

# فصل اول :

## معرفی پایگاه داده ها

### ۱-۱- تعریف پایگاه داده ها:

مفهوم پایگاه داده ها از نظر مؤلفین مختلف در بیان با تفاوت‌هایی همراه ولی از نظر تکنیکی به گونه ای مشابه تعریف شده است. یکی از تعاریف مناسب آن به فرم زیر می باشد:

بانک اطلاعاتی، مجموعه ای است از داده های ذخیره شده و پایا (در مورد انواع موجودیتهای یک محیط عملیاتی و ارتباطات بین آنها) بصورت مجتمع و مبتنی بر یک ساختار، تعریف شده بطور صوری با حداقل افزونگی، تحت مدیریت یک سیستم کنترل متمرکز، مورد استفاده یک یا چند کاربر، بطور اشتراکی و همزمان. با توجه به این تعریف می توان دریافت که از دیدگاه تخصصی هر مجموعه ای از فایلها ی ذخیره شده لزوما پایگاه داده محسوب نمیگردد. در ادامه برخی اصطلاحات موجود در تعریف پایگاه داده توضیح داده شده اند:

#### ▪ مجتمع Integrity و مبتنی بر یک ساختار

به معنی آن است که کل داده های عملیاتی محیط مورد نظر کاربران مختلف، در قالب یک ساختار مشخص بصورت یکجا ذخیره شده باشند. به عبارتی پراکنده گی در ذخیره سازی داده های محیط وجود نداشته باشد.

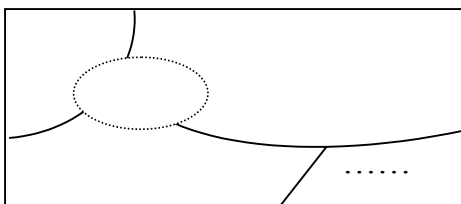
#### ▪ تعریف شده بصورت صوری:

به معنی آن است که داده ها به کمک احکام خاصی، در کادر تعریف فایلها ی مورد نیاز، تشریح و تعریف شوند و این کار زبان خاصی را لازم دارد.

#### مثالی برای درک بهتر مفهوم پایگاه داده

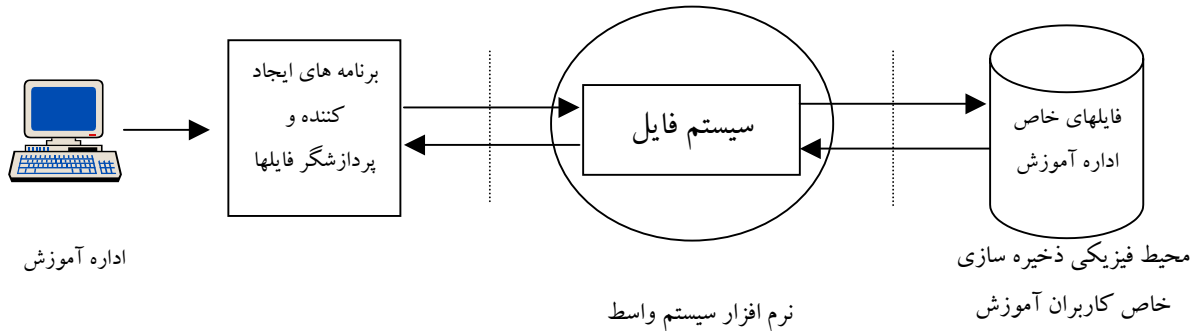
□ محیط عملیاتی دانشگاه را در نظر بگیرید که دارای بخش های عملیاتی مختلف می باشد. فرض می شود که سه بخش امور آموزش، امور دانشجویی و امور مالی دانشگاه بخشهایی هستند که می خواهیم برای آنها یک سیستم ذخیره و بازیابی کامپیوتری ایجاد نماییم و نیز فرض می کنیم که تنها نوع موجودیت مورد نظر، موجودیت دانشجو باشد و 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 موجودیتهای وجود دارند که در این مثال مورد بحث قرار نمی گیرند.

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



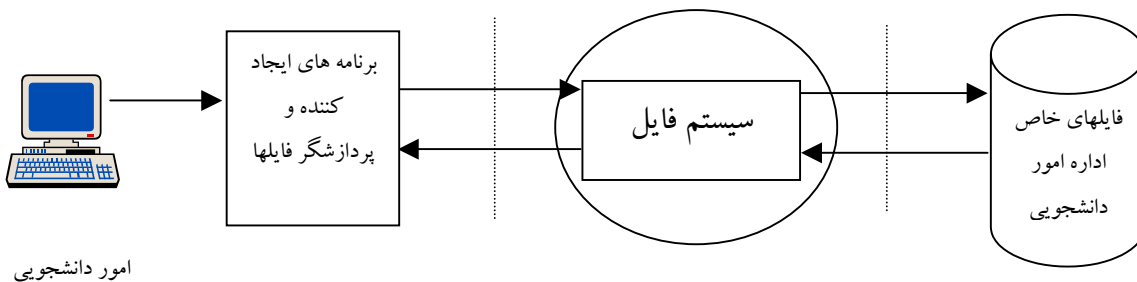
## الف - مشی غیر بانکی (سیستم فایل پردازی) File Processing

در این روش هر یک از زیر محیط های عملیاتی مستقلاً 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:

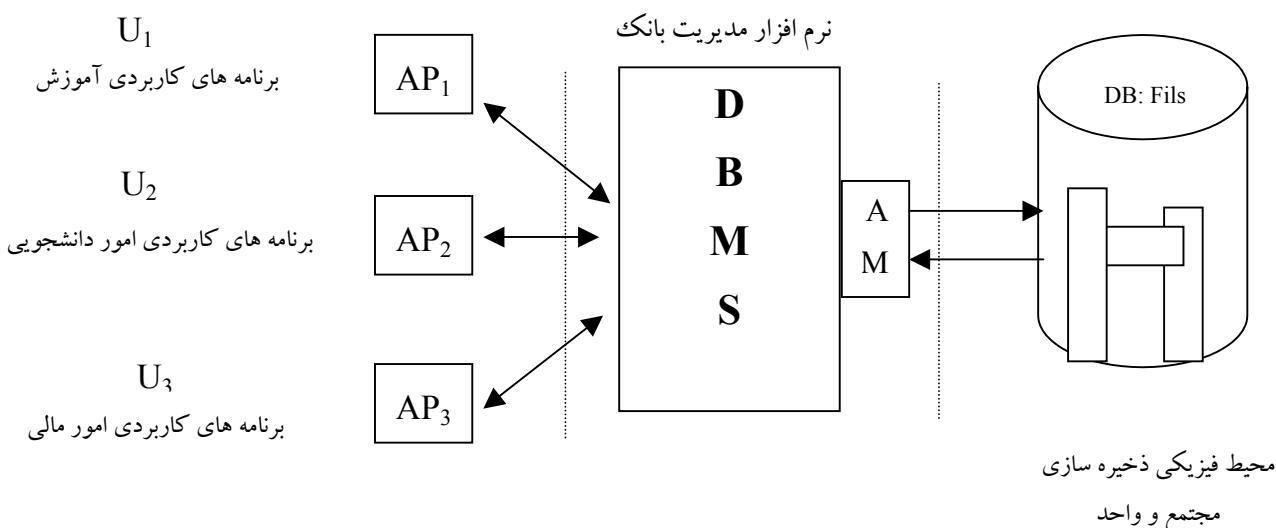
- ۱- در روش فایل پردازی داده ها از هم مجزا می باشند.
- ۲- محیط ذخیره سازی نامجمع است (تعدادی سیستم جداگانه و محیط ذخیره سازی جداگانه)
- ۳- برنامه های کاربردی وابسته به قالب فایل می باشند.
- ۴- عدم سازگاری در داده ها و فایلها دیده می شود.
- ۵- اشتراکی نبودن داده ها: داده های زیر محیط ۱ مورد استفاده کاربران زیر محیط ۲ نمی توانند قرار گیرند.
- ۶- 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 یکسری از عملیات منسجم روی آن سیستم وجود ندارد.

۸- مصرف غیر بهینه امکانات سخت افزاری و نرم افزاری و حجم زیاد برنامه سازی و استفاده غیر بهینه از مهارت و وقت تیمهای برنامه سازی .

### 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 و کاربران 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 های مجتمع: کل داده ها بصورت یک بانک مجتمع دیده می شوند و از طریق DBMS با آنها ارتباط برقرار می شود.

۲- عدم وابستگی برنامه های کاربردی به داده ها و فایلها: زیرا DBMS خود به مسائل فایلینگ می پردازد و کاربران در محیط انتزاعی هستند.

۳- تعدد شیوه های دستیابی به داده ها

۴- عدم وجود ناسازگاری در داده ها

۵- اشتراکی بودن داده ها

۶- امکان ترمیم داده ها

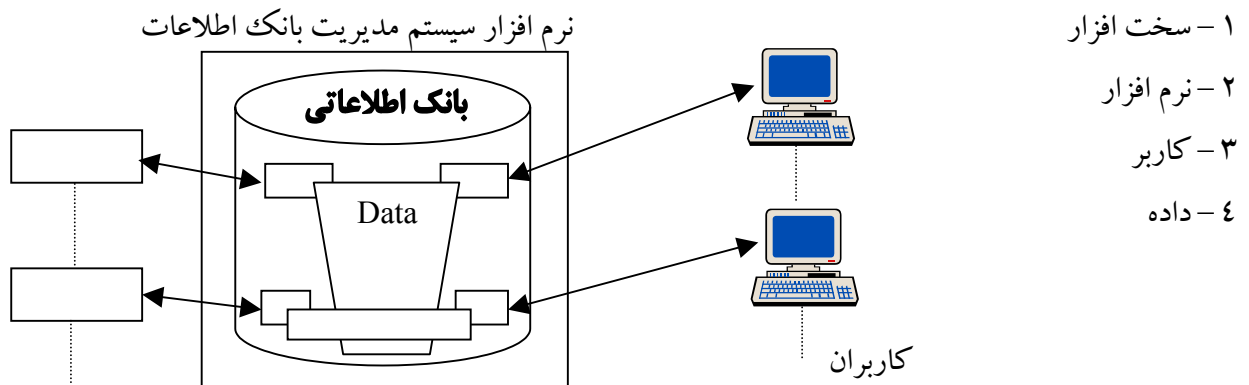
۷- کاهش افزونگی

۸- کاهش زمان تولید سیستم ها

۹- امکان اعمال ضوابط دقیق ایمنی

### ۱-۲- عناصر اصلی تشکیل دهنده محیط پایگاه داده ها:

محیط بانک اطلاعاتی ، نظیر هر محیط دیگر ذخیره و بازیابی، از عناصر زیر تشکیل می شود:



### ۱-۲-۱- سخت افزار :

سخت افزار محیط بانکی را می توان بصورت زیر تقسیم بندی نمود:

الف- سخت افزار ذخیره سازی داده ها

منظور همان رسانه های ذخیره سازی است که معمولاً برای ذخیره سازی داده ها از دیسکهای سریع با ظرفیت بالا استفاده می شود.

ب- سخت افزار پردازشی

منظور همان کامپیوتر یا ماشین است. ماشینهای خاص برای محیطهای بانک اطلاعاتی نیز طراحی و تولید شده اند که به نام DBM ماشینهای بانک اطلاعاتی نیز خوانده می شوند. این ماشینها از نظر معماری، حافظه اصلی،... و سایر اجزاء از ویژگیها و جنبه هایی برخوردارند که شرح آن در این جزوه نمی گنجد.

ج- سخت افزار ارتباطی

منظور مجموعه امکانات سخت افزاری است که برای برقراری ارتباط بین کامپیوتر و دستگاههای جانبی و نیز بین دو کامپیوتر یا بیشتر بکار گرفته می شوند، اعم از اینکه ارتباط نزدیک باشد و یا ارتباط دور. سخت افزار ارتباطی خاص محیط های بانکی نیست و در هر محیط غیر بانکی نیز ممکن است مورد نیاز باشند.

### ۱-۲-۲- نرم افزار:

نرم افزار محیط بانکی را می توان به دو دسته تقسیم نمود:

۱-۲-۲-۱- نرم افزار کاربردی:

نرم افزاری است که کاربر باید برای تماس با سیستم بانک اطلاعاتی آماده کند.

۱-۲-۲-۲- نرم افزار سیستمی:

بین بانک اطلاعاتی فیزیکی که داده ها بصورت فیزیکی در آن ذخیره می شوند و کاربران سیستم، لایه ای از نرم افزار موسوم به مدیر بانک اطلاعاتی قرار دارد. سیستم مدیریت بانک اطلاعاتی نرم افزاری است که به کاربران امکان می دهد که پایگاه از دید خود را تعریف کنند و به پایگاه خود دستیابی داشته باشند، با پایگاه خود کار کنند و روی آن کنترل داشته باشند.

### ۱-۲-۳- کاربران:

از نظر وظایفی که انجام می دهند به دو دسته کلی تقسیم می شوند:

الف: کاربران با نقش مدیریتی (DBA)

ب: کاربران با نقش استفاده کننده که به دو دسته: کاربران تولید کننده سیستم (برنامه نویسان کاربردی) و استفاده کنندگان نهایی سیستم می تواند تقسیم گردد. برنامه نویسان کاربردی مسئول نوشتن برنامه های کاربردی بانک اطلاعاتی به زبان سطح بالا یا زبانهای نسل چهارم (4GL) هستند.

### ۱-۲-۴- داده:

منظور داده هایی است که در مورد انواع موجودیتهای محیط عملیاتی و ارتباط بین آنها می باشند که اصطلاحاً به آنها داده های عملیاتی و یا داده های پایا گفته می شود. داده های ذخیره شدنی در پایگاه داده ها ابتدا باید در بالاترین سطح انتزاع مدلسازی معنایی شوند. مفاهیم داده ها در هر محیط به کمک موجودیت ها و ارتباطات نمایش داده میشوند.

### □ انتخاب موجودیتها:

#### نوع موجودیت:

عبارتست از مفهوم کلی شیء پدیده و بطور کلی هر آنچه از محیط عملیاتی که می خواهیم در موردش اطلاع داشته باشیم. مثال: دانشکده، درس، دانشجو، گروه آموزشی.

در هر محیط عملیاتی انواع مختلف موجودیتها وجود دارند. طراح پایگاه پس از مطالعه دقیق محیط عملیاتی، مجموعه موجودیتهای محیط را تعیین می کند و این اولین قدم در طراحی پایگاه داده ها است.

**توجه:** تشخیص درست موجودیت ها و شناسایی روابط بین آنها قبل از هر چیز بستگی به این دارد که در مورد چه پدیده هایی چه اطلاعاتی را می خواهیم داشته باشیم. موجودیتهایی انتخاب می شوند که نیازهای اطلاعاتی همه کاربران محیط ناظر به آنها باشد.

### □ صفات خاصه:

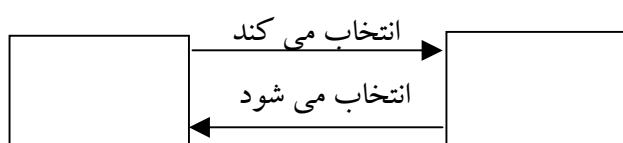
هر موجودیت مجموعه ای از صفات خاصه است که این مجموعه صفات خاصه را نیز باید طراح تعیین کند. هر صفت از نظر کاربران یک نام، یک نوع و یک معنای مشخص دارد .  
 بعنوان مثال: موجودیت کارمند می تواند دارای صفات خاصه شماره کارمندی، نام و حقوق باشد.

### □ ارتباط:

هر نوع ارتباط یک معنای مشخص دارد و با یک نام بیان می شود و نیز می توان گفت که هر نوع ارتباط، عملی است که بین موجودیتها وجود دارد.  
 انواع موجودیت های محیط عملیاتی با یکدیگر ارتباط دارند که معمولاً با یک عبارت فعلی همراه است. این ارتباطات که هر یک سمانتیک خاص را دارد باید شناسایی شده و در پایگاه ذخیره شوند.

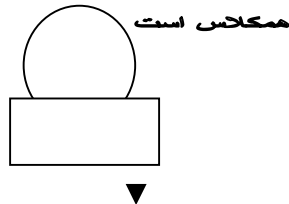
مثال: - دانشجو درس را انتخاب می کند

- درس توسط دانشجو انتخاب می شود



- بین دو موجودیت می تواند بیش از یک ارتباط متفاوت با معنای (سمانتیک) متفاوت وجود داشته باشد.

□ ارتباط ممکن است بین یک نوع موجودیت و خودش باشد. مثال: قطعه X در ساخت قطعه Y بکار می رود. به این نوع ارتباط، ارتباط بازگشتی (Recursive Relationship) نیز گفته می شود.



### ماهیت ارتباط:

تناظر بین عناصر مجموعه نمونه های یک نوع موجودیت، با عناصر مجموعه نمونه های نوع موجودیت دیگر را ماهیت ارتباط گویند.

ماهیت ارتباط بین انواع موجودیت ها عبارتند از:

ارتباط ۱ : ۱ تناظر یک به یک

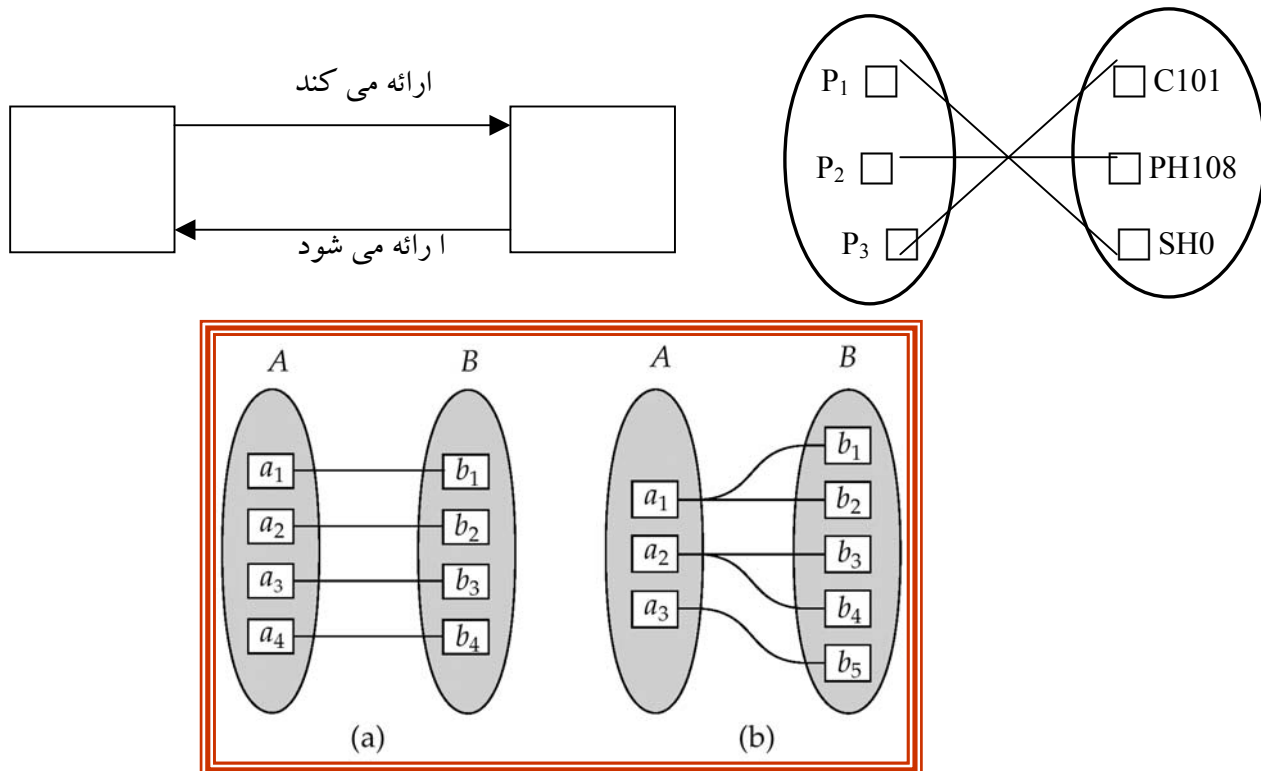
ارتباط ۱ : n تناظر یک به چند

ارتباط n : n تناظر چند به چند

- ماهیت ارتباط بر مبنای قواعد معنایی (سمانتیک) حاکم بر محیط عملیاتی تعیین می شود.

مثال: رابطه بین موجودیت های استاد و درس را در نظر بگیریم.

الف: یک استاد حداکثر یک درس را ارائه می کند و هر درس دقیقاً توسط یک استاد ارائه می شود.



ب: (ارتباط یک به چند) (درس - استاد)

یک درس ممکن است توسط بیش از یک استاد ارائه شود ولی یک استاد حداکثر یک درس را ارائه می کند.

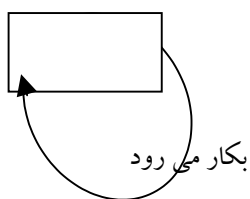
ج: (ارتباط چند به چند):

یک استاد ممکن است بیش از یک درس را ارائه کند و یک درس ممکن است توسط بیش از یک استاد ارائه شود.

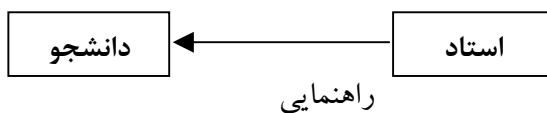
### درجه ارتباط:

تعداد موجودیتهایی که در آن ارتباط مشارکت دارند درجه ارتباط نامیده می شود.

مثال:



ارتباط درجه ۱ یا رابطه بازگشتی:



ارتباط درجه ۲:

### تمرین:

- ۱- سیستم بانکی چه مزایایی بر سیستم غیر بانکی دارد؟
- ۲- به نظر شما معایب بانک اطلاعاتی چیست؟
- ۳- تحت چه شرایطی و نه لزوما باید از تکنولوژی پایگاهی صرف نظر کرد؟
- ۴- تعریف پایگاه داده را در شش منبع معتبر خارجی بررسی نمایید.

## فصل دوم :

# مدل E/R

### ۲-۱- مقدمه :

یک مدل داده ای مجموعه ای از ابزارهای مفهومی برای توصیف داده ها ، ارتباط بین داده ها، معانی داده ها و محدودیت های آنهاست. بعبارتی یک روش تفکر درباره داده ها که به پیاده سازی ربطی ندارد. در یک نگاه کلی میتوان انواع مدل های داده ای را بصورت زیر نام برد :

- مدل های مبتنی بر اشیاء Object based Logical Models

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

- مدل های مبتنی بر رکورد Record based Logical Models

داده ها در قالب رکوردهای ثابت و یا با طول متغییر دیده میشوند. از انواع دیگر مدل های داده ای می توان به مدل های داده ای شبه ساخت یافته ، مدل های شبکه ای و مدل های سلسله مراتبی اشاره نمود.

از جمله مدل های منطقی مبتنی بر اشیاء می توان **مدل E/R** را نام برد. در سال ۱۹۷۶ 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 خود را E / R نامید. این مدل 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 موجودیت رابطه را نیز مطرح ساخت. نمودار E / R روشی برای نمایش ساختاری منطقی یک بانک اطلاعاتی به روش تصویری 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

مدل E/R بکمک نموداری تحت نام خود قابل نمایش است که در این نمودار مفاهیم مربوطه با اشکال مشخصی ترسیم میگردند. در ادامه هر یک از این اشکال نشان داده شده اند.



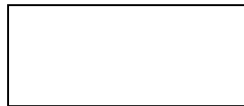
## ۲-۲-۱- موجودیت:

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.



موجودیت قوی



موجودیت ضعیف

## 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

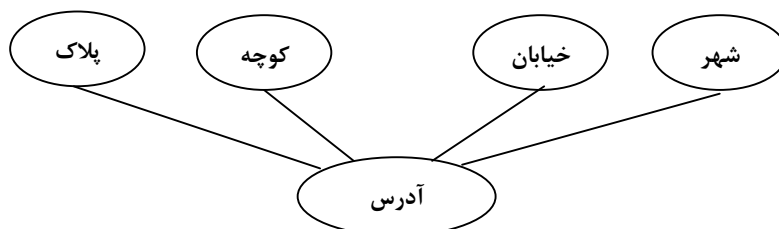
برای مشخص کردن کلید در یک موجودیت زیر صفت یا صفات خاصه کلید خط کشیده می شود.

### 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.



نکته: در مدل بانک اطلاعاتی رابطه ای صفت خاصه مرکب جایی ندارد.

### ج- صفت خاصه تک مقداری / چند مقداری Single Valued / Multi Valued

صفتی که فقط یک مقدار را در هر لحظه از زمان به خود اختصاص دهند به صفات تک مقداری معروفند. به عنوان مثال شماره دانشجویی، تاریخ تولد تک مقداری هستند. اگر برای یک صفت خاصه چندین مقدار بتواند قرار گیرد صفت خاصه چند مقداری نامیده می شود.

بطور مثال: صفت خاصه مدرک و یا تلفن برای استاد چند مقداری محسوب می شود. زیرا یک استاد می تواند دارای چند مدرک و یا تلفن مختلف باشد. در نمودار E / R برای صفت خاصه چند مقداری از بیضی دو خطی استفاده می شود.



### د- صفت خاصه مشتق (استنتاجی) [ادارای مقدار محاسبه شدنی]

صفتی است که در موجودیت وجود خارجی ندارد ولی در صورت لزوم می توان آن را بدست آورد.

مثال: موجودیت: استاد، صفت خاصه: تاریخ تولد، صفت خاصه مشتق: سن

در نمودار ER صفت خاصه مشتق با یک بیضی با مرز نقطه چین مشخص می شود.



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

### ه- 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.

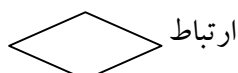
• نام استاد در یک برنامه درسی ترم ممکن است هنوز اعلام نشده باشد.

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

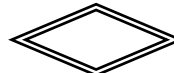
### ۲-۳-۲- ارتباط:

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.



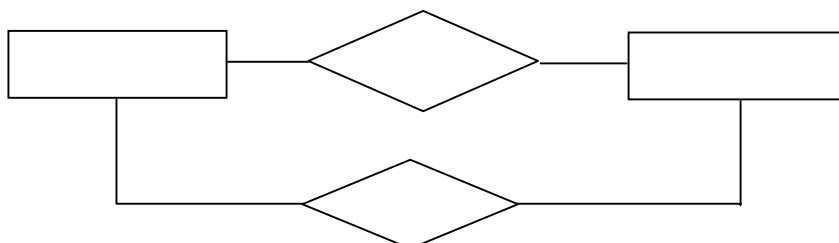
ارتباط



ارتباط با موجودیت ضعیف

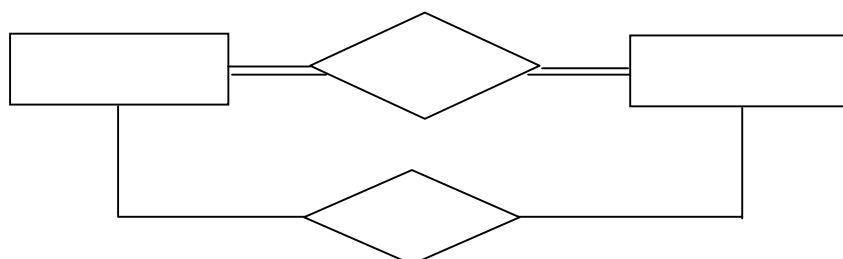
عنصرهای هر رابطه (شامل صفات خاصه و موجودیت ها) بوسیله خطوط پر به رابطه مربوطه وصل می شوند. هر خط

ارتباطی بین موجودیت و رابطه دارای برچسب ۱ یا n می باشد که ماهیت ارتباط را مشخص می کند.



### ۲-۲-۳-۱- وضع مشارکت در ارتباط

انواع موجودیتهایی که بین آنها ارتباط برقرار است شرکت کنندگان آن ارتباط نام دارند. مشارکت یک نوع موجودیت در یک نوع ارتباط ممکن است الزامی (کامل) یا غیرالزامی (ناکامل) باشد. مشارکت یک نوع موجودیت در یک نوع ارتباط را الزامی گویند اگر تمام نمونه های آن نوع موجودیت در آن باید در ارتباط شرکت کنند. در غیر اینصورت مشارکت غیر الزامی (اختیاری) است. مثال : مشارکت دانشجو در ارتباط انتخاب الزامی است ولی مشارکت دانشجو در ارتباط حذف درس الزامی نیست زیرا لزوما همه دانشجویان درس را حذف نمیکنند. در نمودار E/R مشارکت الزامی با دو خط نشان داده میشود.

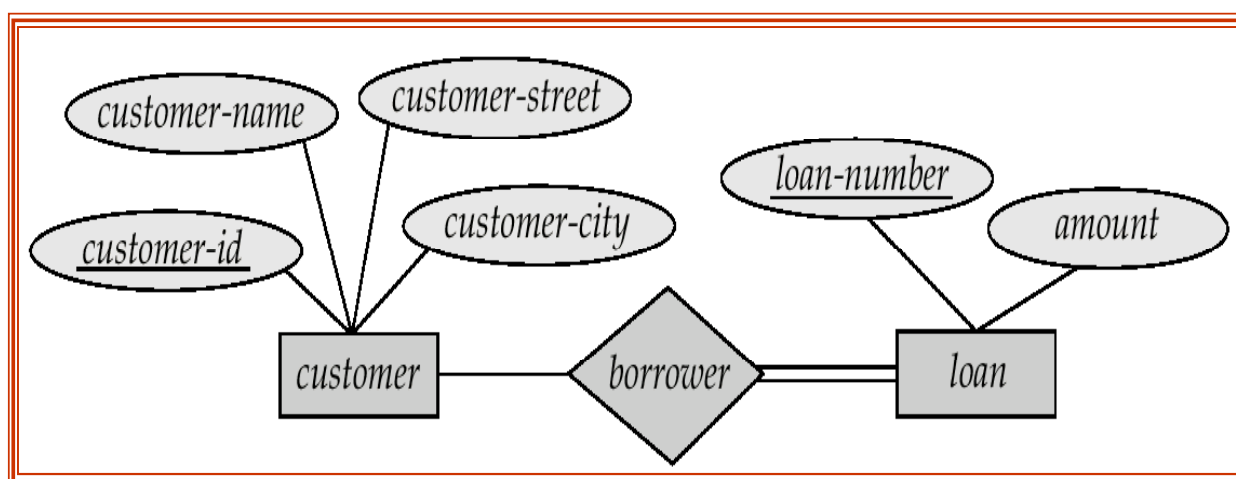


**Total participation:** هر موجودیت حداقل در یک رابطه از مجموعه ارتباط مشارکت دارد. ■

مثال : مشارکت وام در ارتباط وام گرفتن کامل است. هر وام باید حداقل یک مشتری متناظر با ارتباط وام گیرنده دارد. ■

**Partial participation:** برخی موجودیت ها ممکن است در هیچ ارتباطی از مجموعه ارتباطات مشارکت نداشته باشند. ■

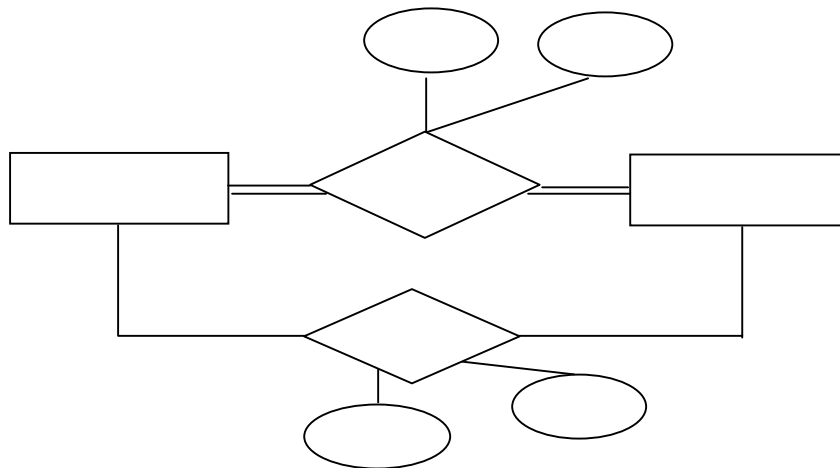
مثلا : همه مشتریان ممکن است وام نگیرند. یعنی مشارکت مشتری ناکامل است. ■



### ۲-۲-۳-۲- نوع ارتباط به مثابه نوع موجودیت (ارتباط موجودیتی)

در یک دید کلی می توان گفت نوع ارتباط خود نوعی موجودیت است . زیرا پدیده ای است که در دنیای واقعی وجود دارد . با توجه به این تعریف می توان گفت چون نوع ارتباط خود نوعی موجودیت است لذا می تواند صفت یا صفات خاصه ای داشته باشد. اما معمولاً فاقد صفت شناسه است. ارتباط یک نوع موجودیت ضعیف با موجودیت قوی معمولاً صفت خاصه ندارد.

در مثال دانشجو و درس و رابطه انتخاب می توان صفت خاصه های ترم و نمره و در رابطه حذف صفات ترم و نوع حذف را در نظر گرفت.

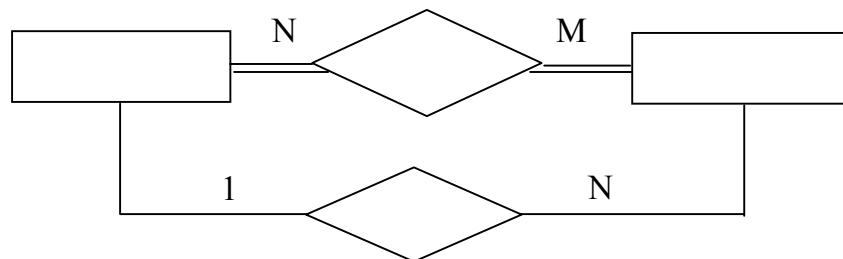


### ۲-۲-۳-۳- ماهیت نوع ارتباط

چگونگی تناظر بین دو مجموعه نمونه های موجودیت را ماهیت ارتباط گویند. می دانیم سه نوع تناظر وجود دارد :  
تناظر یک به یک ، تناظر یک به چند و تناظر چند به چند. این سه گونه تناظر را چنین نشان می دهیم :

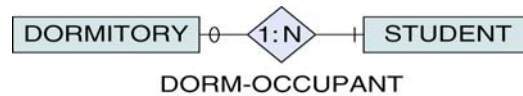
$$1 : 1 , 1 : N , N : M .$$

مثال : ماهیت ارتباط در رابطه حذف تک درس معمولاً 1:N است (یعنی یک دانشجو یک درس را حذف می کند ولی یک درس ممکن است توسط چند دانشجو حذف شود).



برای نمایش ماهیت ارتباط در نمودار E/R روش دیگری نیز وجود دارد . در این روش به هر مشارکت یک نوع موجودیت در یک ارتباط ، یک زوج عدد صحیح به صورت (min,max) انتساب داده می شود. به این معنی که در

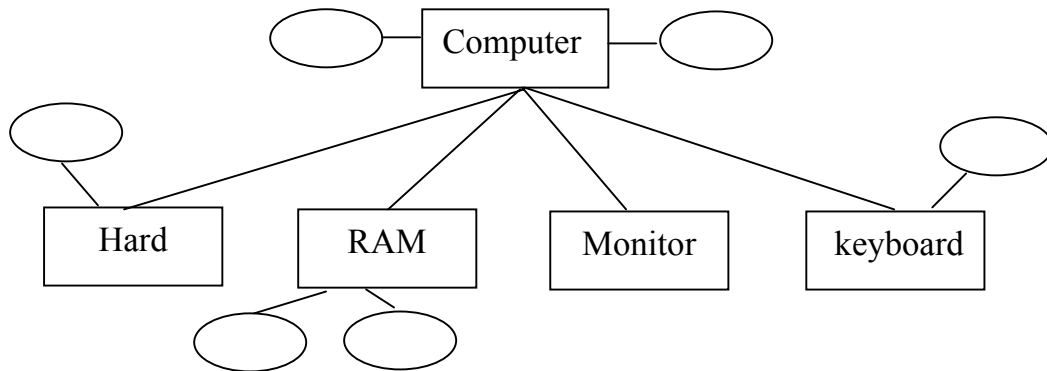
هر لحظه ، هر نمونه موجودیت e از نوع E باید حداقل در min و حداکثر در max نمونه از ارتباط R شرکت داشته باشد. اگر  $\min=0$  مشارکت غیر الزامی (اختیاری) و در غیر اینصورت مشارکت الزامی است .



□ موارد اضافه شده به نمودار E/R

### ۲-۲-۴ تجزیه و ترکیب :

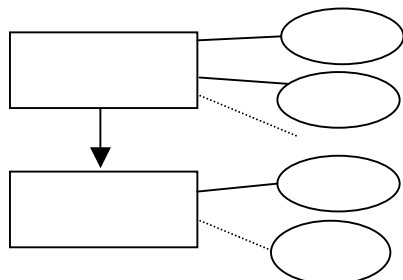
تجزیه یا جداسازی یعنی یک شیء کلی را به اجزاء تشکیل دهنده آن تقسیم کنیم . شیء کلی ، صفات ، ساختار و رفتار خود را دارد و هر یک از اجزاء نیز صفات ، ساختار و رفتار خاص خود را دارند. به این نوع ارتباط در E/R ارتباط “ جزئی است از .. ” و یا IS -A PART -OF گفته می شود .



### ۲-۲-۵ زیر نوع ها و ابرنوع های موجودیت Subtype and Supertype

یک موجودیت می تواند بطور همزمان از انواع مختلفی باشد. مثلاً اگر بعضی کارمندان، برنامه نویس باشند و تمام برنامه نویسان کارمند، آنگاه می توان گفت نوع موجودیت برنامه نویس یک زیر نوع از نوع موجودیت کارمند است.

اگر نوع موجودیت y ، یک زیر نوع از نوع موجودیت x باشد آنگاه خطی جهت دار از مستطیل x به مستطیل y رسم



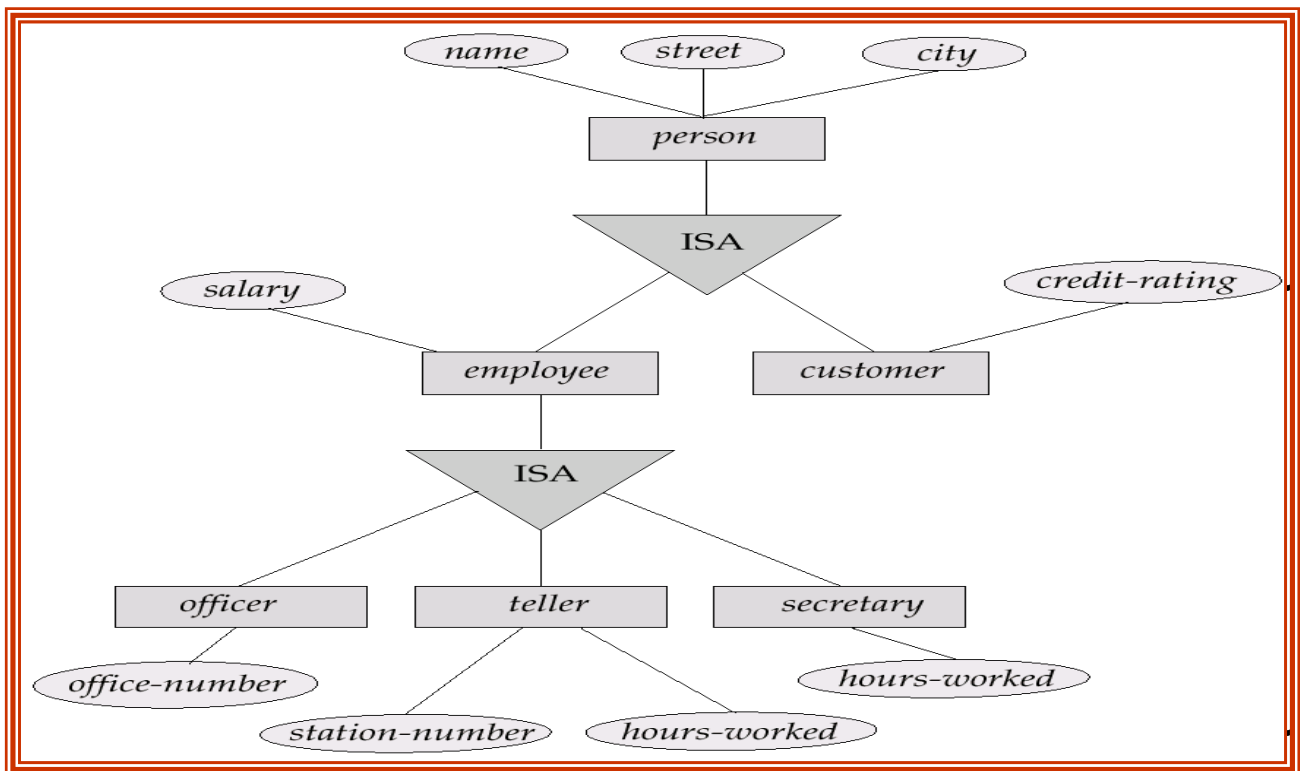
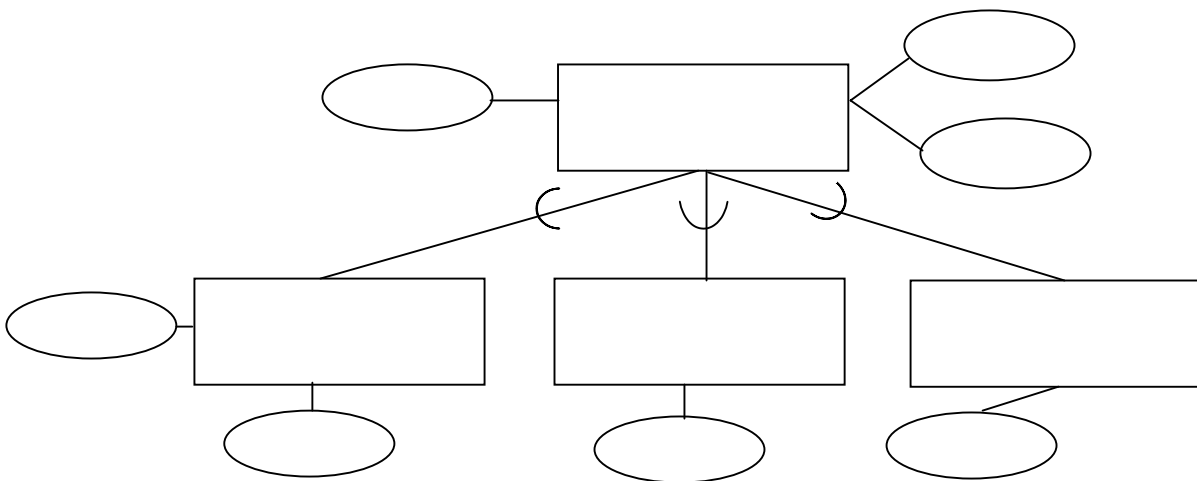
می شود. (هر y یک x است) این ارتباط برای پرهیز از تکرار صفات خاصه بین موجودیتها در یک نمودار بکار می رود

### ۲-۲-۵-۱- تخصیص (SPECIALIZATION):

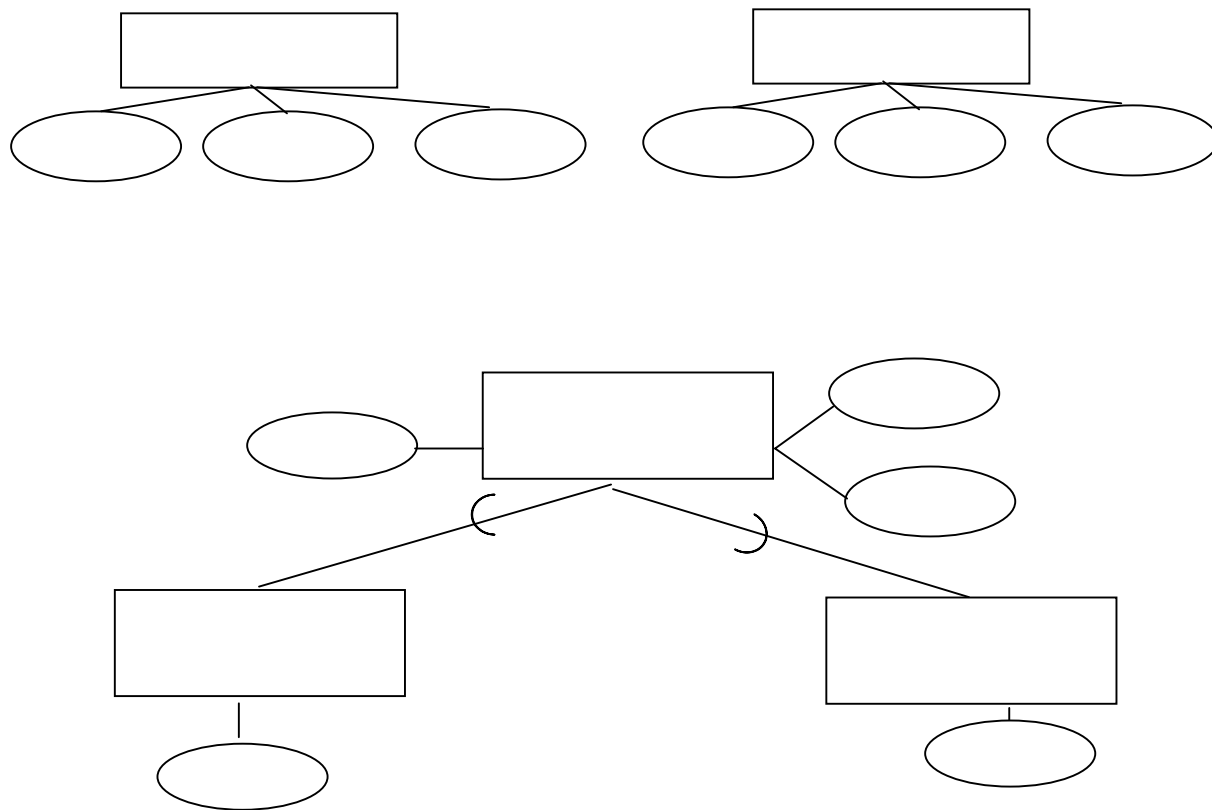
مشخص کردن گونه های خاص یک شیء را تخصیص گویند. بطور مثال اگر شیء موجود زنده را در نظر بگیریم سه گونه خاص آن عبارتند از : انسان ، حیوان و نبات . در نمودار E/R یک موجودیت میتواند زیرنوع هایی داشته

باشد. گوئیم بین هر زیر نوع موجودیت و ابر نوع آن ارتباط "گونه ای است از..." یا هست یک ... IS A وجود دارد.

