SQL و

[۱] Query-By-Example(QBE)

QBE را فراهم میاورند، که QBE بعنوان واسط شهودی پرس و جوی های ساده بشمار میروند و SQL نیز در پرس و جو های پیچیده بکار برده میشود.

شما در این فصل، نسخه QBE پرس وجوهایی را خواهید دید که ساده تر هستند، شاید به استثناء آن هایی که شامل یک جدول بوده و نیازی به استفاده از توابع گروه بندی یا تجمعی ندارند.

۱۷-۲ پرس وجوهای نمونه *StayHome*

در این بخش، بعضی از پرس وجوهای داده ای را که در بخش ۴-۴-۴ لیست شده است را برای *StayHome* در نظر می گیریم. همچنین از شکل شماره گذاری تراکنشی که در آن بخش استفاده شده پیروی می کنیم.

(۱۳) جزئیات شعبه ها در شهر معینی را فهرست کنید

جزئیات شعبه به دو جدول تقسیم شده است. بیشتر جزئیات شعبه در جدول Branch ذخیره شده اما شماره تلفنهای شعبه در جدول Telephone ذخیره شده است. برای دسترسی به دو مجموعه از جزئیات، ما باید این دو جدول را بر اساس ستون branchNo به یکدیگر الحاق کنیم.

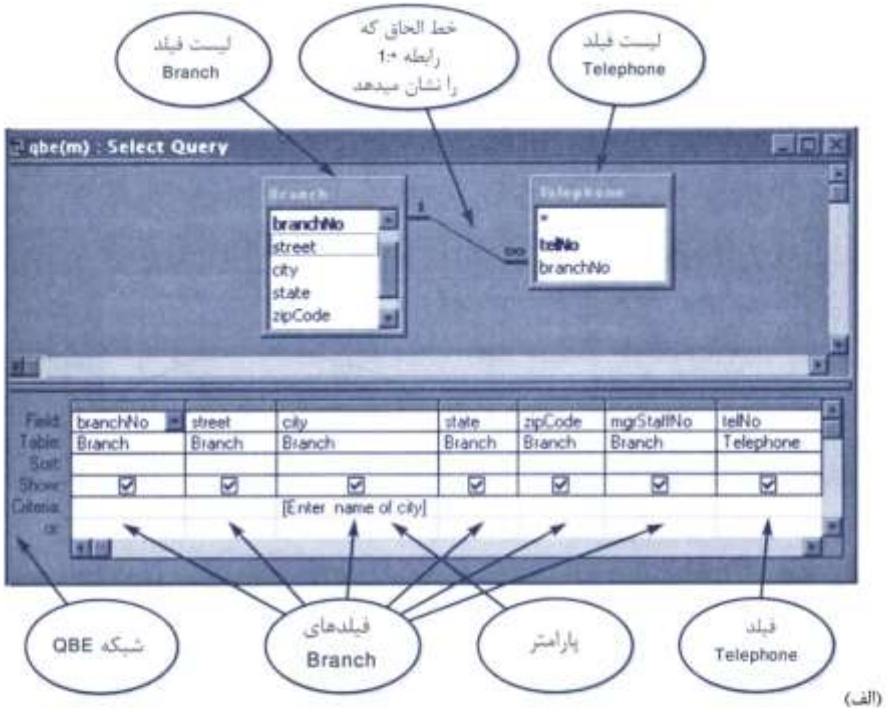
بعلاوه، از ما خواسته شده است که نتیجه را به شعبه هایی در شهر معین محدود کنیم. اگر شهر Seattle بود، برای مثال، می توانیم از شرط (city = Seattle) استفاده کنیم. برای اینکه به کاربر اجازه دهیم که هنگام اجرا شهر را مشخص کند، مایکروسافت اکسس الحاقی را برای SQL استاندارد بنام Parameters فراهم کرده است، که میتواند شهر را بین دو کروشه بعنوان عبارت معیار تایپ کند، برای مثال '[Enter name of city]' که در این صورت مایکروسافت اکسس جعبه محاوره ای با هشدار (prompt) معینی برای پارامتری که قسمتی از پرس وجو را تشکیل می دهد نشان می دهد. پرس وجوی QBE این تراکنش در شکل ۱-۱۷(الف) و جعبه محاوره ای برای پارامتر city در شکل ۱-۱۷(ب) نشان داده شده است. پرس وجوی SQL معادل نیز در شکل ۱-۱۷(ج) نشان داده شده است.

توجه کنید که ما از این اختصارها در عبارت FROM بصورت مخفف استفاده کرده ایم. اگر دید SQL پرس و جوی QBE را بررسی کنید، خواهید دید که مایکروسافت اکسس از اختصارها استفاده نمی کند اما به جای آن از شکل توسعه یافته برای هر جدول استفاده می کند. با این وجود، SQL استاندارد اختصارها را مجاز می داند، مانند آنچه که شما در دید SQL پرس وجوی SQL مایکروسافت اکسس تایپ کنید. همچنین SQL اکسس از شکل INNER JOIN دستور SELECT برای الحاق دو جدول استفاده می کند. همچنین شکل دیگری از دستور SQL را در شکل۱-۱۷(د) نشان داده ایم که، بیشتر شما احتمالا با آن آشنا هستید، و اگر بازهم بخواهید پرس وجو را در دید SQL وارد کنید قابل قبول است. شکل۱-۱۷(ذ) خروجی نمونه ای از اجرای پرس وجو را نشان می دهد.

> توجه کنید که در هر دو نسخه پرس و جوی SQL مجبوریم مشخص کنیم که چگونه جدولهای Branch و Telephone را باید بهم الحاق کنیم. با این وجود، QBE این را به طور خودکار برای ما انجام می دهد. که این را در دیگر پرس وجو های بحث شده در این فصل خواهید دید. توجه کنید که چگونه این، برای ما QBE را ساده تر از SQL می سازد.

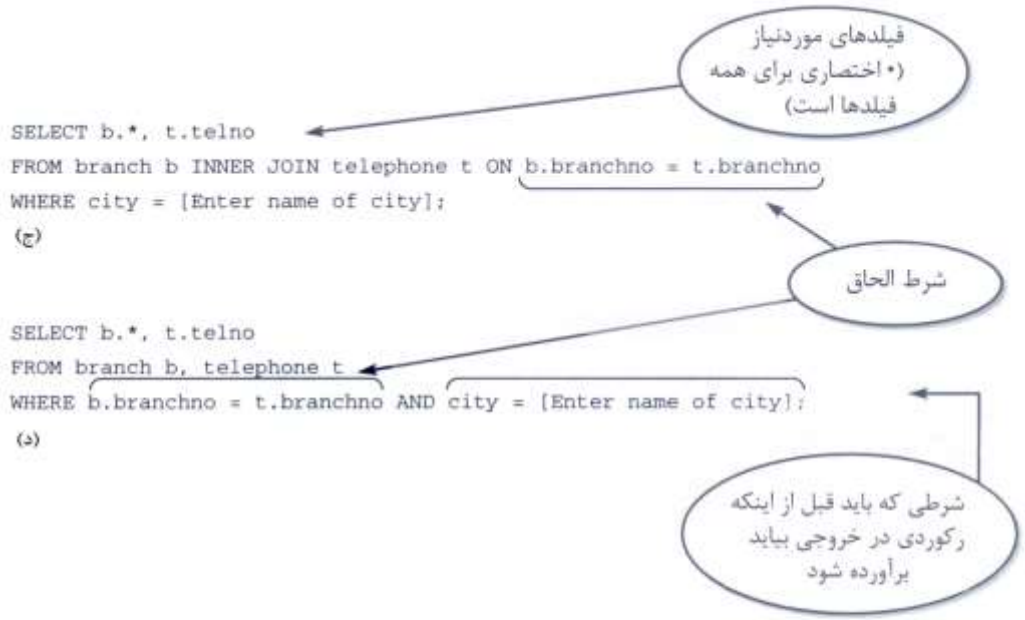
شکل ۱-۱۷

تراکنش (m): (الف) QBE ;(ب) جعبه محاوره ای پارامتر city ;(ج) SQL معادل با inner join; (د) شکل دیگر SQL; (ذ) برگه داده نتیجه نشان دهنده خروجی پرس و جو .



شکل ۱-۱۷

ادامه شکل.



qbe(m) : Select Query

branchNo	street	city	state	zipCode	mgrStaffNo	telNo
B002	City Center F	Seattle	WA	98122	S0010	206-555-6756
B002	City Center F	Seattle	WA	98122	S0010	206-555-8836
B004	16 - 14th Ave	Seattle	WA	98128	S2250	206-555-3131
B004	16 - 14th Ave	Seattle	WA	98128	S2250	206-555-4112

Record: 1 of 4

(ذ)

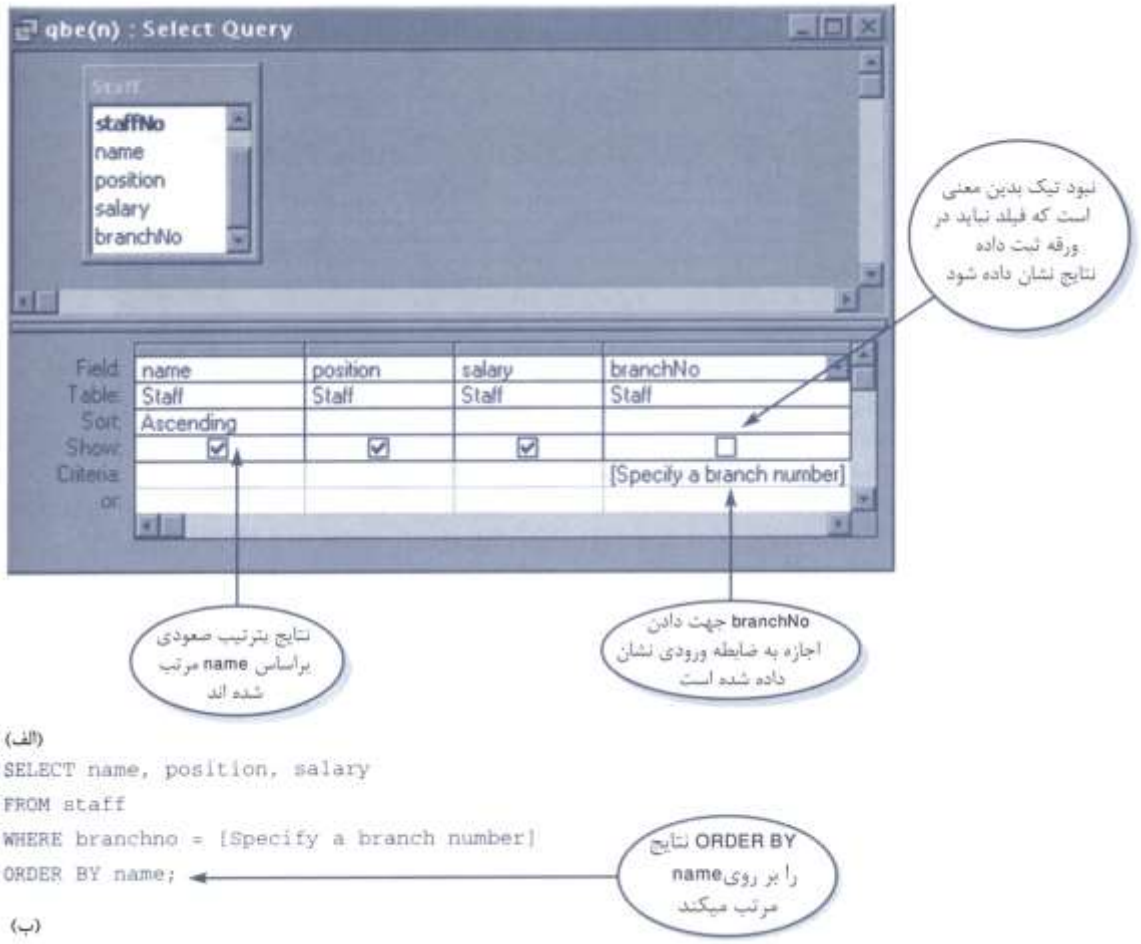
بعد از نشان دادن مثالی از دو نسخه دیگر SQL، نسخه INNER JOIN را در بقیه این فصل نشان می دهیم.

(n) نام، موقعیت، و حقوق پرسنلی را در شعبه معینی، بترتیب نام پرسنلی فهرست کنید.

همه اطلاعات مورد نیاز این پرس و جو قسمتی از جدول Staff است. دوباره، می توانیم از پرس و جوی پارامتردار برای اینکه به کاربران اجازه دهیم شماره شعبه را بهنگام اجرا مشخص کنند استفاده کنیم. برای اینکه خروجی نام پرسنل را داشته باشیم، برای فیلد name، Ascending را از لیست پایین افتادنی موجود در خانه *Sort* انتخاب می کنیم. پرس وجوی QBE در شکل ۲-۱۷(الف) و پرس وجوی SQL معادل نیز در شکل ۲-۱۷(ب) نشان داده شده است.

شکل ۲-۱۷

تراکنش (n): (الف) QBE ;(ب) SQL معادل ;



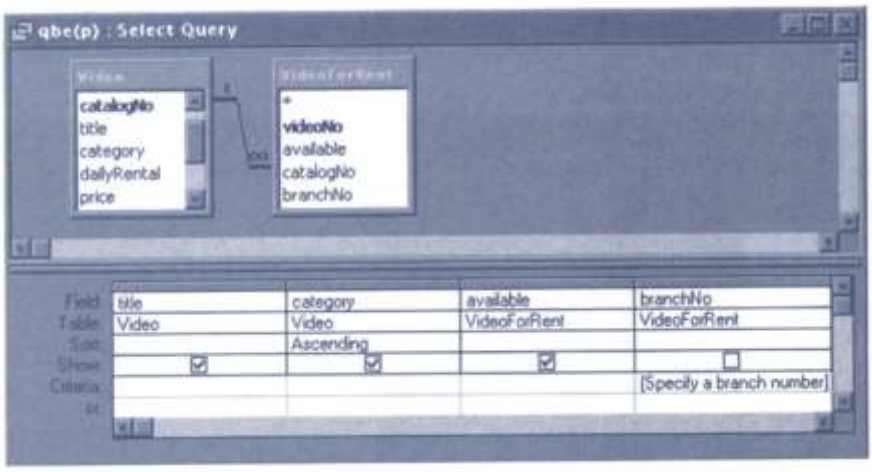
(p) عنوان، فهرست، و در دسترس پذیری همه فیلمهای شعبه مشخصی را بترتیب فهرست لیست کنید.

فیلدهای title و category قسمتی از جدول Video است; فیلدهای available و branchNo قسمتی از جدول VideoForRent است. دو جدول بر روی فیلد catalogNo الحاق شده، و خروجی حاصل بر روی فیلد category مرتب می شوند. پرس وجوی QBE در شکل ۳-۱۷(الف) و پرس وجوی SQL معادل در شکل ۳-۱۷(ب) نشان داده شده است.

شکل ۳-۱۷

تراکنش (p): (الف) QBE

(ب) SQL معادل;



(الف)

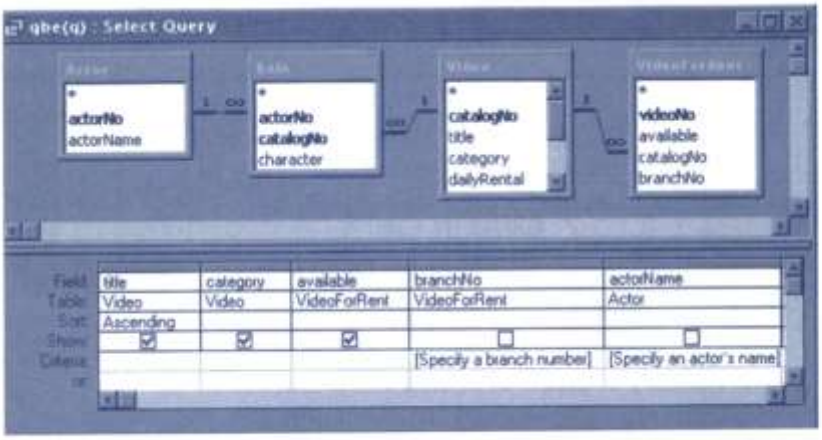
```
SELECT title, category, available
FROM video v INNER JOIN videoforrent vfr ON v.catalogno = vfr.catalogno
WHERE vfr.branchno = [Specify a branch number]
ORDER BY category;
```

(ب)

شکل ۴-۱۷

تراکنش(q) :(الف) QBE

(ب) SQL معادل ;



(الف)

```
SELECT title, category, available
FROM (video v INNER JOIN (actor a INNER JOIN role r ON a.actorno = r.actorno)
       ON v.catalogno = r.catalogno) INNER JOIN videoforrent vfr
       ON v.catalogno = vfr.catalogno
WHERE vfr.branchno = [Specify a branch number] AND
       a.actorname = [Specify an actor's name]
ORDER BY title;
```

(ب)

(q) عنوان، فهرست، در دسترس پذیر بودن همه فیلمها را برای نام بازیگر معینی در شعبه خاصی بترتیب عنوان لیست کنید.

همچون پرس وجوی قبلی فیلدهای title و category بخشی از جدول Video است; فیلدهای available و branchNo بخشی از جدول VideoForRent هستند که با فیلد catalogNo بهم الحاق می شوند. نام بازیگر بخشی از جدول actor است. برای الحاق جدول actor با جدول Video مجبوریم از جدول Role استفاده کنیم. جدولهای Actor و Role بر روی فیلد actorNo الحاق می شوند و جدول Role با جدول Video بر روی فیلد catalogNo الحاق می شوند. پرس و جوی QBE در شکل ۴-۱۷(الف) و پرس وجوی SQL معادل در شکل ۴-۱۷(ب) نشان داده شده است.

(v) تعداد همه فیلمهای موجود در هر فهرست فیلم هر شعبه را، بترتیب شماره شعبه لیست کنید.

فیلد category بخشی از جدول Video است و اطلاعات درباره کپی های هر فیلم بخشی از جدول VideoForRent است. این بار، بجای لیست کردن جزئیات هر کپی، می خواهیم تعداد تمام کپی ها را محاسبه کنیم. برای انجام این، از تابع تجمعی ***Count***، که توسط تغییر نوع پرس وجو به Totals دستیابی می شود استفاده می کنیم. این باعث نمایش ردیف اضافی بنام Total در شبکه[۱] QBE می شود. وقتیکه شما Totals را انتخاب می کنید، همه فیلدهایی که انتخاب کرده اید به طور خودکار به Group By در ردیف Total ست می شوند. ما می خواهیم اطلاعات را براساس فهرست و شماره شعبه گروه بندی کنیم. (ما دنبال Totals برای فهرست هر شعبه هستیم) بنابراین مطمئن شوید که این فیلدها به Group By ست شده باشند، اما ردیف Total را برای فیلد catalogNo به Count از لیست پایین افتادنی تغییر دهید. برای اینکه خروجی با معنا باشد نام فیلد سر فصل را به ورقه ثبت داده نتیجه به Total Videos تغییر می دهیم. زمانیکه پرس وجوی Totals اجرا می شود، ورقه ثبت داده نتیجه یک نمایش لحظه ای[۲] است ، یعنی مجموعه رکوردهایی که قابل بهنگام سازی نیستند.
پرس وجوی QBE در شکل ۵-۱۷(الف) و پرس وجوی SQL معادل نیز در شکل ۵-۱۷(ب) نشان داده شده اند.

> توجه کنید که تابع Count می تواند به هر فیلدی در جدول حاصل اعمال شود. در SQL، این عموما به صورت COUNT(*) نوشته می شود. با این وجود در QBE تابع تنها به فیلد مشخصی اعمال می شود.

(x) شماره کل فیلمها را به بازیگری هر ستاره فیلم، بترتیب نام بازیگر لیست کنید.

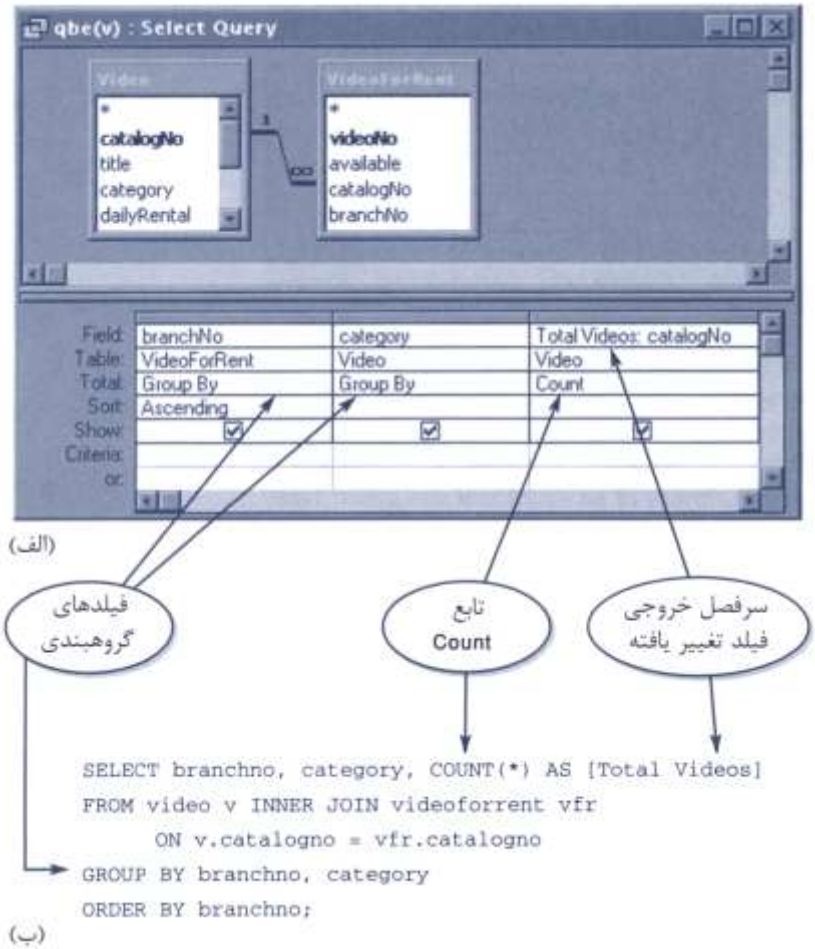
اطلاعات موردنیاز بخشی از جدولهای Video، Role، و Actor می باشد. برای پیدا کردن تعداد همه فیلمهایی که بازیگر ستاره در آن درخشیده است، می خواهیم داده را با actorName گروه بندی کرده و سپس تابع Count را به هر فیلد جدول Video (catalogNo) اعمال کنیم. پرس وجوی QBE در شکل ۶-۱۷(الف) و پرس وجوی SQL معادل در شکل ۶-۱۷(ب) نشان داده شده است.

---

[۱] Grid
[۲] Snapshot

شکل ۵-۱۷

تراکنش (۷): (الف) QBE ;
(ب) SQL معادل ;



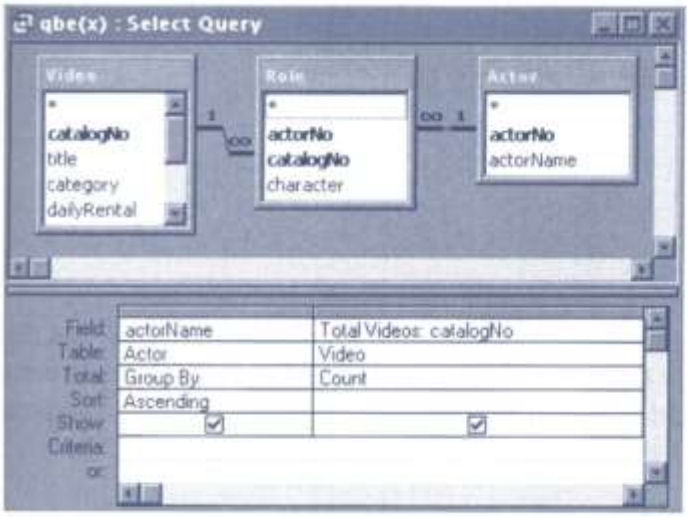
توجه کنید که، می توانستیم از actorNo بعنوان فیلد گروه بندی استفاده کنیم، اما از آنجاییکه می خواهیم نتیجه را با actorName مرتب کنیم از actorName بعنوان فیلد گروه بندی استفاده کردیم.

(Z) همه اجاره روزانه فیلمهای هر شعبه را بترتیب شماره شعبه لیست کنید.

داده ای که ما برای این پرس وجو نیاز داریم بخشی از جدولهای Video و VideoForRent است. جهت جمع کردن قیمتهای اجاره روزانه برای فیلمهای هر شعبه ما ابتدا از پرس وجوی Totals برای گروه بندی داده بوسیله فیلد branchNo در جدول VideoForRent استفاده می کنیم. سپس از تابع Sum برای بدست آوردن مجموع هرگروه جهت جمع کردن قیمتهای روزانه با همدیگر استفاده می کنیم. خروجی حاصل بر روی فیلد branchNo مرتب می شود. پرس وجوی QBE در شکل ۷-۱۷(الف) و پرس وجوی SQL معادل در شکل ۷-۱۷(ب) نشان داده شده است.

شکل ۶-۱۷

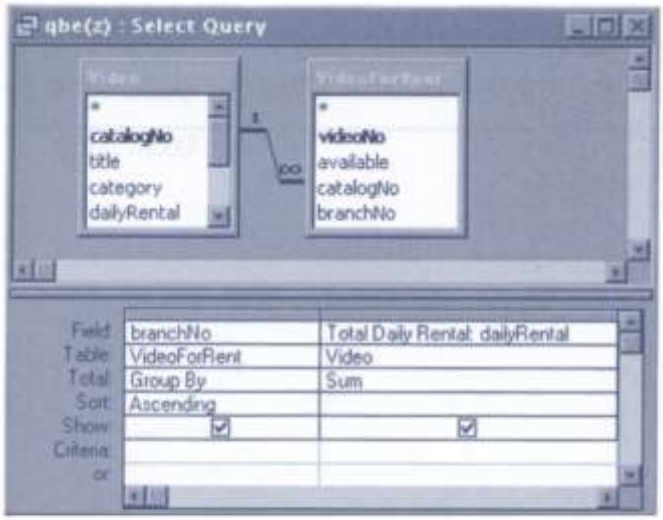
تراکنش (x) : (الف) QBE ;(ب) SQL معادل.




```
SELECT actorname, COUNT(*) AS [Total Videos]
FROM video v INNER JOIN (actor a INNER JOIN role r ON a.actorno = r.actorno)
     ON v.catalogno = r.catalogno
GROUP BY actorname
ORDER BY actorname;
```

شکل ۷-۱۷

تراکنش (z) : (الف) QBE ;(ب) SQL معادل;



(الف)

```
SELECT branchno, SUM(dailyrental) as [Total Daily Rental]
FROM video v INNER JOIN videoforrent vfr ON v.catalogno = vfr.catalogno
GROUP BY branchno
ORDER BY branchno;
```

(ب)

خلاصه فصل

- ✓ SQL زبان غیررویه ای است، که شامل کلمات استاندارد انگلیسی همچون SELECT ، INSERT، UPDATE ، و DELETE می باشد که توسط افراد حرفه ای یا غیرحرفه ای مورد استفاده قرار می گیرد. همچنین یک زبان برای تعریف و دستکاری پایگاه داده های رابطه ای است.
- ✓ هدف دستور SELECT بازیابی و نمایش داده از یک یا چند جدول پایگاه داده است، که یک فرمان بسیار قوی است و همچنین بعنوان فرمانی که مکررا در SQL استفاده می شود معروف است.
- ✓ QBE یک راه دیگر پرس وجو از پایگاه داده است که مبتنی بر گرافیک و اشاره وکلیک می باشد. QBE بعنوان یکی از آسانترین روشها برای کاربران غیر حرفه ای کامپیوتر جهت بدست آوردن اطلاعات از پایگاه داده مشهور شده است.

فصل ۱۸

Perfect Pets- طراحی منطقی پایگاه داده

در این فصل، بررسی موردی دیگری را که به شما کمک می کند تا متدلوژی ارائه شده در فصلهای ۸ تا ۱۶ بهتر درک کنید ارائه می کنیم. در این فصل مراحل متدلوژی طراحی منطقی پایگاه داده *Perfect Pets* و همچنین در فصل بعد طراحی فیزیکی پایگاه داده فوق را بررسی می کنیم. جهت نشان دادن بعضی از جنبه های پیاده سازی فیزیکی در این مثال از DBMS رابطه ای متفاوتی، که این بار اوراکل ۸ است استفاده خواهیم کرد. شما اگر بررسی موردی این بخش را خوانده و سپس مراحل متدلوژی را برای خودتان تمرین کنید می تواند برای شما مفید واقع شود. که بعد از تمرین می توانید راه حل خودتان را با راه حل ما بررسی کنید. خلاصه متدلوژی داده شده را می توانید در ضمیمه **ب** بیابید.

Perfect Pets ۱۸-۱

بررسی موردی که آنرا *Perfect Pets* می نامیم یک محل خصوصی برای مراقبت از حیوانات خانگی در آمریکاست. این سرویس از طریق کلینیک های مختلفی در شهرهای اصلی آمریکا فراهم شده است. رییس *Perfect Pets* از کمبود رابطه و خصوصا از اشتراک اطلاعات و منابع بین کلینیک های مختلف نگران است. برای حل این مشکل رییس جهت کمک به اینکه عمل طبابت هرچه بیشتر موثر و کارا صورت گیرد ایجاد یک سیستم پایگاه داده متمرکز را درخواست کرده است. او سیستم جدید را بدین صورت شرح داده است.

۱-۱-۱۸ نیازمندیهای داده

کلینیک های دامپزشکی

Perfect Pets دارای چندین کلینیک دامپزشکی واقع در شهرهای اصلی آمریکا می باشد. جزئیات هر کلینیک شامل شماره کلینیک، آدرس کلینیک (که شامل خیابان، شهر، ناحیه، و کدپستی)، و شماره های تلفن و فاکس است. هر کلینیک دارای یک مدیر و تعدادی پرسنل (برای مثال، دامپزشکها، پرستارها، منشی ها، و نظافتچی) است. شماره کلینیک در سرتاسر این تمرین منحصر بفرد است.

پرسنل

جزئیاتی که درباره هر پرسنل ذخیره می شود شامل شماره پرسنلی، نام (نام و نام خانوادگی)، آدرس(خیابان، شهر، ناحیه، و کدپستی)، شماره تلفن، تاریخ تولد، شماره امنیت اجتماعی (SSN)، شغل، و حقوق سالیانه فعلی است. شماره پرسنلی در سرتاسر تمرین منحصر بفرد است.

مالک حیوان خانگی

زمانیکه مالک حیوان خانگی نخست با کلینیک *Perfect Pets* تماس می گیرد جزئیات مالک حیوان خانگی ثبت می شود که شامل شماره مالک، نام مالک (نام و نام خانوادگی)، آدرس(خیابان، شهر، ناحیه و کدپستی) و شماره تلفن منزل می باشد. شماره مالک در سرتاسر کلینیک معینی منحصر بفرد است.

حیوان خانگی

جزئیات حیوان خانگی که به درمان نیاز دارد یادداشت می شود، که شامل شماره حیوان، نام حیوان، نوع حیوان، توضیح، تاریخ تولد (اگر نامشخص باشد، تاریخ تخمینی ثبت می شود)، تاریخ ثبت شده در کلینیک، وضعیت جاری (زنده/ مرده)، و جزئیات مالک حیوان خانگی است. شماره حیوان در کلینیک معینی منحصر بفرد است.

معاینات

وقتی حیوان بیمار به کلینیک آورده می شود، دامپزشک مسئول حیوان را معاینه می کند. جزئیات هر معاینه ثبت می شود که شامل شماره معاینه، تاریخ و زمان معاینه، نام دامپزشک، شماره حیوان، نام حیوان، و نوع حیوان و توضیح کاملی از نتایج معاینه می باشد. شماره معاینه در کلینیک معینی منحصر بفرد است. با توجه به نتیجه معاینه دامپزشک ممکن است معالجه(هایی) را برای حیوان پیشنهاد کند.

معالجه

Perfect Pets معالجه های مختلفی را برای حیوانات متفاوت در نظر گرفته است. این معالجه ها به روش استانداردی در همه کلینکها موجود می باشند. جزئیات هر معالجه شامل شماره معالجه، توضیح کاملی از معالجه و هزینه برای مالک حیوان می باشد. برای مثال، معالجه ها ممکن است شبیه موارد زیر باشند:

T123	دوره آنتی بیوتیک پنیسیلین	$ ۵۰,۰۰
T155	جراحی گربه سانان	$ ۲۰۰,۰۰
T112	دوره واکسیناسیون در برابر آنفولانزای گربه سانان	$ ۷۰,۰۰
T56	نگهداری روزانه از سگهای کوچک (شامل تغذیه)	$۲۰,۰۰

برای هر معاینه هزینه ثابت $۲۰,۰۰ مطالبه می شود، که بعنوان نوع معالجه ثبت می شود. شماره معالجه بطور منحصربفرد هر نوعی از معالجه را مشخص کرده و توسط همه کلینیکهای *Perfect Pets* استفاده می شود.

معالجه های حیوان

بر اساس نتایج حاصل از معاینه حیوان بیمار، دامپزشک ممکن است یک یا چند نوع معالجه را پیشنهاد کند. برای هر نوع معالجه، اطلاعات ثبت شده شامل شماره و تاریخ معاینه، شماره حیوان، نام و نوع، شماره معالجه، توضیح، مقدار هرنوع معالجه، و تاریخ شروع و پایان معالجه می باشد. همچنین هرگونه توضیحات اضافی درباره انواع معالجه ثبت می شود.

آغل

در بعضی موارد، لازم است حیوان بیمار به کلینیک مراجعه شود. هرکلینیک حدود ۲۰ تا ۳۰ محل نگهداری حیوان دارد، که هر یک قادر به نگهداری بین یک تا چهار حیوان می باشد. هر آغل دارای شماره آغل منحصر بفرد، ظرفیت، و وضعیت (نشانه ای از در دسترس پذیر بودن می باشد). حیوان بیمار اختصاص داده شده به آغل و جزئیات آغل، و معاینه های موردنیاز حیوان، و هرگونه توضیح اضافی در مورد حیوان نیز ثبت می شوند. همچنین جزئیات حیوان واقع در آغل نیز یادداشت می شود که، شامل شماره آغل، و تاریخی است که حیوان به/ از آغل وارد یا خارج گشته است. بر اساس بیماری حیوان، احتمالا بیش از یک حیوان در یک زمان در آغل موجود باشند. شماره آغل در همه کلینیک منحصربفرد است.

صورتحساب

مالک حیوان مسئول هزینه معالجه ای است که به حیوان داده شده است. بعد از هر معاینه برای مالک صورتحسابی صادر می شود و جزئیاتی باید برای آن ثبت شود شامل شماره صورتحساب، تاریخ صورتحساب، شماره مالک، نام و آدرس کامل مالک، شماره حیوان، نام حیوان، و جزئیات معالجه داده شده می باشد. صورتحساب، هزینه هر معالجه و مجموع هزینه های همه معالجه های داده شده به حیوان را نیز در بر می گیرد.

اطلاعات اضافی نیز در صورتحساب پرداخت ثبت می شود و شامل تاریخی که صورتحساب پرداخت شده و روش پرداخت (برای مثال، چک، پول نقد، یا کارت اعتباری) می باشد. شماره صورتحساب در سرتاسر تمرین منحصربفرد است.

