پایگاه داده عبارتست از مجموعه ای از داده ها یا مجموعه ای از داده های منطقا بهم مرتبط که برای پاسخگویی به نیازهای اطلاعاتی یک سازمان طراحی شده است و مجموعه ای ازداده های پایا که در سیستم های کاربردی در یک سازمان مورد استفاده قرار می گیرد.

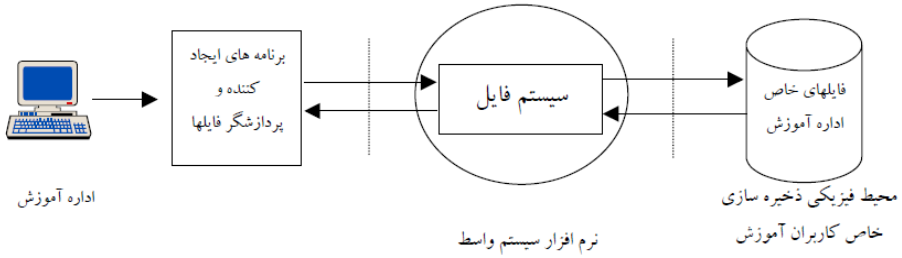
به طور کلی می توان گفت بانک اطلاعاتی مجموعه ای داده های ذخیره شده در یک محل مجتمع به صورت یکپارچه با مدیریت متمرکز دارای حداقل افزونگی در یک ساختار سوری با قابلیت استفاده یک یا چند کاربر به صورت همزمان.

محیط عملیاتی دانشگاه را در نظر بگیرید که دارای بخش های عملیاتی مختلف می باشد . فرض می شود که سه بخش امور آموزش، امور دانشجویی و امور مالی دانشگاه بخشهایی هستند که می خواهیم برای آنها یک سیستم ذخیره و بازیابی کامپیوتری ایجاد نماییم و نیز فرض می کنیم که تنها نوع موجودیت مورد نظر، موجودیت دانشجو باشد و بخشهای فوق می خواهند اطلاعاتی را در مورد این نوع موجودیت داشته باشند . واضح است که در هر یک از بخشهای فوق انواع دیگری از موجودیتها وجود دارند که در این مثال مورد بحث قرار نمی گیرند.

دو روش و مشی کلی در طراحی این سیستم وجود دارد:

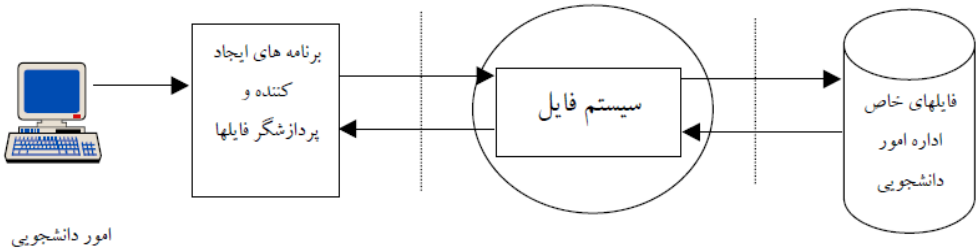
۱. **مشی غیر بانکی**

در این روش هر یک از زیر محیط های عملیاتی مستقلا مطالعه می شود و برای هر زیر مجموعه یک سیستم خاص همان زیر محیط طراحی و تولید می شود، بگونه ای که فقط پاسخگوی همان زیر محیط است. با توجه به مثال مطرح شده هر قسمت از دانشگاه سیستم کاربردی خاص و جداگانه خود را خواهد داشت.



- قالب رکورد از دید آموزش : (دانشکده، سال ورود، تاریخ تولد، نام خانوادگی، نام، شماره دانشجو)

امور دانشجویی نیز فایلهای خاص خود را دارد:



- قالب رکورد از دید امور دانشجویی:

(سال ورود، تاریخ تولد، نام خانوادگی، نام، شماره دانشجو)

مشخصه های این روش عبارتند از:

۱ – در روش فایل پردازی داده ها از هم مجزا می باشند.

۲ – محیط ذخیره سازی نامجتمع است (تعدادی سیستم جداگانه و محیط ذخیره سازی جداگانه)

۳ – برنامه های کاربردی وابسته به قالب فایل می باشند.

٤ – عدم سازگاری در داده ها و فایلها دیده می شود.

٥ – اشتراکی نبودن داده ها : داده های زیر محیط ۱ مورد استفاده کاربران زیر محیط ۲ نمی توانند قرار گیرند.

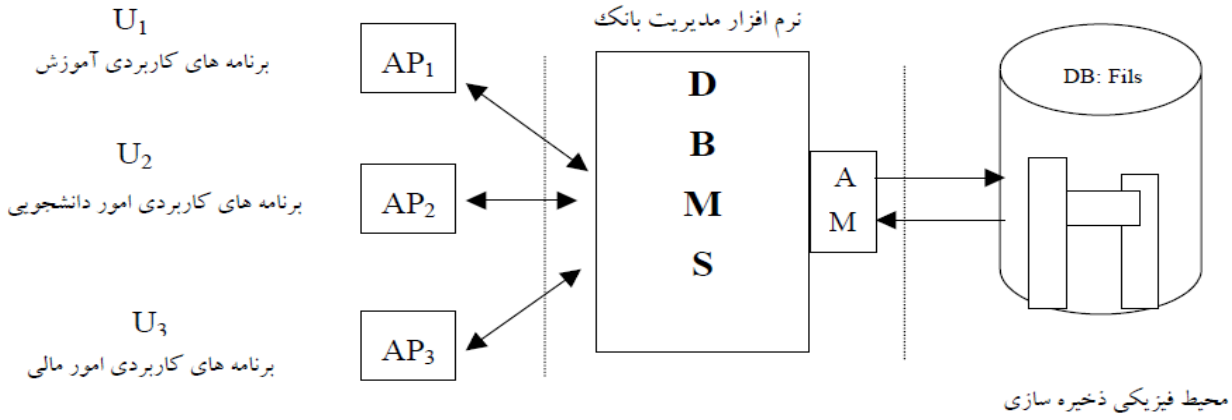
٦ – اطلاعات تکراری و افزونگی در داده ها وجود دارد.

۷ – عدم امکان استانداردهای واحد محیط عملیاتی بدلیل وجود سیستم های متعدد و پراکنده که احیاناً توسط تیم های مختلف طراحی و پیاده سازی شده است امکان اعمال یکسری از عملیات منسجم روی آن سیستم وجود ندارد.

ب – مشی بانکی (پایگاهی) Database Approach

در این روش یک تیم واحد طراحی و پیاده سازی به سرپرستی متخصصی به نام DBA مجموعه نیازهای اطلاعاتی کل محیط عملیاتی مورد نظر مدیریت کل سازمان را بررسی می کند و با توجه به نیازهای اطلاعاتی تمام کاربران محیط و ضمن استفاده از یک نرم افزار خاص به نام DBMS محیط واحد و مجتمع ذخیره سازی اطلاعات ایجاد می شود.با توجه به مثال مطرح شده، رکورد نوع دانشجو فقط یکبار در فایل ذخیره می شود وکاربران مختلف هر یک طبق نیاز اطلاعاتی خود از آن بطور مشترک استفاده می نمایند. در رکورد نوع دانشجو، تمام صفات خاصه مورد نیاز کاربران مختلف وجود دارند و صفات خاصه مشترک، تنها یکبار در رکورد منظور می شوند.

