دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه آزاد اسلامی قزوین

خلاصه درس:

اصول طراحی پایگاه داده ها

(سیستم های مدیریت بانک اطلاعاتی)

تهیه و تنظیم:
دکتر بهروز معصومی

فهرست مطالب

فصل اول:

معرفی پایگاه داده ها و مفاهیم اولیه

۱-۱.مقدمه

امروزه بحث در مورد یک سازمان وابسته به داشتن داده‌ها و اطلاعات و مدیریت داده‌ها و تصمیم گیری‌ها بوده و تصمیم گیری صحیح و مناسب نیازمند اطلاعات[1] مناسب است که برگرفته از واقعیات خام تحت عنوان داده‌ها[2] است. داده‌ها وقتی در یک پایگاه داده ذخیره شوند می‌توانند بهتر و به طور کاراتر مدیریت شوند. در این فصل مفاهیم مقدماتی در خصوص پایگاه داده‌ها مورد بررسی قرار می‌گیرند. همچنین تعریف پایگاه داده و مقایسه آن با سیستم‌های فایل پردازی، عناصر تشکیل دهنده پایگاه داده معرفی می‌گردند.

۱-۲.داده یا اطلاع

برای درک بهتر مفهوم پایگاه داده‌ها بهتر است ابتدا نگاهی به تعریف داده و اطلاع بپردازیم. دو اصطلاح داده و اطلاع واژه‌هایی هستند که بیشتر اوقات به جای یکدیگر به کار برده می‌شوند. داده‌ها واقعیاتی خام[3] هستند. ANSI برای داده دو تعریف ارائه کرده است:

۱ -نمایش واقعیات، پدیده ها، مفاهیم یا معلومات به گونه ای صوری و مناسب برای برقراری، تفسیر یا پردازش توسط انسان یا امکانات خودکار.

۲ - هر نمایشی اعم از کاراکتری یا کمیت های آنالوگ که به آن معنایی منتسب می شود.

اطلاع از داده حاصل می‌شود و در حل مسائل به کار می رود. تعریفی که ANSI برای اطلاع ارائه کرده است چنین است:

اطلاع، معنایی است که انسان به داده منتسب می کند.

در واقع داده، موقعی به اطلاع تبدیل می شود که در یک موقعیت مشخص و در یک بستر خاص و برای حل یک مساله مشخص مورد ارزیابی قرارگیرد. به عبارتی اطلاع نوعی پردازش داده برای آشکار شدن معانی آن

---

[1] Information
[2] Data
[3] raw facts

است. اطلاعات حاصل تکوین و پردازش یا تفسیر داده بوده و شامل خواص ارتباط دهندگی و انتقال دهندگی نیز هست. به طور خلاصه می توان نکات مهم را به صورت زیر خلاصه بندی کرد.

- داده ها تشکیل دهنده بلاک های ساختمانی اطلاعات هستند.
- اطلاعات با پردازش داده ها حاصل می شوند.
- اطلاعات برای آشکار کردن معانی داده ها استفاده می شوند.
- اطلاعات دقیق، متناسب و به موقع در تصمیم گیری مناسب نقشی اساس دارند.

۱-۳.تعریف پایگاه داده ها

مفهوم پایگاه داده ها از نظر مؤلفین مختلف با تفاوتهایی در بیان ولی از نظر تکنیکی به گونه ای مشابه تعریف شده است. یکی از تعاریف مناسب برای پایگاه داده عبارت است از:

پایگاه داده ها، مجموعه ای است از داده های ذخیره شده و پایا (در مورد انواع موجودیتهای یک محیط عملیاتی و ارتباطات بین آنها) که بصورت مجتمع و مبتنی بر یک ساختار به طور صوری با حداقل افزونگی تعریف شده و تحت مدیریت یک سیستم کنترل متمرکزمورد استفاده یک یا چند کاربر به طور اشتراکی و همزمان قرار می‌گیرد.

با توجه به این تعریف می توان دریافت که از دیدگاه تخصصی هر مجموعه ای از فایلها ی ذخیره شده لزوما پایگاه داده محسوب نمی گردد. برخی اصطلاحات موجود در تعریف پایگاه داده در زیر توضیح داده شده اند.