**تخصیص:** یک فرایند از بالا به پایین است. در تخصیص یک موجودیت به گونه های مختلف گروه بندی می شود. این گروهها به عنوان موجودیت سطح پایین تر آن موجودیت منظور می گردند. در نمودار E/R تخصیص با یک مثلث حاوی ISA نشان داده می شود. یک موجودیت سطح پایین تمام خصوصیات مجموعه موجودیت سطح بالاتر را به ارث می برد.



تعمیم عکس عمل تخصیص است به این معنا که با داشتن زیر نوع های خاص ، صفات مشترک بین آنها را در یک مجموعه صفات برای یک ابرنوع موجودیت در نظر می گیریم. تعمیم یک فرایند از پایین به بالاست که تعدادی مجموعه موجودیتی که خصوصیات مشترک دارند را باهم ترکیب می کند.



**توجه :-** تخصیص و تعمیم معکوس یکدیگرند که هر دو در نمودار بایک شکل نشان داده می شوند.

#### ▪ قیود تعریف شده در ارتباط سلسله مراتبی تعمیم/تخصیص

برای مدل سازی دقیق تر یک سازمان ، طراح پایگاه داده ممکن است محدودیت ها /قیدها یی را در یک تعمیم ویژه در نظر بگیرد . یکی از این قید ها تعیین این است که یک موجودیت می تواند عضوی از یک مجموعه موجودیت سطح پایین در نظر گرفته شود. عضویت می تواند بر اساس شرایط و یا توسط کاربر تعیین گردد.

#### ★ بر حسب شرط خاص تعریف شده:

در این حالت عضویت بر اساس یک شرط صریح یا گزاره ای بیان می شود.

#### ★condition-defined

✓ E.g. all customers over 65 years are members of *senior-citizen* entity set;  
*senior-citizen* ISA *person*.

### ★ تعریف شده توسط کاربر:

کاربر پایگاه داده موجودیت را به یک مجموعه موجودیت مورد نظر تخصیص می دهد. نوع دیگر محدودیت از نظر اینکه یک موجودیت سطح به یک یا بیشتر از موجودیت سطح پایین تعلق دارد تعریف می گردد. این نوع محدودیت به دو شکل مجزا (Disjoint) و همپوشانی (Overlapping) وجود دارد.

### ★ Disjoint

به معنای این است که یک موجودیت می تواند فقط به یک مجموعه موجودیت سطح پایین تعلق داشته باشد. در نمودار E/R با نوشتن عبارت disjoint در کنار مثلث ISA این قید تعریف میشود.

### ★ Overlapping

به معنای این است که یک موجودیت می تواند فقط به یک یا بیشتر از مجموعه موجودیت سطح پایین تعلق داشته باشد. قید دیگری که در تعمیم مطرح شده ، قید کامل بودن است که بیانگر این است که یک موجودیت سطح بالاتر باید متعلق به حداقل یکی از مجموعه موجودیت های سطح پایین تر باشد یا خیر . بر این اساس دو نوع قید کامل و جزئی را داریم :

★ **total** : یک موجودیت باید متعلق به یکی از مجموعه موجودیت های سطح پایینتر باشد.

★ **partial** : یک موجودیت می تواند متعلق به یکی از مجموعه موجودیت های سطح پایینتر نباشد.

حالت پیش فرض مساله ، حالت جزئی است . برای بیان قید کامل بودن باید از دو خط استفاده گردد (همانند مشارکت الزامی/کامل در نمودار E/R )

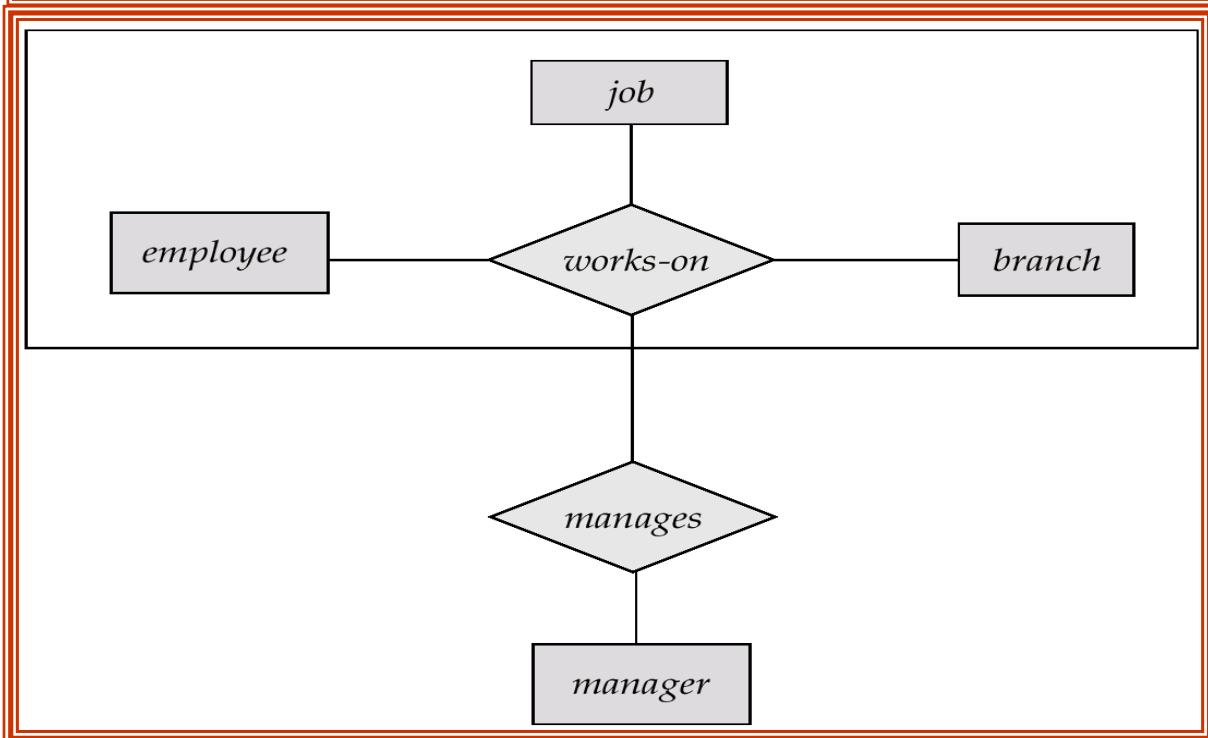
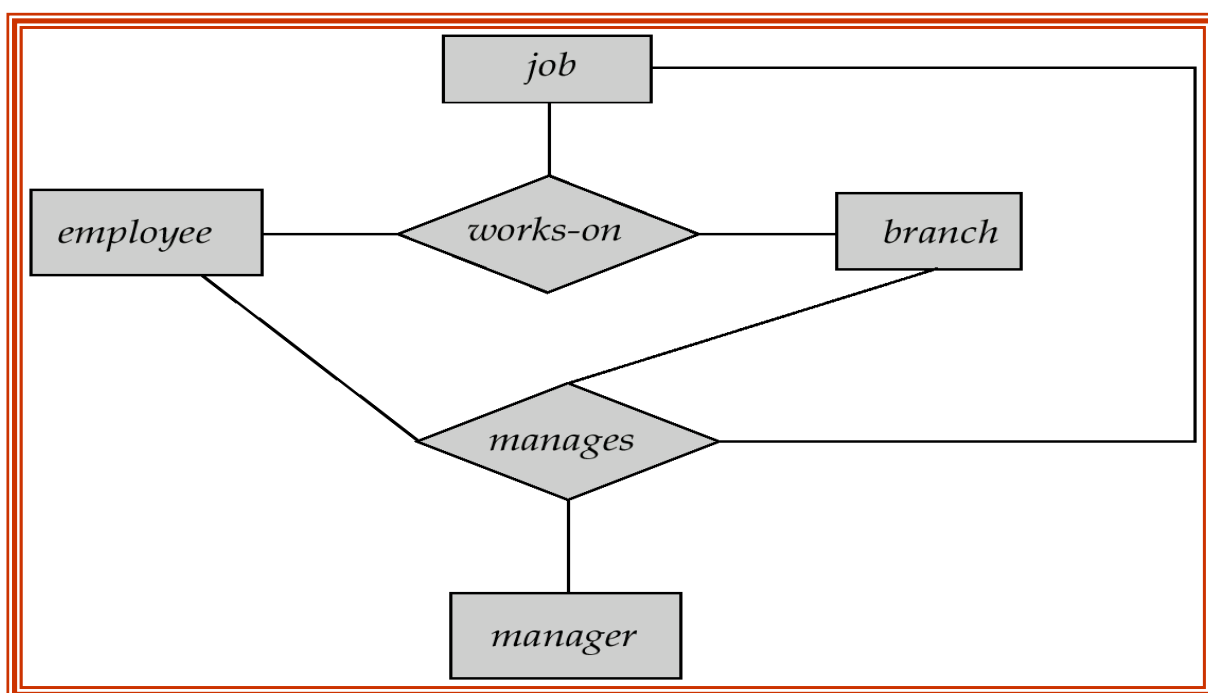
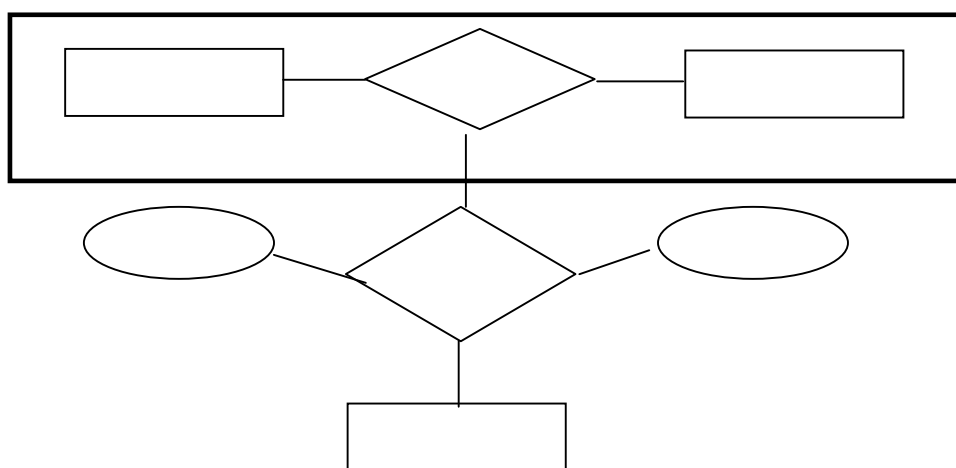
**تمرین** : ۵ مثال مختلف از تعمیم ذکر کنید و در هر کدام قیود مختلف را بررسی نمایید.

### ۲-۲-۶-تجمع Aggregation

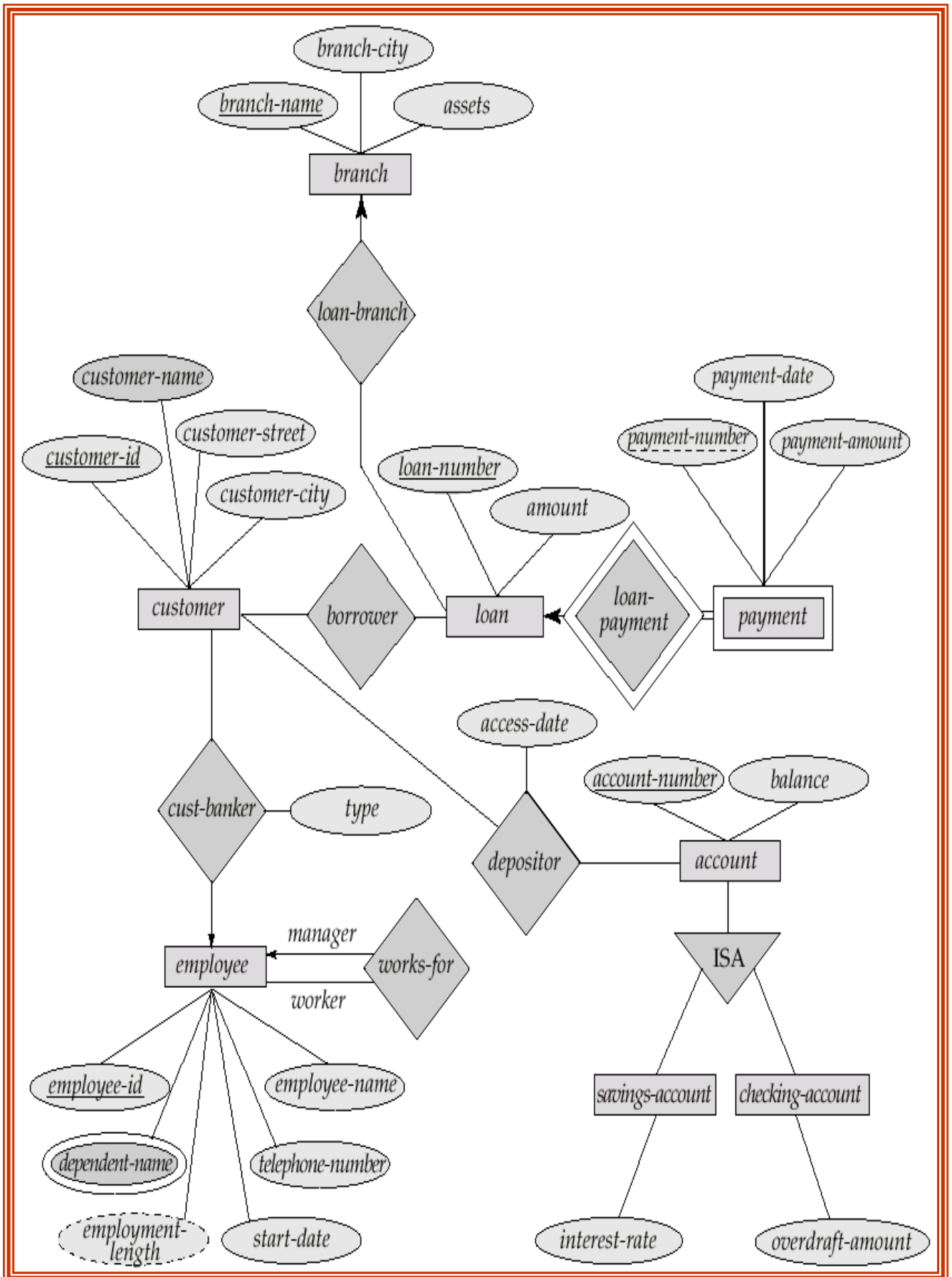
تجمع یعنی ساختن یک نوع موجودیت جدید و واحد بر اساس دو یا بیش از دو نوع موجودیت ، که خود باهم ارتباط دارند. در واقع مجموعه ای از موجودیتها و ارتباطات را با هم مجتمع کرده و به عنوان یک نوع موجودیت واحد در نظر میگیرند. و این نوع موجودیت خود می تواند با نوع موجودیت دیگری ارتباط داشته باشد. در واقع زمانی از تجمع استفاده میشود که بخواهیم ارتباطی را بین ارتباط ها بیان کنیم و یا بخواهیم ارتباطات افزونه را کم کنیم.

مثال : ارتباط بین نوع موجودیتهای دانشجو ، درس و استاد را می توان همانند شکل زیر مدل سازی نمود.



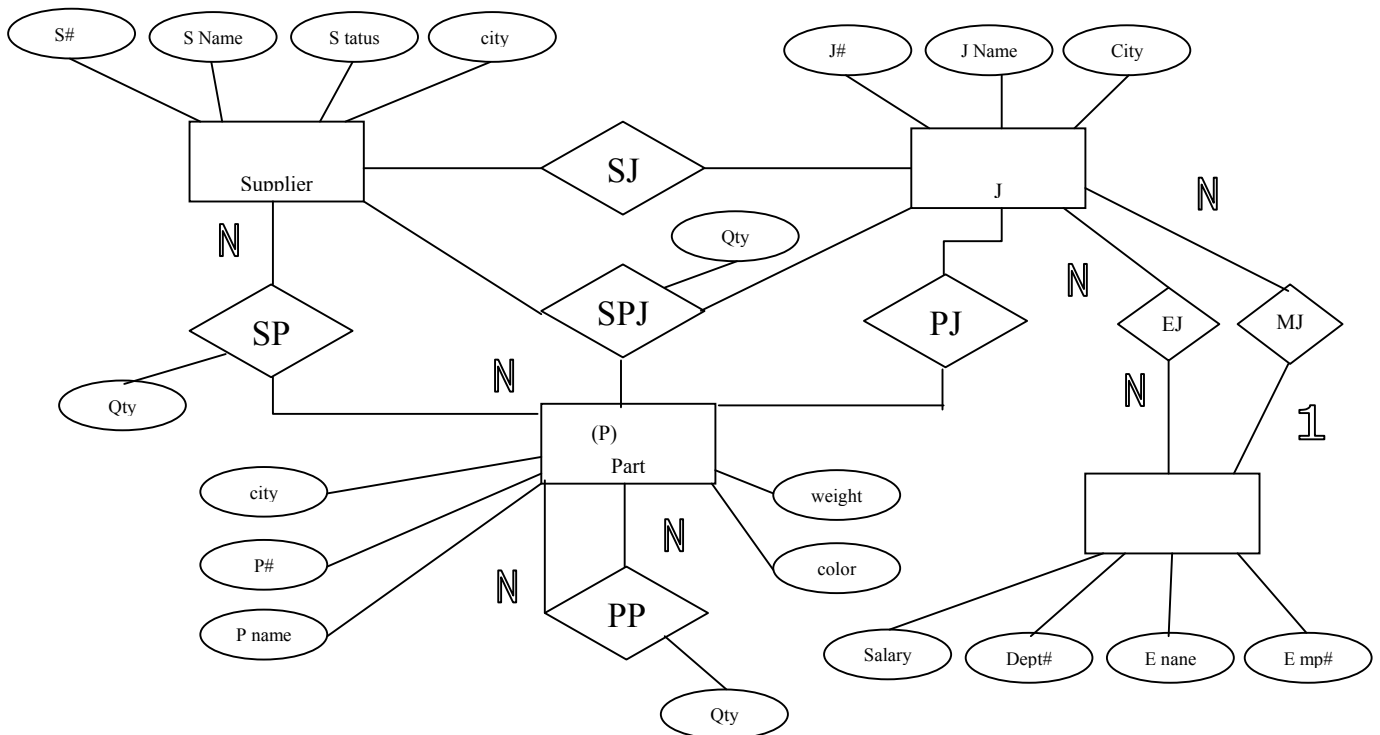


مثال : نمودار E/R یک سیستم بانکی :



### مثالی دیگر :

**محیط عملیاتی:** سازمان یا شرکتی را در نظر می گیریم که پروژه هایی را در دست اجرا دارد. در پروژه ها از قطعاتی در کار ساخت استفاده می شود و تهیه کنندگانی این قطعات را تأمین می کنند. قطعات در پروژه ها استفاده می شوند. هر تهیه کننده در یک شهر دفتر دارد. هر قطعه می تواند در ساخت قطعه دیگر نیز بکار رود. کارمند مدیر پروژه است و یا در پروژه کار می کند.  
یک نمودار ساده E/R می تواند به فرم زیر باشد:



ارتباط ممکن است مابین بیش از دو موجودیت باشد (SPJ). اطلاعاتی که از این ارتباط بین سه موجودیت به

دست می آید همیشه لزوماً همان اطلاعاتی نیست که از ارتباط دو به دو موجودیت ها بدست می آید.

مثال:

- ۱- تهیه کننده  $S_1$  قطعه  $P_1$  را تهیه می کند.
  - ۲- قطعه  $P_1$  در پروژه  $J_1$  بکار رفته است.
  - ۳- تهیه کننده  $S_1$  برای پروژه  $J_1$  قطعه تهیه کرده است.
  - ۴- تهیه کننده  $S_1$  قطعه  $P_1$  را برای استفاده در پروژه  $J_1$  تهیه کرده است.
- همیشه از اطلاع ۱ و ۲ و ۳ نمی توان اطلاع ۴ را نتیجه گرفت.

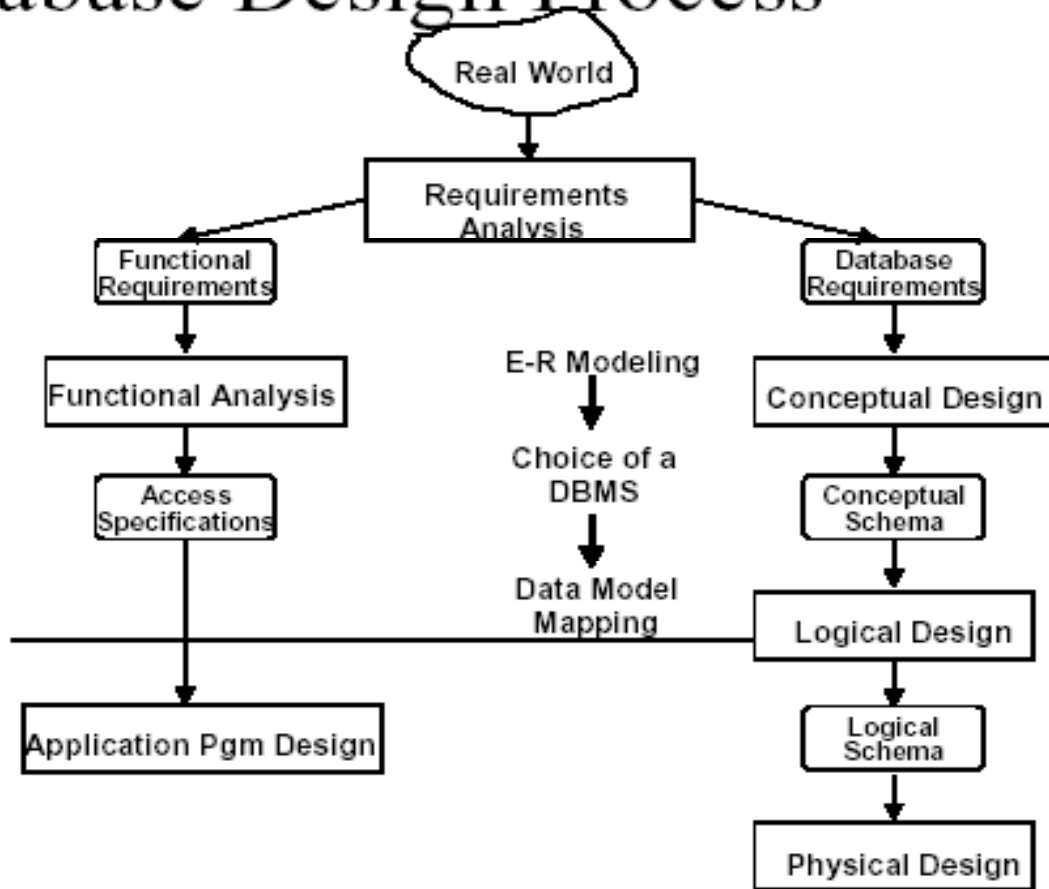
### ۲-۳- اصطلاح دام پیوندی (Connection trap)

اگر از ارتباط بین دو به دوی موجودیتها نتیجه گرفته شود که حتماً ارتباط بین سه موجودیت یا بیشتر از آن بدست آید در اینصورت طراح گرفتار دام پیوندی شده است.

### ۲-۴- طراحی پایگاه داده ها

طراحی یک پایگاه داده مستلزمی مراحل می است که در هر مرحله فعالیتهایی انجام میشود . شکل زیر نمودار ساده شده مراحل طراحی پایگاه داده ها را نشان می دهد.

## Database Design Process



### ۲-۴- طراحی پایگاه داده ها و ابزارهای Case

برای طراحی پایگاه داده ها ابزارهای مختلفی ارائه شده اند . این ابزارها به طراح پایگاه داده کمک میکنند تا در مراحل مختلف مدلسازی و طراحی پایگاه داده ، تصمیم گیری مناسب را انجام دهد. این ابزارها امکان ترسیم نمودار E/R را با استفاده از انتخاب اشیاء از یک جعبه ابزار را بوجود می آورند. بطور کلی مزایای استفاده از این ابزارها را می توان بصورت زیر نام برد :

۱- سادگی فرایند ایجاد نمودارها

۲- تولید خودکار جملات SQL برای تعریف جدولها ، محدودیت ها ، اندیس ها و دیگر اشیاء مدل رابطه ای.

۳- امکان مستند سازی هر موجودیت ، صفت خاصه ، رابطه و محدودیت.

۴-

برخی ابزار ها یی که در حال حاضر استفاده می شوند عبارتند از::

۱- ER STUDIO برای مدلسازی E/R بانک اطلاعاتی

۲- DB Atrisan برای مدیریت پایگاه داده ها و امنیت آن

۳- Oracle Developer 2000 & Designer 2000 برای مدلسازی پایگاه داده و توسعه برنامه های کاربردی

۴- Platinum Enterprise Modeling suite : ER Win , BpWin برای مدلسازی داده ها و پردازش ها

۵- RW Metro برای تبدیل از O-O به مدل رابطه ای.

۶- Rational Rose برای مدلسازی UML و تولید برنامه های کاربردی به زبان جاوا و ++C.

۷- Visio Enterprise Visual Basic برای مدلسازی داده ها و طراحی م مهندسی مجدد.

۸- X Case برای مدلسازی مفهومی .

۹- Case Studio برای مدلسازی E/R بانک اطلاعاتی.

**تمرین:** نمودار E/R هر یک از محیطهای عملیاتی زیر را رسم کنید:

- سیستم اطلاعات یک نمایشگاه بین المللی

- سیستم اطلاعات گیاهان

- سیستم اطلاعات شخصی ( P I S )

- سیستم اطلاعات آزمایشگاه طبی

- سیستم اطلاعات تعمیر و نگهداری کامپیوترها

- سیستم اطلاعات داروخانه یک بیمارستان

- سیستم اطلاعات فعالیتهای کلوب فیلم

- سیستم اطلاعات مسابقات علمی

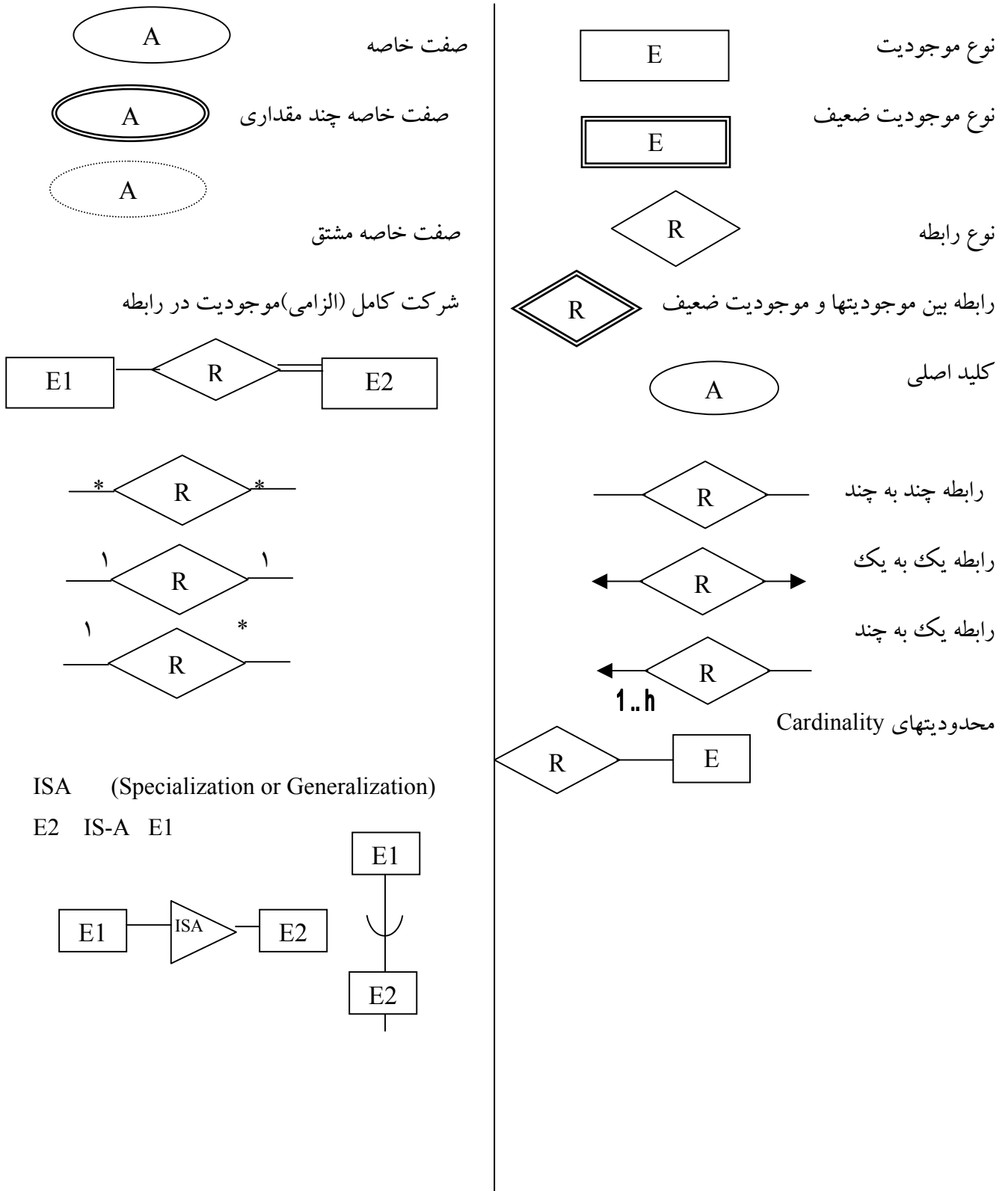
- سیستم اطلاعات یک بانک

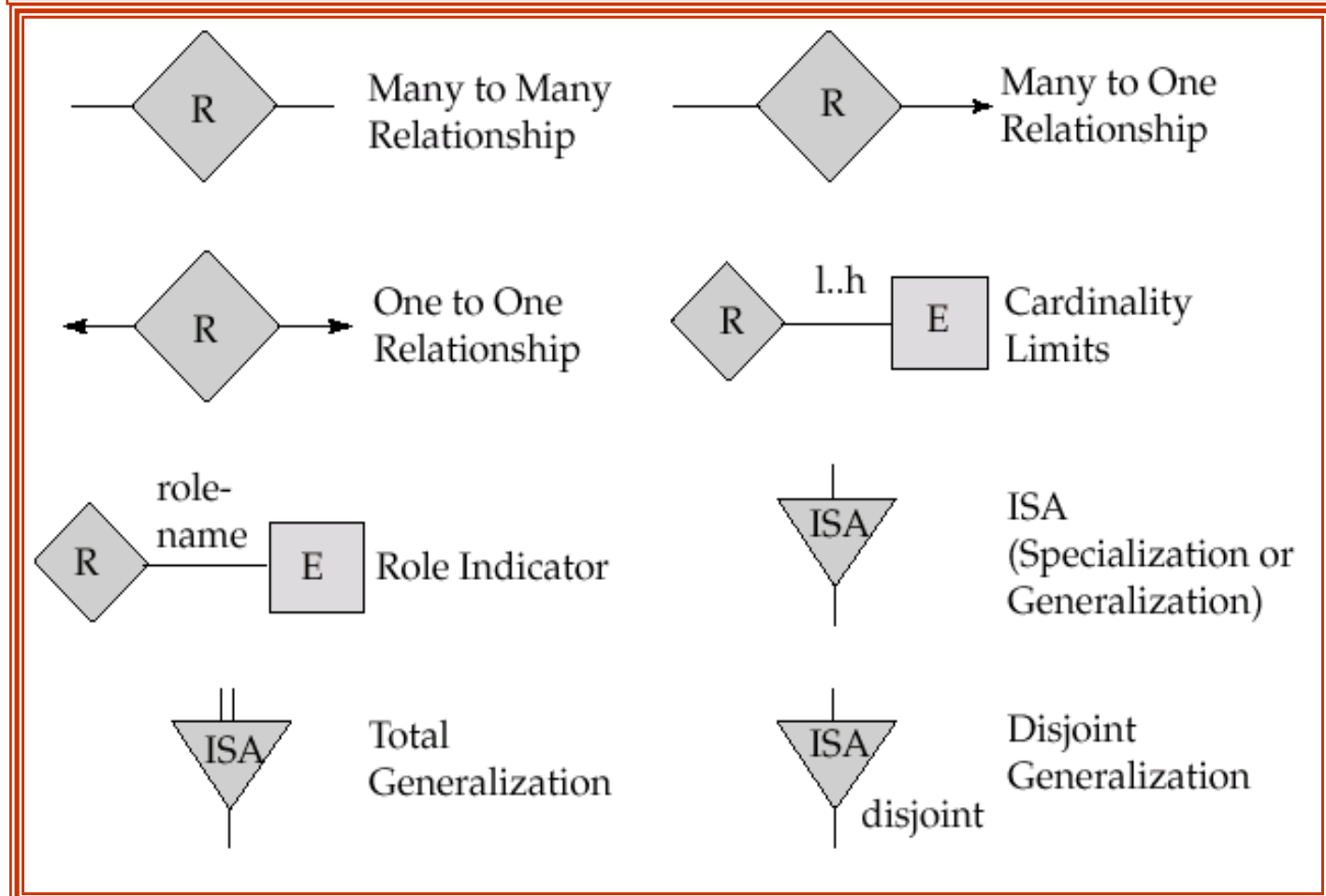
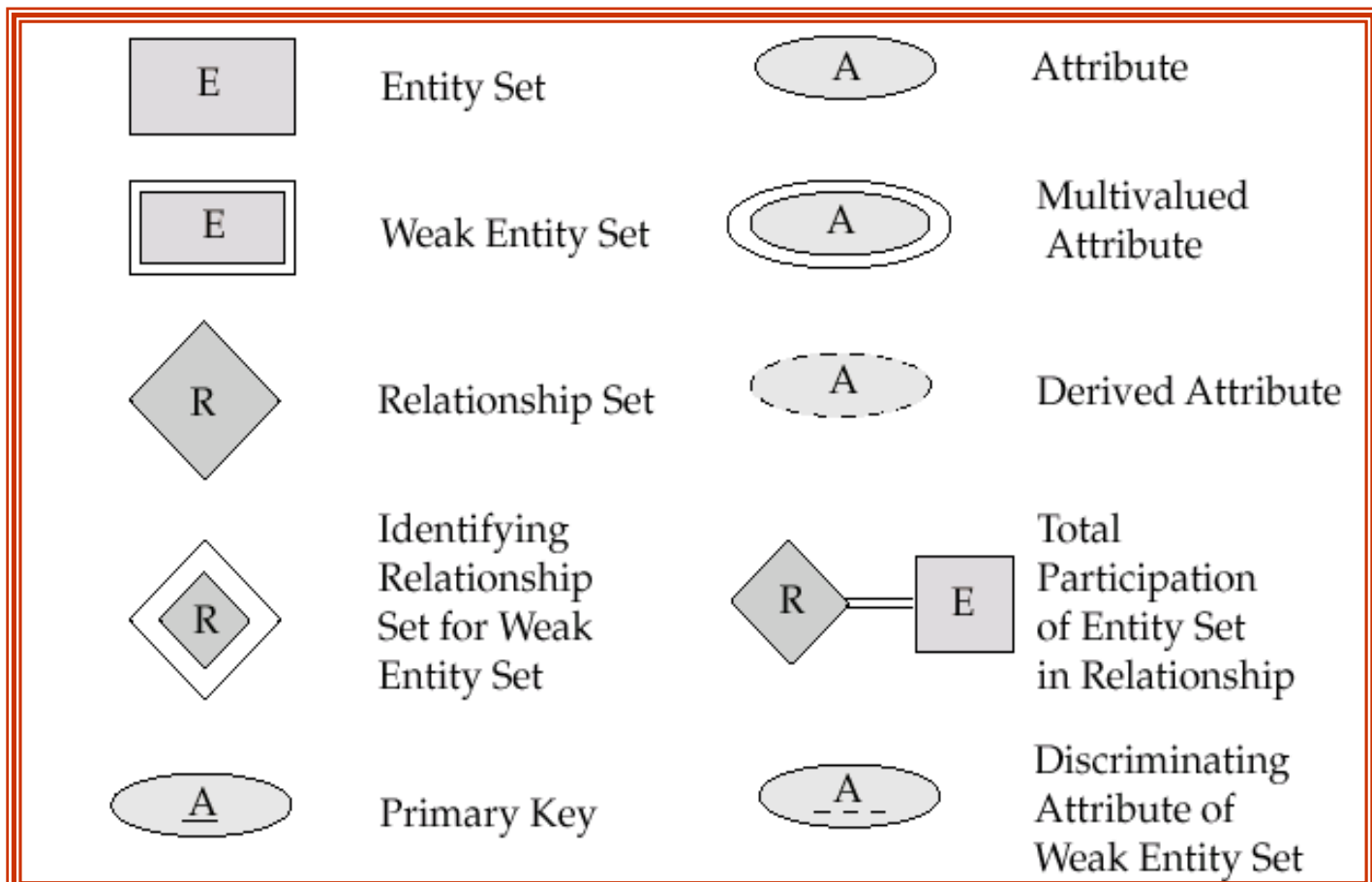
- سیستم اطلاعات موسیقی و موسیقی دانان