در این روش هر کاربری، دید خاص خود را نسبت به داده های ذخیره شده در بانک دارد. دید کاربران مختلف از یکدیگر متفاوت و حتی گاه با هم متضاد است.



مشخصه های این روش :

۱ – داده هـای مجتمع: کـل داده هـا بصورت یک بانک مجتمع دیده می شوند و از طریق DBMS با آنها ارتباط برقرار می شود.

۲ – عـدم وابستگی برنامه های کاربردی به داده ها و فایلها: زیرا DBMS خود به مسائل فایلینگ می پرازد و کاربران در محیط انتزاعی هستند.

۳ – تعدد شیوه های دستیابی به داده ها

٤ – عدم وجود ناسازگاری در داده ها

٥ – اشتراکی بودن داده ها

٦ – امکان ترمیم داده ها

۷ – کاهش افزونگی

۸ – کاهش زمان تولید سیستم ها

۹ – امکان اعمال ضوابط دقیق ایمنی

۲-۱- مقدمه :

یـک مـدل داده ای مجمـوعه ای از ابـزارهای مفهومـی بـرای توصـیف داده ها ، ارتباط بین داده ها،معانی داده ها و محـدودیت هـای آنهاسـت. بعبارتـی یـک روش تفکـر دربـاره داده ها که به پیاده سازی ربطی ندارد.در یک نگاه کلی میتوان انواع مدلهای داده ای را بصورت زیر نام برد :

- مدلهای مبتنی بر اشیاء Object based Logical Models

داده هـا بصورت مجموعه ای از موجودیت ها که نمایش اشیاء در دنیای واقعی میباشند دیده می شوند.از جمله این مدلها مدل داده ای E/Rو مدل شیءگرایی را می توان نام برد.

- مدلهای مبتنی بر رکورد Record based Logical Models

داده ها در قالب رکوردهای ثابت و یا با طول متغییر دیده میشوند.

از انواع دیگر مدلهای داده ای می توان به مدل های داده ای شبه ساختیافته ، مدلهای شبکه ای و مدلهای سلسله مراتبی اشاره نمود.

ازجملـه مـدلهای منطقـی مبتنـی بـر اشـیا می توان **مدل E/R** را نام برد. در سال ۱۹۷٦ یک دانشجوی دکترای کامپیوتر

دانشگاه MIT به نام چن (chen) مدلی برای طراحی بانک اطلاعاتی پیشنهاد کرد که مورد توجه عام واقع شد. وی مدل خـود را E / R نامـید. ایـن مـدل در طول زمان پیشرفت کرد و ساختارهای جدیدی به آن افزوده شد. در مدل E / R هر پایگاه داده دارای دو بخش پدیده یا موجودیت و ارتباط می باشد. Chen نه تنها مدل E / R را معرفی نمود بلکه نمودار موجـودیت رابطه را نیز مطرح ساخت. نمودار E / R روشی برای نمایش ساختاری منطقی یک بانک اطلاعاتی به روش تصـویری است. این نمودارها ابزارهایی راحت و مناسب را برای درک ارتباطات مابین موجودیها فراهم می کنند. (یک تصویر گویا تر از هزار کلمه است.)

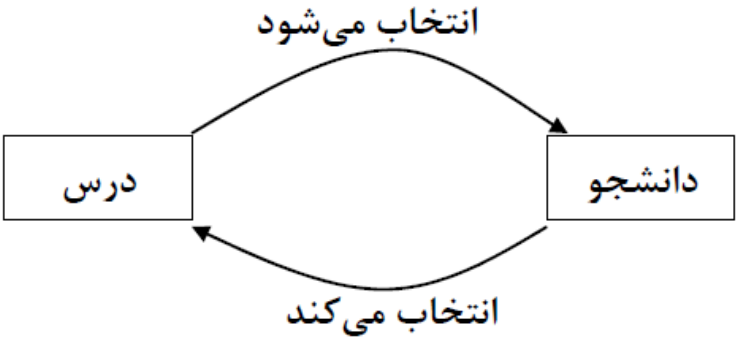
در واقع، شهرت و محبوبیت مدل E / R به عنوان روشی برای طراحی بانک اطلاعاتی احتمالاً به روش رسم نمودارهای E / R مربوط می شد تا به جنبه های دیگر آن.

۲ – ۲ – نمایش نموداری E / R

مـدل E/R بکمـک نمـوداری تحـت نـام خـود قابـل نمایش است که در این نمودار مفاهیم مربوطه با اشکال مشخصی ترسیم میگردند.در ادامه هر یک از این اشکال نشان داده شده اند.

این نمودار نمایشگر ارتباط بین موجودیتهای یک محیط عملیاتی است و به کمک آن داده‌های موجود مدل‌بندی می‌شوند. مثلاً محیط عملیاتی دانشگاه یک مجموعه از انواع موجودیتهای : استاد، دانشجو، کلاس، درس، دانشکده و گروه آموزشی می‌باشد.

مثال ۱ : ارتباط دو موجودیت دانشجو و درس می‌تواند به صورت زیر باشد :



۲-۲-۱- موجودیت:

موجودیت، چیزی است که بصورت متمایز قابل شناسایی باشد. آقای چن موجودیتها را به دو دسته منظم (قوی) و ضعیف دسته بندی کرد.

۱ – **موجودیت منظم(قوی):** موجودیتی است که وجودش وابسته به موجودیت دیگری نیست. مثل موجودیت دانشجو و موجودیت درس که هریک به تنهایی در محیط عملیاتی دانشکده مطرح میباشند.

۲ – **موجودیت ضعیف:** موجودیتی است که وجودش وابسته به موجودیت دیگر است. بطور مثال موجودیت اعضاء خانواده کارمند وابسته به موجودیت کارمند می باشد. و یا موجودیت آثار منتشره استاد، موجودیت ضعیف موجودیت استاد است.

در نمودار E / R موجودیتهای قوی بصورت یک مستطیل نشان داده می شوند که محتوای آن در بر گیرنده نام نوع موجودیت مورد نظر است. در موجودیتهای ضعیف مرز مستطیل بصورت دو خطی است.

۲-۲-۲-: صفات خاصه Attributes

در نمودار E/R صفات خاصه بصورت بیضی نشان داده می شوند که محتوای آن اسم صفت خاصه مورد نظر را در بر می گیرد و به وسیله خطوط تو پر به موجودیت یا رابطه مربوط متصل می شوند.

در یک تقسیم بندی صفات خاصه را می توان به شرح زیر تقسیم بندی نمود:

الف – صفت خاصه کلید(شناسه موجودیت):

یک یا چند صفت خاصه که در یک موجودیت منحصر به فرد است.