- **موجودیت**[1]، هر چیز(شیء) قابل تمایز در دنیای واقعی را گویند که می خواهیم در مورد آن اطلاعات داشته باشیم.
- **مجتمع**[2]، 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 پراکندگی در ذخیره سازی داده های محیط وجود نداشته باشد.
- **تعریف شده بصورت صوری**، در هر محیط ذخیره سازی داده های ذخیره شده باید به گونه ای تعریف شده باشند. برای تعریف نیز امکاناتی لازم است. تعریف شده بصورت صوری به معنی آن است که داده ها به کمک احکام خاصی، در کادر تعریف فایلهای مورد نیاز، تشریح و تعریف شوند و این کار زبان خاصی را لازم دارد.
- **حداقل افزونگی**، افزونگی به معنای تکرار در ذخیره سازی داده هاست. تجمع داده ها و وحدت ذخیره سازی سبب کاهش افزونگی در پایگاه داده ها می شود.
- **تعریف ۲.** پایگاه داده، مجموعه ای از داده ها به صورت مجتمع و اشتراکی است که شامل داده های کاربران نهایی (واقعیات خام) و فرادادها[3](داده هایی در مورد داده ها) است.

با توجه به این تعاریف یک سیستم مدیریت بانک اطلاعاتی (DBMS)[1] به صورت زیر تعریف می شود.

---

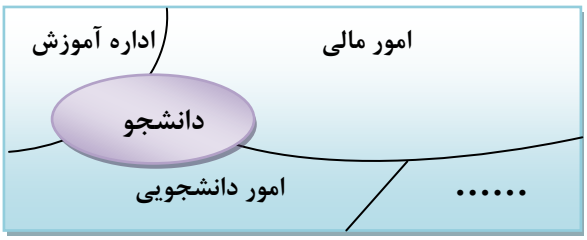
[1] Entity
[2] Integrity
[3] Metadata

یک سیستم مدیریت پایگاه داده شامل مجموعه ای از برنامه هاست که ساختار پایگاه داده ها رامدیریت کرده و دسترسی به داده های ذخیره شده در پایگاه داده ها را کنترل می کند. DBMS امکان به اشتراک گذاری داده ها را در پایگاه داده ها برای کاربران یا برنامه های کاربردی مختلف فراهم می سازد.

۱-۳-۱. یک مثال برای درک بهتر مفهوم پایگاه داده ها.

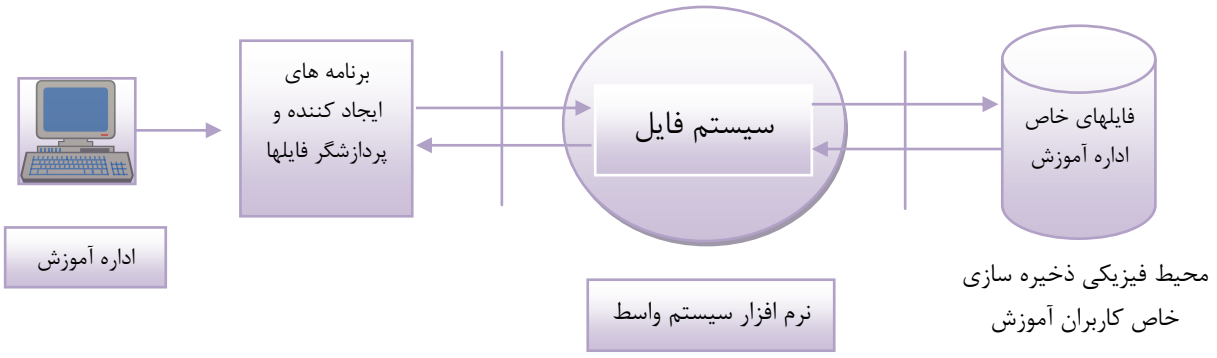
محیط عملیاتی نظیر دانشگاه را در نظر بگیرید که دارای بخش های عملیاتی مختلف است. فرض می شود که سه بخش امور آموزش، امور دانشجویی و امور مالی دانشگاه بخش هایی هستند که می خواهیم برای آنها یک سیستم ذخیره و بازیابی کامپیوتری ایجاد نماییم و نیز فرض می کنیم که تنها نوع موجودیت مورد نظر، موجودیت دانشجو باشد و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

دو روش و مشی کلی در طراحی این سیستم مطرح است:

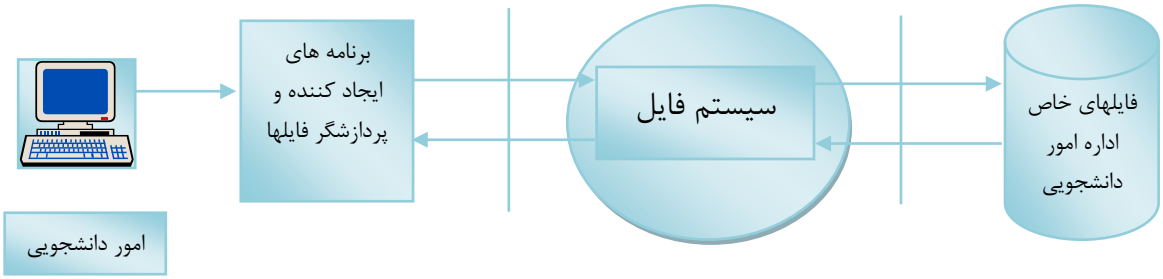


الف – مشی غیر بانکی (سیستم فایل پردازی) [۲]

در این روش هر یک از زیر محیط های عملیاتی به طور مستقل 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.شکل زیر وضعیت کلی این روش را نشان می دهد.



