خطوط جریان داده

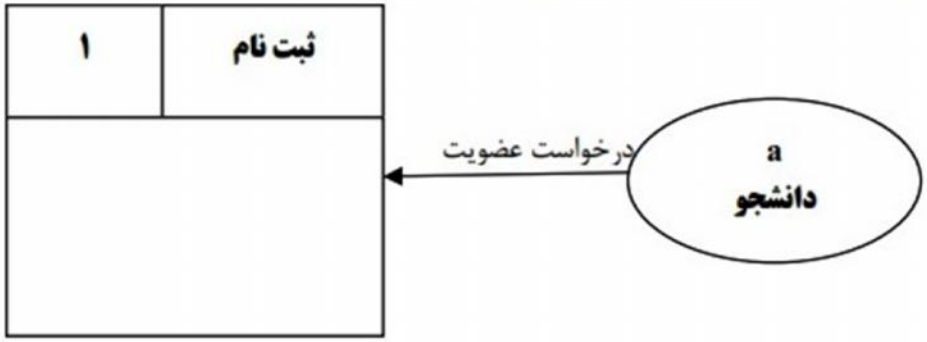
شرح خطوط جریان داده:

باید اطلاعات و داده‌های موجود بر روی خطوط جریان داده‌هاقبل از تشریح پردازه‌های جزئی کاملاً مشخص شوند و اسامی فیلدهای موجود برروی خطوط جریان تشریح شوند تا بتوان بطور دقیق این اسامی را در شرح پردازه‌های جزئی مورد استفاده قرار داد.

برای تشریح خطوط جریان داده‌ها باید کلیه فیلدهای اطلاعاتی را بر روی خطوط جریان مشخص نمود. البته در این مرحله نیازی به مستند نمودن جزئیات دقیق فیلدها نیست، زیرا در تعریف ورودی خروجی توابع نیز ساختار دقیق فیلدها مشخص می شود.

در فیلد فرمت ساختار یا قالب بندی فیلد مشخص می شود. برای این منظور رشته‌های کاراکتری به طول n را بصورت X(n) و اعداد با طول m را به صورت 9(m) مشخص می نمایند. برای اعداد اعشاری با nرقم صحیح و m رقم اعشار قالب بندی 9(n).9(m) بکار برده می شود. چنانچه فیلدی نمایانگر تاریخ روز باشد می توان آنرا از نوع Date معرفی نمود. در ستون توضیحات، شرح مختصری برای فیلد اطلاعاتی مربوطه می توان اضافه نمود.

مثال:



از	به	نام خط	نام فیلد	برچسب	نوع فیلد	شرح(توضیح)
a	۱	درخواست عضویت	تاریخ درخواست	Re_date	X(10)	
			شماره دانشجویی	ID	9(8)	
			نام دانشجو	Name	X(25)	
			رشته	Course	X(10)	
			مقطع تحصیلی	Level	X(10)	
			...			

پردازه‌های جزئی

اگر پردازه در سطحی باشد که تجزیه آن به سطوح پایین تر نیازمندی جدیدی را مشخص نکند و به درک تحلیلگر از سیستم اضافه نکند در اینصورت پردازه را جزئی می نمایند و دیگر آنرا تجزیه نمی کنند.

شماره شناسایی پردازه مقدماتی : ۱
نام پردازه مقدماتی : ثبت نام عضو
شرح : مشخصات دانشجو دریافت می‌گردد. کارت عضویت باید صادر گردد. . . .

- در قسمت شرح می‌توان بوسیله عبارتهای مانند if، while و ... نحوه عملکرد پردازه را مشخص کرد.
- اگر از فرمولهای خاصی برای محاسبات استفاده می‌شود باید این فرمولها مشخص شوند.
- برای مشخص شدن پردازه‌های جزئی (مقدماتی) از علامت */ در قسمت پایین-راست پردازه استفاده می‌شود.

شرح پردازه های جزیی

اگر پردازه ای توسط دیاگرام گردش داده ها گسترش داده نشود به آن پردازه صفت جزیی اطلاق می گردد. برای بیان بدون ابهام چگونگی عملکرد پردازه های جزیی معمولاً از شبه دستورالعمل ها (Pseudo code) استفاده می شود. البته استفاده ازجداول تصمیم گیری تا حدی رایج است. روش رایج دیگر استفاده از فلوچارت بوده است. این روش امروزه مرسوم نیست. در دو بخش بعدی چگونگی استفاده از شبه دستورالعمل ها و جداول تصمیم گیری برای بیان پردازه ها درون DFD بررسی شده است.

شبه دستورالعمل ها

شبه دستورالعمل ها بسیار نزدیک به دستورالعمل های زبانهای برنامه سازی می باشند. اصولاً ، قالب بندی کلی برای این نوع دستورالعمل ها وجود ندارد و وابسته به سلیقه برنامه نویس می باشد. امّا ، نکاتی در مورد چگونگی بکار گیری و تعیین شبه دستورالعملها جهت تعریف پردازه ها وجود دارد که در این قسمت مورد بحث واقع شده است.

برای بیان پردازه ها معمولاً از جملات مختصر و گویای امری در قالبی نزدیک به دستورالعمل های زبانهای برنامه سازی استفاده می شود. در واقع این خود بگونه ای برنامه نویسی است اما، دیگر جزئیات مشخص و مشکلات و محدودیت های عمومی دیگر برنامه سازی از آن حذف می گردد. در این راستا، دستورالعمل های زبانهای سطح بالا و نسل چهارم می توانند تا حدی راه گشا باشند. در اینجا تجربه برنامه نویسی تحلیلگر کمک بسیاری است در بیان عملکرد پردازه ها در قالبی منطقی و مقبول هر زبان برنامه سازی . لذا، فردی که پردازه ها را بیان میکند، باید از تجربه بسیار خوبی در زمینه برنامه نویسی برخوردار بوده ، برنامه نویس قابلی باشد. تجربه و قابلیت آنالیست موجب می گردد که بدون نیاز به هیچگونه تغییر بنیانی و بر اساس ترتیب ظهور شبه دستورالعمل ها، برنامه خود را برای هر شبه دستورالعمل توسعه دهد. در واقع ، شبه دستورالعمل ها باید بتوانند نقش جملات تفسیری (Comment) را برای برنامه داشته باشند.

کار برنامه نویسی امری است دقیق . لذا ، در بیان پردازه ها اگرچه جزئیات برنامه نویسی در نظر گرفته نمی شود اما ، هیچگونه نکته مبهمی نباید وجود داشته باشد. باید همه آمار و ارقام دقیقا" در شرح پردازه ها مشخص شوند و از مجموعه شبه دستورالعمل های محدودی نزدیک به زبان برنامه سازی سطح بالا استفاده شود. از معادلات جبری نیز می توان به عنوان شبه دستورالعمل استفاده نمود. افعالی مثل :

بخوان ، بنویس ، بگذار یا بپذیر (Read, Write, Put , Accept) ،

پیداکن ، جستجو کن یا مشخص کن (Find, Search, Locate) ،

جمع کن ، تفریق کن ، ضرب کن و تقسیم کن (Add, Subtract, Multiply, Divide) ،

محاسبه کن ، قرار بده ، انتقال بده (Compute, Set, Move) ،

جایگزین کن (replace) و

مرتب کن (sort)

استفاده می شود. معمولاً در حدود ٤٠ تا ٥٠ فعل برای بیان کامل پردازه ها کافی است. مفعول این جملات ساده امری (شبه دستورالعمل ها) باید حتی المقدور شامل داده هایی باشد که به نحوی مشخص و معین شده ، برای خواننده مبهم نباشد. برای مثال به کد زیر توجه نمایید:

قرار بده مجموع_روزانه را مساوی با صفر

مادامیکه سفارش بیشتری در فایل سفارشات وجود دارد

انجام بده

بخوان سفارش بعدی را در فایل سفارشات با شرط تاریخ_سفارش = تاریخ_امروز

اگر مبلغ_سفارش > ۵۰۰۰ تومان

آنگاه نمایش بده برای واحد حسابداری مبلغ_سفارش و نام_مشتری را

وگرنه نمایش بده مبلغ_سفارش ، نام_مشتری و سفارش_دهنده را.