مثال: نوع موجودیت: دانشجو : صفت خاصه کلید: شماره دانشجو

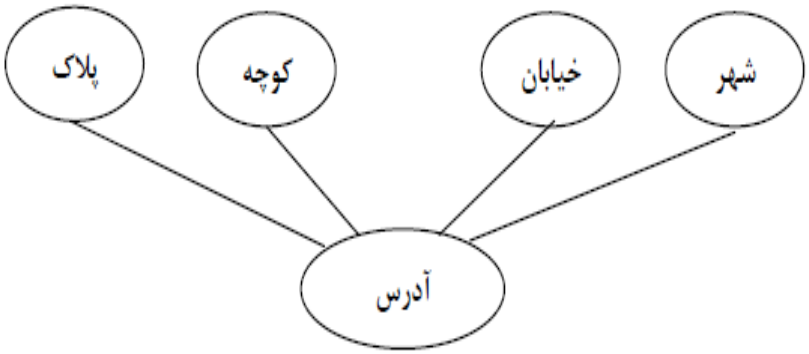
نوع موجودیت: گروه درسی : صفت خاصه کلید: شماره درس، شماره گروه و ترم

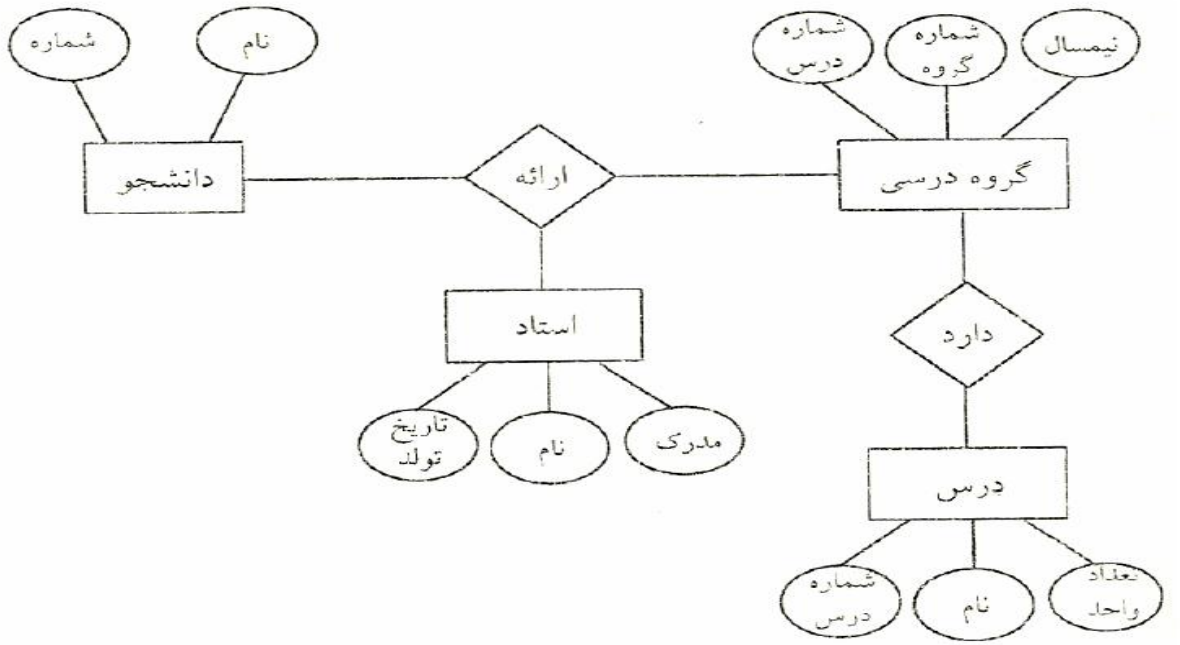
برای مشخص کردن کلید در یک موجودیت زیر صفت یا صفات خاصه کلید خط کشیده می شود.

ب – صفت خاصه ساده / مرکب:

- صفت خاصه ساده صفتی است که به اجزاء کوچکتر تجزیه پذیر نباشد.
- صفت خاصه مرکب صفتی است که به اجزاء کوچکتر تجزیه پذیر باشد.

بعنوان مثال: صفت خاصه آدرس می تواند به قسمتهایی نظیر شهر، کوچه، خیابان ... تقسیم شود.

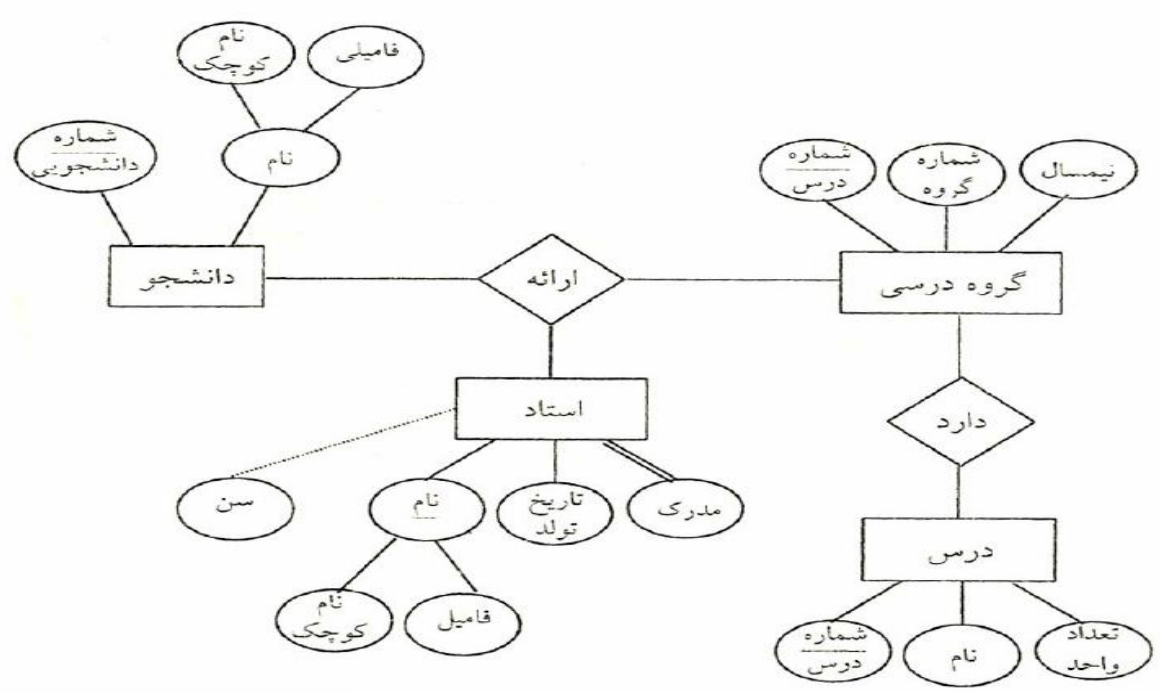




کلید عبارت است از یک یا چند صفت که در یک موجودیت منحصر به فرد باشد. مثلاً در موجودیت دانشجو شماره دانشجویی کلید است. چون هر دانشجو یک شماره یکتا دارد. ولی نام نمی‌تواند کلید باشد. گاهی اوقات یک صفت تنها نمی‌تواند کلید باشد بلکه مجموعه‌ای از دو یا چند صفت با همدیگر کلید می‌شوند. مثلاً نام و شماره شناسنامه هر یک به تنهایی کلید نیست ولی هر دو با هم کلید می‌شوند. برای مشخص کردن کلید یک موجودیت زیر آن صفت خط می‌کشیم. (...، نام پدر، نام، شماره دانشجو)

بعضی از صفتها ساده هستند مثل شماره دانشجویی ولی بعضی از صفتها مرکب (تجزیه پذیر) هستند مثل آدرس که خود از صفتهای شهر، خیابان، کوچه و پلاک تشکیل یافته است. در واقع صفت مرکب صفتی است که هم خودش معنی‌دار است و هم بخشهایی از آن در بانک اطلاعاتی رابطه‌ای (جدولی) صفت مرکب نداریم.

در نمودار EER صفات ترکیبی را در دو سطح می‌کشیم.



می‌توان اجزاء صفات مرکب را داخل پرانتز نوشت.

مثل : «(نام کوچک و فامیلی) نام و نام دانشجویی» دانشجو

مثلاً در موجودیت استاد نام تک مقداری است چون هر استاد فقط یک نام دارد ولی صفت مدرک چند مقداری است چون استاد ممکن است چندین مدرک داشته باشد. صفتهای چند مقداری را در مدل EER با دو خط ترسیم می‌کنیم. در مدل رابطه‌ای صفت چند مقداری نداریم.