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



[1] Databse Management System
[2] File Processing Approach

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 هستند.

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

۳ – برنامه های کاربردی وابسته به قالب فایل و خصوصیات فیزیکی آن هستند.

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

۵ – اشتراکی نبودن داده ها به این معنی که، 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 اعمال یکسری از عملیات منسجم روی آن سیستم وجود ندارد.

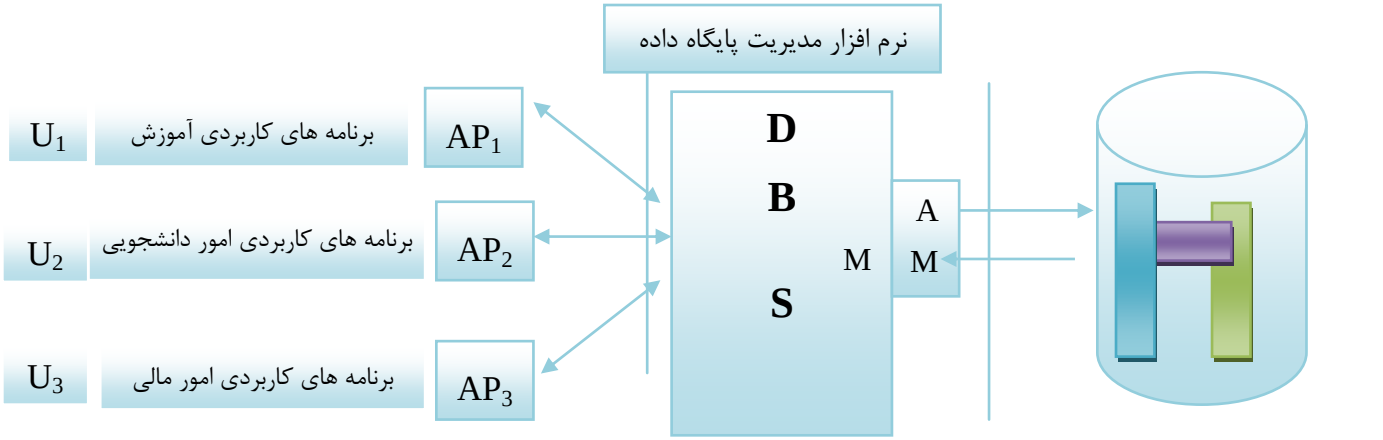
۸- مصرف غیر بهینه امکانات سخت افزاری و نرم افزاری و حجم زیاد برنامه سازی و استفاده غیر بهینه از مهارت و وقت تیمهای برنامه سازی .

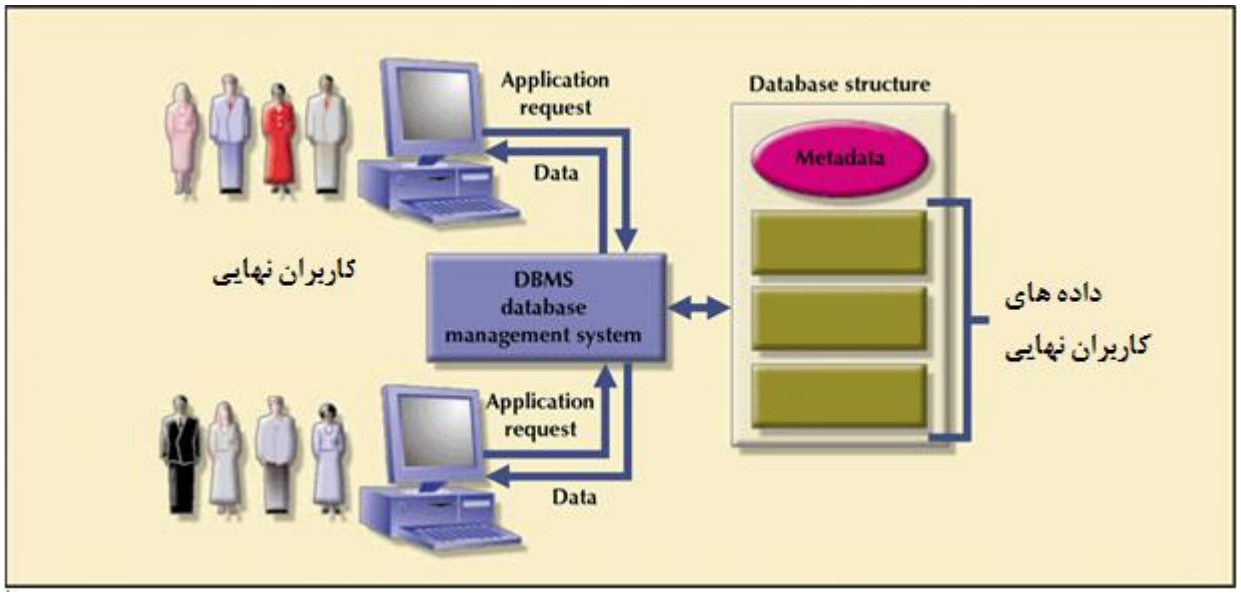
ب – مشی بانکی[1]