تدارکات جراحی، غیر جراحی و داروئی

هر کلینیک انباری از تدارکات جراحی (برای مثال، سرنگها، پارچه استریل، و نوار زخم) و تدارکات غیر جراحی (برای مثال، کیفهای پلاستیکی، پیش بند، سطلهای زباله، برچسب نام حیوان، و غذای حیوان) را نگهداری می کند. جزئیات تدارکات جراحی و غیرجراحی شامل شماره و نام تکه، توضیح تکه، تعداد موجود در انبار (این در آخرین روز هر ماه مشخص می شود)، سطح سفارش دوباره، تعداد سفارش دوباره، و هزینه می باشد. شماره تکه به طور منحصر بفرد هر نوع از تدارکات جراحی و غیر جراحی را مشخص می کند. شماره تکه برای هر تکه جراحی و غیر جراحی منحصربفرد بوده و در سرتاسر این تمرین استفاده می شود.

هر کلینیک همچنین انباری از تدارکات داروئی (برای مثال، آنتی بیوتیکها و مسکّن ها) را نیز نگهداری می کند. جزئیات تدارکات داروئی شامل شماره و نام دارو، توضیح، مقدار استعمال دارو، و روش تجویز، و تعداد موجود در انبار، سطح سفارش دوباره و تعداد سفارش دوباره و هزینه می باشد. شماره دارو منحصرا هر نوع از تدارکات دارویی را مشخص می کند. شماره دارو برای هر فراورده داروئی منحصر بوده و در سرتاسر این تمرین استفاده می شود.

قرار ملاقات

اگر بعدها نیاز باشد که حیوان توسط دامپزشک مجددا بررسی شود، به مالک و حیوان قرار ملاقاتی تعیین می شود و جزئیات هر قرار ملاقات ثبت می شود و شامل شماره ملاقات، شماره مالک، نام مالک (نام و نام خانوادگی)، شماره تلفن منزل، شماره حیوان، نام حیوان، نوع حیوان، و ساعت و تاریخ ملاقات می باشد. که شماره ملاقات در کلینیک مشخصی منحصر بفرد است.

۲-۱-۱۸ نیازمندیهای تراکنش

موارد لیست شده زیرین تراکنش هایی هستند که باید بوسیله کاربرد پایگاه داده *Perfect Pets* پشتیبانی شوند.

(۱) تراکنشهای نگهداری

(a) رکوردهایی را ایجاد و نگهداری کنید که جزئیات کلینیک های *Perfect Pets* و اعضای پرسنل هر کلینیک را نگهداری کند.

(b) رکوردهایی را ایجاد کنید که جزئیات مالکان حیوان را نگهداری کند.

(c) جزئیات حیوانات را ایجاد و نگهداری کنید.

(d) رکوردهایی را ایجاد کنید که جزئیات انواع معالجات در دسترس برای حیوان را ثبت و نگهداری کند.

(e) رکوردهایی را ایجاد کنید که جزئیات معاینات و معالجات داده شده به حیوانات را ثبت و نگهداری کند.

(f) رکوردهایی را ایجاد کنید که جزئیات صورتحساب به مالکان حیوانات را جهت معالجه حیوانشان ثبت و نگهداری کند.

(g) رکوردهایی را ایجاد کنید که جزئیات تدارکات جراحی، غیرجراحی و دارویی هر کلینیک را ثبت و نگهداری کند.

(h) رکوردهایی را ایجاد کنید که جزئیات آغلهای موجود در هرکلینیک و تخصیص حیوانات به آغلها را ثبت و نگهداری کند.

(i) قرارهای ملاقات مالک حیوان/ حیوان هر کلینیک را ایجاد و نگهداری کنید.

(۲) تراکنشهای پرس و جو

(a) گزارشی را ارائه کنید که نام مدیر، آدرس کلینیک، و شماره تلفن هر کلینیک را، بترتیب شماره کلینیک لیست کند.
(b) گزارشی را ارائه کنید که نامها و شماره های مالک صاحبان حیوان را به همراه جزئیات حیوان لیست کند.
(c) جزئیات سوابق معاینه ها برای حیوان معینی را لیست کنید.
(d) جزئیات معالجه های داده شده به حیوان بر اساس نتایج معاینه معینی را لیست کنید.
(e) جزئیات صورتحساب پرداخت نشده برای مالک حیوان معینی را لیست کنید.
(f) گزارشی از صورتحساب ها ارائه کنید که در تاریخ معینی پرداخت نشده است، که بترتیب شماره صورتحساب مرتب شده باشد.
(g) جزئیات آغل های در دسترس تاریخ معینی برای کلینیکهای شهر معینی را که با شماره کلینیک مرتب شده باشد را لیست کنید.
(h) گزارشی را ارائه کنید که مجموع حقوق ماهیانه پرسنل هر کلینیک را، که با شماره کلینیک مرتب شده است را فراهم کند.
(i) هزینه حداکثر، حداقل، و متوسط برای معالجه را لیست کنید.
(j) تعداد کل حیوانات موجود در هر نوع را، بترتیب نوع حیوان را لیست کنید.
(k) گزارشی از نامها و شماره های پرسنل برای همه دامپزشکان و پرستاران بالای ۵۰ سال را، که با نام پرسنل مرتب شده باشند را لیست کنید.
(l) قرارهای ملاقات برای تاریخ معینی و برای کلینیک مشخصی را لیست کنید.
(m) تعداد کلی حیوانات موجود در هر کلینیک که با شماره کلینیک مرتب شده است را لیست کنید.
(n) گزارشی از جزئیات صورتحساب برای مالکان حیوان بین ۱۹۹۷ تا ۱۹۹۹، مرتب شده با شماره صورتحساب را ارائه کنید.
(o) شماره حیوان، نام، و توضیحی از حیوان که توسط مالک معینی تصاحب شده است را لیست کنید.
(p) گزارشی را ارائه کنید که تدارکات داروئی که در هرکلینیک نیاز به سفارش دوباره دارند را، به ترتیب شماره کلینیک لیست کند.
(q) هزینه کلی تدارکات غیر جراحی و جراحی در انبار هر کلینیک را، بترتیب شماره کلینیک لیست کنید.

۲-۱۸ استفاده از متدلوژی طراحی منطقی پایگاه داده

در این بخش، قصد داریم تا از طریق مراحل موجود در متدلوژی طراحی منطقی پایگاه داده به مدل داده منطقی قابل قبولی که نیازمندیهای بالا را برای *Perfect Pets* برآورده سازد برسیم. در اینجا فرض خواهیم کرد که مرحله تحلیل و جمع آوری نیازمندیها فقط در یک دید کاربر مشخص شده است.

مرحله ۱-۱ موجودیتها را شناسایی کنید

مرحله اول در طراحی منطقی پایگاه داده مشخص کردن موجودیتهای اصلی ای است که شما بایستی آنها را در پایگاه داده نشان دهید. از توضیحاتی در بالا برای *Perfect Pets* داده شد، میتوانید موجودیتهای زیر را مشخص کنید:

Clinic Staff
PetOwner Pet
Examination Treatment
Pen PetTreatment
Invoice Appointment
Stock (with specializations Surgical, NonSurgical , and Pharmaceuticals)

موجودیتها را مستند کنید

بعد از شناسایی موجودیتها، به آنها نامی بدهید که برای کاربر با معنی و واضح باشد، و این اطلاعات را در دیکشنری داده ثبت کنید. شکل ۱-۱۸ نسخه خلاصه ای از دیکشنری داده را که موجودیتها را برای *Perfect Pets* مستند کرده است نشان میدهد.

شکل ۱-۱۸

خلاصه ای از دیکشنری داده *Perfect Pets* که موجودیتها را توصیف می کند.

Entity name	Description	Aliases	Occurrence
Clinic	Veterinary clinics.	Surgery	One or more *Perfect Pet* clinics located in main cities throughout the US.
Staff	General term describing all staff employed by *Perfect Pets*.	Vet, Nurse, Secretary	Each member of staff works at a particular clinic.
PetOwner	Owners of pets taken to *Perfect Pets*.		Owner takes his/her pet to a particular clinic.

شکل ۲-۱۸

اولین پیش نویس رابطه های *Perfect Pets*.

Entity	Relationship	Entity
Clinic	Has	Staff
	Holds	Stock
	Registers	Pet
	Provides	Pen
	Schedules	Appointment
	IsContactedBy	PetOwner
Staff	Manages	Clinic
	Performs	Examination
PetOwner	Owns	Pet
	Pays	Invoice
	Attends	Appointment
Pet	Undergoes	Examination
	IsAllocatedTo	Pen
	Attends	Appointment
Examination	ResultsIn	PetTreatment
Treatment	UsedIn	PetTreatment
Invoice	ResultsFrom	Examination

شکل ۳-۱۸ محدودیتهای کثرت برای رابطه های شناسایی شده در بالا.

Entity	Multiplicity	Relationship	Entity	Multiplicity
Clinic	1..*	Has	Staff	1..1
	1..*	Holds	Stock	1..*
	1..*	Registers	Pet	1..1
	20..30	Provides	Pen	1..1
	1..*	Schedules	Appointment	1..1
	1..*	IsContactedBy	PetOwner	1..1
Staff	0..1	Manages	Clinic	1..1
	0..*	Performs	Examination	1..1
PetOwner	1..*	Owns	Pet	1..1
	1..*	Pays	Invoice	1..1
	1..*	Attends	Appointment	1..1
Pet	1..*	Undergoes	Examination	1..1
	0..*	IsAllocatedTo	Pen	1..*
	1..*	Attends	Appointment	1..1
Examination	1..*	ResultsIn	PetTreatment	1..1
Treatment	1..*	UsedIn	PetTreatment	1..1
Invoice	1..1	ResultsFrom	Examination	1..1

مرحله ۲-۱ رابطه ها را مشخص کنید

بعد از شناسایی موجودیتها، مرحله بعد مشخص کردن همه رابطه هایی است که بین موجودیتها وجود دارد. برای *Perfect Pets*، ممکن است رابطه های نشان داده شده در شکل ۲-۱۸ را مشخص کنید.

محدودیتهای کثرت رابطه ها را مشخص کنید

بعد از شناسایی روابطی که می خواهید مدل کنید، حال بایستی کثرت رابطه را معین کنید. برای *Perfect Pets* شما باید محدودیتهای کثرت نشان داده شده در شکل ۳-۱۸ را مشخص سازید.

استفاده از مدلسازی موجودیت- رابطه (ER)

در سرتاسر مرحله طراحی پایگاه داده نسخه های متفاوتی از مدل ER را که نشان دهنده *Perfect Pets* است را ایجاد خواهید کرد. شکل ۴-۱۸ طرح اولیه مدل ER برای *Perfect Pets* را نشان می دهد.

مرحله ۳-۱ صفتها را مشخص کرده و با موجودیتها یا رابطه ها مرتبط سازید

مرحله بعد شناسایی صفات مرتبط با موجودیتها و رابطه هایی است که مشخص کرده اید. برای *Perfect Pets* شما باید صفات موجودیتهای مربوط را، همانطور که در شکل ۵-۱۸(الف) نشان داده شده اند مشخص کنید.

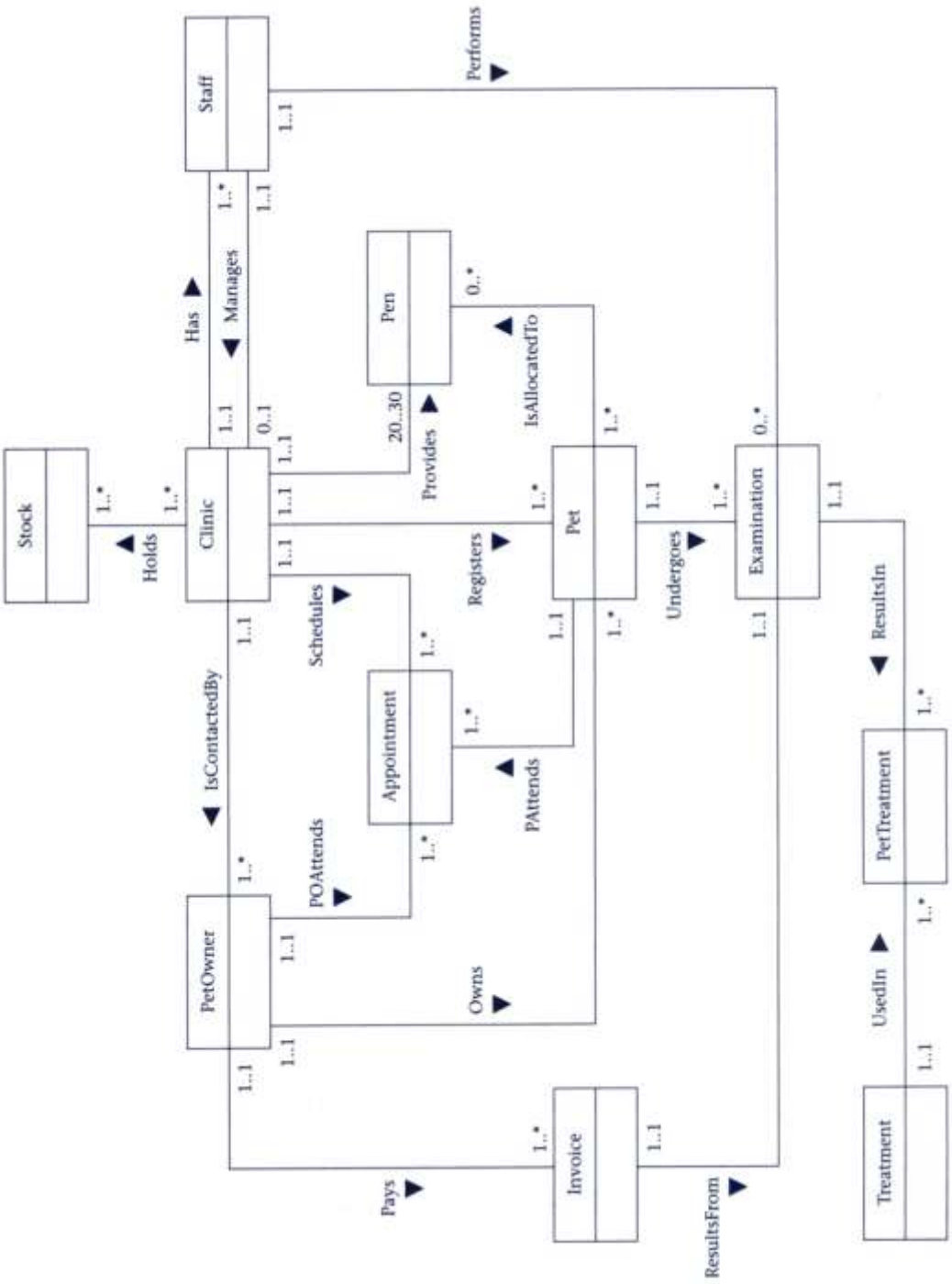
با این وجود، زمانیکه اطلاعات آغل ها را بررسی می کنید، ممکن است در مرتبط کردن صفات dateIn/ dateOut که تاریخی را که حیوان به آغل وارد/ خارج می شود و صفت comments با موجودیت Pen یا Pet دارد دچار مشکل شوید. به طور مشابه، ممکن است در مرتبط ساختن صفت inStock که مقدار موجود در انبار را برای فهرستهای متفاوتی از تدارکات بیان می کند و صفتهای reorderLevel/ reorderQty را با موجودیتهای Clinic یا Stock به مشکل برخورید. در هر دو این موارد، همانطور که در شکل۵-۱۸(ب) نشان داده شده است باید موجودیت جدیدی را مشخص کرده یا صفات را با رابطه های متناظر مربوط سازید.

شما همچنین باید مراقب باشید که موقعیکه رخداد صفت بطور واقعی رابطه بین موجودیتها را بیان می کند صفت مشابهی را در دو موجودیت در نظر نگیرید. برای مثال، در خصوصیات نیازمندیهای داده شده در بخش ۱-۱-۱۸، در معاینات مقرر شده است که جزئیات هر معاینه شامل 'نام دامپزشک' نیز باشد. در اینجا ممکن است گمراه شوید که نام دامپزشک را در هر دو موجودیت Staff و Examination بیاورید. بنابراین، این اشتباه است: نمایش نام دامپزشک در این موقعیت رابطه را بیان می کند و شما نباید آنرا بعنوان صفتی از Examination در نظر بگیرید. اگر اینکار را انجام دهید، منجر به این خواهد شد که جدول Examination در فرم نرمال سوم (3NF) نباشد.

نکته این یک اشتباه عمومی است که طراحان بی تجربه معمولا مرتکب میشوند. بنابراین باید مراقب باشید.

شکل ۴-۱۸

طرح اولیه مدل ER برای *Perfect Pets*.



شکل ۵-۱۸

صفات *Perfect Pets*: (الف) صفات مرتبط با موجودیتها; (ب) صفات مرتبط با رابطه ها.

Entity	Attributes
Clinic	clinicNo, address (street, city, state, zipCode), telNo, faxNo
Staff	staffNo, sName (sFName, sLName), sAddress (sStreet, sCity, sState, sZipCode), sTelNo, DOB, sex, SSN, position, salary
PetOwner	ownerNo, oName (oFName, oLName), oAddress (oStreet, oCity, oState, oZipCode), oTelNo
Pet petStatus	petNo, petName, petType, petDescription, pDOB, dateRegistered,
Examination	examNo, examDate, examTime, examResults
Treatment	treatNo, description, cost
Pen	penNo, penCapacity, penStatus
Invoice	invoiceNo, invoiceDate, datePaid, paymentMethod
Stock: Item	itemNo, itemName, itemDescription, itemCost
Stock: Pharmacy drugCost	drugNo, drugName, drugDescription, dosage, methodAdmin,
Appointment	appNo, aDate, aTime
PetTreatment	startDate, endDate, quantity, ptComments

(الف)

Relationship	Attributes
IsAllocatedTo	dateIn, dateOut, comments
Holds	inStock, reorderLevel, reorderQty

(ب)

مرحله ٤-١ دامنه های صفت را مشخص کنید

شما حال باید دامنه های لازم را برای پشتیبانی از صفاتی که در مرحله قبل در دیکشنری داده مشخص کردید اضافه کنید.

مرحله ٥-١ صفتهای کلید اصلی و کاندید را مشخص کنید

این مرحله در رابطه با شناسایی کلید(های) کاندید موجودیت و سپس انتخاب یکی از آنها بعنوان کلید اصلی است. در فرایند شناسایی کلیدهای اصلی، توجه کنید که آیا موجودیت قوی است یا ضعیف.

هنگام مشخص کردن کلیدهای کاندید، باید توجه کنید که شماره کلینیک موجودیت Clinic، شماره پرسنل موجودیت Staff، شماره معالجه موجودیت Treatment، شماره صورتحساب موجودیت Invoice، و شماره تکه/ دارو موجودیتهای Stock در سرتاسر تمرین منحصر بفرد باشد. از طرف دیگر، شماره مالک موجودیت PetOwner، شماره حیوان موجودیت Pet، و شماره آغل موجودیت Pen فقط برای کلینیک معینی منحصر بفرد باشد. شرکتها معمولا به قسمتهای مختلف درجاتی از خود مختاری را می دهند. بنابراین در سیستم پایگاه داده متمرکز شده بعضی اوقات مناسبتر است که در همه شرکت یکتایی را داشته باشیم. در بحث با مدیریت *Perfect Pets* موافقت شده است که همه شماره ها بایستی بجای هر کلینیک به کل دوره طبابت اختصاص یابند. در صورت عملی نشدن این تصمیم، لازم خواهد بود که شماره کلینیک را به آن شماره هایی که تنها در کلینیک منحصر هستند جهت بدست آوردن یکتایی عمل بیافزاییم.

با در نظر گرفتن این موضوع، هم اکنون باید کلیدهای اصلی نشان داده شده در شکل ۶-۱۸ را مشخص کنید (دیگر کلیدها نیز در شکل ۹-۱۸ نشان داده شده اند). خصوصا، بایستی PetTreatment را بعنوان موجودیت ضعیف مشخص کرده باشید.

مرحله ٦-١ موجودیتها را تخصص/ تعمیم دهید (مرحله اختیاری)

مشخص ساختن موجودیت Stock به شکل کنونی کاملا جهت ادامه دادن متدلوژی طراحی منطقی پایگاه داده قابل قبول است. با این وجود، اطلاعات دیگری وجود دارد که شما می خواهید جهت هر چه بیشتر دقیق شدن مدلتان به آن بیافزایید. خصوصیات نیازمندیها بیان می کند که جهت تمایز بین تدارکات جراحی و غیر جراحی شماره تکه منحصری وجود دارد. همچنین تدارکات دارویی وجود دارد که دارای شماره داروئی منحصری است. بعلاوه این دو نوع از تدارکات دارای صفاتی هستند که اندکی با هم تفاوت دارند. بنابراین، ما می توانیم Surgical/ Non-Surgical Stock و Pharmaceutical را از انواع مشخصی از موجودیت Stock در نظر بگیریم. این تخصص/ تعمیم در شکل ۶-۱۸ نشان داده شده است. برای سادگی، Surgical/ Non-Surgical Stock را بعنوان item و Pharmaceutical را بعنوان Pharmacy نامیده ایم.

همچنین ممکن است Vet، Nurse، Secretary، و Cleaner را بعنوان نوع مشخصی از Stock در نظر بگیرید. اگر چه این عناوین شغل همه دارای صفات مشابهی هستند تنها موجودیت Vet است که باید در رابطه Performs با Examination شرکت کند. این روش، کاملا روش معتبری از مدلسازی پرسنل است. با این وجود، برای ساده نگه داشتن مدل، مرحله تخصص/ تعمیم را حذف کرده ایم.

مرحله ۷-۱ خصوصیاتی را که با مدل رابطه ای سازگار نیستند حذف کنید

در این نقطه، شما حالا مدل داده منطقی محلی برای *Perfect Pets* دارید. با این وجود، مدل داده ای شامل بعضی ساختار های داده ای است که به آسانی بوسیله DBMS های رابطه ای قابل مدلسازی نیست. در این مرحله، چنین ساختارهای داده ای را به اشکالی تبدیل می کنید که به آسانی توسط سیستمهای رابطه ای پردازش شوند. خصوصا شما بایستی:

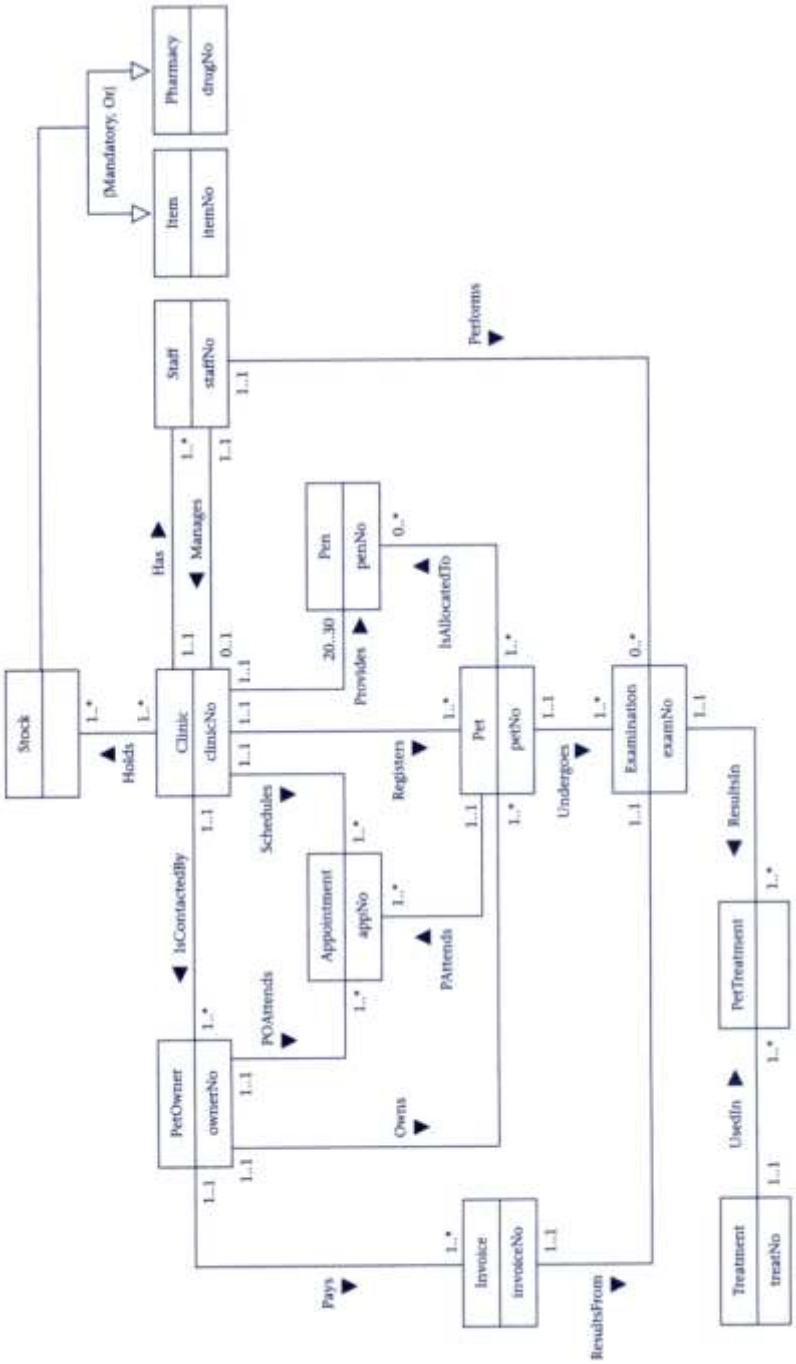
۱) رابطه های چند به چند باینری را حذف کنید.
۲) رابطه های بازگشتی چند به چند (*:*) را حذف کنید.
۳) رابطه های پیچیده را حذف کنید.
۴) رابطه های چند مقداره را حذف کنید.
۵) رابطه های یک به یک (1:1) را دوباره بررسی کنید.
۶) رابطه های افزونه را حذف کنید.

رابطه های چند به چند (*:*)

از شکل ۶-۱۸، دو رابطه چند به چند وجود دارد: Clinic *Holds* Stock و Pet *IsAllocatedTo* Pen. در هر دو مورد، رابطه ها، همچنانکه در شکل ۵-۱۸(ب) نشان داده شده است، دارای صفات مربوط به آنها هستند. برای مدل کردن صحیح این موقعیتها در مدل رابطه ای، مجبورید که موجودیتهای میانی را در نظر بگیرید، که ما ClinicStock و PetPen می نامیم، که در شکل ۷-۱۸ نشان داده شده است.

شکل ۶-۱۸

مدل ER برای Perfect Pets با کلیدهای اصلی نشان داده شده و تخصص/ تعمیم stock.



رابطه یک به یک (۱:۱)

از شکل ۴-۱۸، دو رابطه یک به یک وجود دارد: Staff *Manages* Clinic و Invoice *ResultsFrom* Examination با این وجود، در هر دو مورد این دو موجودیت بطور آشکارا از هم مجزا بوده و نباید با هم ادغام شوند.

رابطه های افزونه

از شکل ۶-۱۸، تعدادی رابطه بین PetOwner ، Pet، Clinic، و Appointment وجود دارند ، و بررسی نزدیکتر رابطه ها برای مشخص کردن هر گونه رابطه افزونه ای می تواند مفید باشد. اول از همه، توجه کنید که موجودیتهای PetOwner/Pet شرکت اجباری در رابطه های Owns /PAttends/ POAttends داشته، و بدین معنی که PetOwner ممکن است مالک چندین حیوان باشد. بنابراین، برای هر Appointment معین می توانیم مالک را از طریق رابطه POAttends شناسایی کنیم، اما ما نمی توانیم Pet را از طریق رابطه Owns مشخص کنیم. با این وجود، برای هر Appointment معین ما می توانیم Pet را از طریق رابطه POAttends شناسایی کرده و برای هر Pet معین PetOwner را از طریق رابطه Owns مشخص کنیم،که این به ما می گوید که رابطه PAttends افزونه است. به روش مشابه، از طریق رابطه PAttends میتوانیم Pet را مشخص کنیم و از طریق رابطه Registers می توانیم کلینیک شامل شده در Appointment را مشخص کنیم که این به ما می گوید که رابطه Schedules نیز یک رابطه افزونه است.

توجه داشته باشید که رابطه IsContactedBy بین Clinic و PetOwner همچنین رابطه ای افزونه بنظر می آید. با این وجود، *Perfect Pets* جزئیات مالکان حیوان را بهنگام اولین تماس یادداشت می کند و فقط جزئیات حیوانات را در اولین قرار ملاقات بدست می آورد، و بنابراین رابطه های IsContactedBy حفظ می شود. مدل داده منطقی اصلاح شده در شکل ۷-۱۸ نشان داده شده است.

مرحله ۸-۱ مدل را جهت پشتیبانی از تراکنشهای کاربر بررسی کنید

در این مرحله، شما مدل داده منطقی محلی را که توسعه داده اید را بررسی می کنید تا مشخص کنید که مدل از تراکنشهای مشخص شده کاربران پشتیبانی کند. که این شامل بررسی موارد زیر است:

- صفات مورد نیاز که در مدل داده وجود دارد، و
- اینکه صفتها کجا در بیش از یک موجودیت ظاهر شده اند، یعنی مسیری بین این دو موجودیت وجود دارد؛ بعبارت دیگر، رابطه مشخص شده ای که مستقیم یا غیرمستقیم بین دو موجودیت وجود دارد.

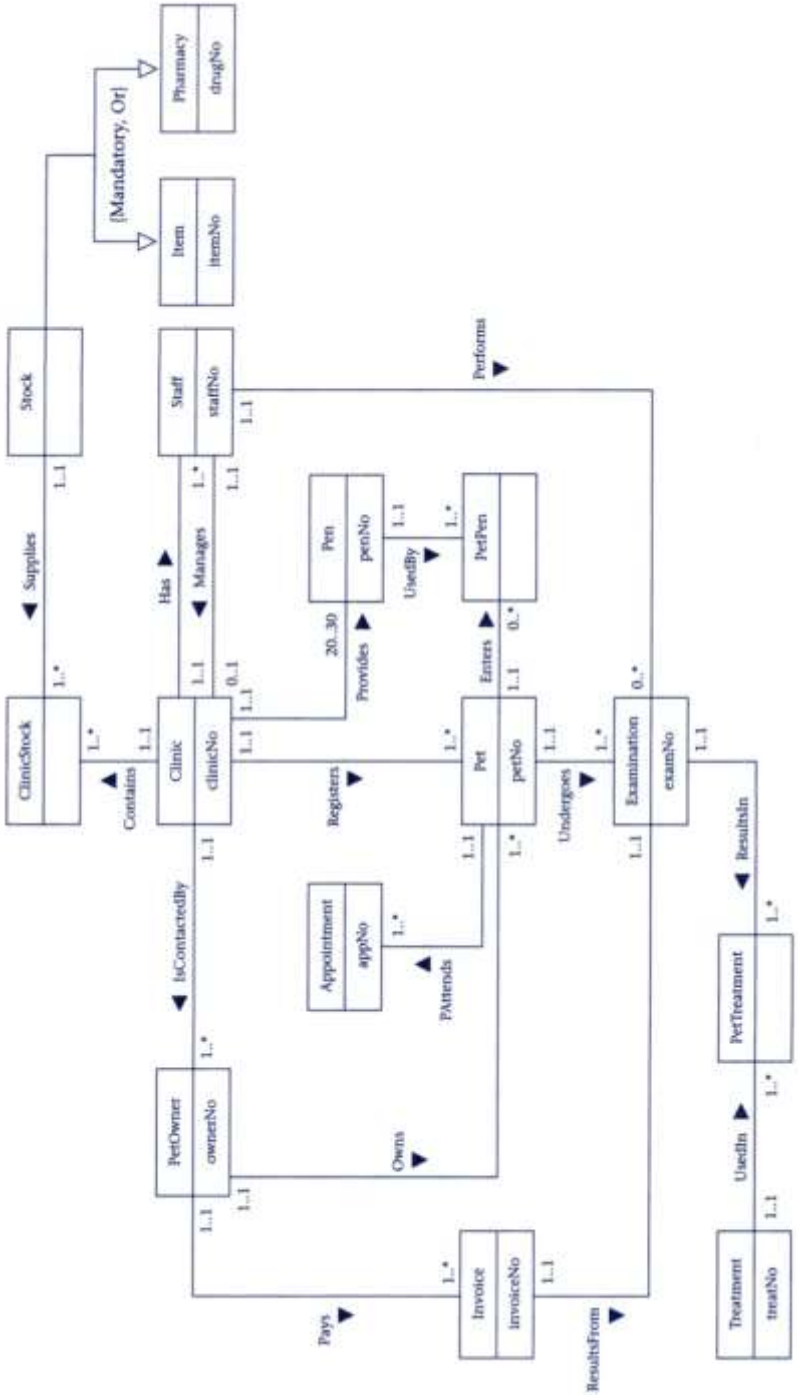
نمودار مسیر تراکنش برای تراکنشهای پرس وجوی مشخص شده در بخش ۲-۱-۱۸ در شکل ۸-۱۸ نشان داده شده است، و شما می توانید بررسی کنید که آیا صفات مورد نیاز از موجودیت منفرد دسترس پذیر هستند و یا از موجودیتهای چندگانه از طریق یک یا چند رابطه در دسترس پذیر باشند.

مرحله ۱-۲ جدولها را برای مدل داده منطقی ایجاد کنید

در این مرحله، شما جدول هایی را برای مدل داده منطقی محلی جهت نمایش موجودیتها و رابطه ای نشان داده شده در دید کاربر این تمرین با استفاده از زبان تعریف پایگاه داده (DBDL) برای پایگاه داده های رابطه ای ایجاد می کنید.

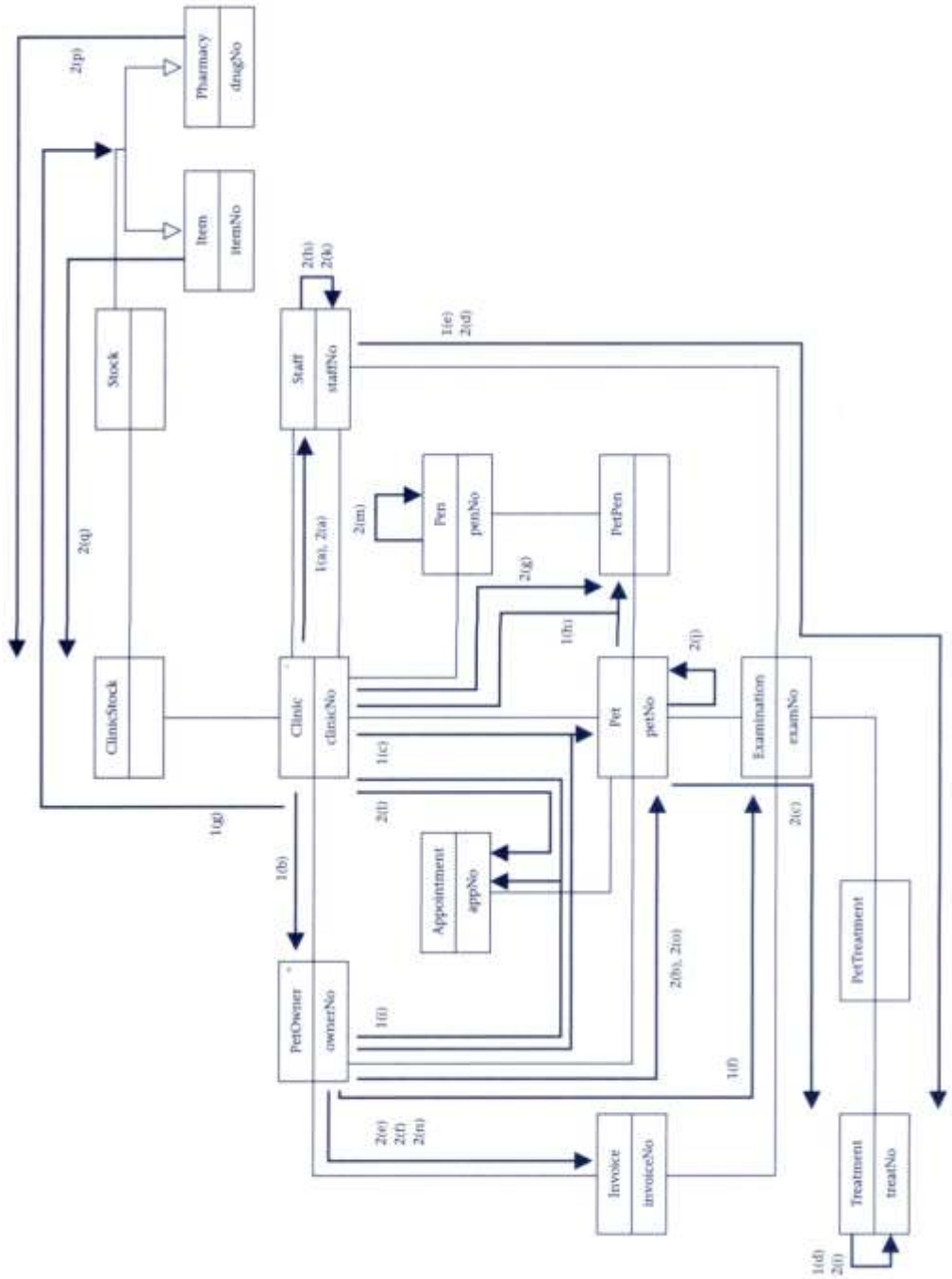
شکل ۷-۱۸

مدل داده منطقی محلی اصلاح شده *Perfect Pets* بعد از برداشتن مشخصه هایی که با مدل رابطه ای سازگار نیستند.