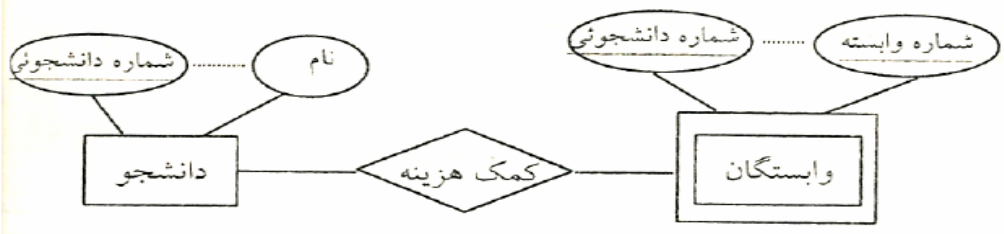
صفت مشتق صفتی است که به کمک صفتهای دیگر می‌توان آن را محاسبه کرد. مثلاً سن استاد یک صفت مشتق است که با توجه به تاریخ تولد قابل محاسبه می‌باشد. تصمیم‌گیری در مورد صفت مشتق به عهده طراح است مثلاً معدل کل برای دانشجو بهتر است مشتق باشد زیرا مرتباً با گذراندن دروس بیشتر عوض می‌شود ولی برای فارغ‌التحصیلان معدل کل بهتر است بخشی از پدیده باشد.

ممکن است وجود یک پدیده وابسته به وجود پدیده دیگری باشد یعنی در صورت حذف عضوی از آن پدیده، عضوهای وابسته هم لازم باشد به طور خودکار حذف شوند. مثلاً به محض حـذف دانشجو از بانک دانشگاه (مثلاً بر اثر فارغ‌التحصیلی یا اخراج شدن) وابستگان او نیز (مثل همسر و فرزند) از سیستم کمک هزینه باید حذف شوند.

این نوع وابستگی را، وابستگی وجودی و پدیده وابسته را موجودیـت ضـعیف (Weak Entity) می‌نامند. پدیده وابسته باید کلید پدیده اصلی را که بـه آن وابـسته اسـت بـه ارث ببـرد تـا بـه سادگی قابل شناسایی باشد. پدیده وابسته را با دو مستطیل تو در تو نمایش می‌دهیم.

مانند شکل زیر.

مثال ۲۰ :

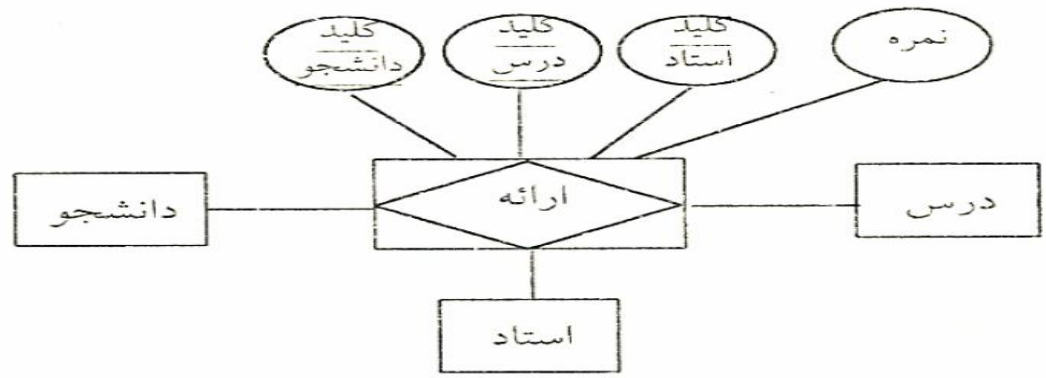


ارتباط‌ها نیز می‌توانند صفت داشته باشند. مثلاً نمره در نمودار بانک اطلاعات دانشگاه می‌تواند صفت ارتباط ارائه باشد. شاید تصور شود که نمره مربوط به پدیده دانشجو یا درس است.

ولی این تصور غلط است زیرا یک دانشجو چند نمره (در دروس مختلف) و یک درس نیز چند نمره (برای دانشجویان مختلف) دارد.

از طرف دیگر ممکن است دانشجویی در درسی از یک استاد نمره ۷ و در ترم بعد از استاد دیگری نمره ۱۴ گرفته باشد بنابراین صفت نمره را باید به ارتباط ارائه که سه پدیده دانشجو ـ درس و استاد را به هم مرتبط می‌کند نسبت داد.

چنین ارتباط‌هایی با یک لوزی درون مستطیل نشان داده می‌شوند و کلید آنها کلیدهای همه پدیده‌های مربوطه را شامل می‌شود مانند شکل زیر :



نکته : تصمیم گیری در مورد صفت مشتق در یک موجودیت بعهده طراح است.

۵- صفت خاصه هیچمقدار پذیر :

هیچمقدار یعنی یک مقدار ناشناخته و یا مقدار غیر قابل اعمال .اگر مقدار یک صفت در یک یا بیش از یک نمونه از یک نوع موجودیت برابر هیچمقدار باشد آن صفت خاصه هیچمقدار پذیر است.

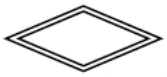
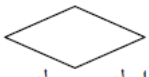
- مثال : شماره تلفن یک نمونه استاد ممکن است در دست نباشد.
- نام استاد در یک برنامه درسی ترم ممکن است هنوز اعلام نشده باشد.

توجه : در نمودار E/R این خاصیت نشان داده نمی شود.

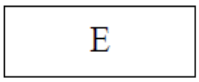
۲-۲-۳-: ارتباط:

هر ارتباط بصورت یک لوزی نشان داده می شود که محتوای آن در بر گیرنده نام نوع رابطه مورد نظر است.

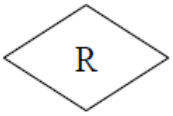
در نمودار ER ارتباط با موجودیت ضعیف بصورت لوزی دو خطی نشان داده می شود.

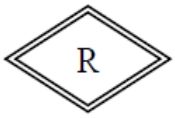
ارتباط با موجودیت ضعیف  ارتباط 

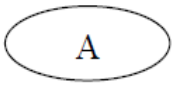
عنصرهای هر رابطه (شامل صفات خاصه و موجودیت ها) بوسیله خطوط پر به رابطه مربوطه وصل می شوند. هر خط ارتباطی بین موجودیت و رابطه دارای برچسب ۱ یا n می باشد که ماهیت ارتباط را مشخص می کند.

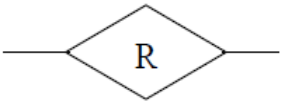
نوع موجودیت 

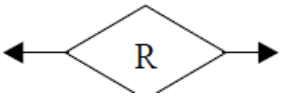
نوع موجودیت ضعیف 

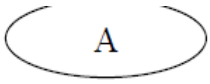
نوع رابطه 

رابطه بین موجودیتها و موجودیت ضعیف 

کلید اصلی 

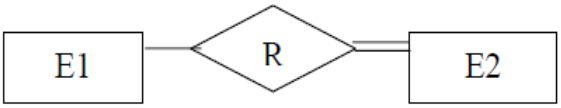
رابطه چند به چند 

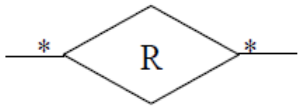
رابطه یک به یک 

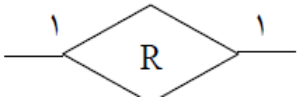
صفت خاصه 

صفت خاصه چند مقداری 

صفت خاصه مشتق 

شرکت کامل (الزامی)موجودیت در رابطه 





محیط بانک اطلاعاتی از عناصر اصلی زیر تشکیل شده است :

۱ـ سخت‌افزار ۲ـ نرم‌افزار ۳ـ کاربر ۴ـ داده‌ها

۱ـ سخت‌افزار محیط بانکی را می‌توان به صورت زیر تقسیم‌بندی کرد:

الف) سخت‌افزار ذخیره‌سازی داده‌ها

ب) سخت‌افزار پردازنده مرکزی

ج) سخت‌افزار ارتباطی

۲ـ نرم‌افزار محیط بانکی را می‌توان به دو دسته الف) نرم‌افزار کاربردی ب) نرم‌افزار سیستمی تقسیم‌بندی کرد.