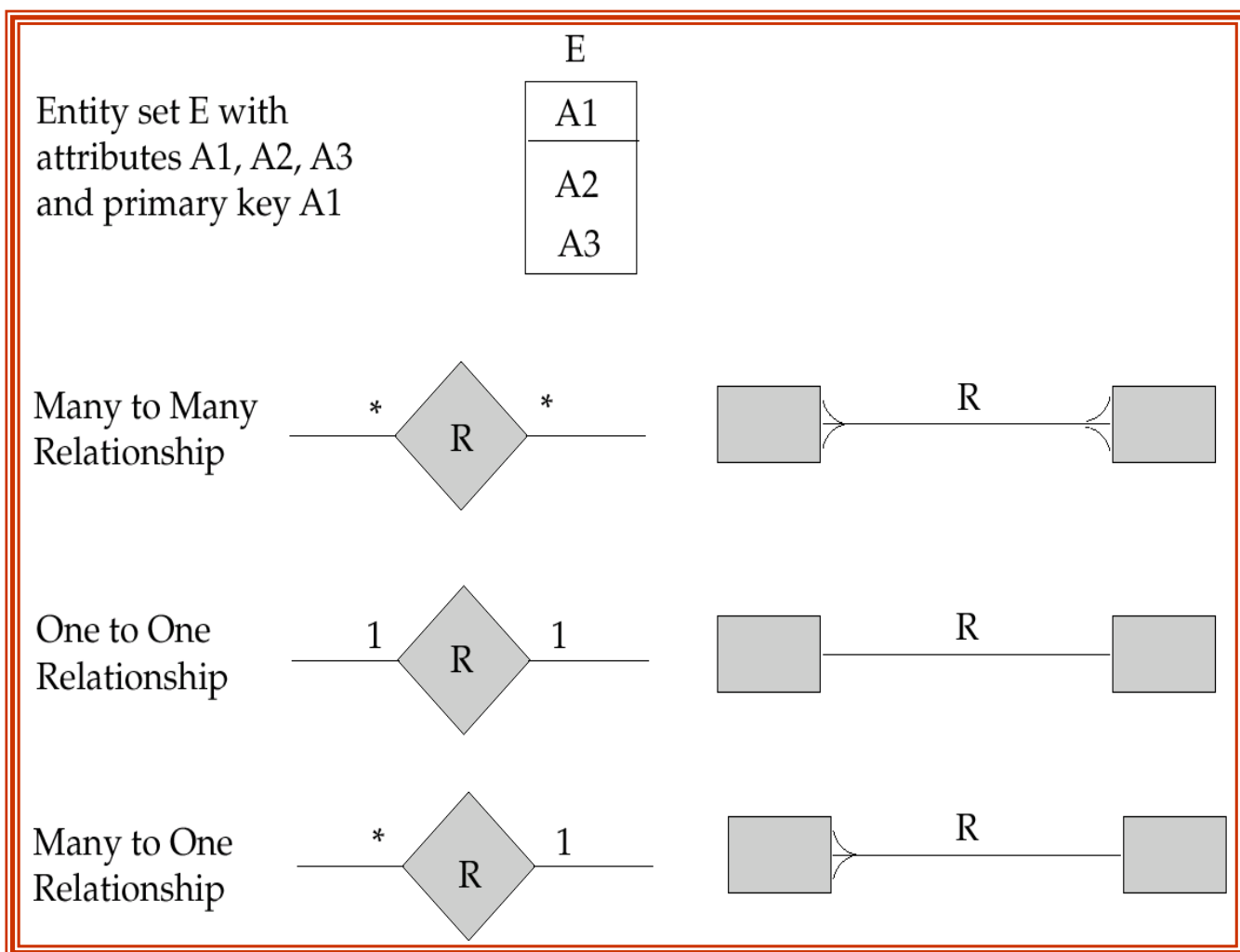
- سیستم اطلاعات نقاشی و نقاشان

- سیستم اطلاعات یک مرکز تحقیقاتی

خلاصه شکلهای بکاررفته در نمودار E/R

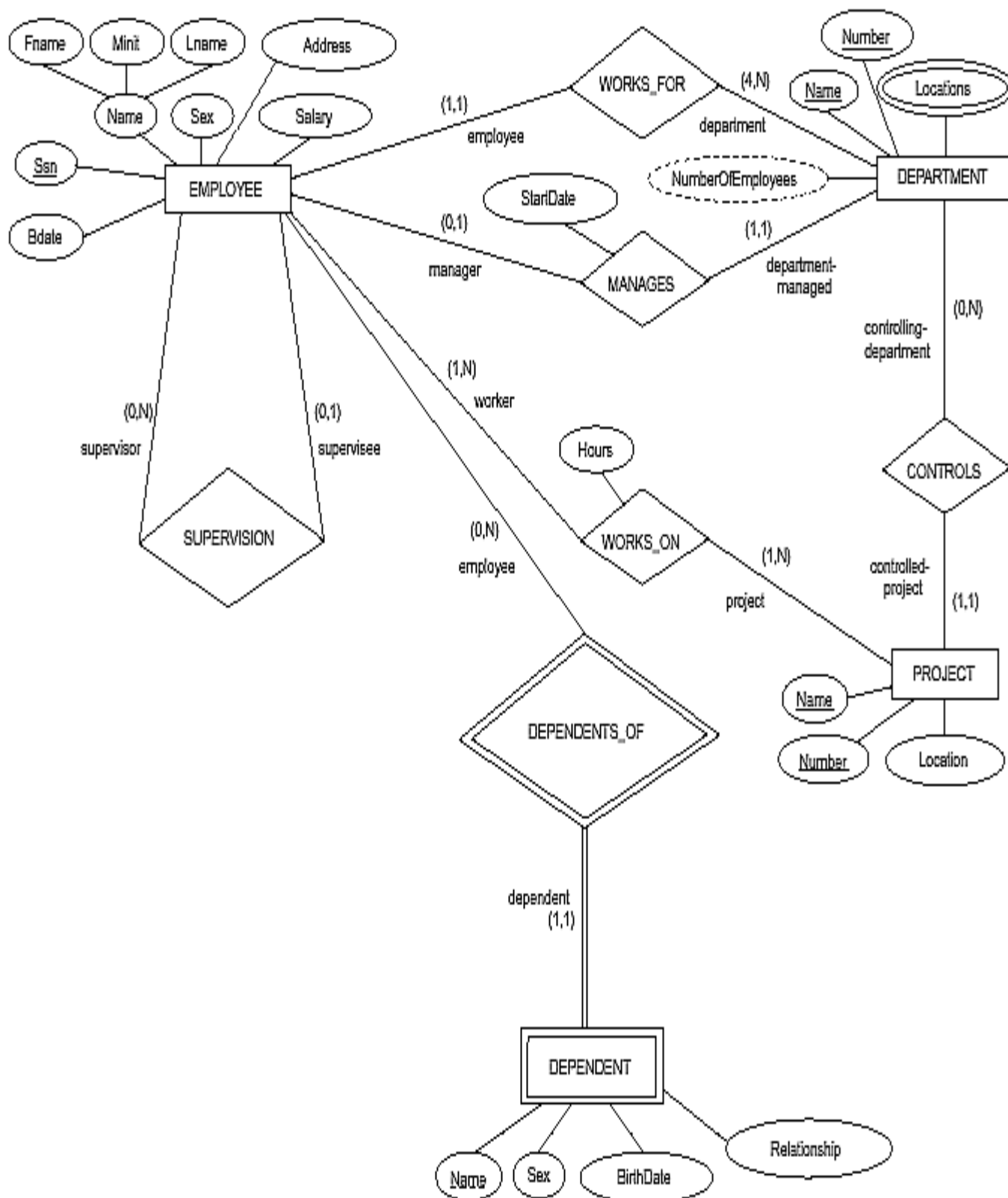






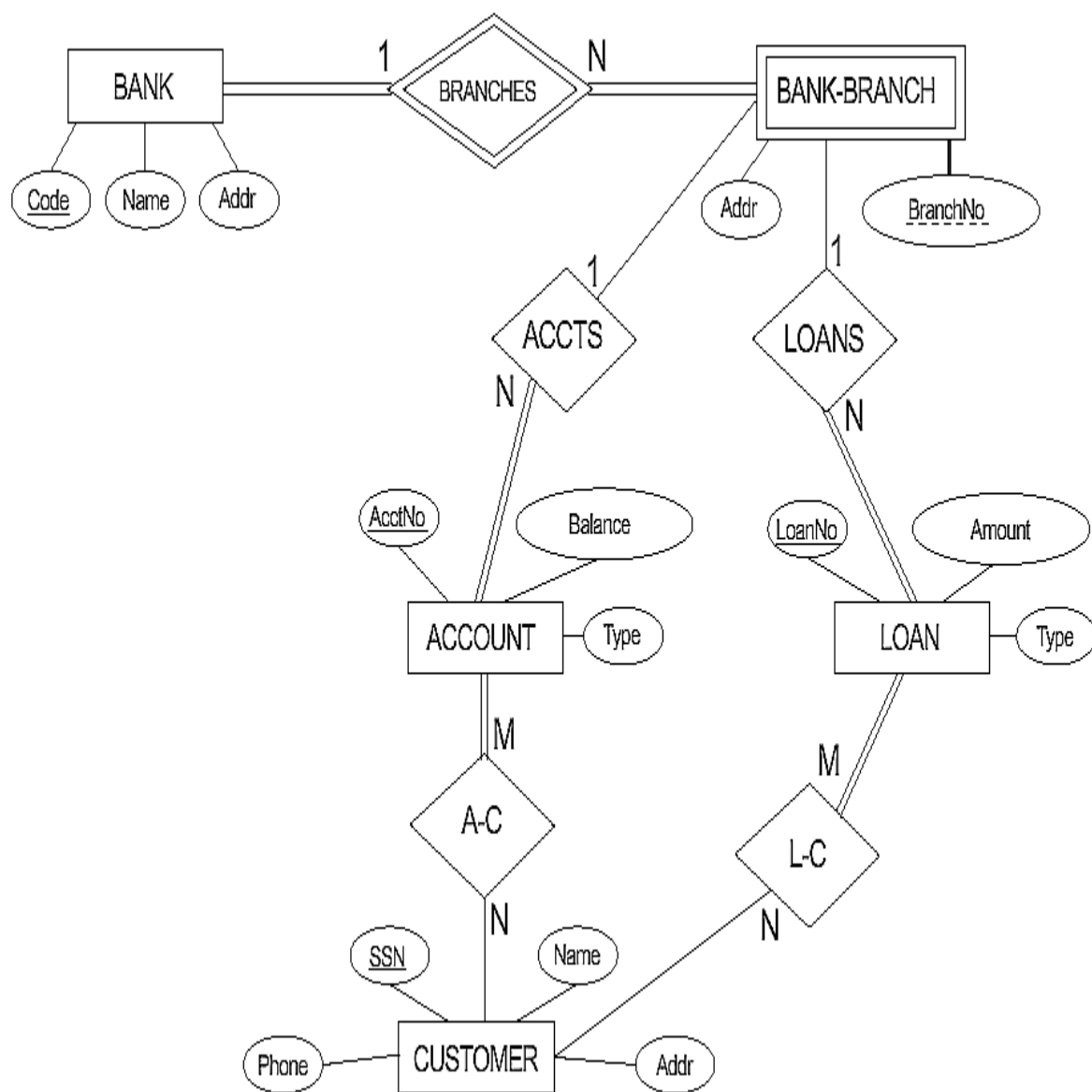


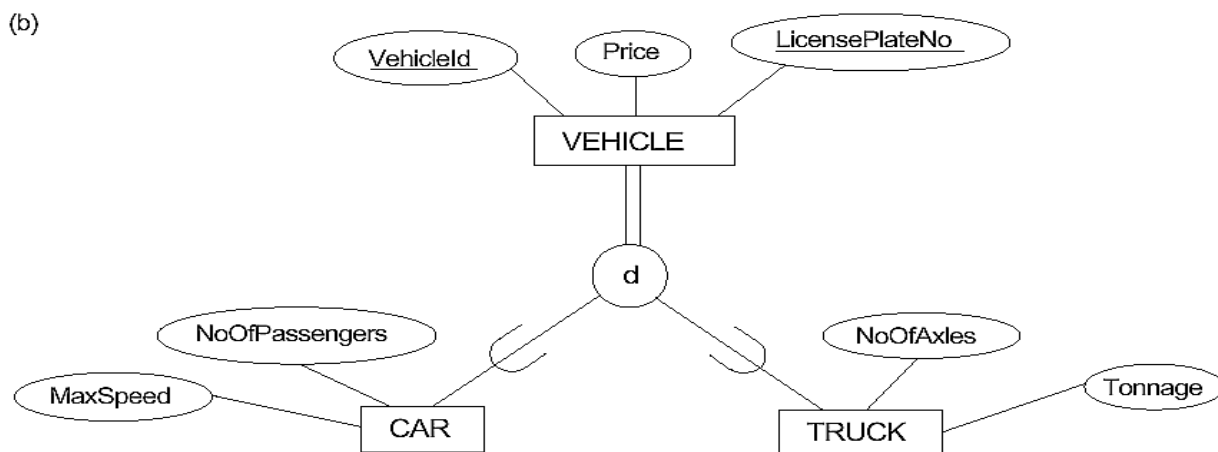
۲-۵- چند مثال از نمودار E/R



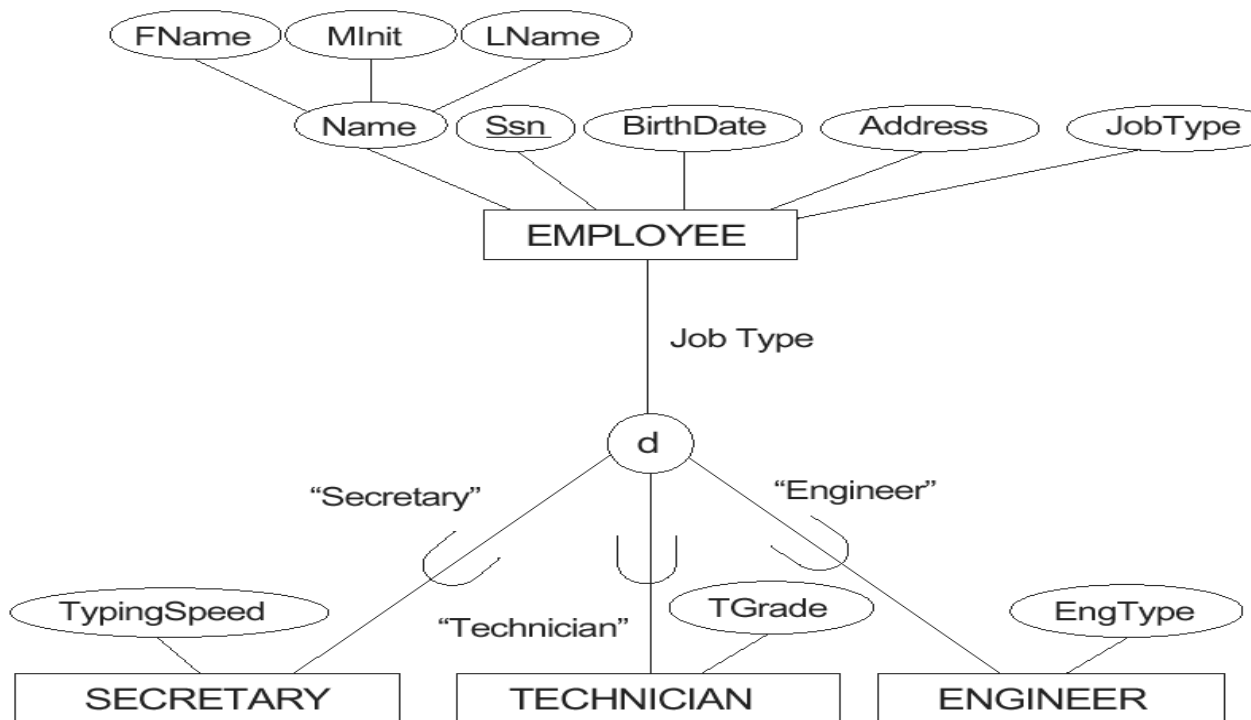
□ مثالی از یک سیستم کارمند - پروژه

□ نمودار E/R یک سیستم بانکی:





مثالی از تخصیص و تعمیم: □



۱- در مدل سازی داده ها با روش E/R گاه مشکلاتی بروز می نماید که از جمله آنها تله ارتباطی را می توان نام برد. دو حالت رایج تر این تله ارتباطی عبارتند از: تله یک چندی و تله شکاف ، با ذکر مثالی این دو حالت را توضیح دهید.

۱۲) یک اژانس مسافرتی قصد دارد با تولید یک DBS داده های مربوط به تمام کارکنان خود و نیز کلیه تورهای داخلی و خارجی خود (اعم از زمینی، هوایی، دریایی و یک روزه) را به همراه مشخصات مسافران و مسوولان هر تور ثبت کند. در پایان هر تور، مسافران برگه نظرخواهی را در مورد کیفیت تور و مسوولان آن پر می کنند. این اژانس برای رزرو بلیط هواپیمای کلیه خطوط هوایی و نیز رزرو اتاق در کلیه هتل های داخلی و خارجی هم اقدام می کند.

الف) یک ERD برای این سیستم رسم کنید و در صورت نیاز مفروضات خود را با ذکر آنها در مدلسازی دخالت دهید.

۳

۱۳) می خواهیم با تولید وب سایت [iranianmoviesdb.com](http://iranianmoviesdb.com)، مشخصات تمامی فیلمها و سینماگران ایرانی را از ابتدا تا کنون ثبت کنیم. مشخصات هر فیلم شامل موارد زیر می باشد:  
نام، سال تولید، لیست عوامل (به تفکیک مسوولیت)، ژانر، خلاصه داستان، میانگین ارزیابی بازدید کنندگان سایت از فیلم (۰ تا ۱۰)، مدت زمان، یادداشتهای نوشته شده بر فیلم (توسط بازدیدکنندگان ثبت شده سایت)، جوایز داخلی و خارجی که فیلم دریافت کرده یا نامزد دریافت آنها بوده، فروش، عکسهای سر صحنه و پشت صحنه، پوستر(ها) و تیزر(ها).

در مورد سینماگران هم می خواهیم داده های زیر را داشته باشیم:  
نام، تاریخ و محل تولد، خلاصه زندگینامه، گالری عکس، لیست تمام فیلمهایی که در آنها فعالیت داشته (به تفکیک سمت وی در آنها) و جوایز داخلی و خارجی که دریافت کرده و یا نامزد دریافت آنها بوده است.

به علاوه می خواهیم در صورت تمایل هر بازدید کننده، با دادن شناسه به وی اطلاعاتی از او ثبت کنیم و به کاربران ثبت شده امکاناتی از قبیل داشتن لیست فیلمها و سینماگران مورد علاقه، ارزیابی فیلمها (۰ تا ۱۰) و نیز نوشتن یادداشت در مورد فیلمها بدهیم.

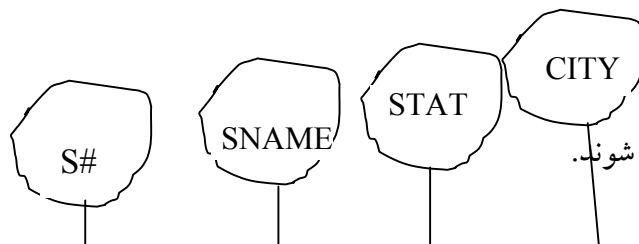
یک ERD برای این سیستم رسم کنید و در صورت نیاز مفروضات خود را با ذکر آنها در مدلسازی دخالت دهید.

## فصل سوم :

### مدل رابطه ای

#### ۳-۱- تعریف رابطه :

رابطه از دو مجموعه تشکیل شده است : یکی موسوم به مجموعه عنوان یا heading رابطه و دیگری مجموعه بدنه یا body . عنوان، مجموعه ای ثابت از صفات خاصه  $A_1, \dots, A_n$  است که این صفات خاصه هر یک مقادیرشان را از یک میدان ( حوزه ) می گیرند. مجموعه بدنه مجموعه ایست متغیر در زمان از چند تایی مقادیر صفات خاصه بنام تاپل ( tuple ). معمولاً در مدل رابطه ای از اصلاح جدول بجای رابطه نیز استفاده می شود. یعنی جدول امکانی است برای نمایش مفهوم رابطه بخاطر تامین و وضوح کاربردی. معمولاً رابطه را بصورت  $r(R)$  نمایش میدهند که به R شمای رابطه نیز گفته میشود.  $R = (A_1, A_2, A_3, \dots)$



#### ۳-۱-۱- تعریف دامنه / میدان :

مجموعه ایست که مقادیر یک صفت خاصه از آن برگرفته می شوند. مثال: موجودیت تهیه کننده را در نظر می گیریم:

S#	SNAME	Status	City
۱۰۰	نام ۱	۲۰	تهران
۲۰۰	نام ۲	۳۰	قزوین
۳۰۰	نام ۳	۵۰	کرج
۲۵۰	نام ۴	۱۵	تبریز

ارتباط بین مدل رابطه ای و نمایش جدولی:

مدل رابطه ای (تئورسین)	ساختار جدولی (طراح)
رابطه	جدول
تاپل	سطر
صفت خاصه	ستون
میدان	مقادیر مجاز یک ستون

وقتی که اسم و مجموعه عنوان رابطه مشخص باشد گویند ذات یا جوهر رابطه ( Intension ) معلوم است. به بدنه رابطه بسط رابطه ( Extension ) نیز گویند.

نکته: ذات رابطه ثابت در زمان است اما بسط رابطه در زمان متغیر است.

### ۳-۱-۲-درجه رابطه:

تعداد صفات خاصه رابطه را درجه آن گویند. اگر درجه رابطه یک باشد رابطه یگانی ، رابطه درجه دو را دوگانی (binary) ، رابطه درجه سه را سه گانی (Ternary) و رابطه با درجه N را Nگانی (N-ray) گویند.

### ۳-۱-۳-کاردینالیتی رابطه:

به تعداد تاپلهای رابطه در یک لحظه از حیات آن کاردینالیتی رابطه گویند. کاردینالیتی رابطه در طول حیات رابطه متغیر است. بعنوان مثال، رابطه از درجه رابطه ۴ میباشد.

### ۳-۱-۴- خصوصیات رابطه:

۱\_ به کمک یک ساختار ساده بنام جدول قابل نمایش است.

۲\_ تاپل تکراری در رابطه وجود ندارد. زیرا بدنه رابطه مجموعه است و در مجموعه عناصر تکراری وجود ندارد.

۳\_ تاپلهادر رابطه نظم خاصی ندارند. این خاصیت نیز از مجموعه بودن بدنه رابطه نتیجه می شود.

۴\_ صفات خاصه نظم ندارند. این خاصیت نیز از مجموعه بودن عنوان رابطه نتیجه می شود

۵\_ عناصر تشکیل دهنده تاپل اتمیک هستند یعنی تجزیه نشدنی می باشند. عبارتی گوئیم یک فقره داده تجزیه نشدنی است اگر نتوان آن را به مقادیر دیگر تجزیه کرد. بعنوان مثال: مقادیر تاریخ ماهیتی غیر اتمیک دارد زیرا از سه جزء ماه ، سال و روز تشکیل شده است . در این مثال خاص اتمیک و یا غیر اتمیک بودن تاریخ بستگی به دید طراح در طراحی دارد بنابراین این دو مفهوم مطلق نیستند و به معنایی که طراح برای داده ها قائل می شود بستگی دارد.

### ۳-۲- مفهوم میدان و نقش آن در عملیات روی بانک:

میدان : مجموعه ای از مقادیر است که یک یا بیش از یک صفت خاصه از آن مقدار می گیرند. میدان در عملیات روی بانک مزایایی دارد که عبارتند از:

۳-۲-۱- سبب ساده تر شدن و کوتاهتر شدن شمای پایگاه داده میگردد. (از نظر تعداد احکام) . زیرا لازم نیست که در تمام رابطه ها ، هر بار مشخصات صفات را بدهیم .

۳-۲-۲\_ کنترل مقداری عملیات در پایگاه.

مقادیر یک صفت خاصه در طول حیات رابطه، از مقادیر میدان برگرفته می شوند. عبارت دیگر باید در میدان وجود داشته باشند. بنابراین به کمک مفهوم میدان می توان عملیات روی بانک را از نظر مقادیر صفات خاصه کنترل کرد.

بطور مثال اگر میدان مقادیر STAUS بصورت  $\{10,20,30,40,50\}$  Domain STATUS = باشد امکان درج اطلاع ( تهران و 60, Sn7, S7 ) در بانک میسر نمی باشد.

۳-۲-۳- امکانی است برای کنترل معنایی پرس و جوها

مثال : شماره قطعاتی را بدهید که وزن آنها برابر تعداد تهیه شده آنها باشد.

صفات وزن و تعداد به اعتبار همنوع بودن قابل مقایسه اند و سیستم می تواند با انجام مقایسه های لازم ، به پرسش کاربر پاسخ دهد. اما این دو صفت به لحاظ مفهومی غیر قابل مقایسه اند، زیرا روی دو میدان ماهیتاً متفاوت تعریف شده اند.

۳-۲-۴- پاسخگویی به بعضی از پرس و جوها را آسان میکند.

اگر امکان تعریف میدان وجود داشته باشد این تعریف وارد کاتالوگ سیستم بعنوان بخشی از شمای ادراکی پایگاه می شود و در شرایطی برخی از کاربران می توانند از آن استفاده کنند.

مثال: درچه رابطه هایی از پایگاه اطلاعاتی از تهیه کنندگان وجود دارد؟

اگر میدان ها تعریف شوند برای پاسخگویی به این پرس و جو فقط مراجعه به کاتالوگ کفایت می کند .

### ۳-۳- مفهوم کلید در مدل رابطه ای:

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

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

### ۳-۳-۱- مفهوم ابر کلید super key

مجموعه ای از یک 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.

### ۳-۳-۱-۴ کلید(نامزد) کاندید: Candidate Key

ابر کلیدی که خاصیتی کاهش ناپذیری داشته باشد کلید کاندید گویند. بعبارت دیگر هر زیر مجموعه از مجموعه عنوان  $(A_i, A_j .. A_k)$  که دو خاصیت زیر داشته باشد کلید کاندید گویند.

۱- یکتایی مقدار (Uniqueness)

به این معنا که در هر لحظه از حیات رابطه مقدار  $(A_i, A_j .. A_k)$  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 مقدار از بین برود.

با توجه به تعاریف می بینیم هر ابرکلید لزوماً کلید کاندید نیست اما هر کلید کاندید جزء مجموعه های ابر کلید رابطه هست.

مثال: در رابطه S صفت خاصه S# کلید کاندید است و در رابطه SP (S#, P#) کلید کاندید است.  
نکته: ۱: رابطه ممکن است بیش از یک کلید کاندید داشته باشد.

نکته ۲: وجود حداقل یک کلید کاندید در رابطه تضمین است زیرا در بدترین حالت با ترکیب تمام صفات خاصه به یکتایی مقدار می رسیم. به رابطه ای که مجموعه عنوانش کلید کاندید آن باشد اصطلاحاً رابطه تمام کلید (All key) نامیده می شود.

نکته ۳: نقش کلید کاندید: امکانی است برای ارجاع به تاپل یعنی نوعی مکانیسم آدرس دهی در سطح تاپل است.

### ۳\_۳\_۲ کلید اصلی: Primary Key

یکی از کلیدهای کاندید است که طراح با توجه به ملاحظات 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 کلیدهای کاندید در پاسخگویی به نیازهای 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 معمولاً در سیستمهای رابطه ای با عبارت Primary Key (Attribute) تعریف میشود.

### ۳\_۳\_۳ - کلید نامزد (بدیل) Alternate Key

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

### ۳\_۳\_۴: 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 )

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



در رابطه department صفت خاصه # dept کلید اصلی است لذا در رابطه Employee کلید خارجی است و نیز صفت خاصه # Emp در جدول Employee کلید اصلی است پس صفت خاصه # Manager - Emp در رابطه department کلید خارجی است و جزئی از کلید اصلی این رابطه هم نیست.

**نکته:** لزومی ندارد Rj از Ri متمایز باشد:

E employe ( Emp # , Ename, Manager -Emp # , Salay )

رابطه: کارمند مدیر است .

**سوال:** نقش کلید خارجی چیست؟

کلید خارجی امکانی است برای ایجاد ارتباط بین تاپلها. بعنوان مثال وجود کلید های خارجی P #, S# در رابطه Sp نمایشگر ارتباطی است که بین تاپلهای رابطه S و تاپلهای رابطه P وجود دارد.

نکته: آیا تنها عامل برقراری ارتباط کلید خارجی است؟

پاسخ منفی است هر صفت خاصه مشترک در عنوان دو رابطه امکانی است برای ایجاد و پیاده سازی نوعی ارتباط.

مثال: S ( S#, ...,city ) , P ( P#, ... city )

وجود City در رابطه امکان ارتباط بین دو موجودیت را بوجود می آورد. در صورتیکه C ity نه کلید خارجی S و نه کلید خارجی P است.

**نکته:** کلید خارجی رابطه را نیز باید به سیستم معرفی نمود . برای این منظور از دستور زیر استفاده میشود :

FOREIGN KEY (Attribute) REFERENCE Relation name

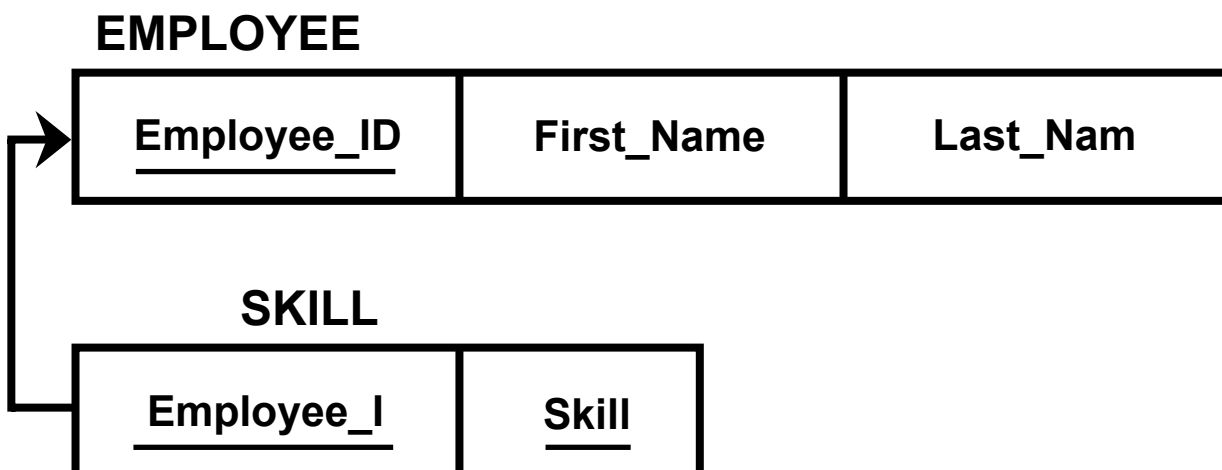
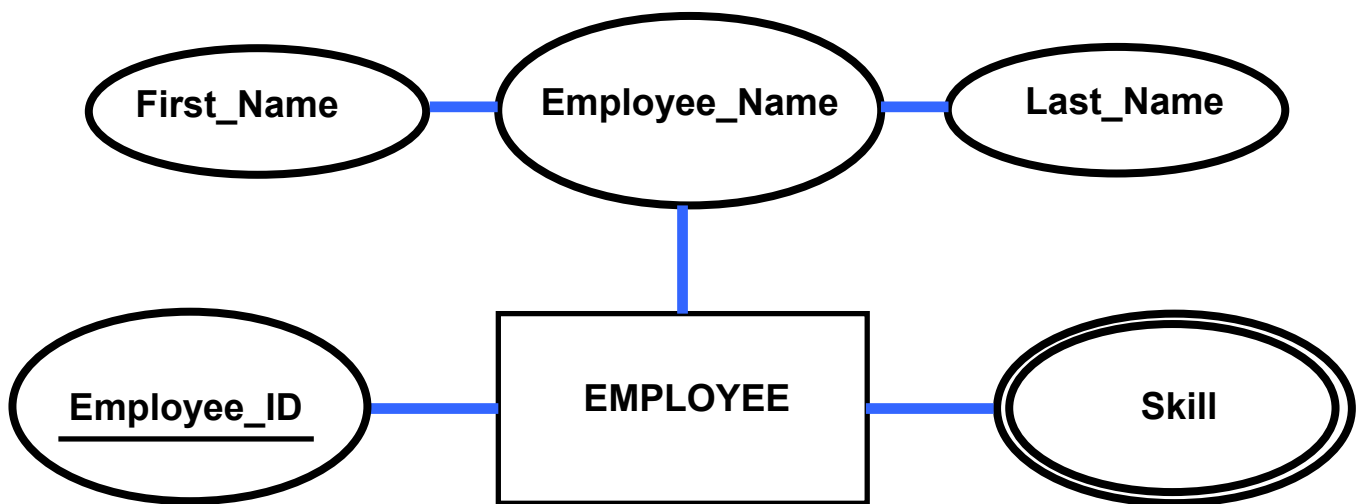
### ۳-۴- تبدیل مدل E/R به مدل رابطه ای

یک پایگاه داده طراحی شده بر اساس مدل موجودیت/ارتباط می تواند توسط مجموعه ای از جدول ها نمایش داده شود. برای تبدیل مدل موجودیت/ارتباط به مدل رابطه ای از یکسری قوانین استفاده می شود که در ادامه آورده شده اند:

**قاعده ۱:** هر موجودیت قوی توسط یک جدول با همان صفات مورد نظر نمایش داده می شود.

- صفات مرکب در مدل رابطه ای وجود نداشته بلکه فقط صفات جزء آن صفات مرکب بطور مجزا در جدول قرار می گیرند.
- صفت چند مقداری M از موجودیت E با یک جدول مجزای EM نشان داده می شود. این جدول دارای ستونهای متناظر با کلید اصلی E و صفت M خواهد بود.

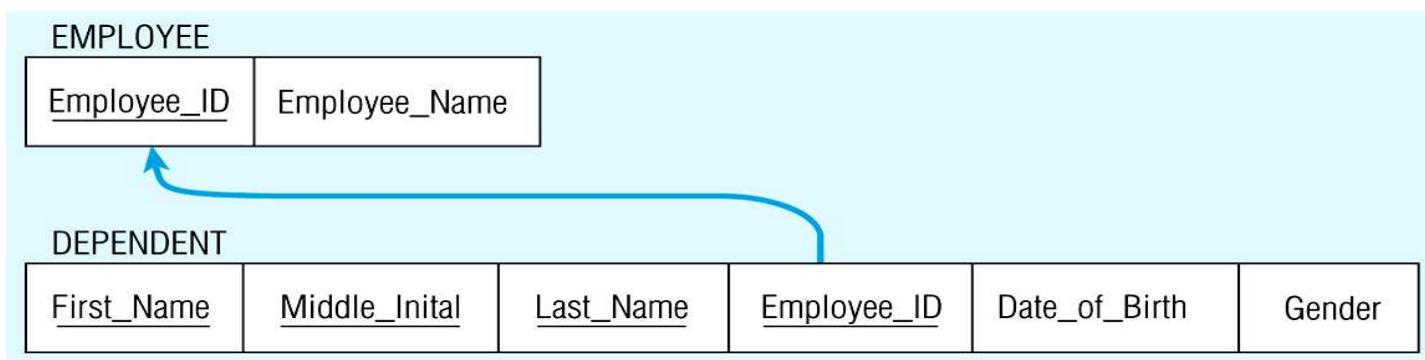
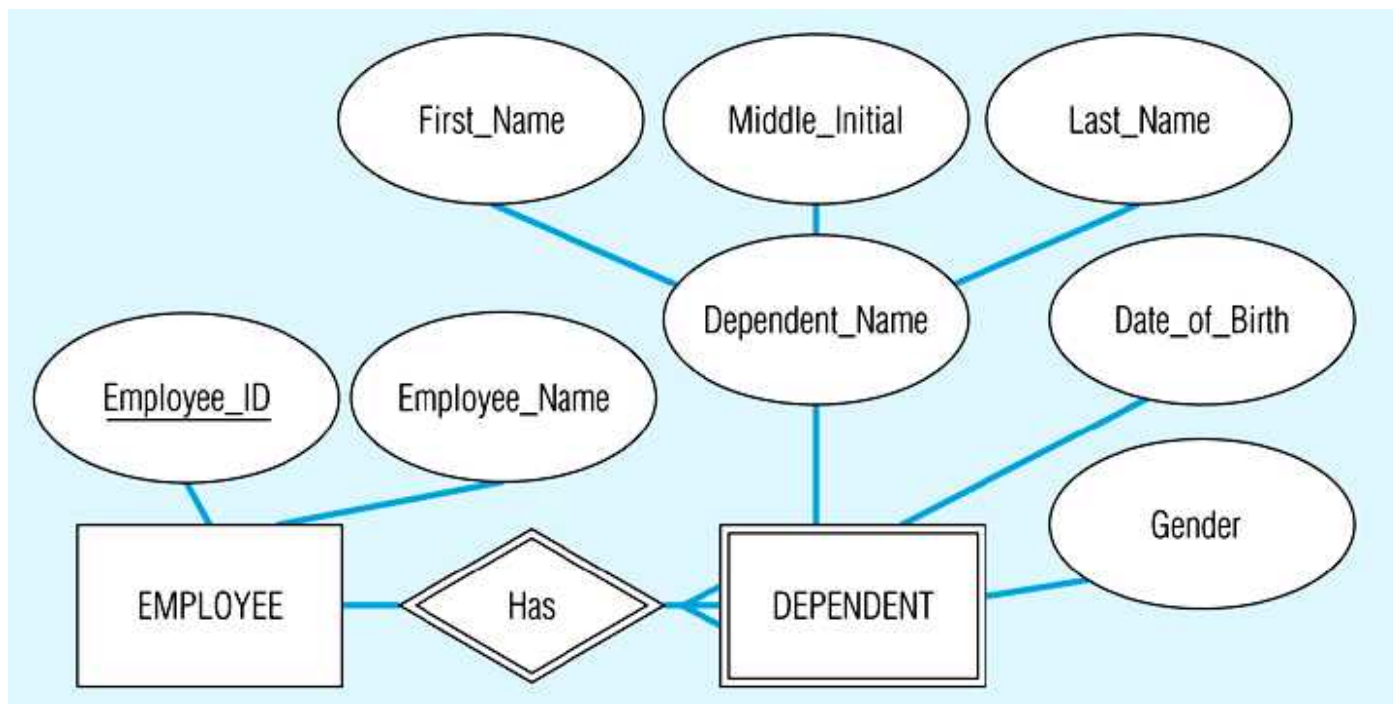
مثال : موجودیت Employee با صفت چند مقداری Skill



**قاعده ۲:** هر موجودیت ضعیف توسط یک جدول با همان صفات مورد نظر به همراه کلید اصلی موجودیت قوی (کلید خارجی این جدول) نمایش داده می شود.

▪ کلید اصلی جدول جدید، ترکیبی است از شناسه موجودیت ضعیف و کلید اصلی موجودیت قوی مورد نظر.

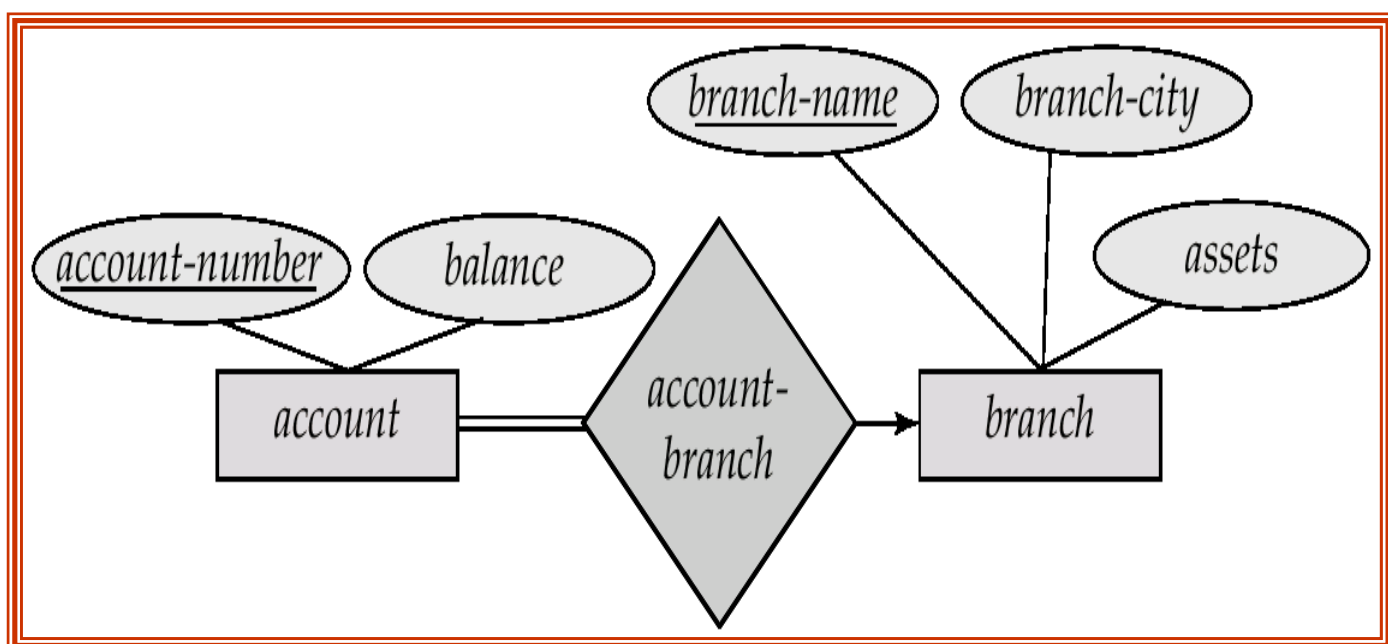
مثال : موجودیت ضعیف DEPENDENT □



### قاعده ۳ : ارتباطات درجه ۲ .

- هر ارتباط درجه دو چند به چند با یک جدول نمایش داده می شود که ستونهایش از کلیدهای اصلی دو موجودیت مشارکت یافته در ارتباط به همراه صفات دیگر مجموعه ارتباط تشکیل می گردند.
- ارتباطات چند به یک یا یک به چند در طرف چند با استفاده از صفت اضافه شده که همان کلید اصلی طرف یک می باشد نمایش داده می شوند. (جدول مجزا نخواهد بود) به عبارت دیگر کلید اصلی طرف یک به عنوان کلید خارجی طرف چند منظور می گردد.

مثال : در شکل زیر ، بجای ایجاد یک جدول برای ارتباط `account-branch` صفت `branchname` به جدول `account` اضافه می شود.

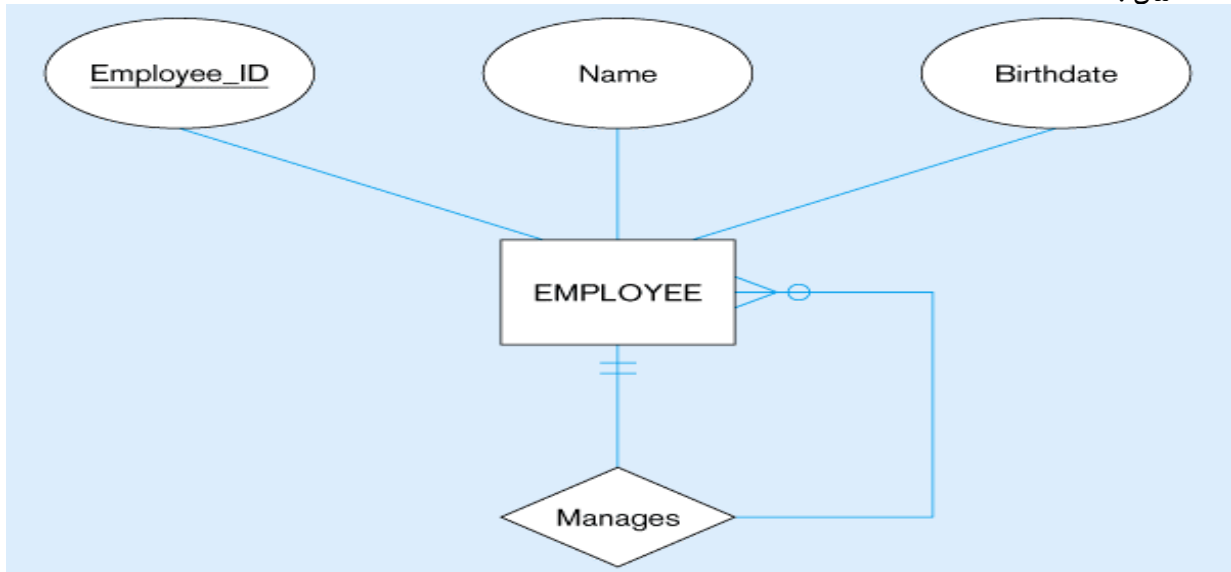


- در ارتباطات یک به یک ، هر طرفی می تواند به عنوان چند انتخاب شود. بدین معنی که کلید اصلی طرف مهمتر به عنوان کلید خارجی طرف کم اهمیت تر اضافه می شود.

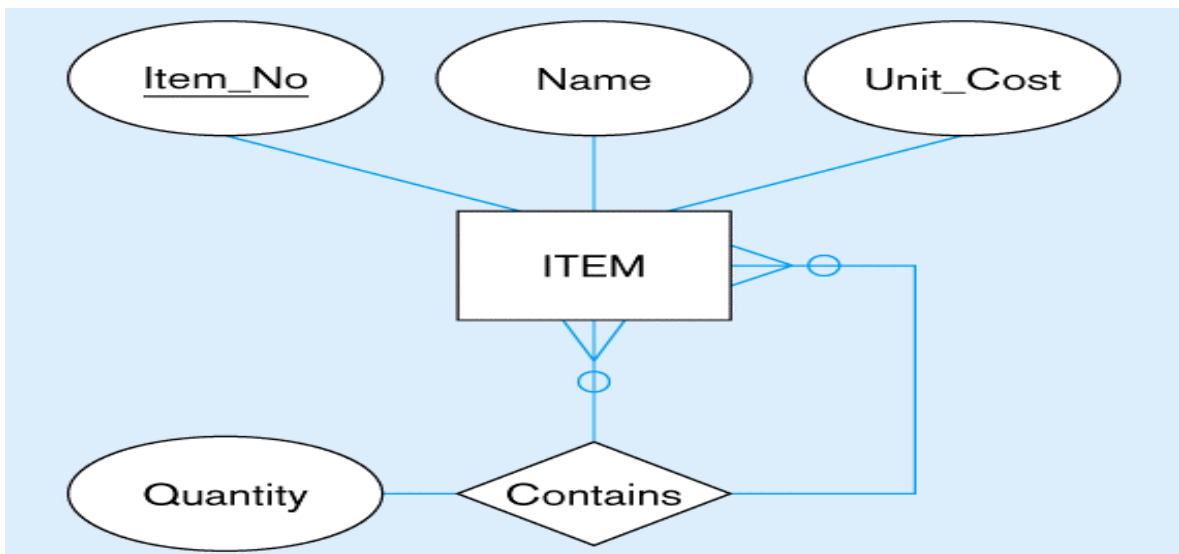
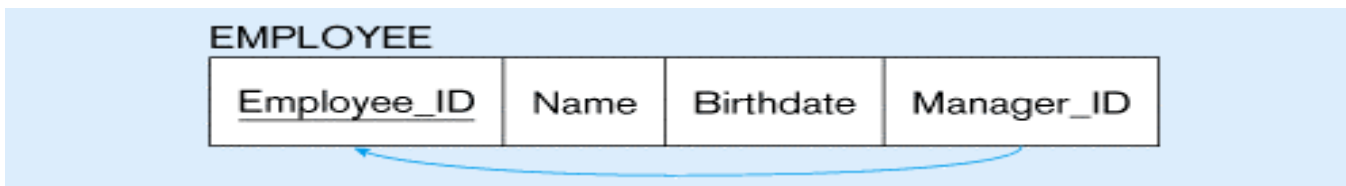
### قاعده ۴ : ارتباطات درجه یک (بازگشتی).

- هر ارتباط درجه یک چند به چند با یک جدول نمایش داده می شود که کلید اصلی اش دو فیلد برگرفته از کلید اصلی موجودیت است که با تغییر نام یکی از آنها از کلید اصلی موجودیت مرتبط تشکیل می شود.
- در ارتباط بازگشتی یک به چند ، کلید خارجی بازگشتی در همان جدول مربوط به موجودیت (با تغییر نام کلید اصلی) داریم.

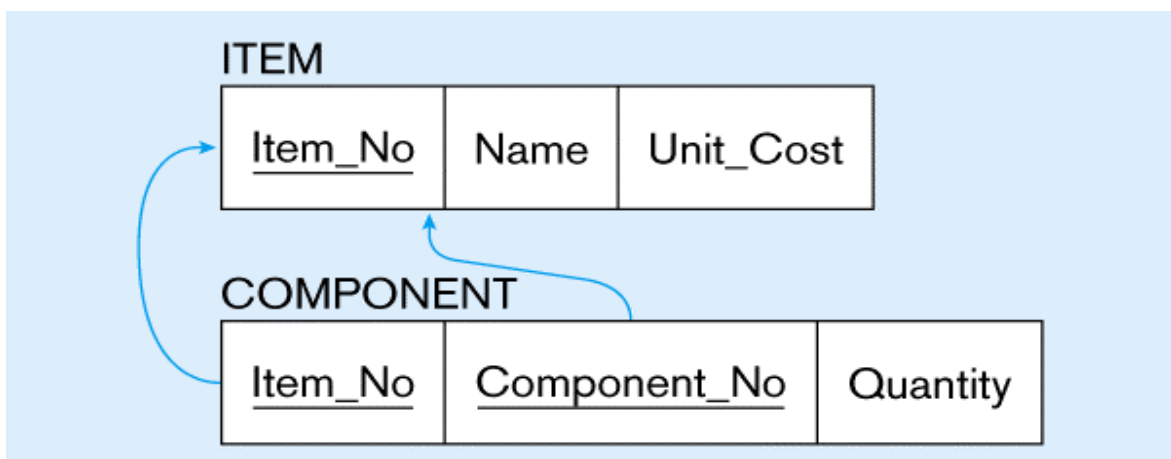
■ مثال :



EMPLOYEE با کلید خارجی، بازگشته،

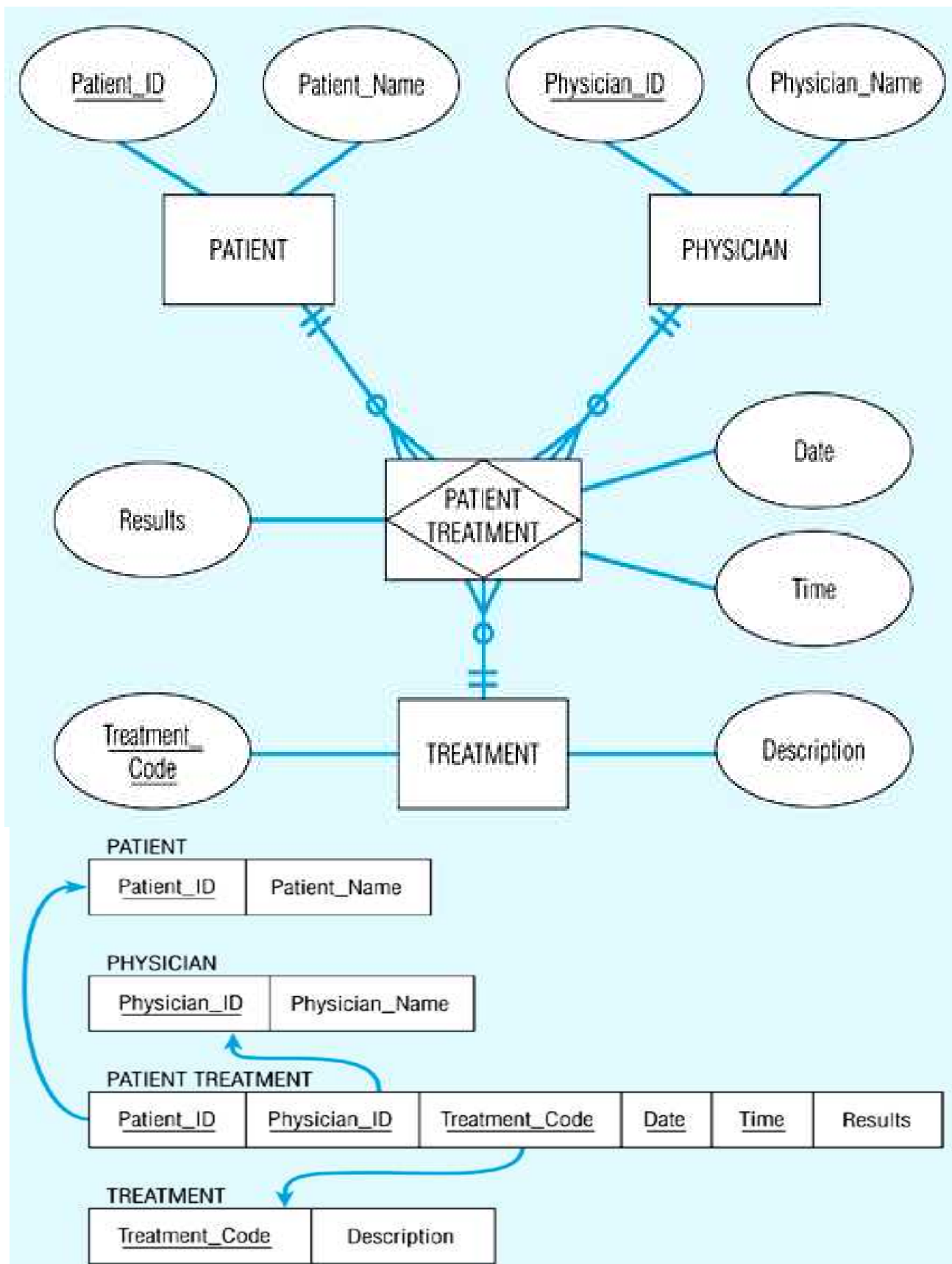


ب) دوجدول بدست آمده



قاعده ۵ : ارتباطات درجه ۳ و بیشتر .