شکل ۸-۱۸

نمودار مسیر تراکنش *Perfect Pets* .



شکل ۹-۱۸

جداولی که از مدل داده منطقی محلی برای *Perfect Pets* ایجاد شده اند.

Clinic (clinicNo, street, city, state, zipcode, telNo, faxNo, mgrStaffNo) **Primary Key** clinicNo **Alternate Key** zipCode **Alternate Key** telNo **Alternate Key** faxNo **Foreign Key** mgrStaffNo **references** Staff(staffNo)	**Staff** (staffNo, sFName, sLName, sStreet, sCity, sState, sZipCode, sTelNo, DOB, sex, SSN, position, salary, clinicNo) **Primary Key** staffNo **Alternate Key** SSN **Foreign Key** clinicNo **references** Clinic(clinicNo)
PetOwner (ownerNo, oFName, oLName, oState, oZipCode, oTelNo, clinicNo) **Primary Key** ownerNo **Foreign Key** clinicNo **references** Clinic(clinicNo)	**Pet** (petNo, petName, petType, petDescription, pDOB, dateRegistered, petStatus, ownerNo, clinicNo) **Primary Key** petNo **Foreign Key** ownerNo **references** Owner(ownerNo) **Foreign Key** clinicNo **references** Clinic(clinicNo)
Examination (examNo, examDate, examTime, examResults, petNo, staffNo) **Primary Key** examNo **Alternate Key** staffNo, examDate, examTime **Foreign Key** petNo **references** Pet(petNo) **Foreign Key** staffNo **references** Staff(staffNo)	**Treatment** (treatNo, description, cost) **Primary Key** treatNo
Pen (penNo, penCapacity, penStatus, clinicNo) **Primary Key** penNo **Foreign Key** clinicNo **references** Clinic(clinicNo)	**PetPen** (penNo, petNo, dateIn, dataOut, comments) **Primary Key** penNo, petNo, dateIn **Alternate Key** penNo, petNo, dateOut **Foreign Key** penNo **references** Pen(penNo) **Foreign Key** petNo **references** Pet(petNo)
PetTreatment (examNo, treatNo, startDate, endDate, quantity, ptComments) **Primary Key** examNo, treatNo **Foreign Key** examNo **references** Examination(examNo) **Foreign Key** treatNo **references** Treatment(treatNo)	**Item** (itemNo, itemName, itemDescription, itemCost) **Primary Key** itemNo
Pharmacy (drugNo, drugName, drugDescription, dosage, methodAdmin, drugCost) **Primary Key** drugNo	**ItemClinicStock** (itemNo, clinicNo, inStock, reorderLevel, reorderQty) **Primary Key** itemNo, clinicNo **Foreign Key** itemNo **references** Item(itemNo) **Foreign Key** clinicNo **references** Clinic(clinicNo)
PharmClinicStock (drugNo, clinicNo, inStock, reorderLevel, reorderQty) **Primary Key** drugNo, clinicNo **Foreign Key** drugNo **references** Pharmacy(drugNo) **Foreign Key** clinicNo **references** Clinic(clinicNo)	**Invoice** (invoiceNo, invoiceDate, datePaid, paymentMethod, ownerNo, examNo) **Primary Key** invoiceNo **Foreign Key** ownerNo **references** Owner(ownerNo) **Foreign Key** examNo **references** Examination(examNo)
Appointment (appNo, aDate, aTime, petNo) **Primary Key** appNo **Foreign Key** petNo **references** Pet(petNo)	

جداول و صفات کلید خارجی را مستند کنید

در پایان مرحله ۱-۲، ترکیب کاملی از جدولهای ایجاد شده از مدل داده منطقی محلی را مستند می کنید. هر جدول با استفاده از DBDL توضیح داده می شود، همانطورکه در شکل ۹-۱۸ نشان داده شده است.

مرحله ۲-۲ مدل را با استفاده از نرمالسازی بررسی کنید

در این مرحله، شما بایستی مطمئن شوید که همه جداول ایجاد شده در مرحله قبل حداقل در فرم نرمال ۳ باشد (3NF).
اگر جداولی را مشخص کنید که در 3NF نباشد ممکن است بیانگر این باشد که قسمتی از مدل داده منطقی اشتباه است یا، زمانیکه جدولها را از مدل مشتق می کردید دچار اشتباه شده اید. با این وجود، شما می توانید بسهولت بررسی کنید که جدولهای مشخص شده در شکل ۹-۱۸ در 3NF است.

مرحله ۳-۲ مدل را در برابر تراکنشهای کاربر بررسی کنید

این مرحله مشابه مرحله ۸-۱ است، به استثنای اینکه شما نگاشتی از موجودیتها به جدولها و فرستادن کلیدهای اصلی را جهت عمل بعنوان کلید خارجی به طور صحیح بررسی می کنید. در این مورد، شما دوباره می توانید بسهولت بررسی کنید که نگاشت به طور صحیح صورت گرفته باشد و جدولها پشتیبانی از تراکنشهای کاربر معین بخش ۲-۱-۱۸ را بخوبی انجام دهند.

مرحله ٤-۲ محدودیتهای جامعیت را تعریف کنید

محدودیتهای جامعیت محدودیت هایی هستند که شما می خواهید اعمال کنید تا پایگاه داده را از ناسازگار شدن حفاظت کند. از پنج نوع محدودیت جامعیت، سه تای آنها در مرحله قبل شناسایی شده و در دیکشنری داده ثبت شده اند، که بنامهای: داده موردنیاز، محدودیتهای دامنه صفت، و جامعیت موجودیت می باشد. ما دو تای باقی را اینجا در نظر می گیریم : جامعیت ارجاع و قواعد تجاری.

جامعیت ارجاع

اینجا دو موضوع وجود دارد که در نظر می گیریم:

(۱) *مشخص کنید که آیا Null ها برای کلید خارجی مجاز هستند.* بطورکلی، اگر شرکت جدول فرزند در رابطه اجباری باشد در این صورت راهبرد این است که Null نباید مجاز باشد. از طرف دیگر، اگر شرکت جدول فرزند اختیاری باشد، در این صورت Null میتواند مجاز باشد.

(۲) *محدودیتهای وجود را تحت اینکه کدام کلید خارجی شاید درج، بهنگام، و حذف شده باشد مشخص کنید.* بطورکلی این شامل مشخص کردن دو عمل برای هر کلید خارجی می باشد: عمل ON UPDATE و عمل ON DELETE ، مربوط به اینکه اگر رکورد در جدول والد بهنگام/ حذف شود چه چیزی برای حفظ جامعیت ارجاع باید اتفاق بیافتد. شکل ۱۰-۱۸ اعمال لازم برای کلیدهای خارجی مشخص شده در شکل ۹-۱۸ نشان می دهد.

قواعد تجاری

بالاخره، در نظر بگیرید که آیا انواع دیگری از محدودیتهایی که *Perfect Pets* تعریف کرده وجود دارد که در جایی دیگر از مدل داده ای پوشش داده نشده باشد. چنین محدودیتهایی بطور کلی **قواعد تجاری** نامیده میشوند.

همه محدودیتهای جامعیت را مستند کنید

همه محدودیتها بایستی جهت در نظر گرفتن طی طراحی فیزیکی پایگاه داده در دیکشنری داده مستند شوند.

مرحله ۵-۲ مدل داده منطقی محلی را با کاربران بازبینی کنید

مدل داده منطقی محلی بایستی حال تکمیل شده و کاملا مستند شده باشد. در این نقطه، باید مدل و مستندات پشتیبانی کننده را با کاربران بازبینی کنید. ما فرض خواهیم کرد که این هیچ نوع تغییر ی را در طراحی ایجاد نمی کند. همانطور که در شروع این فصل گفتیم، فرض کردیم که فقط یک دید کاربر وجورد دارد و بنابراین مرحله ۳ غیرلازم است. بنابراین، این متدلوژی طراحی پایگاه داده *Perfect Pets* را تکمیل می کند. در مرحله بعد، ما به فاز طراحی فیزیکی پایگاه داده خواهیم پرداخت.

فصل ۱۹

Perfect Pets – طراحی فیزیکی پایگاه داده

در این فصل، به مرحله طراحی فیزیکی پایگاه داده بررسی موردی پایگاه داده *Perfect Pets* که در فصل قبل معرفی شد می پردازیم. جهت نشان دادن بعضی از جنبه های پیاده سازی فیزیکی از اوراکل ۸ استفاده می کنیم. همانطور که در آغاز فصل قبل یادآوری کردیم، شما می توانید مراحل متدلوژی را برای خودتان تمرین کنید بعد راه حل خود را با راه حل ما بسنجید که می تواند جهت تسلط هر چه بیشتر به مطلب مفید واقع شود. همچنین خلاصه متدلوژی را نیز می توانید در ضمیمه **ب** بیابید.

۱۹-۱ استفاده از متدلوژی طراحی فیزیکی پایگاه داده

در این بخش، به مراحل موجود در متدلوژی طراحی فیزیکی پایگاه داده جهت رسیدن به طراحی فیزیکی مناسب و پیاده سازی پایگاه داده *Perfect Pets* می پردازیم.

مرحله ۱-٤ جداول پایه را در DBMS هدف طراحی کنید

هنگام طراحی منطقی پایگاه داده برای نشان دادن موجودیتها و رابطه های مدل داده منطقی تعدادی **جدول پایه** ایجاد کردید. همچنین به همراه این جداول مجموعه ای از مستندات پشتیبانی کننده ای را با جزئیات زیر تولید کردید:

- برای هر جدول، صفات آنها، کلیدهای اصلی، فرعی، و خارجی ، و محدودیتهای جامعیت ؛
- برای هر صفت، دامنه آنها، مقدار پیش فرض اختیاری، و اینکه آیا Null ها در بر می گیرد، و آیا صفت مشتق شده است؛

برای نشان دادن طراحی جداول پایه، از این اطلاعات برای تعریف دامنه ها، مقادیر پیش فرض و شاخص Null استفاده می کنید. برای مثال، برای جدول Pen پایگاه داده *Perfect Pets* ، احتمالا طراحی نشان داده شده در شکل ۱-۱۹ را با استفاده از زبان طراحی پایگاه داده توسعه یافته (DBDL) تولید کرده اید. همچنین از این اطلاعات برای مشخص کردن اینکه چگونه جداول پایه را در DBMS هدف، که در این مورد اوراکل ۸ خواهد بود پیاده سازی کنید، استفاده می کنید.

ایجاد جداول پایه در اوراکل ۸

در بعضی از سیستمها که کاملا با SQL استاندارد ۱۹۹۲ (SQL2) سازگار نشده اند، هیچگونه پشتیبانی برای عبارت های PRIMARY KEY، FOREIGN KEY، DEFAULT وجود ندارد. مشابها، بعضی سیستمها هم دامنه ها را پشتیبانی نمی کنند. با این وجود، اوراکل ۸ تعدادی از شرطهای SQL2 CREATE TABLE را پشتیبانی می کند، بنابراین شما می توانید موارد زیر را تعریف کنید:

- کلیدهای اصلی، را با استفاده از شرط PRIMARY KEY؛
- کلیدهای فرعی، را با استفاده از کلمه کلیدی UNIQUE؛
- مقادیر پیش فرض، را با استفاده از شرط DEFAULT؛
- ستونهای ناتهی، را با استفاده از کلمه کلیدی NOT NULL؛
- کلیدهای خارجی، را با استفاده از شرط FOREIGN KEY؛
- دیگر محدودیتهای جدول یا ستون را با استفاده از شرطهای CHECK و CONSTRAINT؛

با این وجود، برای ایجاد دامنه ها امکاناتی در اوراکل ۸ وجود ندارد. اگرچه اوراکل ۸ اجازه می دهد که انواع تعریف شده توسط کاربر، ایجاد شوند. بعلاوه، انواع داده ای اندکی متفاوت از SQL استاندارد است، همچنانکه در شکل ۱-۱۹ نشان داده شده است.

در فصل ۱۲، دیدیم که مایکروسافت اکسس نوع داده Autonumber داشت که هر وقت که رکورد درج می شد اعداد متوالی جدید را بعنوان مقدار ستون ایجاد می کرد. اوراکل چنین داده ای را ندارد، اما در عوض امکان مشابهی از طریق دستور (غیراستاندارد) SQL CREATE SEQUENCE دارد. برای مثال، دستور:

```
CREATE SEQUENCE appnoseq
START WITH 1 INCREMENT BY 1 CACHE 30;
```

دنباله ای، بنام appNoSeq ، ایجاد می کند که با مقدار اولیه ۱ شروع شده و هر بار یک مرتبه افزایش می یابد. شرط CACH 30 مشخص می کند که باید اوراکل ۳۰ عدد متوالی را در ابتدا اختصاص دهد و برای دستیابی سریعتر آنها را در حافظه نگه دارد.

شکل ۱-۱۹

DBDL برای جدول Pen.

```
domain Pen_Numbers          fixed length character string length 4
domain Pen_Capacity         integer value, between 1 and 4
domain Pen_Status           one character, indicating whether pen
                            is available (A) or not available (N)
domain Clinic_Numbers       fixed length character string length 5

pen(  penNo          Pen_Numbers      NOT NULL,
      penCapacity    Pen_Capacity     NOT NULL DEFAULT 2,
      penStatus      Pen_Status       NOT NULL DEFAULT 'A',
      clinicNo       Clinic_Numbers   NOT NULL)
      Primary Key penNo
      Foreign Key clinicNo References Clinic(clinicNo)ON UPDATE CASCADE ON DELETE NO ACTION
```

جدول ۱-۱۹ فهرست بخشی از انواع داده موجود در اوراکل.

نوع داده	استفاده	اندازه
char(size)	داده کاراکتری با طول ثابت را ذخیره می کند. (پیش فرض ۱)	بیش از ۲۰۰۰ بایت
nchar(size)	مشابه نوع داده char، بجز اینکه ماکزیمم طول توسط مجموعه کاراکتر پایگاه داده مشخص میشود (برای مثال، American English، Eastern European، یا Korean).	
varchar2(size)	داده کاراکتری با طول متغیر را ذخیره می کند.	بیش از ۴۰۰۰ بایت
nvarchar2(size)	مشابه varchar2 با پیش بینی برای نوع داده nchar.	
Varchar	در حال حاضر مشابه char. با این وجود، استفاده از varchar2 پیشنهاد میشود از آنجائیکه varchar ممکن است با مقایسه معنایی متفاوت در نسخه های بعدی نوع داده مجزایی بشود.	بیش از ۲۰۰۰ بایت
number(l,d)	اعداد ممیز ثابت و ممیز شناور را ذخیره می کند، جاییکه l نشانگر طول و d تعداد حرف اعشار را نشان میدهد. برای مثال، number(5,2) شامل چیزی بیش از ۹۹۹,۹۹ نیست.	1.0E-130..9.99E125
decimal(l,d), dec(l,d) , یا numeric(l,d)	مشابه number. بخاطر سازگاری با SQL استاندارد فراهم شده است.	
integer, int, smallint	بخاطر سازگاری با SQL استاندارد فراهم شده است.	
Date	تاریخ را از 1 Jan 4712 BC تا 31 Dec A.D. 4712 ذخیره می کند.	
Blob	شی بزرگ باینری.	بیش از ۴ گیگا بایت.
Clob	شی بزرگ کاراکتری.	بیش از ۴ گیگابایت
raw(size)	داده باینری خام، همچون ترتیبی از گرافیکها، کاراکترها یا تصاویر دیجیتال شده .	بیش از ۲۰۰۰ بایت

هر بار که دنباله ایجاد می شود، می توانید به مقادیر آن در دستورات SQL با شبه ستونهای زیرین دسترسی بیابید:

CURRVAL که مقدار کنونی دنباله را بر میگرداند
NEXTVAL که دنباله را افزایش داده و مقدار جدیدی را بازمیگرداند

برای مثال دستور SQL زیر:

```
INSERT INTO appointment (appno, adate, atime, petno)
VALUES (appnoseq.NEXTVAL, SYSDATE, '12.00','010090');
```

رکورد جدیدی را داخل جدول appointment با مقداری برای ستون appNo (شماره قرار ملاقات) بدنبال مقدار بعدی دنباله درج می کند.

ایجاد جدول خالی در اوراکل ۸ با استفاده از SQL*Plus

برای نشان دادن روند ایجاد جدول خالی در اوراکل، ابتدا از SQL*Plus استفاده می کنیم، که واسط SQL محاوره ای، مبتنی بر خط فرمان پایگاه داده اوراکل است. شکل ۲-۱۹ ایجاد جدول Pen با استفاده از دستور SQL CREATE TABLE اوراکل را نشان می دهد.

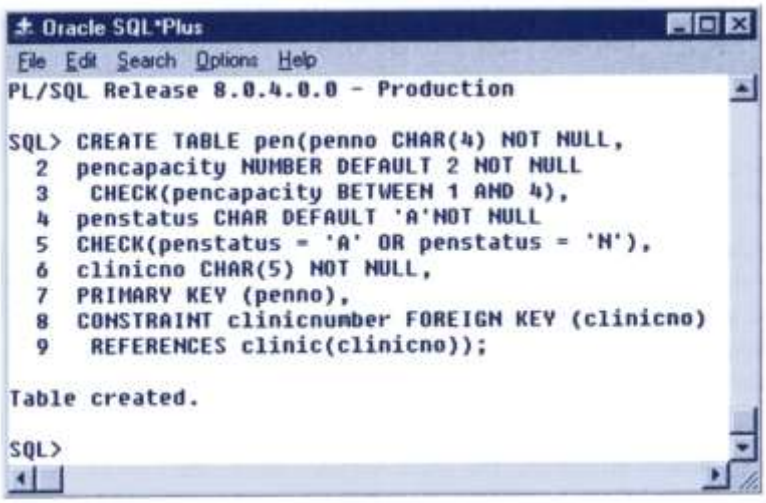
اوراکل اجازه می دهد که محدودیتهای نام، فعال (تنظیم پیش فرض) یا غیرفعال شوند. در موقعیتهای مشخص، شاید مطلوب باشد که محدودیتها را بخاطر دلایل کارایی موقتا غیرفعال کنیم، بطور مثال:

- زمانیکه مقادیر بزرگی از داده ها را داخل جدول با استفاده از SQL*Loader لود می کنیم؛
- زمانیکه عملیات دسته ای را که تعداد عمده ای تغییر به جدول اعمال می کنند را انجام می دهیم؛
- زمانیکه در حال Import یا Export کردن جدول هستیم.

به طور پیش فرض، اورکل جامعیت ارجاع را بر کلیدهای خارجی دارای نام اعمال می کند. بنابراین، اعمال جامعیتی ON DELETE NO ACTION و ON UPDATE NO ACTION را اجرا می کند. همچنین اجازه می دهد که شرط اضافی ON DELETE CASCADE اجازه حذف از جدول والد به طور آبشاری به جدول فرزند را بدهد. با این وجود، اوراکل نه عمل ON UPDATE CASCADE را پشتیبانی می کند، و نه عملهای SET DEFAULT یا SET NULL را. اگر به هر یک از این عملها نیازی باشد، مجبورید که آن ها را یا بعنوان تریگرها و یا داخل کد برنامه کاربردی پیاده سازی کنید. ما این عمل را به طور مختصر در مرحله ۲-۴ در نظر خواهیم گرفت.

شکل ۲-۱۹

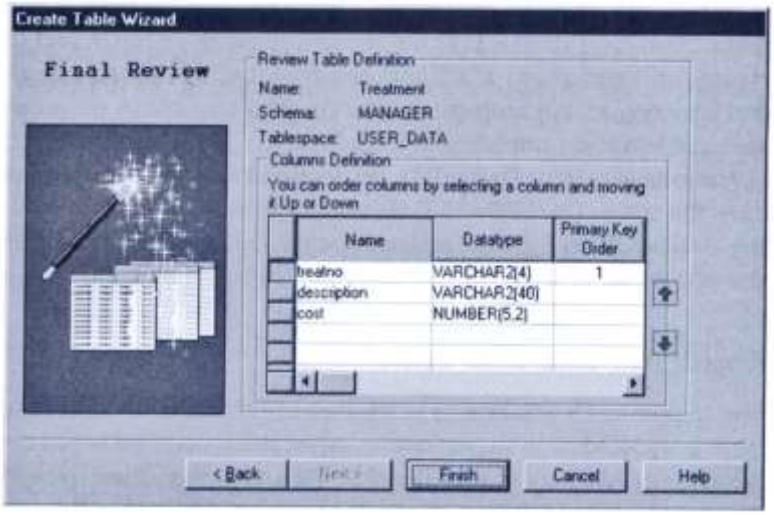
ایجاد جدول Pen با استفاده از دستور SQL اوراکل CREATE TABLE.



ایجاد جدول با استفاده از ویزارد جدول

روش دیگر در اوراکل ۸ استفاده از ویزارد ایجاد جدول است که قسمتی از *مدیریت شِما*[۱] می باشد. با استفاده از یک سری شکلهای محاوره ای، ویزارد ایجاد جدول شما را از طریق پروسه تعریف هر یک از ستونها با نوع داده مربوطه پیش می برد، و همچنین با تعریف هرگونه محدودیت بر ستونها و/ یا محدودیت بر جدول که ممکن است نیاز داشته باشید، و تعریف فیلدهای کلید را میسر می سازد. شکل ۳-۱۹ شکل نهایی ویزارد ایجاد جدول استفاده شده جهت ایجاد جدول Treatment را نشان می دهد.

[۱] Schema Manager



شکل ۳-۱۹

جدول Treatment ایجاد شده با استفاده از ویزارد ایجاد جدول اوراکل.

طراحی جداول پایه را مستند کنید

طراحی جداول پایه بایستی کاملا بهمراه دلایل انتخاب طراحی پیشنهادی مستند شوند. مشخصا، دلایل انتخاب یک روش را زمانیکه چندین روش وجود دارد مستند کنید.

مرحله ۲-٤ طراحی قواعد تجاری برای DBMS هدف

بروز سازی جداول احتمالا توسط قواعد تجاری[۱] محدود شود. بار دیگر، طراحی چنین قواعدی وابسته بر DBMS هدف است؛ بعضی سیستمها امکانات بیشتری را نسبت به دیگری جهت تعریف قواعد تجاری فراهم می کنند. در فصل۱۲، دیدیم که اگر سیستم SQL 1992 استاندارد را قبول کند پیاده سازی بعضی قواعد ساده می شود. همانطور که در بالا دیدیم اوراکل ۸ اجازه می دهد که محدودیتها بعنوان بخشی از دستور SQL CREATE TABLE با استفاده از شرطهای CHECK و CONSTRAINT تعریف شوند، و همچنین اجازه می دهد محدودیتهای اضافی با استفاده از before triggers و after triggers تعریف شوند. برای قابلیت انعطاف بیشتر، اوراکل ۸ همچنین اجازه می دهد که رویه ها ایجاد شده و از SQL فراخوانی شوند.

برای مثال، از شکل۱-۱۹ کلید خارجی ClinicNo در جدول Pen باید عمل on update cascade را داشته باشد. متاسفانه، همانطور که دیدید، دستور CREATE TABLE اوراکل این عمل را پشتیبانی نمی کند. بنابراین، این عمل می تواند با استفاده از تریگرها که در شکل ۴-۱۹ نشان داده شده است پیاده سازی شود.

تریگر۱ (Pen_Clinic_Check_Before)

تریگر موجود در شکل ۴-۱۹ (الف) اجرا می شود[۲] زمانیکه ستون ClinicNo جدول Pen بهنگام شود. تریگر بررسی می کند که بعد از اینکه بروز سازی اتفاق بیافتد مقدار جدید معینی در جدول Clinic وجود دارد. اگر استثنا Invalid_Clinic روی دهد، تریگر پیام خطایی را نشان می دهد و از رخداد هر گونه تغییر جلوگیری می کند. نقاط زیرین بایستی مورد توجه قرار گیرند:

- کلمه کلیدی *BEFORE* نشان می دهد که تریگر بایستی قبل از اینکه بهنگام ساز ی ستون ClinicNo به جدول Pen اعمال گردد، اجرا شود.

---

[۱] Business rule
[۲] fired

- کلمه کلیدی *FOR EACH ROW* نشان می دهد که این یک تریگر سطح ردیف[1] است، که برای هر سطر جدول Pen هر موقع که تراکنش بروز شود اجرا می شود. نوع دیگری از تریگر، تریگر سطح دستور[2] است که یکبار برای هر تراکنش اجرا می شود. مثالهایی از تریگرهای سطح دستور را مختصرا خواهیم دید.
- شرط WHEN شرطی را مشخص می کند که بایستی برای اجرا شدن تریگر مهیا شوند.
- کلمه کلیدی new جهت مراجعه به مقدار جدیدی از ستون و کلمه کلیدی old جهت رجوع به مقدار قدیمی ستون استفاده می شود.

تغییرات بر پشتیبانی تریگرها در جدول Clinic

سه تریگری که در شکل ۴-۱۹(ب) نشان داده شده اند هر زمان که ستون ClinicNo جدول Clinic بروز شود اجرا می شوند. قبل از تعریف تریگرها، شماره توالی updateSequence ایجاد شده است بهمراه متغیر عمومی updateSeq (که به سه تریگر از طریق بسته seqPackage قابل دستیابی است). بعلاوه، جدول Pen جهت افزودن ستونی به نام updateId اصلاح می شود، که جهت نشانه گذاری اینکه آیا رکورد بروز شده است، برای جلوگیری از بیش از یکبار بروز شدن طی عمل آبشاری استفاده می شود.

تریگر۲ (Cascade_ClinicNo_Update1)

تریگر (سطح دستور)، Cascade_ClinicNo_Update1 ، قبل از اینکه بروز سازی ستون ClinicNo در جدول Clinic به شماره توالی جدیدی ست شود اتفاق می افتد.

تریگر ۳(Cascade_ClinicNo_Update2)

تریگر (سطح ردیف)، Cascade_ClinicNo_Update2، جهت بهنگام سازی همه رکوردهای جدول Pen که مقدار قدیمی ClinicNo را دارند (:old.clinicno) به مقدار جدید (:new.clinicno)، و رکورد را بعنوان بروز شده علامت گذاری می کند.

---

[1] Row-level
[2] Statement-level

شکل ۴-۱۹

تریگرهای اوراکل جهت اجرای ON UPDATE CASCADE بر روی کلید خارجی clinicNo در جدول Pen وقتیکه کلید اصلی clinicNo در جدول Clinic بهنگام شود: (الف) تریگری برای جدول Pen ;(ب) تریگرهایی برای جدول Clinic.

```
--Before the clinicNo column is updated in the Pen table, fire this trigger
--to verify that the new foreign key value is present in the Clinic table.
CREATE TRIGGER Pen_Clinic_Check_Before
   BEFORE UPDATE OF clinicno ON pen
   FOR EACH ROW WHEN (new.clinicno IS NOT NULL)
DECLARE
   dummy CHAR(5);
   invalid_clinic EXCEPTION;
   valid_clinic EXCEPTION;
   mutating_table EXCEPTION;
   PRAGMA EXCEPTION_INIT (mutating_table,-4091);
--Use cursor to verify parent key value exists.
--Use FOR UPDATE OF to lock parent key's record so that it cannot be deleted
--by another transaction before this transaction completes.
CURSOR update_cursor (sn CHAR(5)) IS
   SELECT clinicno FROM clinic
   WHERE clinicno = sn
   FOR UPDATE OF clinicno;
   BEGIN
         OPEN update_cursor (:new.clinicno);
         FETCH update_cursor INTO dummy;
--Verify parent key. Raise exceptions as appropriate.
         IF update_cursor%NOTFOUND THEN
               RAISE invalid_clinic;
         ELSE
               RAISE valid_clinic;
         END IF;
         CLOSE update_cursor;
   EXCEPTION
   WHEN invalid_clinic THEN
         CLOSE update_cursor;
         raise_application_error(-20000, 'Invalid Clinic Number'||:new.clinicno);
   WHEN valid_clinic THEN
         CLOSE update_cursor;
--A mutating table is a table that is currently being modified by an INSERT, UPDATE,
--or DELETE statement, or one that might need to be updated by the effects of a declarative
--DELETE CASCADE referential integrity constraint.
--This error would raise an exception, but in this case the exception is OK, so trap it,
--but don't do anything.
   WHEN mutating_table THEN
         NULL;
   END;
```

تریگر BEFORE

شرطی که در آن تریگراجرا میشود

تریگر سطح-ردیف

clinicNo نامعتبر مشخص شده است

(الف)

شکل ۴-۱۹

ادامه . . .

```
--Create a sequence number and a public variable UPDATESEQ.
CREATE SEQUENCE updatesequence INCREMENT BY 1 MAXVALUE 500 CYCLE;
CREATE PACKAGE seqpackage AS
   updateseq NUMBER;
END seqpackage;
CREATE or REPLACE PACKAGE BODY seqpackage AS END seqpackage;

--Add a new column to the Pen table to flag changed records.
ALTER TABLE pen ADD updateid NUMBER;

--Before updating the Clinic table using this statement trigger, generate a new
--sequence number and assign it to the public variable UPDATESEQ.
CREATE TRIGGER Cascade_ClinicNo_Update1
   BEFORE UPDATE OF clinicno ON clinic
   DECLARE
         dummy NUMBER;
         BEGIN
               SELECT updatesequence.NEXTVAL
               INTO dummy FROM dual;
               seqpackage.updateseq :=dummy;
         END;

--Create a row after-trigger that cascades the update to the Pen table.
--Only cascade the update if the child row has not already been updated by the trigger.
CREATE TRIGGER Cascade_ClinicNo_Update2
   AFTER UPDATE OF clinicno ON clinic
   FOR EACH ROW
   BEGIN
         UPDATE pen SET clinicno = :new.clinicno, updateid = seqpackage.updateseq
         WHERE pen.clinicno = :old.clinicno AND updateid IS NULL;
   END;

--Create a final statement after-trigger to reset the updateid flags
CREATE TRIGGER Cascade_ClinicNo_Update3
   AFTER UPDATE OF clinicno ON clinic
   BEGIN
         UPDATE pen SET updateid = NULL
         WHERE updateid = seqpackage.updateseq;
   END;
```

جهت Package نگه داشتن عدد ترتیبی

افزودن ستون اضافی به جدول Pen

تریگر سطح-دستور before

قرار دادن عدد ترتیبی جدید برای بهنگام سازی

تریگر سطح-ردیف after

بهنگام سازی جدول Pen و قرار دادن علامت بهنگام شده برای این رکوردها

تریگر سطح-دستور after ; بازنشانی علامت ها برای رکوردهای بهنگام شده

(ب)

تریگر ۴ (Cascade_ClinicNo_Update3)

تریگر (سطح دستور) نهایی، Cascade_ClinicNo_Update3، بعد از اینکه بروز سازی رکوردهای علامت گذاری شده به بدون علامت تغییر یابند اجرا می شود.

درباره جزئیات اینکه چگونه این تریگرها کار می کنند زیاد نگران نباشید. آنچه که اهمیت دارد این است که به این نکته توجه کنید که برای پیاده سازی این عملها مقدار زیادی کدنویسی لازم است. به عبارت دیگر، فکر کنید که اگر DBMS این کارکردها را خودش فراهم میکرد چقدر در زمان و هزینه صرفه جویی می شد.

مرحله ۱-۵ تراکنشها را تحلیل کنید

بعد از ایجاد جداول پایه، محدودیتهای جامعیت، و قواعد تجاری، مرحله بعد تحلیل تراکنشهایی است که برای کمک به مشخص کردن سازمان فایل مناسب و شاخصهای هر جدول پایه بکار می رود. اجازه دهید فرض کنیم که تراکنشهای مشخص شده در بخش ۲-۱-۱۸ مهمترین تراکنشهای *Perfect Pets* باشند. برای اینکه بر نواحی که ممکن است مساله ساز باشد تمرکز کنیم، در فصل ۱۳ پیشنهاد کردیم که یک راه ممکن است انجام یکی از کارهای زیر باشد:

(۱) تمام مسیرهای تراکنش را به جدولها نگاشت کنید.
(۲) مشخص کنید که کدام جدول مکررا توسط تراکنشها مورد دستیابی قرار می گیرند.
(۳) تراکنشهای انتخابی را که شامل این جدولها هستند تحلیل کنید.

مرحله اول در مراحل ۸-۱ و ۳-۲ به آن پرداخته شد(شکل ۹-۱۸ را ببینید). برای انجام مرحله دوم، باید تخمینی از تکراری که کدام جدولها مورد دسترسی قرار خواهند گرفت داشته باشید. در صورت امکان، بایستی اطلاعات تکرار را به نمودار مسیر تراکنش بیافزایید. این عمل بعضی اوقات نمودار را بسیار درهم و برهم کرده و تفسیر آن را مشکل می کند، و ممکن است ترجیح دهید که اطلاعات را به صورت مجزا نگه دارید. بهنگام مذاکره با پرسنل *Perfect Pets*، اطلاعات تکراری که در شکل ۵-۱۹ نشان داده شده بدست آمده است.

بعلاوه، شما مجبورید که تکرار احتمالی هر تراکنش را تخمین بزنید. همه این اطلاعات باید تحلیل شوند تا قسمتهایی که نیاز به ملاحظات بخصوصی دارند مشخص شوند.