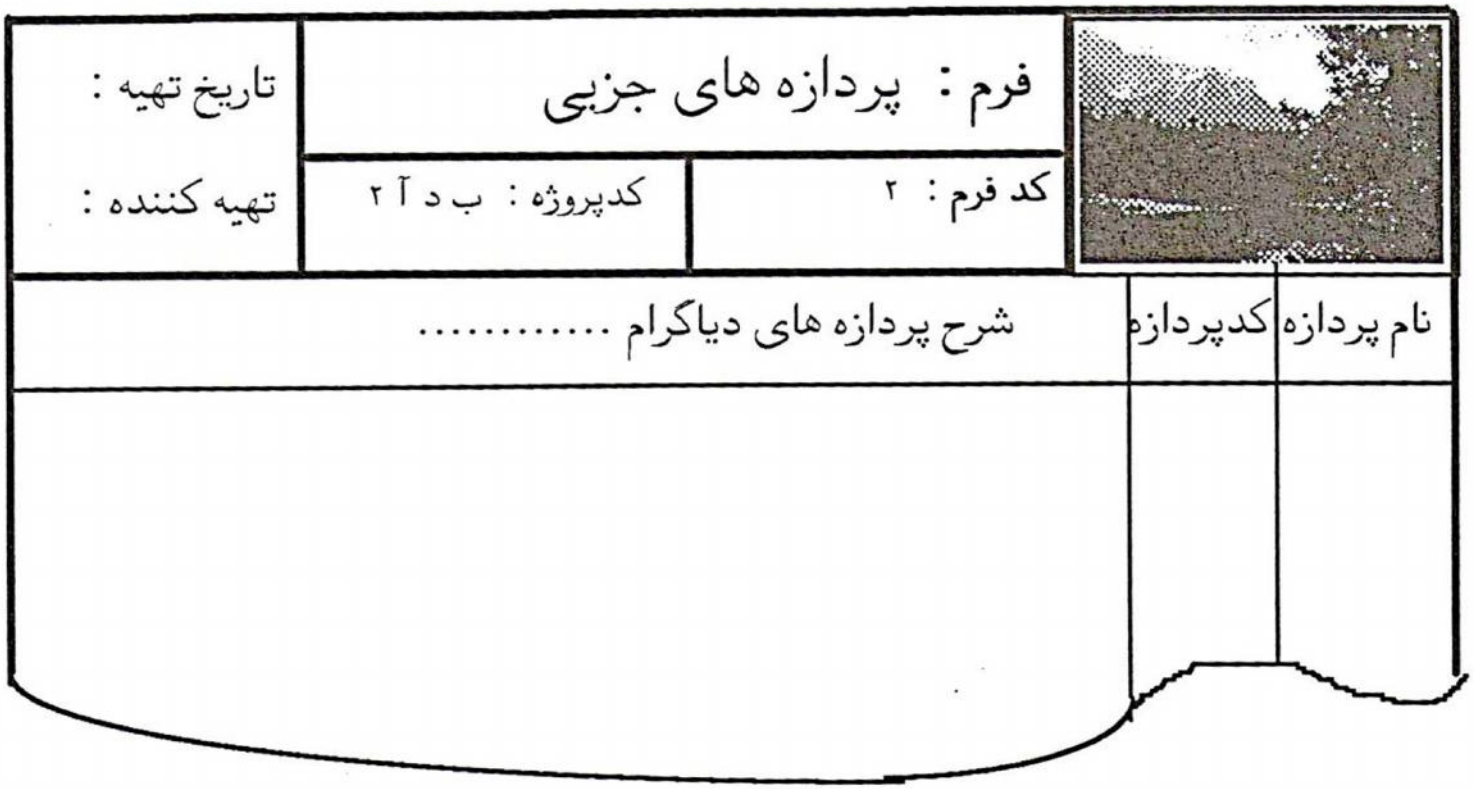
پایان اگر

محاسبه کن مجموع_روزانه = مجموع_روزانه + میزان_سفارش

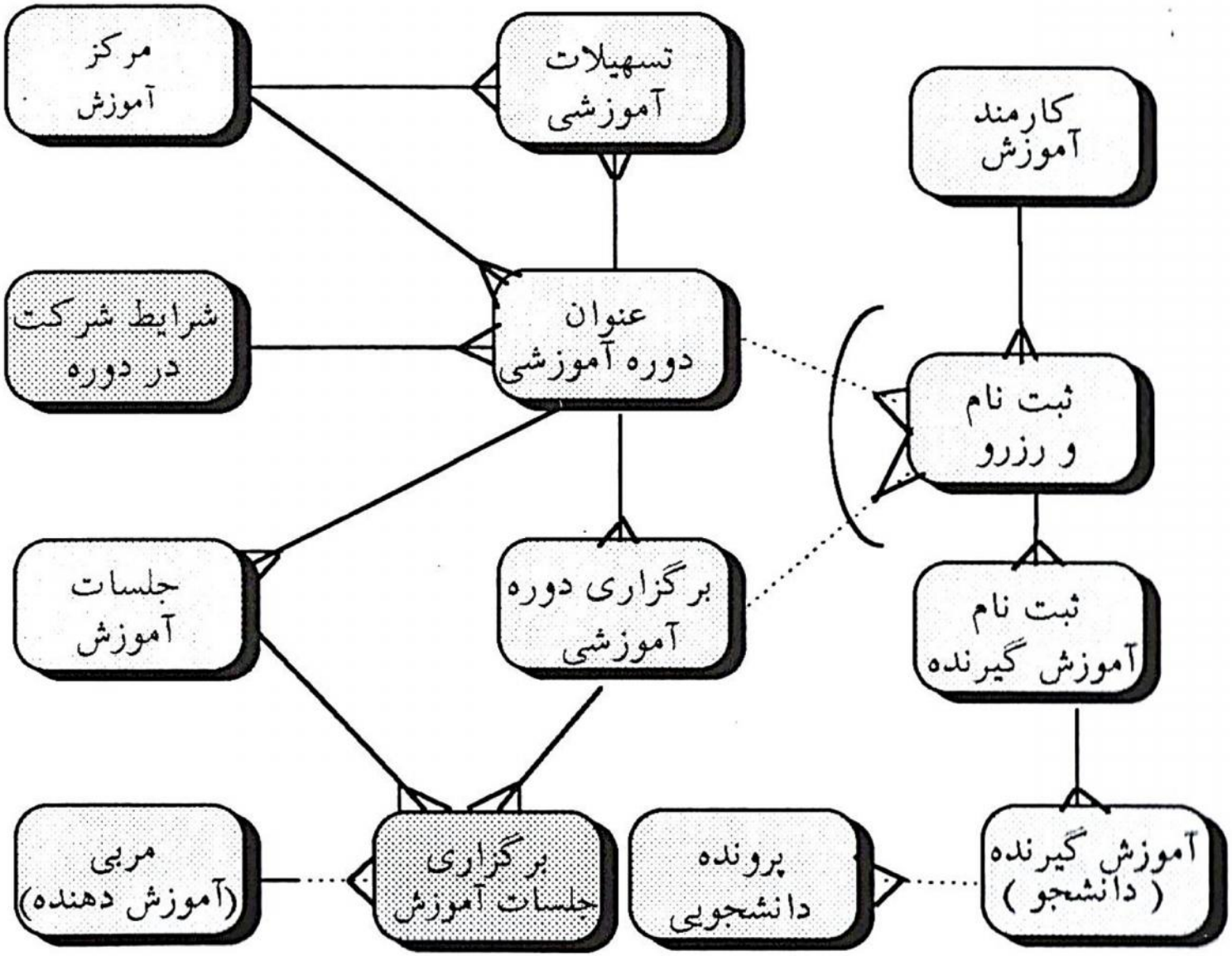
نمایش بده برای واحد حسابداری مجموع_روزانه را

پایان مادامیکه

فیلدهای اطلاعاتی مثل مبلغ_سفارش و ساختار فایل یا جدول (table) سفارش باید از قبل معین شده باشد. از جملات Goto یا پرش باید اکیداً دوری نمود. ساختارهای رایج برای برنامه نویسی ساخت یافته از قبیل مادامیکه (while) ، اگر (if) ، چنانچه (Case) و تکرارکن (repeat) را می توان استفاده نمود. برای نمایش پردازه های جزیی فرم زیر استفاده می شود.



انباره های منطقی



مدل منطقی داده ها

یک انباره ، مکانی برای حفظ و نگهداری داده ها و اطلاعات است. لذا، در هنگام نام گذاری انباره ها از بکار بردن کلمه داده یا اطلاعات در نام آنها باید خود داری کرد. یک انباره چندین نمونه از موجودیتها را ذخیره می کند . برای نمونه انباره دانشجویان در سیستم آموزش دانشگاه حاوی نام فقط یک دانشجو نیست. این انباره حاوی اطلاعات کلیه دانشجویان است. ساختار رکورد انباره دانشجویان را موجودیت دانشجو می گویند. بنابراین ، مشاهده می کنید که نام انباره باید نمایانگر جمع و یا مجموعه ای از داده ها باشد . پس ، بهتر است که برای نامگذاری انباره ها اسامی مفرد استفاده نشود.

باید توجه داشته باشید که در داخل دیاگرام گردش داده ها تنها از طریق پردازه ها است که می توان به انباره ها دسترسی داشت. وجود خط جریان از یک انباره به پردازه نمایانگر خواندن داده از داخل انباره به توسط آن پردازه است. بالعکس وجود خط جریان از پردازه به انباره نمایانگر عمل بروز رسانی داده های درون انباره به توسط پردازه است. واضح است که قبل از اصلاح اطلاعات درون یک انباره باید آن اطلاعات را از داخل انباره خواند. در دیاگرام گردش داده ها خط جریانی جداگانه برای خواندنِ داده هایی از یک انباره که قرار است بلافاصله اصلاح شود ترسیم نمی شود.

مدل منطقی

مدل منطقی داده‌ها(Logical Data Model)

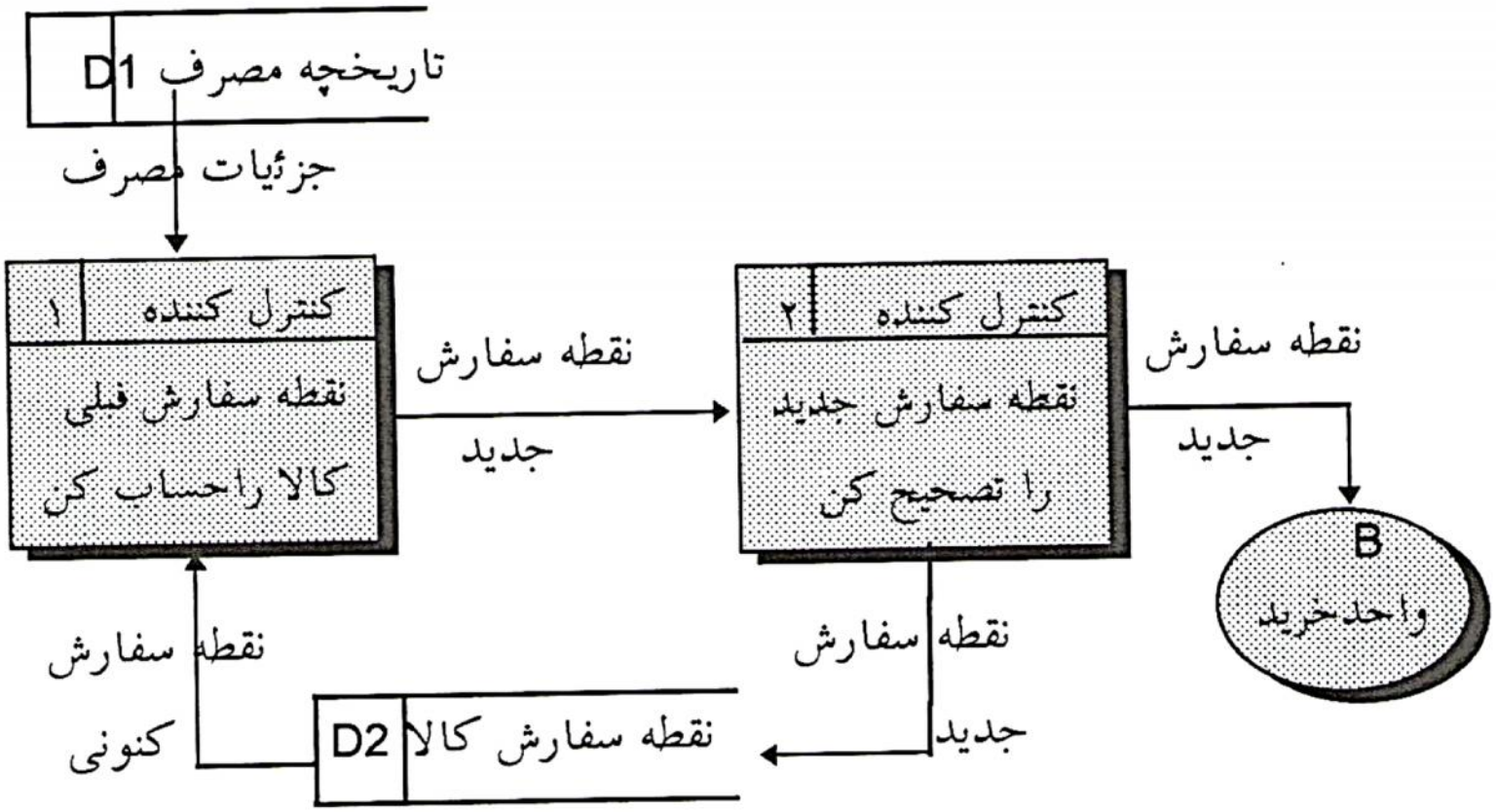
مدل منطقی داده‌ها نمایانگر روابط منطقی بین داده‌ها و اطلاعاتی است که در یک سیستم نگهداری می‌شوند.