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 کرده و با توجه به نیازهای اطلاعاتی تمام کاربران محیط و با استفاده از یک نرم افزار خاص به نام DBMS محیط واحد و مجتمع ذخیره سازی اطلاعات ایجاد می‌شود. با توجه به مثال مطرح شده، رکورد نوع دانشجو فقط یک بار در فایل ذخیره می‌شود و کاربران مختلف هر یک طبق نیاز اطلاعاتی خود از آن به طور مشترک استفاده می‌نمایند. در رکورد نوع دانشجو، تمام صفات خاصه مورد نیاز کاربران مختلف وجود دارند و صفات خاصه مشترک، تنها یک بار در رکورد منظور می‌شوند.

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

[1] Database Approach





برخی مشخصه‌های این روش :

۱ – داده‌های مجتمع: کل داده‌ها به صورت یک بانک مجتمع دیده می‌شوند و از طریق DBMS با آن‌ها ارتباط برقرار می‌شود.

۲ – عدم وابستگی برنامه‌های کاربردی به داده‌ها و فایل‌ها، زیرا DBMS خود به مسائل فایلی پردازی می‌پردازد و کاربران در محیط انتزاعی هستند.

۳ – تعدد شیوه‌های دست‌یابی به داده‌ها

۴ – عدم وجود ناسازگاری در داده‌ها

۵ – اشتراکی بودن داده‌ها

۶ – امکان ترمیم داده‌ها

۷ – کاهش افزونگی

۸ – کاهش زمان تولید سیستم ها

۹ – امکان اعمال ضوابط دقیق ایمنی

۱-۴.عناصر اصلی تشکیل دهنده محیط پایگاه داده ها

محیط بانک اطلاعاتی، مشابه با هر محیط دیگر ذخیره و بازیابی، از چهار عنصر سخت افزار، نرم افزار، کاربر و داده تشکیل می شود.

۱-٤-۱. 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 تولیدشده اند که به نام DBM[1] 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 جا 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.

۱-٤-۲. نرم افزار:

نرم افزار محیط بانکی را می‌توان به دو دسته تقسیم نمود:

۱-٤-۲-۱. نرم افزار کاربری

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

۱-٤-۲-۲. نرم افزار سیستمی

بین بانک اطلاعاتی فیزیکی که داده‌ها به صورت فیزیکی در آن ذخیره می‌شوند و کاربران سیستم، لایه‌ای از نرم افزار موسوم به مدیر بانک اطلاعاتی قرار دارد. سیستم مدیریت بانک اطلاعاتی نرم افزاری است که به کاربران امکان می‌دهد 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.

۱-٤-۳. کاربران

از نظر وظایفی که انجام می‌دهند به دو دسته کلی تقسیم می‌شوند:

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 می‌تواند تقسیم گردد. برنامه نویسان کاربردی مسئول نوشتن برنامه‌های کاربردی بانک اطلاعاتی به زبان سطح بالا یا زبان‌های نسل چهارم (4GL) هستند.

---

[1] Database Machine

۱-٤-٤. داده

منظور داده‌هایی است که در مورد انواع موجودیت‌های محیط عملیاتی و ارتباط بین آن‌ها می‌باشند که اصطلاحاً به آن‌ها داده‌های عملیاتی و یا داده‌های پایا گفته می‌شود. داده‌های ذخیره شدنی در پایگاه داده‌ها ابتدا باید در بالاترین سطح انتزاع مدل‌سازی معنایی شوند .مفاهیم داده‌ها در هر محیط به کمک موجودیت‌ها و ارتباطات نمایش داده می‌شوند.

□انتخاب موجودیت‌ها،

نوع موجودیت، عبارتست از مفهوم کلی شیء، پدیده و به طور کلی هر آنچه از محیط عملیاتی که می‌خواهیم 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 می‌کند و این اولین قدم در طراحی پایگاه داده‌ها 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.