نرم‌افزار کاربردی : نرم‌افزاری است که کاربر باید برای تماس با سیستم بانک اطلاعاتی آماده کند. این نرم‌افزار به کمک یک زبان سطح بالا و یک زبان داده‌یی (Data Language) و برخی تسهیلات نرم‌افزاری برای تماس با بانک ساخته می‌شود.

نرم‌افزار سیستمی : که از نرم‌افزار سیستمی خاص بانک (یعنی DBMS) و نرم‌افزار سیستمی عمومی (یعنی سیستم عامل) تشکیل شده است. DBMS در یک تعریف مقدماتی، سیستمی است که به کاربران امکان می‌دهد عملیات موردنظرشان را (مثل تعریف داده‌ها ـ بازیابی داده‌ها و ذخیره‌سازی داده‌ها) انجام دهند. DBMS که نرم‌افزاری پیچیده است میهمان یک سیستم عامل است و از امکانات سیستم عامل در انجام وظایفش استفاده می‌کند.

۳ـ کاربر : یکی از کاربران مهم در سیستم بانک اطلاعاتی، اداره کننده بانک اطلاعاتی Data Base Administrator (DBA) است. اداره کننده بانک فردی است که مسئولیت ایجاد، پیاده‌سازی و نگهداری بانک را در محیط عملیاتی بر عهده دارد.

کاربر دیگر برنامه‌نویس یا DBP می‌باشد. کاربر نهایی (End User) فردی است که از برنامه‌های نوشته شده استفاده می‌کند.

۴ـ داده : منظور از داده در اینجا داده‌هایی است که در مورد موجودیتهای مختلف محیط عملیاتی، می‌خواهیم ذخیره کنیم و نیز ارتباط بین انواع موجودیتها و اصطلاحاً به آن «داده‌های عملیاتی» می‌گوئیم.

مزایا و محاسن سیستم بانک اطلاعاتی که دلایل ایجاد آن نیز به شمار می‌آید عبارتند از :

۱ـ مولینگ داده‌های عملیاتی براساس سمانتیک آنها

۲ـ وحدت ذخیره‌سازی کل داده‌های محیط عملیاتی

۳ـ اشتراکی شدن داده‌ها

۴ـ کاهش میزان افزونگی

۵ـ تعدد شیوه‌های دستیابی و تسهیل دستیابی به داده‌ها

۶ـ عدم وجود ناسازگاری در داده‌ها

۷ـ تامین سیستم کاراتر برای ذخیره و بازیابی

۸ـ تضمین جامعیت، بی‌نقصی و دقت (Accuracy) داده‌ها

۹ـ امکان اعمال ضوابط دقیق ایمنی

۱۰ـ امکان ترسیم داده‌ها (تجمع داده‌ها در یک سیستم متمرکز آنها را آسیب‌پذیر می‌کند)

۱۱ـ تامین استقلال داده‌یی

۱۴ـ ایجاد تعادل بین نیازهای حتی گاه متضاد کاربران

۱۵ـ تسهیل گسترش موارد کاربردی و رشدپذیری محیط ذخیره‌سازی

۱۶ـ تسریع در دریافت پاسخ پرس‌و جوها

۱۷ـ تسهیل در دریافت گزارش‌های متنوع و آمارهای مختلف، اکثر DBMSهای امروزی دارای مولد گزارش (Report Generator) هستند.

۱۸ـ در دسترس بودن داده‌ها و سیستم. یعنی داده‌ها در هر لحظه و هر جا که کاربر درخواست کند در اختیارش قرار بگیرند. در دسترس بودن سیستم از نظر کاربر بدین معناست که خود سیستم کمترین نقص و از کارافتادگی را داشته باشد.

۱۹ـ وضوح بخشیدنی به دید کاربران نسبت به داده‌های ذخیره شده

۲۰ـ تسهیل در ایجاد تغییرات و هماهنگی با نیازهای جدید کاربران

۲۱ـ تعداد زبانهای میزبان

۲۲ـ تعداد انواع کاربران (از نظر سطح و نحوه تماس با بانک)

۲۳ـ کاهش هزینه‌های سازمان

۲۴ـ تامین امکانات سازماندهی مجدد. در یک سیستم DBMS برای سازماندهی مجدد سطح داخلی و فیزیکی بانک روتین‌های کارا برای سازماندهی مجدد فایلها وجود دارد به نحوی که کارآیی بانک همواره در وضعیتی مطلوب نگاه داشته می‌شود. این سازماندهی نباید پیرو درخواست کاربر باشد بلکه خود DBMS باید براساس ضوابط و پارامترهایی کار سازماندهی مجدد را انجام دهد.

در مدل رابطه ای چند مفهوم در خصوص کلید مطرح است که عبارتند از :

- ابر کلید
- کلید کاندید
- کلید اصلی
- کلید بدیل
- کلید خارجی

مفهوم ابر کلید super key

مجموعه ای ازیک یا چند صفات خاصه را که دارای یکتایی مقدار باشند ابرکلید گویند. به بیان دیگر ، هر ترکیبی از صفات خاصه رابطه که در هیچ دو تاپل مقدار یکسان نداشته باشد.

کلید(نامزد) کاندید: Candidate Key

ابر کلیدی که خاصیتی کاهش ناپذیری داشته باشد کلید کاندید گویند. بعبارت دیگر هر زیر مجموعه از مجموعه عنوان (Ai, Aj ..Ak) که دو خاصیت زیر داشته باشد کلید کاندید گویند.

۱_ یکتایی مقدار (Uniqueness)

به این معنا که در هر لحظه از حیات رابطه مقدار (A i , Aj ..Ak) یکتا باشد.

۲_ کاهش ناپذیری minimatlity

به این معنی است که از نظر تعداد اجزاء در حداقل باشد در عین حال که یکتایی محفوظ بماند. گوئیم زیر مجموعه ای کاهش ناپذیر است یا حداقل اجزاء دارد اگر یکی از عناصر این زیرمجموعه حذف شود در زیرمجموعه باقیمانده خاصیت یکتایی مقداراز بین برود.

کلید اصلی: Primary Key

یکی از کلیدهای کاندید است که طراح با توجه به ملاحضات محیط عملیاتی، خود انتخاب می کند. دو ضابطه در تعیین کلید اصلی از بین کلیدهای کاندید باید در نظر گرفته شوند.

۱_ نقش و اهمیت کلید اصلی نسبت به سایر کلید های کاندید در پاسخگویی به نیاز های کاربران

۲_ کوتاهتر بودن طول کلید کاندید از نظر طول رشته بایتی حاصله از ترکیب صفات خاصه.

کلید اصلی شناسه تاپل است و بایستی به نوعی به سیستم معرفی شود که معمولا در سیستمهای رابطه ای با عبارت Primary Key (Attribute) تعریف میشود.