موجودیتهای یک سیستم توسط اطلاعات و داده‌های به کار رفته درون آن سیستم مشخص می گردند. موجودیتها معمولاً یا به صورت واقعی و یا به صورت مفهومی هستند. برای نمونه ماشین و محصول موجودیتهای واقعی و عنوان درس یک مفهوم موجود می باشد.

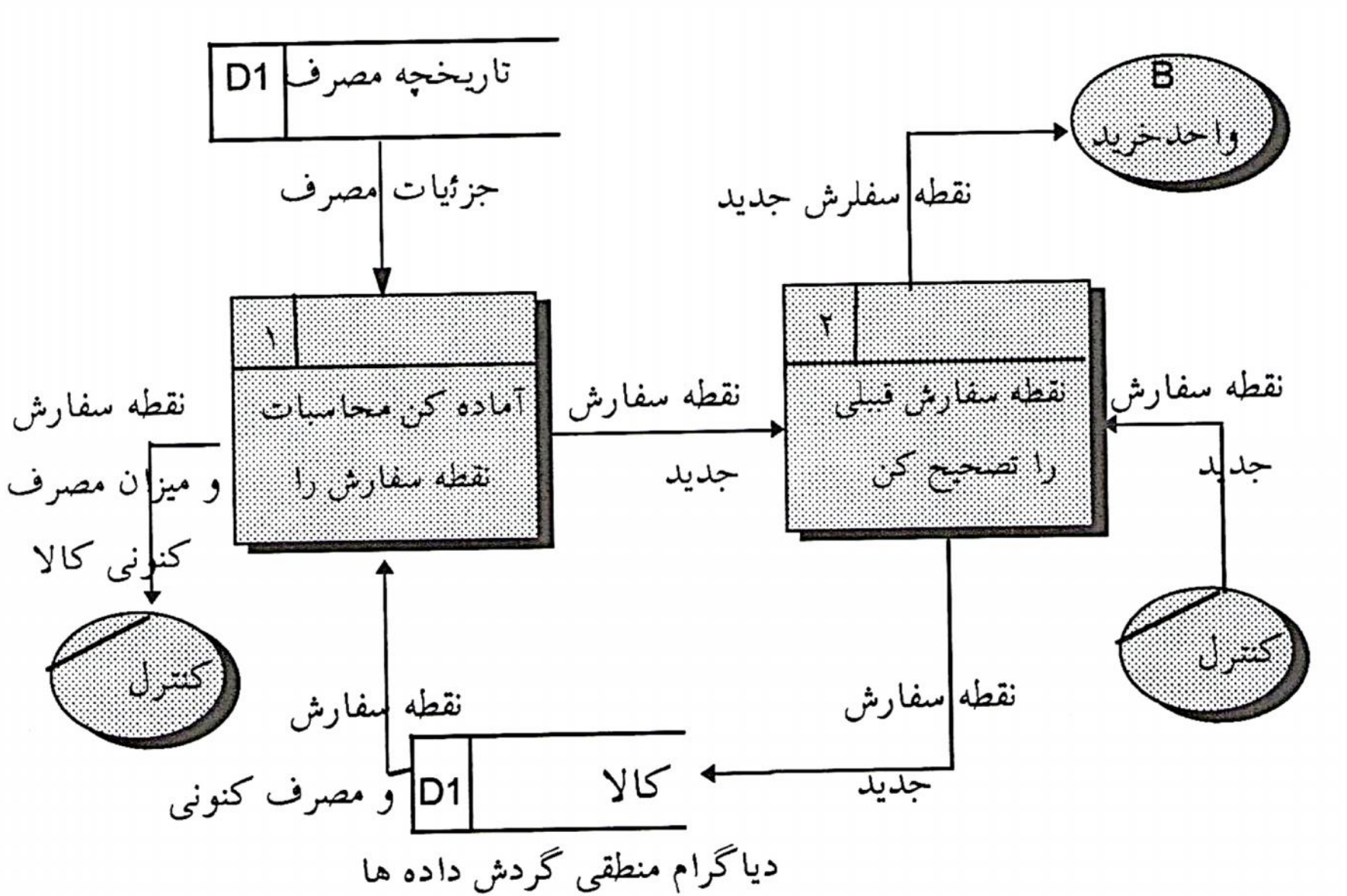
موجودیت منطقی: هر گونه شیء یا مفهومی که به نگهداری اطلاعات پیرامون آن نیاز داریم

بطور کلی یک موجودیت یا عنصر دارای مشخصات زیر است:

۱. باید نمایانگر ساختار داده‌ها و اطلاعات مورد توجه سیستم باشد.
۲. چندین رکورد یا سابقه برای یک موجودیت وجود داشته باشد.
۳. برای هر رکورد موجود یک کلید دسترسی یگانه موجود باشد.



محاسبه حداقل موجودی کالا در انبار

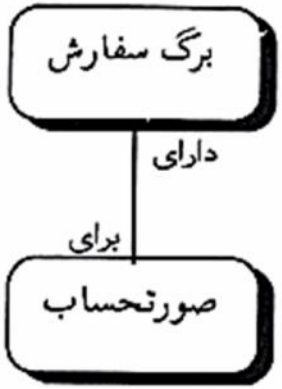


دیاگرام منطقی گردش داده ها

ارتباط بین موجودیتها

ارتباط یک به یک (1:1)

اگر برای یک برگ سفارش کالا فقط یک صورتحساب موجود باشد آنگاه ارتباط بین این دو موجودیت یک به یک و به صورت زیر نمایش داده می‌شود.



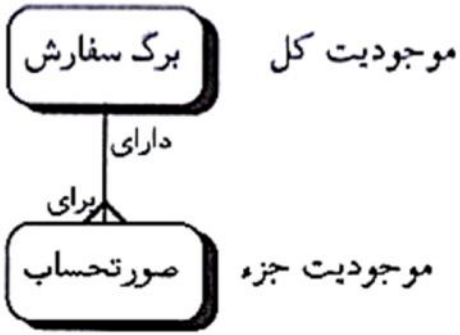
ارتباط یک به یک

هر برگ سفارش باید داری یک برگ صورتحساب باشد.

هر صورتحساب باید برای یک برگ سفارش باشد.

رابطه یک به چند (1:m)

اگر برای یک برگ سفارش کالا امکان دریافت بیش از یک صورتحساب باشد، رابطه یک به چند و به صورت زیر است.

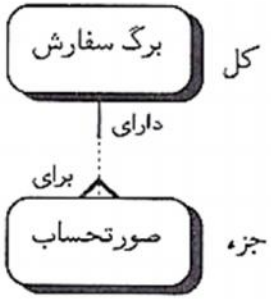


ارتباط یک به چند

هر برگ سفارش باید دارای یک یا چند صورتحساب باشد.

هر صورتحساب باید برای یک برگ سفارش باشد.

ممکن است مأمور خرید بدون دریافت برگ رسمی سفارش اقدام به خرید اجناس ضروری نماید و برای یک صورتحساب برگ سفارشی موجود نباشد. در اینصورت از خطوط خط چین برای نمایش ارتباط بین صورتحساب و برگ سفارش استفاده می‌شود.



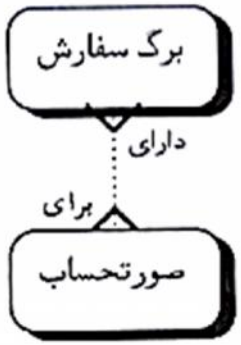
ارتباط یک به چند

هر برگ سفارش باید داری یک یا چند صورتحساب باشد.

هر صورتحساب ممکن است برای یک برگ سفارش باشد.

رابطه چند به چند(m:n)

یک صورتحساب ممکن است برای بیش از یک برگ سفارش صادر شود و یا اینکه برای یک برگ سفارش اصلاً خریدی انجام نشود و صورتحسابی وجود نداشته باشد.



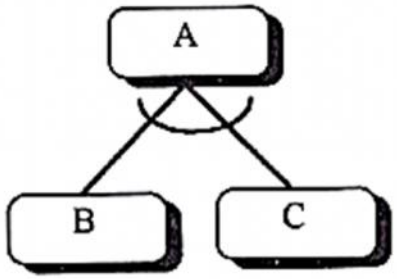
ارتباط چند به چند

هر برگ سفارش ممکن است دارای یک یا چند صورتحساب باشد.

هر صورتحساب ممکن است برای یک یا چند برگ سفارش باشد.

رابطه یا انحصاری (Exclusive OR)

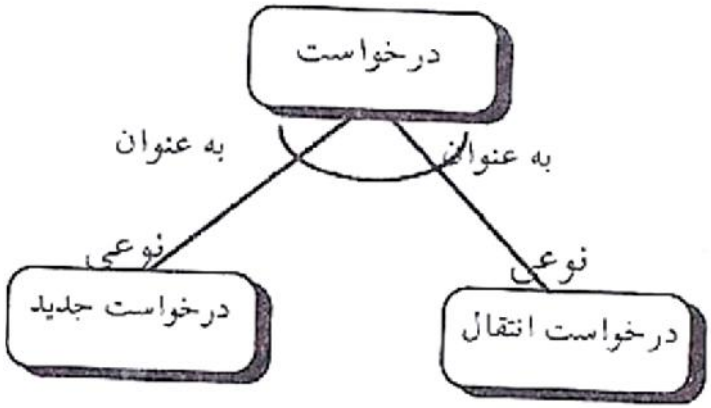
رابطه بین موجودیتها ممکن است به صورت یا انحصاری باشد. به عبارت دیگر در حالت کلی، موجودیت A در ارتباط با موجودیت B یا C است اما، در یک زمان نمی‌تواند در ارتباط با هر دو باشد.