- هر ارتباط n تایی با یک جدول نمایش داده می شود که کلید اصلی اش n فیلد برگرفته از کلید اصلی موجودیتهای مشارکت یافته در ارتباط n تایی می باشد.



قاعده ۶ : ارتباطات تعمیم/تخصیص .

- روش اول: بازای هر موجودیت سطح بالا و پایین یک جدول جداگانه در نظر گرفته شده که در جدول سطح پایین کلید اصلی موجودیت سطح بالاتر به عنوان کلید خارجی قرار می گیرد.

table      table attributes  
*person* (*name, street, city* )  
*customer* (*name, credit-rating*)  
*employee*(*name, salary*)

نکته : اشکال این روش ؟ جمع آوری اطلاعات مستلزم مراجعه به دو جدول است .

- روش دوم : بازای هر موجودیت صفات کلید اصلی و غیره موجودیت سطح بالاتر در موجودیت سطح پایینتر قرار می گیرد. در صورتی تعمیم کامل باشد نیازی به وجود جدول سطح بالاتر نخواهد بود.

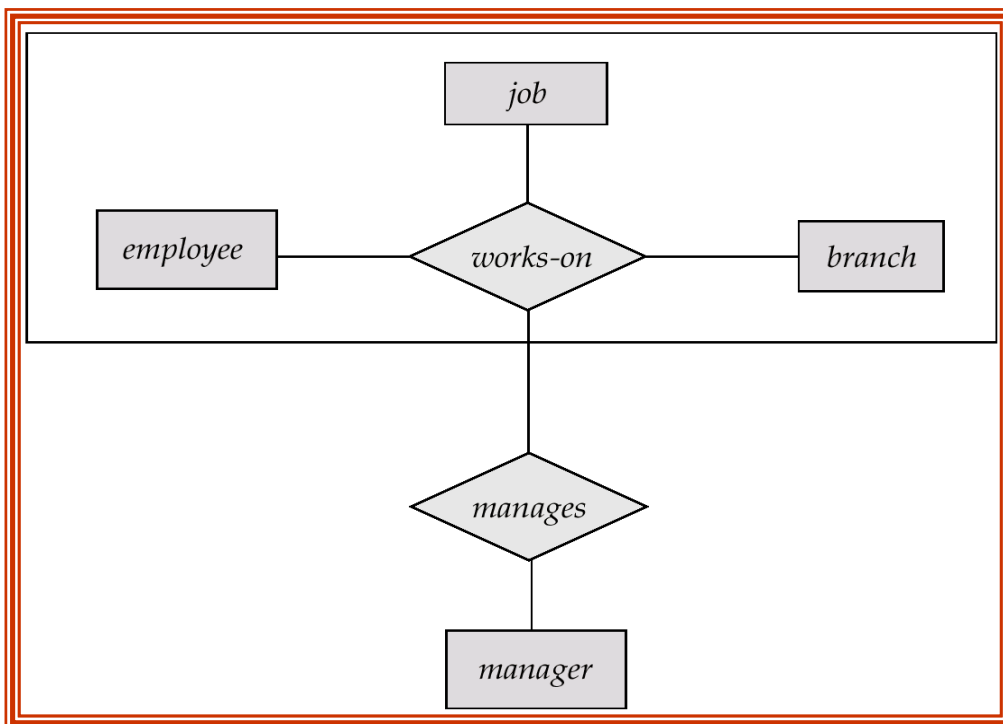
table      table attributes  
*person*      (*name, street, city*)  
*customer* ( *name, street, city, credit-rating*)  
*employee* ( *name, street, city, salary*)

نکته : اشکال این روش ؟ افزونگی

- **قاعده ۷: تجمع :** برای نمایش تجمع ، یک جدول شامل کلید اصلی ارتباط تجمع شده ، کلید اصلی مجموعه موجودیت مشارکت یافته ، صفات مورد نظر ایجاد می شود.

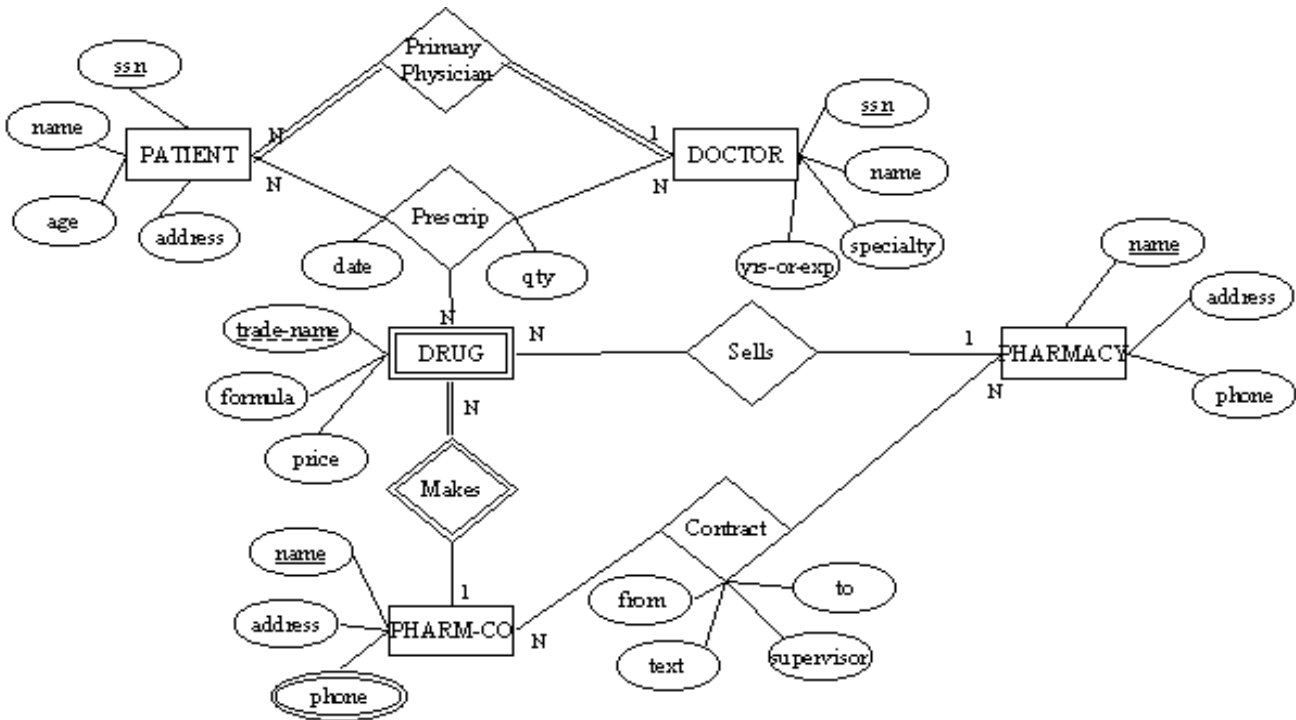
مثال : رابطه تجمع manages بین ارتباطات *manager* , *Works-on* بصورت یک جدول زیر است :

*manages*(*employee-id, branch-name, title, manager-name*)

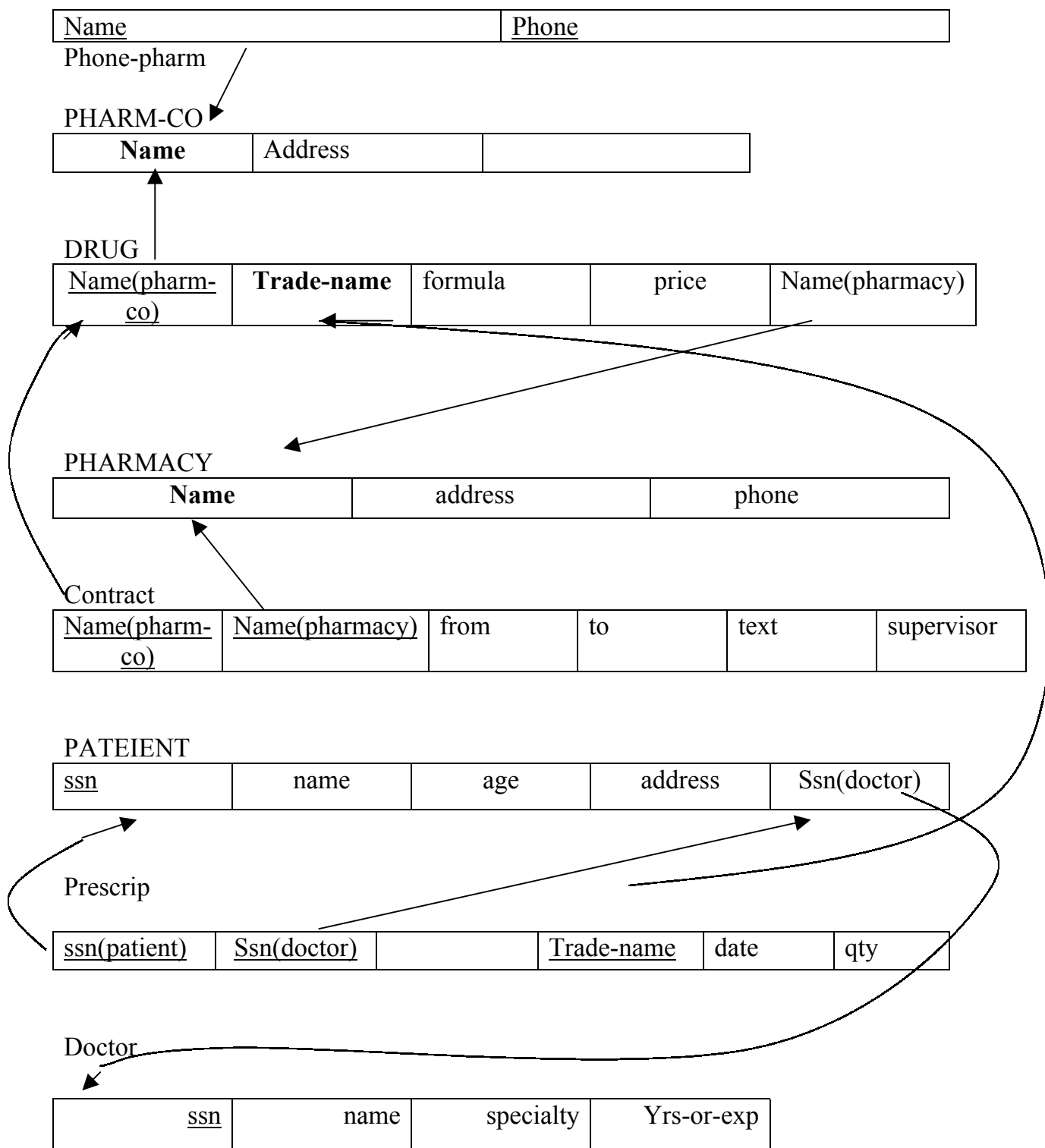


مثال: ۴- نمودار E/R زیر را در نظر بگیرید سپس جداول پایگاه داده را بر اساس نمودار استخراج نمایید. (کلیدهای خارجی و اصلی مشخص کردند).

PATIENT : بیمار  
 DOCTOR : پزشک  
 DRUG : دارو  
 PHARMACY : داروخانه  
 PHARM-CO : شرکت دارو سازی







### ۳-۵- قوانین جامعیت در سیستم های رابطه ای:

جامعیت پایگاه داده یعنی : صحت ، دقت و سازگاری داده های ذخیره شده در پایگاه در تمام لحظات . هر سیستم مدیریت پایگاه داده ها باید بتواند جامعیت پایگاه داده ها را کنترل و تضمین نماید. برای کنترل و تضمین جامعیت ، نیاز به مجموعه ای از قواعد و محدودیتهاست که در یک محیط عملیاتی خاص روی داده های همان محیط باید اعمال شوند و بنام قوانین جامعیتی خوانده می شوند. این قواعد بطور کلی به دو دسته تقسیم می شوند.

#### الف - قواعد خاص یک محیط (کاربری)

قواعدی هستند که توسط یک کاربر مجاز تعریف می شوند . این قواعد در مورد یک پایگاه داده خاص مطرح شده و عمومیت ندارند. گاه به این قواعد ، قواعد محیطی وابسته به داده و یا محدودیتهای جامعیت معنایی نیز می گویند. بطور مثال ، در بانک اطلاعات تهیه کنندگان و قطعات می توان قاعده ای را بصورت زیر تعریف کرد:

مقدار Qty همیشه بین صفر و ۱۰۰۰۰ باشد. وجود چنین قاعده ای در بانک ایجاب می کند که سیستم از انجام هرگونه عملی که سبب می شود مقدار Qty خارج از طیف مقادیر بشود ، جلوگیری کند.

در سیستمهای موجود برای معرفی این قواعد از مکانیسم اظهار (Assertion) استفاده میشود. اظهار مجموعه ای از شرط و یا شرایطی است که همیشه باید در روی عملیات پایگاه رعایت شوند. دستور تعریف اظهار بصورت زیر می باشد

CREATE ASSERTION <assertion name > CHECK < Predicate>:

#### ب - قواعد جامعیت عام

قواعد عام قواعدی هستند که بستگی به سیستم و بانک خاص ندارند بلکه در مدل رابطه ای مطرح بوده و قابل اعمال در هر سیستم رابطه ای هستند. به این قواعد متا قاعده نیز گفته میشود.

در مدل رابطه ای دو قاعده جامعیت وجود دارد: یکی ناظر به کلید اصلی و دیگری ناظر به کلید خارجی این قواعد عبارتند از :

۱- قاعده جامعیت موجودیتی Entity Integrity Rule

۲- قاعده جامعیت ارجاعی Referential Integrity Rule

### ۳-۵-۱- قاعده جامعیت موجودیتی: C1

هیچ جز تشکیل دهنده کلید اصلی نباید دارای " مقدار هیچ (NuLL) باشد.

NullValue مفهومی در بانک اطلاعاتی است که در واقع مقداری است که برای نمایش مقدار ناشناخته یا مقدار غیر قابل اعمال بکار می رود و با فضای خالی و صفر فرق دارد.

: NullValue

۱- مقدار ناشناخته: مثل کارمندی که کد اداره اش مشخص نیست و یا خانه ای در کوچه ای ساخته شد و هنوز پلاک شهرداری ندارد.

۲- مقدار غیر قابل اعمال (برای یک صفت فاصله) مثل مشخصات همسر کارمندی که مجرد است.

• دلیل قاعده جامعیت موجودیتی:

چون کلید اصلی شناسه تاپل است و از سوی دیگر شناسه یکی نمونه مشخص و متمایز از یک نوع موجودیت است چگونه می تواند عامل تمایز خودش ناشناخته باشد.

این قاعده از طریق معرفی کلید اصلی در تعریف رابطه بر سیستم اعمال می شود.

۳-۵-۲- قاعده جامعیت ارجاعی : C2

اگر صفت خاصه  $A_i$  از رابطه  $R_2$  ، کلید خارجی در این رابطه باشد (که معنایش این است در رابطه دیگر کلید اصلی است)

صفت خاصه  $A_i$ :

۱- می تواند هیچ مقدار داشته باشد در غیر اینصورت

۲- حتماً باید مقدار داشته باشد که در رابطه ای که در آنجا کلید اصلی است وجود داشته باشد.

به رابطه  $R_2$  رابطه ارجاع دهنده یا رجوع کننده Referencing و به رابطه دیگر رابطه مرجع یا مقصد گویند.

مثال:  $sp.s\#$  کلید خارجی در  $Sp$  است. مقادیر  $S\#$  در  $Sp$  باید چنان باشند که حتماً باید در  $s$  وجود داشته باشند.

نکته: در این مثال خاص  $S\#$  در  $Sp$  نمی تواند مقدار Null داشته باشد زیرا خود جزئی از کلید اصلی است.

• دلیل توجیه کننده قاعده جامعیت ارجاعی:

نمی توان به نمونه موجودیتی ارجاع داد که در جهان واقعی وجود ندارد.

این قاعده از طریق حکم کلید خارجی و تعریف رابطه به سیستم اعلام می شود.

۳-۵-۳- تبعات قواعد جامعیت:

قواعد جامعیت باید در تمام مدت حیات رابطه های یک محیط عملیاتی برقرار بوده، رعایت شوند از آنجائیکه این دو قاعده ناظر بر وضعیت بانک هستند. هر وضعیتی از بانک که در آن این قواعد رعایت نشده باشند ناصحیح تلقی می گردد.

Delete From S

فرض کنید دستور `Where S# = 'S2'` در خواست شده باشد.

و سیستم بخواهد تاپل  $S_2$  را از  $S$  حذف کند در اینصورت قاعده دوم خدشه دار می شود مگر اینکه سیستم روش خاصی در حل این مشکل داشته باشد.

ارجاع  $Sp$  به  $S$  مشکل خواهد داشت زیرا  $S_2$  وجود ندارد.

S مرجع

S#
S <sub>1</sub>
S <sub>2</sub>
S <sub>3</sub>
S <sub>4</sub>

SP رجوع کننده

S#	P#	Qty
S <sub>1</sub>	P <sub>1</sub>	100
S <sub>2</sub>	P <sub>1</sub>	70
S <sub>1</sub>	P <sub>3</sub>	200
S <sub>4</sub>	P <sub>2</sub>	60

برای حفظ قاعده جامعیت ارجاعی چهار روش مشهور است:

### الف - روش Restricted (محدود شده) ویا حذف تعویقی

در این روش اگر درخواست حذف تاپلی از رابطه مرجع شود تا زمانیکه تاپلهایی در رابطه رجوع کننده وجود دارند که به این تاپل ارجاع می دهند عمل حذف انجام نمی شود.

### ب - روش آبشاری یا تسلسلی Cascaded

در این روش باحذف یک تاپل از رابطه مرجع تمام تاپلهای رجوع کننده به آن تاپل نیز حذف می شوند.

بطور مثال اگر درخواست **Delete From S Where S = 'S2'** بیاید پس از اجرای آن حکم بایستی حکم زیر نیز اجرا شود :

**Delete From Sp  
Where S# = 'S2'**

این حکم ممکن است بطور اتوماتیک توسط D BMS تولید شود و ممکن است لازم باشد طراح آن را بنویسد لازم به ذکر است اگر دستور دوم را داشته باشیم عکس آن صادق نیست یعنی نیاز به حذف در S نمی باشد.

### ج - روش هیچ مقدار گذاری Nullified

در این روش اگر درخواست شود تاپلی از رابطه مرجع حذف شود، این حذف انجام می شود اما در پی آن مقدار کلید خارجی در رابطه ارجاع دهنده NULL گذاشته می شود. (بشرط اینکه کلید خارجی جز کلید اصلی نباشد)

مثال:

Delete From S  
Where S# = 'S2'  
↓  
Update SP  
Set S# = Null  
Where S# = 'S2'

توجه : در مثال بالا نمی توان روش هیچ مقدار گذاری را انجام داد زیرا # S در SP جز کلید اصلی است.  
نکته : به مجموعه جداول و تعاریف کلید اصلی و کلیدهای دیگر و محدودیتها نیز شمای رابطه گفته می شود.

### د - روش مقدار گذاری با مقدار پیش فرض SET TO DEFAULT

در این روش ، با حذف تاپل مرجع ، کلید خارجی در تاپل های رجوع کننده به آن با مقدار پیش فرض مقدار گذاری میشود.

### ۳-۵-۴- راههای اعمال قواعد جامعیت :

بطور کلی میتوان راههای اعمال قواعد جامعیت را بصورت زیر نام برد :

۱- معرفی کلید اصلی

۲- اعلام صفت هیچمقدار ناپذیر

۳- معرفی کلید خارجی

۴- اعلان محدودیتهای مورد نظر در شمای پایگاه داده

۵- نوشتن راه انداز (Trigger)

راه انداز مکانیسمی است برای راه اندازی اجرای یک عمل در پی انجام یک عمل دیگر . به بیان دیگر در صورت بروز یک رویداد ، عملی باید انجام شود تا سازگاری پایگاه داده تامین گردد. مثلاً در پی انجام یک عمل بهنگام سازی در تاپلی از یک رابطه ، تاپل (هایی) از رابطه های دیگر نیز بهنگام در می آیند.

۶- معرفی میدان و مقادیر آن

۷- معرفی وابستگی های تابعی بین صفات خاصه (به فصل هفتم مراجعه شود).

### ۶-۳- مشخصات سیستم های رابطه ای:

درجه یا میزان یا حد رابطه بودن سیستمهای رابطه ای متفاوت است. از نظر C ODD سیستمی در حداقل رابطه ای است. اگر :

۱- پایگاه جدولی را برای کاربر تامین کند.

۲- سه عمل اساسی از عملیات در رابطه ها را داشته باشد: عمل گزینش یا تحدید ، عمل پرتو و عمل پیوند اما یک سیستم کاملاً رابطه ای باید خصوصیات زیر را داشته باشد:

۱- پایگاه رابطه ای را ایجاد کند: امکانات کامل تعریف انواع رابطه ها را داشته باشد.

۲- امکانات کامل DML رابطه ای

۳- امکان تعریف مجموعه کاملی از قوانین جامعیت بشرح دیده شده را داشته باشد.

• یک پایگاه داده رابطه ای دارای سه مؤلفه اساسی است :

۱- Objects از نظر ساختاری رابطه اند

۲- Operators عملگرهای رابطه ای

۳- Rules قوانین جامعیت

### تمرین :

۱- نحوه تعریف انواع کلید را در یک سیستم رابطه ای بررسی نمایید :

۲- راههای قواعد جامعیت را در یک سیستم رابطه ای بررسی نمایید.

## فصل چهارم : مدل رابطه ای - عملیات روی رابطه ها

۴-۱- مقدمه :

زبان کار با داده ها دارای مجموعه ایست از عملگرها که در کادر مدل داده ای عمل می کنند . در سیستم بانک رابطه ای عملگرهای عمل کننده روی رابطه عملگرهای جبر رابطه ای و محاسبات رابطه ای می باشند.

### ۴-۲- جبر رابطه ای :

Codd در مقاله خود ۸ عملگر برای کار با رابطه ها تعریف کرده است که این عملگرها به دو دسته تقسیم می شوند:

الف: عملگرهای متعارف در مجموعه ها نظیر: اجتماع ، اشتراک ، تفاضل و ضرب دکارتی

ب: عملگرهای خاص نظیر: محدودیت ، تصویر یا پرتو ، ترکیب یا پیوند و تقسیم

### ۴-۲-۱- عملگر گزینش یا تحدید (Select)

این عملگر تاپلهایی را از یک رابطه گزینش می کند. عبارتی زیر مجموعه ای افقی از یک رابطه را بر می دارد. گزینش تاپل یا تاپلهایی از رابطه، معمولاً بر اساس شرط یا شرایطی صورت می پذیرد.

• نماد تحدید :  $\sigma_p(r)$

• P شرط گزینش است ، r رابطه (جدول)

•  $\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$

که p فرمولی محاسباتی است شامل and، Or، not یا هر یک از جملات بفرم :

$\langle \text{attribute} \rangle \text{op} \langle \text{attribute} \rangle$  or  $\langle \text{constant} \rangle$  که op یکی از =، #، <، >، ≤، ≥، است.

A	B	C	D
$\alpha$	$\alpha$	1	7
$\alpha$	$\beta$	5	17
$\beta$	$\beta$	12	10

Relation r:

A	B	C	D
$\alpha$	$\alpha$	1	7
$\beta$	$\beta$	23	10

$\sigma_{A=B \wedge D > 5}(r)$

### ۴-۲-۲- عملگر پرتو Project

این عملگر زیر مجموعه ای عمودی از یک رابطه را استخراج می کند صفات خاصه (ستونهای) پاسخ اعمال عملگر دارای ترتیبی هستند که در عملگر مشخص می شود.

• نماد عملگر  $\pi_{A_1, A_2, \dots, A_K}^{(r)}$  که  $A_2, A_1, \dots$  اسامی صفات خاصه و I نام رابطه است

• نتیجه بعنوان رابطه ای با K ستون تعریف می شود که با حذف ستونهایی که در لیست قرار ندارند بدست آمده است.

• سطرهای تکراری از نتیجه حذف می شوند.

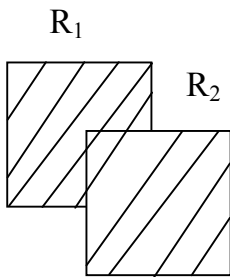
• مثال :

A	B	C
$\alpha$	10	1
$\alpha$	20	1
$\beta$	30	1
$\beta$	40	2

$$\Pi_{A,C}(r) = \begin{array}{|c|c|} \hline A & C \\ \hline \alpha & 1 \\ \alpha & 1 \\ \beta & 1 \\ \beta & 2 \\ \hline \end{array}$$

### ۴-۲-۳- عملگر اجتماع union

اجتماع دو رابطه رابطه ایست که تاپلهایش در یک یا هر دو رابطه وجود دارند.



$$R = R_1 \text{ union } R_2$$

• نماد عملگر:  $r \cup s$

•  $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

•  $r \cup s$  موقعی معتبر است که:

۱-  $r, s$  بایستی تعداد صفات خاصه برابر داشته باشند.

۲- حوزه (میدان) صفات خاصه دو مجموعه بایستی سازگار باشند. بعنوان مثال صفت خاصه  $i$  ام از هر دو رابطه یکسان باشند.

مثال:

Relation  $r, s$

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

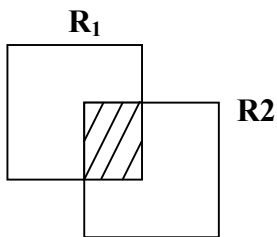
r

A	B
$\alpha$	2
$\beta$	3

s

$r \cup s$ :

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1
$\beta$	3



### ۴-۲-۴- اشتراک Intersect

اشتراک دو رابطه، رابطه ایست که تاپلهایش در هر دو رابطه وجود داشته باشند

$$R = R_1 \cap R_2$$

• نماد  $r \cap s$

•  $r \cap s = \{t \mid t \in r \text{ and } t \in s\}$

•  $r$  و  $s$  باید از نظر صفات خاصه برابر و صفات متناظر از یک میدان بر گرفته شوند

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

r1

A	B
$\alpha$	1
$\beta$	3

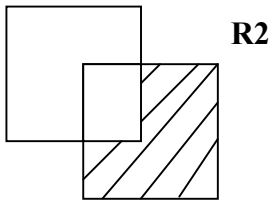
r2

A	B
$\alpha$	1

$r_1 \cap r_2$

۴-۲-۵ : عملگر تفاضل : set difference

تفاضل دو رابطه ، رابطه ایست که تاپلهایش در رابطه اول موجود باشند و در رابطه دوم وجود نداشته باشند . R1



نماد :  $r - s$

• تعریف عملگر :  $r - s = \{t \mid t \in r \text{ and } t \notin s\}$

• اختلاف دو مجموعه بین دو مجموعه سازگار انجام می شود . S و R باید دارای ستونهای یکسان باشند.

• مثال :

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

r

A	B
$\alpha$	2
$\beta$	3

s

A	B
$\alpha$	1
$\beta$	1

r - s

۴-۲-۶ : حاصلضرب کارتیزین : Cartesian Product

حاصل رابطه ایست حاوی ترکیب های ممکن تاپلهای دو رابطه که باید در هم ضرب شوند .

• نماد عملگر :  $r \times s$

• تعریف عملگر :  $r \times s = \{(t,q) \mid t \in r \text{ and } q \in s\}$

• فرض می شود صفات خاصه  $r(R)$  و  $s(S)$  مجزا باشند  $R \cap S = \emptyset$

• اگر صفات خاصه  $r(R)$  و  $s(S)$  مجزا نباشند تغییر نام بایستی صورت پذیرد .

• مثال :

A	B
$\alpha$	1
$\beta$	2

r

C	D	E
$\alpha$	10	a
$\beta$	10	a
$\gamma$	10	b

S

$r \times s$  :

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	10	a
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\gamma$	10	b

۴-۲-۷ : ترکیب (پیوند) join (Natural join)

حاصل رابطه ای است که تاپلهای آن از پیوند (ترکیب) تاپلهایی از دو رابطه بدست می آید بشرط تساوی مقادیر یک یا بیش از یک صفت خاصه .

• نماد عملگر :  $r \bowtie s$



- فرض  $r$  رابطه روی اسکیمای  $R$  و  $s$  رابطه ای روی اسکیمای  $S$  باشد. نتیجه رابطه ای در اسکیمای  $R \cup S$  است که با در نظر گرفتن هر جفت تاپل  $tr$  از  $r$  و  $ts$  از  $s$  بدست می آید.
- اگر  $tr$  و  $ts$  دارای مقادیر یکسان در هر یک از صفات خاصه  $R \cap S$  باشند یک تاپل  $t$  به نتیجه اضافه میشود بطوریکه :  $t$  همان مقادیر  $tr$  در  $r$  و  $ts$  در  $s$  را دارد.

مثال :  $R = (A,B,C,D)$

$S = (E,B,D)$

اسکیمای نتیجه  $== (A,B,C,D,E)$

- $r \infty s$  بصورت زیر تعریف میشود :

$$\prod_{r,A,r,B,r,C,r,D,s,E} (\bar{0}_{r.B=s.B \text{ and } r.D=s.D} (r \times S))$$

مثال :

A	B	C	D
$\alpha$	1	$\alpha$	a
$\beta$	2	$\gamma$	a
$\gamma$	4	$\beta$	b
$\alpha$	1	$\gamma$	a
$\delta$	2	$\beta$	b

$r$

B	D	E
1	a	$\alpha$
3	a	$\beta$
1	a	$\gamma$
2	b	$\delta$
3	b	$\epsilon$

$s$

A	B	C	D	E
$\alpha$	1	$\alpha$	a	$\alpha$
$\alpha$	1	$\alpha$	a	$\gamma$
$\alpha$	1	$\gamma$	a	$\alpha$
$\alpha$	1	$\gamma$	a	$\gamma$
$\delta$	2	$\beta$	b	$\delta$

$r \infty s$

### Division

### ۴-۲-۸ عملگر تقسیم

دو رابطه یکی از درجه  $m+n$  و دیگری از درجه  $n$  را برهم تقسیم می کند. حاصل رابطه ایست حاوی مقادیری از صفات خاصه رابطه از درجه  $m+n$  که مقادیر صفت خاصه دیگر به تمامی در رابطه درجه  $n$  وجود داشته باشند.

- نماد  $\div$
- این عملگر برای پرس و جوهایی که عبارت برای تمام (for all) را دارند مناسب است.
- فرض کنید  $r$  و  $s$  رابطه هایی روی اسکیمای  $R$  و  $S$  باشند بطوریکه :

$$R = (A1, \dots, Am, B1, \dots, Bn)$$

$$S = (B1, \dots, Bn)$$

نتیجه تقسیم  $r$  بر  $s$  رابطه ایست در اسکیمای  $R - S$

$$R - S = (A1, \dots, Am)$$

$$r \div s = \{t \mid t \in \prod_{R-S}(r) \wedge \forall u \in s(tu \in r)\}$$

• مثال :

A	B
$\alpha$	1
$\alpha$	2
$\alpha$	3
$\beta$	1
$\gamma$	1
$\delta$	1
$\delta$	3
$\delta$	4
$\beta$	2

B
1
2

s

A
$\alpha$
$\beta$

r ÷ s

A	B	C	D	E
$\alpha$	A	$\alpha$	a	1
$\alpha$	A	$\gamma$	a	1
$\alpha$	A	$\gamma$	b	1
$\beta$	A	$\gamma$	a	1
$\beta$	A	$\gamma$	b	3
$\gamma$	A	$\gamma$	a	1
$\gamma$	A	$\gamma$	b	1
$\gamma$	A	$\beta$	b	1

r

D	E
a	1
b	1

s

A	B	C
$\alpha$	a	$\gamma$
$\gamma$	a	$\gamma$

r ÷ s

### ۴-۳ - عملگرهای اضافه شده و عملیات دیگر جبر رابطه ای

#### ۴-۳-۱ : عملگر تغییر نام Rename

این امکان را می دهد که به یک رابطه با بیش از یک اسم رجوع شود .

$$P_X(E)$$

• عبارت E را تحت نام X بر می گردانند.

• اگر یک عبارت جبر رابطه ای E ، n ستون داشته باشد  $P_X(A_1, \dots, A_n)E$

به معنی آن است که نتیجه عبارت E تحت نام X با صفات خاصه تغییر نام یافته  $A_1, \dots, A_n$  بر می

گرداند.

#### ۴-۳-۲ : عملگر بسط Extend

عملگر است که برای گسترش عنوان یک رابطه بکار می رود . که معمولاً صفت خاصه اضافه شده مقادیر آن با ارزیابی یک عبارت محاسباتی مشخص بدست می آید .

#### Extend P Add (weight \* 454) As GMWT

GMWT وزن قطعات بر حسب گرم می باشد که در واقع تبدیل شده پوند به گرم است .

### Aggregate Operator

### ۴-۳-۳ : عملگر های جمعی

عملگرهایی که برای شمارش، مجموع، میانگین و می نیمم و ماکزیمم بکار می روند. در واقع مجموعه ای از مقادیر را گرفته و یک مقدار تکی را بعنوان خروجی باز می گردانند.

Min , Max , Avg , Sum , Count

مثال : مجموع قطعات تهیه شده توسط تهیه کننده s1 .  $\sigma_{s\#} = 's1' \rho \text{ sum}(\text{Qty}) (\text{sp})$

یا  $\rho \text{ sum}(\text{Qty}) (\text{sp})$

### Assignment

### ۴-۳-۴ - عملگر انتساب

- نماد ←
- این عملگر یک روش مناسب برای بیان پرس و جوهای پیچیده است .
- انتساب بایستی به یک متغیر رابطه ای موقت نسبت داده شود :

### ۴-۳-۵ - عملگر نیم پیوند Semi Join

این عملگر گونه ای دیگر از پیوند طبیعی است که در آن ، تنها تاپلهای پیوند شدنی از رابطه سمت چپ در رابطه جواب وارد میشوند. این عملگر در پایگاه داده های توزیع شده کاربرد دارد .

$$R1 \bowtie R2 = \prod_{\text{Attribute}(R1)} (R1 \bowtie R2)$$

### ۴-۳-۵ - عملگر نیم تفاضل Semi Minus

این عملگر بصورت زیر تعریف شده است :

$$R1 \text{ SEMIMINUS } R2 = R1 \text{ MINUS } (R1 \bowtie R2)$$

### ۴-۴ - مجموعه کامل عملگرها در جبر رابطه ای :

از عملگرهای مطرح شده ، برخی مبنایی هستند به این معنا که مجموعه آنها از نظر عملیاتی کامل است و هر عملگر دیگر را می توان بر حسب عملگرهای این مجموعه بیان کرد . این مجموعه کامل بصورت زیر است :

{Select , Project , Union , Minus, Time }

برخی عملگرهای تعریف شده با این مجموعه عبارتند از :

$$R1 \cap R2 = R1 - (R1 - R2) = R2 - (R2 - R1)$$

$$R1 \cap R2 = (R1 \cup R2) - (R1 - R2) \cup (R2 - R1)$$

$$R1(Y,X) \div R2(X) = R1[Y] - ((R1[Y] \times R2) - R1)[Y]$$

### ۴-۵ - برخی خواص عملگرها :

اگر R,S,T رابطه باشند داریم :

- 1-  $R \bowtie S = S \bowtie R$
- 2-  $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$
- 3-  $\prod_{A_1..A_p} (\prod_{A_i..A_j} (R)) = \prod_{A_1..A_p} (R)$
- 4-  $\sigma_{A_i='a'}(R) \cup \sigma_{A_i='a'}(S) = \sigma_{A_i='a'}(R \cup S)$
- 5-  $\sigma_{A_i='a'}(R) \cap \sigma_{A_i='a'}(S) = \sigma_{A_i='a'}(R \cap S)$
- 6-  $\sigma_{A_i='a'}(R) - \sigma_{A_i='a'}(S) = \sigma_{A_i='a'}(R - S)$

$$7- \prod A_1..A_p (R) \cup \prod A_1..A_p (S) = \prod A_1..A_p (R \cup S)$$

مثال : بانک اطلاعاتی تهیه کنندگان، قطعات، پروژه را در نظر بگیرید. ساختار آن در زیر ارائه شده است با استفاده از جبر رابطه ای به پرس و جوهای زیر پاسخ دهید .

S (s# , sname , status , city)  
 P (p# , pname , color , weight , city)  
 J (j# , jname , city)  
 SPJ (s# , p# , j# , Qty)

۱- جزئیات کامل تمام پروژه های شهر "تهران" را مشخص کنید .

$\sigma_{city = 'تهران'}(J)$  یا  $J \text{ where } city = 'تهران'$

۲- اسامی تهیه کنندگان قطعه P2 را بدهید :

$\prod_{Sname} (\sigma_{P\# = 'P2'}(S \Join SPJ))$  یا  $\prod_{Sname} (S \text{ join } SPJ \text{ where } P\# = 'P2')$

یا  $temp1 \leftarrow S \text{ join } SPJ$   
 $temp2 \leftarrow temp1 \text{ where } P\# = 'P2'$   
 $result \leftarrow \prod_{Sname} (temp2)$

۳- اسامی تهیه کنندگانی که اقلا یک قطعه آبی را تهیه می کنند :

$((P \text{ where } color = 'آبی') \Join SPJ) \Join S [sname]$

$\prod_{Sname} ((\sigma_{color = 'آبی'}(P) \Join SPJ) \Join S)$

۴- اسامی تهیه کنندگانی را بدهید که تمام قطعات را تهیه می کنند :

$\prod_{Sname} ((\prod_{(S\#, P\#)} (SPJ) \div \prod_{P\#} (P) \Join S)$

۵- اسامی تهیه کنندگانی که قطعه P2 را تهیه نمی کنند :

$(\prod_{S\#} (S) - (\prod_{S\#} (\sigma_{P\# = 'P2'}(SPJ)) \Join S [sname])$   
 یا  $(S[S\#] - (SPJ \text{ where } P\# = 'P2') [S\#]) \Join S [sname]$

۶- شماره قطعاتی را مشخص کنید که توسط یک تهیه کننده در شهر تهران یا پروژه ای در شهر تهران عرضه می شود :

$SPJ \Join (S \text{ Where } city = 'تهران') [P\#] \cup SPJ \Join (J \text{ Where } city = 'تهران') [P\#]$

$\prod_{P\#} (SPJ \Join (\sigma_{city = 'تهران'}(S))) \cup \prod_{P\#} (SPJ \Join (\sigma_{city = 'تهران'}(J)))$

مثال ۲ : یک بانک اطلاعاتی در محیط عملیاتی بانک را در نظر بگیرید که دارای جداول زیر می باشد :

شعبه بانک (نام شعبه، شهر شعبه، داراییها )

مشتری (نام مشتری، آدرس )

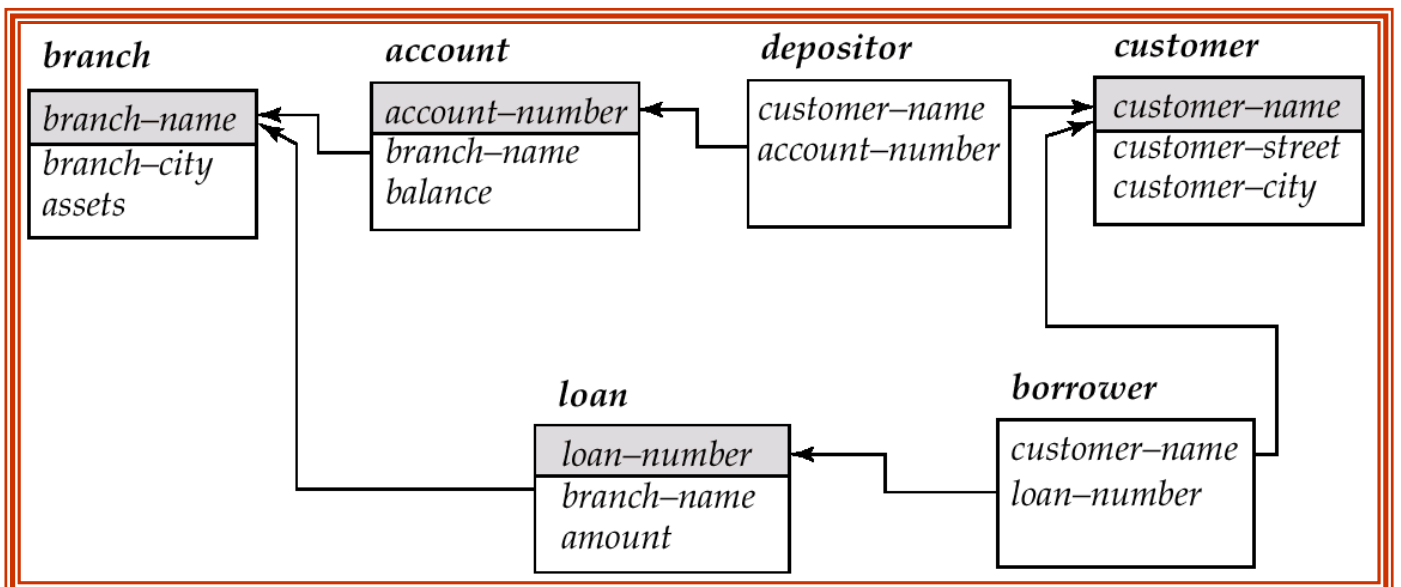
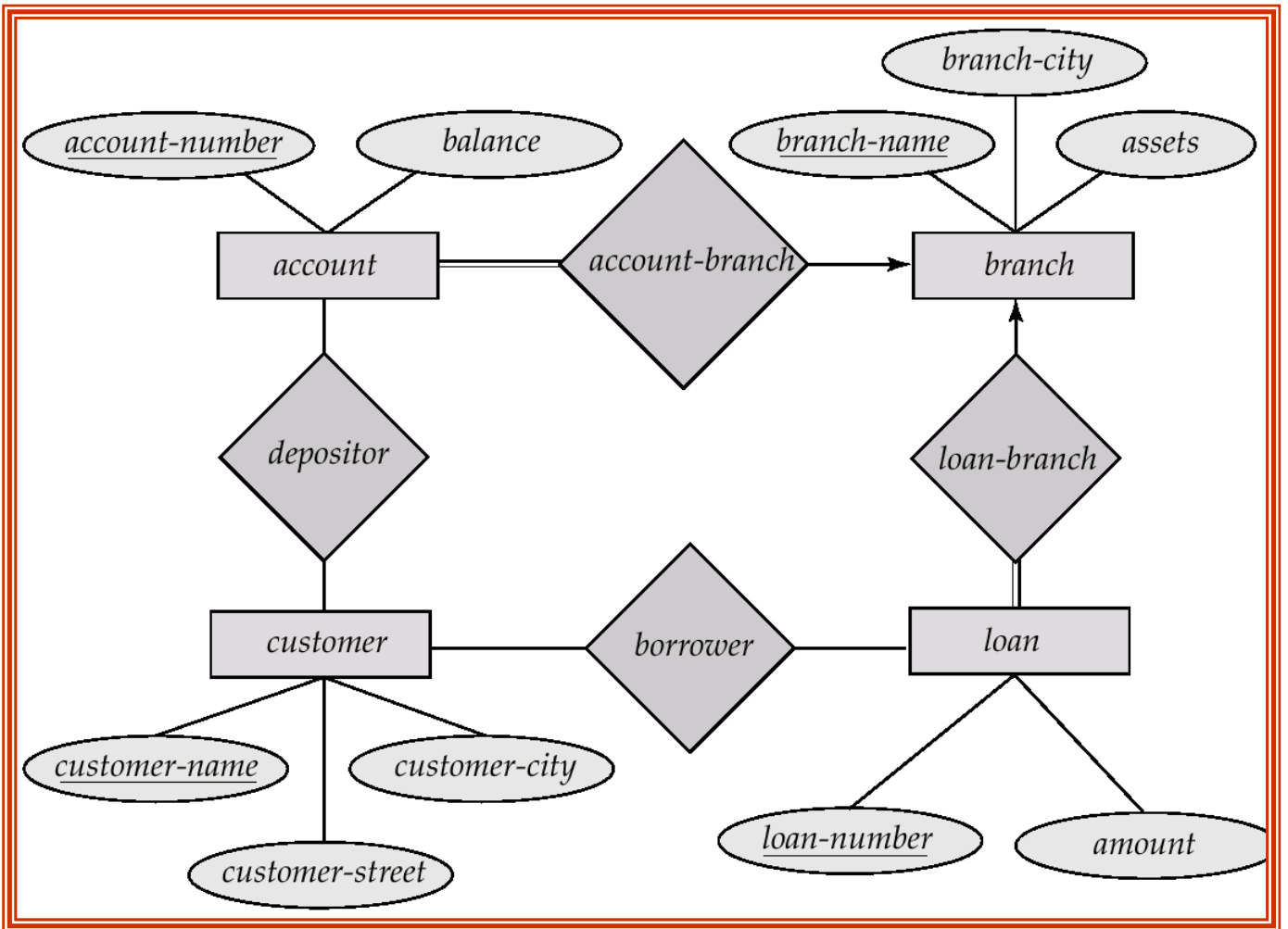
حساب (شماره حساب ، نام شعبه، موجودی )

وام (شماره وام، نام شعبه، مقدار وام )

سپرده گذاری (نام مشتری، شماره حساب)

وام گیرنده (نام مشتری، شماره وام)

branch (branch\_name , branch\_city , assets)  
 customer (customer\_name , customer\_street , customer\_city)  
 account (account\_number , branch\_name , balance)  
 loan (loan\_number , branch\_name , amount)  
 depositor (customer\_name , account\_number)  
 borrower (customer\_name , loan\_number)



هاquery :

۱- تمامی وامهای بیش از ۱۰۰۰۰۰۰ ریال را بدهید :

$\sigma_{\text{amount} > 100000}(\text{loan})$

۲- شماره وامهایی را که بیش از ۱۰۰۰۰۰۰ ریال می باشند بدهید :

$\Pi_{\text{loan\_number}}(\sigma_{\text{amount} > 100000}(\text{loan}))$

۳- اسامی تمامی مشتریانی که وام گرفته و شماره حساب دارند را بدهید :

$\Pi_{\text{customer\_name}}(\text{borrower}) \cap \Pi_{\text{customer\_name}}(\text{depositor})$

۴- اسامی تمام مشتریانی را که در شعبه مرکزی وام گرفته اند بدهید :

$\Pi_{\text{customer\_name}}(\sigma_{\text{branch\_name} = \text{'مرکزی'}}(\sigma_{\text{borrower.loan\_number} = \text{loan.loan\_number}}(\text{borrower} \times \text{loan})))$

۵- اسامی تمام مشتریانی که در شعبه نادری وام گرفته اند را بدهید :

Q1:  $\Pi_{\text{customer\_name}}(\sigma_{\text{branch\_name} = \text{'نادری'}}(\sigma_{\text{borrower.loan\_number} = \text{loan.loan\_number}}(\text{borrower} \times \text{loan})))$

Q2:  $\Pi_{\text{customer\_name}}(\sigma_{\text{loan.loan\_number} = \text{borrower.loan\_number}}(\sigma_{\text{branch\_name} = \text{'نادری'}}(\text{loan})) \times \text{borrower}))$

۶- اسامی تمام مشتریانی را که یک حساب در تمام شعبه های شهر قزوین دارند را بدهید :

$\Pi_{\text{customer\_name}, \text{branch\_name}}(\text{depositor} \bowtie \text{account}) \div \Pi_{\text{branch\_name}}(\sigma_{\text{branch\_city} = \text{'قزوین'}}(\text{branch}))$

### • نکات مهم جبر رابطه ای :

۱- جبر رابطه ای زبانی است روشمند یعنی برنامه ساز نه تنها به سیستم می گوید چه می خواهد بلکه نحوه بدست آوردن آنچه را که می خواهد نیز بیان می کند .