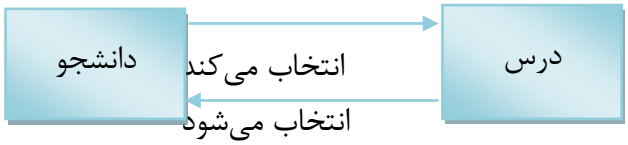
□صفات خاصه

هر موجودیت دارای مجموعه‌ای 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. به عنوان مثال: موجودیت کارمند می‌تواند دارای صفات خاصه شماره کارمندی، نام و حقوق باشد.

- **ارتباط:**

هر نوع ارتباط یک **معنای مشخص** دارد و با یک نام بیان می‌شود و نیز می‌توان گفت که هر نوع ارتباط ,عملی است که بین موجودیت‌ها وجود دارد. انواع موجودیت‌های 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 می‌کند.



- درس توسط دانشجو انتخاب می‌شود.

- بین دو موجودیت می‌تواند بیش از یک ارتباط متفاوت با معانی متفاوت وجود داشته باشد.
- ارتباط ممکن است بین یک نوع موجودیت و خودش باشد. مثال: قطعه X در ساخت قطعه Y به کارمی رود. به این نوع ارتباط ،ارتباط بازگشتی (Recursive Relationship) نیز گفته می‌شود.

•ماهیت ارتباط:

تناظر بین عناصر مجموعه نمونه‌های یک نوع موجودیت، با عناصر مجموعه نمونه‌های نـوع موجودیـت دیگـر را ماهیت ارتباط گویند. ماهیت ارتباط بین انواع موجودیت‌ها 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 سمانتیک حاکم بر محیط عملیاتی تعیین می‌شود.

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

الف : یک استاد حداکثر یک درس را ارائه می‌کند و هر درس دقیقاً توسط یک استاد ارائه می‌شود.

ب: یک استاد حداکثر یک درس را ارائه می‌کند ولی یک درس ممکن است توسط بیش از یک استاد ارائه شود. (ارتباط یک به چند)

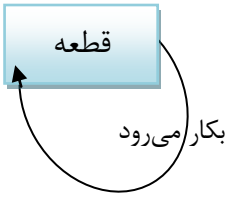
ج: یک استاد ممکن است بیش از یک درس را ارائه کند و یک درس ممکن است توسط بیش از یک استاد ارائه شود. (ارتباط چند به چند)

•درجه ارتباط:

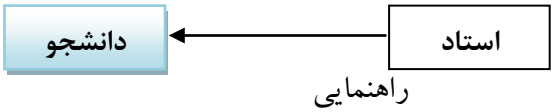
تعداد موجودیت‌هایی که در آن ارتباط مشارکت دارند درجه ارتباط نامیده می‌شود.

مثال:

ارتباط درجه ۱ یا رابطه بازگشتی:



ارتباط درجه ۲:



راهنمایی

۱-۵. انواع سیستم‌های مدیریت پایگاه داده‌ها

یک سیستم مدیریت پایگاه داده‌ها می‌تواند انواع متفاوتی از پایگاه داده‌ها را پشتیبانی کند. از نظر تعداد کاربران، موقعیت سایت‌های در برگیرنده داده‌ها و کاربرد تقسیم بندی‌های مختلفی صورت گرفته است. تعداد کاربران تعیین کننده این است که یک پایگاه داده تک کار بره[1] یا چند کار بره[2] است.

- یک پایگاه داده تک کاربره در هر لحظه فقط از یک کاربر حمایت می‌کند. به عبارتی دیگر، اگر کاربر الف از پایگاه داده استفاده می‌کند، کاربران ب و ج تا زمانی که کار کاربر الف با پایگاه داده به پایان نرسیده است منتظر می‌مانند. اگر یک پایگاه داده تک کاربری روی یک کامپیوتر شخصی اجرا شود به آن **پایگاه داده رومیزی**[3] گفته می‌شود.