رابطه یا انحصاری

هر درخواست باید به عنوان یک درخواست انتقال و

یا باید به عنوان یک درخواست جدید باشد.

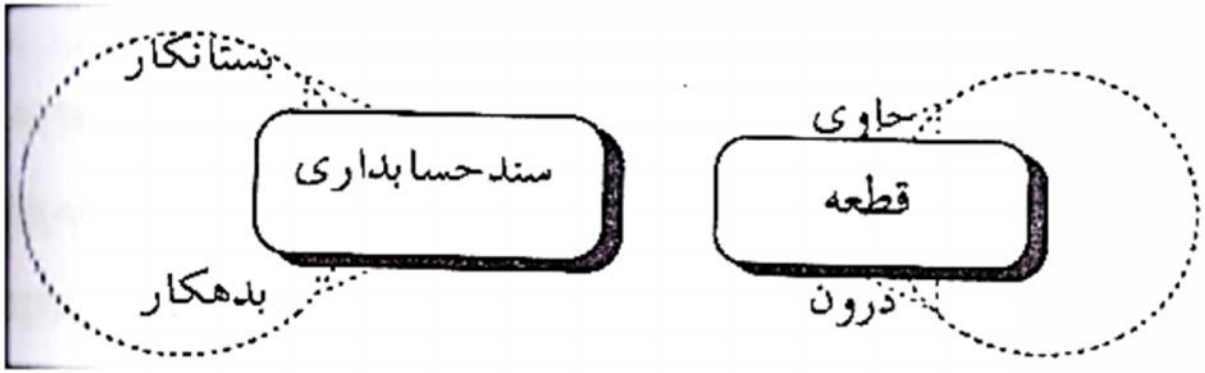


نمایش "یا انحصاری"

رابطه بازگشتی

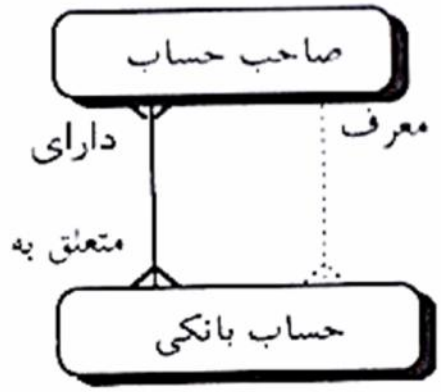
ممکن است یک موجودیت در ارتباط با خودش باشد. برای مثال درون یک قطعه از یک دستگاه ممکن است چند قطعه دیگر وجود داشته باشد و یک قطعه را می‌توان در چندین قطعه دیگر به کار برد. بنابراین، موجودیت قطعه در ارتباط یک به چند با خود است.

یک سند حسابداری ممکن است طرف بدهکار و یا بستانکار یک یا چند سند دیگر باشد. همچنین یک سند ممکن است دارای یک یا چند طرف بدهکار یا بستانکار باشد.



چند ارتباط بین دو موجودیت

بین دو موجودیت ممکن است بیش از یک ارتباط موجود باشد. برای مثال در یک سیستم بانکی می‌توان در نظر داشت که افراد ممکن است در بانک حساب دیگری داشته باشند. یک حساب بانکی نیز ممکن است به چند نفر متعلق باشد.



چند ارتباط بین دو موجودیت

کار در کلاس ۷:

مدل منطقی سیستم تولید پوشاک را ترسیم نمایید.

ترسیم مدل منطقی داده‌ها

برای ترسیم مدل منطقی داده‌ها در اولین مرحله باید موجودیتهای سیستم را مشخص نمود. سپس چگونگی ارتباط بین موجودیتها معین می‌شود.

مثال: در سیستم ساده سفارشات، مشتری سفارش کالا می دهد و صورتحساب برای وی ارسال می‌شود. موجودیتهای این سیستم عبارتند از:

- مشتری
- سفارش
- صورتحساب
- کالا

برای مشخص شدن هر موجودیت باید خواص یا فیلدهای آن موجودیت را مشخص کرد.

۱) مشتری : (کد مشتری، نام، تلفن، آدرس، شماره حساب بانکی)

۲) سفارش: (کد سفارش، کد مشتری، نام مشتری، آدرس مشتری، تلفن مشتری، تاریخ دریافت سفارش، {کد کالا، نام کالا، واحد، تعداد سفارش})

۳) صورتحساب: (کد صورتحساب،کد سفارش، کد مشتری، نام مشتری، آدرس مشتری، تلفن مشتری، تاریخ صورتحساب، {کد کالا، نام کالا، واحد، تعداد سفارش، قیمت واحد، قیمت کل}، مبلغ کل صورتحساب)

۴) کالا: (کد کالا، نام کالا، واحد، تعداد، نقطه سفارش، مکان)

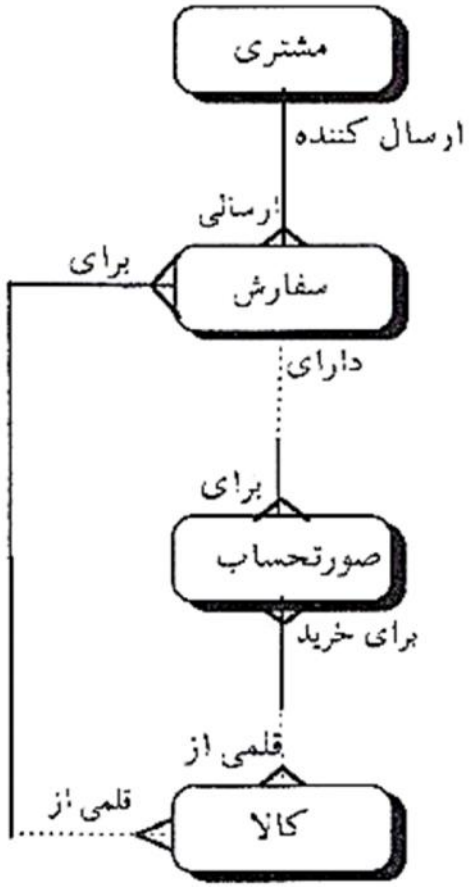
آکولاد به معنای تکرار است. برای مثال در یک برگ سفارش کالا ممکن است چند قلم کالای متفاوت مشخص شود.

پس از تعیین موجودیتها باید روابط را مشخص کرد. برای هر رابطه بین دو موجودیت نکات زیر باید معین شود:

۱) درجه ارتباط

۲) قطعیت ارتباط(اختیاری یا اجباری بودن ارتباط)

۳) نام



مدل منطقی داده ها برای سیستم سفارشات

یک مشتری باید حداقل یک سفارش کالا داده باشد.

برای یک برگ سفارش ممکن است صورتحسابی موجود نباشد و یا اینکه چند صورتحساب تهیه شود.

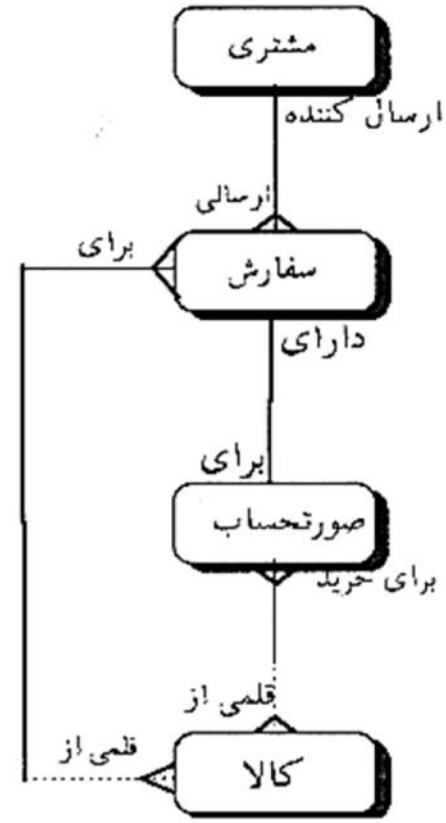
صورتحساب باید برای خرید یک یا چند کالا باشد. اما، این دلیل نمی‌شود که برای هر کالا باید حتماً یک صورتحساب وجود داشته باشد. ممکن است کالا در لیست سفارشات باشد، اما صورتحسابی برای خرید آن وجود نداشته باشد.

تبدیل روابط

هدف از مدل سازی منطقی بهینه سازی ساختار بانک اطلاعاتی است. (برای هر موجودیت یک فایل یا جدول (Table) در نظر گرفته می‌شود.)

۱- حذف رابطه یک به یک حتمی: رابطه یک به یک حتمی بین دو موجودیت را با ادغام آن دو موجودیت می‌توان حذف نمود.

مثال: با در نظر گرفتن فیلدهای موجودیتهای سفارش و صورتحساب در شکل قبل می‌توان ارتباط بین این دو موجودیت را بصورت یک به یک و حتمی در نظر گرفت. همیشه برای هر سفارش یک صورتحساب باید وجود داشته باشد و بالعکس برای هر صورتحساب باید همیشه یک برگ سفارش موجود باشد. پس نیازی به دو پرونده مجزا برای حفظ نمونه‌های این دو موجودیت نیست و در واقع این دو یکی هستند.

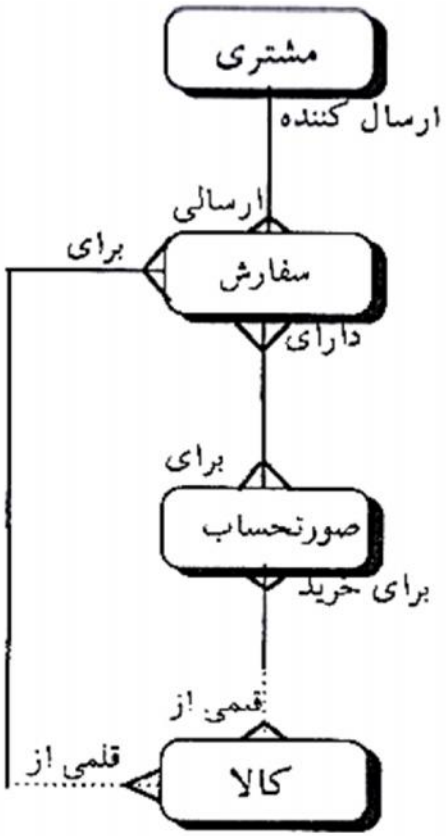


مدل منطقی داده ها با ارتباط یک به یک

مدل پس از ادغام ارتباط یک به یک

به‌این ترتیب با ادغام دو موجودیت سفارش و صورتحساب موجودیت جدیدی به نام فروش ایجاد می‌شود.