۲- معادل است با محاسبات رابطه ای یعنی هر توانی که مجموعه امکانات محاسباتی دارد جبر رابطه ای نیز دارد .

۳- جبر رابطه ای فقط برای بازیابی نیست بلکه می توان در عملیات ذخیره سازی و به هنگام سازی نیز استفاده کرد .

مثال : درج کنید  $\langle S7, S n7, \dots \rangle$

در جبر رابطه ای می توان از عملگر اجتماع برای درج استفاده نمود :  $r \cup E$  ←  $r$  , رابطه و

$E$  رابطه ثابت و دارای یک تاپلی است .  $S \cup \{s\# = s7, sname = sn7, \dots\}$

مثال : حذف کنید  $\langle S4, Sn4, \dots \rangle$

$S \text{ Minus } \{s\# = s4\}$

از عملگر تفریق برای حذف از یک رابطه استفاده می شود .  $r \leftarrow r - E$

$r$  رابطه و  $E$  رابطه ای با شرایط حذف می باشد .

## ۶-۵- محاسبات رابطه ای Relational calculus

محاسبات رابطه ای امکان دیگری است برای انجام عملیات روی رابطه ها و کار با بانک رابطه ای محاسبات رابطه ای حالت ناروشمند (غیر رویه ای) دارد یعنی در آن عملیات لازم برای اشتقاق رابطه ای از رابطه های دیگر تصریح نمی شود. به عبارتی تنها آنچه که باید بازیابی شود تشریح می شود و چگونگی بازیابی آنها بیان نمی شود.

محاسبات رابطه ای در دو شاخه بیان می شود:

۱- محاسبات روی تاپل ها Tuple Relational Calculus

۲- محاسبات روی میدان ها Domain Relational calculus

ایده اولیه استفاده از محاسبات رابطه ای نخستین بار توسط Knuth مطرح شد و سپس کاد نحوه استفاده از آن را برای انجام عملیات روی رابطه های بانک رابطه ای بیان کرد. بر اساس کارهای کاد زبان Alpha طراحی شد که البته به مرحله پیاده سازی نرسید. بعدها زبانی بنام Quel برگرفته از Alpha طراحی شد که به نوعی می توان آن را نمونه پیاده سازی Alpha دانست.

یکی از مهمترین جنبه های محاسبات رابطه ای، مفهوم متغیر طیفی یا متغیر تاپلی است. متغیر تاپلی متغیری است که مقادیرش از یک رابطه برگرفته می شود و به عبارت دیگر، طیف مقادیر مجازش، همان تاپلهای مجموعه بدنه رابطه است. متغیر تاپلی را بایستی تعریف کرد که در زبان Quel بصورت زیر است:

Range of SX is S ;  
RETRIEVE ( SX.S#) WHERE SX.CITY = `TEHRAN`

در اینجا متغیر SX همان متغیر تاپلی است که مقادیرش تاپلهای S هستند.

در محاسبات رابطه ای با متغیر میدانی نیز بجای متغیر تاپلی، متغیر میدانی وجود دارد و زبانهای متعددی مبتنی بر اینگونه از محاسبات رابطه ای طراحی شده اند که از جمله می توان به زبانهای زیر اشاره کرد.

ILL -

Formal Query Language FQL -

Query By Example QBE -

### ۶-۵-۱: محاسبات رابطه ای تاپلی

گرامر زبان مبتنی بر محاسبات رابطه ای را می توان در کتابهای مختلف بانک اطلاعاتی از جمله کتاب DATE مشاهده کرد که خارج از بحث این درس می باشد. در اینجا بطور خلاصه برخی مفاهیم اساسی موجود در محاسبات رابطه ای بیان می شود و سپس به ذکر چند مثال بسنده می کنیم.

#### ۶-۵-۱-۱: تعریف متغیر تاپلی:

متغیر تاپلی به کمک حکم زیر تعریف می شود:

Range of T is x1 ; x2 ; x3 ; ... ; xn

که در آن T متغیر تاپلی و X1 و X2 و ... Xn عبارات محاسبات رابطه ای هستند و نمایانگر رابطه هایی مثل R1 , R2 و Rn ... می باشند . بدیهی است اگر در لیست رابطه ها تنها یک رابطه داشته باشیم در اینصورت متغیر تاپلی مقادیرش را از همان رابطه R می گیرد .

RANGE OF SX IS S

مثال :

یا RANGEVAR SX RANGES OVER S ;

RANGE OF SPX IS SP ;

یا RANGEVAR SPX RANGES OVER SP ;

### ۵-۶-۱-۲ : عملگر ها

دو عملگر در محاسبات رابطه ای تعریف شده اند که به آنها سور گویند . که عبارتند از :

- سور وجودی Exist Quantifire به معنای وجود دارد :

- سور همگانی FOR.ALL Quantifire به معنای برای همه :

به کمک این دو سور عبارات محاسبات رابطه ای نوشته می شوند و در آن گزاره هایی بر نهاده میشود . حاصل ارزیابی این عبارات ممکن است true و یا false باشد .

یادآوری : فرض  $N \in X$

EXISTS X (X>10) —————> True

EXISTS X (X<-5) —————> False

N1	N2	N3
1	2	3
1	2	4
1	3	4

1) EXISTS T (T(N1)=1) : True

2) FORALL T (T(N1)=1) : True

3) EXISTS T (T(N1)=1 AND T(N3)>5) : False

T متغیر تاپلی روی جدول NumB

◀ توجه : سور همگانی FORALL را می توان به کمک سور وجودی EXISTS بیان نمود .

FORALL X(f) = NOT EXISTS X (NOT f)

### ۵-۶-۲ : استفاده در حساب محمولات در فرموله کردن پرس و جو ها :

سوال ۱ : شماره تهیه کنندگان ساکن C2 با وضعیت بیشتر از ۲۰ را بیابید .

Sx.s# WHERE sx.city = 'c2' AND sx.STATUS > 20

سوال ۲ : شماره جفت تهیه کنندگان ساکن یک شهر را بدهید .

Sx.s# , Sy.s# WHERE sx.city = sy.city AND sx.s# < sy.s#

سوال ۳ : اسامی تهیه کنندگان قطعه p2 را بدهید .

Sx.SNAME WHERE EXISTS Spx (Spx.s# = Sx.s# AND Spx.p# = 'p2')



سوال ۴: نام تهیه کنندگانی را بدهید که دست کم یک قطعه قرمز را تهیه می کنند .

SX.SNAME  
WHERE EXISTS SPX (SX.S# = SPX.S# AND EXISTS PX (PX.P# = SPX.P# AND  
PX.COLOR = 'RED'))

سوال ۵: نام تهیه کنندگانی را مشخص کنید که حداقل یک قطعه از قطعات عرضه شده توسط تهیه کننده S2 را تامین

می کنند .  
SX.SNAME  
WHERE EXISTS SpX ( EXISTS SPY ( SX.S# = SPX.S# AND SPX.P# = SPY.P# AND SPY.S#  
= 'S2' ))

سوال ۶: اسامی تهیه کنندگانی را بدهید که تمام قطعات را تهیه می کنند .

SX.SNAME WHERE FORALL PX (EXISTS SPX (SPX.S# = SX.S# AND SPX.P#  
= PX.P# ))  
FORALL جواب پرس و جو بدون  
Sx.SNAME WHERE NOT EXISTS PX (NOT EXISTS SPX (SPX.S# = SX.S# AND SPX  
.P# = PX.P# ))

سوال ۷: شماره و وزن قطعاتی را تهیه کنید که وزن هر قطعه بر حسب گرم بیشتر از ۱۰۰۰ باشد .

(px.p# , px.WEIGHT \* 454 AS GWMT)  
WHERE (px.WEIGHT \* 454 > 1000)

سوال ۸: شهر قطعاتی را مشخص کنید که بیش از سه قطعه آبی در آن انبار شده باشد .

Px.city  
WHERE COUNT (PY WHERE PY.city = px.city AND py.color = 'BLUE' ) > 3

### • نکات مهم در خصوص آنالیز رابطه ای :

- ۱- محاسبات رابطه ای ناروشمند است و تفاوت اصلی اش با جبر رابطه ای همین است .
- ۲- جبر رابطه ای و محاسبات رابطه ای معادلند .
- ۳- هر دو اکمال رابطه ای دارند یعنی هر رابطه متصور از رابطه های ممکن قابل اشتقاق به کمک هر یک از این دو امکان می باشد .
- ۴- زبانهای مبتنی بر محاسبات رابطه ای به دلیل ناروشمند بودن سطحشان بالاتر است .
- ۵- هر پرس و جو مبتنی بر محاسبات رابطه ای را می توان بفرم  $\{t/p(t)\}$  نیز نمایش داد .  
بطوریکه :
- مجموعه ای از تمام تاپلهای  $t$  است که شرط  $p$  برای آن درست است .
- $t$  یک متغیر تاپلی است و  $t[A]$  نشان دهنده مقدار تاپل  $t$  روی صفت خاصه  $A$  می باشد .
- $t \in r$  به این معناست که تاپل  $t$  در رابطه  $r$  است .

- p یک فرمول است که شرط محاسبات می باشد .

مثال : مشخصات تهیه کنندگان با وضعیت بیش از ۱۰ را بدید .

$$S \wedge t[\text{status}] > 10 \in \{t/t\}$$

تمرینات این فصل :

۱) دو رابطه A و B را در نظر می گیریم. پیوند این دو رابطه به صورت A JOIN B نوشته می شود. اگر مجموعه عنوان (Heading) این دو رابطه، عنصر مشترک نداشته باشد، عبارت A JOIN B معادل با کدام عبارت جبر رابطه ای است؟ اگر مجموعه عنوان دو رابطه یکسان باشند، چطور؟

۲) Codd واضح تنوری مدل رابطه ای، هشت اپراتور در جبر رابطه ای تعریف کرده است که عبارتند از: اجتماع، تفاضل، ضرب، گزینش، پرتو، اشتراک، پیوند و تقسیم. اولاً عملکرد این اپراتورها را مطالعه کرده و به کمک شکل‌های ساده نشان دهید.

ثانیاً: پنج اپراتور اول را اپراتورهای ابتدایی یا مبنایی می نامند. اپراتورهای پیوند، اشتراک و تقسیم را بر حسب اپراتورهای مبنایی تعریف کنید. ( اپراتور مبنایی یعنی اپراتوری که به کمک اپراتور (های) دیگر قابل تعریف نباشد.)

\* با فرض پایگاه تهیه کنندگان و قطعات به صورت زیر، به دو سوال بعد پاسخ دهید:

S(S#, SNAME, STATUS, CITY)

P(P#, PNAME, COLOR, WEIGHT, CITY)

SP(S#, P#, QTY)

SPJ(S#, P#, J#, QTY)

J(J#, JNAME, JCITY)

۳) مقدار عبارت جبری زیر چیست؟

S JOIN SP JOIN P

۴) به کمک اپراتورهای جبر رابطه ای، به پرس و جوهای زیر روی پایگاه S و P و J و SPJ پاسخ دهید:

۱. محموله هایی را بیابید که تعداد آنها بین ۳۰۰ و ۷۵۰ باشد.
۲. لیست سه تاییهای تشکیل شده از شماره تهیه کننده، شماره قطعه و شماره پروژه را بیابید، به نحوی که تهیه کننده، قطعه و پروژه از یک شهر باشند.
۳. شماره قطعاتی را بیابید که توسط تهیه کننده ساکن C2 برای پروژه ای از شهر C2 تهیه شده باشند.
۴. اسامی پروژه هایی را که S1 برای آنها قطعه تهیه کرده است، بیابید.
۵. جدولی ایجاد کنید حاوی شماره قطعاتی که یا توسط تهیه کننده ای ساکن C2 تهیه شده باشد و یا برای پروژه ای از شهر C2.

۵) به کمک اپراتورهای محاسبات رابطه ای، به پرس و جوهای شماره ۱ تا ۴ از سوال ۵ پاسخ دهید.

۶) به کمک اپراتورهای محاسبات رابطه ای، به پرس و جوهای زیر پاسخ دهید:

۱. شماره تهیه کنندگانی را بیابید که وضعیت آنها از وضعیت S1 کوچک تر باشد.

۷) فرض می کنیم که R1 و R2 سازگار از نظر نوع (Type Compatible) باشند. در هر یک از موارد زیر، کلید اصلی را مشخص کنید:

R1 UNION R2	(ث)	الف) یک گزینش دلخواه از R1
R1 MINUS R2	(ج)	ب) یک پرتو دلخواه از R1
R1 JOIN R2	(چ)	پ) R1 TIME R2
R1 DIVIDE R2	(ح)	ت) R1 INTERSECT R2

۸) اپراتور وراپیوند (پیوند بیرونی یا Outer Join) چیست؟

۹) عملکرد اپراتور ورا اجتماع (Outer Union) چیست؟

۱۰) با استفاده از پایگاه رابطه ای طرح شده در کلاس (شامل STT، COT و STCOT) یک مثال از اپراتور SEMIJOIN و یک مثال از اپراتور SEMIMINUS قید کنید.

\* تمرینهای زیر برای مطالعه بیشتر پیشنهاد می شوند:

۱) عبارت  $R1 \text{ SEMIJOIN } R2$  را به سه عبارت معادل بیان کنید.

۲) رابطه R از درجه n چند پرتو دارد؟

## فصل پنجم :

# آشنایی با زبان SQL

### ۱-۵- مقدمه :

SQL یک زبان استاندارد برای کار روی بانک اطلاعاتی رابطه ای است و هر محصول نرم افزاری موجود در بازار از آن پشتیبانی می کند . SQL اولین بار در اوایل دهه ۱۹۶۰ در بخش تحقیقات IBM طراحی شد و برای اولین بار در مقیاس بزرگ روی کامپیوترهای IBM به نام سیستم R پیاده سازی شد و سپس بر روی انواع متعددی از محصولات تجاری در IBM و محصولات دیگر پیاده سازی شد . در این فصل زبان SQL مورد بحث SQL 92 یا SQL 2 است که 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 تقریباً توسط هر محصول نرم افزاری بانک اطلاعات پشتیبانی می شود و هر فرد حرفه ای نیازمند آن است .

### ۲-۵- 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 کارا کتری با طول ثابت که توسط کاربر n مشخص می شود .
  - Varchar (n) رشته های با طول متغیر حداکثر به طول n
  - در Oracle نوع Varchar2 برای رشته های با طول متغیر و حداکثر ۲۰۰۰ کارا تر تعریف شده است .
  - Int اعداد صحیح
  - Small int اعداد کوچک
  - Numeric (p,d) اعداد اعشاری p حداکثر تعداد رقمها و d تعداد 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 ۴ رقم برای سال ، ماه و روز :
- مثال : 2001 - 7 - 27
- Time برای نمایش ساعت .

time '09 : 00 : 30 '  
time '09 : 00 : 30 : 75 '

• Timestamp شامل تاریخ و زمان

Timestamp : ' 2001 - 1 - 27 09 : 30 : 27 . 75'

• Interval دوره های زمانی

مثال : interval '1' day

مقادیر Interval می توانند به مقادیر date / time / timestamp اضافه شوند .

• امکان استخراج فیلد از داده های تاریخ و زمان وجود دارد .

extract (year from r . time 1)

• امکان تبدیل رشته به تاریخ و زمان نیز وجود دارد .

cast < string - valued expression > as date

• در SQL 92 امکان تعریف دامنه نیز وجود دارد که برای اینکار دستور زیر بایستی نوشته شود :

Create domain نام جدید نوع [ not null ]

create domain Dollars numeric(12, 2)

★ create domain AccountType char(10)

constraint account-type-test

check (value in ('Checking', 'Saving'))

## ۵-۲-۱ : دستورات تعریف جداول

Create table r (  $A_1D_1, A_2D_2, \dots, A_nD_n, \dots$ , ( integrity - constraint  $k$  ) )

\* نام رابطه است .

\* هر  $A_i$  نام صفت خاصه در شمای r است .

\*  $D_i$  نوع داده ای در دامنه صفت خاصه  $A_i$  است . مثال :

Create table branch

( branch - name char (15) not null ,

branch - city char (30) ,

assest integer )

## ❖ محدودیت های جامعیت در تعریف جداول :

- not null
- Primary key (  $A_1, \dots, A_n$  )

کلید اصلی انتخاب not null را بطور خودکار برای فیلد در نظر خواهد گرفت .

• Check ( P ) که P شرط است برای کنترل مقدار فیلدها .

- FOREGIN Key ( A<sub>1</sub>... ) References

مثال:

Create table	branch
(branch – neme	char (15) ,
Branch – city	char (30) ,
Assests	Integer ,
Primary key	( branch – name ) ,
Check ( assests >= 0 )	
<hr/>	
Create table	S
( SNO	char (5) ,
sname	char (20) ,
status	Numeric (5) ,
city	char (15) ,
Primary key	( SNO ) ;
<hr/>	
Create table	P
( PNO	char (6) ,
color	char (20) ,
weight	Numeric (5,1) ,
city	char (15) ,
Primary key	( PNO ) ,
<hr/>	
Create table	SP
( SNO	char (5) ,
PNO	char (6) ,
Qty	Numeric (9) ,
Primary key ( SNO , PNO ) ,	
Foreign key ( PNO ) References	P ,
Foreign key ( SNO ) References	S ;

### ۶-۲-۲: حذف و تغییر جداول Drop and Alter tables

عبارت Drop table تمام اطلاعات یک جدول و خود جدول را از بانک اطلاعاتی حذف می کند و عبارت alter table برای اضافه کردن و یا حذف صفات خاصه به جدول موجود بکار می رود. با اضافه کردن صفت خاصه جدید تمامی تاپلهای رابطه مقدار null را برای آن صفت خاصه می گیرند. فرم کلی دستور بصورت زیر است:

**alter table r add A D**

A اسم صفت فاصله جدید و D دامنه آن می باشد.

**Alter table r drop A** صفت خاصه را از جدول حذف می کند.

- این دستور ( در عمل حذف صفت خاصه ) توسط بیشتر بانکهای اطلاعاتی پشتیبانی نمی شود.

**Alter table r Modify A NewType**

- این دستور برای تغییر نوع یک صفت خاصه بکار می رود.

### ۶-۳: احکام کار با داده ها در SQL

در SQL چهار دستور اساسی برای کار با داده ها وجود دارد:

- SELECT
- UPDATE

- DELETE
- INSERT

### ۶-۳-۱ : احکام بازیابی داده ها

در SQL تنها یک حکم واحد برای بازیابی وجود دارد و آن دستور SELECT است . شکل کلی این دستور به فرم زیر می باشد :

```
SELECT      A1, A2, ... An
FROM        r1, r2, ... rm
[where     P ]
[ Group by columns ]
[ Having   P ]
[ ORDER   BY  A1 ... Ak
```

A<sub>i</sub> ها صفات خاصه و R<sub>i</sub> ها رابطه ها و P شرط است .

در صورت استفاده از سه قسمت اول پرس و جو معادل جبر رابطه ای زیر می باشد :

$$\pi_{A_1, A_2, \dots, A_n} (\sigma_P^{(r_1 \times r_2 \times \dots \times r_n)})$$

عبارت جلوی دستور select معادل عمل پرتو در جبر رابطه ای است و عبارت جلوی where معادل همان تحدید یا گزینش است . برخی نکات مهم این دستور عبارتند از :

- عبارت ( \* ) در جلوی select معادل تمام صفات خاصه رابطه است .
- در SQL ممکن است نتیجه پرس و جو حاوی داده های تکراری باشد لذا برای حذف مقادیر تکراری کلمه distinct را بایستی بعد از select بکار برد .

```
Select      distinct  A1, ...
From        ri
```

- در صورت استفاده از کلمه all بعد از select حذف مقادیر تکراری انجام نخواهد گرفت .
  - در جلوی select می توان از عبارات ریاضی شامل عملگرهای + و - و × و / نیز استفاده کرد .
- ```
Select      S#, P#, Qty * 5      from  SP
```
- در جلوی عبارت where شرط بکار می رود که می تواند با and ، or و not نیز ترکیب شود .
  - در شرط جلوی where از عبارت between نیز می توان استفاده کرد .
- ```
Select      *      From      SP
              Where   Qty  between  2      and  5
```
- اگر چند جدول جلوی عبارت from آورده شود به منزله حاصلضرب دکارتی رابطه هاست .
  - در SQL امکان دوباره نامیدن رابطه و یا صفات خاصه با استفاده از عبارت as وجود دارد . قالب کلی دستور بصورت زیر است :

```
Oldname      as      new name
یا Oldname   new name
```

- متغیرهای تاپلی در عبارت بعد از From از طریق کلمه as تعریف می شوند .

```
Select *
      From S as TS , SP as Tsp
      Where TS.S# = Tsp.S#
```

- عبارت **ORDER BY** به معنی آن است که کاربر می خواهد جواب را بطور منظم روی صفت خاصه مورد نظرش ببیند . در صورتی که در انتها عبارت d esc بیاید ترتیب نزولی و اگر asc بیاید ترتیب صعودی است .

مثال : order by S # desc

- در صورت استفاده از چند جدول در جلوی عبارت from و استفاده از شرط خاص می توان عمل **پیوند** را نیز انجام داد .

مثال : مشخصات تهیه کنندگان و قطعات در یک شهر را بدهید .

```
Select S . * , P . * from S , P
```

```
Where S . city = P . city
```

- می توان عمل پیوند را با شرطهای اضافی نیز انجام داد .

```
Select S . * , P . * from S , P
      Where S . city = P . city AND S . Status > 10
```

- **نکته** : می توان یک جدول را با خودش نیز ترکیب ( پیوند ) کرد .

مثال : شماره جفت تهیه کنندگان ساکن یک شهر را بدهید .

```
Select First . S # , Second . S #
```

```
From S First , S Second
      Where First . city = Second . city
            AND First . S # < Second . S #
```

- در زبان SQL امکان مقایسه رشته ها و تطابق آنها نیز وجود دارد . این امکان توسط دو عملگر % و - میسر است : علامت درصد % برای تطابق هر زیر رشته بکار می رود و علامت underline = برای تطابق یک کاراکتر بکار می رود .

```
Select P# , ...
```

```
From P
      Where Pname like " w %"
```

% w یعنی آنهایی که با w آغاز شده اند . و " % W " یعنی آنهایی که به w ختم شده اند .

```
Select *
      From S
            Where city like ' % C --- '
      شهر حداقل چهار حرف داشته باشد و چهارمین حرف از آخر با C آغاز شود .
```

```
SELECT SNAME FROM S WHERE SNAME LIKE ' C _ _ '
```

- اسامی تهیه کنندگانی که دقیقاً سه حرف داشته باشند .

- در صورتی که عدم تطابق بخواهد چک شود می توان به جای LIKE از NOT LIKE استفاده نمود .

- الحاق رشته ها در SQL توسط عملگر || انجام می گیرد .



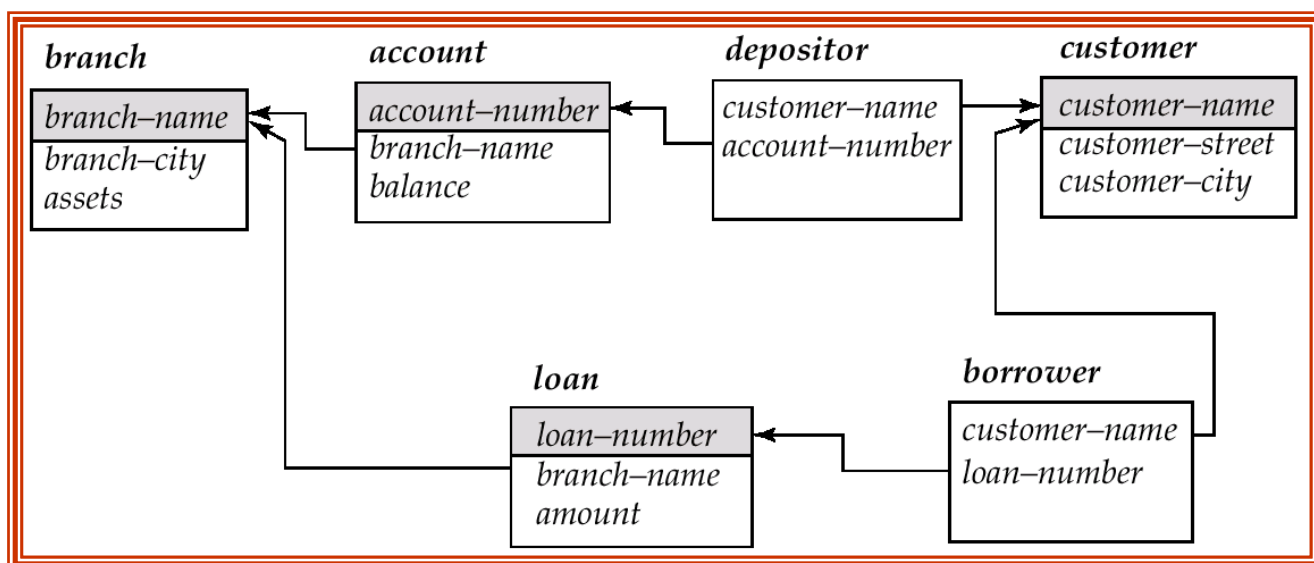
- عملیات اجتماع و اشتراک و تقیض نیز در SQL وجود دارند. عبارت UNION برای اجتماع و عبارت INTERSECT برای اشتراک و عبارت EXCEPT برای عملگر منها بکار می روند.  
مثال : شماره قطعاتی را مشخص کنید که یا وزن آنها بیش از ۱۶ پوند باشد و یا تهیه کننده S2 آنها را تهیه کرده باشد .

```
SELECT P.P#
FROM P WHERE WEIGHT > 16.0
UNION
SELECT SP.P#
FROM SP
WHERE SP.S# = 'S2'
```

سطرهای تکراری اضافی همیشه از نتیجه یک UNION ، INTERSECT و یا EXCEPT حذف می شوند. اما در SQL نسخه های UNION ALL ، EXCEPT ALL فراهم شده که در آن سطرهای تکراری باقی می ماند.

مثال های اضافی :

شمای به کار رفته در این مثال :



- Find the names of all branches in the *loan* relations, and remove duplicates  
**select distinct branch-name  
from loan**
- To find all loan number for loans made at the Perryridge branch with loan amounts greater than \$1200.  
**select loan-number  
from loan  
where branch-name = 'Perryridge' and amount > 1200**
- Find the name, loan number and loan amount of all customers having a loan at the Perryridge branch.

```
select customer-name, borrower.loan-number, amount
    from borrower, loan
    where borrower.loan-number = loan.loan-number and
           branch-name = 'Perryridge'
```

■ Find the name, loan number and loan amount of all customers; rename the column name *loan-number* as *loan-id*.

```
select customer-name, borrower.loan-number as loan-id, amount
    from borrower, loan
    where borrower.loan-number = loan.loan-number
```

■ Find the customer names and their loan numbers for all customers having a loan at some branch.

```
select customer-name, T.loan-number, S.amount
    from borrower as T, loan as S
    where T.loan-number = S.loan-number
```

■ Find the names of all branches that have greater assets than some branch located in Brooklyn.

```
select distinct T.branch-name
    from branch as T, branch as S
    where T.assets > S.assets and S.branch-city = 'Brooklyn'
```

■ Find the names of all customers whose street includes the substring "Main".

```
select customer-name
    from customer
    where customer-street like '%Main%'
```

■ Match the name "Main%"

```
like 'Main\%' escape '\'
```

■ List in alphabetic order the names of all customers having a loan in Perryridge branch

```
select distinct customer-name
    from borrower, loan
    where borrower.loan-number = loan.loan-number and
           branch-name = 'Perryridge'
    order by customer-name
```

■ Find all customers who have a loan, an account, or both:

```
(select customer-name from depositor)
union
(select customer-name from borrower)
```

■ Find all customers who have both a loan and an account.

```
(select customer-name from depositor)
intersect
(select customer-name from borrower)
```

■ Find all customers who have an account but no loan.

```
(select customer-name from depositor)
except
(select customer-name from borrower)
```

• توابع جمعی SQL

- در زبان SQL توابع جمعی نیز وجود دارند . این توابع روی لیستی از مقادیر یک ستون یا رابطه عمل کرده و یک مقدار را برمی گردانند . توابع جمعی عبارتند از :

- A VG برای محاسبه میانگین

- M AX , M IN مینیمم ، ماکزیمم

- S U M مجموع مقادیر

- C O U N T تعداد مقادیر

مثال :

```
select max(status)
from S
```

تعداد تهیه کنندگان را می دهد .

```
SELECT COUNT(*)
FROM S
SELECT COUNT (DISTINCT P#)
FROM SP
```

□ عبارت group by در دستور باعث می شود رابطه داده شده بعد از جمله from را برحسب مقادیر ستون داده شده گروه بندی کرده و آنگاه حکم SELECT در این جدول بازآرایی شده اجرا می شود .

مثال : Q<sub>1</sub> : شماره قطعات و کل تعداد تهیه شده از هر قطعه را بدهید .

```
SELECT P# , SUM(QTY)
FORM SP
GROUP BY P# ;
```

S#	P#	Qty
S <sub>1</sub>	P1	103
S <sub>2</sub>	P <sub>1</sub>	80
S <sub>3</sub>	P <sub>1</sub>	120
S <sub>1</sub>	P <sub>2</sub>	200
S <sub>2</sub>	P <sub>2</sub>	150
S <sub>1</sub>	P <sub>3</sub>	
⋮	⋮	

P#	Sum(QTY)
P1	300
P2	350
⋮	

**نکته :** صفات خاصه در دستور select خارج از توابع جمعی بایستی در لیست گروه Group by ظاهر شده باشند در غیر این صورت خطا رخ می دهد .

• مثال : شماره قطعاتی را بدهید که توسط بیش از یک تهیه کننده تهیه می شود .

```
Select      P#
           From    sp
           Group by p#
           Having  count(*) >1
```

**نکته :** Having معنای مستقل ندارد و همیشه با Group by می آید و نقش آن در گروه همانند نقش Where در سطر می باشد .

■ Find the average account balance at the Perryridge branch.

```
select avg (balance)
       from account
       where branch-name = 'Perryridge'
```

■ Find the number of tuples in the *customer* relation.

```
select count (*)
       from customer
```

■ Find the number of depositors in the bank.

```
select count (distinct customer-name)
       from depositor
```

■ Find the number of depositors for each branch.

```
select branch-name, count (distinct customer-name)
       from depositor, account
       where depositor.account-number = account.account-number
       group by branch-name
```

■ Find the names of all branches where the average account balance is more than \$1,200.

```
select branch-name, avg (balance)
       from account
       group by branch-name
       having avg (balance) > 1200
```

- SQL در برخورد با مقدار NULL به عنوان یک عملوند در عمل مقایسه نمی تواند تصمیمی بگیرد. یعنی سطرهای دارای NULL در ستون مورد نظر را در کار دخالت نمی دهد. اما اگر از عبارت IS NULL استفاده شود آنگاه سیستم با مقدار NULL برخورد می کند.

```
Select S#
      From sp
      Where (status > 15) OR (status is NULL)
```

- نتیجه هر عمل ریاضی روی NULL نیز NULL است.
- تمام توابع جمعی بجز Count(\*) تا بله های با مقادیر NULL در آن صفت خاصه را نادیده می گیرند.

5 + NULL → NULL

★E.g. Find all loan number which appear in the *loan* relation with null values for *amount*.  
**select loan-number**  
**from loan**  
**where amount is null**

- در برخی نسخه های SQL، توابع دیگری نیز وجود دارند که از جمله می توان توابع زیر را نام برد:
- SYSDATE, NEXT-DAY, LN, EXP, TAN, TANH, SIN, SINH, COS, COSH, CEIL, FLOOR, STDEV, VARIANCE, POWER, MOD, SIGN, CHR, CONCAT, UPPER, LOWER, LPAD, RPAD, LTRIM, RTRIM, REPLACE, TRANSLATE, LENGTH, TO\_CHAR, USER, TO\_NUMBER

### □ پرس و جوهای فرعی

پرس و جوهای فرعی یکی از تواناییهای مهم در SQL می باشد. یک پرس و جو فرعی یک عبارت پرس و جو Select-from-where است که در داخل یک پرس و جو بکار برده می شود.  
 مثال: اسامی تهیه کنندگان قطعه P2 را بدهید.

یک راه برای فرموله کردن این پرس و جو استفاده از مکانیسم پیوند است:

```
Select Sname
      From S,Sp
      Where S.S# = SP.S#
            AND SP.P# = 'P2'
```

راه دیگر استفاده از پرس و جو فرعی است:

```
Select Sname
      From S
      Where S# IN (Select S#
                  From SP
                  Where P# = 'P2')
```

Sub Query

- سیستم ابتدا پرس و جوی فرعی را اجرا می کند که حاصل یک مجموعه از S # ها است .  
مثال : اسامی تهیه کنندگانی را بدهید که اقلا یک قطعه آبی رنگ تهیه می کنند .  
در این پرس و جو سه جدول S ، P ، SP دخالت دارند :

```
Select Sname
From S
Where S# IN (Select S# from SP
              Where P# IN (Select P#
                            From P
                            Where Color = 'آبی'))
```

- اگر نتیجه پرس و جوی فرعی بیش از یک مقدار باشد از عبارت IN استفاده می شود و اگر قطعا تک مقداری باشد می توان از عملگر = استفاده نمود .  
مثال : شماره تهیه کنندگان هم شهر با S1 را بدهید .

```
Select S#
From S
Where city = (Select city from S
              Where S# = 'S1')
```

جواب پرس و جوی فرعی بالا یک مقدار است .

- می توان در پرس و جوهای فرعی از **تابع جمعی** نیز استفاده کرد .  
مثال : شماره تهیه کنندگانی را بدهید که مقدار وضعیت آنها از ماکزیمم مقدار وضعیت موجود در S کمتر باشد

```
SELECT S#
FROM S
WHERE STATUS < (SELECT MAX (STATUS)
                 FROM S);
```

- در SQL می توان از **سور وجودی** نیز استفاده کرد .  
مثال : اسامی تهیه کنندگان قطعه 2 P را بدهید .

```
SELECT SNAME
FROM S
WHERE EXISTS (SELECT * FROM SP
              WHERE SP.S# = S.S# AND SP.P#='P 2')
```

- عبارت EXISTS در SQL دارای ارزش درست است اگر فقط اگر نتیجه ارزیابی (Select ...) تهی نباشد .

### • نکات مهم در مورد مثال مطرح شده :

۱. وجود S در S.S# ضروری ولی SP در SP.S# برای وضوح بیشتر است . وقتی که در یک پرس و جوی درونی از جدولی از پرس و جوی بیرونی ارجاع می دهیم . قید کردن شناسه آن ستون الزامی است .
۲. وجود شرط SP.S# = S.S# به این معنا است که سیستم عمل پیوند را انجام دهد . هر چند ظاهرش همان است و از این دیدگاه استفاده از exists همیشه کارا تر از شبیه سازی عمل پیوند است .

- Find all customers who have both an account and a loan at the bank.

```
select distinct customer-name
from borrower
where customer-name in (select customer-name
                        from depositor)
```

- Find all customers who have a loan at the bank but do not have an account at the bank

```
select distinct customer-name
from borrower
where customer-name not in (select customer-name
                            from depositor)
```

- Find all customers who have both an account and a loan at the Perryridge branch

```
select distinct customer-name
from borrower, loan
where borrower.loan-number = loan.loan-number and
       branch-name = "Perryridge" and
       (branch-name, customer-name) in
       (select branch-name, customer-name
        from depositor, account
        where depositor.account-number =
            account.account-number)
```

- Find all branches that have greater assets than some branch located in Brooklyn.

```
select distinct T.branch-name
from branch as T, branch as S
where T.assets > S.assets and
       S.branch-city = 'Brooklyn'
```

- Same query using > **some** clause

```
select branch-name
from branch
where assets > some
        (select assets
         from branch
         where branch-city = 'Brooklyn')
```

(= **some**) = **in**

However, (<> **some**) = **not in**

(<> **all**) = **not in**

However, (= **all**) = **in**

■ Find the names of all branches that have greater assets than all branches located in Brooklyn.

```
select branch-name
  from branch
  where assets > all
    (select assets
     from branch
     where branch-city = 'Brooklyn')
```

■ The **unique** construct tests whether a subquery has any duplicate tuples in its result.

■ Find all customers who have at most one account at the Perryridge branch.

```
select T.customer-name
  from depositor as T
  where unique (
    select R.customer-name
    from account, depositor as R
    where T.customer-name = R.customer-name and
          R.account-number = account.account-number and
          account.branch-name = 'Perryridge')
```

■ Find all customers who have at least two accounts at the Perryridge branch.

```
select distinct T.customer-name
  from depositor T
  where not unique (
    select R.customer-name
    from account, depositor as R
    where T.customer-name = R.customer-name
    and R.account-number = account.account-number
    and account.branch-name = 'Perryridge')
```

۶-۳-۲ : احکام تغییر بانک اطلاعاتی:

۶-۳-۱ : احکام حذف داده ها

برای حذف یک یا چند تاپل از جدول می توان از دستور delete استفاده کرد :

```
DELETE FROM R
      WHERE P
```



مثال : حذف کنید اطلاع <S3,P4,...> را .

```
DELETE FROM SP
WHERE S# = 'S3' AND P# = 'P4'
```

در دستور زیر بخاطر قاعده جامعیت بایستی دستور زیر نیز اجرا شود .

```
DELETE FROM P
WHERE P# = 'P5'
DELETE FROM SP
WHERE P# = 'P5'
```

□ دستور زیر برای حذف تمام سطرهای Sp بکار می رود:

```
DELETE FROM SP
```

۶-۳-۲-۲ : اضافه کردن تاپل و یا تاپلهای جدید یک رابطه ( جدول )

با استفاده از دستور INSERT می توان تاپلهایی را به جدول اضافه نمود . فرمت کلی دستور بصورت زیر است :

```
INSERT INTO R (A1,A2,...)
VALUES ( V1,V2,.....)
```

مثال :

```
INSERT INTO P
VALUES ( 'P8', 'PN8', 'BLUE', '10', 'C3' ),
```

□ مثالی از درج با استفاده از داده های جدول دیگر :

```
INSERT INTO ACCOUNT (
SELECT LOAN-NUMBER, BRANCH-NAME, 200
FROM LOAN
WHERE BRANCH-NAME = 'نادری' ;
```

در جمله INSERT دستور FROM ..... SELECT ابتدا قبل از هر درجی ارزیابی شده و سپس نتیجه به رابطه مورد نظر درج می شود .

در غیر این صورت دستور INSERT INTO TABLE1 SELECT \* FROM TABLE1 دچار مشکل می شود .

۵-۳-۲-۳ : حکم تغییر رکورد

شکل کلی این حکم بصورت زیر است . در این صورت تمام رکوردهای جدول که حائز شرط داده شده باشند با توجه به مقدار داده شده در Set تغییر داده می شوند .

```
UPDATE TABLE
SET FIELD = عبارت
[ WHERE PERDICATE ]
```

مثال :

۱. رنگ قطعه P2 را به زرد تغییر داده و به وزن آن ۵ واحد بیفزائید .

```
UPDATE P
```

```
SET    COLOR = 'زرد',
WEIGHT = WEIGHT + 5
WHERE  P# = 'P2'
```

۲. وزن قطعات دارای وزنی با مقدار ۱۰ گرم یا بیشتر را ۵ واحد اضافه کنید و قطعات کمتر از ۱۰ گرم را ۳ واحد اضافه کنید .

```
UPDATE P
SET    WEIGHT = WEIGHT + 5
WHERE  WEIGHT = 10
UPDATE P
SET    WEIGHT = WEIGHT + 3
WHERE  WEIGHT < 10
```

- ترتیب جملات بالا مهم است .
- می توان با استفاده از جمله CASE بطور ساده تر پرس و جو را نوشت :

```
UPDATE P
SET    WEIGHT = CASE
                WHEN WEIGHT >= 10 THEN
                    WEIGHT + 5
                ELSE WEIGHT + 3
            END
```

■ Provide as a gift for all loan customers of the Perryridge branch, a \$200 savings account. Let the loan number serve as the account number for the new savings account

```
insert into account
select loan-number, branch-name, 200
from loan
where branch-name = 'Perryridge'
insert into depositor
select customer-name, loan-number
from loan, borrower
where branch-name = 'Perryridge'
and loan.account-number = borrower.account-number
```

■ Increase all accounts with balances over \$10,000 by 6%, all other accounts receive 5%.

★ Write two **update** statements:

```

update account
set balance = balance * 1.06
where balance > 10000
update account
set balance = balance * 1.05
where balance <= 10000
    
```

★ The order is important

★ Can be done better using the **case** statement (next slide)

■ Same query as before: Increase all accounts with balances over \$10,000 by 6%, all other accounts receive 5%.

```

update account
set balance = case
    when balance > 10000 then balance * 1.06
    else balance * 1.05
end
    
```

## ۶ - ۴: SQL و سطح خارجی

می دانیم دید، عبارتی نامدار از جبر رابطه ای است. در SQL برای تعریف دید از دستور زیر استفاده می شود.

```

CREATE VIEW VIEW_NAME [( COLUMN[,COLUMN, ...][,...]).]
AS SUB QUERY
    
```

تمام قدرت دستور Select در خدمت view است و در واقع مکانیزم اشتقاق view یک پرس و جو است که از طریق Select داده می شود. پرس و جوی مربوطه در زمان تعریف view اجرا نمی شود بلکه فقط به عنوان تعریف با شمای خارجی در کاتالوگ نگهداری می شود تا هرگاه لازم باشد سیستم به آن مراجعه کند.  
مثال:

```

CREATE VIEW PARTS (P#,PNAME,WT,CITY)
AS SELECT P#,PNAME, WEIGHT,CITY
FROM P
WHERE COLOR = 'RED'
    
```

با حکم بالا یک دید بنام Parts تعریف می شود.

■ A view consisting of branches and their customers

```

create view all-customer as
(select branch-name, customer-name
from depositor, account
where depositor.account-number = account.account-number) union
    
```

```
(select branch-name, customer-name
from borrower, loan
where borrower.loan-number = loan.loan-number)
```

## ۶ - ۴ - ۱ : عملیات در دید

### الف) بازیابی :

عمل بازیابی از نظر تئوری مشکلی ندارد هر چند در بعضی سیستم ها محدودیت هایی در این زمینه وجود دارد چون view ماهیتا جدول است . لذا همان حکم Select نیز برای آن عمل می کند .

مثال :

```
SELECT * FROM PARTS
WHERE P# = 'P2'
```

برای اجرای حکم بالا بایستی سیستم آن را به حکمی در سطح ادراکی تبدیل کند و برای این منظور شرط یا شرایط داده شده در تعریف دید را با شرط در حکم بازیابی ترکیب می کند .

مثال :

```
SELECT *
FROM PARTS
```

دید کاربر را به عینیت درمی آورد .

چون VIEW خود یک جدول است و لذا می توان روی آن VIEW تعریف کرد و بدین ترتیب سطوح دیگری از انتزاع را ایجاد کرد .

■ Find all customers of the Perryridge branch

```
select customer-name
from all-customer
where branch-name = 'Perryridge'
```

■ Find the average account balance of those branches where the average account balance is greater than \$1200.

```
select branch-name, avg-balance
from (select branch-name, avg (balance)
from account
group by branch-name)
as result (branch-name, avg-balance)
where avg-balance > 1200
```

□ در برخی سیستم ها در عمل بازیابی از VIEW محدودیت هایی وجود دارد که از جمله می توان مشکل تابع

جمعی را مطرح کرد .

```
CREATE VIEW PQ
```

```
AS SELECT SP.P#, SUM(SP.QTY) AS TOTALQTY
FROM SP
GROUP BY SP.P#
```

دستور زیر غیرمجاز است .

```
SELECT AVG(PQ.TOTALQTY) AS PT
FROM PQ
```

### ب) عملیات ذخیره سازی در VIEW : ( بروز در آوری دیدها )

تمام دیدهای قابل تعریف در SQL قابل به هنگام سازی نیستند . به بیان دیگر دیدهایی وجود دارند که نمی توان از طریق آنها عمل درج ، تغییر و حذف را انجام داد . دیدها را معمولا به دو دسته تقسیم می کنند .