مرحله ۲-۵ سازمانهای فایل را انتخاب کنید

هدف این مرحله این است که در صورتیکه DBMS هدف اجازه بدهد سازمان فایل بهینه را برای هر جدول انتخاب کنیم. برای اینکه این مرحله را متقبل شوید، باید از چگونگی عملکرد DBMS هدف در سطوح منطقی و فیزیکی اطلاع داشته باشید. در این مرحله، بررسی می کنیم که چگونه اوراکل داده را ذخیره می کند و همچنین درباره موارد زیر نیز بحث می کنیم:

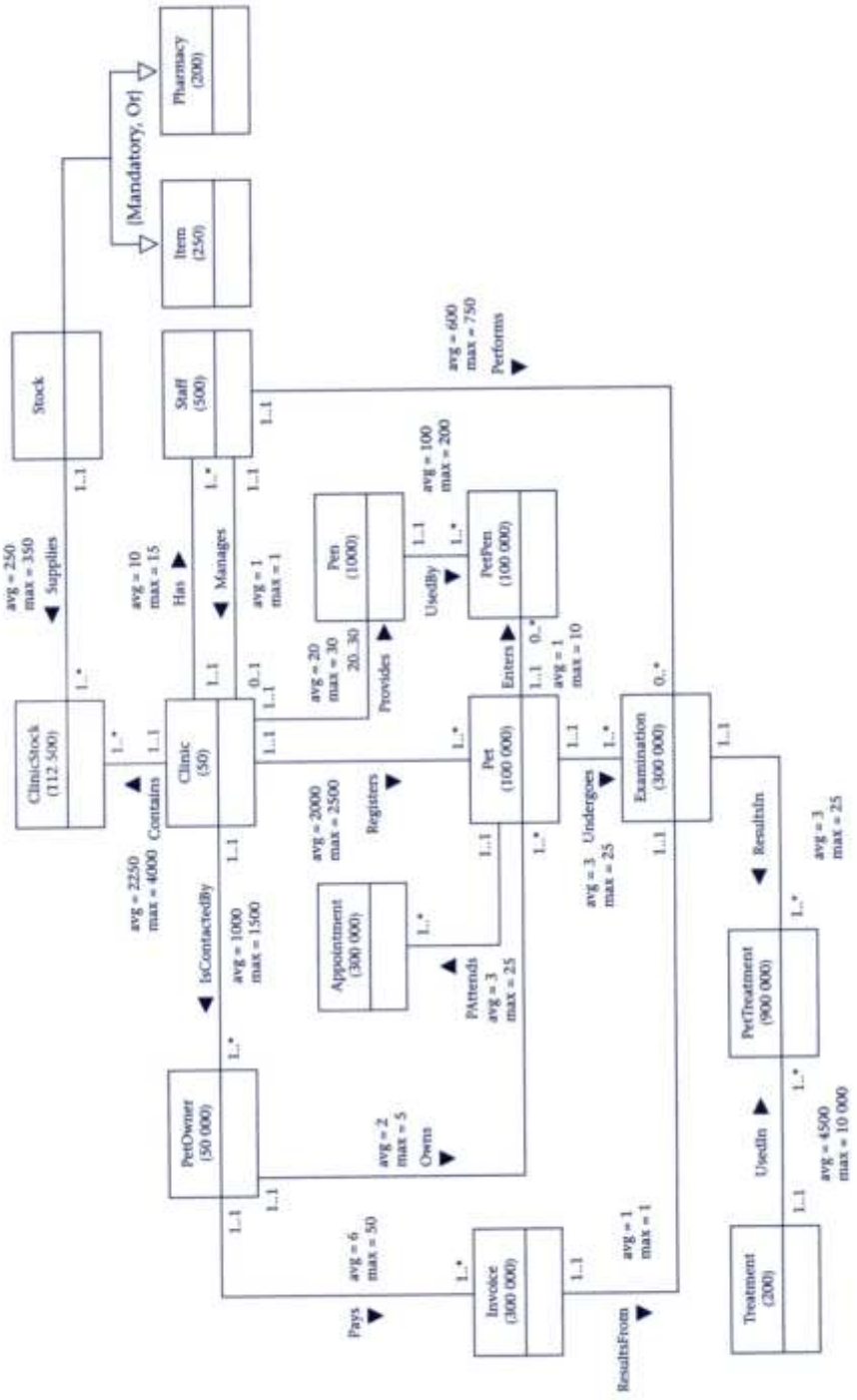
- جداول خوشه ای[1]
- پارامترهای مدیریت فضا.

قطعا بحث کمی فنی بنظر می رسد اما این بحث در شما احساسی را بوجود می آورد که می توانید در عمل این مرحله را قبول کنید.

[1] Clustered tables

شکل ۵-۱۹

مدل داده منطقی *Perfect Pets* که رخدادهای مورد انتظار را نشان می دهد.



ساختار منطقی پایگاه داده اوراکل

در سطح منطقی، همچنانکه در زیر توضیح می دهیم اوراکل فضاهای جدول[۱] ، شِماها [۲]، و بلوکهای داده و توسعه ها[۳]/ سگمنتها، را نگهداری می کند.

فضاهای جدول

پایگاه داده اوراکل به واحدهای ذخیره سازی منطقی با نام فضاهای جدول تقسیم بندی می شود. فضای جدول برای گروه بندی ساختارهای منطقی مرتبط به یکدیگر استفاده می شود. برای مثال، فضاهای جدول عموما جهت ساده سازی بعضی عملیات مدیریتی همه اشیا برنامه های کاربردی را گروه بندی می کنند.

هر پایگاه داده اوراکل شامل فضای جدولی با نام SYSTEM است که به طور خودکار هنگام ایجاد پایگاه داده ایجاد خواهد شد. فضای جدول SYSTEM همیشه شامل جداول کاتالوگ سیستم برای داخل پایگاه داده است. یک پایگاه داده کوچک ممکن است تنها تیاز به فضای جدول سیستم داشته باشد؛ با این وجود، توصیه می شود که حداقل یک فضای جدول اضافی جهت ذخیره داده کاربر مجزا از کاتالوگ سیستم، که برای کاستن تماس بین اشیا دیکشنری و اشیا شِماهای فایلهای داده مشابه بکار می رود، ایجاد کنید(جدول ۱-۱۳ را ببینید). شکل ۶-۱۹ پایگاه داده اوراکلی را نشان می دهد که شامل فضای جدول SYSTEM و فضای جدول USER_DATA است.

فضای جدول جدید می تواند با استفاده از فرمان CREATE TABLESPACE ایجاد شود؛ برای نمونه:

```
CREATE   TABLESPACE  user_data
DATAFILE   'DATA3.ORA'  size  100k;
```

جدول سپس می تواند با فضای جدول بخصوصی با استفاده از دستور CREATE TABLE یا ALTER TABLE مرتبط شود؛ برای مثال:

```
CREATE  TABLE pen (penno char (4) NOT NULL…)
                TABLESPACE   user_data;
```

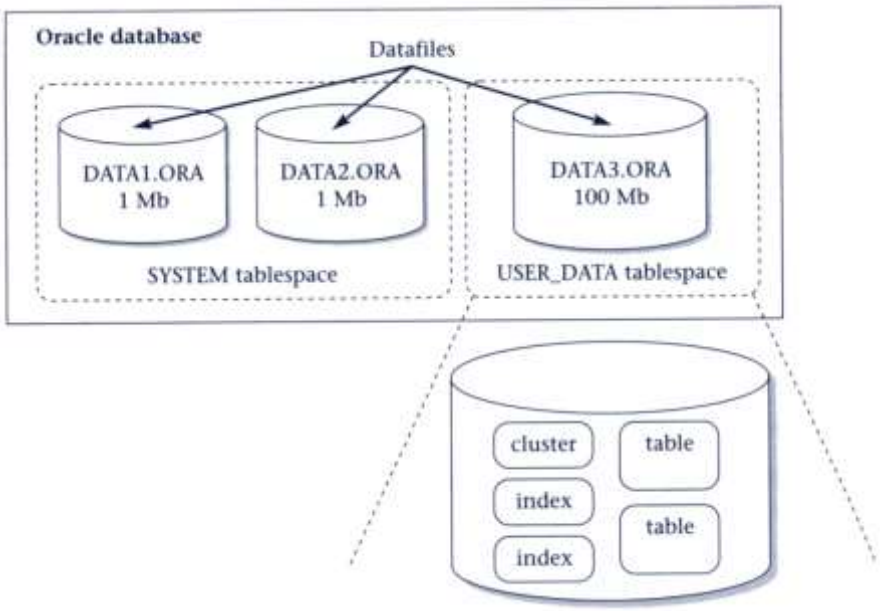
اگر هیچ فضای جدولی مشخص نشود در این صورت هنگام ایجاد جدول جدید، زمانیکه حساب کاربر ایجاد می شود فضای جدول پیش فرض مرتبط با کاربر استفاده می شود. ما در مرحله ۲-۷ چگونگی مشخص شدن این را خواهیم دید.

کاربران، شِماها، و اشیا شِما

کاربر (بعضی از اوقات *نام کاربر*) نامی است که در پایگاه داده تعریف می شود که از طریق آن میتوان به اشیا مرتبط شده و به آنها دسترسی داشته باشید. *شِما* نامی برای مجموعه اشیایی، همچون جداول، دیدها، و خوشه ها، و رویه ها است که مربوط به کاربر مشخصی می باشد. شِماها و کاربران به DBA در مدیریت امنیت پایگاه داده کمک می کنند. برای دستیابی به پایگاه داده، کاربر بایستی برنامه پایگاه داده را اجرا کند (همچون فرم اوراکل یا SQL*Plus) و با استفاده از نام کاربری که در پایگاه داده تعریف شده به آن متصل شود. زمانیکه کاربر پایگاه داده ایجاد شد شِمای متناظر با نام مشابه با نام کاربر ایجاد می شود. بصورت پیش فرض، هنگام اتصال کاربر به پایگاه داده، کاربر به همه اشیا در برگرفته شده در شِمای متناظر دسترسی دارد. از آنجاییکه کاربر تنها با شِمای مشابه نام خودش ارتباط دارد، عبارات کاربر و شِما به جای هم استفاده می شوند.

---

[۱] Tablespace
[۲] Schema
[۳] Extent



شکل ۶-۱۹

رابطه بین پایگاه داده اوراکل، فضای جدول، و فایلهای داده.

توجه کنید که هیچ ارتباطی بین فضای جدول و شِما وجود ندارد؛ اشیاء در شِماهای مشابه می تواند در فضای جدول متفاوتی باشند؛ و فضای جدول می تواند اشیایی از شِماهای متفاوت را نگه دارد.

بلوکهای داده، توسعه ها، و سگمنتها

بلوک داده[۱] واحد کوچکی از فضای ذخیره سازی است که اوراکل می تواند استفاده کند یا تخصیص دهد. یک بلوک داده به تعداد مشخصی از فضای بایتهای فیزیکی دیسک اشاره می کند. شما اندازه بلوک داده هر پایگاه داده اوراکل را زمان ایجاد پایگاه داده تنظیم می کنید. سایز بلوک داده بایستی مضربی از سایز بلوک سیستم عامل باشد تا از I/O غیرلازم جلوگیری بعمل آورد.

سطح بعدی فضای منطقی پایگاه داده توسعه[۲] نامیده می شود. توسعه تعداد بخصوصی از بلوکهای داده متوالی اختصاص داده شده است که برای ذخیره کردن نوع مشخصی از اطلاعات بکار می رود. سطح بالای هر توسعه سگمنت نامیده می شود. سگمنت مجموعی از توسعه های تخصیص داده شده به ساختار منطقی مشخصی است. برای مثال، هر داده جدول در سگمنت داده خودش ذخیره می شود، در حالیکه هر داده شاخص در سگمنت شاخص مربوط به خود ذخیره می شود. شکل ۷-۱۹ رابطه بین بلوکهای داده، توسعه ها، و سگمنت ها را نشان می دهد.

زمانیکه توسعه های موجود سگمنت پر می شوند اوراکل فضا را به طور پویا تخصیص می دهد. به این علت که توسعه ها به هنگام نیاز تخصیص داده می شوند، بنابراین می توانند بطور مجاور بر روی دیسک قرار گیرند.

ساختار فیزیکی پایگاه داده اوراکل

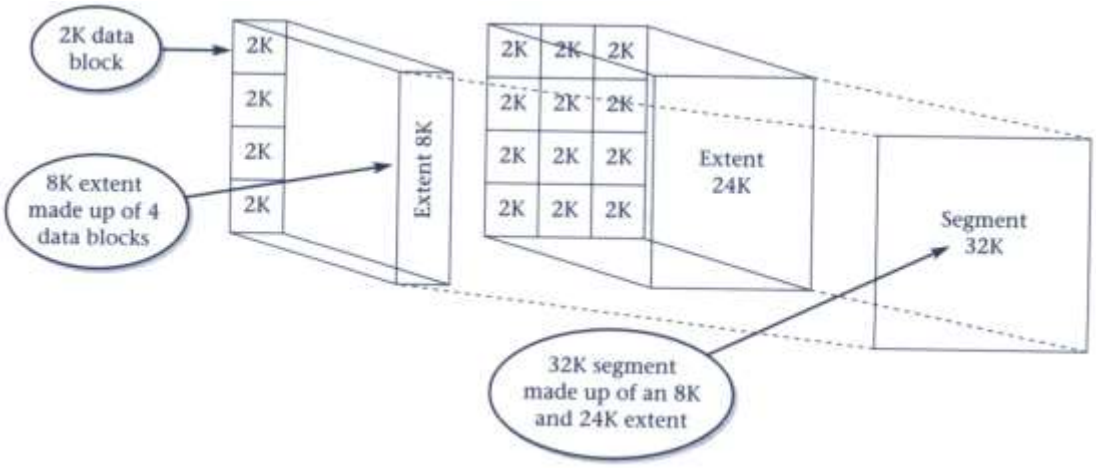
ساختارهای اصلی فیزیکی پایگاه داده اوراکل، فایلهای داده[۳]، فایلهای گزارش انجام دوباره[۴]، و فایلهای کنترل می باشند.

---

۱ Data Block
۲ Extent
۳ Datafile
۴ Redo log file

شکل ۷-۱۹

رابطه بین بلوکهای داده اوراکل، توسعه ها، و سگمنتها.



فایلهای داده

هر پایگاه داده اوراکل یک یا چند فایل داده فیزیکی دارد. داده ساختارهای منطقی پایگاه داده (همچون جداول و شاخصها) به طور فیزیکی در این فایلهای داده ذخیره شده اند. یک یا چند فایل داده فضای جدول را تشکیل می دهد. ساده ترین پایگاه داده اوراکل یک فضای جدول و یک فایل داده خواهد داشت. پایگاه داده های پیچیده تر ممکن است چهار فضای جدول داشته باشند، که هریک شامل دو فایل داده هستند، که مجموعا ۸ فایل داده خواهند شد. معماری فایلهای داده ای و فضای جدول در شکل ۶-۱۹ نشان داده شده است.

فایلهای گزارش انجام دوباره

هر پایگاه داده اوراکل مجموعه ای از دو یا چند فایل گزارش انجام دوباره دارد، که همه تغییرات اعمال شده بر روی داده را برای اهداف بازیابی ثبت می کند. اگر خطایی مانع از نوشته شدن داده تغییر یافته در فایلهای داده شود تغییرات می تواند از فایل گزارش دوباره بازیابی شود، بنابراین از گم شدن کارمان جلوگیری می کند.

فایلهای کنترل

هر پایگاه داده اوراکل فایل کنترلی دارد که شامل ورودیهایی است که ساختار فیزیکی پایگاه داده را مشخص می کند. همچون:

- نام پایگاه داده
- نامها و محلهای فایلهای داده پایگاه داده و فایلهای گزارش انجام دوباره
- نشان زمان ایجاد پایگاه داده.

جدولهای خوشه بندی شده[۱] و غیر خوشه بندی شده[۲]

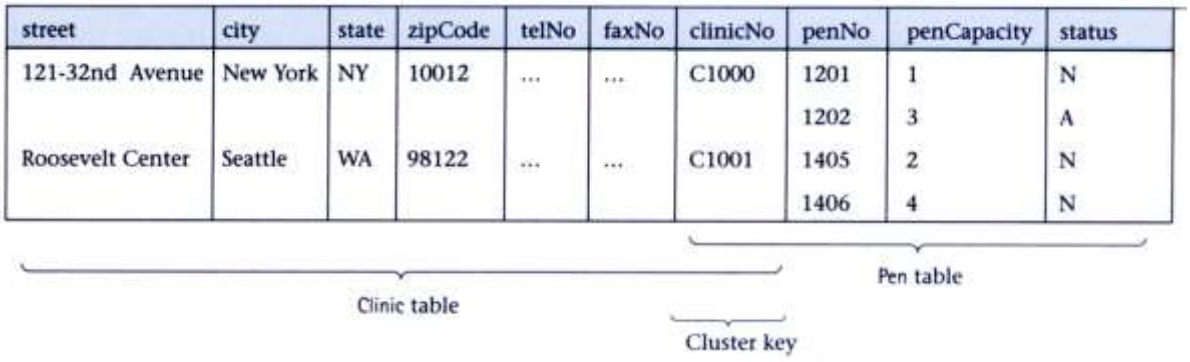
اوراکل ۸ جداول خوشه بندی شده و غیر خوشه بندی شده را پشتیبانی می کند. انتخاب اینکه آیا از این جداول استفاده کنیم بستگی به تحلیل تراکنشهایی دارد که انجام گرفته است، و این انتخاب می تواند روی کارایی تاثیر داشته باشد. در این مرحله، هر دو نوع این ساختارها را بررسی می کنیم و راهنمایی را برای استفاده از آنها فراهم می کنیم.

---

[۱] Clustered
[۲] Non-Clustered

شکل ۸-۱۹

چگونه جدولهای Clinic و Pen بایستی بصورت شاخص دار روی clinicNo ذخیره شوند.



street	city	state	zipCode	telNo	faxNo	clinicNo	penNo	penCapacity	status
121-32nd Avenue	New York	NY	10012	...	...	C1000	1201	1	N
							1202	3	A
Roosevelt Center	Seattle	WA	98122	...	...	C1001	1405	2	N
							1406	4	N

خوشه گروههایی از یک یا چند جدول هستند که بطور فیزیکی با همدیگر ذخیره شده اند، زیرا آنها ستونهای مشترک را به اشتراک می گذارند و اغلب با همدیگر استفاده می شوند. با وجود رکوردهای مرتبطی که بصورت فیزیکی با همدیگر مرتبط شده اند، زمان دستیابی دیسک بهبود می یابد. ستونهای مرتبط جدولها در خوشه **کلید خوشه** نامیده می شود. کلید خوشه تنها یکبار ذخیره می شود، بنابراین خوشه ها مجموعه ای از جدولها را موثرتر از زمانی که بصورت منفرد باشند ذخیره می کند. شکل ۸-۱۹ نشان می دهد که اگر جدولها را بر اساس ستون ClinicNo خوشه بندی کنیم چگونه جداول Clinic و Pen بایستی ذخیره شوند. زمانیکه این دو جدول خوشه بندی شدند، هر مقدار منحصربفرد ClinicNo تنها یکبار در کلید خوشه ذخیره می شود. برای هر مقدار ClinicNo ستونها را از این دو جدول به هم وصل می کند.

اوراکل دو نوع خوشه بندی را پشتیبانی می کند:

(a) خوشه های شاخص دار[۱],

(b) خوشه های درهم[۲].

خوشه های شاخص دار

در *خوشه های شاخص دار*، رکوردهای با کلید خوشه مشابه با هم ذخیره می شوند. اوراکل استفاده از خوشه های شاخص دار را تنها در موارد زیر پیشنهاد می کند:

- پرس وجوها رکوردها را از بین دامنه مقادیر کلید خوشه بازیابی کنند؛
- جداول خوشه بندی شده بصورت غیرقابل پیشگویی رشد کنند.

خوشه ها می تواند کارایی بازیابی داده را، براساس توزیع داده و اینکه چه عملیات SQL ای غالبا بر روی داده انجام می گیرند بهبود بخشد. مشخصا، جدولهایی که در پرس وجو الحاق شده اند بعلت استفاده از خوشه ها سود می برند زیرا رکوردهایی که در الحاق جدولها مشترک هستند با عمل مشابه I/O بازیابی می شوند.

برای ایجاد خوشه شاخص دار در اوراکل با نام Staff_Cluster با ستون کلید خوشه staffNo، شما می توانید از دستور SQL زیرین استفاده کنید:

```
CREATE CLUSTER  staff_cluster
(staffno  CHAR (5))
SIZE   512
STORAGE (INITIAL 100K NEXT 50K PCTINCREASE 10);
```

---

[۱] Indexed cluster

[۲] Hash Cluster

پارامتر SIZE میزان فضایی را که برای ذخیره همه رکوردهای با مقدار کلید خوشه مشابه لازم است را مشخص می کند. این اندازه اختیاری بوده و اگر حذف شود اوراکل یک بلوک داده برای هر مقدار کلید خوشه رزرو می کند. پارامتر INITIAL اندازه اولین ناحیه خوشه را مشخص کرده، و پارامتر NEXT اندازه حافظه بعدی تخصیص داده شده را مشخص می سازد. پارامتر PCTINCREASE درصدی را مشخص می کند که توسعه های سوم و بعدی به چه اندازه نسبت به قبلی باید رشد داشته باشند (پیش فرض ۵۰). در مثال ما، مشخص کردیم که هر توسعه بعدی بایستی ۱۰ درصد بزرگتر از حوزه قبلیش باشد.

راهنمایی هایی جهت استفاده از خوشه های شاخص دار

زمانیکه می خواهید جدولها را خوشه بندی کنید راهنمایی های زیر می تواند مفید باشد:

- جدولهایی را برای خوشه بندی در نظر بگیرید که اغلب در دستورات الحاق مورد دستیابی قرار می گیرند.
- جدولهایی را که اغلب الحاق می شوند یا اگر مقادیر ستون مشترکشان بطور مکرر تغییر می یابند را خوشه بندی نکنید. (تغییر مقدار کلید خوشه رکوردها مدت زمان بیشتری را نسبت به تغییر مقدار جداول بدون خوشه بندی می گیرد، چون اوراکل ممکن است برای نگهداری خوشه مجبور به بردن رکوردهای تغییر یافته به بلوک دیگری باشد).
- جدولهایی را که در یکی از آنها نیاز به جستجوی کامل دارید خوشه بندی نکنید(جستجوی کامل جدول خوشه بندی شده می تواند مدت زمان بیشتری نسبت به جدول خوشه بندی نشده بگیرد. از آنجائیکه جدولها با هم ذخیره شده اند اوراکل احتمالا بلوکهای بیشتری را می خواند).
- جدولهای خوشه بندی در برگرفته شده در رابطه های یک به چند را در نظر بگیرید اگر اغلب رکوردی را از جدول والد انتخاب می کنید و سپس رکوردهایی را متناوبا از جدول فرزند انتخاب می کنید. (رکوردهای فرزند در بلوکهای داده مشابه رکورد والد ذخیره می شوند، بنابراین وقتی که شما آنها را انتخاب می کنید در حافظه هستند، و باعث می شود که اوراکل I/O کمتری را انجام دهد)
- اگر اغلب تعدادی از چندین رکورد فرزند والد مشابهی را انتخاب می کنید ذخیره سازی جدول فرزند را بتنهایی در خوشه در نظر بگیرید (این مقیاس کارایی پرس وجوهایی را که رکوردهای فرزند والد مشابهی را انتخاب می کند بهبود می بخشد اما از کارایی جستجوی کامل جدول والد نمی کاهد.)
- جدولهایی را خوشه بندی نکنید که در آنها داده از همه جدولهای با مقدار کلید خوشه مشابه بیش از یک یا دو بلوک اوراکل تخطی کند (جهت دستیابی به رکورد در جدول خوشه بندی شده، اوراکل همه بلوکهای در برگرفته شده رکوردهایی با آن مقدار را می خواند. اگر این رکوردها چندین بلوک را اشغال کنند، دستیابی به رکورد واحد می تواند به خواندنهای بیشتری نسبت به دستیابی به رکورد مشابه در جدول خوشه بندی نشده نیاز داشته باشد.)

خوشه های درهم

خوشه های درهم و نیز جدول داده خوشه در عمل مشابه خوشه های شاخص می باشند. با این وجود، رکوردی که در خوشه درهم ذخیره شده است براساس نتایج اعمال تابع درهم به مقدار کلید خوشه رکورد می باشد. همه رکوردهای با مقدار کلید درهم مشابه با همدیگر بر روی دیسک ذخیره شده اند. اوراکل استفاده از خوشه های رکورد را در موارد زیر پیشنهاد می کند:

- پرس وجوهایی که رکوردها را براساس همسانی شرایط دربرگیرنده همه ستونهای کلید خوشه بازیابی می کنند. (برای مثال، همه رکوردها را برای کلینیک S1000 برگردانید)
- جداول خوشه بندی شده ای که ساکن هستند یا شما می توانید تعداد حداکثر رکوردها و همچنین مقدار حداکثر فضای مورد نیاز توسط خوشه را هنگام ایجاد آن مشخص سازید.

برای ایجاد خوشه درهم در اوراکل با نام Exams_Cluster که توسط ستون examNo خوشه بندی شده، شما می توانید از دستور SQL زیرین استفاده نمایید:

```
CREATE CLUSTER exams_cluster
    (examno   NUMBER (6, 0))
HASH IS examno HASHKEYS 300000;
```

زمانیکه خوشه درهم را ایجاد کردید می توانید جدولی را ایجاد کنید که قسمتی از ساختار باشد، برای مثال:

```
CREATE TABLE examination
(examno NUMBER (6, 0) PRIMARY KEY,
                                …)
CLUSTER exams_cluster (examno);
```

راهنما برای استفاده از خوشه های درهم

هنگام استفاده از خوشه های درهم راهنمایی های زیرین می تواند مفید باشد:

- خوشه های درهم را برای ذخیره جدولهایی که مکررا با استفاده از شرط جستجوی شامل شرایط برابری با ستونهای مشابه مورد دستیابی قرار می گیرند استفاده کنید.
  این ستونها را بعنوان کلید خوشه معین کنید.

- اگر می خواهید مشخص کنید که برای نگه داشتن همه رکوردها با مقدار کلید خوشه معین هم در حال و هم آینده چه مقدار فضا نیاز می باشد، جدول را در خوشه درهم ذخیره کنید.
- اگر فضا اندک باشد و نتوانید فضای اضافی را برای رکوردهایی که در آینده درج خواهند شد تخصیص دهید، از خوشه های درهم استفاده نکنید.
- از خوشه درهم برای ذخیره کردن جدول هایی که دائما رشد می کنند استفاده نکنید بطوریکه اگر فرایند ایجاد گاه به گاه خوشه درهم جدید بزرگتر برای نگه داشتن جدول غیر عملی باشد.
- اگر اغلب جستجوی داخلی جدول موردنیاز باشد و به این علت که بایستی مقدار قابل توجهی از فضا را به خوشه درهم در پیش بینی رشد جدول در نظر گرفته باشید، بنابراین جدول را در خوشه درهم ذخیره نکنید.(چنین جستجوهای کامل بایستی همه بلوکهای اختصاص داده شده به جدول را بخواند حتی ممکن است بعضی از بلوکها شامل چندین رکورد باشد. ذخیره سازی جدول به طور مجزا می تواند سبب افزایش تعداد بلوکهای خوانده شده توسط جستجوی کامل جدول باشد.)
- اگر برنامه کاربردی شما مکررا مقادیر کلید خوشه را تغییر می دهند جدول را در خوشه درهم ذخیره نکنید.
- ذخیره سازی جدول واحد در خوشه درهم می تواند مفید باشد، صرفنظر از اینکه آیا جدول اغلب با دیگر جداول الحاق می شود، و کمک می کند که درهم سازی برای جدول بر اساس راهنمایی قبلی مناسب باشد.

PCTFREE* و *PCTUSED

دو پارامتر مدیریت فضای، *PCTFREE* و *PCTUSED*، ممکن است همچنین تاثیر مهمی بر روی کارایی داشته باشند. شما این پارامترها را زمانیکه جدولی را ایجاد می کنید یا اینکه در حال ایجاد خوشه ای هستید (که سگمنت داده مختص خودشان را دارند) مشخص می سازید. شما همچنین می توانید ذخیره سازی *PCTFREE را* زمانیکه شاخصی را ایجاد یا اصلاح می کنید مشخص کنید (که سگمنت شاخص مختص خود را دارد). پارامترها بدین صورت استفاده می شوند:

- *PCTFREE* حداقل درصدی از بلوک داده ای را که بعنوان فضای خالی برای بروز سازی های احتمالی رکوردهای موجود در بلوک رزرو می شود را ست می کند (مقدار پیش فرض۱۰ است).
- *PCTUSED* حداقل درصدی از بلوک که می توانند برای ثبت داده بعلاوه هرگونه هزینه موردنیاز اوراکل قبل از اینکه رکوردهای جدید به بلوک اضافه شوند را تنظیم می کند (مقدار پیش فرض ۴۰ است). پس از اینکه بلوک داده به محدودیت مشخص شده توسط PCTFREE پر شد، اوراکل بلوکهای غیر قابل دسترس جهت درج رکوردها را در نظر می گیرد تا زمانیکه درصدی از آن بلوکها زیر پارامتر PCTFREE بیافتند. زمانیکه آن مقدار مورد دستیابی قرار بگیرد، اوراکل از فضای خالی بلوک داده تنها بعنوان بهنگام سازی رکوردهای کنونی که در بلوک داده در برگرفته شده است استفاده می کند.

مقدار کمتر PCTFREE فضای کمتری را برای بروز سازی رکوردهای موجود رزرو می کند و اجازه می دهد درج ها برای پر کردن بلوک کاملتر صورت گیرد. این عمل ممکن است فضای زیادی را برای شما نگه دارد اما هزینه پردازش را افزایش می دهد زیرا از آنجائیکه ناحیه فضای خالی با رکوردهای جدید/ بروز شده پر می شوند بنابراین بلوکها اغلب نیازمند سازمان یابی دوباره هستند. مقدار کم برای PCTUSED فضای غیرقابل استفاده را در پایگاه داده افزایش می دهد اما هزینه های پردازش را در طی عملیات بروز سازی و درج کاهش می دهد.

بدیهی است که مجموع PCTFREE و PCTUSED بیشتر از ۱۰۰ نخواهد شد. اگر مجموع کمتر از صد باشد، تنظیمات بهینه برای استفاده متعادل از فضا و I/O مجموع دو پارامتری می شود که از صد با درصدی از فضای اشغال شده بوسیله یک رکورد اختلاف دارد. برای مثال، اگر اندازه بلوک 2048 بایت بهمراه یک صد بایت اضافی باشد، و اندازه رکورد 390 بایت باشد، که ۲۰ درصد فضای موجود بلوک می باشد، بنابراین یک ارزش خوب برای مجموعه PCTFREE و PCTUSED ۸۰ درصد خواهد بود تا از فضا به بهترین نحو استفاده شود. بعبارت دیگر، اگر مجموع برابر با صد شود، اوراکل بیش از PCTFREE از فضای آزاد استفاده نخواهد کرد، که منتج به بالاترین هزینه پردازش خواهد شد. استفاده ترکیبی از PCTFREE و PCTUSED در شکل ۹-۱۹ نشان داده شده است.

قبل از اینکه بتوانیم درباره سازمانهای فایل برای پایگاه داده Perfect Pets تصمیم گیری کنیم، ابتدا نیاز داریم که بررسی کنیم چه شاخصهایی می خواهیم جهت بالا بردن کارایی اضافه کنیم.

شکل ۹-۱۹

استفاده ترکیبی PCTFREE و PCTUSED با PCTFREE=20 درصد و PCTUSED=40 درصد.

مرحله ۳-۵ شاخصها را انتخاب کنید

اوراکل به طور خودکار شاخص را به هر کلید اصلی می افزاید. بعلاوه اوراکل پیشنهاد می کند که آشکارا شاخصهای UNIQE را روی جدولها تعریف نکنید درعوض محدودیتهای جامعیت UNIQE را بر روی ستونهایی که می خواهید تعریف کنید. اوراکل محدودیت جامعیت UNIQE را توسط تعریف خودکار شاخص UNIQE بر روی کلید واحد انجام می دهد. استثنای این پیشنهاد معمولا وابسته بر کارایی است. برای مثال، استفاده از CREATE TABLE ...AS SELECT با محدودیت UNIQE آهسته تر از ایجاد جدول بدون محدودیت و سپس بطور دستی ایجاد شاخص UNIQE می باشد.

اجازه دهید فرض کنیم که جدولهایی با تعریف کلید های اولیه، فرعی، و خارجی ایجاد و مشخص شده باشند. آنچه که حال باید انجام دهید مشخص کردن این است که آیا به شاخصی نیاز داریم و اینکه آیا هرگونه شاخص دیگری مورد نیاز می باشند. در فصل ۱۳، پیشنهاد کردیم که لیست خواسته ها را ایجاد کنید و سپس هر شاخص بالقوه در لیست خواسته ها را جهت مشخص کردن اینکه آیا هنگامی که بهنگام سازی رخ می دهد افزایش کارایی پرس وجو مهم تر از تنزل کارایی است را در نظر بگیرید. قبل از ایجاد لیست خواسته ها ایده خوبی که وجود دارد این است که از آنجاییکه جدولهای کوچک معمولا در حافظه بدون نیاز به شاخصهای اضافی پردازش می شوند از توجه بیشتر به جدولهای کوچک چشم پوشی کنیم. از شکل ۵-۱۹ می توانیم ببینیم که جداول کوچک در پایگاه داده *Perfect Pets* عبارتند از Clinic، Staff، Pen، Treatment، Item، و Pharmacy. بنابراین ما به این جدولها زیاد توجه نمی کنیم. اگر حالا جداول باقیمانده را در نظر بگیریم می توانیم خلاصه ای از فعل و انفعالات بین جداول پایه تراکنشهای بازیابی لیست شده در بخش ۲-۱-۱۸ را تولید کنیم، چنانکه در جدول ۲-۱۹ نشان داده شده است.

بر اساس راهنماییهای داده شده در بالا برای خوشه ها بطور آشکار دیده می شود که هیچ سود واقعی از استفاده از شاخصها عایدمان نمی شود. بنابراین، ممکن است منافع کارایی در افزودن شاخصها همچنانکه در جدول ۳-۱۹ دیده می شود وجود داشته باشد.

جدول ۲-۱۹ فعل وانفعالات بین جداول و تراکنشهای بازیابی.

Table	Transaction	Access	Frequency (per day)
Appointment	2(l)	join: petNo	250
		search condition: aDate	
Examination	2(c), 2(d)	join: Pet on petNo	100
	2(d)	join: Staff on staffNo	
Invoice	2(e), 2(f)	join: PetOwner on ownerNo	10
		search condition: datePaid IS NULL	
	2(n)	join: PetOwner on ownerNo	1 per month
		search condition: invoiceDate	
ItemClinicStock	2(q)	search condition: inStock < reorderLevel	50 per month
Pet	2(b)	join: PetOwner on ownerNo	50
	2(j)	group: petType	1
		order by: petType	
		aggregate: count on petType	
	2(l)	join: Clinic on clinicNo	250
	2(o)	join: PetOwner on ownerNo	1500
PharmClinicStock	2(p)	search condition: inStock < reorderLevel	50 per month

جدول ۳-۱۹ شاخصهای اضافی پایگاه داده *Perfect Pets*.

جدول	شاخص
Pet	clinicNo
	ownerNo
Appointment	aDate
	petNo
Invoice	ownerNo
	invoiceDate

مرحله ۱-۶ داده های مشتق شده را در نظر بگیرید

نیازمندیهای تعیین شده در بخش ۱-۱-۱۸ مشخص می کند که تنها یک بخش مشتق شده، بنام مجموع هزینه های تمام معالجات داده شده به حیوان بایستی در صورتحساب ثبت شود. الگوریتم برای مشتق کردن این اطلاعات می تواند با استفاده از دستور SQL زیرین نوشته شود:

```
SELECT SUM (pt.quantity*t.cost)
FROM invoice i INNER JOIN (examination e INNER JOIN
(treatment t INNER JOIN pettreatment pt
ON t.treatno = pt.treatno) ON e.examno = pt.examno)
ON i.examno = e.examno;
```

اگر دسترسی به جدول invoice مکررا صورت پذیرد، بوسیله ذخیره هزینه کلی در جدول invoice سودمندیهای کارایی نصیب مان می شود. بنابراین، از داده تکراری مورد انتظار معین شده در جدول ۲-۱۹ می توانید ببینید که دستیابی به جدول invoice مشخصا به صورت مکرر نیست، و بنابراین ممکن است در این مورد تصمیم بگیرید که مجموع را تنها در صورت نیاز محاسبه کنید.