در حالت کلی برای یک سفارش ممکن است بیش از یک صورتحساب وجود داشته باشد. همچنین ، ممکن است یک صورتحساب برای چند سفارش یک مشتری تهیه شود. در اینصورت رابطه چند به چند و بصورت شکل زیر خواهد بود.



۶,۱۴- مدل منطقی داده ها با ارتباط چند به چند

۲- حذف ارتباط چند به چند: ارتباطات چند به چند موجب افزونگی در حجم داده‌ها می شوند،لذا باید حذف شوند.

رکورد فایل سفارش را در نظر بگیرید. جهت مشخص کردن صورتحسابهای مربوط به یک سفارش باید کد صورتحسابهای متفاوت را در رکوردهای فایل سفارش مشخص نمود. چون طول رکوردها ثابت است پس باید یک تعداد حداکثر برای تعداد فیلدها کد صورتحساب در نظر گرفت. بنابراین ساختار موجودیت سفارش بصورت زیر تبدیل خواهد شد:

کد سفارش	تاریخ	کد صورتحساب ۱	...	کد صورتحساب n

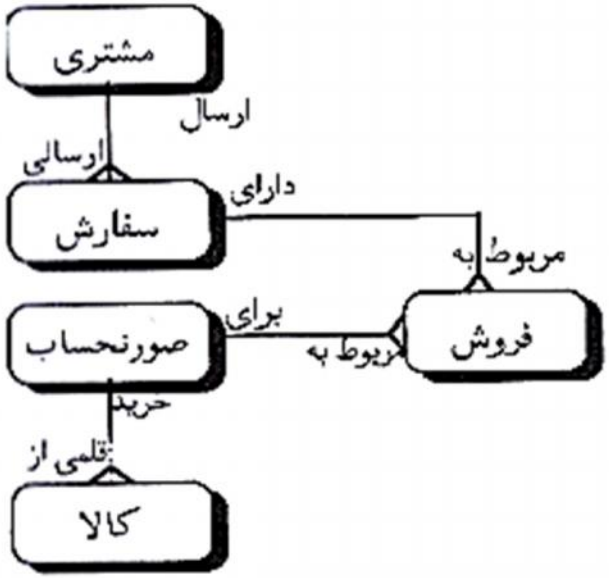
با معین کردن حداکثر تعدا صورتحسابها و ثابت نمودن تعدادآنها نه تنها یک محدودیت بوجود آورده شده، بلکه افزونگی حجم اطلاعات نیز ایجاد شده است.

اگر برای هر سفارش چند صورتحساب وجود داشته باشد و برای هر صورتحساب فقط یک سفارش موجود باشد، رابطه یک به چند است و مشکلی وجود ندارد. زیرا با در دست داشتن شماره سفارش می‌توان در داخل فایل صورتحساب به جستجو پرداخت و کلیه صورتحسابهایی که کد سفارش کالا در آنها با کد سفارش مورد نظر یکی است را مشخص نمود.

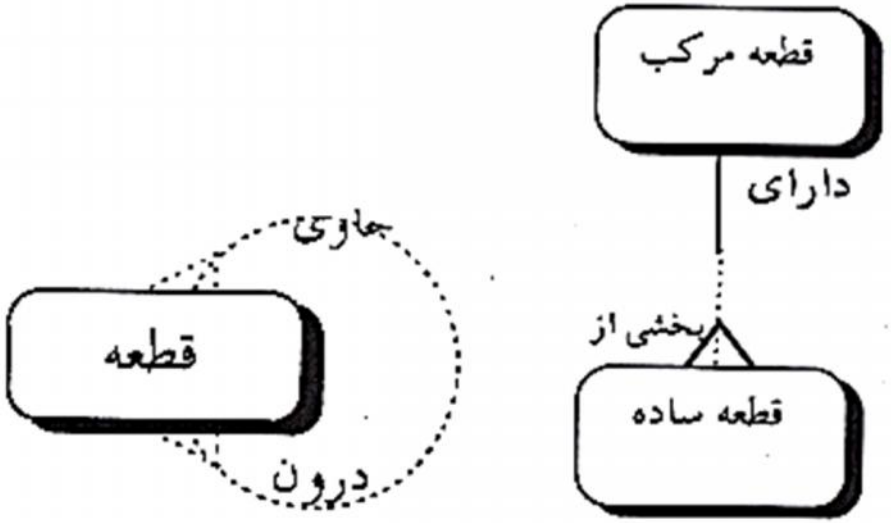
باید روابط چند به چند را با افزایش یک موجودیت واسطه تبدیل به دو رابطه یک به چند نمود.

فروش: (کد سفارش،کد صورتحساب)

بنابراین، ساختار منطقی داده‌ها برای سیستم ساده سفارشات به صورت زیر تبدیل می‌شود.



۳- حذف ارتباط چند به چند حلقوی: این روابط را با افزایش موجودیتهای واسطه به صورت زیر می‌توان تبدیل نمود.



تحلیل رابطه ای داده ها

تحلیل رابطه ای داده ها ، روشی است برای شناخت رابطه بین داده هایی که در یک سیستم نگهداری می شوند. هدف از این نوع تحلیل ، ایجاد ساختار منطقی داده ها با حداقل افزونگی در حجم داده های نگهداری شونده ، در سیستم است. حاصل تحلیل رابطه ای باید با حاصل روش ایجاد مدل منطقی که در بخشهای قبلی توضیح داده شد یکسان باشد. در واقع تحلیل رابطه ای در SSADM روشی است برای اطمینان از صحت مدل منطقی داده ها که با تحلیل موجودیتها و روابط بین آنها ایجاد می شود.

از دیدگاه تحلیل رابطه ای ، یک رابطه معادل با یک موجودیت است. ساختار رابطه به صورت یک جدول دو بعدی است . هر ستون یک خاصیت یا در واقع فیلدی از رابطه را معین می کند. هر ردیف جدول نمایانگر مقادیر فیلدها و یا ویژگیهای رابطه است. در واقع یک جدول نوعی فایل با شرایط زیر است :

- هیچ دو ردیف جدول با یکدیگر یکسان نیستند.
- ترتیب ردیفهای جدول مهم نیست .
- ترتیب ستون های جدول مهم نیست .
- هر ستون دارای نامی یگانـه اسـت. به عبارت دیگر اسامی فیلدها تکراری نیست.
- هر یک از فیلدها و یا ویژگیهای یک رابطه طبیعی ، ساده بوده ، خود یک رابطه نیستند.

منظور از رابطه طبیعی یا نرمالیزه نوعی خاص از رابطه است که دارای کلیه شرایط فوق است. در روابط غیر طبیعی آخرین شرط از شرایط فـوق صادق نیسـت. برای نمونـه در شکل زیر رابطه کارمندی یک رابطه طبیعی اسـت. در ایـن شـکل ، *شماره کارمندی* ،کلید اصلی[1] برای دسترسی به اطلاعات کارمندی اسـت. مقدارکلید اصلی در هر ردیف باید متفاوت از

---

[1] کلید اصلی = Primary Key

ردیف های دیگر باشد. در اینجا شماره کارمندی یک کلید ساده است. کلید ساده فقط شامل یک فیلد است. کلید دسترسی به روابط ممکن است ترکیبی از چند فیلد باشد.

شماره کارمندی	نام	واحد	شغل	رتبه
۷۰-۱۲۵٦٤٥	فنی زاده	نقلیه	راننده	٤
٦٥=١٤٥٦٧٨	مزرعه	خدمات	باغبان	۲
۷۳-٦٦٦٦٦٦	مدیری	اداری	مدیر	١٤
۷۲-٦٥٤١۳۲	مظلومی	اداری	کارمند جز	١٠
۷۲-۲۰۲۰۲۰	حق پناه	خدمات	آبدارچی	١

تحلیل رابطه ای با رجوع به فرمها و داده های ورودی و خروجی سیستم آغاز می شود. با استفاده از روش نرمالیزه کردن داده ها، که در این بخش توضیح داده خواهد شد ، داده های زاید و قابل محاسبه حذف می شوند . داده های باقیمانده بطوری دسته بندی می شوند که بتوان از طریق ارتباط دادن دسته ها کلیه اطلاعات مورد نیاز را بدست آورد. هر یک از این دسته ها یک رابطه یا موجودیت می باشد.

نرمالیزه کردن

Normalize

نرمالیزه کردن ساختار داده ها

اصولاً ، کلمه نرمالیزه کردن به مفهوم تولید بر اساس استاندارد خاص است . در آنالیز رابطه ای با ساده کردن روابط بین خواص داده ها، رابطه های ساده یا در اصطلاح نرمالیزه ایجاد می کنند. روابط نرمالیزه شده بیانگر ساختار با حداقل فضای لازم جهت ذخیره داده های مرتبط هستند. در یک جدول یا فایل نرمالیزه یا به عبارت دیگر در یک جدول طبیعی ، ردیفها مستقل از یکدیگر هستند. این ویژگی ، موجب سهولت در عملیات بروز رسانی و حذف ردیفها یا رکورد ها از داخل جدول می شود. در صورت وابستگی رکوردها در هنگام انجام عمل حذف و یا بروز رسانی اطلاعاتِ یک رکورد ، باید مراقب تاثیرات این عملیات برروی رکوردهای وابسته با آن رکورد بود. این نکته ای است بسیار حائز اهمیت که باید مد نظر طراحان باشد. عمل نرمالیزه کردن برروی موجودیتها درسه مرحله معمولا" انجام می شود . در ادامه این بخش ، سه مرحله نرمالیزه کردن ، بطور ساده و عملی توصیح داده خواهد شد.

الف- مرحله یک نرمالیزه کردن

در مرحله یک نرمالیزه یا ساده کردن ، ابتدا خاصیت های تکراری در روابط تشخیص داده می شوند . خاصیت ها یا فیلدهای تشخیص داده شده از داخل رابطه مربوطه حذف می شوند. سپس این خاصیت های تکراری حذف شده ، در قالب رابطه جدیدی ظاهر می شوند. برای نمونه به برگ سفارش کالا که در شکل ٦,١٧ ارائه شده توجه کنید. ساختار این موجودیت به صورت زیر است :

سفارش : (کد سفارش ، کد مشتری ، نام مشتری ، آدرس مشتری ، تلفن مشتری ،
تاریخ دریافت سفارش ، { کد کالا ، نام کالا ، واحد ، تعداد سفارش })

در موجودیت سفارش اطلاعات اقلام کالای مورد سفارش تکراری است. لذا، این اطلاعات تکراری حذف شده ، موجودیت قلم سفارش ایجاد می شود.