---

[1] single-user
[2] multiuser
[3] Desktop database

- پایگاه داده چند کاربری در هر لحظه از چندین کاربر حمایت می کند. در صورتی که یک پایگاه داده چند کاربری تعداد کمی از کاربران را پشتیبانی کند (به طور مثال کم‌تر از ۵۰) یا فقط در حد یک دپارتمان از یک سازمان را حمایت کند به آن پایگاه داده گروه کاری[۱] گویند. اگر پایگاه داده برای یک سازمان بزرگ استفاده شود و تعداد کاربران زیادی(بیش‌تر از ۵۰ و معمولاً ۱۰۰) داشته باشد به آن پایگاه داده سازمان [۲] گفته می‌شود.
- از نقطه نظر مکانی که پایگاه داده در آن قرار گرفته است پایگاه داده را به دو نوع پایگاه داده متمرکز[۳] و پایگاه داده توزیع شده [۴] تقسیم بندی می‌کنند. در یک پایگاه داده متمرکز داده‌ها فقط در یک سایت قرار دارند و در پایگاه داده توزیع شده داده‌ها در چندین سایت توزیع شده ولی کاربر احساس توزیع‌شدگی داده‌ها را نخواهد داشت.
- در حال حاضر تقسیم بندی دیگری از نظر کاربردی انجام گرفته که بر اساس کاربرد تحلیل گونه یا تراکنشی به دو نوع عملیاتی و تحلیلی (به صورت مخزن داده‌ای) تقسیم بندی شده‌اند. در پایگاه‌های داده‌ای عملیاتی هدف، انجام درخواست‌ها و پاسخگویی به تراکنش‌های کاربران است که در آن، پردازش تراکنش‌ها مشتمل بر انجام عملیات روزانه مانند خرید و فروش و عملیات بانکی و مانند آن بوده و داده‌های مورد استفاده در این تراکنش‌ها داده‌های به روز، جاری و با جزئیات هستند، در حالیکه پایگاه دادهٔ تحلیلی، بستر مناسبی فراهم می‌آورد که داده‌ها به منظور پاسخگویی به پرسش‌های تحلیلی به صورت بایگانی شده، سر جمع شده و سازمان یافته، ذخیره شوند. پایگاه داده تحلیلی شامل داده‌هایی است که برای انجام تصمیم گیری‌ها و تحلیل‌ها مناسب است. یک پایگاه داده تحلیلی عبارت است از مخزن داده جمع آوری شده‌ای از منابع اطلاعاتی مختلف، توزیع شده، احتمالاً ناهمگون، تحت یک ساختار چند بُعدی و به صورت یکپارچه. وظیفه اصلی و مهم‌ترین کاربرد پایگاه‌های داده تحلیلی انجام پردازش‌های تحلیلی بر خط (OLAP[۵]) می‌باشد. متناظر این عمل در پایگاه‌های داده عملیاتی، انجام و پاسخگویی به تراکنش‌های کاربران است که پردازش‌های تراکنشی بر خط (OLTP[۶]) نامیده می‌شود.
- سرویس‌دهنده‌های پردازش تحلیلی بر خط که در لایه میانی معماری پایگاه داده تحلیلی قرار دارند، سه نوع هستند:
- ROLAP (پردازش تحلیلی بر خط رابطه‌ای) Relational OLAP

یک سرویس‌دهنده ROLAP، از نوع توسعه یافته‌ای از سیستم‌های مدیریت پایگاه‌های داده رابطه‌ای استفاده می‌کند. پردازش تحلیلی بر خط رابطه‌ای بر اساس نوع ارتباط جدول واقعیت با جداول بعد به اشکال مختلفی مدل می‌شوند.

[1] Workgroup database
[2] Enterprise database
[3] Centralized database
[4] Distributed database
[5] On-Line Analytical Processing
[6] On-Line Transactional Processing

- MOLAP (پردازش تحلیلی بر خط چندبعدی) Multi-dimensional OLAP

یک پایگاه داده تحلیلی چندبعدی داده را به شکل یک مکعب داده می‌بینند. یک مکعب داده مانند فروش اجازه می‌دهد که داده‌ها در ابعاد مختلف مدل شوند و از دیدگاه‌های مختلف مورد بررسی قرار گیرند.

- HOLAP (پردازش تحلیلی بر خط ترکیبی) Hybrid OLAP

HOLAP نیز ROLAP و MOLAP را با یکدیگر ترکیب می‌کند. به عنوان مثال از ROLAP برای داده‌های مربوط به سابقه و تاریخچه استفاده می‌شود، در حالی که، داده‌هایی که به تناوب مورد دسترسی هستند، در یک MOLAP جداگانه نگهداری می‌شوند.