۱. دیدهای فاقد مشکل در عملیات به هنگام سازی (پذیرا)

۲. دیدهای دارای مشکل (ناپذیرا)

آنچه که در سیستم های موجود به عنوان به هنگام سازی دیدها انجام می شود در بعضی موارد از نظر منطقی قابل دفاع نیست و یا برعکس از نظر منطقی شدنی است ولی سیستم های موجود انجام نمی دهند . یکی از ایرادهایی که به سیستم های رابطه ای می گیرند نیز بحث به هنگام سازی دید است .

#### • نظر چمبرلن در سیستم R :

دیدی قابل به هنگام سازی است اگر روی یک جدول مبنا تعریف شده باشد و هر سطر دید متناظر با سطر شخصی از جدول مبنا باشد و هر ستون دید نیز متناظر با ستون مشخص و نامداری از جدول مبنا باشد .  
مثال (۱) :

```
CREATE VIEW SUPC2
AS SELECT S#,SNAME
FROM S
WHERE CITY = 'C2'
```

فرض کنیم دستور را داشته باشیم :

```
UPDATE SUPC2
Set Sname = '****'
Where S# = 'S2'
```

این دستور بایستی به دستوری در سطح ادراکی نگاشته شود .

```
UPDATE S
SET SNAME = '****'
WHERE S# = 'S2'
AND CITY = 'C2'
```

در صورتی که بخواهیم سطری به دید درج کنیم :

```
INSERT INTO SUPC2
VALUES(S12,SN12)
```

کمترین مشکل آن است که در دو ستون city ، status پدیده هیچمقدار بروز می کند . صرفنظر از اینکه وجود این پدیده مطلوب نیست ، بروز آن می تواند یکی از قواعد جامعیت پایگاه را خدشه دار کند .

```
INSERT INTO S
VALUES <S12,SN12,.....>
```

در خصوص دیدهای تک جدولی چمبرلینی می توان گفت که غیر از پدیده هیچمقدار مشکلی ندارد .

مثال (۲) :

```
CREATE VIEW V1
AS SELECT ( S#,STATUS)
FROM S;
```

```
CREATE VIEW V2
AS SELECT STATUS,CITY
FROM S;
```

□ هر دو دید بالا روی یک جدول تعریف شده اند و هر دو دید ماهیتا حاصل عمل پرتو روی جدول S می باشند . دید V1 قابل به هنگام سازی است ( غیر از مسأله NULL VALUE در عمل درج ) و دید V2 قابل به هنگام سازی نیست

□ دید V1 موسوم به دید حافظ کلید است در حالیکه دید V2 حافظ کلید اصلی نیست . دید چمبرلینی همان دید حافظ کلید است و چون در سالهای 78-79 مفهوم کلید اصلی تعریف نشده بود لذا وی آن را مطرح نکرد .

مثال (۳) : دید آماری :

```
Create view V3(Pnum,SumQ)
AS Select P#,sum(QTY)
From SP
Group by P#;
```

این دید حاوی یک فیلد مجازی است ( به عینیت در آوردن غیر مستقیم ) :  $sumQ$  یک فیلد مجازی است و به عینه روی جدول فیلد متناظر ندارد و لذا هیچ گونه عملی روی این ستون نمی توان انجام داد .

```
INSERT INTO V3 (Pnum,SumQ)
Values < P11 , 111>
```

مقداری است برای  $sum(Qty)$  و نه برای Qty . لذا عمل درج امکان پذیر نیست .

• به هنگام سازی دیدهای حاصل عمل پیوند (ترکیب) :

طبق نظر چمبرلین این دیدها قابل به هنگام سازی نیستند اما این نظر رد شده است . فرض می کنیم بجای رابطه S داشته باشیم :

```
SX(S#,Sname,city)
SY(S#,Status)
```

و S# در هر دو رابطه SY, SX کلید است .

$SX \infty SY = S$

اگر دیدی داشته باشیم بفرم مقابل :

```
CREATE VIEW S
AS SELECT SX.S#,SNAME,STATUS,CITY
FROM SX,SY
```

WHERE SX.S# = SY.S#

در دید بالا یک پیوند داریم و پیوند از نوع  $P_K - P_K$  است. (صفت خاصه دخیل در پیوند در هر دو رابطه کلید اصلی است) دیدهای حاصل چنین پیوندهایی مشکلی در عملیات به هنگام سازی ندارند. وجود پیوندهای  $P_k - P_k$  سبب شده است که تناظر یک یک بین سطرها و ستونهای جداول مبنای زیرین وجود داشته باشد.

```
Insert Into S (S#,Sname,Status,city)
Values (S9,Sn9,20,C9)
```



```
Insert Into SX (S#,Sname,city)
Values (S9,Sn9,C9)
```

```
Insert Into SY (S#, Status)
Values (S9,20)
```

### • دیدهای ناپذیرا :

اگر این اصل پذیرفته شود که عملیات تاپلی در پایگاه داده رابطه ای از طریق کلید اصلی و یا یکی از کلید های کاندید انجام گردد، در اینصورت دید تعریف شده روی یک رابطه که فاقد کلید کاندید (اصلی) رابطه مبنای باشد، عملیات ذخیره سازی را نمی پذیرد لذا دو حالت زیر را غیر پذیرا در نظر میگیریم :

الف) دید پرتوی فاقد کلید :

ب) حاصل عمل پیوند (ترکیب) FK-FK :

این نوع دید، اگر عوارض جانبی عملیات ذخیره سازی را پذیریم، پذیرای عملیات ذخیره سازی است ولی چون این عوارض قابل توجه هستند، لذا آنرا جزء دیدهای غیر پذیرا منظور میکنند.

### • در SQL/92 به هنگام سازی دیدها از قوانین زیر پیروی می کنند :

۱. عبارت جدولی که حوزه دید را تعیین می کند نباید شامل کلمات UNION و JOIN باشد.
۲. قسمت SELECT عبارت انتخاب مستقیما شامل DISTINCT نباشد.
۳. قسمت FROM دقیقا شامل یک جدول ارجاع باشد.
۴. عبارت SELECT نباید حاوی GROUP و HAVING باشد.

### نکات مهم :

۱. قابلیت به هنگام سازی در view به گونه ای است که یا هر سه عمل INSERT و UPDATE و DELETE می توانند بر روی یک دید اعمال شوند و یا هیچ کدام را نمی توان اعمال کرد.
۲. در view این امکان وجود ندارد که بعضی ستونها را به هنگام سازی کرد و برخی ستونها را در داخل همان دید به هنگام سازی نکرد.

مثال (۳) : فرض کنید دید V5 بصورت زیر تعریف شده باشد :

```
CREATE VIEW V5
AS Select S#,Status,city
From S
```

Where Status > 15  
With check option;

S	V5
<u>S#.....Status</u>	<u>S#.....Status</u>
S1.....16	S1.....16
S2.....10	S3.....17
S3.....17	
S4.....15	

تهیه کنند S2 یا S4 در این دید وجود ندارد . آیا کاربر حق دارد عمل زیر را انجام دهد ؟

Insert Into V5  
Values (S2,18)

در این صورت بایستی جلوی عمل درج را بگیرد زیرا باعث تکرار در کلید می شود . و نیز آیا کاربر حق دارد دستور مقابل را وارد کند ؟

update V5  
Set Status = 10  
Where S# = 'S3'

در چنین مواردی در بعضی سیستم ها گزینه With check option را قرار می دهند . به این معنی که اگر عملیات درج و به هنگام سازی جامعیت اعمال شده توسط عبارت تعریف کننده دید را نقض کنند آنگاه این عملیات روی دید رد می شوند .

## ۵-۵- امکانات امنیتی SQL:

می دانیم امنیت به معنی حفاظت داده ها در قبال کاربران غیر مجاز میباشد . جنبه های مختلفی درباره امنیت مطرح است که از جمله می توان موارد زیر را نام برد :

- جنبه های قانونی و اجتماعی
- کنترل های فیزیکی
- مسائل عملیاتی
- کنترل های سخت افزاری
- پشتیبانی سیستم عامل

آنچه در این بحث حائز اهمیت است امکانات امنیتی کنترل داده ها درون بانک اطلاعاتی است که برخی از آنها با دستورات SQL تعریف می شوند. از جمله این امکانات می توان تعریف کاربران ، قوانین و امتیازات را نام برد .

کاربر :

کاربر نامی است که با استفاده از آن میتوان وارد پایگاه داده شد. برای تعریف کاربر دستورات زیر بکار می روند:

```
CREATE USER user
IDENTIFIED {BY password | EXTERNALLY}
[DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
```



[QUOTA {integer [K|M] | UNLIMITED} ON tablespace]  
[PROFILE profile]

ALTER USER user  
[IDENTIFIED {BY password | EXTERNALLY}]  
[DEFAULT TABLESPACE tablespace]  
[TEMPORARY TABLESPACE tablespace]  
[QUOTA {integer [K|M] | UNLIMITED} ON tablespace]  
[PROFILE profile]  
[DEFAULT ROLE { role [, role] ...  
| ALL [EXCEPT role [, role] ...] | NONE}]

## □ امتیازها

امتیاز، اجازه انجام یک عمل روی پایگاه داده هاست . برای اعطاء امتیاز از دستور GRANT و برای لغو آن از دستور REVOKE استفاده می شود.

■ The grant statement is used to confer authorization

**grant** <privilege list>  
**on** <relation name or view name> to <user list>

■ <user list> is:

★ a user-id

★ *public*, which allows all valid users the privilege granted

★ A role (more on this later)

■ Granting a privilege on a view does not imply granting any privileges on the underlying relations.

■ The grantor of the privilege must already hold the privilege on the specified item (or be the database administrator).

■ **select**: allows read access to relation, or the ability to query using the view

★ Example: grant users U1, U2, and U3 **select** authorization on the *branch* relation:

**grant select on branch to U1, U2, U3**

■ **insert**: the ability to insert tuples

■ **update**: the ability to update using the SQL update statement

■ **delete**: the ability to delete tuples.

■ **references**: ability to declare foreign keys when creating relations.

■ **usage**: In SQL-92; authorizes a user to use a specified domain

■ **all privileges**: used as a short form for all the allowable privileges

■ Roles permit common privileges for a class of users can be specified just once by creating a corresponding “role”

■ Privileges can be granted to or revoked from roles, just like user

■ Roles can be assigned to users, and even to other roles

■ SQL:1999 supports roles

**create role teller**

**create role manager grant select on branch to teller**

**grant update (balance) on account to teller**

**grant all privileges on account to manager**

**grant teller to manager**

**grant teller to alice, bob**

**grant manager to avi**

دستور لغو مجوز :

■ The **revoke** statement is used to revoke authorization.

**revoke**<privilege list>

**on** <relation name or view name> **from** <user list> [**restrict**|**cascade**]

■ Example:

**revoke select on branch from U1, U2, U3 cascade**

■ Revocation of a privilege from a user may cause other users also to lose that privilege; referred to as cascading of the **revoke**.

■ We can prevent cascading by specifying **restrict**:

**revoke select on branch from U1, U2, U3 restrict** With **restrict**, the **revoke** command fails if cascading revokes are required.

## ۵-۶-تعریف تراکنش

تراکنش واحد برنامه نویسی است که شامل یکسری عملیات مرتبط برای دسترسی و تغییر اطلاعات یک بانک اطلاعاتی که در جهان واقعی در حکم یک عمل واحد تلقی می شوند. معمولاً "دستورات تراکنش با دستور شروع تراکنش (begin transaction) آغاز و با یک عمل commit و یا undo پایان می پذیرد. در خصوص تراکنش چند نکته وجود دارد :

- طراحی صحیح correctness

برنامه نویس باید عملیات اجرایی یک تراکنش را بصورت واحد یکپارچه طراحی کند و این به خود dbms ربطی ندارد.

- خواندن اطلاعات

هر مورد اطلاعاتی مورد نیاز تراکنش باید فقط یکبار خوانده شود . بعبارت دیگر در داخل یک تراکنش یک رکورد دوبار خوانده نشود.

- نوشتن اطلاعات:

هر مورد اطلاعاتی مورد عمل در تراکنش در صورت تغییر فقط یکبار نوشته شود.

## ۵-۶-۱-ویژگیهای تراکنش

برای تراکنش ها چهار ویژگی نیز ذکر کرده اند که می توان آنها را بصورت زیر نام برد:

### الف : ویژگی اتمی بودن Atomicity

تراکنش ها ، ساده و غیر قابل تجزیه هستند بعبارتی کلیه عملیات هر تراکنش یا تماماً اجرا می شوند و یا هیچکدام اجرا نمی گردند.

### ب: ویژگی سازگاری consistency

تراکنش ها سازگاری پایگاه داده را حفظ می کنند. بعبارت دیگر تراکنش پایگاه داده را از یک حالت سازگار به حالت سازگار دیگری تبدیل می کند.

### ج: ویژگی جداسازی Isolation

تراکنش ها از یکدیگر مجزا هستند یعنی اثر مخرب روی یکدیگر ندارند.

### د: ویژگی های پایداری Durability

پس از آنکه تراکنش پذیرفته شد اثر آن را در بانک باقی می ماند حتی اگر سیستم اندکی بعد از کار بیفتد.

## ۵-۶-۲- مثال از تراکنش

فرض کنیم می خواهیم مبلغ ۵۰۰۰۰ ریال از حساب A به حساب B منتقل کنیم داریم :

1.read (A)

2.A:=A-50000

3.write(A)

4.read(B)

5.B:=B+50000

6.write(B)

در اینصورت با توجه به خواص تراکنش ها داریم :

الف) خاصیت سازگاری

مجموع مقادیر A و B پس از اجرای تراکنش تغییر نمی کند.

ب) خاصیت اتمی بودن:

اگر تراکنش پس از مرحله ۳ و قبل از مرحله ۶ متوقف گردد ، سیستم تضمین می کند که تغییرات در بانک ثبت نگردند.

ج) پایداری:

پس از آنکه اجرای تراکنش مورد تایید قرار گرفت و تراکنش کامل گردید ، این تغییرات در بانک پایدار خواهد بود.

د) جداسازی:

اگر بین مراحل ۳ و ۶ یک تراکنش دیگر اجازه دستیابی به تغییرات در بانک را داشته باشد باعث ناسازگاری در بانک خواهد گردید (مجموع  $A+B$  کمتر از مقدار اصلی خواهد شد) لذا نبایستی امکان اجازه تراکنش های دیگر برای به هنگام سازی پایگاه را بوجود آورد.

### ۵-۶-۳-حالت های اجرای تراکنش

الف: ناقص Aborted

در این حالت در حین اجرای تراکنش اشکالی پیش آمده است که منجر به توقف اجرای آن شده است و لذا تراکنش نیمه تمام رها می شود.

ب: برگشت Rolled back

در صورت بروز اشکال در اجرای یک تراکنش ، برای حفظ یکپارچگی اطلاعات ، اثرات احتمالی بخشی از تراکنش که اجرا شده روی بانک اطلاعاتی باید خنثی شود. به این حالت برگشت گفته می شود. مسئولیت این امر بعهده dbms است.

ج: پذیرش شده Committed

حالتی است که عملیات تراکنش بطور کامل موفقیت آمیز انجام شده و اثر آن نیز ثبت شده است پس از اجرای تراکنش خنثی کردن تغییرات احتمالی تراکنش روی بانک غیر ممکن است.

در بیشتر سیستم های پایگاه داده ، هر دستور SQL که اجرای موفق داشته باشد ، بطور خودکار پذیرفته می شود.

در SQL:1999 برای نوشتن به فرم تراکنش از دستور زیر استفاده میشود.

**begin atomic**

...

**end**

## فصل ششم:

### نرمالتر سازی رابطه ها

۶-۱ مقدمه :

ایده اصلی نرمالتر سازی رابطه ها بر مبنای رفع آنومالی های رابطه ها است. می دانیم اصطلاح آنومالی یعنی بروز وضعیت نامطلوب در انجام عمل که می تواند ناممکن بودن انجام یک عمل و یا بروز تبعات نامطلوب در انجام یک عمل و یا بروز دشواری ( فزونکاری) در عملیات باشد. برای رفع آنومالی ها باید رابطه ها نرمالتر شوند. بعنوان مثال رابطه ی مانند SPC را زیر در نظر می گیریم:

S#	P#	QTY	CITY
$c_2$	100	$p_1$	$s_1$
$c_2$	200	$p_2$	$s_1$
$c_2$	150	$p_3$	$s_1$
$c_3$	100	$p_1$	$s_2$
$c_3$	80	$p_2$	$s_2$
$c_3$	90	$p_1$	$s_3$

اهداف نرمال سازی :

- ۱- کاهش برخی از آنومالی ها
- ۲- کاهش افزونگی
- ۳- تامین طرح بهتر برای پایگاه داده قابل درک تر
- ۴- اعمال برخی قواعد جامعیتی ناشی از وابستگی تابعی

این رابطه عناصرش اتمیک ( ساده ) می باشند که به آن رابطه نرمال INF نیز می گویند. اما این رابطه در عملیات آنومالی دارد:

۱- در عمل درج: درج کن اطلاع  $\langle s_1, c_1 \rangle$  S صاکن Cv است .

این درج ناممکن است تا وقتیکه ندانیم چه قطعه ای را تهیه کرده است.

۲- در عمل به هنگام سازی:

شهر  $s_1$  را عوض کنید. عمل منطقاً تاپلی به عملی مجموعه ای تبدیل می شوند. و به نوعی به هنگام سازی منتشر شونده داریم.

۳- در عمل حذف :

با حذف اطلاع  $\langle s_3, p_1, 90 \rangle$  اطلاع ناخواسته از بین می رود (  $s_3$  ساکن شهر  $c_3$  است).

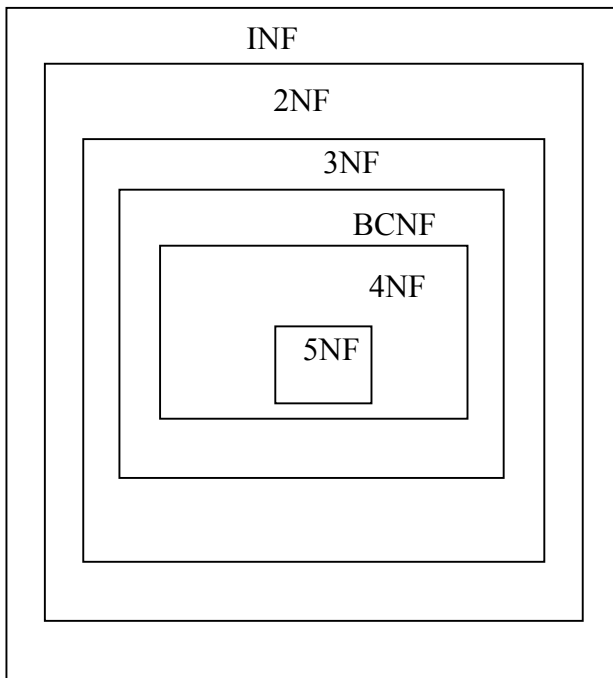
□ رابطه spc خوش ساختار نیست روش های نرمالتر سازی بعنوان یک ابزار طراحی به طراح می گوید در یک

محیط عملیاتی مشخص چه رابطه هایی داشته باشد، در هر رابطه چه صفات خاصه ای تا رفتار DBMS در عملیات

روی پایگاه با کمترین آنومالی همراه باشد. در مثال فوق دلیل بروز آنومالیهای رابطه SPC پدیده ای است بنام اختلاط اطلاعاتی یعنی اطلاعات در مورد دو پدیده ( موجودیت ) بطور غیر لازم در یکدیگر مخلوط شده اند عبارتی اطلاع در مورد تهیه کننده و شهرش با اطلاع قطعه مخلوط شده است.

### ۶-۲- شکل های نرمال ( سطوح مختلف نرمال )

سطوح مختلف نرمال را می توان بصورت زیر بیان نمود:



- 1NF -
- 2NF -
- 3NF -
- BCNF -
- 4NF -
- 5NF -
- DK/NF -

شکل مقابل سطوح مختلف نرمال را نشان می دهد.

قبل از بررسی سطوح نرمال برخی مفاهیم مورد نیاز را توضیح می دهیم:

### ۶-۳- وابستگی تابعی (functional dependency)

گوئیم صفت خاصه  $R.Y$  با  $R.X$  وابستگی تابعی دارد اگر برای هر مقدار متمایز  $X$  فقط یک مقدار  $Y$  متناظر باشد در اینصورت می گوئیم  $Y$  با  $X$  وابستگی دارد و به  $X$  دترمینان و به  $Y$  وابسته نیز گویند. و بصورت

$$X \rightarrow Y$$

نشان می دهیم

X	Y	Z
X <sub>1</sub>	Y <sub>1</sub>	Z <sub>1</sub>
X <sub>1</sub>	Y <sub>2</sub>	Z <sub>1</sub>
X <sub>1</sub>	Y <sub>3</sub>	Z <sub>1</sub>
X <sub>2</sub>	Y <sub>1</sub>	Z <sub>2</sub>
X <sub>2</sub>	Y <sub>2</sub>	Z <sub>2</sub>

مثال مقدماتی: فرض کنید رابطه مقابل را داریم

در اینصورت داریم:

$$X \rightarrow Z$$

$$Z \rightarrow X$$

$$X \not\rightarrow Y$$

در تعریف وابستگی بایستی به دو نکته توجه داشت:

۱- وابستگی تابعی باید برای تمام رابطه ها درست باشد یعنی از مفهوم و معنی آن صفات سرچشمه بگیرد نه از موارد خاص در یک یا چند رابطه . بعنوان مثال در جدول زیر وابستگی های زیادی دیده می شود که در واقع صحیح نیست.

استاد	درس	ترم	کلاس
حمیدی	اسمبلی	۷۹۱	۱۰۶
شریفی	مبانی کامپیوتر	۷۹۲	۱۰۵
رحیمی	مدار الکتریکی	۷۹۳	۱۰۴
زینالی	مدار منطقی	۸۰۱	۳۰۱

استاد → کلاس

کلاس → درس

استاد → درس

درس → استاد

۲- وابستگی تابعی برای تعریف محدودیت های پایگاه داده نیز بکار می رود. یک وابستگی تابعی ممکن است برای یک پایگاه داده درست و در پایگاه داده دیگر غلط باشد لذا طراح پایگاه داده می تواند قواعد بانک اطلاعات خود را با وابستگی تابعی نیز بیان نماید.

وابستگی های تابعی زیر را می توان برای رابطه SPC در نظر گرفت.

$(s\#, p\#) \rightarrow Qty$

$(s\#, p\#) \rightarrow city$

$(s\#) \rightarrow city$

$(s\#, p\#) \rightarrow s\#$

$(s\#, p\#) \rightarrow (city, Qty)$

### ۱-۳-۶- مفهوم وابستگی تابعی کامل (FFD)

صفت خاصه  $y$  از رابطه  $R$  با صفت خاصه  $x$  از آن FD کامل دارد هر گاه  $y$  با  $x$ ، FD داشته باشد اما با هیچکدام از اجزا تشکیل دهنده آن FD نداشته باشد و آن را بصورت  $R.X \Rightarrow R.Y$  نشان می دهیم بعنوان مثال در رابطه SPC

داریم :  $(S \neq, P \neq) \rightarrow Qty$  و پس  $(s \neq, p \neq) \Rightarrow Qty$

$s \neq \rightarrow Qty$

$p \neq \rightarrow Qty$

• اگر برای تمام صفات خاصه  $y$  در  $R$  داشته باشیم  $y \rightarrow x$  در اینصورت  $x$  را ابر کلید  $R$  می نامند و بصورت  $R \rightarrow x$  نمایش می دهند . اگر این وابستگی از نوع FFD باشد آنگاه  $X$  کلید کاندید  $R$  است .

### ۲-۳-۶- تعریف وابستگی تابعی بدیهی:

اگر  $Y$  زیر مجموعه ای از  $X$  باشد آنگاه  $x \rightarrow y$  این وابستگی تابعی را بدیهی (trival) می نامیم . بعبارت دیگر یک وابستگی تابعی را بدیهی گویند اگر و فقط اگر سمت راست آن زیر مجموعه ای از سمت چپ باشد.

□ ممکن است بعضی از وابستگی های تابعی را از وابستگی های تابعی دیگر نتیجه گرفت بعنوان مثال از وابستگی تابعی  $(city, Qty) \rightarrow (s \neq, p \neq)$  می توان دو وابستگی  $city \rightarrow Qty, (s \neq, p \neq) \rightarrow Qty$  را نتیجه گرفت.

□ مجموعه تمام وابستگی های تابعی که توسط مجموعه معینی از وابستگی های تابعی بدست می آیند را بستار رابطه گویند. عبارت دیگر اگر F مجموعه ای از FD های رابطه F باشد، مجموعه تمام FD هایی که F قابل استنتاج هستند را بستار (پوششی) (CLOSURE) مجموعه F گویند و با  $F^+$  نمایش می دهند، اولین تلاش در جهت حل این مسأله در مقاله ای که توسط آرمسترانگ منتشر شد، صورت گرفت که مجموعه ای از قوانین استنتاج که بعنوان اصول آرمسترانگ نامیده می شدند را ارائه داد که به کمک آن می توان وابستگی های تابعی جدیدی را از وابستگی های تابعی موجود استنتاج کرد.

### ۶-۳-۳- اصول آرمسترانگ

۱- قاعده انعکاسی (Reflexivity) :

اگر B زیر مجموعه ای از A باشد در اینصورت  $A \rightarrow B$

۲- قاعده افزایش (augmentation) اگر  $A \rightarrow B$  در اینصورت  $(A, C) \rightarrow (B, C)$

۳- قاعده تعدی (transitivity) : اگر  $A \rightarrow B$  و  $B \rightarrow C$  در اینصورت  $A \rightarrow C$

۴- قاعده تجزیه پذیری (decomposition) : اگر  $A \rightarrow (B, C)$  در اینصورت  $A \rightarrow B, A \rightarrow C$

۵- قاعده اجتماع (union) : اگر  $A \rightarrow B$  و  $A \rightarrow C$  در اینصورت  $A \rightarrow (B, C)$

۶- قاعده شبه تعدی (psoudo transitivity) : اگر  $A \rightarrow B$  و  $(C, B) \rightarrow D$  در اینصورت  $(A, C) \rightarrow D$

۶- قاعده ترکیب : اگر  $A \rightarrow B$  و  $C \rightarrow D$  در اینصورت  $(A, C) \rightarrow (B, D)$

### ۶-۳-۴- بستار یک مجموعه از صفات خاصه

اگر F مجموعه ای از FD باشد، گاه لازم می آید که مجموعه تمام صفات خاصه رابطه R را که با یک صفت خاصه یا مجموعه ای از صفات خاصه مثلاً A از رابطه R وابستگی داشته باشند، مشخص نماییم این مجموعه از صفات خاصه را بستار A نامیده و آن را با  $A^+$  نمایش می دهیم. می توان  $A^+$  را با محاسبه  $F^+$  و انتخاب آن FD هایی که در آن A درمینان است بدست آورد.

**تمرین** : فرض کنید  $ad \rightarrow C, A \rightarrow D, CD \rightarrow B$  نان دهید AD ابر کلید است ولی کلید کاندید نیست.

مثال : فرض کنید متغیر رابطه ای R با صفات خاصه A, B, C, D, E, F, FD زیر داده شده است نشان دهید وابستگی

تابعی  $(A, D) \rightarrow F$  برای R برقرار است.

$R=(A, B, C, D, E, F)$

$FD=\{ A \rightarrow (B, C), B \rightarrow E, (C, D) \rightarrow (E, F)\}$

1)  $A \rightarrow (B, C)$



- 2)  $A \rightarrow C$  تجزیه  
 3)  $(A,D) \rightarrow (C,D)$  بسط پذیری  
 4)  $(C,D) \rightarrow (E,F)$   
 5)  $(A,D) \rightarrow (E,F)$   
 6)  $(A,D) \rightarrow F$

مثال : رابطه R با صفات خاصه  $R=(U,V,W,X,Y,Z)$  و وابستگی تابعی F بصورت زیر داده شده است .  $F^+$  را محاسبه کنید.

$$F = \{ U \rightarrow (X,Y), X \rightarrow Y, (X,Y) \rightarrow (Z,V) \}$$

$$F^+ = \{ U \rightarrow X, U \rightarrow Y, X \rightarrow Y, (X,Y) \rightarrow (Z,V), U \rightarrow (Z,V) \}$$

### ۶-۳-۵- مجموعه وابستگی بهینه:

با استفاده از قواعد سه گانه زیر می توان یک مجموعه وابستگی را به مجموعه بهینه معادل آن تبدیل کرد:

- ۱- سمت راست هر وابستگی فقط یک صفت خاصه باشد
  - ۲- هر صفتی که  $F^+$  را تغییر نمی دهد از سمت چپ حذف شود
  - ۳- وابستگی های تکراری و اضافی حذف شود.
- استنتاج منطق بعضی وابستگیها از وابستگیهای دیگر به ما امکان می دهد تا با داشتن مجموعه ای از وابستگی های رابطه مجموعه کمینه وابستگی ها را بدست آورد.
- مثال : با توجه به مثال قبل مجموعه وابستگی پوششی بهینه را بدست آورید.

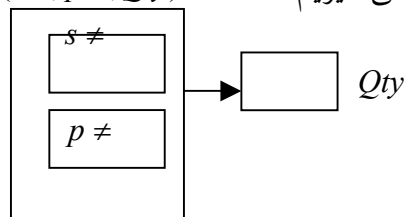
$$F^+ = \{ U \rightarrow (X,Y), X \rightarrow Y, (X,Y) \rightarrow (Z,V), U \rightarrow (Z,V) \}$$

- ◆  $U \rightarrow (X,Y) \Rightarrow U \rightarrow X, U \rightarrow Y$
- ◆  $U \rightarrow (Z,V) \Rightarrow U \rightarrow Z, U \rightarrow V$
- ◆  $(X,Y) \rightarrow (Z,V) \Rightarrow X \rightarrow Y, X \rightarrow (Z,V)$
- ◆  $X \rightarrow (Z,V) \Rightarrow X \rightarrow Z, X \rightarrow V$
- ◆  $F_{OPT} = \{ U \rightarrow X, U \rightarrow Y, U \rightarrow Z, U \rightarrow V, X \rightarrow Y, X \rightarrow Z, X \rightarrow V \}$

### ۶-۳-۶- نمودار وابستگی تابعی

می توان وابستگی تابعی را با استفاده از نمودار نشان داد. در این نمودار صفات خاصه در مستطیل قرار می گیرند و خطی جهت دار از آنها به هر یک از صفات وابسته رسم می شود.

مثال: جدول sp را در نظر می گیریم  $sp(s \neq, p \neq, Qty)$



## ۶-۴-۶- نرمال سازی (normalization)

### ۶-۴-۱- رابطه نرمال یک INF

رابطه ای را INF گویند اگر مقادیر تمام صفات خاصه اش اتمیک باشند.

مثال : رابطه زیر را در نظر می گیریم:

FIRST : (S#,P#,QTY,CITY,STATUS)

FD ها رابطه بصورت زیر است:

$(S\#,P\#) \rightarrow Qty$  ,  $(S\#,P\#) \rightarrow CITY$  ,  $(S\#,P\#) \rightarrow STATUS$

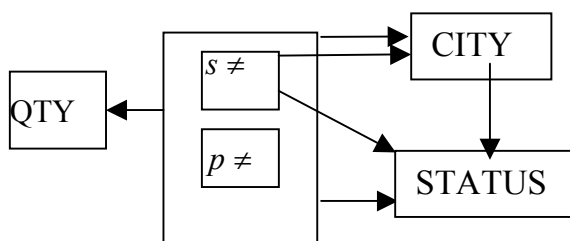
$(S\#) \rightarrow CITY$  : هر تهیه کننده ای در یک شهر ساکن است

$(S\#) \rightarrow STATUS$  : هر تهیه کننده ای یک مقدار وضعیت دارد

$CITY \rightarrow STATUS$  : تمام تهیه کنندگان ساکن یک شهر یک وضعیت دارد:

نمودار FD نیز بفرم مقابل است .

First



$s \neq$	$p \neq$	Qty	city	Status
$s_1$	$p_1$	۱۰۰	$c_1$	۱۰
$s_1$	$p_2$	۱۲۰	$c_1$	۱۰
$s_1$	$p_3$	۸۰	$c_1$	۱۰
$s_2$	$p_1$	۹۰	$c_2$	۱۵
$s_3$	$p_1$	۱۰۰	$c_2$	۱۵
$s_4$	$p_1$	۶۰	$c_1$	۱۰

### ♦ آنومالیاها:

۱- درج کن اطلاع  $\langle sv, c3, 14 \rangle$

این درج ناممکن است تا ندانیم چه قطعه ای تهیه کرده است.

۲- حذف کن  $\langle s3, p1, 100 \rangle$

منجر به حذف اطلاع ناخواسته  $\langle s3, c2, 15 \rangle$  می شود

رابطه FIRST رابطه خوش ساختاری نیست ، این رابطه باید با انتخاب پرتوهای مناسب به دو رابطه تجزیه شود:

$second(s \neq, status, city)$  و  $sp(s \neq, p \neq, Qty)$

نکته : رابطه FIRST باید بگونه ای تجزیه شود که در رابطه های حاصله FD ناکامل وجود نداشته باشد.

### ۶-۴-۲ - رابطه 2NF

رابطه ای 2NF است که :

۱- INF باشد

۲- هر صفت خاصه غیر کلید با کلید اصلی وابستگی تابعی کامل داشته باشد.

با توجه به تعریف 2NF می بینیم رابطه FIRST ، 2NF نیست، رابطه SECOND و SP هر دو 2NF می باشند.

نکته ۱- FD های بین مجموعه صفات خاصه یک محیط بیانگر قوانین سمانتیک حاکم بر آن محیط می باشند. بعنوان مثال وقتی می گوئیم درس  $PR \neq \rightarrow CO \neq$  استاد یعنی این قاعده بر محیط حاکم است که هر استاد فقط یک درس می دهد. این قوانین سمانتیک باید بنحوی به سیستم داده شود. اینگونه قواعد نوعی قواعد جامعیتی برگرفته از محیط عملیاتی هستند که موسوم به قوانین جامعیت ناشی از وابستگی تابعی می باشند.

نکته ۲- برای تبدیل INF به 2NF از عملگر پرتو بطور مناسب استفاده می شود.

### • آنومالیهای رابطه SECOND

۱- در درج : درج کن اطلاع  $\langle C5,18 \rangle$  : وضعیت داده شده به شهر C5 ، ۱۸ است این عمل ناممکن است تا ندانیم چه تهیه کننده ای در شهر ساکن است . زیرا کلید  $S \neq$  است.

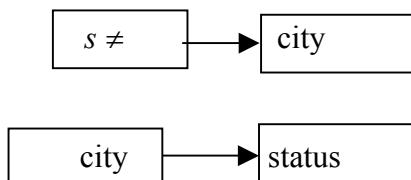
۲- در حذف : می دانیم تهیه کنندگانی ساکن شهرهایی هستند. اطلاع  $\langle S5,15 \rangle$  را حذف کن این حذف منجر به حذف اطلاع  $\langle C4,15 \rangle$  می گردد.

۳- در به هنگام سازی : وضعیت داده شده به شهر C2 را عوض کن در اینجا عمل تاپلی به عمل مجموعه ای تبدیل میشود.

$s \neq$	city	Status
$s_1$	$c_1$	۱۰
$s_2$	$c_2$	۲۰
$s_3$	$c_2$	۲۰
$s_4$	$c_1$	۱۰
S5	c4	

رابطه second هم باید با عملگر پرتو مناسب به دو رابطه تجزیه شود. فرض کنیم این رابطه به دو رابطه

$cs(city, status)$  و  $sc(s \neq, city)$  تجزیه شود.



CS	
city	status
$c_1$	۱۰
$c_2$	۲۰
$c_4$	۱۰

SC	
$s \neq$	City
$s_1$	$c_1$
$s_2$	$c_2$
$s_3$	$c_3$

مشخص است با ترکیب SC و CS هر گاه لازم باشد به رابطه SECOND می رسمیم.

### • علت آنومالیهای SECOND

در رابطه SECOND نوعی وابستگی خاص بنام وابستگی با واسطه (از طریق تعدی) وجود دارد.  
تعریف وابستگی با واسطه:

در رابطه  $R(A,B,C)$  اگر داشته باشیم  $B \rightarrow C$  ,  $B \rightarrow A$  ,  $A \rightarrow B$

می گوئیم C به A از طریق B وابسته است :  $A \rightarrow C \Rightarrow (A \rightarrow B , B \rightarrow C)$

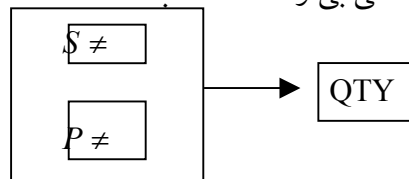
در مثال قبل داریم:  $S\# \rightarrow CITY$  و  $CITY \rightarrow STATUS$  می گوئیم STATUS ضمن اینکه خود مستقیماً بی واسطه با  $S\#$  وابستگی دارد از طریق CITY نیز به آن وابسته است.

### ۶-۴-۳: رابطه 3NF

رابطه ای را 3NF گویند هر گاه:

۱- 2NF بوده

۲- هر صفت خاصه غیر کلید با کلید اصلی وابستگی بی واسطه داشته باشد.



مثال: رابطه SP و 3NF است

### ۶-۴-۴: رابطه BCNF

این رابطه تعریفی مستقل از سطوح کلاسیک کادی دارد :

رابطه ای BCNF است که در آن هر دترمینان کلید کاندید باشد .

مثال: رابطه FIRST ، BCNF نیست زیرا در رابطه داریم:  $S\# \rightarrow City$  و  $S\#$  دترمینان است اما کلید کاندید نیست .

در سطوح کلاسیک Codd مفهوم کلید کاندید مطرح نیست ولیکن در BCNF مطرح است و چون یک رابطه ممکن

است بیش از یک کلید کاندید داشته باشد BCNF باید بیشتر بررسی شود .

هر رابطه BCNF 3NF است ولی هر 3NF ای BCNF نیست . بلکه باید بررسی شود. لذا دو حالت را در نظر

می گیریم :

#### الف: رابطه هایی با یک کلید کاندید .

در این حالت می توان گفت : اگر رابطه 3NF باشد قطعاً BCNF هم هست .

مثال: رابطه S و SP

#### ب: رابطه هایی با بیش از یک کلید کاندید :

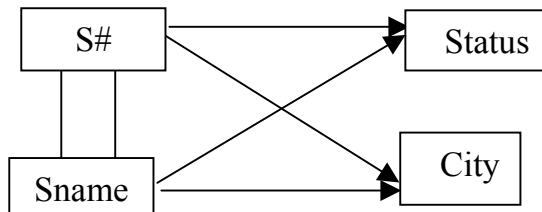
در این حالت نیز می توان دو حالت را در نظر گرفت

۱- عدم وجود همپوشانی در کلیدهای کاندید

۲- وجود همپوشانی در کلیدهای کاندید

و منظور از همپوشانی : اگر داشته باشیم  $(y,z)$  ,  $(x,y)$  ,  $R : (x,y)$  وجود  $y$  عنصر مشترک را همپوشانی گویند .  
 مثال ۱ رابطه  $S$  را در نظر می گیریم . فرض کنیم علاوه بر  $S\#$  ،  $Sname$  هم کلید کاندید باشد ( اسامی تکراری نداشته باشیم )

طبق تعریف کلیدهای کاندید نمودار  $FD$  بصورت زیر است :



این رابطه  $BCNF$  است زیرا هر دو ترمینان کلید کاندید و  $3NF$  هم می باشد .

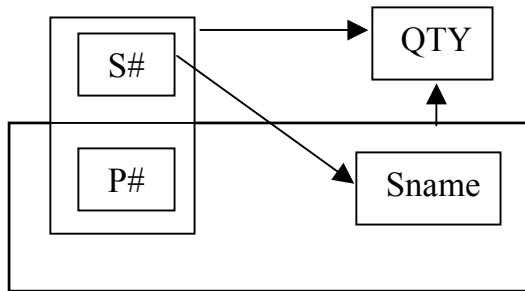
این رابطه  $1NF$  است زیرا عناصرش اتمیک هستند .

این رابطه  $2NF$  است زیرا  $1NF$  است و وابستگی ناکامل نداریم .

این رابطه  $3NF$  است زیرا تعدی نداریم .

◆ در این حالت (نبور صفت مشترک) اگر  $3NF$  است  $BCNF$  نیز می باشد.

مثال ۲ رابطه  $SPS$  را در نظر بگیریم :  $( S\# , P\# , SNAME , QTY )$  در اینجا دو کلید کاندید داریم که با هم همپوشانی دارند . نمودار وابستگی تابعی بفرم زیر است :



این رابطه  $BCNF$  نیست زیرا  $S\#$  دترمینان

است ولی کلید کاندید نیست . می خواهیم

بررسی کنیم این رابطه چند  $NF$  است .

◆ این رابطه  $1NF$  است زیرا صفات خاصه اش اتمیک هستند .

◆  $2NF$  نیز می باشد زیرا وابستگی ناکامل نداریم .

البته ظاهراً به نظر می رسد وابستگی ناکامل وجود دارد ولیکن اینطور نیست زیرا  $Sname$  خود جزئی از کلید کاندید است در حالیکه در تعریف  $2NF$  کادی آمده است هر صفت خاصه غیر کلید و اصلاً عضویت صفت خاصه در کلید کاندید مطرح نیست . لذا  $2NF$  می باشد .

◆ این رابطه  $3NF$  نیز می باشد زیرا تعدی وجود ندارد .

می بینیم SPS ، 3NF است ولی BCNF نیست . نکته جالبتر آنکه رابطه sps اختلاط اطلاعاتی دارد با این همه با داشتن دو کلید کاندید 3NF است در حالیکه معمولاً وجود پدیده اختلاط اطلاعاتی رابطه را در حد 1NF یا حداکثر 2NF نگه می دارد .

**نتیجه :** صرف گفتن رابطه ای اختلاط اطلاعاتی دارد لزوماً معنایش این نیست که سطح نرمالیتی آن پایین است . در عمل برای طراحی رابطه ها تا سطح BCNF نرمال می شوند . سطوح بالاتر بیشتر جنبه تئوریک و پژوهشی دارد و معنایش این است که تقریباً تمام رابطه هایی که BCNF هستند عملاً 5NF و 4NF هستند بعبارت دیگر رابطه هایی که BCNF باشد اما 4NF و یا 5NF نباشند بسیار کم اند .

**مثال ۳ :** رابطه ای که 3NF هست اما BCNF نیست .

فرض کنید در محیط آموزشی قواعد زیر موجودند :

۱ . یک دانشجو یک درس را فقط با یک استاد اخذ می کند .

۲ . یک استاد فقط یک درس تدریس می کند .

۳ . درس ممکن است توسط بیش از یک استاد تدریس شود .

در این رابطه دو کلید کاندید داریم :

SCP

ST#	CO#	PR#
ST <sub>1</sub>	C <sub>1</sub>	P <sub>1</sub>
ST <sub>2</sub>	C <sub>1</sub>	P <sub>1</sub>
ST <sub>1</sub>	C <sub>2</sub>	P <sub>2</sub>
ST <sub>2</sub>	C <sub>2</sub>	P <sub>3</sub>
ST <sub>3</sub>	C <sub>2</sub>	P <sub>2</sub>

و ST# و CO#

ST# و PR#

و کلیدهای کاندید با هم همپوشانی دارند .

این رابطه BCNF نیست زیرا PR# دترمینان است ولی کلید کاندید نیست .

اما 3NF هست .

PR# → CO#

(ST# , CO#) → PR#

#### ۶-۴-۵ - رابطه 4NF

(multivalued dependency)

♦ وابستگی چند مقداری MVD

وابستگی چندمقداری نوعی وابستگی بین دو مجموعه مستقل از صفات خاصه است . وابستگی چندمقداری

(A<sub>1</sub>,A<sub>2</sub>,...,A<sub>n</sub>) →→ (B<sub>1</sub>,B<sub>2</sub>,...,B<sub>m</sub>) در رابطه R برقرار است اگر برای دو تاپل t و u در R که در تمام مقادیر A

مشترکند تاپل دیگر V وجود داشته باشد که :

۱ . در مقادیر A با t و u مشترک باشد .

۲ . در مقادیر B با t مشترک باشد .

۳. در تمام ستونهای دیگر R با u مشترک باشد .

**مثال ۱** جدول تدریس اساتید را شامل کد استاد ، کد دانشکده ، شهر دانشکده ، کد درس و کتاب درس در نظر می گیریم . فرض کنیم دانشکده هایی که استاد در آنها تدریس می کند و دروسی که درس می دهد از هم مستقل باشند یعنی وابستگی تابعی نداشته باشند . اگر استاد در چند دانشکده درس بدهد و دروس مختلف را نیز تدریس کند افزونگی داریم . با توجه به جدول مقابل داریم :

دانشکده های استاد (100) و نیز دروسی که تدریس می کند تکرار شده است ( افزونگی ) این در حالی است که جدول فوق تا سطح BCNF نرمال سازی شده است .

PR#	College	City	Co#	Book
100	01	تهران	C1	B1
100	02	قزوین	C1	B1
100	01	تهران	C2	B2
100	02	قزوین	C2	B2
100	02	تهران	C3	B3
	02	قزوین	C3	B3

**مثال ۲** : رابطه CTX حاوی اطلاعات درس ، مدرس و کتاب در نظر می گیریم . یک درس می تواند توسط هر یک از مدرسین مشخص شده و با استفاده از تمام کتابهای مشخص شده تدریس شود . مثلاً درس C1 می تواند توسط t1 و t2 تدریس شود هم با استفاده از کتاب x1 و هم با استفاده از کتاب x2 . در واقع می بینیم به یک صفت خاصه مجموعه ای از مقادیر متناظر است .