سفارش : (کد سفارش ، کد مشتری ، نام مشتری ، آدرس مشتری ، تلفن مشتری ،
تاریخ دریافت سفارش)

قلم سفارش : (کدسفارش، کد کالا ، نام کالا ، واحد ، تعداد سفارش)

به همین ترتیب موجودیت صورتحساب با ساختار زیر را در نظر بگیرید.

صورتحساب : (کد صورتحساب، کدسفارش، کدمشتری، نام مشتری، آدرس مشتری ،
تلفن مشتری ، تاریخ صورتحساب ، { کد کالا ، نام کالا ، واحد ،
تعداد سفارش ، قیمت واحد ، قیمت کل } ، مبلغ کل صورتحساب).

در مرحله یک ساده سازی موجودیت صورت حساب به صورت زیر ساده می شود.

صورتحساب : (کد صورتحساب ، کدسفارش ، کدمشتری ، نام مشتری ،آدرس مشتری ، تلفن مشتری ، تاریخ صورتحساب ، مبلغ کل صورتحساب)

قلم صورتحساب : (کد صورتحساب ، کد کالا ، نام کالا ، واحد ، تعداد سفارش ، قیمت واحد، قیمت کل) .

ب- مرحله ۲ نرمالیزه کردن

در مرحله دوم ساده سازی ، باید خاصیت های غیرکلیدی را آزمود و مشخص کرد که کدامیک وابسته به بخشی از کلید ترکیبی است و به کل کلید وابسته نیست. پس از یافتن این خاصیت ها باید آنها را از داخل رابطه حذف نمود. برای نمونه ، در رابطه فلم سفارش که

دارای کلید ترکیبی است ، فیلدهای نام کالا و واحد کالا وابسته به کل کلید ترکیبی نیستند. این فیلدها وابسته به کد کالا هستند. کُد کالا بخشی از کلید ترکیبی برای قلم سفارش است . مشخصات یک کالا وابسته به کد کالا است و با هر سفارش جدید فرق نمی کند . پس ، باید مشخصات کالا را از قلم سفارش حذف نمود. با داشتن کد کالا می توان به فایل کالا مراجعه کرد و مشخصات کالا که شامل نام کالا و واحد شمارش آن است را از داخل فایل مربوطه بدست آورد. بنابراین ساختار موجودیت قلم کالا به صورت زیر ساده می شود.

قلم سفارش : (کدسفارش ، کد کالا ، تعداد سفارش)

به همین ترتیب می توان موجودیت قلم صورتحساب را ساده نمود.

قلم صورتحساب : (کد سفارش ، کد کالا ، تعداد سفارش ، قیمت واحد ، قیمت کل) .

ج- مرحله ۳ نرمالیزه کردن

در مرحله سوم ساده سازی ، فیلدها و یا به عبارت دیگر خاصیت های غیرکلیدی مورد بررسی قرار می گیرند. در طی این بررسی ، کلیه خاصیت های غیرکلیدی که از خاصیت های غیرکلیدی دیگر قابل دسترسی ، و یا محاسبه هستند را حذف می کنند. برای نمونه اکنون به کل موجودیتهای سیستم ساده سفارشات که در بالا مطرح شد توجه کنید :

- ♣ سفارش : (کد سفارش ، کد مشتری ، نام مشتری ، آدرس مشتری ، تلفن مشتری ، تاریخ دریافت سفارش)
- ♣ قلم سفارش : (کدسفارش، کد کالا ، تعداد سفارش)
- ♣ صورتحساب : (کد صورتحساب ، کدسفارش ، کدمشتری ، نام مشتری ، آدرس مشتری ، تلفن مشتری ، تاریخ صورتحساب ، مبلغ کل صورتحساب)
- ♣ قلم صورتحساب : (کد صورتحساب، کد کالا ، تعداد سفارش ، قیمت واحد ، قیمت کل)

با توجه به اینکه می توان با داشتن کد مشتری مشخصات وی را از فایل یا جدول مشتری بدست آورد ، فیلدهای نام مشتری ، آدرس مشتری و تلفن مشتری از داخل موجودیت های سفارش و صورتحساب حذف می شوند. به همین ترتیب ، در موجودیت قلم صورتحساب ، قیلد قیمت کل را می توان از فیلدهای تعداد سفارش و قیمت واحد محاسبه کرد. فیلد مبلغ کل صورتحساب در داخل رابطه صورتحساب نیز قابل محاسبه از قلم

صورتحساب است. نکته قابل توجه دیگر این است که تعـــداد کالای سفارش داده شده را می توان از موجودیت قلم سفارش بدست آورد لذا، این فیلد را از موجودیت قلم صورتحساب باید حذف نمود.

به این ترتیب موجودیتهای سیستم ساده سفارش کالا که در بخش قبل مطرح شد ، به صورت زیر بدون هیچگونه افزونگی ساده می شود.

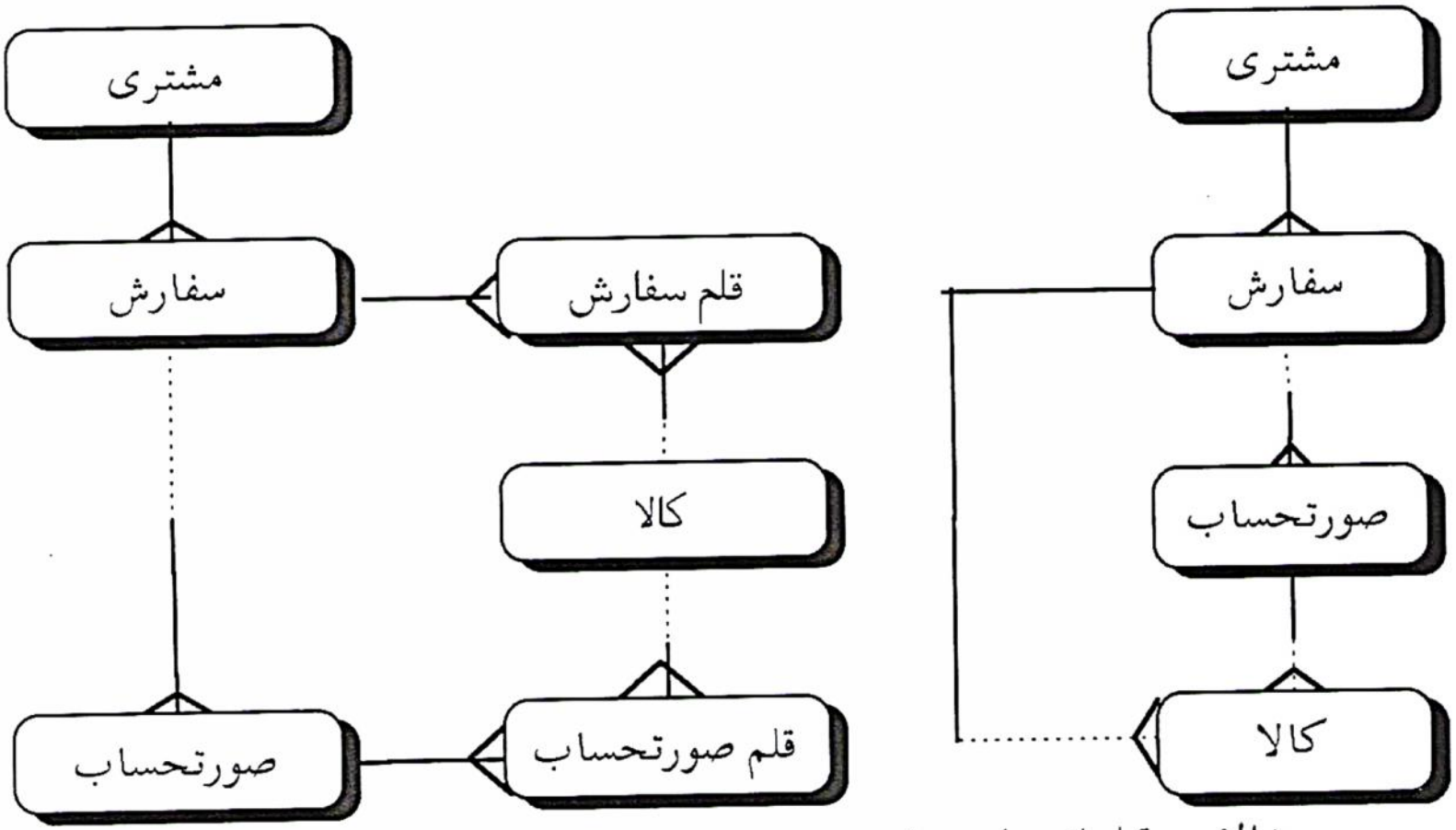
- مشتری : (کد مشتری ، نام ، تلفن ، آدرس ، شماره حساب بانکی)،
- سفارش : (کد سفارش ، کد مشتری ، تاریخ دریافت سفارش)
- قلم سفارش : (کدسفارش، کد کالا ، تعداد سفارش)
- صورتحساب : (کد صورتحساب ، کدسفارش ، کدمشتری ، تاریخ صورتحساب)،
- قلم صورتحساب : (کد صورتحساب، کد کالا ، قیمت واحد)
- کالا : (کد کالا ، نام کالا ، واحد ، تعداد ، نقطه سفارش ، مکان) .

در شکل بعد مدل منطقی داده ها قبل و بعد از عمل نرمالیزه کردن و ساده سازی ساختار رابطه ها ارائه شده است. مدل منطقی جدید با در نظر گرفتن فیلدهای موجودیتها ایجاد شده است. برای نمونه رابطه سفارش دارای خاصیتی به نام کد مشتری است. کد مشتری کلید اصلی برای دسترسی به رکورد مشتری است. لذا ، ارتباط بین مشتری و سفارش یک به چند است. به عبارت دیگر یک مشتری می تواند بیش از یک سفارش داشته باشد. کد سفارش بخشی از کلید اصلی رابطه قلم سفارش است. بنابراین ، یک کد سفارش خاص ممکن است بخشی از کلید دسترسی به چند رکورد قلم سفارش باشد. پس ، رابطه بین سفارش و قلم سفارش یک به چند است.