تمرین‌های این فصل.
۱ - **برای هر یک از مفاهیم زیر تعریفی ارایه داده و ارتباط آن‌ها را با یکدیگر بیان کنید: داده، اطلاع، دانش (Knowledge)و عقل(wisdom).**
۲ -**تفاوت‌های اصلی بین مشی طراحی سیستم پایگاهی و سیستم ناپایگاهی را شرح دهید.**
۳ -**تعریف پایگاه داده ها را در شش منبع معتبر خارجی بررسی کنید.**

فصل دوم

مدل E/R

۲-۱. مقدمه

در این فصل، مفهوم مدل سازی داده‌ها مورد بررسی قرار می‌گیرد. مدل سازی داده‌ها اولین قدم در طراحی پایگاه داده‌ها محسوب می‌شود. در واقع مدل سازی داده‌ها همانند پلی ارتباطی بین اشیاء در دنیای واقعی و داده‌های ذخیره شده در یک کامپیوتر محسوب می‌گردد. محققین همواره به دنبال روشی بودند تا پایگاه داده خود را در قالب آن طراحی کنند و آنگاه بتوانند آن را در هر مدلی پیاده سازی نمایند. یک مدل داده‌ای مجموعه‌ای از ابزارهای مفهومی برای توصیف داده‌ها، ارتباط بین داده‌ها و جامعیت داده‌هاست. به عبارتی یک روش تفکر درباره داده‌ها است که به پیاده سازی ربطی ندارد. .در یک نگاه کلی می‌توان انواع مدل‌های داده‌ای را به صورت زیر نام برد :

- مدل‌های مبتنی بر اشیاء[۱]

در این مدل‌ها، داده‌ها به صورت مجموعه‌ای از موجودیت‌ها که نمایش اشیاء در دنیای واقعی می‌باشند دیده می‌شوند. از جمله انواع مدل‌های منطقی مبتنی بر اشیا می‌توان **مدل E /R** را نام برد.

- مدل‌های مبتنی بر رکورد[۲]

در این گونه مدل‌ها، داده‌ها در قالب رکوردهای ثابت و یا با طول متغیر دیده می‌شوند.

۲-۲. مدل E/R

در سال ۱۹۷۶ یک دانشجوی دکترای کامپیوتر دانشگاه MIT به نام چن[۳] 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 می‌باشد. چن نه تنها مدل E/R را معرفی نمود بلکه نمودار E/R را نیز مطرح ساخت. نمودار E/R روشی برای

---

[1] Object based Logical Models
[2] Record based Logical Models
[3] chen

نمایش ساختاری منطقی یک بانک اطلاعاتی به روش تصویری است. این نمودارها ابزارهایی راحت و مناسب را برای درک ارتباطات مابین موجودیت‌ها فراهم می‌کنند. (یک تصویر گویاتر 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 می‌شد تا به جنبه‌های دیگر آن.

۲-۳. نمایش نموداری E/R

مدل E/R به کمک نموداری تحت نام خود قابل نمایش است که در این نمودار مفاهیم مربوطه با اشکال مشخصی ترسیم می گردند.

۲-۳-۱. موجودیت

موجودیت چیزی است که به صورت متمایز قابل شناسایی باشد. آقای چن موجودیت‌ها را به دو دسته منظم[1] (قوی) و ضعیف[2] دسته بندی کرد.

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

۲ – موجودیت ضعیف. موجودیتی است که وجودش وابسته به موجودیت دیگر است. به طور مثال موجودیت اعضاء خانواده کارمند وابسته به موجودیت کارمند می‌باشد. موجودیت آثار منتشره استاد، موجودیت ضعیف موجودیت استاد است.

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

۲-۳-۲. صفات خاصه[3]

در نمودار E/R صفات خاصه به صورت بیضی نشان داده می‌شوند 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 می‌شوند.

در یک تقسیم بندی صفات خاصه را می‌توان 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.

---

[1] Regular entity
[2] Weak entity
[3] Attributes

مثال: در موجودیت دانشجو، صفت خاصه کلید شماره دانشجویی می‌تواند باشد. و یا برای هر فرد کد ملی ۱۰ رقمی یک صفت کلید موجودیت افراد است.

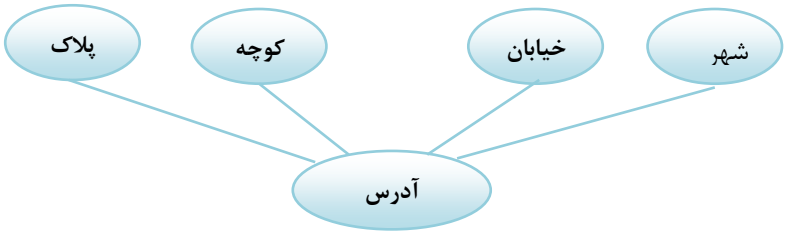
مثال دیگر. در نوع موجودیت گروه درسی، صفت خاصه کلید، ترکیب صفات شماره درس، شماره گروه وترم است.

نکته: 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. به این معنی که هم خود صفت معنی دار است و هم اجزاء آن.