C	T	X
c <sub>1</sub>	$\left. \begin{matrix} t_1 \\ t_2 \end{matrix} \right\}$	$\left. \begin{matrix} x_1 \\ x_2 \end{matrix} \right\}$
c <sub>1</sub>		
c <sub>2</sub>	t <sub>1</sub>	$\left. \begin{matrix} x_1 \\ x_3 \end{matrix} \right\}$
	t <sub>1</sub>	

$$C \twoheadrightarrow X \text{ و } C \twoheadrightarrow T$$

می توان وابستگی چند مقداری را بفرم زیر تعریف نمود :

رابطه R با صفات خاصه A و B و C را در نظر بگیریم .

می گوئیم B با A وابستگی چندمقداری دارد و چنین نمایش می دهیم

$A \twoheadrightarrow B$  اگر و فقط اگر مجموعه مقادیر B متناظر مقادیر A و C تنها به نداشته باشد .

فاگین نشان داد که در رابطه  $R(A,B,C)$  وابستگی چندمقداری  $A \twoheadrightarrow B$  وجود دارد اگر و فقط اگر وابستگی چندمقداری  $A \twoheadrightarrow C$  نیز برقرار باشد . به بیان دیگر در یک رابطه با سه صفت خاصه ، همیشه وابستگی چند مقداری بصورت جفت وجود دارد .

♦ **تعریف :** رابطه‌ای را 4NF گویند اگر و فقط اگر یک وابستگی چندمقداری مثل  $A \twoheadrightarrow B$  در R وجود داشته باشد تمام صفات خاصه R با A وابستگی تابعی داشته باشند. به بیان دیگر همه وابستگی‌های موجود در R بصورت  $X \rightarrow K$  باشند. (یعنی یک وابستگی تابعی بین صفات خاصه X و کلید کاندید K). بر اساس این تعریف می‌توان نتیجه گرفت: **رابطه R با سه صفت خاصه در چهارمین صورت نرمال است اگر BCNF باشد و تمام MVD های آن FD باشند.**

می‌بینیم رابطه CTX، 4NF نیست زیرا یک MVD دارد که FD نیست ( $C \twoheadrightarrow X$ ).  
اگر CTX را به دو رابطه CT و CX تجزیه کنیم CT و CX رابطه 4NF هستند.

### ۶-۴-۶- رابطه 5NF

• تعریف وابستگی پیوندی: Join Dependency:

اگر R یک رابطه و ستونهای هر یک از رابطه های  $R_1, R_2, \dots, R_n$  زیرمجموعه ای از ستونهای R باشند، آنگاه R دارای وابستگی پیوندی روی  $R_1, R_2, \dots, R_n$  است اگر و تنها اگر داشته باشیم:

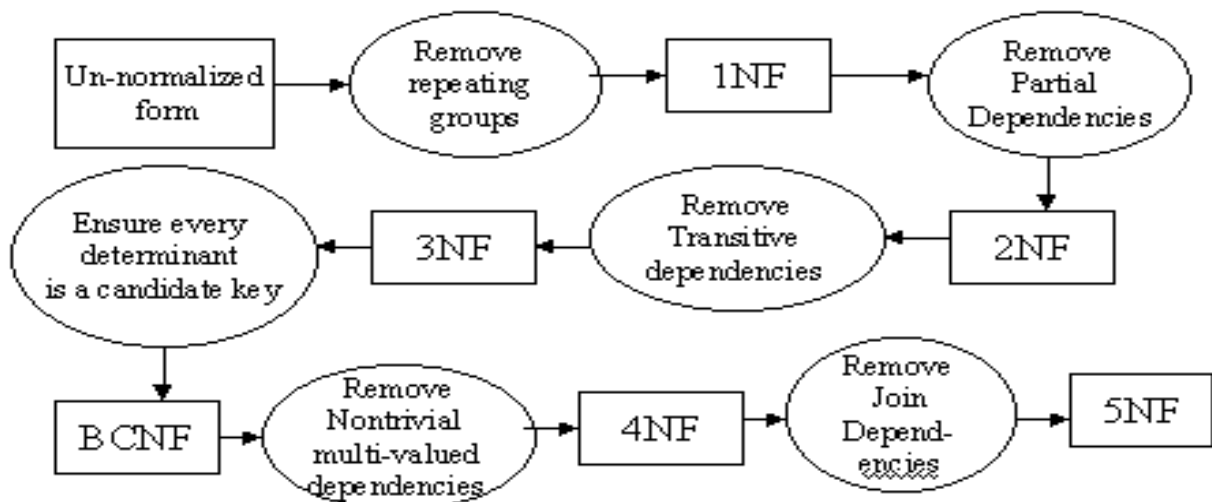
$$R = R_1 \bowtie R_2 \bowtie R_3 \dots \bowtie R_n$$

• **رابطه 5NF**

رابطه R را 5NF گویند اگر و تنها اگر فقط به کلیدهای کاندیدش وابستگی پیوندی داشته باشد. عبارتی دیگر وجود هر وابستگی پیوندی در آن ناشی از کلیدهای کاندید باشد. از این تعریف این نتیجه بدست می‌آید که اگر بتوانیم یک وابستگی پیوندی در رابطه R پیدا کنیم که در همه پرتوهایش کلید کاندید رابطه وجود نداشته باشد رابطه 5NF نیست.

دیت و فاگین نشان داده اند که:

- اگر رابطه ای 3NF باشد و تمام کلیدهای کاندید آن صفات ساده باشند آن رابطه 5NF است.
- اگر رابطه ای BCNF باشد و حداقل یکی از کلیدهای کاندید آن صفات ساده باشند آن رابطه 4NF است.





## ۶-۵- تجزیه خوب و بد

در فرایند نرمالترسازی مواردی وجود دارد که در آنها تجزیه یک رابطه به چند گونه امکان پذیر است. طراح بایستی تجزیه خوب و بد را باز شناسد. بعنوان مثال رابطه S ECOND را در نظر می گیریم :

SECOND(S#, CITY, STATUS)

وابستگی های تابعی این رابطه بفرم S# → CITY ، STATUS ، CITY → STATUS ، STATUS → S# می باشد.

قبلاً این رابطه به دو رابطه S C(S#, City) و C S(City, Status) تجزیه شد. این تجزیه تنها تجزیه ممکن نیست بلکه تجزیه های دیگری متصور است :

$$C : \begin{cases} SS(S\#, Status) \\ CS(City, Status) \end{cases} \quad B : \begin{cases} SC(S\#, City) \\ SS(S\#, Status) \end{cases}$$

کدامیک از این سه تجزیه را باید انتخاب کرد ؟

تجزیه B مطلوبیت اولیه را ندارد زیرا مشکلاتی در آن وجود دارد. مثلاً نمی توان این اطلاع را که شهر خاصی دارای مقدار وضعیت خاصی است در بانک درج کرد تا زمانی که ندانیم چه تهیه کننده ای در آن شهر ساکن است. از نظر تئوری تجزیه ای بهتر است که دو رابطه حاصل از آن از هم مستقل باشند. اگر رابطه R به دو رابطه R1 و R2 تقسیم شود گوئیم R1 و R2 از هم مستقلند اگر شرایط قضیه ریسانن را داشته باشند :

## ۶-۵-۱- قضیه ریسانن :

اگر R1 و R2 دو پرتو مستقل از R باشند، این دو پرتو از یکدیگر مستقلند اگر و فقط اگر

۱. تمام وابستگی های تابعی موجود در رابطه R در R1 و R2 با هم وجود داشته باشند و یا از

وابستگی های موجود در R1 و R2 منطقاً قابل استنتاج باشند.

۲. صفات خاصه مشترک در R1 و R2 اقلأ در یکی از آنها کلید کاندید باشد.

باتوجه به قضیه ریسانن می بینیم تجزیه اولیه، تجزیه خوبی است زیرا CITY صفت خاصه مشترک در یکی از رابطه ها یعنی CS کلید کاندید است و تمام وابستگی های تابعی قابل استنتاج هستند.

S# → City

City → Status و S# → Status منطقاً قابل استنتاج است.

بررسی تجزیه B : در این تجزیه داریم S# → City و S# → Status و نمی توان وابستگی City → Status را از این دو وابستگی منطقاً استنتاج کرد.

لازم به ذکر است که در تجزیه یک شما (Schema) به چند شمای کوچکتر باید تجزیه بدون گمشدگی اطلاعات باشد یعنی بازای تمام جداول مربوطه از پیوند طبیعی آن جداول دقیقاً جداول اصلی بدست آید.

۶-۵-۲-رابطه اتمیک:

رابطه‌ای که به عناصر مستقل تجزیه نشود (طبق رابطه ریسانس) به رابطه اتمیک موسوم است. اتمیک بودن به این معنا نیست که نباید تجزیه شود ولی لزومی به تجزیه آنها نیست یعنی در صورت تجزیه ممکن است به رابطه نرمالتری نرسید. به عنوان مثال:

$S(S\#, Sname, Status, City)$  می‌تواند به دو رابطه  $SX(S\#, Sname, Status)$  و  $SY(S\#, City)$  تجزیه شود که از نظر نرمالیتی فرقی ندارد و ممکن است بدلائیل دیگر تجزیه شده باشد.

۶-۶- نمونه مسائل این فصل:

مجموعه حداقل FD های این رابطه را بدست آورید.

حل: با توجه به نمودار FD ها داریم:

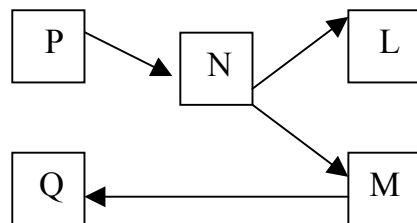
- |                      |                           |
|----------------------|---------------------------|
| 1. $P \rightarrow N$ | 4. $N \rightarrow Q$      |
| 2. $N \rightarrow L$ | 5. $N \rightarrow (L, M)$ |
| 3. $P \rightarrow L$ | 6. $M \rightarrow Q$      |

FD شماره 3 افزونه است زیرا منطقاً از FD های 1 و 2 قابل

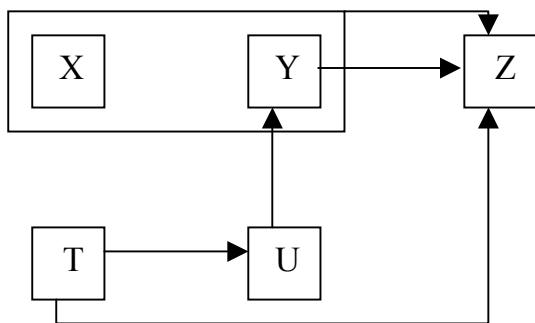
استنتاج است. از FD شماره 2 و 5 داریم:  $N \rightarrow M$

با توجه به FD های 6 و 7، FD شماره 4 افزونه است، بنابراین مجموعه حداقل FD ها بصورت زیر است:

$P \rightarrow N$  و  $N \rightarrow L$  و  $N \rightarrow M$  و  $M \rightarrow Q$



۲- در نمودار FD های زیر مجموعه حداقل FD ها را بدست آورید.



1.  $(X, Y) \rightarrow Z$
2.  $Y \rightarrow Z$
3.  $T \rightarrow U$
4.  $U \rightarrow Y$
5.  $T \rightarrow Z$

FD شماره 5 افزونه است.

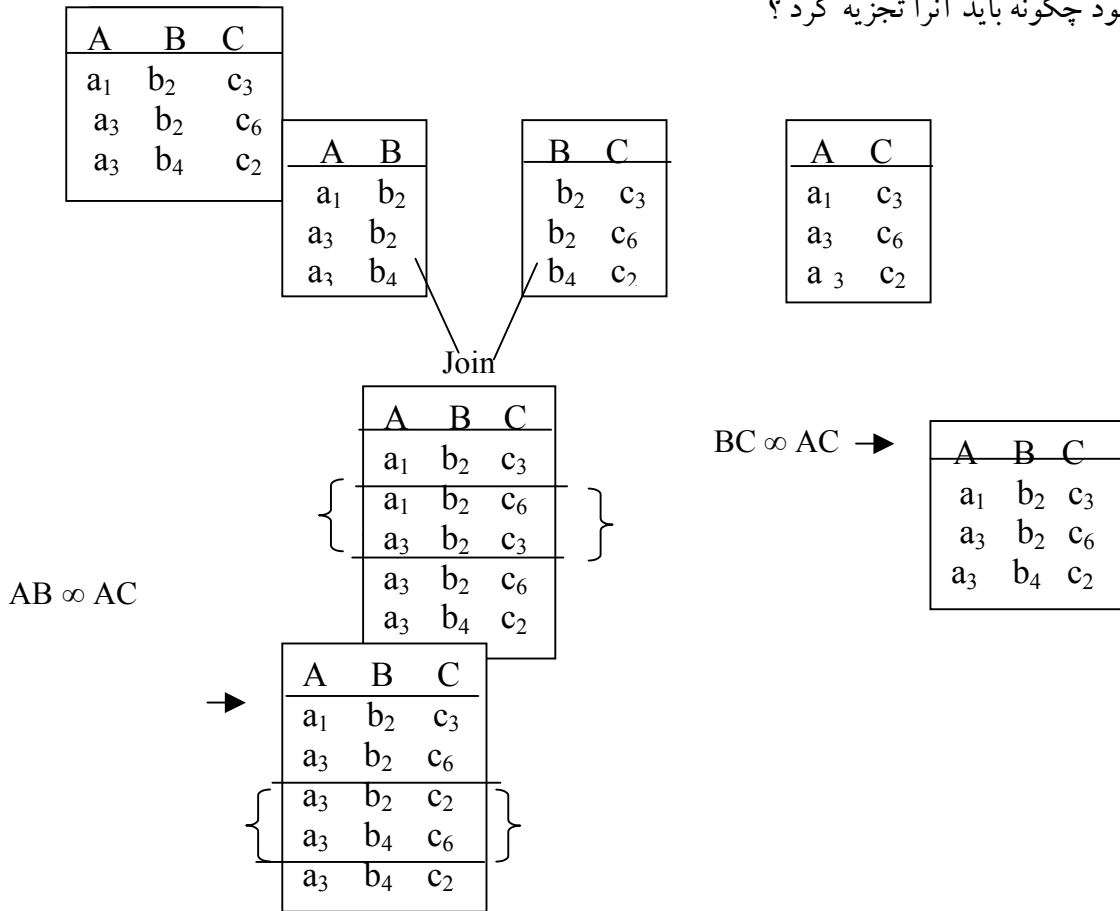
FD شماره 1 نیز افزونه است. چرا؟

پس داریم:  $T \rightarrow U$  ,  $Y \rightarrow Z$

[ چون از  $Z \rightarrow Y$  می‌توان نتیجه گرفت  $(X, Y) \rightarrow Z$  زیرا اگر  $(X, Y) \rightarrow Z$  یعنی  $(X1, Y1, Z1)$  و

(X1, Y1, Z1) و چون (Y1, Z2), (Y1, Z1) لذا  $Y \not\rightarrow Z$  خلاف فرض است پس  $Y \rightarrow Z$  [

۳- رابطه  $R(A,B,C)$  را در نظر می گیریم . در یک لحظه از حیات رابطه ، بسط آن چنین است فرض کنیم که این رابطه باید تجزیه شود چگونه باید آنرا تجزیه کرد ؟



می بینیم که تجزیه  $R$  بصورت  $R : (AB,AC)$  یا  $R : (AB,BC)$  مناسب نیست زیرا با پیوند تجزیه ها تاپل افزونه بروز می کند اما در تجزیه  $R : (BC,AC)$  این پدیده نامطلوب را در پی ندارد لذا این تجزیه مناسب است.

## فصل هفتم :

# معماری سیستم بانک اطلاعاتی :

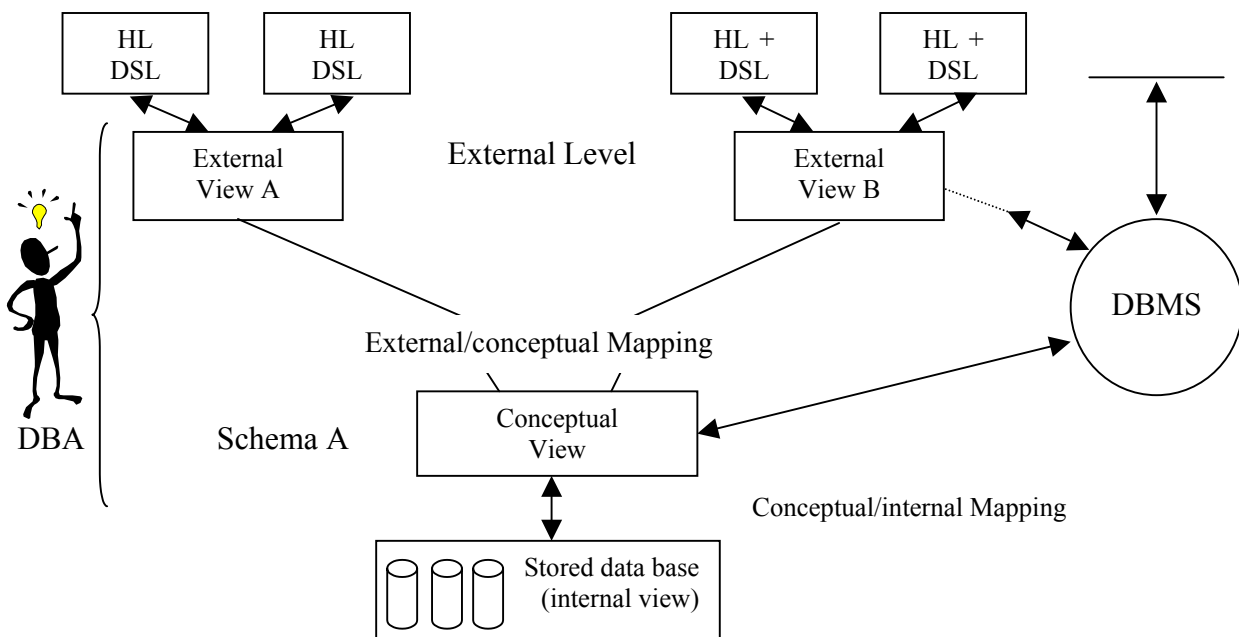
۳-۱- مقدمه :

می دانیم طراح بانک، تصور یا درک خود را از محیط عملیاتی (جهان واقعی) و در واقع دید خود را از داده های عملیاتی محیط بصورت نمودار E/R متجلی می سازد. این نمودار نمایش داده های عملیاتی بانک در بالاترین سطح انتزاع می باشد و از سویی دیگر محیط فیزیکی بانک که پایین ترین و عینی ترین سطح بانک است مجموعه ای است از فایلها با ساختار مشخص و ارتباطات بین آنها، لذا بایستی بین بالاترین سطح انتزاعی و پایین ترین سطح عینی آن سطوح واسطی وجود داشته که در این سطوح واسط داده های عملیاتی محیط هم بصورتی که طراح می بیند و هم بصورتی که هر یک از کاربران به نحوی تعریف شوند.

با این توصیف در می یابیم که یک سیستم بانک اطلاعاتی سیستمی است چند سطحی که طبعاً معماری خاص خود را دارد. از آنجاییکه طراحان مختلف سیستم های بانکی طرحهای متفاوتی برای معماری چنین سیستمی ارائه و پیاده سازی کرده اند لذا ANSI نیز طرحی استاندارد برای معماری سیستم بانک اطلاعاتی عرضه کرده است.

### ۲-۷- معماری ANSI

معماری ANSI به سه سطح مختلف تقسیم بندی می شود که به ترتیب عبارتند از: سطح داخلی، سطح ادراکی و سطح خارجی. شکل زیر بیانگر این معماری است:



همانطور که در شکل دیده می شود معماری سیستم بانک اطلاعاتی از اجزاء زیر تشکیل شده است

۱- زبان میزبان HL

۲- زبان فرعی داده ای DSL

۳- دید خارجی External View

۴- دید مفهومی Conceptual View

۵- دید داخلی Internal View

۶- تبدیلات بین سطوح mapping

۷- کاربر User

۸- اداره کننده پایگاه DBA

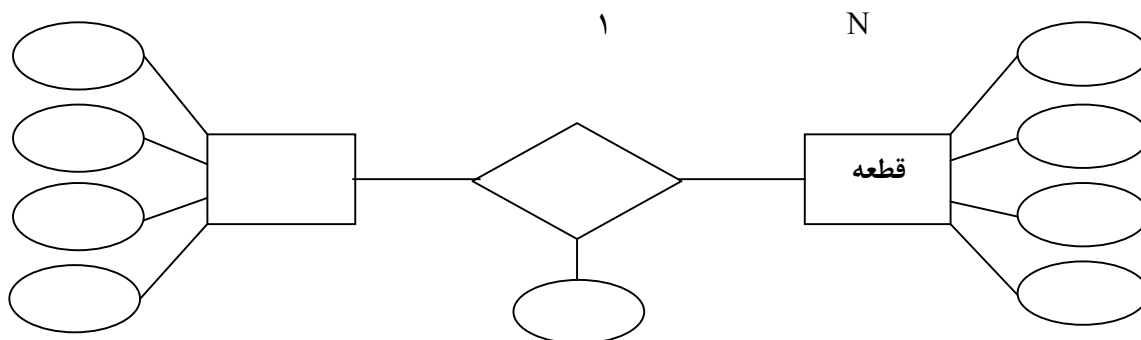
۹- سیستم مدیریت بانک اطلاعاتی DBMS

### ۷-۳- شرح اجزاء معماری پایگاه داده ها:

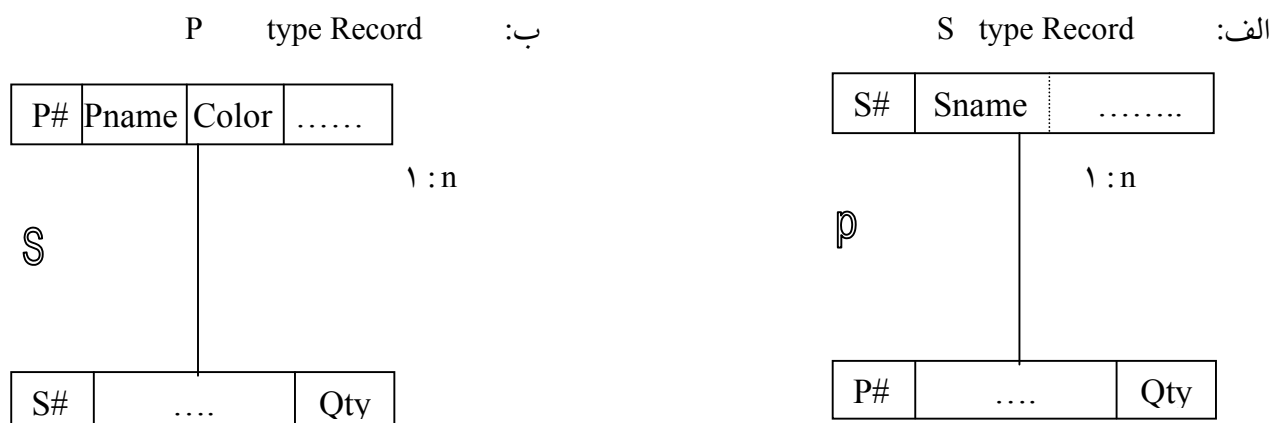
#### ۷-۳-۱- دید مفهومی (ادراکی)

دید طراح بانک از داده های ذخیره شده در آن می باشد. این دید دیدی جامع و سراسری می باشد، یعنی جامع تمام نیازهای کاربران است. این دید تشکیل دهنده سطح ادراکی و سطح ادراکی از سطوح انتزاعی پایگاه داده ها است. دید ادراکی باید به کمک امکاناتی نظیر احکام تعریف کننده از این زبان داده ای و ساختار داده ای تعریف شود که به تعریف آن شمای ادراکی گفته می شود. در واقع شمای ادراکی در معنای عام نوعی برنامه حاوی تعریف داده ها و ارتباطات بین آنها و نیز مجموعه ای از قواعد عملیاتی می باشد. این قواعد عملیاتی ناظر به داده های محیط عملیاتی هستند. از جمله ساختارهای داده ای رایج برای تعریف شمای ادراکی می توان به ساختارهایی نظیر رابطه ای، سلسله مراتبی، شبکه ای و هایپرگراف اشاره نمود.

مثال ۱) با توجه به رابطه بین قطعه و تهیه کننده به کمک احکام سلسله مراتبی دید ادراکی پایگاه را تعریف کنید.



DS سلسله مراتبی نوعی درختواره است که یک ریشه دارد و در سطوح مختلف دارای اعضا یا وابستگی می باشد به دو صورت آن را مدل می کنیم.



توجه: فیلد Qty در سلسله مراتب PS در S قرار دارد و در سلسله مراتب SP در P.

▪ نمایش دید ادراکی به کمک DS سلسله مراتبی : (شمای ادراکی)

از دید طراحی داده ها و ارتباطات بصورت یک درخت دیده می شود که ریشه آن حاوی اطلاعات در مورد قطعات و وابسته یا فرزند آن ریشه حاوی اطلاعاتی در مورد تهیه کنندگان. در ساختار داده سلسله مراتبی تعدادی درختواره وجود دارد که طراح این ساختار را به DBMS ای که آن را می پذیرد خواهد داد.

نمونه سازی از شمای ادراکی بصورت غیر صوری:

۱- نام سلسله مراتب PS است

۲- ریشه رکورد نوع P است و فیلدهای P ( P# کاراکتر. ۱ ، Pname کاراکتر. ۲ ... )

و شناسه ریشه P# است.

۳- وابسته یا فرزند رکورد نوع S می باشد.

فیلدهای S ( S# و Qty ..... ) است. و شناسه S S# می باشد.

مثال (۲) پایگاه رابطه ای:

در این ساختار برای طراحی پایگاه داده معمولاً برای هر موجودیت یک جدول در نظر گرفته می شود و برای هر صفت خاصه یک ستون و هر سطر که بعداً پر می شود یک نمونه موجودیت می باشد.

S				P			
S#	Sname	Status	City	P#	Pname	Color	Weight
S <sub>1</sub>	Sn <sub>1</sub>	۱۰	تهران	P <sub>1</sub>	Pn <sub>1</sub>	آبی	۱۰
S <sub>2</sub>	Sn <sub>2</sub>	۱۵	قزوین	P <sub>2</sub>	Pn <sub>2</sub>	زرد	۱۵
S <sub>3</sub>	SN <sub>3</sub>	۷	شیراز				

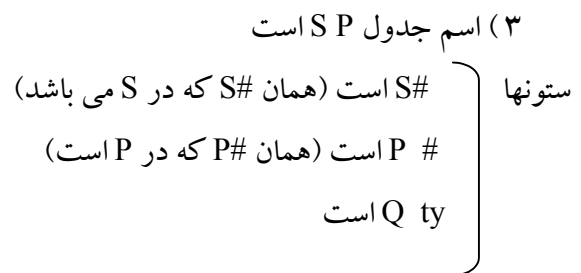
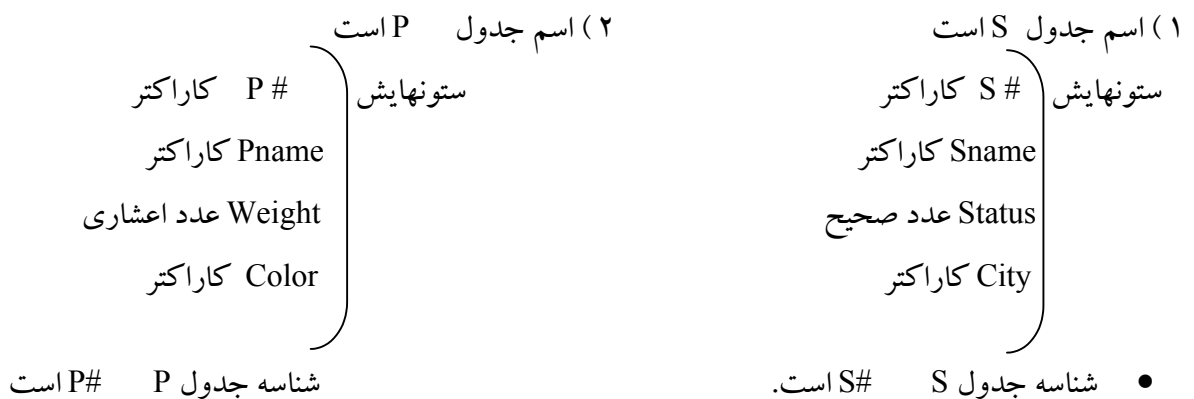
- برای نمایش ارتباطات بین دو یا بیش از دو موجودیت یک راه و البته رایج ترین است که جدولی دیگر طراحی می شود و ارتباطات به کمک آن نمایش داده می شود. در این جدول نشان دهنده ارتباط بین دو موجودیت شناسه موجودیتهای مرتبط آورده می شود.

- $P_1$  توسط  $S_1$  تهیه می شود

- $S_1$  را به تعداد ۱۰۰ تهیه می کند.

SP		
S#	P#	Qty
S <sub>1</sub>	P <sub>1</sub>	۱۰۰
S <sub>2</sub>	P <sub>2</sub>	۵۰
S <sub>3</sub>	P <sub>3</sub>	۵۵

- هر سطر در عین حال که نمایشگر یک موجودیت است در عین حال نمایشگر یک نمونه ارتباط نیز می باشد.
- ❖ شمای ساده پایگاه جدولی:



نکته: شناسه SP (S#, P#) با هم است.

### ۷-۳-۲- دید خارجی:

دید کاربر خاص نسبت به داده های ذخیره شده در پایگاه است در محدوده نیازهای اطلاعاتی مورد نظرش . دید خارجی در سطح خارجی معماری بانک مطرح و از سطوح انتزاعی است. دید خارجی مبتنی بر دید ادراکی است یعنی بر اساس دید ادراکی تعریف می شود. دید خارجی نیز برای معرفی شدن نیاز به یک ساختار یا مدل داده ای دارد که معمولاً همان مدلی است که در سطح ادراکی است. یعنی اگر طراح پایگاه را جدولی می بیند کاربران نیز بصورت جدولی می بینند. دید خارجی نیز باید به کمک احکامی تعریف شود که به تعریف آن شمای خارجی گویند.

u1

مثال:

Myv1		Myv3	
S#	STA	Sname	City
S1	۱۰	S2	قزوین
S2	۱۵		

نمونه ساده شده از شمای خارجی

۱- اسم دید myv1 است

۲- این دید روی جدول S تعریف می شود.

۳- ستونهایش S# و STA است.

۴- ستون S# از ستون S# مشتق می شود.

۵- ستون STA از ستون Status مشتق می شود.

۶- شرط یا شرایط دید C1 و C2، و ..... است.

نکته: عمدتاً View ها تعریف می شوند تا روی آنها پرس و جو (Quary) انجام شود و پرس و جو معمولاً بازیابی است.

### ۷-۳-۳- دید یا سطح داخلی:

این دید در سطح داخلی، پایین ترین سطح معماری بانک مطرح است. این سطح، سطحی واسط بین محیط فیزیکی پایگاه و سطوح انتزاعی آن می باشد که DBMS به مسائل و جنبه های مختلف فایلینگ می پردازد. البته تا حدی نظارت و دخالت DBA نیز در آن نقش دارد. که میزان دخالت و محدوده اختیارات DBA در سیستم های مختلف متفاوت است. دید داخلی به وسیله شمای داخلی توصیف می شود که نه تنها انواع مختلف رکوردهای ذخیره شده را تعریف می کند بلکه مشخص می کند چه شاخصهایی وجود دارد و فیلهای ذخیره شده چگونه نمایش داده می شوند.

### ۷-۳-۴- HL زبان میزبان:

یکی از زبانهای متعارف سطح بالاست. برای برنامه نویسان کاربردی این زبان می تواند یکی از زبانهای ++C، جاوا و یا زبان اختصاصی باشد که به زبانهای اختصاصی اغلب زبانهای نسل چهارم نیز گفته می شود، زبان میزبان مسئول تهیه و تدارک امکانات متعدد غیر بانک اطلاعاتی نظیر متغیرهای محلی، عملیات مفهومی و منطق تصمیم گیری و غیره می باشد.

### ۷-۳-۵- DSL زبان داده ای فرعی:

هر DBMS یک DSL دارد. DSL مجموعه احکامی است برای تعریف داده ها و کار با داده ها و کنترل آنها. هر زبان داده فرعی مشخص عملاً ترکیبی از احکام زیر است:

۱- احکام تعریف داده ها (DDL) Data Definition Language

۲- احکام تعریف کار با داده ها (DML) Data Manipulation Language

۳- احکام کنترل کار با داده ها (DCL) Data Control Language



هر یک از این سه دسته احکام باید برای سطوح سه گانه پایگاه نیز وجود داشته باشد. DSL ها را از نظر نیاز یا عدم نیاز به زبان میزبان به دو دسته مستقل و ادغام شدنی تقسیم می کنند.

زبان فرعی داده ای مستقل، زبانی است که به زبان میزبان نیاز ندارد و زبان فرعی داده ای ادغام شده، زبانی است که همراه زبان HL استفاده می شود. به بیان دیگر احکام آن باید به نحوی در احکام زبان برنامه سازی ادغام شوند. مکانیزم ادغام در سیستم های مختلف متفاوت و بطور کلی به دو صورت ادغام صریح و ادغام ضمنی وجود دارد. در ادغام ضمنی، احکام زبان داده ای بطور صریح در متن زبان میزبان جای داده نمی شوند بلکه از طریق حکم فراخوانی بکار برده می شوند.

#### برخی نکات مهم در مورد DSL

- هر DBMS دارای یک DSL است.
- هر DSL در کادر مفاهیم یک مدل داده ای مشخص طراحی می شود و عملگرهای آن نیز در کادر همان مفاهیم عمل می کنند.
- اصل وحدت احکام در آن رعایت شده باشد. مثلاً برای انجام عمل درج که منطقیاً یک حکم واحد داشته باشد و ترجیحاً همان حکم واحد در سطح خارجی و هم در سطح ادراکی عمل نماید.

#### **۷-۳-۶- نگاشت: Mapping**

علاوه بر سه سطح از معماری، معماری پایگاه از چند نگاشت (تبدیل) مختلف تشکیل می شود.

۷-۳-۶-۱- نگاشت مفهومی / داخلی:

تناظر بین دید ادراکی و بانک اطلاعاتی ذخیره شده را تعریف و مشخص می کند که چگونه رکوردهای ادراکی و فیلدها در سطح داخلی نمایش داده شوند.

۷-۳-۶-۲- نگاشت خارجی / ادراکی:

تناظر بین دید خارجی خاص و دید مفهومی را تعریف می کند. در واقع مکانیسمی است برای برقراری تناظر بین دیده های خارجی مختلف و دید واحد ادراکی. DBMS های متعارف حداقل دو محور تبدیل دارند: تبدیل داده و تبدیل احکام.

- تبدیل داده ها یعنی تبدیل داده های تعریف شده در سطح خارجی به داده های تعریف شده در سطح ادراکی و بالاخره به داده های تعریف شده در سطح داخلی.
- تبدیل احکام یعنی تبدیل حکم عمل کننده در سطح خارجی به حکم عمل کننده در سطح ادراکی و در نهایت به حکم یا احکامی در سطح داخلی. این تبدیل از جمله وظایف مهم هر سیستم مدیریت بانک اطلاعاتی است.

### ۷-۳-۷- سیستم مدیریت بانک اطلاعاتی:

سیستم مدیریت بانک اطلاعاتی نرم افزاری است که مدیریت بانک اطلاعاتی را عهده دار است و مجموعه ای است از برنامه ها که واسط بین کاربران و محیط فیزیکی ذخیره و بازیابی می باشند. این نرم افزار واسط به کاربران امکان می دهد تا داده های خود را تعریف کنند و به داده های خود دسترسی داشته و با آنها کار کنند.

#### • اجزاء تشکیل دهنده DBMS:

##### الف) بخش هسته ای Kernel شامل:

- ۱- پیش کامپایلر
- ۲- پردازشگر پرس و جو
- ۳- بهینه ساز پرس و جو
- ۴- مدیریت فایلها (برای سطح داخلی)
- ۵- واحد دریافت درخواست کاربر و انجام مقدمات کار
- ۶- واحد لود پایگاه داده ها

##### ب) بخش مدیریتی محیط پایگاه داده ها:

- ۱- واحد کنترل همزمانی عملیات
- ۲- واحد کنترل جامعیت پایگاه
- ۳- واحد کنترل ایمنی پایگاه
- ۴- واحد کنترل ترمیم پایگاه
- ۵- واحد تولید نسخه های پشتیبان

##### ج) بخش امکانات جانبی:

- ۱- امکانات پرس و جو به کمک مثال و به کمک فرم
- ۲- روالهای مخصوص تجزیه و تحلیل آماری که معمولاً مورد استفاده DBA هستند
- ۳- ابزارهای ایجاد برنامه های کاربردی
- ۴- نرم افزارهای مخصوص محیط شبکه ای
- ۵- امکانات گرافیکی
- ۶- امکانات دسترسی به داده های دور .

اینکه سیستم مدیریت بانک چگونه درخواستهای کاربران را عملی می سازد بستگی به نوع آن دارد. بطور خلاصه نحوه اجرای درخواست کاربر بصورت زیر می باشد:

- ۱- دریافت درخواست کاربر و انجام بررسی های اولیه (معتبر بودن کاربر)
- ۲- بررسی و تحلیل درخواست کاربر
- ۳- بررسی شمای خارجی کاربر برای مشخص شدن محدوده دید کاربر از پایگاه

۴- بررسی شمای ادراکی برای تعیین نحوه نگاشت عملیات سطح خارجی به ادراکی

۵- انجام تبدیلات لازم

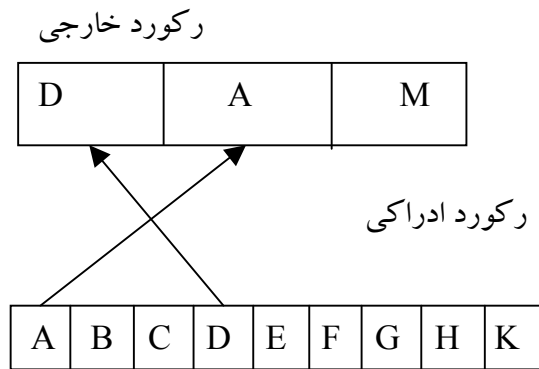
۶- بررسی شمای داخلی و تبدیل احکام سطح ادراکی به سطح داخلی

۷- دستیابی به فایل‌های فیزیکی و اجرای درخواست کاربر

• اصطلاح به عینیت در آوردن (materialized)

اگر درخواست کاربر بازیابی باشد اصطلاحاً گویند DBMS داده های مورد نظر را به عینیت در می آورد که به دو فرم وجود دارد:

۱- به عینیت در آوردن مستقیم: موقعیتی که داده مورد نظر کاربر آنچه در رکورد خارجی خواسته متناظر زیرین داشته باشد یعنی مشخصاً در سطح ادراکی متناظر داشته باشد.



۲- غیر مستقیم:

موقعیتی که داده مورد نظر کاربر فیلد یا فیلدهای متناظر زیرین نداشته باشد بلکه حاصل پردازش باشد مثال: فیلد میانگین مقادیر یک فیلد (فیلد های مجازی)

۷-۳-۸- مدیر بانک اطلاعاتی DBA :

فردی است با تخصص بالا در تکنولوژی بانکهای اطلاعاتی و دانش و فن کامپیوتر. این فرد معمولاً در پروژه های بزرگ تیمی از افراد متخصص در اختیار دارد و وظیفه کلی تیم طراحی، ایجاد، پیاده سازی، نگهداری و گسترش و اداره بانک برای یک محیط عملیاتی است. امروز به سبب اهمیت بسیار بالای داده در سازمانها، داده های یک سازمان نیاز به اداره کننده دارند یعنی فردی با سمت اداره کننده داده ها یا به اختصار (DA). این فرد که لزوماً نباید متخصص در کامپیوتر و یا بانک اطلاعاتی باشد مدیریت کل داده های سازمان را بر عهده دارد و با هماهنگی با مدیریت سازمان، خط مشی ها و تصمیماتی در مورد داده های مؤسسه خود را اتخاذ می کند.

در مدیریت سازمان وقتی از سرمایه سازمان بحث می کنند می گویند از پنج بخش تشکیل می شود.

نرم افزار، سخت افزار، نیروی متخصص، بودجه و داده.

و اما وظایف اداره کننده بانک در طیفی از وظایف مدیریتی تا وظایف فنی و علمی جای دارد. در واقع می توان DBA را بعنوان تیمی تخصصی به سرپرستی DBA (مدیر بانک اطلاعاتی) در نظر گرفت که با متخصص هایی نظیر DA،

مسئول مستقیم تیم های برنامه سازی، مدیر کنترل کننده عملکرد خود سیستم و ... در خصوص بانک اطلاعاتی همکاری دارند.

بطور کلی می توان وظایف DBA را به شرح زیر بیان نمود:

- ۱- همکاری با DA در تفهیم اهمیت و نقش داده سازمان.
- ۲- همکاری در معرفی تکنولوژی بانک اطلاعاتی و جنبه های مختلف ارجحیت آن بر تکنولوژی غیر بانکی
- ۳- تلاش در مجاب کردن سازمان در استفاده از تکنولوژی کارا تر
- ۴- مطالعه دقیق محیط عملیاتی و تشخیص نیازهای کاربران مختلف
- ۵- بازشناسی موجودیت ها و ارتباطات بین آنها و تعیین صفات خاصه هر یک از انواع موجودیتها
- ۶- رسم نمودار E-R
- ۷- تخمین حجم اطلاعات ذخیره شدنی در بانک
- ۸- مشارکت در تعیین سیستم مدیریت بانک اطلاعاتی با توجه به امکانات کامپیوتری و محیط
- ۹- مشاوره و مشارکت در تعیین سیستم کامپیوتری و پیکربندی آن از لحاظ ملزومات سخت افزاری و نرم افزاری سیستم عامل
- ۱۰- طراحی سطح ادراکی بانک و نوشتن شمای ادراکی
- ۱۱- ایجاد پایگاه با داده های تستی
- ۱۲- تعریف دیدهای خارجی کاربران برنامه ساز
- ۱۳- نظارت در نوشتن شمای خارجی
- ۱۴- نظارت در جمع اوری داده ها و ورود داده ها
- ۱۵- تست پایگاه با داده های واقعی (آغاز پیاده سازی پایگاه)
- ۱۶- تعیین ضوابط دستیابی به بانک برای کاربران با توجه به نیازهای اطلاعاتی آنها
- ۱۷- نظارت و دخالت در تهیه مستندات سیستم
- ۱۸- تأمین جامعیت بانک از طریق حفظ کیفیت، کنترل دستیابی و حفاظت از محرمانگی محتوای بانک
- ۱۹- کمک به کاربران و آموزش آنان در برنامه ریزی در دستیابی به داده ها و کار با آنها
- ۲۰- حفظ ایمنی بانک
- ۲۱- پیش بینی روشهای ترمیم و استراتژی لازم برای پشتیبانی
- ۲۲- معاصر نگه داشتن پایگاه با پیشرفتهای تکنولوژیک
- ۲۳- تلاش در جهت ارتقاء سطح تخصصی افراد و کاربران
- ۲۴- نظارت به کارایی و پاسخ به تغییر نیازها

• دیکشنری داده ها (کاتالوگ سیستم)

یک DBMS برای انتخاب نحوه اجرای عملیات روی بانک اطلاعاتی تحت کنترل خود اطلاعاتی را از آن بانک تحت عنوان کاتالوگ سیستم نگهداری می کند. در واقع کاتالوگ جایی است که تمام شماهای مختلف (خارجی، مفهومی و داخلی) و تمام نگاهشتهای متناظر با آنها در آن نگهداری می شوند. به بیان دیگر کاتالوگ شامل اطلاعات تفصیلی (که گاه فرا داده نامیده می شوند) مربوط به اشیاء متعددی است که در خود سیستم قرار دارند.

بطور کلی اطلاعات زیر در آن نگهداری می شود:

- ◀ نام ساختارهای داده ای در چارچوب مدل داده ای مشخص مثلاً نام جداول در بانک رابطه ای
- ◀ نام موجودیتها و ارتباطات بین آنها
- ◀ نام صفات خاصه هر موجودیت، نوع و طیف مقادیر
- ◀ شماهای خارجی کاربران
- ◀ شماهای ادراکی
- ◀ مشخصات فنی کاربران و چگونگی حق دستیابی آنها به داده ها و محدوده عملیات مجاز آنها
- ◀ رویه های تبدیل بین سطوح مختلف
- ◀ تاریخ ایجاد داده ها

۷-۴- دلایل استفاده از بانک اطلاعاتی: Why Database ?

۷-۴-۱- مزایای بانک اطلاعاتی چند کاربری :

- ۱- امکان مدلسازی داده های عملیاتی بر اساس سمانتیک آنها
  - ۲- وحدت ذخیره سازی کل داده های محیط عملیاتی
- وجود سطح ادراکی در معماری پایگاه داده ها امکان می دهد تا کل داده های عملیاتی یکبار آنگونه که طراح می بیند تعریف و ذخیره شوند. این وحدت ذخیره سازی در عین تعدد نشاندهنده دیدهای کاربران است.
- ۳- اشتراکی شدن داده ها
  - امکان استفاده کاربران از داده واحد ذخیره شده بصورت اشتراکی
  - ۴- کاهش میزان افزونگی
  - ۵- تضمین جامعیت داده ها
  - ۶- امکان اعمال ضوابط دقیق ایمنی
- سیستم مدیریت بانک اطلاعاتی با اعمال کنترل متمرکز از هرگونه اقدام برای دستیابی غیر مجاز به داده ها جلوگیری می کند
- ۷- امکان ترمیم داده ها

سیستم مدیریت بانک اطلاعاتی با مکانیسم هایی خسارت ناشی از بروز نقص ها و اشتباهات را جبران کرده و داده - های ذخیره شده را ترمیم می کند بنحوی که محتوای بانک وضعیت صحیح خود را باز یابد.