باید توجه نمایید که هدف اصلی از مراحل نرمالیزه کردن ، کم کردن تعداد فیلدها و بالنتیجه کاهش اندازه رکورد فایلها است. البته این به بهای افزایش تعداد فایلها و اغلب طولانی شدن زمان دسترسی به اطلاعات تمام می شود. لذا ، در مراحل پیاده سازی سعی می شود که با کاهش تعداد فایلها یا در اصطلاح جداول و اضافه کردن فیلدها سرعت دسترسی به اطلاعات افزایش داده شود. در واقع برای افزایش سرعت دسترسی به اطلاعات ، برخلاف مراحل نرمالیزه کردن عمل می کنند.

در بخشهای قبلی نمونه هایی بسیار ساده از مدل منطقی داده ها برای یک سیستم ساده سفارشات مطرح شد . در بخش بعد چند نکته جهت ترسیم بهینه مدل منطقی داده ها برای

سیستم های بزرگ و آزمون صحت مدل ارائه شده است.



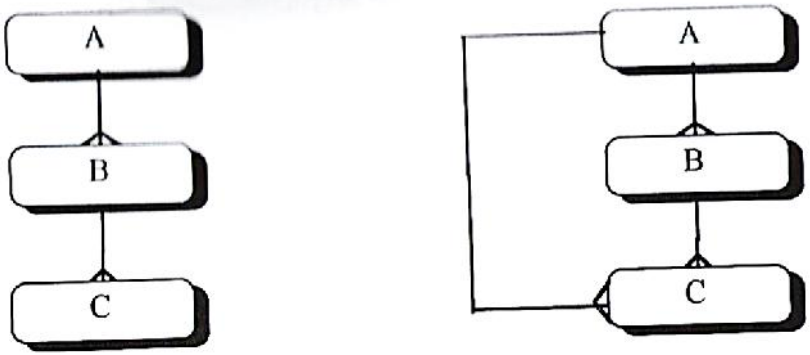
(الف- قبل از ساده سازی) (ب- پس از ساده سازی)

مدل منطقی داده ها قبل و پس از ساده سازی

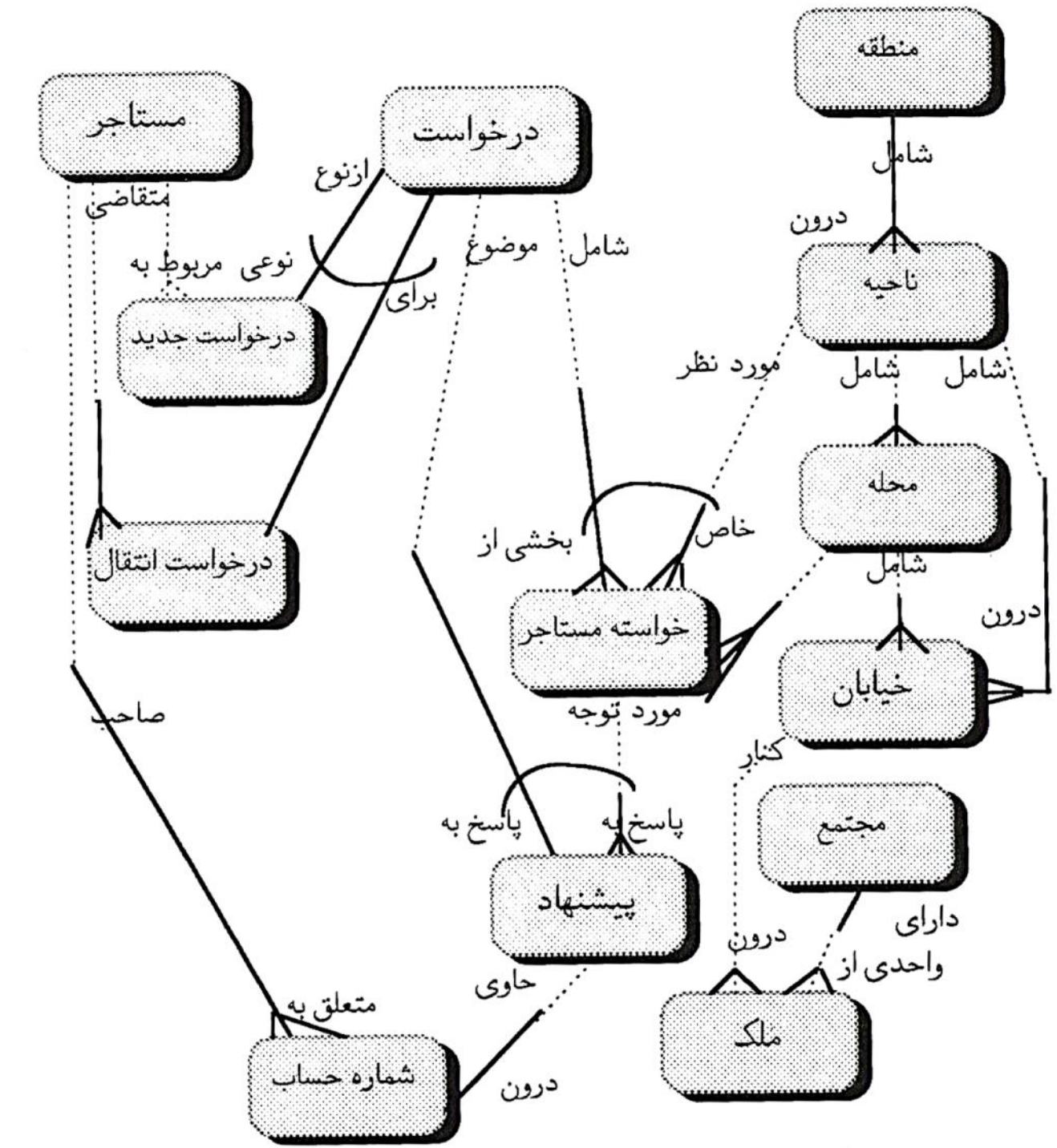
ترسیم بهینه

در هنگام ترسیم مدل منطقی داده ها بهتر است که موجودیتهای کل که در ارتباط چند به یک با موجودیتی دیگر نمی باشند را در بالای صفحه قرار داد. رابطه سلسله مراتبی موجودیت ها را با ترسیم آنها از بالا به پائین صغحه می توان نشان داد. البته در هنگام ترسیم خطوط ارتباط دهنده موجودیت ها ، باید تلاش نمود تا حتی المقدور خطوط یکدیگر را قطع ننمایند. شکل قبل نمایانگر مدل منطقی داده ها برای یک سیستم معاملات ملکی است.

باید از ایجاد روابط زائد بین موجودیتهاخودداری شود. برای نمونه در شکل ٦,١٨ اگر ارتباط مستقیم بین A و C را بتوان از طریق B بطور غیر مستقیم برقرار کرد ، بهتر است که این رابطه مستقیم را جهت سادگی مدل حذف نمود. البته در طرح فیزیکی برای پیاده سازی سیستم کامپیوتری ممکن است بخاطر افزایش سرعت دسترسی به اطلاعات دو مرتبه ارتباط مستقیم برقرار شود.



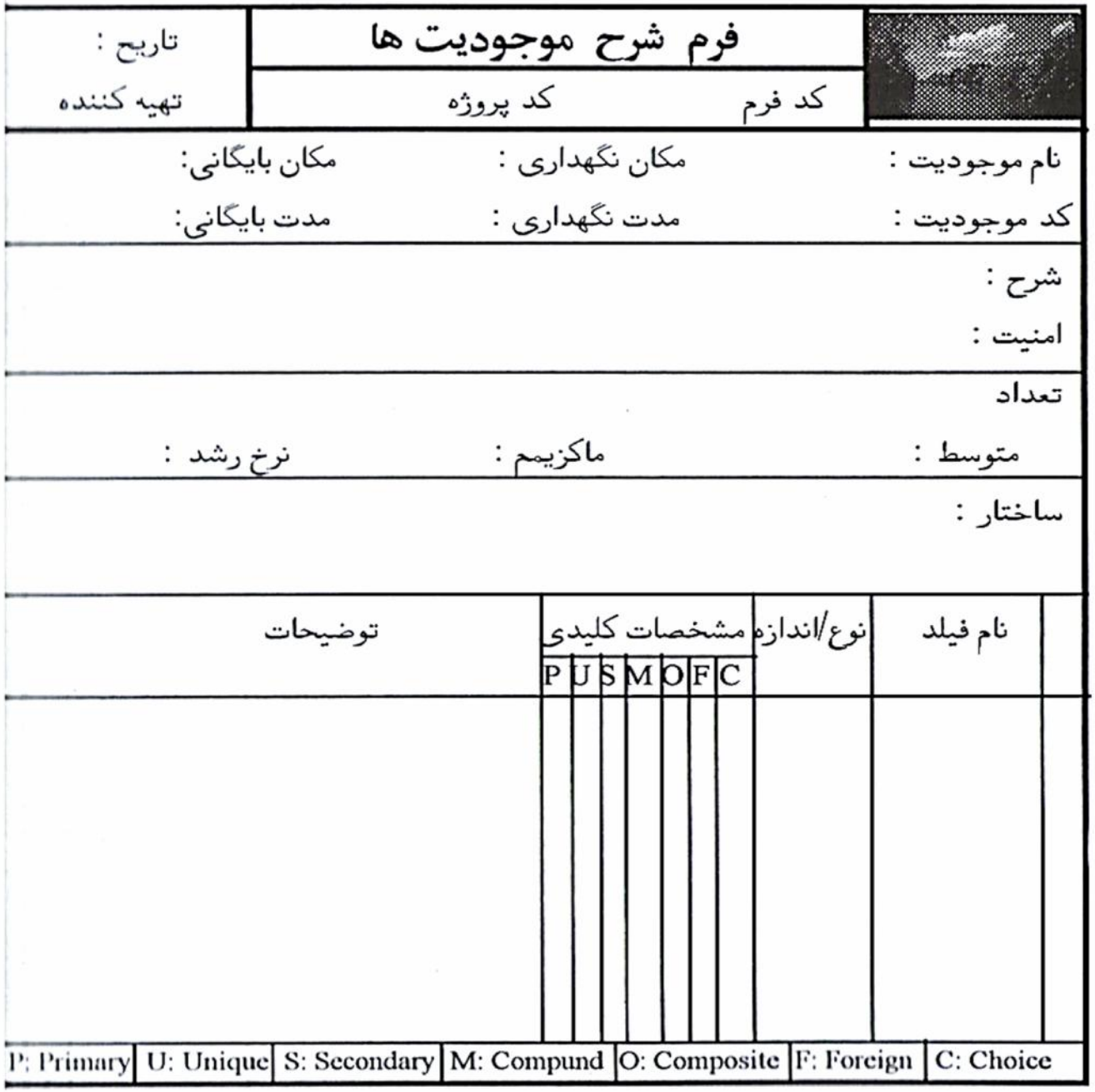
حذف رابطه زائد


منطقه
شامل
درون
ناحیه
شامل
شامل
مورد نظر
محله
شامل
درون
خیابان
کنار
مجتمع
دارای
واحدی از
درون
ملک
مستاجر
متقاضی
درخواست
ازنوع
نوعی
مربوط به
برای
درخواست جدید
درخواست انتقال
صاحب
موضوع
شامل
خاص
بخشی از
خواسته مستاجر
مورد توجه
پاسخ به
پاسخ به
پیشنهاد
حاوی
متعلق به
شماره حساب
درون