مرحله ۲-٦ ستونهای تکراری یا الحاق جدولها با همدیگر را در نظر بگیرید

در مرحله ۷-۱، رابطه Appointment *Schedules* Clinic بصورت افزونه در نظر گرفته شد. بنابراین، این باعث مشکلات کارایی بالقوه ای می شود. برای مثال، جدول Pet تنها برای مشخص کردن زمان در دسترس برای قرار ملاقات هر کلینیک مشخص می شود. در این مورد، شاید آنچه که ارجحیت داشته باشد تولید نمونه دیگری از رابطه *Schedules* و افزودن کلید اصلی جدول Clinic (clinicNo) به جدول Appointment جهت عمل بعنوان کلید خارجی باشد.

مفهوم نرمالسازی دوباره را در نظر بگیرید

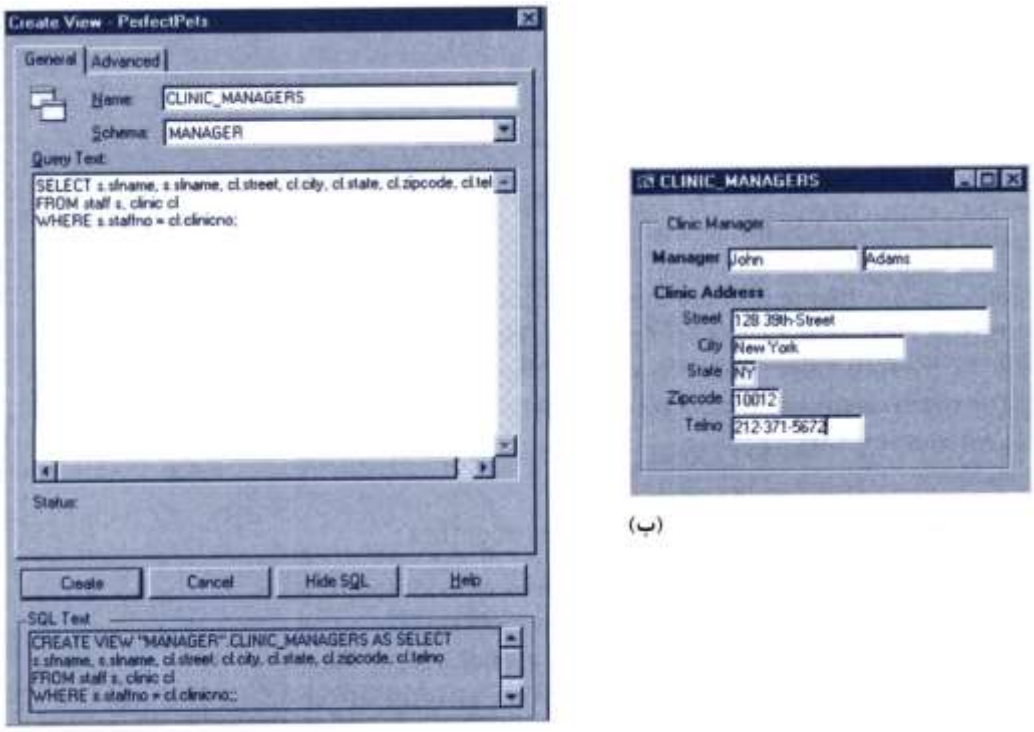
بعلت افزودن ستون ClinicNo به جدول Appointment، احتمالا ایجاد شاخص روی این ستون برای بهبود کارایی تراکنش ۲(۱) می تواند ارزنده باشد.

مرحله ۱-۷ دیدهای کاربر را طراحی کنید

اوراکل ۸ دستور SQL CREATE VIEW را پشتیبانی می کند، بنابراین دیدها می تواند به آسانی برای هر دید کاربر ایجاد شود. بعلاوه، با استفاده از Oracle Forms Builder، می توانید فرمها را بر اساس یک یا چند جدول یا بر اساس دید ایجاد کنید. برای مثال، می توانید تصمیم بگیرید که دیدی را برای جزئیات مدیر کلینیک ایجاد کنید. شکل۱۰-۱۹(الف) ایجاد دیدی بنام Clinic_managers را با استفاده از Schema Manager نشان می دهد، و شکل۱۰-۱۹(ب) فرم ساخته شده از این دید را نشان می دهد.

شکل ۱۰-۱۹

ایجاد و استفاده از دیدهای کاربر: (الف) ایجاد دید با استفاده از Oracle Schema Manager ; (ب) فرم ساخته شده از این دید.



(ب)

(الف)

مرحله ۲-۷ قواعد دسترسی را طراحی کنید

بعنوان یکی از مراحل تحلیل پایگاه داده، بایستی انواع کاربرانی را که با سیستم کار خواهند کرد و سطح دسترسی که بایستی به آنها داده شود تا کارهای مختص بخود را انجام دهند را مشخص کنید. همانطور که در مرحله ۲-۷ فصل ۱۵ ذکر شد، امنیت پایگاه داده معمولا شامل امنیت سیستم و امنیت داده است. یک شکل امنیت سیستم بکار رفته توسط اوراکل مکانیزم نام کاربر و رمز عبور می باشد، بدین صورت که کاربر قبل از اینکه بتواند به پایگاه داده دسترسی داشته باشد، مجبور به تهیه نام کاربر و رمز عبور معتبر می باشد اگرچه مسئولیت هویت کاربران می تواند به سیستم عامل محول شود. شکل۱۱-۱۹ ایجاد کاربر جدیدی بنام ADAMS با مجموعه تصدیق رمز عبور را نشان می دهد. زمانیکه کاربر ADAMS سعی دارد به پایگاه داده متصل شود، این کاربر با جعبه محاوره ای Connect یا Log On مشابه آنچه که در شکل ۱۲-۱۹ نشان داده شده است مواجه می شود، که از او می خواهد تا با وارد کردن نام کاربر و رمز عبور به پایگاه داده متصل شود.

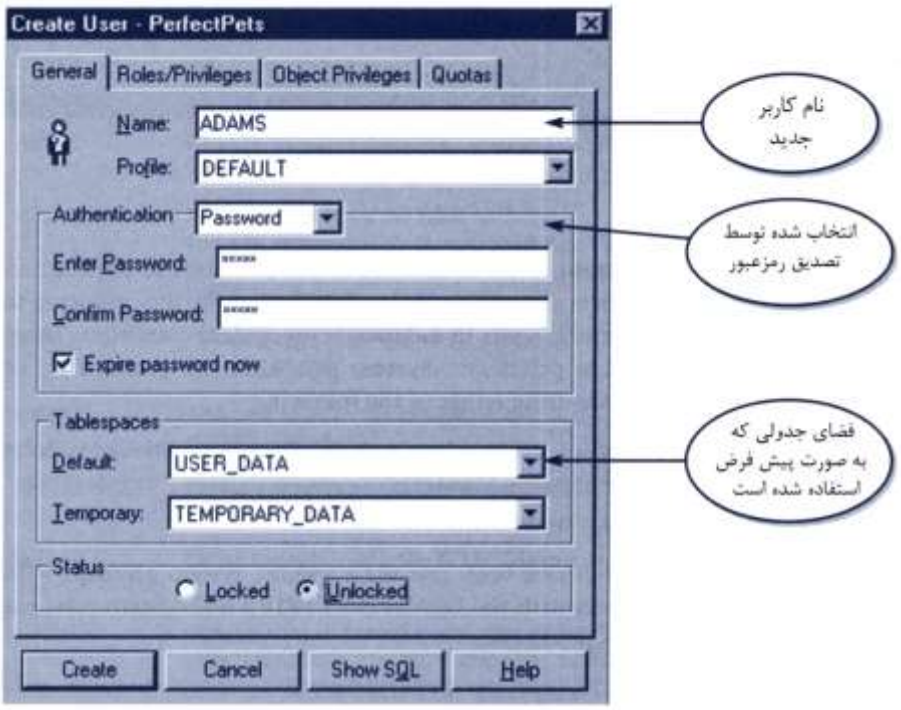
امتیازها

امتیاز[1] حقی است که برای اجرای نوع مشخصی از دستور SQL یا دسترسی به اشیا کاربران دیگر به کاربران داده می شود. چندین نمونه از امتیازها شامل اعطای حق به:

- متصل شدن به پایگاه داده (ایجاد session) ؛
- ایجاد جدول؛
- انتخاب ردیفهایی از جداول دیگرکاربران.

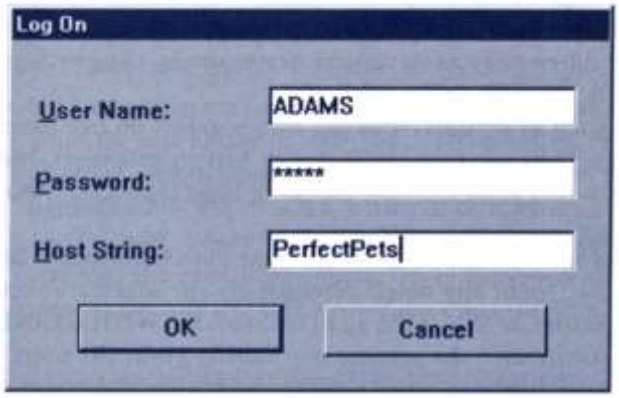
---

[1] Privilege



شکل ۱۱-۱۹

ایجاد کاربر جدید بنام ADAMS، با مجموعه تصدیق رمز عبور.



شکل ۱۲-۱۹

جعبه محاوره ای Connect که نام کاربر، رمز عبور، و پایگاه داده مورد نظر را درخواست می کند.

شما امتیازات را به کاربران اعطا می کنید بنابراین کاربران می توانند وظایف موردنیاز کارشان را انجام دهند. اعطای امتیازاتی بیش از حد مجاز به کاربران می تواند امنیت را به خطر اندازد، بنابراین باید تنها به کاربران امتیازاتی را که برای انجام کارشان به آن نیاز دارند را بدهید. در اوراکل، دو نوع امتیاز وجود دارد:

(a) امتیازات سیستم
(b) امتیازات اشیاء.

امتیازات سیستم

امتیاز سیستم حقی است که برای انجام کار خاصی یا انجام عملی روی اشیا شِمایی از نوع مشخص داده می شود. برای مثال، امتیازاتی که برای ایجاد فضاهای جدول و ایجاد کاربران پایگاه داده داده می شوند از نوع امتیازات سیستمی هستند. بیش از ۸۰ امتیاز سیستمی وجود دارد. امتیازات سیستم می توانند به کاربران یا *نقشها*[۱] اعطا شده، یا پس گرفته شوند، که با استفاده از یکی از این دو، صورت می گیرد:

[۱] Roles

- جعبه محاوره ای Grant System Privileges/ Roles و جعبه محاوره ای Revoke System Privileges/Roles از قسمت Oracle Security Manager؛
- دستورات SQL GRANT و REVOKE .

با این وجود، تنها به کاربرانی امتیازات سیستمی اعطا می شود که دارای امتیازات ADMIN OPTION بوده یا کاربرانی باشند که امتیازات GRANT ANY PRIVILEGES را داشته باشند.که در این صورت می تواند به آنها امتیازات سیستم داده یا پس گرفته شود.

امتیازات شی

امتیاز شی امتیاز یا حقی است که برای انجام عمل خاصی روی جدول، دید، دنباله، رویه، تابع، یا بسته[1] مشخصی داده میشود. امتیازات شی مختلفی برای انواع متفاوتی از اشیا موجود میباشد. برای مثال، امتیاز جهت حذف کردن سطرها از جدول Pen یک امتیاز شی است.

بعضی از اشیای شِما (همچون خوشه ها، شاخصها، تریگرها) هیچگونه امتیاز شی مربوطی را ندارند؛ کاربرد آنها با امتیازات سیستم کنترل می شود. برای مثال، جهت اصلاح خوشه، کاربر باید صاحب خوشه باشد یا امتیاز سیستم ALTER ANY CLUSTER را داشته باشد.

کاربر به طور خودکار تمام سیستمهای شی را برای اشیا شِمای دربرگرفته شده در شِمای او دارا می باشد.کاربر می تواند هر امتیاز شی ای را که در اختیار دارد به دیگر کاربران اعطا کند. اگر اعطا شامل WITH GRANT OPTION (از دستور GRANT) باشد ، صاحب امتیاز می تواند امتیاز شی را به کاربران دیگر اعطا کند؛ در غیر اینصورت، صاحب امتیاز می تواند از امتیاز استفاده کند ولی نمیت واند به دیگر کاربران اعطا کند.

امتیازات شی برای جداول و دیدها در جدول ۴-۱۹ نشان داده شده است.

نقشها

کاربر می تواند امتیاز را به دو روش مختلف دریافت کند:

- شما می توانید امتیازات را به کاربران آشکارا اعطا کنید. برای مثال، می توانید امتیاز جهت درج رکوردها در جدول Clinic را به کاربر ADAMS اعطا کنید.

برای مثال:

```
GRANT INSERT ON clinic TO ADAMS ;
```

- شما همچنین می توانید امتیازات را به نقش (گروهی از امتیازات را می نامیم) اعطا کنید، و سپس نقش را به یک یا چند کاربر عطا کنید. مثلا، می توانید امتیازات را به انتخاب، درج، بهنگام سازی، و حذف رکوردها از جدول Clinic به نقشی با نام DEPUTYMANAGER عطا کنید، که بترتیب می تواند به کاربران ADAMS و GLENN اعطا شود. کاربر می تواند به چندین نقش دسترسی داشته، و به چندین کاربر می تواند نقشهای مشابه اختصاص داده شود. شکل ۱۳-۱۹ اعطای امتیازات به نقش DEPUTYMANAGER را با استفاده از مدیریت امنیت اوراکل[2] نشان می دهد.

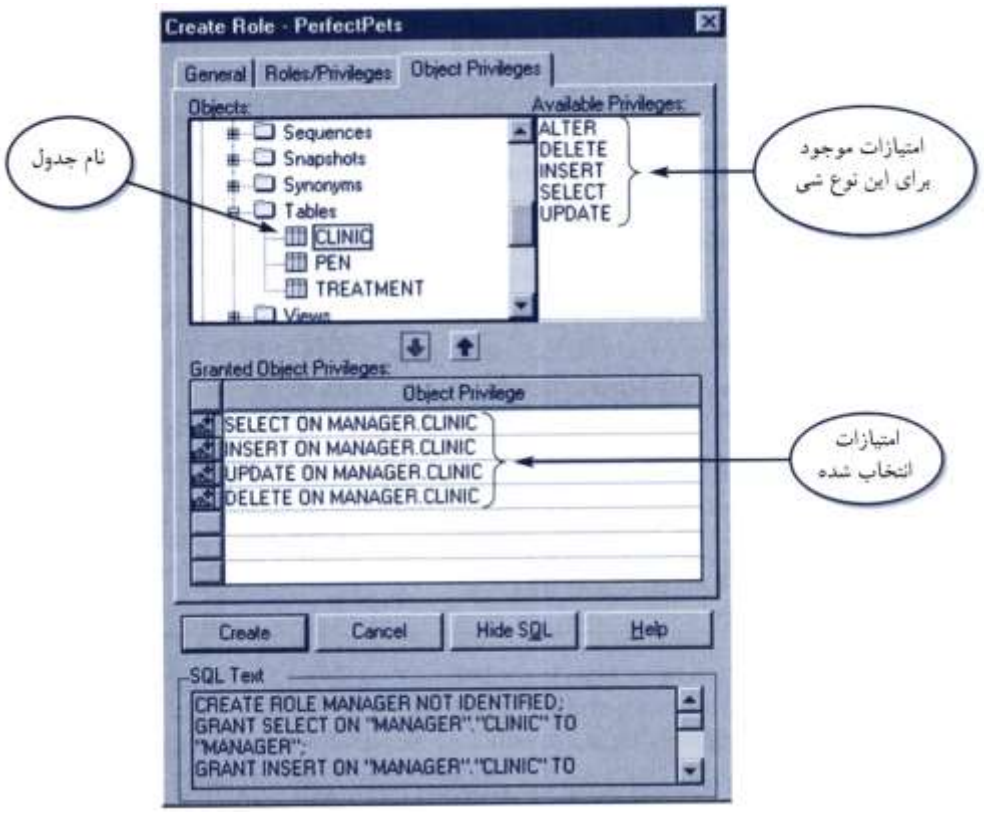
بعلت اینکه نقشها جهت سادگی و مدیریت بهتر امتیازات مجاز می باشند، شما باید معمولا امتیازات را به نقشها عطا کنید نه کاربران مشخصی.

---

[1] Package
[2] Oracle Security Manager

شکل ۱۳-۱۹

دادن امتیازات انتخاب، درج، بهنگام سازی، و حذف بر روی جدول Clinic به نقش DEPUTYMANAGER.



پیاده سازی

هم اکنون در موقعیتی هستید که پیاده سازی جداول پایه، سازمان فایلها، شاخصها، دیدها، و مکانیزمهای امنیت را شروع کنید و، بعداز آن انتشار پایگاه داده را آغاز کنید. با این وجود، همانطور که در فصل ۱۶ بحث شد، این پایان طراحی پایگاه داده نیست – بازبینی مداوم و نظارت بر سیستم کارکردی فعالیت حیاتی برای رسیدن به موفقیت مداوم سیستم است. بعلاوه بشدت احتمال دارد که بعد از اینکه سیستم شروع به فعالیت کرد تغییراتی بعنوان نتایج بازخورد کاربران و نیازمندیهای دیگر نیاز باشد. در بیشتر موارد، تغییرات تنها بصورت تزیینی بوده و نیاز به اطلاح واسط های کاربری دارند که بدون تاثیری روی خود پایگاه داده صورت می گیرند. دیگر، با این وجود ممکن است نیاز به اصلاح ساختار پایگاه داده داشته باشیم و در اینگونه موارد است که، مجبورید برای اطمینان از اینکه تغییرات به طور صحیح طراحی و پیاده سازی شوند بعضی از مراحل متدلوژی طراحی فیزیکی منطقی را دوباره انجام دهید.

ضمیمه الف

دیگر نمادگذاری های مدلسازی داده

آنچه در این ضمیمه خواهید آموخت :

- دیگر نماد گذاری های مدل سازی داده

در فصل ۵، آموختید که چگونه با استفاده از نماد در حال توسعه UML مدل موجودیت-رابطه (ER) را ایجاد کنید.
در این ضمیمه دو نماد متناوب دیگر را که اغلب جهت ایجاد مدلهای داده ای استفاده می شود نشان می دهیم. مدل ER اولی نماد Chen و دومی نماد Crow's Feet نامیده می شود. ما هر کدام را بوسیله نمایش جدولی که نماد استفاده شده برای هر یک از مفاهیم اصلی مدل ER را نشان می دهد نمایش می دهیم و سپس نمادگذاری را با استفاده از مثال مدل ER نشان داده شده در شکل ۱۶-۸ ارائه می کنیم.

۱-الف مدلسازی ER با استفاده از نمادگذاری Chen

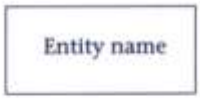
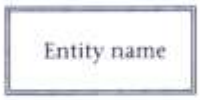
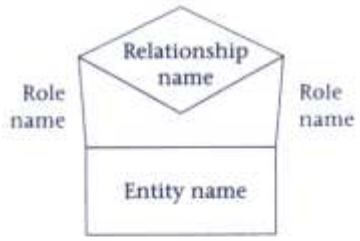
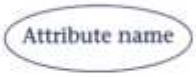
جدول ۱- الف نمادگذاری Chen را برای مفاهیم اصلی مدل ER و شکل ۱- الف مدل ER شکل۱۶-۸ را که دوباره ترسیم شده را با استفاده از نمادگذاری Chen نشان می دهد.

۲-الف مدلسازی ER با استفاده از نمادگذاری Crow's Feet

جدول ۲- الف نمادگذاری Crow's Feet را برای مفاهیم اصلی مدل ER و شکل ۲- الف مدل ER شکل ۱۶-۸ را که دوباره ترسیم شده است را با استفاده از نمادگذاری Crow's Feet نشان می دهد.

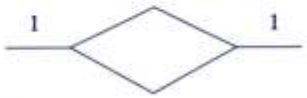
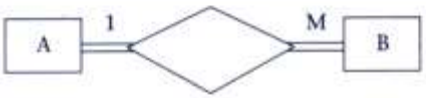
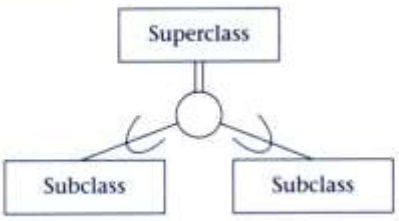
جدول ۱- الف

نماد Chen برای مدلسازی ER .

نماد	معنی
	Strong entity
	Weak entity
	Relationship
	Relationship associated with a weak entity
	Recursive relationship with role names to identify the roles played by the entity in the relationship
	Attribute
	Primary key attribute
	Multi-valued attribute

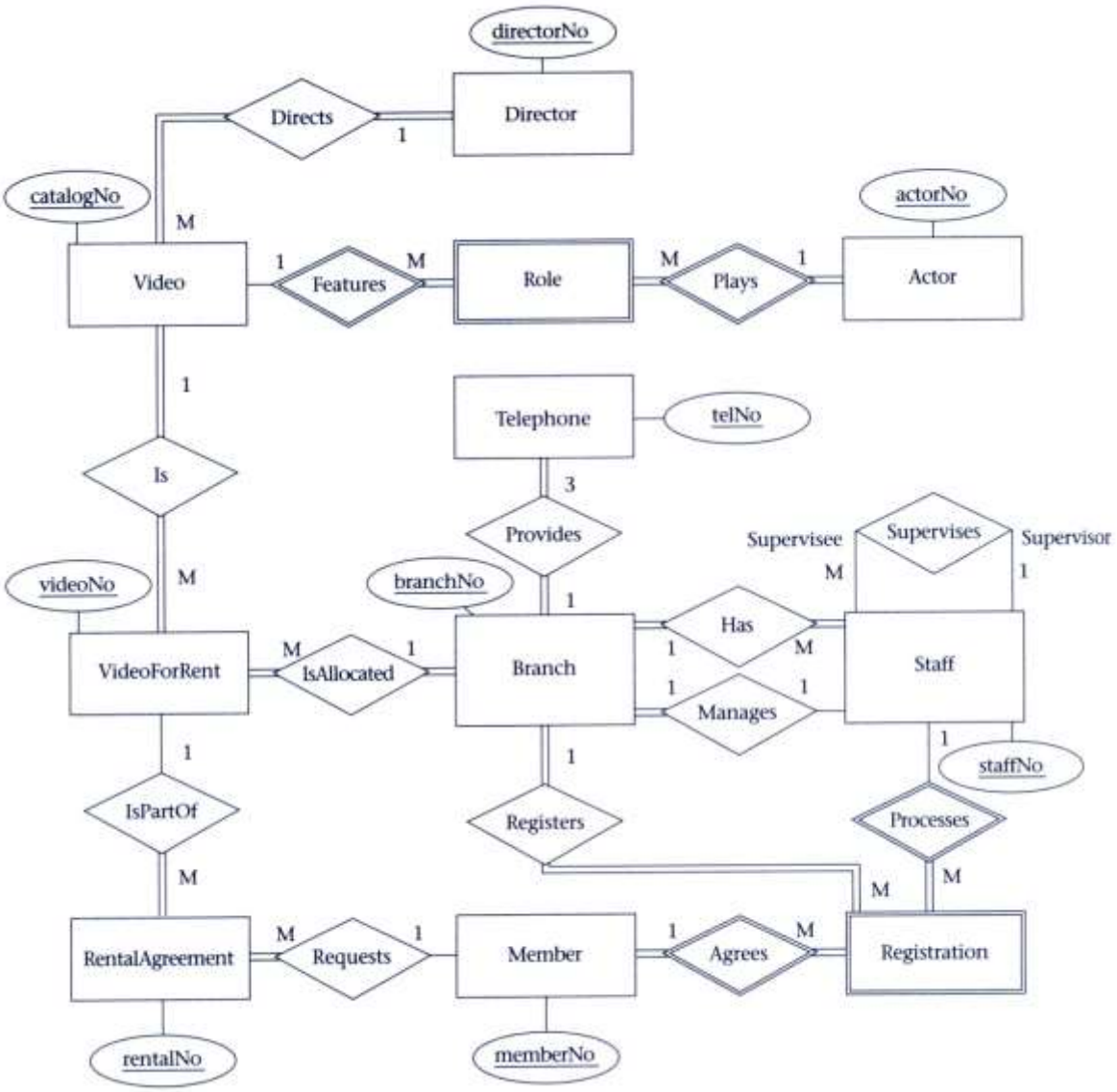
جدول ۱- الف

ادامه . . .

	Derived attribute
	One-to-one (1:1) relationship
	One-to-many (1:M) relationship
	Many-to-many (M:N) relationship
	One-to-many relationship with mandatory participation for both entities A and B
	One-to-many relationship with optional participation for entity A and mandatory participation for entity B
	One-to-many relationship with optional participation for both entities A and B
	Generalization/Specialization. If circle contains 'd' relationship is disjoint; if circle contains 'o' relationship is nondisjoint. Double lines from superclass represents mandatory participation; single line represents optional participation

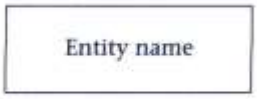
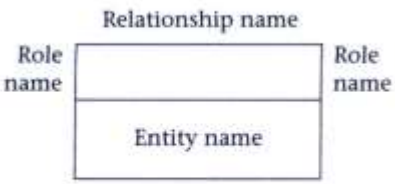
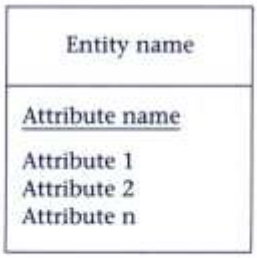
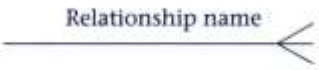
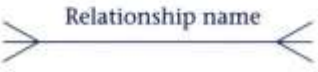
شکل ۱- الف

مدل ER شکل ۱۶-۸ با استفاده از نماد گذاری Chen .



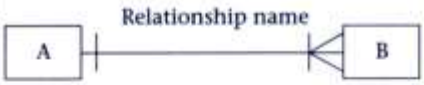
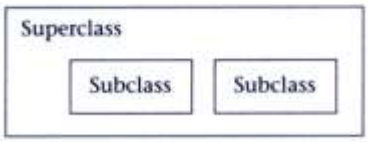
جدول ۲-الف

نمادگذاری Crow's Feet برای مدلسازی ER .

نماد	معنی
	Entity
	Relationship
	Recursive relationship with role names to identify the roles played by the entity in the relationship
	Attributes are listed in the lower section of the entity symbol The primary key attribute is underlined Multi-valued attribute placed in curly braces { }
	One-to-one relationship
	One-to-many relationship
	Many-to-many relationship

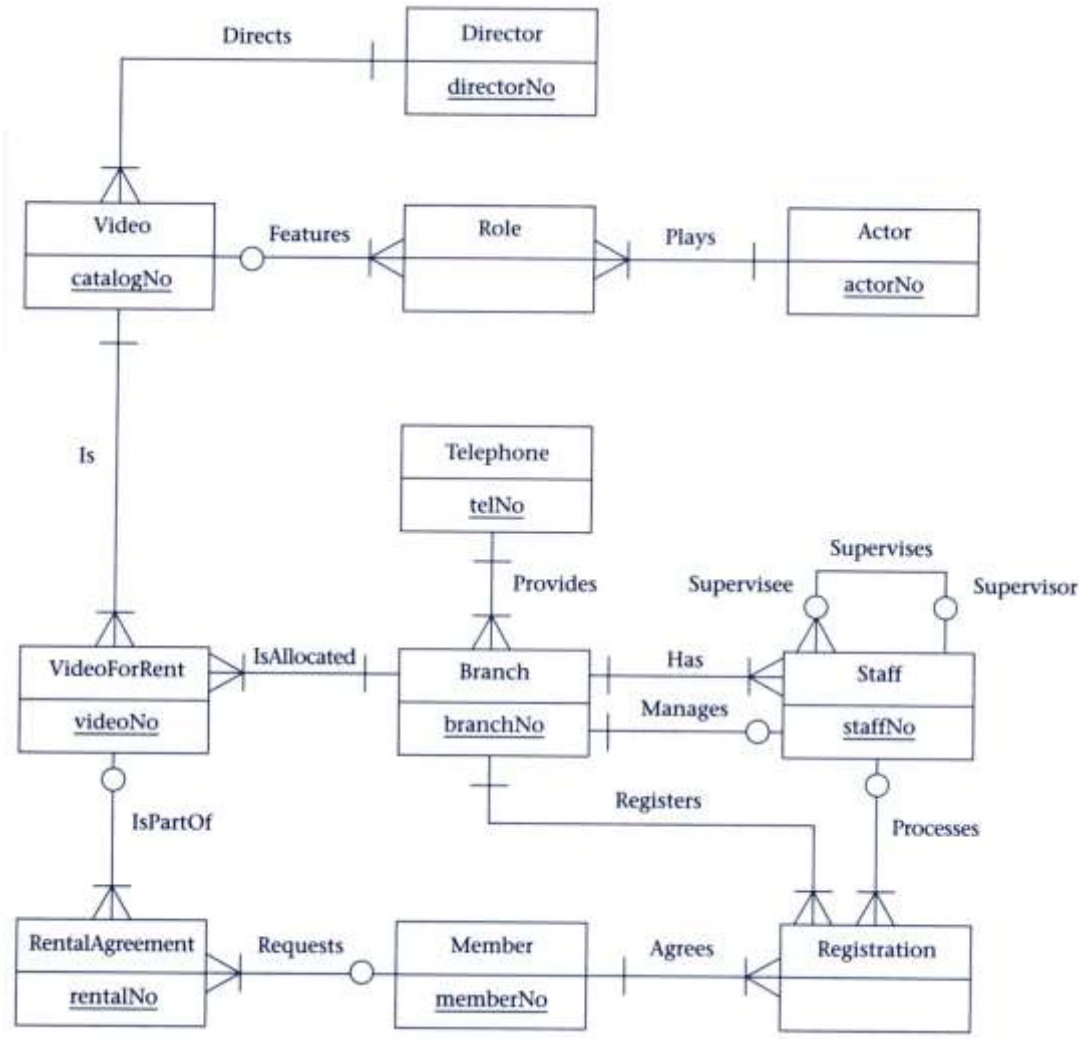
جدول ۲-الف

ادامه . . .

	One-to-many relationship with mandatory participation for both entities A and B
	One-to-many relationship with optional participation for entity A and mandatory participation for entity B
	One-to-many relationship with optional participation for both entities A and B
	'Box-in-box' convention is widely used to represent generalization/ specialization, and supported by several CASE tools, including Oracle CASE Designer.

شکل ۲-الف

مدل ER شکل ۱۶-۸ که با استفاده از نمادگذاری Crow's Feet دوباره رسم شده است.



ضمیمه ب

خلاصه متدلوژی طراحی پایگاه داده

آنچه در این ضمیمه خواهید آموخت:

- طراحی پایگاه داده از دو فاز اصلی تشکیل شده است: طراحی پایگاه داده منطقی و فیزیکی.
- مراحل در برگرفته شده در فاز های اصلی متدلوژی طراحی پایگاه داده.

ما در این کتاب، متدلوژی طراحی پایگاه داده های رابطه ای را ارائه کردیم. این متدلوژی از دو فاز اصلی، طراحی منطقی پایگاه داده و طراحی فیزیکی پایگاه داده تشکیل شده است، که با جزئیات در فصلهای ۷ تا ۱۶ توضیح داده شدند. در این ضمیمه، مراحل در برگرفته شده در این فازها را برای آن دسته از خوانندگانی که درحال حاضر با طراحی پایگاه داده آشنایی دارند به طور خلاصه ارائه می کنیم.

مرحله ۱ برای هر دید مدل داده منطقی محلی را بسازید

هنگام تحلیل، تعدادی از دیدهای کاربر را مشخص کرده اید. بر اساس میزان همپوشانی این دیدها، برای اینکه دیدها قابل مدیریت باشند ممکن است تصمیم به ادغام بعضی از دیدها بگیرید. هدف این مرحله ساختن مدل داده منطقی محلی شرکت (یا بخشی از آن) برای هریک از این دیدها (احتمالا ادغام شده) می باشد.

مرحله ۱-۱ موجودیتها را مشخص کنید

موجودیتهای اصلی که در دید شرکت وجود دارند را مشخص کرده و آنها را مستند کنید.

مرحله ۱-۲ رابطه ها را مشخص کنید

رابطه های مهم موجود بین موجودیتهایی که مشخص کرده اید را شناسایی کنید. همچنین محدودیتهای کثرت رابطه ها را نیز مشخص کنید. رابطه ها را نیز مستند کنید. در صورت لزوم از مدلسازی موجودیت-رابطه (ER) نیز استفاده کنید.

مرحله ۱-۳ صفات را مشخص کرده و آنها را با موجودیتها یا رابطه ها مرتبط کنید

صفات را با موجودیتها یا رابطه های مناسب مرتبط کنید. صفات ساده/ ترکیبی، تک مقداره/ چند مقداره، و همچنین صفات مشتق را مشخص کنید. صفات را مستند کنید.

مرحله ٤-١ دامنه های صفات را مشخص کنید

دامنه های صفات مدل داده منطقی محلی را مشخص کنید. دامنه های صفات را مستند کنید.

مرحله ٥-١ صفات کلید کاندید و اصلی را مشخص کنید

کلیدهای کاندید هر موجودیت را شناسایی کنید، و اگر بیش از یک کلیدکاندید وجود دارد یکی از آنها را بعنوان کلید اصلی و دیگری را بعنوان کلید فرعی انتخاب کنید. کلیدهای کاندید، اصلی، و فرعی هر موجودیت قوی را مستند کنید.

مرحله ٦-١ موجودیتها را تخصص/ تعمیم دهید(مرحله اختیاری)

هر کجا که نیاز باشد موجودیتهای سوپرکلاس و زیرکلاس را، مشخص کنید.

مرحله ٧-١ خصوصیاتی که با مدل رابطهای سازگار نیستند حذف کنید.

با حذف خصوصیات ناخواسته مدل داده منطقی محلی را اصلاح کنید. رابطه های باینری چند به چند (*:*)، رابطه های بازگشتی چند به چند(*:*) ، رابطه های پیچیده، صفات چند مقداره را حذف کنید، رابطه های ۱:۱ را دوباره بررسی کنید، و رابطه های افزونه را حذف کنید.

رابطه ٨-١ مدل داده ای را جهت پشتیبانی از تراکنشهای کاربر بررسی کنید

مطمئن شوید که مدل منطقی محلی تراکنشهای موردنیاز توسط دیدها را پشتیبانی می کند.

مرحله ٢ جدولهای هر مدل داده منطقی محلی را ایجاد کرده وآنها را بررسی کنید

ایجاد جداول برای هر مدل داده منطقی محلی، و بررسی ساختار جداول.

مرحله ١-٢ برای مدل داده منطقی محلی جداول را ایجاد کنید

در این مرحله جداول پایه مدل داده منطقی محلی را برای نشان دادن موجودیتها، رابطه ها، صفات، و محدودیتهای توصیف شده در دید شرکت ایجاد کنید. ساختار جداول از اطلاعاتی مشتق شده اند که مدل داده منطقی محلی را توصیف می کند. این اطلاعات در برگیرنده مدل ER، دیکشنری داده، و هر مستند دیگری که مدل را توصیف می کند می باشد. همچنین، هر گونه کلیدهای اصلی یا کاندید جدید دیگری را که در نتیجه فرایند ایجاد جداول برای مدل داده منطقی محلی مشتق شده اند را ایجاد کنید. قواعد ساده جهت ایجاد جداول بدین صورت می باشند:

(a) برای هر موجودیت، جدولی را ایجاد کنید که شامل همه صفات ساده موجودیت باشد.