کلید نامزد(بدیل) Alternate Key

هر کلید کاندید غیر از کلید اصلی کلید بدیل نامیده می شود.اگر همه کلید های کاندید رابطه و نیز خود کلید اصلی به سیستم معرفی شوند ، دیگر نیازی به تصریح کلید دیگر با عبارت Alternate Key نیست.

کلید خارجی Foreign Key

هر صفت خاصه ای از رابطه Rj (ساده یا مرکب) مانند Ai که در رابطه ای دیگر مثلاً Ri کلید اصلی باشد کلید خارجی Rj نامیده می شود.

مثال : صفت خاصه S# در جدول SP کلید خارجی است و صفت خاصه P# در رابطه SP نیز کلید خارجی است.

نکته: لزومی ندارد که کلید خارجی یک رابطه جزء تشکیل دهنده کلید اصلی همان رابطه باشد هر چند در مثال بالا چنین است.

مثال: Department (Dept # , Dname , manager - Emp #, budget)

Employe (Emp # , Ename, Dept # , Salary)

SQL یک زبان استاندارد برای کار روی بانک اطلاعاتی رابطه ای است و هر محصول نرم افزاری موجود در بازار آن پشتیبانی می کند . SQL اولین بار در اوایل دهه ۱۹۶۰ در بخش تحقیقات IBM طراحی شد و برای اولین بار د مقیاس بزرگ روی کامپیوترهای IBM به نام سیستم R پیاده سازی شد و سپس بر روی انواع متعددی از محصولا تجاری در IBM و محصولات دیگر پیاده سازی شد . در این فصل زبان SQL مورد بحث SQL 92 یا SQL 2 است ک نام رسمی آن International Standard Database Language SQL (1992) می باشد .

SQL بجای دو اصطلاح رابطه و متغیر رابطه ای از اصطلاح جدول استفاده می کند . S QL تا تبدیل شدن به یک زبان رابطه ای کامل فاصله زیادی دارد . با این همه SQL استاندارد است و تقریبا توسط هر محصول نرم افزاری بانک اطلاعات پشتیبانی می شود و هر فرد حرفه ای نیازمند آن است .

۵-۲- احکام تعریف داده ها (DDL) در SQL

این احکام شامل تعریف جداول ، شاخص ، حذف جداول ، شاخص و تغییرات می باشد .

انواع دامنه ها در SQL به شرح زیر است :

- Char (n) یک رشته کاراکتری با طول ثابت که توسط کاربر n مشخص می شود .
- Varchar (n) رشته های با طول متغیر حداکثر به طول n
- Int اعداد صحیح
- Small int اعداد کوچک
- Numeric (p,d) اعداد اعشاری p حداکثر تعداد رقمها و dتعداد رقم های اعشاری .
- Double , real
- Float(n) اعداد اعشاری با دقت حداقل n رقم .
- Not null می تواند در انتهای تعریف فیلد به کار رود بطوریکه صفت خاصه مورد نظر نمی تواند مقدار null داشته باشد .
- Date تاریخ شامل ٤ رقم برای سال ، ماه و روز :

  مثال : 2001 – 7 – 27

- Time برای نمایش ساعت .

با دستور Create domain می توان دامنه‌های جدیدی را تعریف کرد.

مثال ۱:

```
Create domain seasion char(8)
Default ‘bahar’
Check (Value in ‘bahar’ , ‘tabestan’ , ‘paeez’ , ‘zemestan’)
```

با دستور فوق دامنه ای جدید به نام Seasion تعریف می‌شود که فقط مقادیر اساسی فصلها را به خود می‌گیرد و مقدار پیش فرض آن بهار بوده و از نوع کاراکتری ۸ خانه‌ای می‌باشد.

تذکر: با دستور Alter domain می‌توان تعریف میدان را تغییر داد و با دستور Drop domain می‌توان میدان تعریف شده‌ای را حذف کرد.

عملکردهای ریاضی عبارتند از : + ،- ، × ، /

عملگر || برای الصاق دو رشته کاراکتری استفاده می شود. مثلا با دستور NAME||FAMILY دو متغیر رشته ای با هم ترکیب می‌شوند. در بعضی از پیاده سازیها علامت + برای چسباندن دو رشته استفاده می‌شود.

عملکردهای مقایسه عبارتند از = ، =~ ، < ، =<، > ، =>

علامت =~ به معنای نامساوی می باشد. علامت مساوی و نامساوی در بعضی پیاده سازیها متفاوت است. رابط های منطقی عبارتند از NOT,OR ,AND

تذکر : وقتی که فیلدی از یک رکورد NULL (پوچ- هیچ-هیچ مقدار)باشد معنایش این

است که مقدار آن فیلد ناشناخته است. هر فیلدی می‌تواند حاوی مقدار NULL باشد مگر آنکه در تعریف آن NOTNULL به کار رفته باشد. هنگام درج یک رکورد، اگر برای فیلدی از آن، مقداری مشخص نشده باشد، SQL به طور خودکار آنرا NULL می دهد. اگر یکی از عملوندهای عبارت محاسباتی مقدار NULL داشته باشد تمام عبارت برابر NULL می‌شود. مثلاً اگر فیلد WEIGHT برابر NULL باشد عبارت 454*WEIGHT نیز NULL می‌شود.

مقدار NULL در عمل مقایسه نیز نقش خاصی دارد اگر یکی از عملوندهای مقایسه ای NULL باشد نتیجه نیز NULL خواهد شد. نقش مقدار ناشناخته یا NULL(که در جداول زیر با ؟ نشان داده شده است) در رابطه‌های منطقی به صورت زیر است:

SQL

در اینجا دستورات تعریف بانک : Create Index , Drop table , Alter table , Create table و Drop Index را شرح می‌دهیم.

Create table

با این دستور می توان یک جدول مبنا ساخت. جدول مبنا جدولی است مستقل و نامدار.

مثال :

```
Create table s
(S# char(5) NOTNULL ,
sname char (20) NOT NULL,
```

status smallint ,

city char (15) NOT NULL,

primary key (S#))

با اجرای دستور فوق جدول مبنای خالی S ایجاد می شود که ۴ فیلد دارد و دارای کلید اصلی S# است. پس از ایجاد جداول می‌توان مثلاً با دستور INSERT رکوردهایی را در آن درج کرد. با عبارت Foreign key بعد از primanry key می‌توان کلید خارجی نیز تعریف کرد.

مثال : دستور زیر جدول sp را تعریف می کند:

```
Create table SP
   (S# char (5),
      P# char (6),
      Qty Numeric (9),
      Primary key (S# ,P#),
      Foreign key (S#) References S
            On delete cascade
            On update cascad,
    Foreign key (P#) References P
            On delets cascade
            On update  cascade,
    Check (Qty >1  AND Qty <1000))
```

نام پس از References معین می‌کند که این کلید خارجی به کدام جدول ارجاع می‌شود عباراتOn delete cascade ,On update cascad می‌گویند هنگامی که این کلید در

جدول اصلی خودش حذف شد یا تغییر کرد، در این جدول هم این حذف یا تغییرات اعمال گردد. وجود این قیدهای On delete , On update اختیاری می‌باشند.

قسمت Check اختیاری می‌باشد برای بیان قوانین جامعیتی به کار می رود. در جدول فوق فیلد Qty باید مقداری ، بین ۱ تا ۱۰۰۰ داشته باشد.

تذکر: با کلمه کلیدی UNIQUE می‌توان کلیدهای فرعی را مشخص ساخت که نمی تواند تکراری باشند.