اصولاً در صورتی از صحت یک مدل منطقی داده ها می توان اطمینان حاصل نمود که اولا" پردازه هایی درون دیاگرام گردش داده ها برای ایجاد ، حذف و اصلاح موجودیت های آن مدل داده ها وجود داشته باشد . ثانیا"، بتوان اطمینان حاصل نمود که اگر یک موجودیت درون یک انباره قرار دارد درون انباره دیگری تعریف وجود نداشته ، فقط در یک انباره تعریف شده باشد. برای نمایش هر انباره بخشی از مدل منطقی داده های سیستم مشخص کننده ساختار آن انباره است را بطور جداگانه مشخص می کنند.

کار در کلاس ۸:

مدل منطقی سیستم تولید پوشاک را نرمالیزه کنید (۳ مرحله).



	فرم شرح موجودیت ها	تاریخ :
	کد فرم کد پروژه	تهیه کننده

نام موجودیت : مکان نگهداری : مکان بایگانی:

کد موجودیت : مدت نگهداری : مدت بایگانی:

شرح :

امنیت :

تعداد

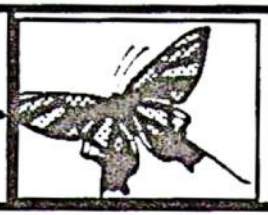
متوسط : ماکزیمم : نرخ رشد :

ساختار :

نام فیلد	نوع/اندازه	مشخصات کلیدی							توضیحات
		P	U	S	M	O	F	C	

P: Primary | U: Unique | S: Secondary | M: Compund | O: Composite | F: Foreign | C: Choice

فرم شرح موجودیتها

شرکت راه نور 	فرم : شرح فرم های ورودی / خروجی	تاریخ تهیه :
	کد فرم :	کد پروژه :	تهیه کننده

نام فرم : ..... ترتیب : . نرخ رشد :.. مدت نگهداری : .....

کد فرم : .... صفحه .. از .... کلید: ..... انباره ..... مدت بایگانی : .....

نام فیلد	برچسب	نوع /اندازه	تولید کننده	توضیحات

فرم کلی برای شرح فیلدهای فرم های ورودی / خروجی

طراحی واسط کاربر

طراحی واسط کاربر (User Interface)

برای طراحی واسط کاربر از نمودار جکسون استفاده می‌شود. نمودار جکسون براساس واکنش موجودیتها در مقابل رخدادها (دیدگاه رفتاری) ترسیم می‌شود. به نمودار جکسون نمودار ساختار ورودی/خروجی نیز گفته می‌شود.

این نمودار توالی، اختیاری بودن و تکرار داده‌هایی که از طریق واسط کاربر از محدوده سیستم عبور می کند را نشان می دهد.

علائم

نمودار جکسون بصورت یک درخت رسم می‌شود که ریشه درخت بصورت یک مستطیل نمایش داده می‌شود و نام ساختار ورودی/خروجی یا واسط کاربر را معرفی می کند.

برگهای درخت که بوسیله مستطیل مشخص می شوند اقلام داده‌ای را مشخص می کنند. هر برگ می‌تواند ورودی یا خروجی باشد.

مستطیلهایی که بین ریشه و برگها قرار می گیرند چگونگی گروه‌بندی و سازماندهی اقلام داده‌ای را در واسط کاربر مشخص می کنند. به‌این مستطیلها، مستطیل ساختاری می‌گویند.

توالی Sequence :

توالی زمانی مستطیلها از چپ به راست است.

گزینش Selection :

از بین چند مستطیل فقط یکی انتخاب می‌شود.برای مشخص کردن گزینش از علامت o در گوشه بالا-راست مستطیل استفاده می‌شود.

تکرار Iteration :

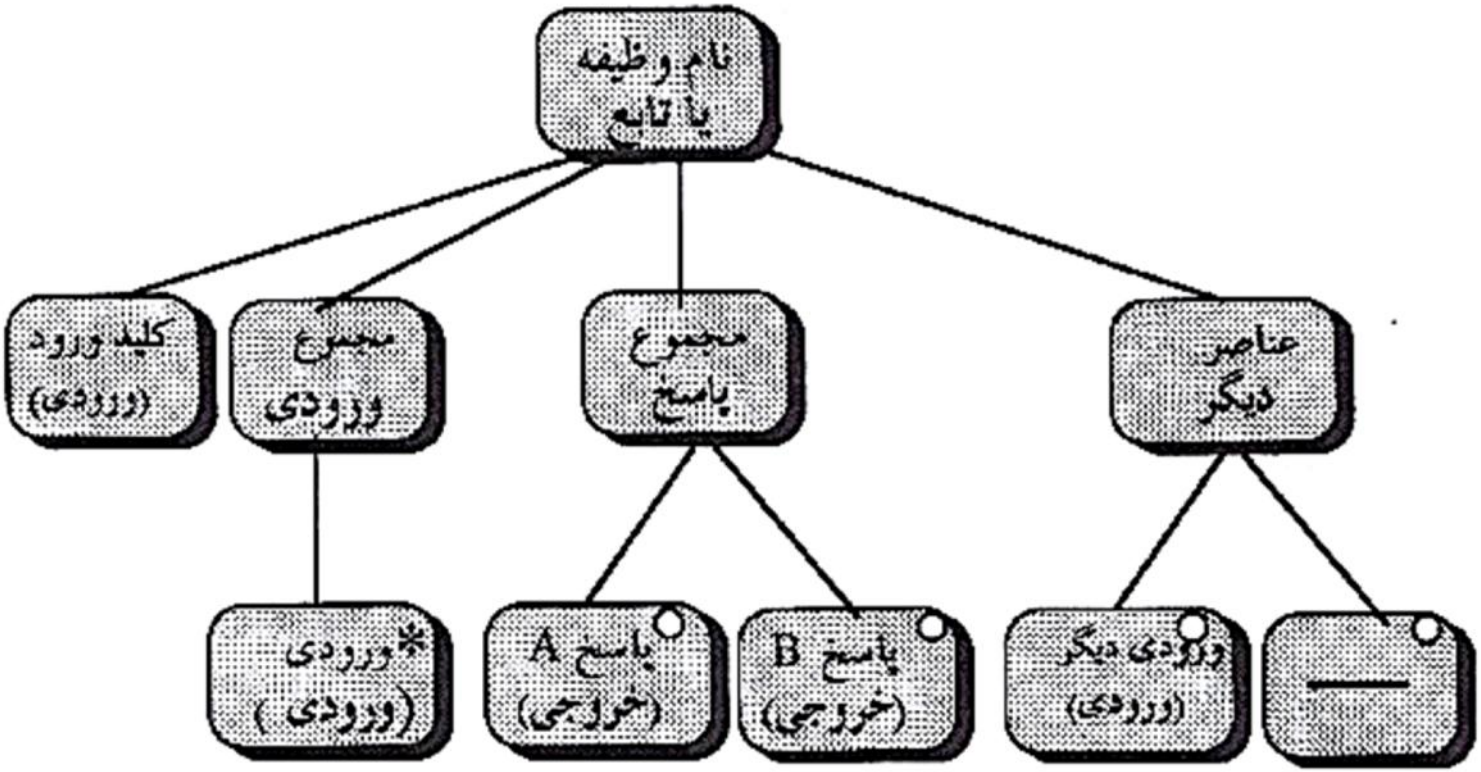
از علامت * در گوشه بالا-راست مستطیل استفاده می‌شود.یعنی اقلام داده‌ای داخل مستطیل چند بار تکرار خواهند شد.

قواعد رسم نمودار جکسون

۱- همه مستطیلهایی که در زیر یک مستطیل والد(پدر) قرار دارند باید از یک نوع باشند.
۲- مستطیلهای تکرار باید یک مستطیل توالی یا انتخاب به عنوان پدر داشته باشند.
۳- مستطیلهای انتخاب (گزینش) باید یک مستطیل ساختاری (میانی) به عنوان پد داشته باشند.

مثال: طبق قانون اول نمودار زیر اشتباه است.

مثال:نمونه‌ای کلی از ساختار ورودی/خروجی توابع



نمونه ای کلی از ساختار ورودی / خروجی توابع

فرهنگ داده‌ای(Data Dictionary)

فرهنگ داده‌ای لیستی از تمام عناصر داده است که جزئیات داده‌ها را مشخص می کند.

فرهنگ داده‌ای برای هر داده از قسمتهای زیر تشکیل می‌شود:

Name : نام داده (اطلاعات)

Alias : اسامی مستعار

Where/how used : داده کجا و چطور استفاده می‌شود.(input/output)

Content Description : شرح محتویات

Supplementary Information : اطلاعات تکمیلی

علائم جهت نمایش محتویات

=	ترکیبی است از
+	ترتیب
[|]	انتخاب
$\{\}^n$	n بار تکرار
()	داده اختیاری
.....	توضیح

مثال: فرهنگ داده‌ای برای شماره تلفن

Name :Telephone Number

Alias : none

Where/How used : dial phone (input)

Content Description :

Telephone Number =[Local number| Long distance number]

Local number =(prefix)+access number

Long distance number =0+area code+Local number

Area code =[111|311|411|761]

Prefix = * a 3 digit number that never starts with 0 or 1*

Access number = * any 4 digit number*

مثال:شماره دانشجویی

Name : Student number

Alias : Id

Where/how used : sign up(input)

Content description :

Id = [Normal|Guest]

Normal = year+coursecode+sequential number

Guest= year+coursecode+sequentialnumber+m

Course code=[11|12|13|14]

Year = *any 4 digit number*

Sequential number = * any 3 digit number*