(b) هر رابطه توسط مکانیزم کلید اصلی/ خارجی نشان داده می شوند. برای تصمیم گیری درباره اینکه کجا کلید خارجی را بفرستید، بایستی در رابطه موجودیتهای 'والد' و 'فرزند' را شناسایی کنید. سپس موجودیت والد کپی کلید اصلی خودش را بداخل جدول فرزند، جهت عمل بعنوان کلید خارجی می فرستد. جدول ب-۱ خلاصه ای از اینکه چگونه موجودیتهای والد و فرزند را شناسایی کنید را ارائه می کند.

(c) برای هر رابطه سوپرکلاس/ زیر کلاس، سوپرکلاس را بعنوان موجودیت والد و زیرکلاس را بعنوان موجودیت فرزند مشخص کنید. گزینه های متفاوتی درباره اینکه چگونه ممکن است از بهترین نمایشی از رابطه بعنوان یک یا چند جدول را نشان دهید موجود می باشد. انتخاب گزینه مناسب تر بسته بر محدودیتهای شرکت و غیراتصال برای رابطه های سوپر کلاس/ زیرکلاس، همچنانکه در جدول ب-۲ نشان داده شده است می باشد.

جدول ب-۱ خلاصه ای از چگونگی شناسایی موجودیتهای والد و فرزند در رابطه.

رابطه	شناسایی
۱:* رابطه باینری	**والد:** طرف ۱ ; **فرزند:** طرف *
۱:* رابطه بازگشتی	**والد:** طرف ۱; **فرزند:** طرف *
۱:۱ رابطه باینری:	
الف) اشتراک اجباری در دو طرف	جدولها را در یک جدول ادغام کنید
ب) اشتراک اجباری در یک طرف	**والد:** طرف اختیاری;**فرزند:** طرف اجباری
ج) اشتراک اختیاری در دو طرف	اختیاری بدون اطلاعات بیشتر
۱:۱ رابطه بازگشتی:	
الف) اشتراک اجباری در دو طرف	جدولها را در یک جدول با دو کپی از کلیداصلی
ب) اشتراک اجباری در یک طرف	همچون الف)، یا جدول جدید جهت نشان دادن رابطه ایجاد کنید
ج) اشتراک اختیاری در دو طرف	جدول جدیدی برای نشان دادن رابطه ایجاد کنید

جدول ب-۲ گزینه های در دسترس برای نمایش رابطه سوپرکلاس/ زیرکلاس براساس محدودیتهای اشتراک و انفصال.

محدودیت اشتراک	محدودیت انفصال	جداول موردنیاز
اجباری	غیرمنفصل {And}	یک جدول
اختیاری	غیرمنفصل {And}	دو جدول: یک جدول برای سوپرکلاس و یک جدول برای همه زیرکلاسها
اجباری	منفصل {Or}	چندین جدول:: یک جدول برای هر سوپرکلاس/ زیرکلاس ترکیبی
اختیاری	منفصل {Or} و یکی برای هر زیرکلاس	چندین جدول: یک جدول برای سوپرکلاس

مرحله ۲-۲ ساختار جداول را با استفاده از نرمالسازی بررسی کنید

هدف این مرحله بررسی گروهبندی ستونهای جدولهای ایجاد شده در مرحله ۱-۲ میباشد. ترکیب هر جدول را با استفاده از قواعد نرمالسازی بررسی کنید. هر جدول بایستی حداقل در فرم نرمال سوم باشد (3NF).

مرحله ۳-۲ جداول را جهت پشتیبانی از تراکنشهای کاربر بررسی کنید

در این مرحله، مطمئن شوید که جداول از تراکنشهای موردنیاز دید پشتیبانی کنند. تراکنشهایی که توسط دید موردنیاز می باشند می توانند از خصوصیات نیازمندیهای کاربر مشخص شوند.

مرحله ٤-۲ محدودیتهای جامعیت را تعریف کنید

محدودیتهای جامعیت معین شده در دید شرکت را مشخص کنید. این شامل تعیین داده موردنیاز، محدودیتهای دامنه صفت، جامعیت موجودیت، جامعیت ارجاع، و قواعد تجاری می باشند. همه محدودیتها ی جامعیت را مستند کنید.

مرحله ٥-۲ مدل داده منطقی محلی را با کاربران بازبینی کنید

مطمئن شوید که مدل داده منطقی محلی نمایش صحیحی از دید شرکت تحت مدل (یا بخشی از آن) می باشد.

مرحله ۳ مدل داده منطقی سراسری را ساخته و بررسی کنید

مدلهای داده منطقی محلی منفرد را داخل مدل داده منطقی سراسری منفرد که بیانگر شرکت تحت مدل می باشد با یکدیگر ترکیب کنید.

مرحله ۱-۳ مدلهای داده منطقی محلی را داخل مدل سراسری ادغام کنید

مدلهای داده منطقی محلی واحد را داخل یک مدل داده منطقی سراسری ادغام کنید. بعضی از کارهایی که در این مرحله باید انجام گیرد بصورت زیر می باشند:

(۱) نام موجودیتها/ جداول و کلیدهای اصلیشان را بازبینی کنید.
(۲) نامهای رابطه ها را بازبینی کنید.
(۳) موجودیتها/ جداول را از مدلهای داده محلی ادغام کنید.
(۴) موجودیتها/ جداول منحصر به هر مدل داده محلی را (بدون ادغام) شامل کنید.
(۵) رابطه ها را از مدل داده منطقی محلی ادغام کنید.
(۶) رابطه های منحصر به هر مدل داده محلی را (بدون ادغام) شامل کنید .
(۷) موجودیتها/ جداول و رابطه های گم شده را بررسی کنید .
(۸) کلیدهای خارجی را بررسی کنید .
(۹) محدودیتهای جامعیت را بررسی کنید .
(۱۰) مدل داده منطقی سراسری را ترسیم کنید .
(۱۱) مستندات را بهنگام کنید.

مرحله ۲-۳ مدل داده منطقی سراسری را بررسی کنید

این مرحله معادل مراحل ۳-۲ و ۴-۲ می باشد که شما در این مرحله ساختار جداول ایجاد شده برای مدل داده سراسری را با استفاده از نرمال سازی بررسی می کنید و سپس بررسی می کنید که این جداول توانایی پشتیبانی از همه تراکنشهای کاربر داشته باشد.

مرحله ۳-۳ قابلیت رشد در آینده را بررسی کنید

مشخص کنید که آیا احتمال تغییرات مهم دیگر در آینده نزدیک وجود دارد و تعیین کنید که مدل داده منطقی سراسری با این تغییرات قابل تطبیق باشد.

مرحله ٤-۳ مدل داده منطقی سراسری را با کاربران بازبینی کنید

مطمئن شوید که مدل داده منطقی سراسری نمایش صحیحی از شرکت (یا بخشی از شرکت تحت مدل) می باشد

مرحله ۴ مدل داده منطقی سراسری را به DBMS هدف ترجمه کنید

تولید مجموعه کاری اساسی جداول پایه مدل داده منطقی سراسری.

مرحله ۱-٤ جداول پایه DBMS هدف را طراحی کنید

تصمیم بگیرید که چگونه جداول پایه را که در مدل داده منطقی سراسری مشخص کرده اید در DBMS هدف نشان دهید. طراحی جداول را مستند کنید.

مرحله ۲-۴ قواعد تجاری DBMS هدف را طراحی کنید

قواعد تجاری DBMS هدف را طراحی کنید. طراحی قواعد تجاری را مستند کنید.

مرحله ۵ طراحی نمایش فیزیکی

سازمان فایلی را که جهت ذخیره جداول پایه استفاده خواهد شد را مشخص کنید؛ یعنی، راهی را که جداول و رکوردها در وسیله ذخیره سازی جانبی نگه داشته خواهند شد را مشخص کنید. افزودن شاخصها جهت بهبود کارایی را در نظر بگیرید.

مرحله ۵-۱ تراکنشها را تحلیل کنید

عملکرد تراکنشهایی را که روی پایگاه داده اجرا خواهند شد را درک کرده و تراکنشها تحلیل کنید.

مرحله ۵-۲ سازمانهای فایل را انتخاب کنید

سازمان فایل موثر برای هر جدول پایه را مشخص کنید

مرحله ۵-۳ شاخها را انتخاب کنید

مشخص کنید که آیا افزودن شاخصها کارایی سیستم را افزایش خواهند داد.

مرحله ۶ معرفی افزونگی کنترل شده را در نظر بگیرید

مشخص کنید که آیا معرفی افزونگی به شیوه کنترل شده با کم کردن قواعد نرمالسازی، کارایی سیستم را افزایش خواهد داد.

مرحله ۶-۱ داده های مشتق شده را درنظر بگیرید

در نظر بگیرید که چگونه داده مشتق شده نمایش داده خواهد شد. یعنی اینکه آیا داده مشتق شده را هر زمانی که نیاز داریم محاسبه کنیم یا افزونگی را در نظر بگیریم و داده مشتق شده را بصورت ستون در جدول نشان دهیم.

مرحله ۶-۲ ستونها ی تکراری یا الحاق جداول به همدیگر را در نظر بگیرید

ستونهای تکراری یا الحاق جدولها به همدیگر را جهت رسیدن به کارایی بهبود یافته در نظر بگیرید. خصوصا، ترکیب رابطه های یک به یک (۱:۱)، تکرار ستونهای غیرکلید در رابطه های یک به چند(*:۱) جهت کاهش الحاقها، تکرار ستونهای کلید خارجی در رابطه های یک به چند(۱:*) جهت کاهش الحاقها، تکرار ستونها در رابطه های چند به چند(*:*) جهت کاهش الحاقها، معرفی گروهای تکرار، ادغام جداول جستجوگر[۱] با جداول پایه، و ایجاد جداول خلاصه را در نظر بگیرید.

مرحله ۷ مکانیزمهای امنیت را طراحی کنید

مقیاسهای امنیت را جهت پیاده سازی پایگاه داده تعیین شده کاربر طراحی کنید.

[۱] Lookup Table

مرحله ۱-۷ دیدهای کاربر را طراحی کنید

دیدهای کاربری را که هنگام مرحله تحلیل مشخص شده است را طراحی کنید.

مرحله ۲-۷ قواعد دستیابی را طراحی کنید

قواعد دستیابی به جداول پایه و دیدهای کاربر را طراحی کنید. مقیاسهای امنیت و دیدهای کاربر را مستند کنید.

مرحله ۸ سیستم عملکردی را میزان و بازبینی کنید

سیستم کارکردی را بازبینی کرده و کارایی سیستم را با اصلاح تصمیمهای طراحی غیر مناسب یا انعکاس نیازمندیهای تغییرات بهبود بخشید.

ضمیمه د

مدلهای داده متداول

آنچه در این فصل خواهید آموخت:

- توضیحات بیشتر درباره ساختن مدلهای داده منطقی.
- درباره مدلهای داده منطقی متداول.

در این ضمیمه، بعضی از مدلهای داده متداولی را که می تواند مفید باشد معرفی می کنیم. درحقیقت، برآورد شده است که این مدلها یک سوم از عمومی ترین مدلهای داده ای هستند که قابل اعمال به بیشتر شرکتها می باشد و دو سوم باقیمانده یا مخصوص صنعت یا مخصوص شرکت می باشند. بنابراین، بیشتر کار مدلسازی داده ایجاد دوباره ساختهایی است که زمانهای قبل در دیگر شرکتها تولید شده اند.

بنابراین دو هدف عمده این ضمیمه فراهم کردن موارد زیر است:

(۱) دانش اضافی درباره ساختن مدلهای داده؛

(۲) الگوهای مدل داده که می تواند در شرکت شما نیز مفید واقع شود. مدلهایی که در اینجا ارائه شده است شرکت شما را به طور دقیق نشان نمی دهد، اما می توانند نقطه آغازی جهت آنچه که شما بتوانید مدل مناسبی را جهت برآوردکردن نیازمندیهای شرکت توسعه دهید باشد.

ما مدلهایی را برای نواحی تجاری متداول زیر ارائه کرده ایم:

- ثبت سفارش مشتری
- کنترل موجودی
- مدیریت سرمایه
- مدیریت پروژه
- مدیریت آموزش
- مدیریت منبع انسانی
- مدیریت حقوق

همچنین مدلهای داده زیرین را که کمتر متداول بوده اما هنوز می توانند هم از لحاظ دید تجاری و هم از جنبه یادگیری مفید باشد را نیز ارائه می کنیم.

- کرایه وسیله نقلیه
- اسکان دانشجو
- حمل ونقل مشتری
- ناشر چاپ
- کتابخانه منطقه
- اجاره ملک
- آژانس مسافرتی
- نتایج دانشجو

در هر مورد، توضیح مختصری از نیازمندیهای مساله را ارایه کرده، و مثالهایی از مدل داده منطقی را نشان می دهیم و سپس مدل را بصورت مجموعه ای از جدولها نگاشت می کنیم. در اینجا فرض بر این است که شما با نمادگذاری مدل سازی بکار رفته از طریق این کتاب آشنا شده اید. اگر چنین نیست، برای مطالعه مدل ER به فصل ۵ مراجعه کنید، که در آنجا مفاهیم اصلی و نمادگذاری که در این ضمیمه از آن استفاده می کنیم را شرح می دهد. همچنین میتوانید خلاصه ای از متدلوژی را در ضمیمه ب پیدا کنید.

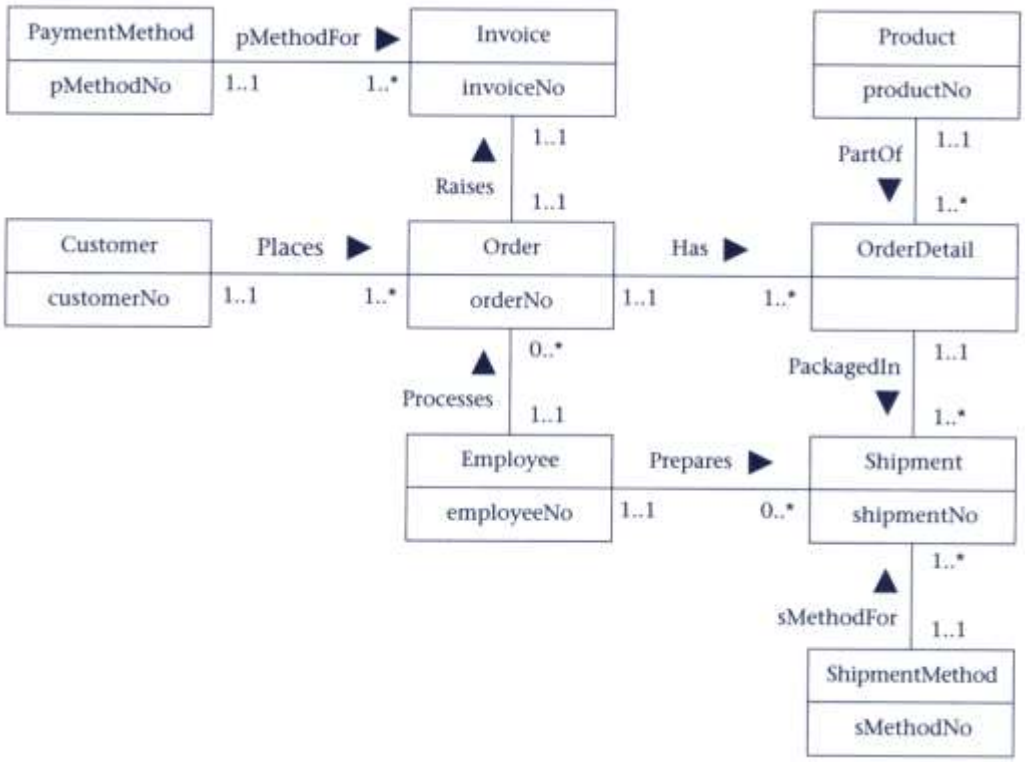
۱- د ثبت سفارش مشتری

شرکتی می خواهد پایگاه داده ای را جهت ثبت فعالیتهای سفارش ایجاد کند. مشتری می تواند یک یا چند سفارش بدهد، که هر سفارش برای یک یا چند محصول می باشد. به هر سفارش یک صورتحساب ارائه می شود، که می تواند به چندین روش پرداخته شود، همچون چک، کارت اعتباری، یا پول نقد. نام کارمندی که در ابتدا سفارش مشتری را پردازش می کند ثبت می شود.

کارمند موجود در بخش حمل مسئول بسته بندی سفارش و فرستادن آن به مشتری است. اگر محصول سفارشی در انبار نباشد، باید باربری اعزام شود تا ببیند چه چیزی در انبار موجود است بنابراین شاید بیش از یک محموله نیاز باشد تا سفارش انجام گیرد. مدل داده منطقی در شکل د-۱ و جداول مربوطه در شکل د-۲ نشان داده شده است.

شکل د-۱

مدل منطقی داده برای ثبت سفارش مشتری .



شکل ۲ – د

جداول ثبت سفارش مشتری.

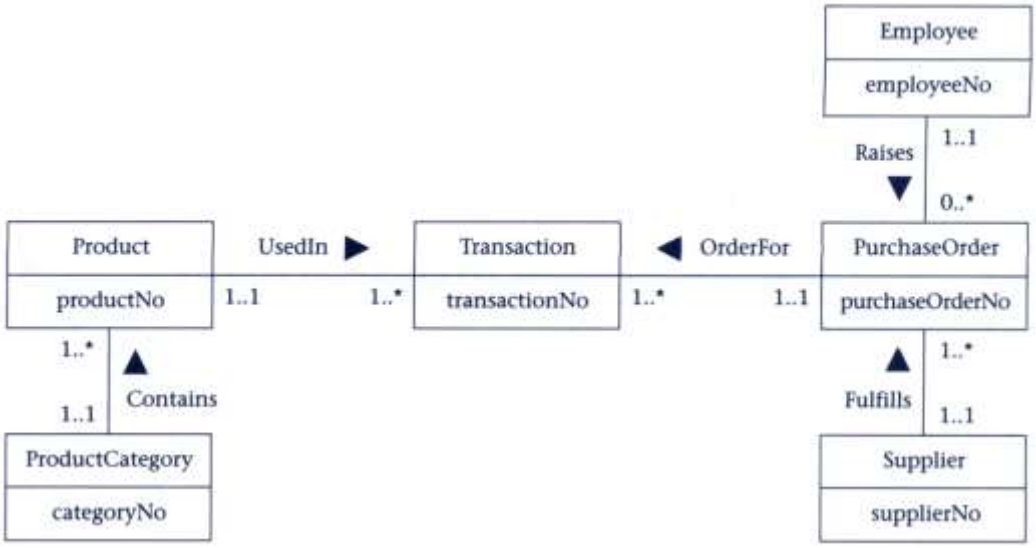
Customer	(customerNo, customerName, customerStreet, customerCity, customerState, customerZipCode, custTelNo, custFaxNo, DOB, maritalStatus, creditRating) Primary Key customerNo Alternate Key custTelNo Alternate Key custFaxNo
Employee	(employeeNo, title, firstName, middleName, lastName, address, workTelExt, homeTelNo, empEmailAddress, socialSecurityNumber, DOB, position, sex, salary, dateStarted) Primary Key employeeNo Alternate Key socialSecurityNumber
Invoice	(invoiceNo, dateRaised, datePaid, creditCardNo, holdersName, expiryDate, orderNo, pMethodNo) Primary Key invoiceNo Foreign Key orderNo references Order(orderNo) Foreign Key pMethodNo references PaymentMethod(pMethodNo)
Order	(orderNo, orderDate, billingStreet, billingCity, billingState, billingZipCode, promisedDate, status, customerNo, employeeNo) Primary Key orderNo Foreign Key customerNo references Customer(customerNo) Foreign Key employeeNo references Employee(employeeNo)
OrderDetail	(orderNo, productNo, quantityOrdered) Primary Key orderNo, productNo Foreign Key orderNo references Order(orderNo) Foreign Key productNo references Product(ProductNo)
PaymentMethod	(pMethodNo, paymentMethod) Primary Key pMethodNo
Product	(productNo, productName, serialNo, unitPrice, quantityOnHand, reorderLevel, reorderQuantity, reorderLeadTime) Primary Key productNo Alternate Key serialNo
Shipment	(shipmentNo, quantity, shipmentDate, completeStatus, orderNo, productNo, employeeNo, sMethodNo) Primary Key shipmentNo Foreign Key orderNo, productNo references OrderDetail(orderNo, productNo) Foreign Key employeeNo references Employee(employeeNo) Foreign Key sMethodNo references ShipmentMethod(sMethodNo)
ShipmentMethod	(sMethodNo, shipmentMethod) Primary Key sMethodNo

۲- د کنترل موجودی

شرکتی می خواهد پایگاه داده ای را جهت کنترل موجودی اش ایجاد کند که، شامل تعدادی از محصول است که به چندین فهرست، همچون پوشاک، غذا، و لوازم تحریر تقسیم شده است. زمانیکه محصول به تهیه کننده دوباره سفارش داده می شود کارمند سفارش خرید را ایجاد می کند. رکوردهای پیگیری، واحدهای فروخته شده و هرگونه ضایعات دیگر را تدارک میبیند. مدل داده منطقی در شکل د-۳ و جداول مربوطه در شکل د- ۴ نشان داده شده است.

شکل د-۳

مدل منطقی داده کنترل موجودی .



شکل د- ۴

جداول کنترل موجودی .

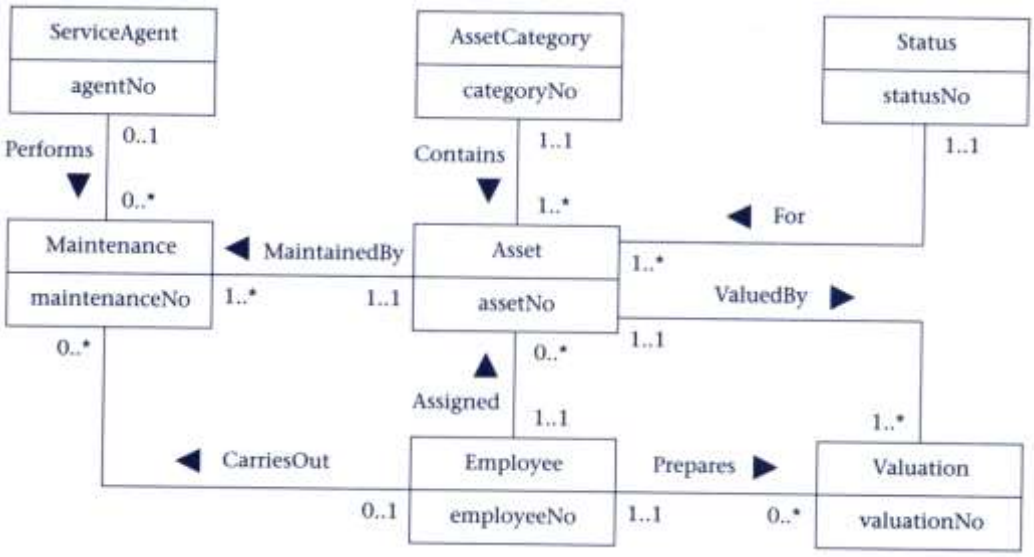
Employee	as defined in Section D.1.2
Product	(productNo, productName, serialNo, unitPrice, quantityOnHand, reorderLevel, reorderQuantity, reorderLeadTime, categoryNo) Primary Key productNo Alternate Key serialNo Foreign Key categoryNo references ProductCategory(categoryNo)
ProductCategory	(categoryNo, categoryDescription) Primary Key categoryNo
PurchaseOrder	(purchaseOrderNo, purchaseOrderDescription, orderDate, dateRequired, shippedDate, freightCharge, supplierNo, employeeNo) Primary Key purchaseOrderNo Foreign Key supplierNo references Supplier(supplierNo) Foreign Key employeeNo references Employee(employeeNo)
Supplier	(supplierNo, supplierName, supplierStreet, supplierCity, supplierState, supplierZipCode, suppTelNo, suppFaxNo, suppEmailAddress, suppWebAddress, contactName, contactTelNo, contactFaxNo, contactEmailAddress, paymentTerms) Primary Key supplierNo Alternate Key supplierName Alternate Key suppTelNo Alternate Key suppFaxNo
Transaction	(transactionNo, transactionDate, transactionDescription, unitPrice, unitsOrdered, unitsReceived, unitsSold, unitsWastage, productNo, purchaseOrderNo) Primary Key transactionNo Foreign Key productNo references Product(productNo) Foreign Key purchaseOrderNo references PurchaseOrder(purchaseOrderNo)

د-۳ مدیریت سرمایه

شرکتی میخواهد پایگاه داده ای را جهت نظارت بر هر یک از دارایی هایش(همچون PC ها، پرینترها، اتومبیلها، میزها، صندلیها) ایجاد کند. دارایی ها به چندین فهرست همچون کامپیوترها و اثاثیه تقسیم می شوند. دارایی به کارمند اختصاص داده شده است. بر اساس قاعده، کارمند بخش مالی هر دارایی را برای مشخص شدن ارزش کنونی بازار بررسی می کند، و تاریخ و ارزش کنون دارایی را ثبت می کند. از نتیجه ارزیابی شرکت می تواند درباره فروش دارایی تصمیم گیری کند. همچنین بر اساس قاعده، بر روی هر دارایی تعمیراتی انجام می گیرد. در بعضی موارد تعمیر بوسیله کارمند انجام می گیرد اما در دیگر موارد، دارایی جهت تعمیر به یک شرکت خارجی فرستاده می شود. مدل داده منطقی در شکل د-۵ و جداول مربوطه در شکل د-۶ نشان داده شده است.

شکل ۵ – د

مدل منطقی داده مدیریت دارایی .



شکل ۶- د

جداول مدیریت دارایی .

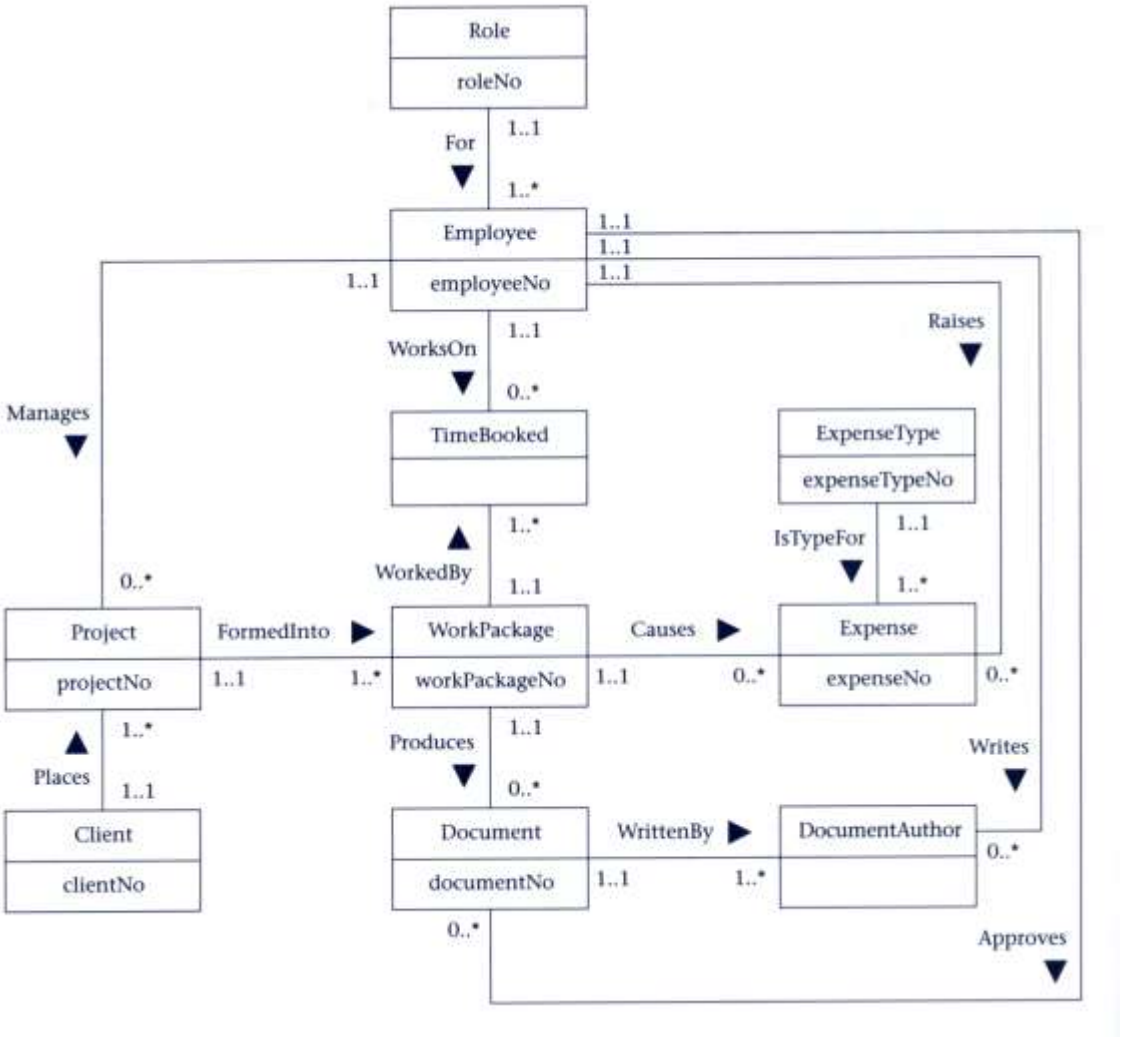
Employee	As defined in Section D.1.2
Asset	(assetNo, assetDescription, serialNo, dateAcquired, purchasePrice, currentValue, dateSold, nextMaintenanceDate, employeeNo, assetCategoryNo, statusNo) Primary Key assetNo Alternate Key serialNo Foreign Key employeeNo references Employee(employeeNo) Foreign Key assetCategoryNo references AssetCategory(assetCategoryNo) Foreign Key statusNo references Status(statusNo)
AssetCategory	(assetCategoryNo, assetCategoryDescription) Primary Key assetCategoryNo
Maintenance	(maintenanceNo, maintenanceDate, maintenanceDescription, maintenanceCost, assetNo, employeeNo, agentNo) Primary Key maintenanceNo Foreign Key assetNo references Asset(assetNo) Foreign Key employeeNo references Employee(employeeNo) Foreign Key agentNo references ServiceAgent(agentNo)
ServiceAgent	(agentNo, agentName, agentStreet, agentCity, agentState, agentZipCode, agentTelNo, agentFaxNo, agentEmailAddress, agentWebAddress, contactName, contactTelNo, contactFaxNo, contactEmailAddress) Primary Key agentNo Alternate Key agentName Alternate Key agentTelNo Alternate Key agentFaxNo
Status	(statusNo, statusDescription) Primary Key statusNo
Valuation	(valuationNo, valuationDate, valuationPrice, assetNo, employeeNo) Primary Key valuationNo Foreign Key assetNo references Asset(assetNo) Foreign Key employeeNo references Employee(employeeNo)

۴- د مدیریت پروژه

شرکت مشاوره ای میخواهد پایگاه داده ای را جهت مدیریت پروژه هایشان ایجاد کند. هر پروژه برای مشتری خاصی است و مدیر پروژه معینی دارد. پروژه به تعدادی بسته های کاری تقسیم شده و، کارمندان زمان و هزینه هر بسته کاری را می پردازند. هر کارمند نقش خاصی دارد که میزان هزینه برای مشتری را تعریف می کند. کارمند می تواند به طور اضافه کار، در چندین بسته کاری مرتبط با پروژه مشابه کار کند. بعلاوه، بیشتر، اما نه همه، بسته های کاری تعدادی مستندات مرتبط قابل تحویلی دارند که هر یک شاید توسط بیش از یک کارمند نوشته شده باشند. مدل داده منطقی در شکل د-۷ و جداول مربوطه در شکل د-۸ نشان داده شده است.

شکل ۷- د

مدل منطقی داده مدیریت پروژه .



شکل ۸- د

جداول مدیریت پروژه .