Alter table

با این دستور می توان تغییراتی در یک جدول موجود داد.

مثال :

```
ALTER TABLES
ADD  DISCOUNT  SMALLINT
```

دستور فوق (به همراه کلمه کلیدی ADD) ستونی به نام اختیاری DISCOUNT را به جدول s اضافه می‌کند.

تمام رکوردهای موجودS با اجرای این حکم به جای چهار فیلد ، ۵ فیلد خواهند داشت و مقدار فیلد پنجم در تمام سطرها NULL است. در دستور ALTER نمی توان عبارت NOT NULL را به کار برد.

با این دستور می‌توان تعریف کلید اصلی یا کلید خارجی را به تعریف جدول اضافه یا از آن حذف کرد. همچنین می توان یک قاعده جامعیت جدید را برای یک جدول وضع کرد یا آن را حذف کرد.

با عبارت < نام ستون > ALTER یا < نام ستون > Modify می توان جلوی Alter table تعریف یک ستون را تغییر داد.

مثال :

```
Alter table SP
Modify (S# char (10));
```

این مثال در واقع نوع داده را تغییر نمی‌دهد بلکع طول S# را از ۵ به ۱۰ افزایش می‌دهد. اگر بخواهیم نوع داده را به طور کلی عوض کنیم، اکثر نسخه های SQL از اینکار جلوگیری می‌کنند مگر آنکه ستونهای مربوطه فاقد داده باشند.

مثال :

```
Alter table SP Modify (S# Smallint);
```

تذکر : حذف یک ستون در بعضی از SQL ها پیاده سازی نشده است زیرا حذف ستون ممکن است تأثیر نامطلوبی روی ارتباط با یکدیگر بگذارد. ولی در بعضی نسخه‌ها با عبارت < نام ستون > Drop بعد از Alter table می توان ستونی را حذف کرد.

DROP TABLE

برای از بین بردن یک جدول استفاده می‌شوند مثل Drop table S. با اجرای این دستور تمام شاخصها و دیدهای تعریف شده روی جدول و همچنین تمام کلیدهای خارجی جدول به طور خودکار از بین می رود.

در SQL برای کار با داده‌ها چهار دستور وجود دارد: DELETE ,UPDATE,INSERT, SELECT در ادامه این دستورات را شرح می‌دهیم و برای این کار از جداول بانک اطلاعاتی تهیه کنندگان و قطعات (جداول SP,P,S) استفاده می‌کنیم. که جهت سادگی رجوع ، این جداول را در صفحه ای مجزا در انتهای فصل آورده ایم.

SELECT

مهمترین دستور SQL بوده و برای بازیابی یک اطلاعات خاص استفاده می شود. سه عمل ∞, Π, σ جبر رابطه‌ای را می‌توان با این دستور به راحتی انجام داد.

ساده ترین شکل این دستور به صورت زیر است:

Select نام فیلدها

from نام جدول

where شرط جستجو

مثال :

```
select S# ,status
froms    S
where    city= ‘C2’
```

خروجی ⟶

S#	status
S1	20
S3	30
S4	20

مثال :می‌توان نام جدول را همراه نام فیلدها به کار برد.

دستورات مقابل دقیقاً مثال قبل عمل می‌کنند.

```
select    S.S# ,status
froms    S
where    city= ‘C2’
```

هر چند که در مثال فوق استفاده از نام جدول اختیاری است ولی در بعضی موارد که جلوتر خواهیم گفت الزامی می شود. در مثال فوق select یک زیر مجموعه افقی عمودی از جدول S را داد.

فرمت کلی دستور select به صورت زیر است:

```
SELECT [DISTINCT] item(s)
FROM    table(s)
[WHERE شرط] [GROUP BY فیلد(ها)] [HAVING شرط]
[ORDER BY فیلد(ها)]
```

مثال : دستور روبرو :

select P# from SP

تمام مقادیر ستون P# از جدول SP را می‌دهد (حتی به صورت تکراری)

(البته زیر هم نوشته می‌شوند)

P1 ,P2 ,P3 ,P4 ,P5 ,P6 , P1 ,P2 ,P2 ,P2 ,P4 ,P5

مثال : دستور روبرو:

Select Distinct P# from SP

به علت استفاده از Distinct مقادیر ستون P# از جدول SP را با حذف تکراری‌ها می دهد

یعنی خروجی به صورت زیر می‌شود:

(البته زیر هم نوشته می‌شوند)

P1 ,P2 ,P3 ,P4 ,P5 ,P6

اگر به جای Distinct از عبارت ALL استفاده شود آنگاه تکراری‌ها نوشته می‌شوند(البته ALL حالت پیش فرض است)

مثال : شماره تمام قطعات و وزن هر یک را بر حسب گرم بدهید. فرض کنید وزنها بر حسب پوند در جدول P باشند.

```
Select P# , 'weight in gram=' ,WEIGHT * 454
From P
```

مثال : دستور زیر کل جدول S را چاپ می‌کند:

```
select *
from S
```

وجود × به معنای تمام ستونها می باشد. دستور فوق معادل دستور زیر است:

```
select S# ,Sname ,status ,city from S
```

مثال : شماره تهیه کنندگانی را بیابید که ساکن C2 بوده و وضعیت آنها از 20 بیشتر باشد:

خروجی :

```
Select S#                    S#
```

From S S3

Where city = ‘C2’ AND status >20

مثال : شماره وضعیت تهیه‌کنندگان ساکن C2 را بر اساس نظم نزولی مقادیر وضعیت بدهید.

خروجی :

```
Select  S# ,status
From    S
Where   city= ‘C2’
Order  by  status  DESC
```

S#	Status
S3	30
S1	20
S4	20

تذکر: ستون جلوی order باید نام ستونی از جدول جواب باشد، پس دستور زیرز غلط است:

select S# from S order by city⇒ غلط

تذکر : می توان به جای نام ستون، شماره ستون را نوشت. ستونها از چپ به راست از یک شماره‌گذاری می‌شوند. این کار امکان می‌دهد تا نتیجه پرس و جو براساس مقادیر یک ستون محاسبه شده که فاقد اسم است، منظم گردد.

مثال :

```
Select  P# ,weight * 454
From    P
Where   city= ‘C2’
Order  by  2,P#
```

عدد ۲ یعنی ستون دوم جدول جواب

P#	
P1	5448
P5	4558
P4	6356
P2	7718
P3	7718

به کمک between می‌توان وجود یک مقدار در یک محدوده و به کمک in می‌توان وجود یک مقدار را در مجموعه ای از مقادیر بررسی کرد.

مثال ۲۰: خروجی این دستور چیست؟

```
Select  P# ,color ,weight
From P
Where  weight  between  16 and 19
```

P#	Color	Weight
P2	Green	17
P3	Blue	17
P6	Red	19

می توان از not between نیز استفاده کرد.

مثال : خروجی این دستور چیست؟

```
Select  P#,weight
From P
Where  weight  in (12 ,16,17)
```

P#	Weight
P1	12
P2	17
P3	17
P5	12

می توان از not in نیز استفاده کرد.

select like

علامت درصد(٪) به جای مجموعه ای از کاراکترها و علامت زیر خط(_) به جای یک کاراکتر می آید.