۸- تأمین استقلال داده ها

هم دلیل این تکنولوژی و هم هدف آن می باشد

### تعریف استقلال داده ای:

مصونیت دیدهای کاربران و برنامه های کاربردی در قبال تغییراتی که در سطوح معماری پایگاه پدید می آیند.

استقلال داده ای دو وجه دارد: استقلال داده ای منطقی و فیزیکی

استقلال داده ای فیزیکی یعنی مصونیت دیدهای کاربران و برنامه های کاربردی در قبال تغییراتی که در سطح داخلی پایگاه پدید می آیند، در DBMS های واقعی استقلال داده ای فیزیکی تقریباً صد در صد است زیرا با توجه به معماری چند سطحی پایگاه کاربران در سطح خارجی در یک محیط انتزاعی و منفک از فایلینگ عمل می کنند.

استقلال داده ای منطقی:

یعنی مصونیت برنامه ای کاربردی و دید کاربران در قبال تغییراتی که در سطح ادراکی پدید می آیند. تغییرات سطح ادراکی از دو جنبه پدید می آیند:

۱- از رشد پایگاه در سطح ادراکی

۲- در سازماندهی مجدد سطح ادراکی

### دلایل رشد پایگاه:

- مطرح شدن نیازهای جدید برای کاربران
- مطرح شدن کار برانی جدید با نیازهای اطلاعاتی جدید

### دلایل سازماندهی مجدد:

- تأمین محیط ذخیره سازی کاراتر برای بخشی از پایگاه
- تأمین ایمنی بیشتر برای پایگاه
- تأمین کارایی عملیاتی بیشتر برای DBMS از طریق کاهش آنومالی ها

۹- تسریع در دریافت پاسخ پرس و جوها

۱۰- تسهیل در دریافت گزارشهای متنوع آماری

۱۱- امکان اعمال استانداردها

با کنترل متمرکز روی بانک اطلاعاتی، DBA می تواند اطمینان دهد که تمام استانداردهای مطلوب در نمایش داده ها مورد توجه قرار گرفته است. استانداردهای مطلوب ممکن است حاوی یک یا تمام موارد زیر باشند: استانداردهای بخش، تأسیسات، شرکت، صنعت، ملی و بین المللی.

۱۲- استفاده بهتر از سخت افزار

## ۷-۴-۲- معایب بانک اطلاعاتی (چند کاربری)

- ۱- هزینه بالای نرم افزار
- ۲- هزینه بالای سخت افزار
- ۳- هزینه بیشتر برای برنامه سازی
- ۴- هزینه بالا برای انجام مهندسی مجدد به منظور تبدیل سیستم از مشی فایل پردازی به مشی پایگاهی
- ۵- پیچیده بودن سیستم و نیاز به تخصص بیشتر

## ۷-۵- معماری سیستم پایگاه داده ها

منظور از معماری سیستم پایگاه داده ها ، نحوه پیکربندی اجزای سیستمی است که در آن حداقل یک پایگاه داده ، یک سیستم مدیریت پایگاه داده ، یک سیستم عامل ، یک کامپیوتر با دستگاههای جانبی و تعدادی کاربر وجود دارد و خدمات پایگاهی به کاربران ارائه میکنند. این معماری بستگی به دو عنصر اصلی یعنی سخت افزار و نرم افزار مدیریت DBMS دارد. بطور کلی چهار معماری برای سیستم پایگاه داده ها وجود دارند :

- ۱- معماری متمرکز
- ۲- معماری سرویس دهنده - سرویس گیرنده
- ۳- معماری توزیع شده
- ۴- معماری با پردازش موازی

## ۷-۵-۱- معماری متمرکز Centralized

در این معماری یک پایگاه داده ها روی یک سیستم کامپیوتری و بدون ارتباط با سیستم کامپیوتری دیگر ایجاد میشود. سخت افزار این سیستم می تواند کامپیوتر شخصی ، متوسط و یا بزرگ باشد. سیستمی که بر روی کامپیوتر شخصی ایجاد می شود ، بیشتر برای کاربردهای کوچک و با امکانات محدود است .

## ۷-۵-۲- معماری سرویس دهنده / سرویس گیرنده CLIENT/SERVER

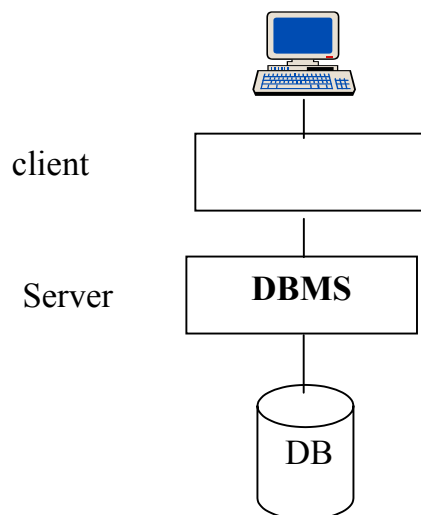
هر معماری که در آن قسمتی از پردازش را یک برنامه ، سیستم و یا ماشین انجام دهد و انجام قسمت دیگری از پردازش را از برنامه ، سیستم و یا ماشین دیگر بخواهد معماری سرویس دهنده و سرویس گیرنده نامیده می شود. در واقع ، این معماری یک سیستم بانک اطلاعاتی را بصورت یک ساختار دو قسمتی در نظر می گیرد : سرویس دهنده و سرویس گیرنده. تمام داده ها در بخش سرویس دهنده ذخیره شده و تمام برنامه های کاربردی در بخش سرویس گیرنده قرار میگیرند.

### الف- سرویس دهنده :

یک سیستم بانک اطلاعاتی است که از تمام عملیات اصلی یک DBMS پشتیبانی میکند. در واقع میتوان نام دیگری برای DBMS را سرویس دهنده نیز بطور ساده در نظر گرفت.

### ب- سرویس گیرنده :

سرویس گیرنده ها برنامه های کاربردی مختلفی هستند که بر روی DBMS قرار دارند. از قبیل : برنامه های کاربردی نوشته شده توسط کاربران و برنامه های کاربردی تعبیه شده در سیستم.  
شکل ساده شده ای از معماری سرویس دهنده / سرویس گیرنده را در زیر می بینید :



معماری سرویس دهنده ، سرویس گیرنده بصورت های چند سرویس گیرنده - یک سرویس دهنده ، یک سرویس گیرنده - چند سرویس دهنده و چند سرویس گیرنده - چند سرویس دهنده نیز مطرح است.

#### مزایای این معماری نسبت به معماری متمرکز :

- تقسیم پردازش
- کاهش ترافیک شبکه در معماری حول شبکه
- استقلال ایستگاه های کاری
- اشتراک داده ها

#### ۷-۵-۳ معماری توزیع شده : Distributed

این معماری حاصل ترکیب دو تکنولوژی است : تکنولوژی پایگاه داده ها و تکنولوژی شبکه . در یک نگاه کلی میتوان پایگاه داده توزیع شده را مجموعه ای از داده های ذخیره شده که از نظر منطقی متعلق به یک سیستم می باشند و روی سایت های مختلف یک یا بیش از یک شبکه توزیع گردیده اند ، در نظر گرفت.  
برخی ویژگیهای این معماری عبارتند از :

- مجموعه ای است از داده های مرتبط و مشترک
- داده ها به بخشهای تقسیم و در سایت ها توزیع شده اند.
- بعضی از بخشها ممکن است در چند نسخه (بطور تکراری) در سایتها ذخیره شوند.
- سایتها از طریق یک شبه با هم ارتباط دارند.
- داده های ذخیره شده در هر سایت تحت کنترل یک DBMS می باشند.



مهمترین اصل در سیستم پایگاهی توزیع شده این است که سیستم باید چنان عمل کند که از نظر کاربران مشابه با یک پایگاه داده متمرکز باشد. این ویژگی تحت عنوان شفافیت در سیستمهای توزیع شده مطرح است و تفاوت سیستمهای توزیعی و سرویس گیرنده - سرویس دهنده نیز در همین است.

### ۷-۵-۳-۱- مزایای این معماری

- سازگاری و هماهنگی با ماهیت سازمانهای نوین
- کارایی بیشتر در پردازش داده ها بویژه در سیستمهای بسیار بزرگ
- دستیابی بهتر به داده ها

### ۳-۵-۳-۲- معایب

- پیچیدگی طراحی سیستم
- پیچیدگی پیاده سازی
- هزینه بیشتر

### ۷-۵-۴- معماری با پردازش موازی: Parallel

این معماری با ساخت و گسترش ماشینهای موازی ، برای ایجاد پایگاه دادهای بسیار بزرگ مورد توجه قرار گرفت . این معماری گسترش یافته معماری توزیع شده است و برای تامین کارایی و دستیابی سریع طراحی میشود. برای ایجاد پایگاه داد ها با معماری پردازش موازی ، بطور کلی چندین طرح از این معماری وجود دارد که مطالعه آنها خارج از مطالب این درس است:

- معماری با حافظه مشترک
- معماری با دیسک مشترک
- معماری سلسله مراتبی

### ۷-۶- سیستم پایگاه داده های همراه Mobile Database System

با رشد سریع تکنولوژی ارتباطات ، نوع جدیدی از سیستم پایگاه داده ها پدید آمده و در حال گسترش است که سیستمهای پایگاه داده همراه نامیده میشوند. در این معماری ، یک یا بیش از یک کامپیوتر متوسط یا بزرگ نقش سرویس دهنده را ایفا میکند. هر کاربر کامپیوتر کوچک همراه خود را دارد که در آن داده های عملیاتی و برنامه های کاربردی مورد نیازش ذخیره شده اند. کاربر میتواند از هر جایی با سیستم سرویس دهنده مورد نظرش مرتبط بوده و پردازش های مورد نظرش را انجام دهد.

### موضوعات تحقیقاتی:

- ۱- به نظر شما چه عواملی در انتخاب یک DBMS نقش دارند ؟
- ۲- یک DBMS رابطه ای را در نظر گرفته و اجزای آن را بررسی نمایید.

۳- ODBC

۴- JDBC

تمرینات این فصل:

- ۱) سیستم پایگاهی چه مزایایی نسبت به سیستم ناپایگاهی دارد؟
- ۲) نقش پایگاه داده ها در مدیریت و فعالیت یک سازمان چیست؟
- ۳) به نظر شما معایب تکنولوژی پایگاه داده ها چیست؟
- ۴) چرا معماری پایگاه داده ها باید در چند سطح طراحی می شود؟
- ۵) چگونه در سیستم پایگاه داده ها، "وحدت ذخیره سازی" در عین تعدد دیدهای کاربران، تامین می شود؟
- ۶) سطوح انتزاعی در معماری پایگاه داده ها یعنی چه؟ چگونه تامین می شود؟
- ۷) سطح خارجی معماری ANSI (مفهوم دید) چه مزایا و چه معایبی دارد؟
- ۸) مراحل طراحی و پیاده سازی پایگاه داده ها را شرح دهید.
- ۹) در طراحی پایگاه فیزیکی چه عواملی را باید در نظر گرفت؟
- ۱۰) کدامیک از فازهای طراحی پایگاه داده ها مستقل از DBMS انجام می شود؟
- ۱۱) استقلال داده ای چیست؟ تامین استقلال داده ای فیزیکی آسان تر است یا استقلال داده ای منطقی؟
- ۱۲) تغییرات در محیط فایلینگ پایگاه کدامند؟
- ۱۳) وظایف DBA چیست؟
- ۱۴) DBMS چگونه به درخواست کاربر پاسخ می دهد؟ (توضیح با رعایت ترتیب عملیات)
- ۱۵) متاداده (Meta Data) چیست؟ حاوی چه اطلاعاتی است؟ به چه کار می آید؟
- ۱۶) با یک طرح، دو بخش اصلی DBMS و محیط پایگاه داده ها را نشان دهید و اجزای تشکیل دهنده DBMS را نام ببرید.
- ۱۷) DSL چیست و چه خصوصیات و جنبه هایی باید داشته باشد؟
- ۱۸) رویه ای و DSL نارویه ای چیست؟
- ۱۹) یکی از مزایای تکنولوژی پایگاه داده ها، ایجاد سازش بین نیازهای گاهی متضاد کاربران است. این تضادها از چه نظرهایی مطرح هستند؟
- ۲۰) به نظر شما تحت چه شرایطی می توان (و نه لزوماً باید!) از استفاده از تکنولوژی پایگاه داده ها صرف نظر کرد؟ در این صورت چگونه محیط ذخیره و بازیابی اطلاعات را باید ایجاد کرد؟
- ۲۱) در انتخاب DBMS چه هزینه هایی را باید در نظر گرفت؟
- ۲۲) چه جنبه هایی از سطح داخلی را باید در انتخاب DBMS در نظر گرفت؟

## فصل هشتم : ترمیم

### ۸-۱- تعریف ترمیم

ترمیم : یعنی امکان تشخیص اشکالات احتمالی ایجاد شده در حین اجرای برنامه و بازسازی بانک اطلاعاتی که احتمالاً در اثر بروز اشکال در سیستم ناقص شده است. برای تشخیص اشکال و ترمیم بانک دو کار بایستی انجام شود:  
الف : اقدامات قبل از بروز مشکل

وقتی برنامه اجرای عادی خود را انجام می دهد باید DBMS یکسری اطلاعات را ذخیره کند تا اگر مشکلی پیش آمد بتواند ترمیم نماید . در این مرحله بعنوان پیش بینی بروز مشکل ، در حین اجرای برنامه اقدام به ذخیره سازی اطلاعاتی می شود تا در صورت بروز مشکل بتوان با استفاده از آنها اثر نامطلوب روی اطلاعات را خنثی نمود.  
ب : اقدامات بعد از بروز اشکال

شامل اقداماتی است که DBMS با استفاده از اطلاعات ذخیره شده در مرحله قبل انجام می دهد تا ناهماهنگی و ناسازگاری ایجاد شده در بانک در اثر بروز اشکال را رفع نماید.

### ۸-۲- مدیریت ترمیم:

یکی از قسمتهای DBMS مدیریت ترمیم می باشد که مسئولیت حفظ یکپارچگی اطلاعات در حین اجرای برنامه به عهده آن است . این مسئولیت بصورت زیر انجام می شود:

الف: تشخیص اشکال : اگر در حین اجرای برنامه مشکلی در سیستم رخ داده باشد تشخیص داده می شود.  
ب: حفظ یکپارچگی : اگر برنامه نیمه کاره رها شده باشد باعث عدم یکپارچگی اطلاعات شده و آن را رفع می کند.

### ۸-۲-۱- دسته بندی خطاها :

۱- خطاهای تراکنش ها

الف) خطاهای منطقی: در صورت کامل نشدن تراکنش

ب) خطاهای سیستمی : بانک اطلاعاتی ممکن است یک تراکنش در حال اجرا را متوقف کند که این امر ممکن است در اثر خطا و یا وجود بن بست صورت گیرد.

۲- خطای خرابی سیستم :

خطای سخت افزار و یا خطای نرم افزاری باشد که باعث خرابی سیستم می گردد.

۳- خطای دیسک:

خرابی هد دیسک و یا خطاهای مشابه که باعث از بین رفتن بخشی از یک دیسک می گردند.

### ۸-۳- ترمیم با استفاده از ثبت وقایع (Log-Based Recovery)

رایجترین شیوه رفع اشکال ، ثبت وقایع می باشد که در حین اجرای عادی تراکنش یکسری اطلاعات بنام log که مبین تغییرات اعمال شده در اطلاعات است ذخیره شده تا در صورت لزوم از آنها برای ترمیم اطلاعات استفاده گردد. در حین اجرای هر تراکنش اطلاعات زیر در log ثبت می گردند:

-ثبت شروع تراکنش : وقتی که تراکنش Ti شروع می شود اطلاعات  $\langle Ti, Start \rangle$  بر روی رکورد log ثبت می گردد.

-ثبت عملیات تراکنش که در ازای هر عمل Write انجام می گیرد . اطلاعات  $\langle Ti, Xi, V1, V2 \rangle$  در log ثبت می شود که V1 مقدار  $X_j$  قبل از عمل Write و V2 مقدار نوشته شده در X می باشد. (بعد از عمل پردازش)

-ثبت پایان تراکنش : وقتی که تراکنش پایان یافت رکورد  $\langle Ti, Commit \rangle$  در log نوشته می شود.

### برخی نکات مهم :

#### ۱- زمان ثبت رکورد log:

قبل از اجرای هر دستور write در یک تراکنش ، رکورد شرح وقایع آن باید ایجاد شده و در حافظه ثانویه ذخیره شده باشد تا در صورت لزوم بتوان از آن برای اعمال ترمیم اطلاعات استفاده کرد.

#### ۲- نحوه استفاده از log:

در صورت بروز مشکل در اجرای تراکنش پس از راه اندازی مجدد سیستم اطلاعات log را بررسی کرده و در صورت لزوم اقدام به ترمیم اطلاعات می کند.

#### ۳- دستورات مورد استفاده در ترمیم:

در شیوه log دو دستور زیر برای ترمیم بانک اطلاعاتی استفاده می شوند:

الف) دستور Redo

با اجرای این دستور ، عملیات اجرا شده توسط تراکنش برای حصول یکپارچگی بانک اطلاعاتی با استفاده از اطلاعات ذخیره شده در log دوباره اجرا می شوند. بایستی توجه داشت با اجرای دستور فوق عملیات تراکنش ممکن است روی بعضی از اطلاعات دوبار و بعضی دیگر یکبار اعمال شوند. در اینصورت اجرای دوباره تراکنش نباید باعث عدم یکپارچگی اطلاعات شود که با استفاده از رکورد ثبت وقایع انجام می شود.

ب) دستور Undo

با اجرای این دستور عملیات اجرا شده توسط تراکنش برای حصول یکپارچگی بانک اطلاعاتی با استفاده از اطلاعات ذخیره شده در log خنثی می گردند.

#### ۸-۴- متدهای ترمیم بکار رفته در روش log-based (ثبت وقایع)

می دانیم بحث ترمیم موقعی مطرح است که حداقل یک عمل نوشتن داشته باشیم . در رابطه با اعمال نتیجه یک تراکنش دو شیوه مطرح می شود:

۱- تاخیر اعمال تغییرات.

۲- اعمال آنی تاثیرات.

### ۸-۴-۱- روش تاخیر اعمال تاخیرات Deferred Database Modification

در این روش تمامی اعمال نوشتن تا انتهای تراکنش به تاخیر می افتند. عبارتی اگر در یک تراکنش دو دستور write(A) و write(B) بصورت زیر داشته باشیم:

```
read(A)
write(A)
.read(B)
..
write(B)
```

دستور write(A) و write(B) را در آخر تراکنش انجام می دهد.

اطلاعاتی که در این روش بایستی در log ثبت گردد بصورت زیر می باشد:

-تراکنش با نوشتن دستور  $\langle Ti, Start \rangle$  در log آغاز می شود.

با توجه به عملکرد مکانیزم ترمیم نیازی به نگهداری  $V1$  در write(x) نمی باشد (چون عمل Undo نداریم) لذا اطلاعات  $\langle Ti, X, V \rangle$  در log ثبت می شود که  $V$  مقدار جدید  $X$  است. عمل نوشتن در این لحظه صورت نمی پذیرد بلکه به تاخیر می افتد.

-قبل از اینکه اولین write اجرا شود رکورد  $\langle Ti, Commit \rangle$  را در log ثبت می کند. (commit سیستم بعد از آخرین write است اما دستور commit ای که در log ثبت می شود قبل از اولین write می باشد).

#### • روش ترمیم:

با توجه به اینکه در چه مرحله ای از تراکنش سیستم دچار مشکل شده است ترمیم بصورت زیر انجام می شود:

#### الف) قبل از پایان عملیات تراکنش:

در اینصورت اطلاعات درون log بصورت زیر می باشد:

```
<Ti,Start>
<Ti,X1,V1>
.
<Ti,Xi,Vi>
```

لذا از عملیات انجام شده صرف نظر می گردد. زیرا اثر اجرای تراکنش قبل از خرابی سیستم روی پایگاه داده منعکس نشده است لذا اجرای آن نادیده گرفته می شود.

#### ب) پس از پایان عملیات تراکنش :

در اینصورت اطلاعات log بصورت زیر است:

```
<Ti,Start>
<Ti,X1,V2>
<Ti,Xn,Vn2>
<Ti,Commit>
```

لذا چون عملیات تراکنش قبل از خرابی سیستم پایان یافته است و از طرفی ممکن است ثبت تمامی آنها انجام نشده است لذا اجرای دوباره تراکنش صورت می پذیرد یعنی دستور  $\langle redo, Ti \rangle$  انجام می شود.  
 مثال: فرض کنید تراکنش های  $T_0$  و  $T_1$  را داریم و  $T_0$  قبل از  $T_1$  اجرا می شود:

**$T_0$  :Read(A)**  
**A:=A-50000**  
**write(A)**  
**Read(B)**  
**B:=B+50000**  
**write(B)**

**$T_1$  :Read(C)**  
**C:=C-100**  
**write(C)**

با توجه به اطلاعات موجود در LOG خواهیم داشت:

$\langle T_0, Start \rangle$  (الف)

$\langle T_0, A, 950000 \rangle$   
 $\langle T_0, B, 2050000 \rangle$

در این حالت هیچ عملی صورت نمی گیرد.

$\langle T_0, Start \rangle$  (ب)

$\langle T_0, A, 950000 \rangle$   
 $\langle T_0, B, 2050000 \rangle$   
 $\langle T_0, Commit \rangle$   
 $\langle T_1, Start \rangle$   
 $\langle T_1, C, 600 \rangle$

در این حالت Redo( $T_0$ )

$\langle T_0, Start \rangle$  (ج)

$\langle T_0, A, 950000 \rangle$   
 $\langle T_0, B, 2050000 \rangle$   
 $\langle T_0, Commit \rangle$   
 $\langle T_1, Start \rangle$   
 $\langle T_1, C, 600 \rangle$   
 $\langle T_1, C, 500 \rangle$   
 $\langle T_1, Commit \rangle$

در این حالت Redo( $T_0$ ) و متعاقبا " Redo( $T_1$ ) انجام می گیرد.

### ۸-۴-۲- شیوه اعمال آنی تغییرات (Immediate database Modification)

در این روش حاصل هر عملیات write روی بانک اطلاعاتی سریعا " به بانک منعکس می شود بعبارت دیگر هر دستور write در محل ظهور خود اجرا می شود. از آنجائیکه در اینجا عمل undo نیز داریم لذا مقدار قبلی و مقدار جدید هر متغیر نیاز است.

اطلاعاتی که در log ثبت می گردند بترتیب زیر می باشد:

۱- ثبت  $\langle Ti, Start \rangle$  قبل از شروع تراکنش این دستور در log ثبت می شود.

۲- ثبت  $\langle Ti, X, V_1, V_2 \rangle$  قبل از اجرای هر write

۳-ثبت  $\langle Ti, Commit \rangle$  پس از اجرای آخرین دستور تراکنش

شیوه ترمیم :

با توجه به اطلاعات موجود در log به یکی از حالات زیر عمل می شود.

**الف) حذف اثر تراکنش Undo(Ti)**

این حالت موقعی اجرا می شود که احتمالاً "بخشی از پایگاه اصلاح شده و اطلاعات موجود در log برای اعمال بقیه اصلاحات نیز کافی نباشد. عبارت دیگر موقعی که دستور  $\langle Ti, Start \rangle$  در log باشد ولی  $\langle Ti, Commit \rangle$  نباشد.

**ب) اجرای دوباره تراکنش redo(Ti)**

این حالت موقعی اجرا می شود که احتمالاً "بخشی از بانک اطلاعاتی اصلاح شده و اطلاعات موجود در log برای اعمال بقیه اصلاحات نیز کافی است. عبارت دیگر در صورتی صادر می شود که  $\langle Ti, Start \rangle$  و  $\langle Ti, Commit \rangle$  در log باشد.

• **مثال از حالت اعمال آنی تغییرات:**

مثال قبل را در نظر بگیرید فرض کنید در log در سه زمان مختلف داده های زیر قرار گرفته باشند.

$\langle T0, Start \rangle$	$\langle T0, Start \rangle$	$\langle T0, Start \rangle$
$\langle T0, A, 1000000, 950000 \rangle$	$\langle T0, A, 1000000, 950000 \rangle$	$\langle T0, A, 1000000, 950000 \rangle$
$\langle T0, B, 2000000, 2050000 \rangle$	$\langle T0, B, 2000000, 2050000 \rangle$	$\langle T0, B, 2000000, 2050000 \rangle$
	$\langle T0, Commit \rangle$	$\langle T0, Commit \rangle$
	$\langle T1, Start \rangle$	$\langle T1, Start \rangle$
	$\langle T1, C, 600, 500 \rangle$	$\langle T1, C, 600, 500 \rangle$
		$\langle T1, Commit \rangle$

الف

ب

ج

عمل ترمیم برای هر یک از سه حالت بالا به صورت زیر است:

الف)  $undo(T0)$  در B مقدار ۲۰۰۰۰۰۰ و در A مقدار ۱۰۰۰۰۰۰ ذخیره می شود

ب)  $undo(T1)$  و  $redo(T0)$

ج)  $redo(T1)$  و  $redo(T0)$

**۸-۵- عملیات کنترل زمانی (Check Point)**

هنگامیکه پس از رفع اشکال سیستم مجدداً "راه اندازی می شود در اولین مرحله اطلاعات log را کنترل کرده و

براساس آن دستورات redo و undo را جهت ترمیم بانک اطلاعاتی انجام می دهد این عمل اگرچه صحت بانک را

به هم نمی زند ولیکن باعث دو اشکال زیر می گردد:

۱- بررسی تمام رکوردهای log زمانگیر است

۲- تعداد زیادی تراکنش دوباره اجرا می شوند در صورتیکه واقعا "نیازی به اجرای آنها نیست و لذا باعث اتلاف وقت

می گردد.

بر اساس عمل ترمیم می توانند در زمانهای مختلف کنترل شود و این کار می تواند به صورت زیر انجام شود:

الف) تمامی رکوردهای log را از حافظه اصلی به حافظه ثانویه منتقل کرد.

ب) تمامی بلوکهای حاوی رکوردهای اصلاح شده به حافظه ثانویه منتقل شود.

ج) یک رکورد تحت عنوان <check point> در log وارد می شود که بیانگر این است که سیستم تا این لحظه سالم بوده و در اجرای عملیات مشکلی پیش نیامده است.

نحوه اعمال کنترل زمانی :

چگونگی برخورد با تراکنش های اجرا شده براساس قرار گرفتن <Ti,Commit> نسبت به رکورد <check point> بصورت زیر مشخص می شود:

(فرض کنید Tc آخرین رکورد check point و Tf زمان خرابی سیستم باشد)

الف) <Ti,Commit> قبل از عمل Tc باشد:

در اینصورت تراکنش نادیده گرفته می شود زیرا عملیات تمام شده و تراکنش ثبت شده است.

ب) <Ti,Commit> بعد از Tn و قبل از Tf باشد:

در اینصورت با اجرای دستور redo تراکنش اجرای دوباره می شود.

ج) عدم حضور <Ti,Commit>

در این حالت تراکنش نیمه تمام رها شده و لذا با توجه به شیوه اجرای تراکنش به یکی از حالت های زیر عمل می شود:

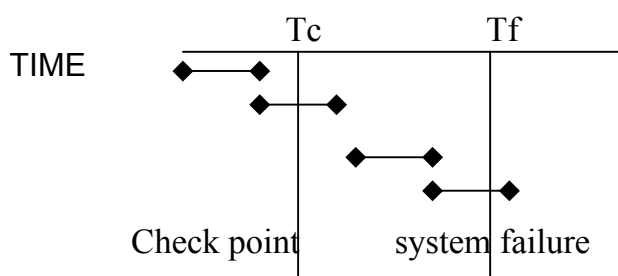
#### • اعمال آنی تغییرات:

در اینصورت اثر احتمالی تراکنش هایی که <Ti,Commit> آنها در log ظاهر نشده است با اجرای دستور undo روی بانک خنثی می شوند.

#### • تاخیر اعمال اصلاحات:

در اینصورت تراکنش هایی که <Ti,Commit> آنها در log ظاهر نشده است نادیده گرفته می شوند.

مثال : فرض کنید در زمانهای مختلف تراکنش های T1 و T4 همراه با Check point بصورت نمودار زیر ظاهر شده باشند.



با توجه به شکل داریم :

• T1 می تواند نادیده گرفته شود.



- T2 و T3 دوباره اجرا می شوند.
- T4 نادیده گرفته می شود (و اثر آن خنثی می گردد).

## یک مثال عملی :

### LitSearch

As part of an introductory databases class at Stanford, I designed [LitSearch](#), a database application which stores data on nearly 4000 titles and their authors as well as literary criticism related to these works. It also contains a full text index of about 100 of them. The project allows users to perform searches on the book metadata, as well as full-text searches on the bodies of the works. It also features a "motif search" which uses synonyms to find passages that are similar to a phrase that the user enters into the search box. It's very unrefined, but can produce some interesting results.

The data was collected from a number of public-domain sources, including [Project Gutenberg](#) and [The Internet Public Library](#).

This page describes the various stages of my project, from the conceptual E/R diagram, to the Java Servlet-based front end. Additionally, I decided to port my database project to [MySQL](#), which I also describe below.

- [Entity-Relationship \(E/R\) Diagram](#)
- [Conversion to a Relational Schema](#)
- [SQL Schema](#)
- [Data Acquisition](#)
- [SQL Interaction](#)
- [MySQL Port](#)
- [Web Application](#)

### LitSearch: Entity/Relationship Diagram

#### Description of Entity Sets

**Words:** word stems that are used throughout all of the works that are cataloged in the database. It's a sort of lexicon containing every stem that the parser has ever come across.

**WordContexts:** the full word that a particular instance of a stem corresponds to. Includes any punctuation/whitespace surrounding the word used.

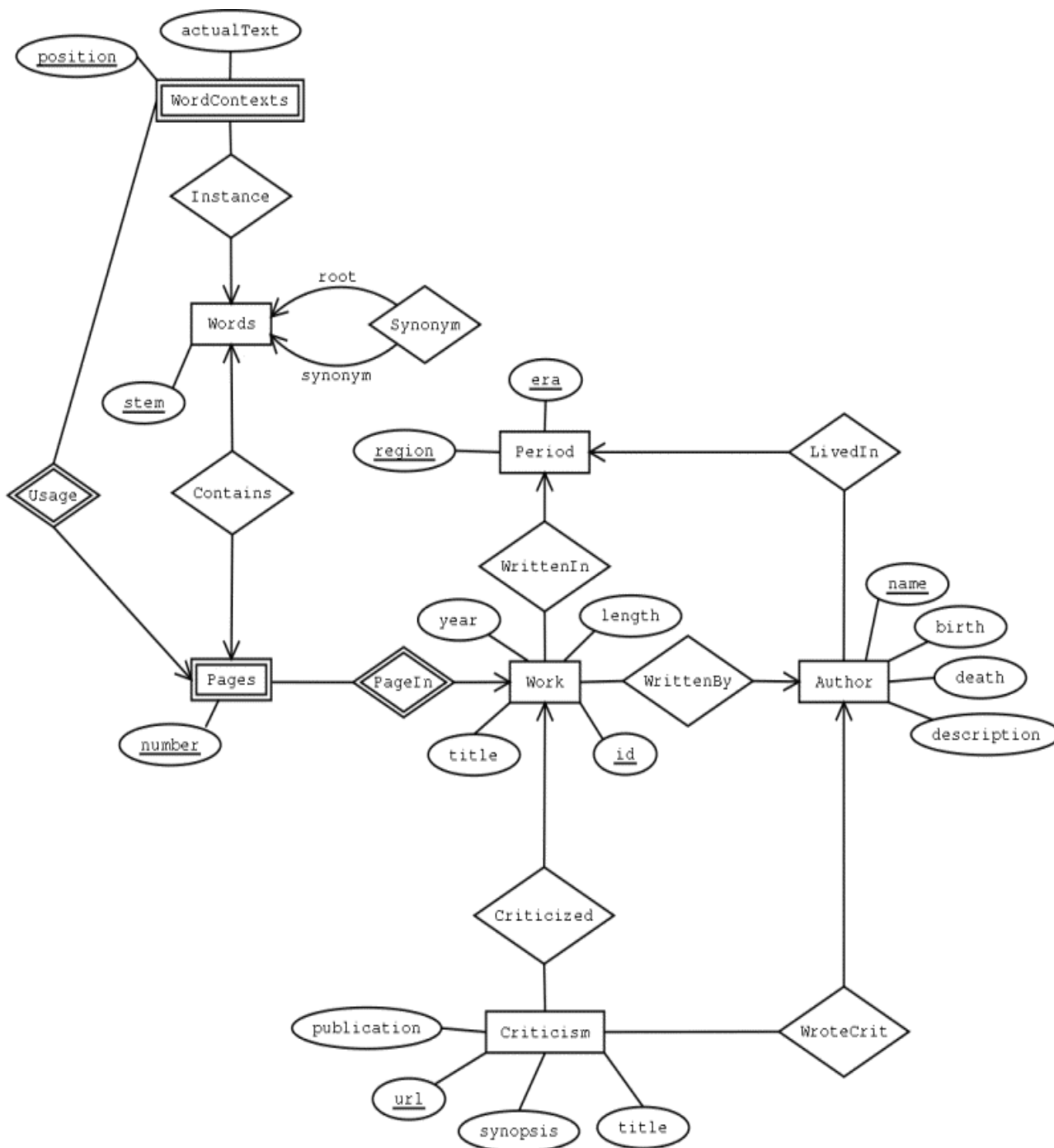
**Criticism:** online literary criticism of a particular work. Contains metadata for the criticism as well as the URL where the criticism may be found.

**Author:** information about authors.

**Period:** information about the various regions and eras used to classify works, such as "19th Century American"

**Works:** information about works (books, collections, etc.) in the database.

**Pages:** data on individual pages in a work. This is a weak entity set because the key here is the page number, so a work is required for uniqueness.



## LitSearch: Conversion to a relational schema

The following is the translation of the E/R Diagram into relations. Everything is in BCNF and 4NF.

### Conversion of Entity Sets

Words (stem)

Criticism(publication, url, synopsis, title)  
Author(name, birth, death, description)  
Period(region, era)  
Works(id, title, year, length)

### Conversion of Relationships

Pages(work, pageNumber)  
WordContexts(work, pageNumber, position, actualText)  
Instance(word, work, pageNumber, position)  
Synonym(root, child)  
WrittenIn(work, region, era)  
Criticized(work, url)  
WroteCriticism(author, url)  
LivedIn(name, region, era)  
WrittenBy(author, work)

### Functional Dependencies

Criticism: url -> publication, synopsis, title  
Author: name -> birth, death, description  
Works: id -> title, year, length  
WordContexts: work, pageNumber, position -> actualText

## LitSearch: SQL Schema

When translated into Oracle SQL, and after constraints and foreign keys are added, the schema becomes:

```
CREATE TABLE Author (  
  name          varchar(128) primary key,  
  birth         int,  
  death        int,  
  description   clob  
);  
  
CREATE TABLE Criticism (  
  title         varchar(255),  
  critic       varchar(128),  
  url          varchar(255) primary key,  
  synopsis     clob,  
  publication  varchar(255),  
  unique (title, critic)  
);  
  
CREATE TABLE Period (  
  region       varchar(64),  
  era         varchar(32),  
  primary key(region, era)  
);  
  
CREATE TABLE Work (  
  id           number(5) primary key,  
  title        varchar(255),  
  year        number(5),  
  length      number(8) check (length >= 0)  
);  
  
CREATE TABLE Word (  
  word         varchar(255),  
  work         number(5),  
  pageNumber   number(5),  
  position     number(5),  
  actualText   clob  
);
```

```
id      number(6) primary key,
word    varchar(32)
);

CREATE TABLE WroteCriticism (
  url      varchar(255) references Criticism(url),
  name     varchar(128) references Author(name),
  primary key (url, name)
);

CREATE TABLE LivedIn (
  region   varchar(64),
  era      varchar(32),
  name     varchar(128) references Author(name),
  primary key (region, era, name),
  foreign key (region, era) references Period(region, era)
);

CREATE TABLE WrittenIn (
  region   varchar(64),
  era      varchar(32),
  work     number(5) references Work(id),
  primary key (region, era, work),
  foreign key (region, era) references Period(region, era)
);

CREATE TABLE Synonym (
  root     number(6) references Word(id),
  child    number(6) references Word(id),
  primary key (root, child)
);

CREATE TABLE Criticized (
  url      varchar(255) references Criticism(url),
  work     number(5) references Work(id),
  primary key (url, work)
);

CREATE TABLE WrittenBy (
  work     number(5) references Work(id),
  name     varchar(128) references Author(name),
  primary key (work, name)
);

CREATE TABLE Page (
  work     number(5),
  pageNumber number(5),
  primary key (work, pageNumber)
);

CREATE TABLE Instance (
  word     number(6),
  work     number(5),
  pageNumber number(5),
  position number(4),
  primary key (word, work, pageNumber)
);

CREATE TABLE WordContext (
  work     number(5),
```

```
pageNumber number(5),
position number(4),
actualText varchar(128),
primary key (word, pageNumber, position)
);
```

## LitSearch: Data Acquisition

### Creating the Synonym table

The synonym table was generated from a comma-separated version of Roget's Thesaurus from the early 1900's (the copyright had expired). Each row in the table contains the root word (the thesaurus entry), and a number of child words (the words listed as synonyms of the word). CreateThesaurus.java is a simple Java program that handles this parsing.

### Parsing Author, Work, and Criticism Metadata

The information about authors, works, and literary criticism was parsed from online resources at Project Gutenberg and the Internet Public Library. I wrote special-case parsers to handle these pages. The parsers for the author data are given below; the others are similar.

### Parsing Works

Works are stored in Project Gutenberg as plain text. I wrote a script to download these texts and then parse their contents into the database. After downloading the text, the header and footer that Project Gutenberg places on the text were removed. Then the body was tokenized to find individual words. Words were then "stemmed", meaning that the suffixes were stripped off such that similar words like "falling", "fallen", and "falls" would all map to the same stem, "fall". This was done such that keyword searches would also return hits containing similar words. It had the additional benefit of keeping the vocabulary size smaller. The algorithm I used was the [Porter Algorithm](#), for which there was a public-domain implementation available.

My source code for acquiring data is very unpolished, since we were not required to turn it in.

Creating the synonym table

- [CreateThesaurus.java](#)
- Parsing Authors
- [getauthors.pl](#)
- [stripauthors.pl](#)
- Parsing Works
- [fetch.pl](#)
- [AddBook.java](#)
- [PorterStemmer.java](#)

## LitSearch: MySQL Port

### Why MySQL?

The accounts we were given on the Oracle server had a 50MB quota. Unfortunately, the database I wanted to load consisted of about 100 works, at 2 to 5 MB each. Factoring in

indices and such, the total disk space required to store this data was well over 1 GB. Figuring that the CS department might balk at a request for so much space for a class project, I decided to run the project on my own box. And not wanting to pay for an Oracle license, I decided to run an open-source alternative, [MySQL](#) and port my SQL code to work under MySQL.

I originally thought that the task of porting to MySQL would be relatively simple. Unfortunately, MySQL does not support some essential functionality. Specifically related to this project, MySQL doesn't support subqueries, views, stored procedures, triggers, and foreign keys. Some workarounds are discussed in MySQL's [list of missing functionality](#). Additionally, I was unable to find a way to hint the MySQL planner, so certain operations that should have used indices apparently were not doing so, causing severe performance problems. As a result, I was forced to change the database schema slightly, and the MySQL schema I used for creating the web application was as follows:

```
CREATE TABLE Author (
name          varchar(128) NOT NULL PRIMARY KEY,
birth         integer NOT NULL,
death        integer NOT NULL,
description   text
);
```

```
CREATE TABLE AuthorCriticized (
url           varchar(255) NOT NULL,
name         varchar(128) NOT NULL,
PRIMARY KEY (url, name)
);
```

```
CREATE TABLE Criticism (
title        varchar(255),
critic       varchar(128),
url          varchar(255) NOT NULL PRIMARY KEY,
synopsis     text,
publication  varchar(255),
keywords     varchar(255)
);
```

```
CREATE TABLE PageContainsWord (
word         mediumint NOT NULL,
page        mediumint NOT NULL,
PRIMARY KEY (word, page)
);
```

```
CREATE TABLE Period (
region       varchar(64) NOT NULL,
era          varchar(32) NOT NULL,
PRIMARY KEY (region, era)
);
```

```
CREATE TABLE PeriodLived (
region       varchar(64) NOT NULL,
era          varchar(32) NOT NULL,
name         varchar(128) NOT NULL,
PRIMARY KEY (region, era, name)
);
```

```
CREATE TABLE PeriodWritten (
region       varchar(64) NOT NULL,
era          varchar(32) NOT NULL,
```

```
work          integer NOT NULL,  
PRIMARY KEY (region, era, work)  
);
```

```
CREATE TABLE Syn (  
root          integer NOT NULL,  
child         integer NOT NULL,  
PRIMARY KEY (root, child)  
);
```

```
CREATE TABLE UniquePage (  
id            mediumint NOT NULL PRIMARY KEY,  
work          smallint NOT NULL,  
page          smallint NOT NULL,  
global_start integer NOT NULL,  
global_end    integer NOT NULL  
);
```

```
CREATE UNIQUE INDEX uniquepage_idx1 ON UniquePage(global_start);
```

```
CREATE TABLE Word (  
id            integer NOT NULL PRIMARY KEY,  
word          varchar(32) NOT NULL  
);
```

```
CREATE INDEX word_idx1 ON Word(word);
```

```
CREATE TABLE WordDetails (  
gpos          integer NOT NULL PRIMARY KEY,  
context       char(24)  
);
```

```
CREATE TABLE WordInstance (  
word          mediumint NOT NULL,  
gpos          integer NOT NULL,  
prev          mediumint NOT NULL,  
PRIMARY KEY (word, gpos)  
);
```

```
CREATE TABLE Work (  
id            integer NOT NULL PRIMARY KEY,  
title         varchar(255),  
year          integer,  
length        integer,  
gutenberg_id varchar(20),  
include       integer  
);
```

```
CREATE TABLE WorkCriticized (  
url           varchar(255) NOT NULL,  
work          integer NOT NULL,  
PRIMARY KEY (url, work)  
);
```

```
CREATE TABLE WrittenBy (  
work          integer NOT NULL,  
name          varchar(128) NOT NULL,  
PRIMARY KEY (work, name)  
);
```

In retrospect, I probably should have used [PostgreSQL](#), which is also open-source, and supports subqueries and most of the functionality that Oracle supports. But I didn't know about it at the time, so I ended up doing the MySQL port.

## LitSearch: Views & Triggers

---

### Views

This section creates two views, one which displays a human-readable view of various information regarding a book including the title, year written, author, region written and era written. The second view displays the text of words which are synonyms. The "Synonym" table stores the id numbers of words that are synonyms -- the view shows what the text of those words are.

```
CREATE VIEW ReadableBookInfo AS
  SELECT Work.title as title, Work.year as year_written,
         Author.name as author, WrittenIn.region as region,
         WrittenIn.era as era
  FROM Work, Author, WrittenIn, WrittenBy
 WHERE Work.id = WrittenBy.work and
        WrittenBy.name = Author.name and
        WrittenIn.work = Work.id;
```

```
CREATE VIEW Synonyms AS
  SELECT w1.word as root, w2.word as child
  FROM Word w1, Word w2, Synonym syn
 WHERE w1.id = syn.root AND
        w2.id = syn.child;
```

---

### Triggers

This section creates two triggers on the database. The first causes a little fake review to be entered in to the database every time a new work is added to the database. The second creates an entry in the synonym table with both entries as the same value (since every word is a synonym of itself) every time a new word is added to the vocabulary.

```
CREATE TRIGGER GoodReviewsTrig
  AFTER INSERT ON Work
  FOR EACH ROW
  WHEN (NEW.title LIKE '%computer%')
  BEGIN
    INSERT INTO Criticism VALUES
      (:NEW.title || ' is a great book',
       'Keith Ito',
       'http://www.stanford.edu/~keithito/' || :NEW.id || '.html',
       'I thought that ' || :NEW.title || ' was the best book of the year',
       'Keiths Reviews',
       'great book');
  END GoodReviewsTrig;
.
```

```
run;

CREATE TRIGGER ReflexiveSynTrig
  AFTER INSERT ON Word
  FOR EACH ROW
  WHEN (NEW.id > 0)
```



```
BEGIN
  INSERT INTO Syn VALUES (:NEW.id, :NEW.id);
END ReflexiveSynTrig;

.
run;
```

### فهرست منابع و مراجع:

۱- H.Korth and A.silberschatz ,Database System Concepts , Fourth Edition ,Mc Graw Hill ,2001.

۲- C.J.Date , Introduction To Database Systems ,7<sup>th</sup> Edition , Addison Wesley ,2000.

۳- Elmasri and Navathe, Fundamentals of Database Systems, third edition,Addison Wesley, 1999.

۴- M.kroenke , Database Processing fundamentals,Design &Implementation ,8<sup>th</sup> Edition.2002.

۵- R.Stephens,Teach Yourself SQL In 21 Days. Second Ed.

۶- روحانی رانکوهی ، سید محمد تقی .مقدمه ای بر پایگاه داده ها ” چاپ چهارم “ انتشارات جلوه .۱۳۷۸.

۷-روحانی رانکوهی ، سید محمد تقی .مفاهیم بنیادی پایگاه داده ها“ چاپ اول ” انتشارات جلوه .۱۳۸۰.