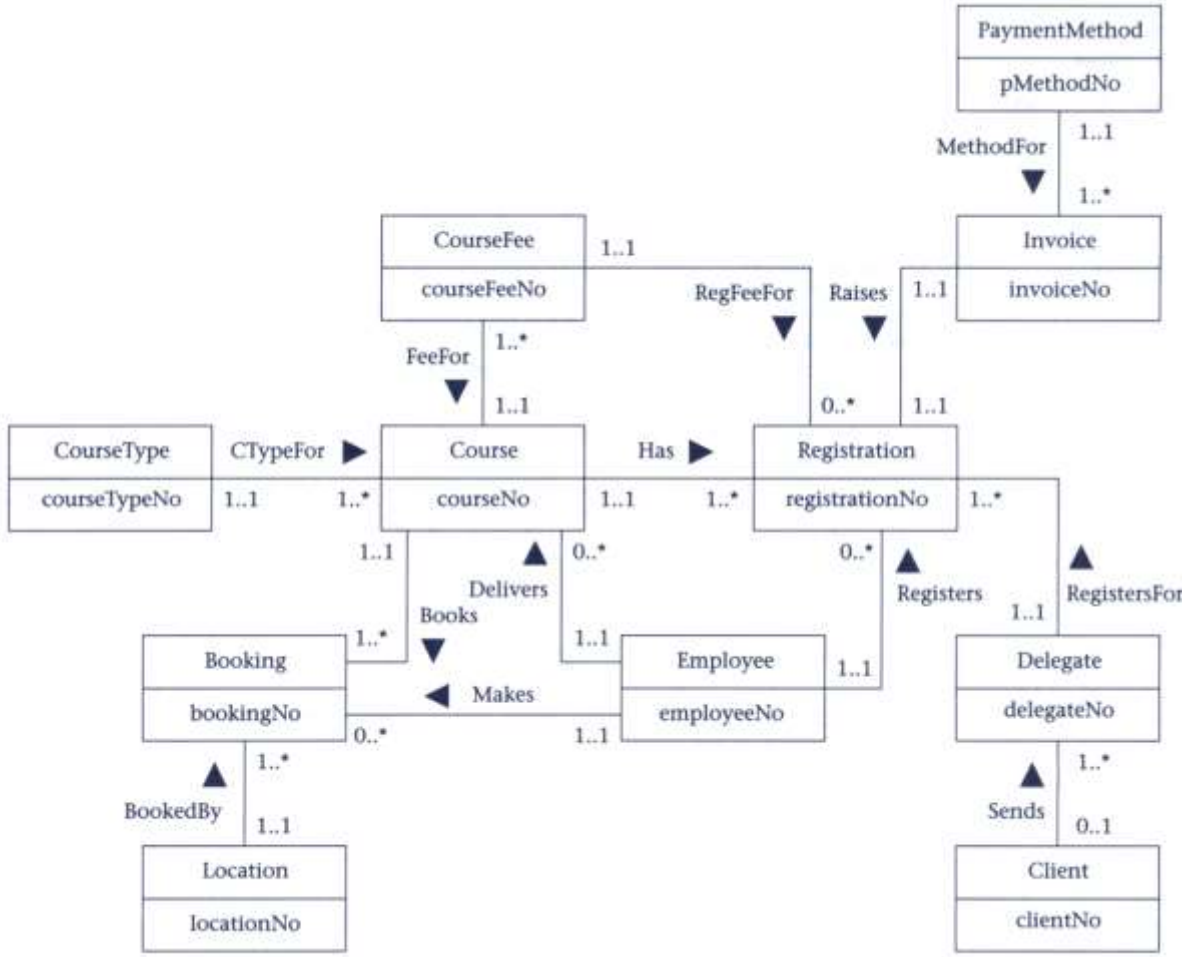
Client	(clientNo, clientName, clientStreet, clientCity, clientState, clientZipCode, clientTelNo, clientFaxNo, clientWebAddress, contactName, contactTelNo, contactFaxNo, contactEmailAddress) Primary Key clientNo Alternate Key clientName Alternate Key clientTelNo Alternate Key clientFaxNo
Document	(documentNo, documentTitle, documentDate, versionNo, workPackageNo, approvedByEmployeeNo) Primary Key documentNo Foreign Key workPackageNo references WorkPackage(workPackageNo) Foreign Key approvedByEmployeeNo references Employee(employeeNo)
DocumentAuthor	(documentNo, employeeNo) Primary Key documentNo, employeeNo Foreign Key documentNo references Document(documentNo) Foreign Key employeeNo references Employee(employeeNo)
Employee	(employeeNo, dateStartRole, firstName, middleName, lastName, address, workTelExt, homeTelNo, empEmailAddress, socialSecurityNumber, DOB, position, sex, salary, dateStarted, roleNo) Primary Key employeeNo Alternate Key socialSecurityNumber Foreign Key roleNo references Role(roleNo)
Expense	(expenseNo, expenseDate, expenseDescription, expenseAmount, workPackageNo, employeeNo, expenseTypeNo) Primary Key expenseNo Alternate Key workPackageNo, employeeNo, expenseDate Foreign Key workPackageNo references WorkPackage(workPackageNo) Foreign Key employeeNo references Employee(employeeNo) Foreign Key expenseTypeNo reference ExpenseType(expenseTypeNo)
ExpenseType	(expenseTypeNo, expenseTypeDescription) Primary Key expenseTypeNo
Project	(projectNo, projectName, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, projectedCost, actualCost, clientNo, managerEmployeeNo) Primary Key projectNo Foreign Key clientNo references Client(clientNo) Foreign Key managerEmployeeNo references Employee(employeeNo)
Role	(roleNo, roleDescription, billingRate) Primary Key roleNo
TimeBooked	(workPackageNo, employeeNo, dateStartWork, dateStopWork, timeWorked) Primary Key workPackageNo, employeeNo Foreign Key workPackageNo references WorkPackage(workPackageNo) Foreign Key employeeNo references Employee(employeeNo)
WorkPackage	(workPackageNo, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, projectedCost, actualCost, projectNo) Primary Key workPackageNo Foreign Key projectNo references Project(projectNo)

د-۵ مدیریت آموزش

شرکت آموزشی میخواهد پایگاه داده ای از اطلاعات آموزشی را ایجاد کند. شرکت چندین سمینار و دوره های آموزشی برگزار می کند. هر آموزش توسط یک عضو پرسنل در بعضی مناطق(همچون سمینار داخلی اتاق S10، هتل هیلتون سوییت ۱۰۰) ارائه می شود. شهریه هر آموزش متفاوت بوده و بر اساس تعداد نماینده هایی که می فرستد متغیر است. برای مثال اگر شرکت یک نفر را بفرستد، شاید شهریه ۱۰۰۰$ و اگر شرکت دو نفر را بفرستد، اولی ممکن است ۱۰۰۰ دلار اما دومی شاید ۷۵۰ دلار باشد. آموزش می تواند با حضور تعدادی از نماینده ها باشد که ناظر بر کیفیت آموزش باشد. نماینده می تواند بعنوان فرد یا از طریق شرکت او ثبت شود. نام کارمندی که نماینده را ثبت نام می کند نیز ثبت می شود. صورتحساب یا از طریق نماینده و یا از طریق فرد فرستاده می شود. مدل داده منطقی در شکل د-۹ و جداول مربوطه در شکل د-۱۰ نشان داده شده است.

شکل د-۹

مدل منطقی داده مدیریت آموزش .



شکل د-۱۰

جداول مدیریت آموزش .

Client	As defined in Section D.4.2
Employee	As defined in Section D.1.2
PaymentMethod	As defined in Section D.1.2
Delegate	(delegateNo, delegateTitle, delegateFName, delegateLName, delegateStreet, delegateCity, delegateState, delegateZipCode, attTelNo, attFaxNo, attEmailAddress, clientNo) Primary Key delegateNo Foreign Key clientNo references Client(clientNo)
Booking	(bookingNo, bookingDate, locationNo, courseNo, bookingEmployeeNo) Primary Key bookingNo Foreign Key locationNo references Location(locationNo) Foreign Key courseNo references Course(courseNo) Foreign Key bookingEmployeeNo references Employee(employeeNo)
Course	(courseNo, courseName, courseDescription, startDate, startTime, endDate, endTime, maxDelegates, confirmed, delivererEmployeeNo, courseTypeNo) Primary Key courseNo Foreign Key delivererEmployeeNo references Employee(employeeNo) Foreign Key courseTypeNo references CourseType(courseTypeNo)
CourseFee	(courseFeeNo, feeDescription, fee, courseNo) Primary Key courseFeeNo Foreign Key courseNo references Course(courseNo)
CourseType	(courseTypeNo, courseTypeDescription) Primary Key courseTypeNo
Invoice	(invoiceNo, dateRaised, datePaid, creditCardNo, holdersName, expiryDate, registrationNo, pMethodNo) Primary Key invoiceNo Foreign Key registrationNo references Registration(registrationNo) Foreign Key pMethodNo references PaymentMethod(pMethodNo)
Location	(locationNo, locationName, maxSize) Primary Key locationNo
Registration	(registrationNo, registrationDate, delegateNo, courseFeeNo, registerEmployeeNo, courseNo) Primary Key registrationNo Foreign Key delegateNo references Delegate(delegateNo) Foreign Key courseFeeNo references CourseFee(courseFeeNo) Foreign Key registerEmployeeNo references Employee(employeeNo) Foreign Key courseNo references Course(courseNo)

د-۶ مدیریت منابع انسانی

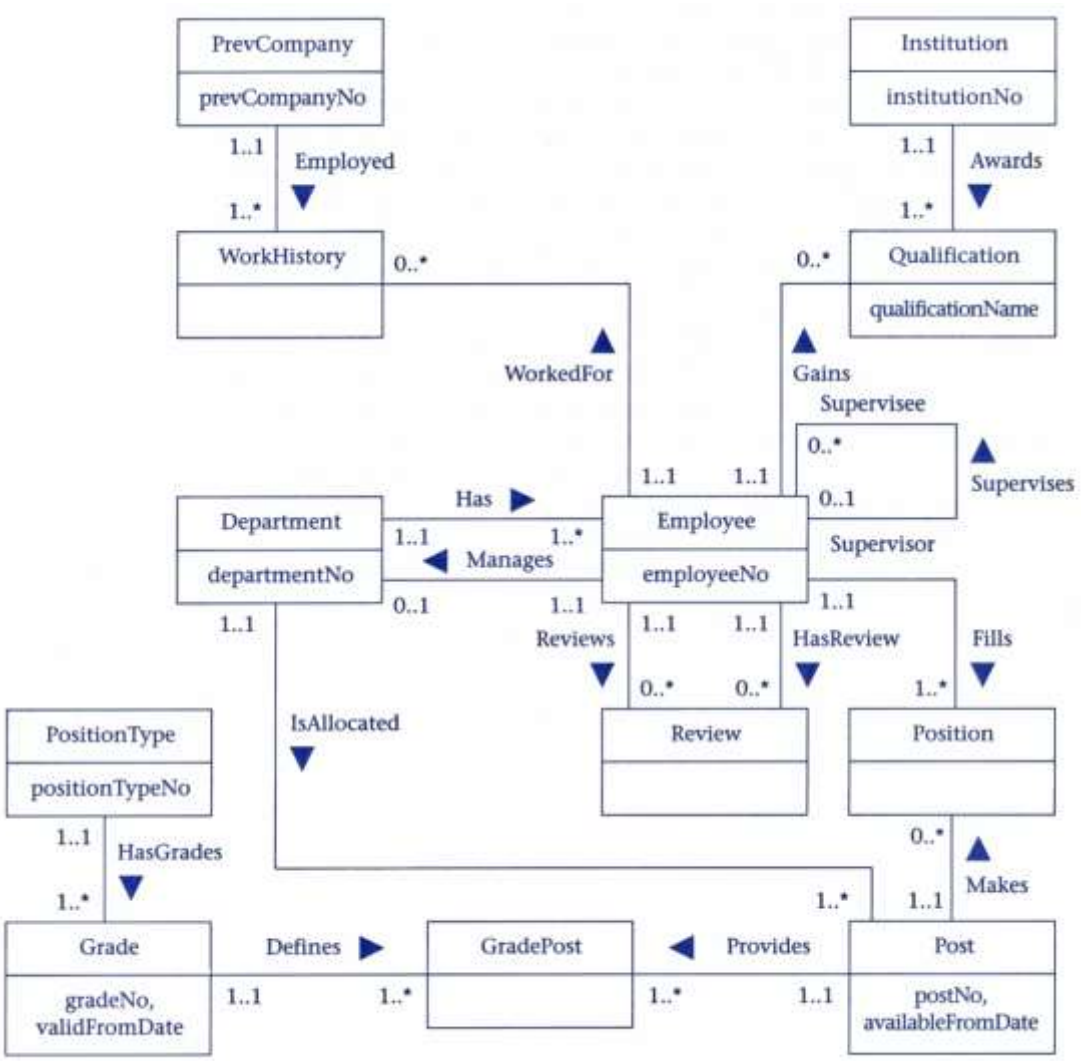
بخش HRM(Human Resource Management) میخواهد پایگاه داده ای را جهت نظارت بر کارمندانش ایجاد کند. شرکت به تعدادی از بخشها تقسیم شده است و کارمندان به یک بخش تخصیص داده شده اند. بخش فوق، مدیر طراحی دارد که مسولیت کلی بخش و کارمندان موجود در بخش بر عهده او می باشد. با این وجود، برای کمک به مدیریت بخش تعدادی از کارمندان برای کمک به گروههای پرسنلی کاندید شده اند. وقتیکه کارمند جدیدی به شرکت می پیوندد اطلاعات سابقه قبلی او و صلاحیت او مورد نیاز است. بر اساس قاعده، هر کارمند نیازمند نظارت دارد که معمولا توسط مدیر انجام می شود اما شاید ممکن است توسط نماینده صورت گیرد.

شرکت تعدادی از انواع شغلها را تعریف کرده است: همچون مدیر، تحلیلگر تجارت، فروشنده، منشی و هر نوع تعدادی درجه مرتبط با آن را دارد که برای بیشتر شغلهای غیر مهم حقوق کارمندان را مشخص می کند. در سطح ارشد حقوق قابل مذاکره است. شغلها به بخش بر اساس حجم کاری شان اختصاص داده می شوند. برای مثال، به بخش شاید دو تحلیلگر تجارت جدید اختصاص داده شده فرستاده شود. شغل شاید توسط یک کارمند پر شود اگر چه چندین بار، تعدادی از پستهای مختلف را پر خواهند کرد.

مدل داده منطقی در شکل د-۱۱ و جداول مربوطه در شکل د-۱۲ نشان داده شده است.

شکل د-۱۱

مدل منطقی داده مدیریت منابع انسانی .



شکل ۱۲- د

جداول مدیریت منابع انسانی .

Department	(departmentNo, departmentName, deptLocation, managerEmployeeNo) Primary Key departmentNo Foreign Key managerEmployeeNo references Employee(employeeNo)
Employee	(employeeNo, title, firstName, middleName, lastName, address, workTelExt, homeTelNo, empEmailAddress, socialSecurityNumber, DOB, position, sex, salary, dateStarted, dateLeft, departmentNo, supervisorEmployeeNo) Primary Key employeeNo Alternate Key socialSecurityNumber Foreign Key departmentNo references Department(departmentNo) Foreign Key supervisorEmployeeNo references Employee(employeeNo)
Grade	(gradeNo, validFromDate, validToDate, gradeDescription, gradeSalary, noDaysLeaveEntitlement, positionTypeNo) Primary Key gradeNo, validFromDate Foreign Key positionTypeNo references PositionType(positionTypeNo)
GradePost	(gradeNo, validFromDate, postNo, availableFromDate) Primary Key gradeNo, validFromDate, postNo, availableFromDate Foreign Key gradeNo, validFromDate references Grade(gradeNo, validFromDate) Foreign Key postNo, availableFromDate references Post(postNo, availableFromDate)
Institution	(institutionNo, institutionName, instAddress, instTelNo, instFaxNo, instWebAddress, contactName, contactTelNo, contactFaxNo, contactEmailAddress) Primary Key institutionNo Alternate Key institutionName Alternate Key instTelNo Alternate Key instFaxNo
Position	(employeeNo, postNo, startDate, endDate) Primary Key employeeNo, postNo, startDate Foreign Key employeeNo references Employee(employeeNo) Foreign Key postNo, startDate references Post(postNo, availableFromDate)
PositionType	(positionTypeNo, positionTypeDescription) Primary Key positionTypeNo
Post	(postNo, availableFromDate, availableToDate, postDescription, salariedHourly, fullPartTime, temporaryPermanent, freeLaborStandardsActExempt, departmentNo) Primary Key postNo, availableFromDate Foreign Key departmentNo references Department(departmentNo)
PrevCompany	(prevCompanyNo, pCompanyName, pCompanyStreet, pCompanyCity, pCompanyState, pCompanyZipCode, pCompanyTelNo, pCompanyFaxNo, pCompanyWebAddress, contactName, contactTelNo, contactFaxNo, contactEmailAddress) Primary Key prevCompanyNo Alternate Key pCompanyName Alternate Key pCompanyTelNo Alternate Key pCompanyFaxNo
Qualification	(qualificationName, employeeNo, gradeObtained, startQualDate, endQualDate, gpa, institutionNo) Primary Key qualificationName, employeeNo Foreign Key employeeNo references Employee(employeeNo) Foreign Key institutionNo references Institution(institutionNo)

شکل ۱۲- د

ادامه . . .

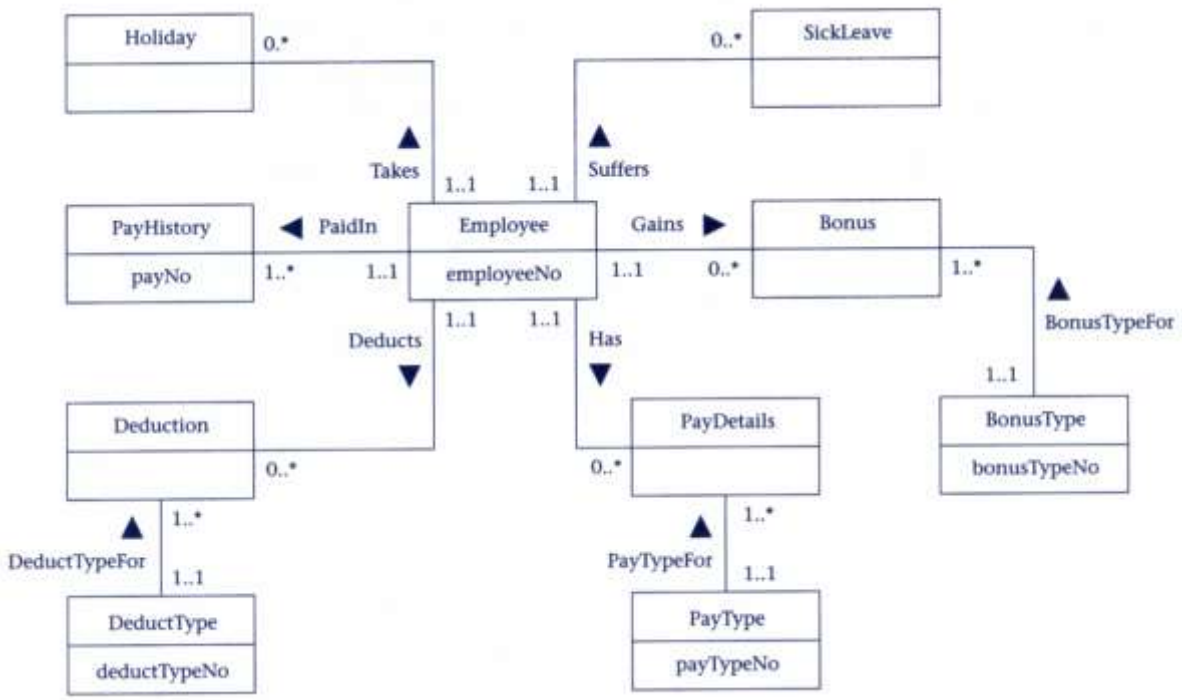
Review	(revieweeEmployeeNo, reviewerEmployeeNo, reviewDate, comments) Primary Key revieweeEmployeeNo, reviewerEmployeeNo, reviewDate Foreign Key revieweeEmployeeNo references Employee(employeeNo) Foreign Key reviewerEmployeeNo references Employee(employeeNo)
WorkHistory	(prevCompanyNo, employeeNo, prevPosition, prevGrade, prevSalary, prevLocation, prevResponsibilities) Primary Key prevCompanyNo, employeeNo Foreign Key prevCompanyNo references PrevCompany(prevCompanyNo) Foreign Key employeeNo references Employee(employeeNo)

د-۷ مدیریت حقوق

بخش حقوق میخواهد پایگاه داده ای را جهت نظارت بر پرداخت حقوق کارمندان ایجاد کند. جهت محاسبه حقوق کارمندان، لیست حقوق نیاز است که در آن میزان کار کارکنان، مجموع تعطیلی، اضافه کاری، پاداش در آن در نظر گرفته شود. کارمند باید چگونگی پرداخت حقوقش را مشخص کند. اگرچه ممکن است بارها تغییر پیدا کند. حقوق بیشتر کارمندان توسط سند بانکی الکترونیک پرداخته می شود اما بعضی دیگر ممکن است بوسیله پول نقد یا چک پرداخته شود. اگر پرداخت الکترونیکی باشد در اینصورت شماره مسیریابی و نوع حساب موردنیاز است. پرداخت می تواند تنها بوسیله یک روش انجام گیرد. چندین دلیل مختلف برای کسری ممکن است وجود داشته باشد . برای مثال مالیات حکومتی، مالیات ناحیه ای، برنامه دارویی، برنامه بازنشستگی، یا پوی پیش دریافت. مدل داده منطقی در شکل د-۱۳ و جداول مربوطه در شکل د-۱۴ نشان داده شده است.

شکل ۱۳- د

مدل منطقی داده مدیریت حقوق .



شکل ۱۴- د

جداول مدیریت حقوق .

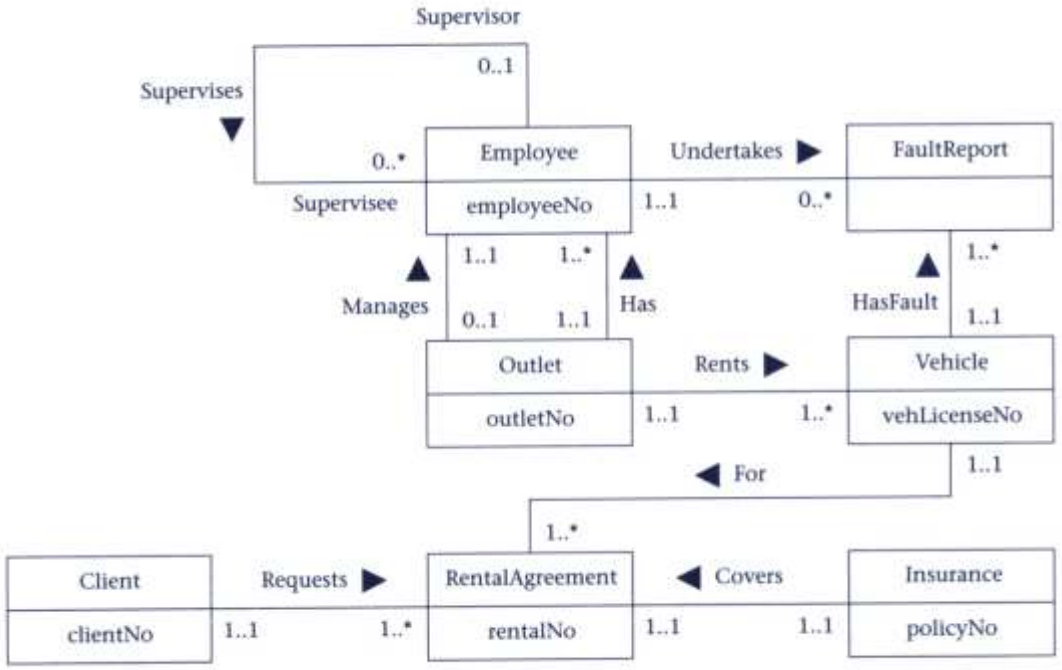
Employee	As defined in Section D.1.2
Bonus	(employeeNo, bonusDate, bonusAmount, bonusTypeNo) Primary Key employeeNo, bonusDate Foreign Key employeeNo references Employee(employeeNo) Foreign Key bonusTypeNo references BonusType(bonusTypeNo)
BonusType	(bonusTypeNo, bonusDescription) Primary Key bonusTypeNo
Deduction	(employeeNo, deductDate, deductAmount, deductTypeNo) Primary Key employeeNo, deductDate Foreign Key employeeNo references Employee(employeeNo) Foreign Key deductTypeNo references DeductType(deductTypeNo)
DeductType	(deductTypeNo, deductDescription) Primary Key deductTypeNo
Holiday	(employeeNo, startDate, endDate) Primary Key employeeNo, startDate Foreign Key employeeNo references Employee(employeeNo)
PayDetails	(employeeNo, startDate, routingNumber, accountType, bankName, bankAddress, payTypeNo) Primary Key employeeNo, startDate Foreign Key employeeNo references Employee(employeeNo) Foreign Key payTypeNo references PayType(payTypeNo)
PayHistory	(payNo, employeeNo, payDate, checkNumber, payAmount) Primary Key payNo Foreign Key employeeNo references Employee(employeeNo)
PayType	(payTypeNo, payTypeDescription) Primary Key payTypeNo
SickLeave	(employeeNo, startDate, endDate, reason) Primary Key employeeNo, startDate Foreign Key employeeNo references Employee(employeeNo)

۸- د کرایه کردن اتومبیل

شرکت کرایه اتومبیل میخواهد پایگاه داده ای را جهت نظارت بر کرایه اتومبیل به مشتریان ایجاد کند. شرکت دارای چندین فروشگاه و هر فروشگاه پرسنلی دارد که شامل مدیر و چندین مکانیک ارشد می باشد که مسئول نظارت بر کارگروههای مکانیکی تخصیص داده شده است. هر فروشگاه انباری از وسایل نقلیه جهت کرایه دارد که شاید توسط مشتریان برای دوره های مختلفی از زمانها از حداقل ۴ ساعت تا حداکثر شش ماه کرایه داده شود. هر توافقنامه کرایه بین مشتری و شرکت بطور منحصر بفرد با استفاده از شماره کرایه مشخص می شود. مشتری باید پوشش بیمه برای هر دوره کرایه وسیله نقلیه داشته باشد. هر وسیله نقلیه جهت تعمیرات بعد از هر کرایه بررسی می شود. مدل داده منطقی در شکل د-۱۵ و جداول مربوطه در شکل د-۱۶ نشان داده شده است.

شکل د-۱۵

مدل منطقی داده کرایه اتومبیل .



شکل د-۱۶

جداول کرایه اتومبیل .

Client	As defined in Section D.4.2
Employee	(employeeNo, title, firstName, middleName, lastName, address, workTelExt, homeTelNo, empEmailAddress, socialSecurityNumber, DOB, position, sex, salary, dateStarted, outletNo) Primary Key employeeNo Alternate Key socialSecurityNumber Foreign Key outletNo references Outlet(outletNo)
FaultReport	(vehLicenseNo, dateChecked, timeChecked, comments, employeeNo) Primary Key vehLicenseNo, dateChecked Foreign Key vehLicenseNo references Vehicle(vehLicenseNo) Foreign Key employeeNo references Employee(employeeNo)
Outlet	(outletNo, outletStreet, outletCity, outletState, outletZipCode, outletTelNo, outletFaxNo, managerEmployeeNo) Primary Key outletNo Alternate Key outletTelNo Alternate Key outletFaxNo Foreign Key managerEmployeeNo references Employee(employeeNo)
RentalAgreement	(rentalNo, dateStart, timeStart, dateReturn, timeReturn, mileageBefore, mileageAfter, policyNo, insuranceCoverType, insurancePremium, clientNo, vehLicenseNo) Primary Key rentalNo Alternate Key policyNo Foreign Key clientNo references Client(clientNo) Foreign Key vehLicenseNo references Vehicle(vehLicenseNo)
Vehicle	(vehLicenseNo, vehicleMake, vehicleModel, color, noDoors, capacity, hireRate, outletNo) Primary Key vehLicenseNo Foreign Key outletNo references Outlet(outletNo)

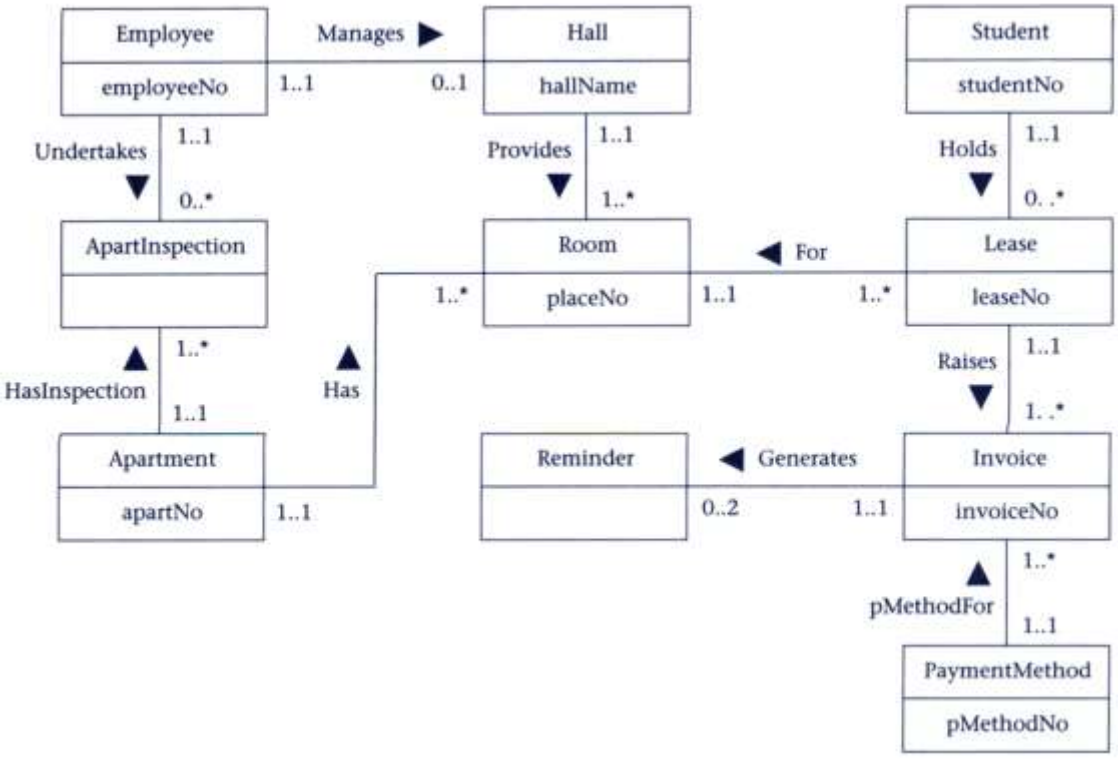
د-۹ اسکان دانشجو

دفتر اسکان دانشگاه میخواهد پایگاه داده ای را جهت نظارت بر تخصیص منزل به دانشجویان ایجاد کند. هر دانشجویی که نیازمند اتاق است بایستی فرم درخواست را پرکند، که جزئیات دانشجو و مدت زمان و نوع اسکان را نگه می دارد. دانشجویان ممکن است اتاق موجود در راهرو محل اقامت یا آپارتمان دانشجویی را کرایه کنند. راهروها تنها شامل اتاقهای واحد می باشد، که دارای شماره اتاق، شماره مکان ، و نرخ هزینه ماهانه می باشند. شماره مکان به طور منحصر هر اتاق موجود در همه راهرو کنترل شده توسط دفتر اسکان را مشخص می کند و هنگامی که دانشجو اتاق را کرایه می کند مورد استفاده قرار می گیرد. هر راهرو توسط عضوی از دفتر اسکان مدیریت می شود.

دفتر اسکان همچنین آپارتمانهای دانشجویی پیشنهاد می کند،که هریک بوسیله شماره آپارتمان منحصربفرد مشخص می شود. این آپارتمانها کاملا مجهز بوده و منزل تک اتاقی را برای گروههای سه، چهار، پنج تفره دانشجویان فراهم می کند. هر اتاق خواب آپارتمان دارای هزینه اجاره ماهانه، شماره اتاق و شماره مکان می باشد. شماره مکان به طور منحصر هر اتاق موجود در همه آپارتمانهای دانشجویی را مشخص می کند و زمانیکه اتاق به دانشجو اجاره داده می شود مورد استفاده قرار می گیرد. آپارتمانها توسط اعضای دفتر اسکان براساس قاعده، مورد بررسی قرار می گیرند تا اطمینان حاصل شود که اتاق به خوبی نگهداشته می شود. درباره توافقنامه اجاره جدید در شروع هر سال تحصیلی با دوره اجاره حداقل یک ترم و حداکثر دوره یک سال مذاکره می شود. دانشجویان هزینه اسکان خود را در سراسر سال تحصیلی می پردازند و به آنها صورتحسابی در شروع هر ترم فرستاده می شود. اگر دانشجو هزینه را در تاریخ مشخص پرداخت نکند دو نامه تذکر فرستاده می شود. مدل داده منطقی در شکل د-۱۷ و جداول مربوطه در شکل د-۱۸ نشان داده شده است.

شکل ۱۷- د

مدل منطقی داده اسکان دانشجو .



شکل د-۱۸

جداول اسکان دانشجو .

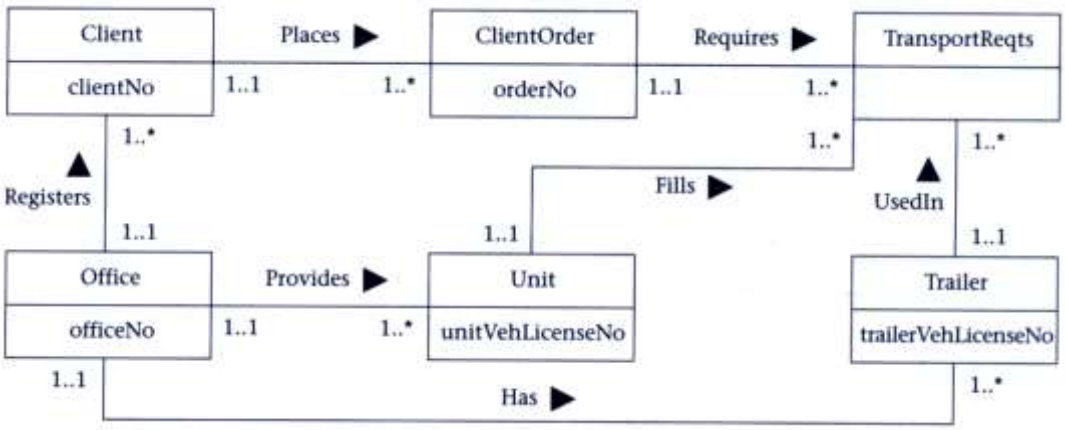
Employee	As defined in Section D.1.2
PaymentMethod	As defined in Section D.1.2
Apartment	(apartNo, apartAddress, noOfRoomsInApart) Primary Key apartNo
ApartInspection	(apartNo, dateOfInspection, comments, status, employeeNo) Primary Key apartNo, dateOfInspection Foreign Key apartNo references Apartment(apartNo) Foreign Key employeeNo references Employee(employeeNo)
Hall	(hallName, hallAddress, hallTelNo, hallFaxNo, noOfRoomsInHall, managerEmployeeNo) Primary Key hallName Alternate Key hallTelNo Alternate Key hallFaxNo Foreign Key managerEmployeeNo references Employee(employeeNo)
Invoice	(invoiceNo, semester, dateDue, datePaid, leaseNo, pMethodNo) Primary Key invoiceNo Foreign Key leaseNo references Lease(leaseNo) Foreign Key pMethodNo references PaymentMethod(pMethodNo)
Reminder	(invoiceNo, dateReminder1sent, dateReminder2sent, dateInterview, comments) Primary Key invoiceNo Foreign Key invoiceNo references Invoice(invoiceNo)
Lease	(leaseNo, duration, dateStart, dateLeave, studentNo, placeNo) Primary Key leaseNo Alternate Key placeNo, dateStart Alternate Key studentNo, dateStart Foreign Key studentNo references Student(studentNo) Foreign Key placeNo references Room(placeNo)
Room	(placeNo, roomNo, rentPerSemester, hallName, apartNo) Primary Key placeNo Alternate Key roomNo, hallName Alternate Key roomNo, apartNo Foreign Key hallName references Hall(hallName) Foreign Key apartNo references Apartment(apartNo)
Student	(studentNo, studentFirstName, studentMiddleIntial, studentLastName, studentHomeStreet, studentHomeCity, studentHomeState, studentHomeZipCode, studentHomeTelNo, studentSex, studentDOB, studentType, studentStatus, accommodationTypeRequired, accommodationDuration) Primary Key studentNo

د-۱۰ حمل و نقل مشتری

شرکت حمل ونقل که متخصص در حمل ونقل بار است میخواهد پایگاه داده ای را جهت کنترل سفارشات مشتری درباره حمل و نقل ایجاد کند. شرکت چندین دفتر جهت پردازش سفارشات مشتری دارد. مشتری از طریق دفتر ثبت نام کرده و می تواند یک یا چند سفارش بدهد. هر سفارش باری را که باید منتقل شود را توصیف می کند که دارای آدرس خود و آدرس محل تحویل می باشد. سپس نیازمندیهای حمل ونقل هر سفارش محاسبه می شود. نیازمندیهای حمل و نقل تعداد واحدها و یدک کشهایی که جهت بار لازم است را توصیف می کند. هر دفتر چندین واحد و یدک کش دارد. یک واحد می تواند یک یا دو یدک کش را بکشد. مدل داده منطقی در شکل د-۱۹ و جداول مربوطه در شکل د-۲۰ نشان داده شده است.

شکل د-۱۹

مدل منطقی داده حمل و نقل مشتری .



شکل د-۲۰

جداول حمل و نقل مشتری .