مثال : مشخصات قطعاتی را بیابید که اسم آنها با حرف C شروع شده باشد.

```
Select *
From P
Where pname like 'C%'
```

P#	Pname	Color	Weight	City
P5	Cam	Blue	12	C3
P6	Cog	Red	19	C2

مثال : دستور زیر نام قطعاتی را می‌دهد که ۳ حرفی بوده و با حرف C شروع می‌شوند:

```
select pname from P where pname like 'C_ _'
```

مثال : دستور زیر نام قطعاتی را می‌دهد که حاوی کاراکتر 'E' نمی‌باشد:

```
select pname from P where pname not like '%E%'
```

مثال : like "_A%B" اسامی را می‌دهد که حرف دوم آنها A بوده و مختوم به B می‌باشند.

تذکر: برای بررسی مقدار NULL بودن یک فیلد باید از عبارت is NULL استفاده کنیم و برای بررسی عدم NULL بودن از عبارت NOT NULL استفاده می کنیم.

مثال :

```
select S# frome S where statuse is NULL
```

تذکر : عملگر like نسبت به بزرگی و کوچکی حروف حساس است.

این توابع عبارتند از :

Count: تعداد مقادیر در یک ستون

Sum : مجموع مقادیر یک ستون

AVG: میانگین مقادیر یک ستون

MAX: بزرگترین مقدار در یک ستون

MIN: کوچکترین مقدار در یک ستون

مثال : کل تعداد تهیه شده از قطعه P2 را بدهید.

```
Select SUM(Qty) From  SP
Where  P#= 'P2'
```

جواب

1000

مثال : شماره تهیه‌کنندگانی را بدهید که مقدار وضعیت آنها کوچکتر از مقدار ماکزیمم وضعیت باشد.

```
Select  S# from  S
Where  status <(select MAX(status) from  S)
```

S#

S1

S2

S4

select ••Group By ••Having

مثال : کل مقدار تهیه شده از هر قطعه را در جدول جواب بدهید(همراه با شماره هر قطعه)

جواب :

```
select  P# ,SUM(Qty)
From  Sp
Group By  P#
```

P#	
P1	600
P2	1000
P3	400
P4	500
P6	100

مثال : برای هر قطعه تهیه شده، شماره قطعه، کل تعداد و ماکزیمم تعداد تهیه شـده از آن را بدون در نظر گرفتن S1 بدهید.

```
Select  P# ,SUM(Qty),MAX(Qty)
From  SP
Where  S#~= 'S1'
Group  By  P#
```

P#		
P1	300	300
P2	800	400
P4	300	300
P5	400	400

تذکر : صفتی که گروه‌بندی روی آن انجام می‌شود حتماً باید در خروجی بیاید. بخش Group by جزو آخرین بخشهای یک دستور است و فقط order by , having می‌تواند بعد از آن بیاید.

Having همواره با group استفاده می‌شود. نقش having در گروه مانند نقش where در سطر است. به عبارت دیگر از having برای در نظر گرفتن گروهها استفاده می‌شود، همانطور که از where برای در نظر نگرفتن سطرهایی در جدول جواب استفاده می‌گردد.

مثال : شماره قطعه تمام قطعاتی که توسط بیش از یک تهیه کننده تهیه شده‌اند را بیابید:

```
select  P# from   Sp
Group  By  P#
Having  Count (*) >1
```

فرم کلی insert:

فرم دوم	فرم اول
Insert	Insert
Into < نام جدول >[(فیلد ۱ [و فیلد ۲ [...])]	Into < نام جدول >[(فیلد ۱ [و فیلد ۲ [...])]
<جستجو >	Values (ثابت ۱ [،ثابت ۲ [...])

insert into hassan(name,nc) values('Ali Ahmadi'**و۰۹۱۷۴۶۵۸۴۱۱**)

:Update

شکل کلی این دستور به صورت زیر است:

Update< نام جدول>

Set <1 فیلد>=<1 مقدار>[,<2مقدار۲>=<فیلد>]...

[where <شرط>]

مثال : رنگ قطعه P2 را به زرد تغییر داده به وزن آن ۵ بیفزایید و شهر محل انبار کردن آن را ناشناخته اعلام کنید.

```
Update  P
Set color = 'Yellow' , weight = weight +5 ,city =NULL
Where  P# = 'P2'
```

تذکر: ممکن است بهنگام سازی همزمان در چند رکورد صورت گیرد.

مثال : تعداد را در محموله‌های تهیه‌کنندگان ساکن C3 صفر کنید .

```
Update  SP  Set Qty =0
Where  S# in (select  S# From  S where city = 'C3'
```

Delete

شکل کلی این دستور که برای حذف سطرها استفاده می شود به صورت زیر است:

Delete From < نام جدول> [where <شرط>]

مثال : تهیه کننده S5 را حذف کنید .

```
Delete From S where S#= 'S5'
```

مثال : تمام محموله‌هایی که تعداد آنها از ۳۰۰ بیشتر است را حذف کنید.

```
Delete From SP where Qty>300
```

مثال : تمام محموله های تهیه‌کنندگان ساکن C3 را حذف کنید.

```
Select From SP
Where S# in (selsct S# from S where city = 'C3'
```

تذکر : اگر در دستورات Delete ,Update,Insert یک پرس و جوی داخلی داشته باشیم، در اینصورت در جلوی From در پرس و جوی داخلی نباید به جدولی ارجاع کرد که قرار است عملیات درج، حذف یا تغییر در آن صورت گیرد.

مثال ۵۵: حذف تهیه کنندگانی که وضعیت آنها از میانگین وضعیتها کمتر است.

```
Delete From S
Where status <(select AVG (Status) From S)
```

دستور فوق غلط است.

create database hashemi;**ساخت پایگاه داده**

use hashemi**استفاده از پایگاه داده هاشمی**

create table student

(

 id int primary key identity **(۱,۱),** **تعریف کلید اصلی و آیدنتیتی به عنوان گام شمارش**

 name nvarchar(**۵۰)و** **فیلد نام از نوع کاراکتری با طول**

family nvarchar(**۵۰**),

 nc bigint **و** **نوع داده عدد صحیح با طول زیاد جهت ذخیره کد ملی**

address nvarchar(**۵۰**)

);

insert into student(name,nc) values('Ali Ahmadi',**۳۴۲**)

Delete from student where name='Ali Ahmadi'

update student set name='Alireza' where nc=**۳۴۲**

select id,name,nc from student where name='mohammad'

بازیابی اطلاعات افرادی که نام وفامیل افرادی که نامشان ahmad **می باشد.**

use hashemi

select name,family from student where name='ahmad'

بازیابی اطلاعات افرادی که نام وفامیل افرادی که ادرس آنها بندرعباس می باشد.

use hashemi

select name,family from student where address=' bandarabbas'

بازیابی اطلاعات افرادی که نامشان رضا می باشد. دقت نمایید نباید اطلاعات تکراری را بازیابی نماید.

se hashemi

select Distinct name from student where name='reza'

بازیابی نام همه افرادی که مشخصات آنها در جدول student **ثبت شده است ضمنا نباید تکراری باشد.**

use hashemi

select Distinct name from student

نام افرادی که فامیلشان هاشمی می باشد را به امیررضا تغییر دهید.

use hashemi

update student set name='amirreza' where family='hashemi'

Select * from table student
insert into student(name ,family) values('mobina','hashemi')
update student set name='hossein',address='minab' where name='narges'
delete from student where name='ali'

پرس وجویی بنویسید که دروس قنبری را نمایش دهد

select * from student inner join sb on student.id=sb.id inner join book on sb.id=book.id where name='ghanbari'

پرس وجویی بنویسید که تمام دروس مربوط به تمام دانشجویان را نمایش دهد

select * from student inner join sb on student.id=sb.id inner join book on sb.id=book.id