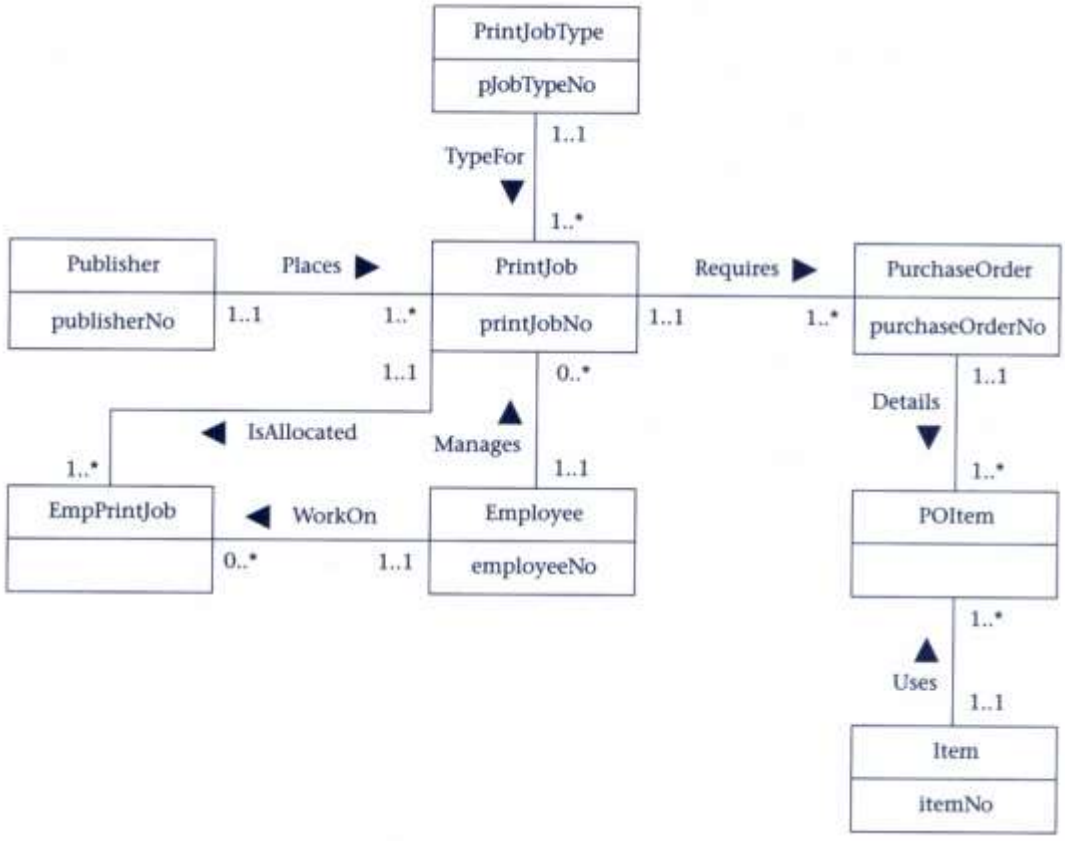
Client	(clientNo, clientName, clientStreet, clientCity, clientState, clientZipCode, clientTelNo, clientFaxNo, clientWebAddress, contactName, contactTelNo, contactFaxNo, contactEmailAddress, officeNo) Primary Key clientNo Alternate Key clientTelNo Alternate Key clientFaxNo Foreign Key officeNo references Office(officeNo)
Office	(officeNo, officeAddress, officeTelNo, officeFaxNo) Primary Key officeNo Alternate Key officeTelNo Alternate Key officeFaxNo
ClientOrder	(orderNo, dateOrder, collectionDate, collectionAddress, deliveryDate, deliveryAddress, loadWeight, loadDescription, clientNo) Primary Key orderNo Foreign Key clientNo references Client(clientNo)
Trailer	(trailerVehLicenseNo, trailerDescription, trailerLength, maxCarryingWeight, officeNo) Primary Key trailerVehLicenseNo Foreign Key officeNo references Office(officeNo)
TransportReqts	(orderNo, transportReqPartNo, unitVehLicenseNo, trailerVehLicenseNo1, trailerVehLicenseNo2) Primary Key orderNo, transportReqPartNo Foreign Key unitVehLicenseNo references Unit(unitVehLicenseNo) Foreign Key trailerVehLicenseNo1 references Trailer(trailerVehLicenseNo) Foreign Key trailerVehLicenseNo2 references Trailer(trailerVehLicenseNo)
Unit	(unitVehLicenseNo, unitDescription, maxPayLoad, officeNo) Primary Key unitVehLicenseNo Foreign Key officeNo references Office(officeNo)

د-۱۱ ناشر چاپ

شرکت چاپی که کارهای چاپی ناشران را انجام می دهد می خواهد پایگاه داده ای را جهت کنترل درخواستهای مشتری برای چاپ ایجاد کند. ناشر کتاب سفارش خود را که کارهای چاپی را توصیف می کند اعلام می دارد. کار چاپی نیازمند استفاده از موادی همچون، مرکب و کاغذ می باشد، که به کار از طریق یک یا چند سفارش خرید تخصیص داده می شود. به هرکار چاپی یک مدیر چاپ اختصاص داده شده است که مسئولیت اطمینان از اینکه کار چاپ به طور صحیح انجام گرفته است را دارا می باشد. برای کارهای چاپی بزرگتر کارمندان دیگری معمولا جهت کمک به کار چاپ اختصاص داده می شوند. مدل داده منطقی در شکل د-۲۱ و جداول مربوطه در شکل د-۲۲ نشان داده شده است.

شکل د-۲۱

مدل منطقی داده ناشر چاپ .



شکل د-۲۲

جداول ناشر چاپ .

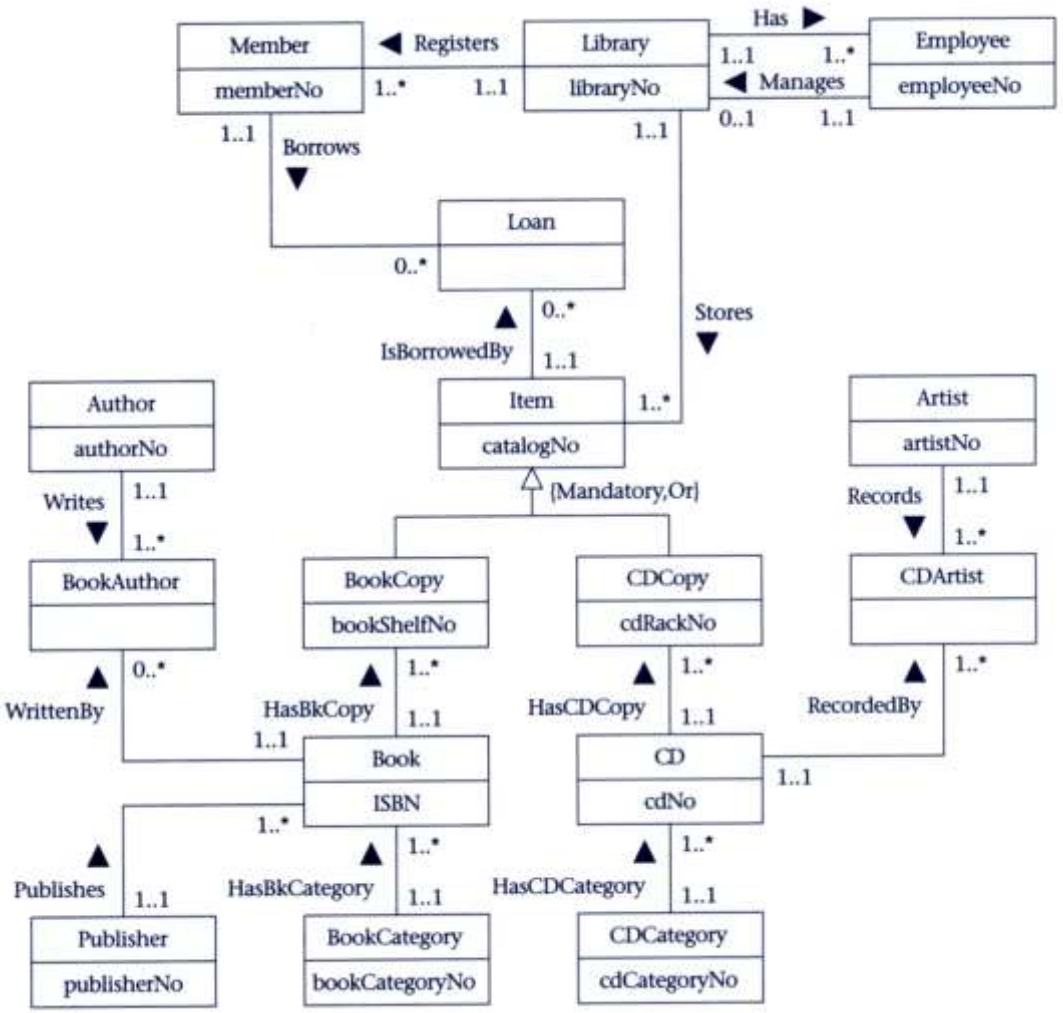
Employee	As defined in Section D.1.2
EmpPrintJob	(employeeNo, printJobNo, jobDate) Primary Key employeeNo, printJobNo Foreign Key employeeNo references Employee(employeeNo) Foreign Key printJobNo references PrintJob(printJobNo)
Item	(itemNo, itemDescription, itemPrice, itemQuantityInStock, itemReorderLevel, itemReorderQuantity, itemReorderLeadTime) Primary Key itemNo
PrintJob	(printJobNo, printJobDescription, printJobDateReceived, printJobDateCompleted, managerEmployeeNo, publisherNo, printJobTypeNo) Primary Key printJobNo Foreign Key managerEmployeeNo references Employee(employeeNo) Foreign Key publisherNo references Publisher(publisherNo) Foreign Key printJobTypeNo references PrintJobType(printJobTypeNo)
Publisher	(publisherNo, publisherName, publisherStreet, publisherCity, publisherState, publisherZipCode, pubTelNo, pubFaxNo, pubWebAddress, contactName, contactTelNo, contactFaxNo, contactEmailAddress, creditRating) Primary Key publisherNo Alternate Key publisherName Alternate Key pubTelNo Alternate Key pubFaxNo
POItem	(purchaseOrderNo, itemNo, quantity) Primary Key purchaseOrderNo, itemNo Foreign Key purchaseOrderNo references PurchaseOrder (purchaseOrderNo) Foreign Key itemNo references Item(itemNo)
PrintJobType	(printJobTypeNo, printJobTypeDescription) Primary key printJobTypeNo
PurchaseOrder	(purchaseOrderNo, printJobNo, purchaseOrderDate) Primary Key purchaseOrderNo Foreign Key printJobNo references PrintJob(printJobNo)

۱۲ – د کتابخانه محلی

منطقه ای میخواهد پایگاه داده ای را جهت کنترل کتابخانه های محلی اش ایجاد کند. هرکتابخانه تعدادی کارمند دارد، یکی از آنها برای مدیریت کتابخانه گماشته شده اند و مسئول نظارت بر کارمندان و مدیریت کلی روز به روز کتابخانه را بر عهده دارند. هرکتابخانه چندین کتاب و سی دی را نگه داری می کند. شهروند قبل از اینکه بتواند کتابی را قرض بگیرد باید عضو کتابخانه شود، اما بعد از آن می تواند از هر کتابخانه منطقه کتاب قرض کند. کتابها در قفسه ها نگهداری شده و سی دی ها نیز در تعدادی قفسه های مجزایی در مرکز کتابخانه نگهداری می شوند. عموما، کتابخانه تعدادی از کپی های عنوان کتاب و سی دی ها را ذخیره می کند. جزئیات ناشران کتاب نگداری می شوند اما ناشران سی دی ها نگهداری نمی شوند. برای پیداکردن یک قلم از سی دی یا کتاب جستجو می تواند براساس عنوان کتاب/ سی دی ، نام مولف/ هنرپیشه، فهرست کتاب/ سی دی ، یا نام ناشر انجام گیرد. مدل داده منطقی در شکل د-۲۳ و جداول مربوطه در شکل د-۲۴ نشان داده شده است.

شکل د-۲۳

مدل منطقی داده کتابخانه منطقه ای .



شکل د-۲۴

جداول کتابخانه محلی .

Publisher	As defined in Section D.11.2
Artist	(artistNo, name) Primary Key artistNo
Author	(authorNo, name) Primary Key authorNo
Book	(ISBN, title, year, publisherNo, bookCategoryNo) Primary Key ISBN Foreign Key publisherNo references Publisher(publisherNo) Foreign Key bookCategoryNo references BookCategory(bookCategoryNo)
BookAuthor	(ISBN, authorNo) Primary Key ISBN, authorNo Foreign Key ISBN references Book(ISBN) Foreign Key authorNo references Author(authorNo)
BookCategory	(bookCategoryNo, bookCatDescription) Primary Key bookCategoryNo
BookCopy	(catalogNo, bookShelfNo, ISBN, dateInStock, libraryNo) Primary Key catalogNo Alternate Key bookShelfNo Foreign Key ISBN references Book(ISBN) Foreign Key libraryNo references Library(libraryNo)
CD	(cdNo, title, releaseDate, cdCategoryNo) Primary Key cdNo Foreign Key cdCategoryNo references CDCategory(cdCategoryNo)
CDArtist	(cdNo, artistNo) Primary Key cdNo, artistNo Foreign Key cdNo references CD(cdNo) Foreign Key artistNo references Artist(artistNo)
CDCategory	(cdCategoryNo, cdCatDescription) Primary Key cdCategoryNo
CDCopy	(catalogNo, cdRackNo, cdNo, dateInStock, libraryNo) Primary Key catalogNo Alternate Key cdRackNo Foreign Key cdNo references CD(cdNo) Foreign Key libraryNo references Library(libraryNo)
Employee	(employeeNo, title, firstName, middleName, lastName, address, workTelExt, homeTelNo, empEmailAddress, socialSecurityNumber, DOB, position, sex, salary, dateStarted, libraryNo) Primary Key employeeNo Alternate Key socialSecurityNumber Foreign Key libraryNo references Library(libraryNo)
Library	(libraryNo, libStreet, libCity, libState, libZipCode, libTelNo, libFaxNo, libWebAddress, managerEmployeeNo) Primary Key libraryNo Alternate Key libTelNo Alternate Key libFaxNo

شکل د-۲۴

ادامه . . .

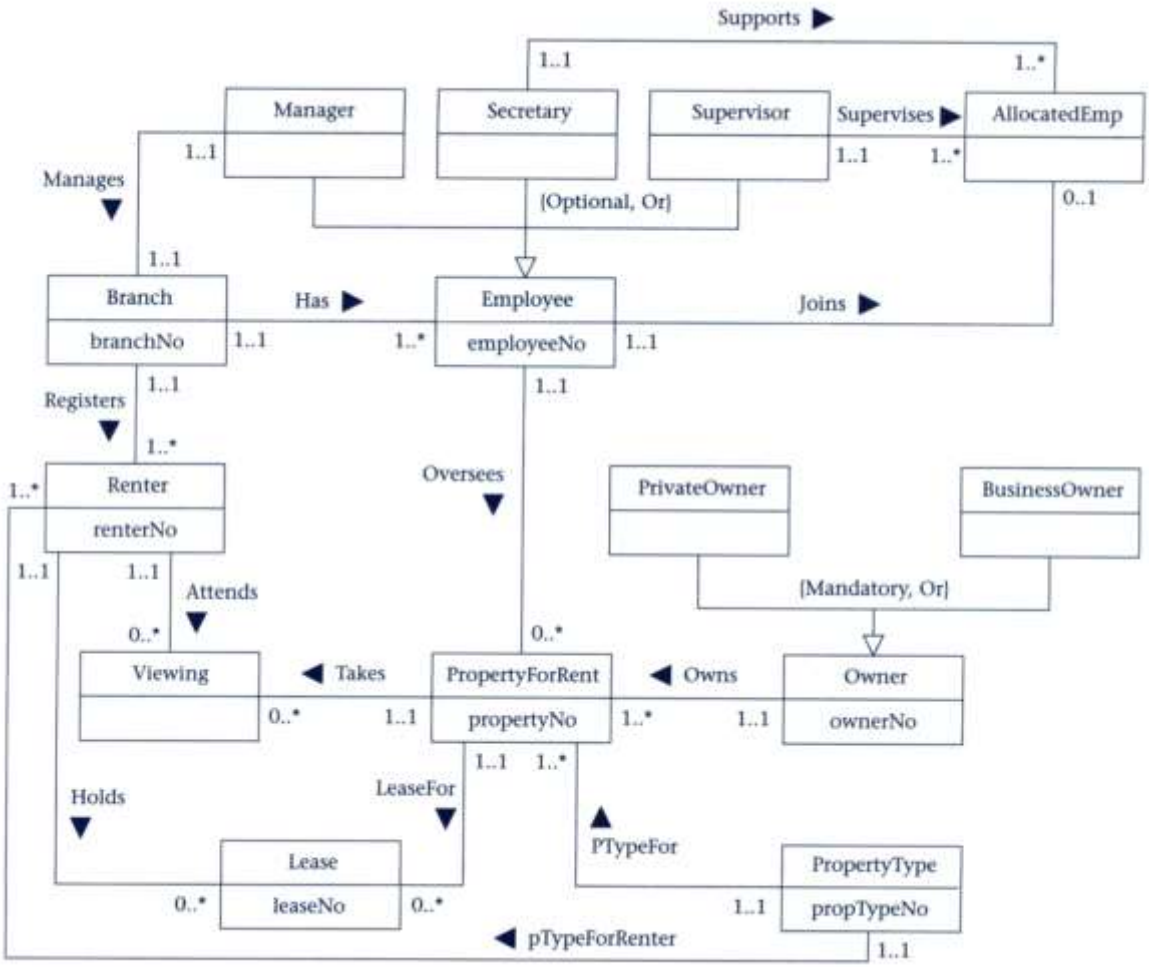
	Foreign Key managerEmployeeNo references Employee(employeeNo)
Loan	(catalogNo, memberNo, dateOut, dateReturn) Primary Key catalogNo, memberNo Foreign Key catalogNo references BookCopy(catalogNo) and CDCopy(catalogNo) Foreign Key memberNo references Member(memberNo)
Member	(memberNo, memTitle, memFirstName, memMiddleName, memLastName, memAddress, memWorkTelExt, memHomeTelNo, memDOB, memSex, dateJoined, libraryNo) Primary Key memberNo Foreign Key libraryNo references Library(libraryNo)

د-۱۳ اجاره مسکن

آژانس مسکن با شعبه های خود میخواهد پایگاه داده ای را جهت کنترل دارایی هایی که به نیمی از مالکان کرایه داده است را ایجاد کند که آنها به مالکان خصوصی و تجاری تقسیم بندی می شوند. درهر شعبه، پرسنل به دارایی های اجاره داده شده سرکشی کرده و مسئول قضاوت بردارایی ها و توافق نامه های ملکی هستند. بعضی پرسنل همچنین نقش ناظر دارند که مسئول سرکشی به گروههای پرسنل و اطمینان از مدیریت موثر شعبه را بر عهده دارند. کار مدیریت هرگروه پرسنل توسط منشی پشتیبانی می شود. مدل داده منطقی در شکل د-۲۵ و جداول مربوطه در شکل د-۲۶ نشان داده شده است.

شکل د-۲۵

مدل منطقی داده اجاره مسکن .



شکل ۲۶- د

جداول اجاره مسکن .

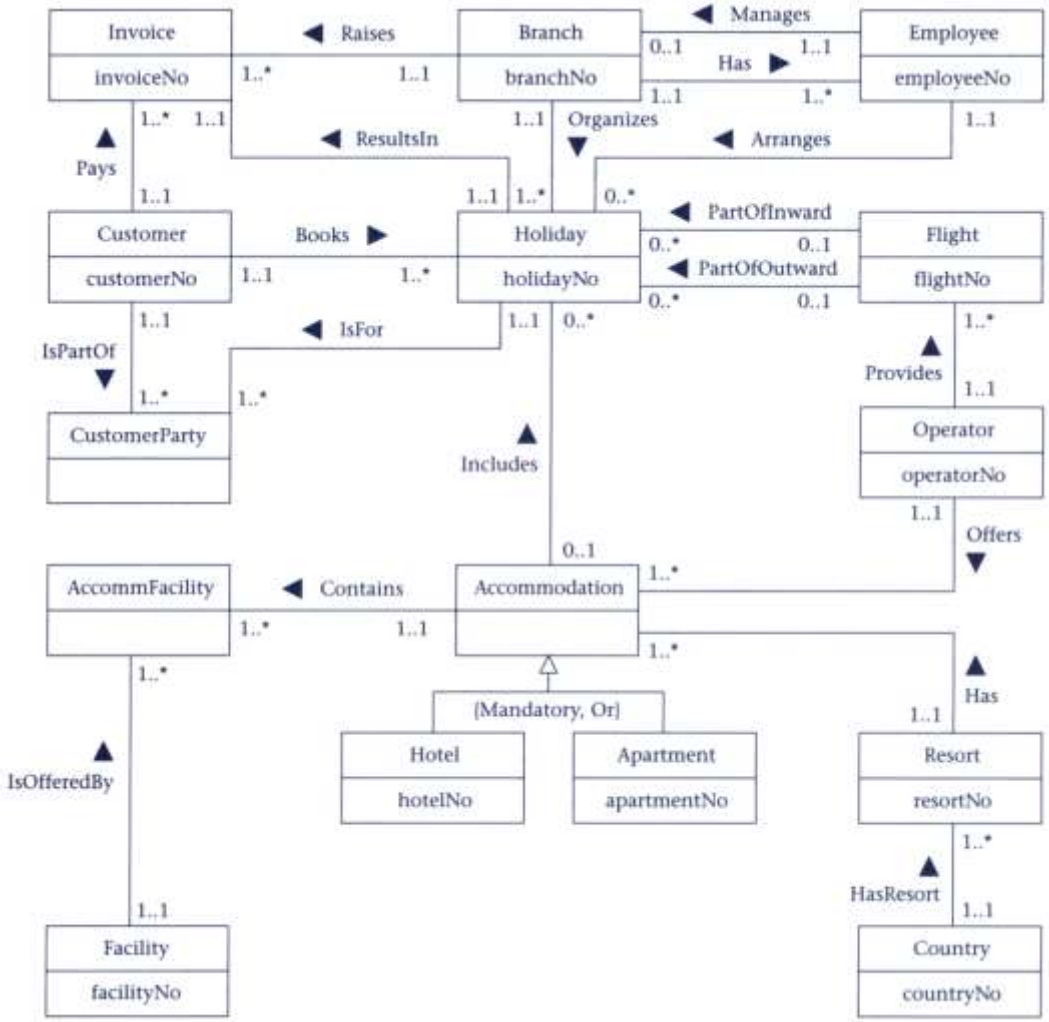
AllocatedEmp	(superviseeEmployeeNo, supervisorEmployeeNo, secretaryEmployeeNo) Primary Key superviseeEmployeeNo Foreign Key superviseeEmployeeNo references Employee(employeeNo) Foreign Key supervisorEmployeeNo references Employee(employeeNo) Foreign Key secretaryEmployeeNo references Employee(employeeNo)
Branch	(branchNo, branchStreet, branchCity, branchState, branchZipCode, branchTelNo, branchFaxNo, managerEmployeeNo) Primary Key branchNo Alternate Key branchTelNo Alternate Key branchFaxNo Foreign Key managerEmployeeNo references Employee(employeeNo)
BusinessOwner	(ownerNo, businessName, businessAddress, businessTelNo, businessFaxNo, contactName, contactTelNo, contactFaxNo, contactEmailAddress) Primary Key ownerNo Alternate Key businessName Alternate Key businessTelNo Alternate Key businessFaxNo
Employee	(employeeNo, title, firstName, middleName, lastName, address, workTelExt, homeTelNo, empEmailAddress, socialSecurityNumber, DOB, position, sex, salary, typingSpeed, dateStarted, branchNo) Primary Key employeeNo Alternate Key socialSecurityNumber Foreign Key branchNo references Branch(branchNo)
Lease	(leaseNo, rentStart, rentFinish, depositPaid, renterNo, propertyNo) Primary Key leaseNo Foreign Key renterNo references Renter(renterNo) Foreign Key propertyNo references PropertyForRent(propertyNo)
PrivateOwner	(ownerNo, ownerName, ownerAddress, ownerTelNo) Primary Key ownerNo
PropertyForRent	(propertyNo, propStreet, propCity, propState, propZipCode, noRooms, rent, propTypeNo, ownerNo, employeeNo, branchNo) Primary Key propertyNo Foreign Key propTypeNo references PropertyType(propTypeNo) Foreign Key ownerNo references PrivateOwner(ownerNo) and BusinessOwner(ownerNo) Foreign Key employeeNo references Employee(employeeNo) Foreign Key branchNo references Branch(branchNo)
PropertyType	(propTypeNo, propTypeDescription) Primary Key propTypeNo
Renter	(renterNo, rFName, rLName, rAddress, rTelNo, maxRent, prefTypeNo) Primary Key renterNo Foreign Key prefTypeNo references PropertyType(propTypeNo)
Viewing	(propertyNo, renterNo, dateView, comments) Primary Key propertyNo, renterNo, dateView Foreign Key propertyNo references PropertyForRent(propertyNo) Foreign Key renterNo references Renter(renterNo)

د-۱۴ آژانس مسافرتی

آژانس مسافرتی میخواهد پایگاه داده ای را جهت رزرو فعالیتهای تعطیلاتی مشتریانش ایجاد کند. این آژانس دارای تعداد بسیاری شعبه گسترش یافته می باشد. مشتریان می توانند یا با صحبت کردن با آژانس، یا تماس گرفتن با تلفن، با شعبه ارتباط برقرار کرده و تعطیلاتی را رزرو کنند. تعطیلات معمولا شامل پرواز با هواپیما یا اسکان می باشد. هر وقت که آژانس مسافرتی تعطیلات مناسبی را برای مشتریان پیدا کرد، نیازمندیهای پرواز و اسکان برای مشتری رزرو می شود. با این وجود، عمل رزرو تنها می تواند به مدت ۲۴ ساعت که زمانی است که مشتری بایستی رزرو را قبول یا رد کند انجام گیرد. زمانی که رزرو قبول شد مشتری صورتحسابی داده می شود که بایستی به طور کامل و حداقل در ۴ هفته قبل از تاریخ حرکت بپردازد. نام کارکنانی که در آغاز عمل رزرو را انجام می دهند ثبت می شود. مدل داده منطقی در شکل د-۲۷ و جداول مربوطه در شکل د-۲۸ نشان داده شده است.

شکل د-۲۷

مدل منطقی داده آژانس مسافرتی



شکل ۲۸- د

جداول آژانس مسافرتی .

Branch	As defined in Section D.13.2
ApartFacility	(apartmentNo, facilityNo, comments) Primary Key apartmentNo, facilityNo Foreign Key apartmentNo references Apartment(apartmentNo) Foreign Key facilityNo references Facility(facilityNo)
Apartment	(apartmentNo, apartmentName, apartmentType, apartmentDescription, apartmentRating, apartmentStreet, apartmentCity, apartmentState, apartmentCountry, apartmentZipCode, noOfRooms, operatorNo, resortNo) Primary Key apartmentNo Foreign Key operatorNo references Operator(operatorNo) Foreign Key resortNo references Resort(resortNo)
Country	(countryNo, countryName) Primary Key countryNo Alternate Key countryName
Customer	(customerNo, customerName, customerStreet, customerCity, customerState, customerZipCode, custTelNo, custFaxNo, nationality, sex, DOB, passportNo) Primary Key customerNo Alternate Key custTelNo Alternate Key custFaxNo Alternate Key passportNo
CustomerParty	(customerNo, holidayNo) Primary Key customerNo, holidayNo Foreign Key customerNo references Customer(customerNo) Foreign Key holidayNo references Holiday(holidayNo)
Employee	(employeeNo, title, firstName, middleName, lastName, address, workTelExt, homeTelNo, empEmailAddress, socialSecurityNumber, DOB, position, sex, salary, dateStarted, branchNo) Primary Key employeeNo Alternate Key socialSecurityNumber Foreign Key branchNo references Branch(branchNo)
Facility	(facilityNo, description, additionalCharge) Primary Key facilityNo
Flight	(flightNo, planeType, seatCapacity, airportDepart, departTime, airportArrive, arriveTime, operatorNo) Primary Key flightNo Foreign Key operatorNo references Operator(operatorNo)
Hotel	(hotelNo, hotelName, hotelStreet, hotelCity, hotelState, hotelCountry, hotelZipCode, hotelTelNo, hotelFaxNo, hotelType, hotelDescription, hotelRating, hotelManagerName, operatorNo, resortNo) Primary Key hotelNo Foreign Key operatorNo references Operator(operatorNo) Foreign Key resortNo references Resort(resortNo)
HotelFacility	(hotelNo, facilityNo, comments) Primary Key hotelNo, facilityNo Foreign Key hotelNo references Hotel(hotelNo) Foreign Key facilityNo references Facility(facilityNo)

شکل ۲۸- د

ادامه . . .

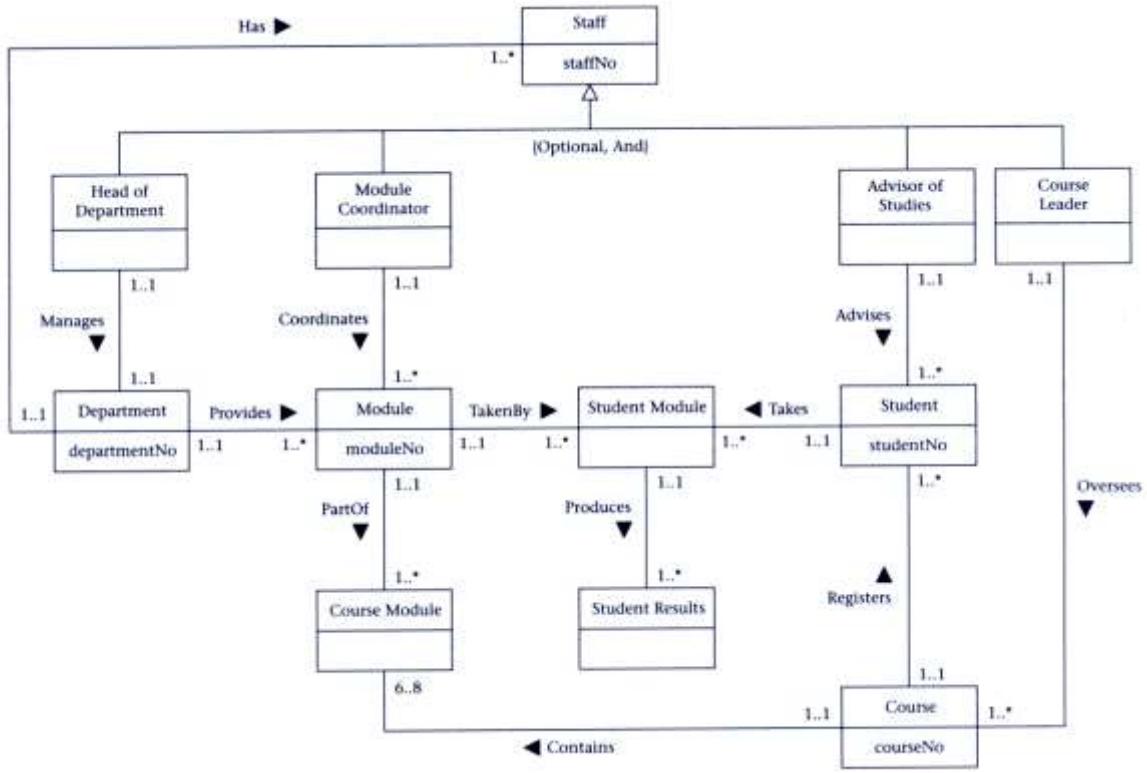
Holiday	(holidayNo, status, dateBooked, cateringType, startDate, finishDate, invoiceNo, totalCost, dateSent, datePaid, bookCustomerNo, hotelNo, apartmentNo, inwardFlightNo, inwardNoOfSeats, outwardFlightNo, outwardNoOfSeats, employeeNo, branchNo) Primary Key holidayNo Foreign Key bookCustomerNo references customer(customerNo) Foreign Key hotelNo references Hotel(hotelNo) Foreign Key apartmentNo references Apartment(apartmentNo) Foreign Key inwardFlightNo references Flight(flightNo) Foreign Key outwardFlightNo references Flight(flightNo) Foreign Key employeeNo references Employee(employeeNo) ForeignKey branchNo references Branch(branchNo)
Operator	(operatorNo, operatorName, operatorType, operatorStreet, operatorCity, operatorState, operatorZipCode, operTelNo, operFaxNo, contactName, contactTelNo, contactFaxNo, contactEmailAddress) Primary Key operatorNo Alternate Key operTelNo Alternate Key operFaxNo
Resort	(resortNo, resortName, distanceFromAirport, timeFromAirport, countryNo) Primary Key resortNo Foreign Key countryNo references Country(countryNo)

د-۱۵ نتایج دانشجو

دانشگاهی میخواهد پایگاه داده ای را جهت ثبت نتایج دانشجویان ایجاد کند. زمانیکه دانشجو به دانشگاه ملحق می شود در دوره آموزشی ثبت نام می کند. هر دانشجو همچنین مشاور آموزشی دارد. هر سال دوره آموزشی از واحدهایی تشکیل شده است. حداقل و حداکثر تعداد واحدهایی که سال آموزشی را تشکلی می دهند، بترتیب ۶ و ۸ می باشند. دانشجو باید هر واحد را گرفته و در سال تعیین شده آن پاس کند قبل از اینکه مجاز به انتقال، به دوره آموزشی دیگر شود. دانشجو معمولا سه فرصت دارد که واحد را پاس کند؛ با این وجود فرصتهای دیگری نیز بر اساس صلاحدید دانشگاه به او داده می شود. واحد مشخصی می تواند به عنوان قسمتی از یک یا چند دوره آموزشی پیشنهاد شود. دانشگاه چندین بخش دارد که هر یک از آنها سهمی از دوره آموزشی را پیشنهاد می کنند. هر بخش دارای رئیس بخش بوده و دوره آموزشی راهنمای دوره آموزشی مختص خود را دارد. هر واحد به عضوی از پرسنل واگذار شده است که هماهنگ کننده واحد نام دارد، که مسئول نظارت بر تدریس و اظهار نظر بر واحدها بر عهده او است. مدل داده منطقی در شکل د-۲۹ و جداول مربوطه در شکل د-۳۰ نشان داده شده است.

شکل د-۲۹

مدل منطقی داده نتایج دانشجو .



شکل د-۳۰

جداول نتایج دانشجو .

Course	(courseNo, courseName, level, entranceRequirements, maxNumber, departmentNo, courseLeaderNo) Primary Key courseNo Alternate Key courseName Foreign Key departmentNo references Department(departmentNo) Foreign Key courseLeaderNo references Department(courseLeaderNo)
CourseModule	(courseNo, moduleNo) Primary Key courseNo, moduleNo Foreign Key courseNo references Course(courseNo) Foreign Key moduleNo references Module(moduleNo)
Department	(departmentNo, departmentName, location, HODstaffNo) Primary Key departmentNo Alternate Key departmentName Foreign Key HODstaffNo references Staff(staffNo)
Module	(moduleNo, moduleName, semesterDelivered, moduleAims, moduleObjectives, moduleSyllabus, moduleResources, moduleModeOfAssessment, moduleCoordinatorStaffNo, departmentNo) Primary Key moduleNo Alternate Key moduleName Foreign Key moduleCoordinatorStaffNo references Staff(staffNo) Foreign Key departmentNo references Department(departmentNo)
Staff	(staffNo, title, firstName, lastName, address, homeTelNo, workTelExt, empEmailAddress, socialSecurityNumber, DOB, position, sex, salary, dateStarted, departmentNo) Primary Key staffNo Alternate Key socialSecurityNumber Foreign Key departmentNo references Department(departmentNo)
Student	(studentNo, studentFirstName, studentMiddleName, studentLastName, studentHomeStreet, studentHomeCity, studentHomeState, studentHomeZipCode, studentHomeTelNo, familyHomeStreet, familyHomeCity, familyHomeState, familyHomeZipCode, familyHomeTelNo, studentDOB, studentSex, nationality, courseNo, advisorStaffNo) Primary Key studentNo Foreign Key courseNo references Course(courseNo) Foreign Key advisorStaffNo references Staff(staffNo)
StudentModule	(studentNo, moduleNo) Primary key studentNo, moduleNo Foreign Key studentNo references Student(studentNo) Foreign Key moduleNo references Module(moduleNo)
StudentResult	(studentNo, moduleNo, attempt, attemptDate, mark, proposal, additionalComments) Primary key studentNo, moduleNo, attempt Foreign Key studentNo, moduleNo references StudentModule(studentNo, moduleNo)