به عنوان مثال: صفت خاصه آدرس می‌تواند 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[1]/چند مقداری[2]

صفاتی 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 جزء صفات تک مقداری هستند.اگر برای یک صفت خاصه چندین مقدار بتواند قرار گیرد صفت خاصه چند مقداری نامیده می‌شود.به طور مثال، صفت خاصه مدرک و یا تلفن برای موجودیت استاد، می‌توانند صفات چند مقداری محسوب می‌شوند. زیرا یک استاد می‌تواند دارای چند مدرک و یا تلفن مختلف باشد. در نمودار E / R برای صفت خاصه چند مقداری از بیضی دو خطی استفاده می‌شود.

د- صفت خاصه مشتق[3] (استنتاجی) [دارای مقدار محاسبه شدنی]

صفتی است که در موجودیت وجود خارجی ندارد ولی در صورت لزوم می‌توان آن را بدست آورد. به عنوان مثال، در موجودیت استاد، صفت خاصه می‌توان صفتی به نام تاریخ تولد در نظر گرفت. در این صورت، صفت

---

[1] Single Valued
[2] Multi Valued
[3] Derived

خاصه مشتق مانند سن مورد نیاز نخواهد بود. زیرا در صورت لزوم می‌توان با تفریق تاریخ تولد از تاریخ جاری سن را به دست آورد.

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

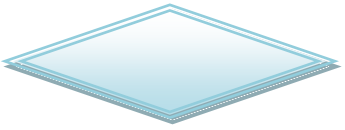
نکته: تصمیم گیری در مورد صفت مشتق در یک موجودیت به عهده طراح است.

۵- صفت خاصه هیچ مقدار[1] پذیر

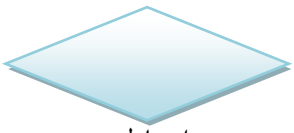
هیچ مقدار یعنی یک مقدار ناشناخته و یا مقدار غیر قابل اعمال. اگر مقدار یک صفت در یک یا بیش از یک نمونه از یک نوع موجودیت برابر هیچ مقدار باشد آن صفت خاصه هیچ مقدار پذیر است. به عنوان مثال، شماره تلفن یک نمونه استاد ممکن است در دست نباشد و یا در یک برنامه درسی‌ترم ممکن است نام استاد در یک روز هنوز اعلام نشده باشد.

۲-۳-۳. ارتباط

هر ارتباط در نمودار E/R به صورت یک لوزی نشان داده می‌شود که محتوای آن در بر گیرنده نام نوع رابطه مورد نظر است. در نمودار ER ارتباط با موجودیت ضعیف به صورت لوزی دو خطی نشان داده می‌شود.

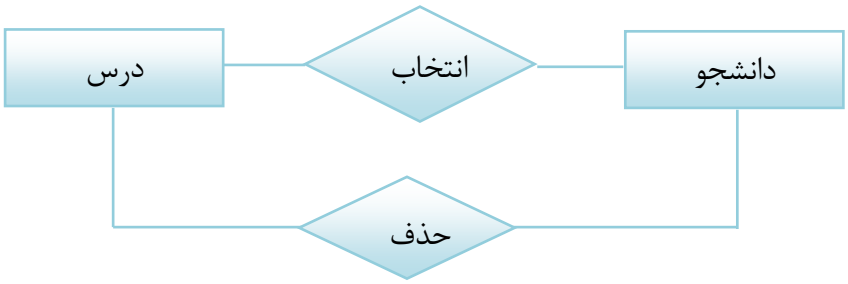


ارتباط با موجودیت ضعیف



ارتباط

عنصرهای هر رابطه (شامل صفات خاصه و موجودیت‌ها) به وسیله خطوط پر به رابطه مربوطه وصل می‌شوند. هر خط ارتباطی بین موجودیت و رابطه دارای برچسب ۱ یا n می باشد که ماهیت ارتباط را مشخص می‌کند.



۲-۳-۳-۱. وضع مشارکت در ارتباط (شرکت اجباری یا اختیاری 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 اجباری (کامل) یا اختیاری (ناکامل) باشد.

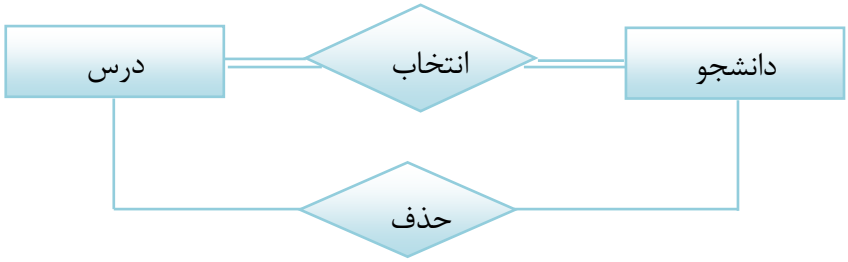
- مشارکت الزامی یا کامل[2]

---

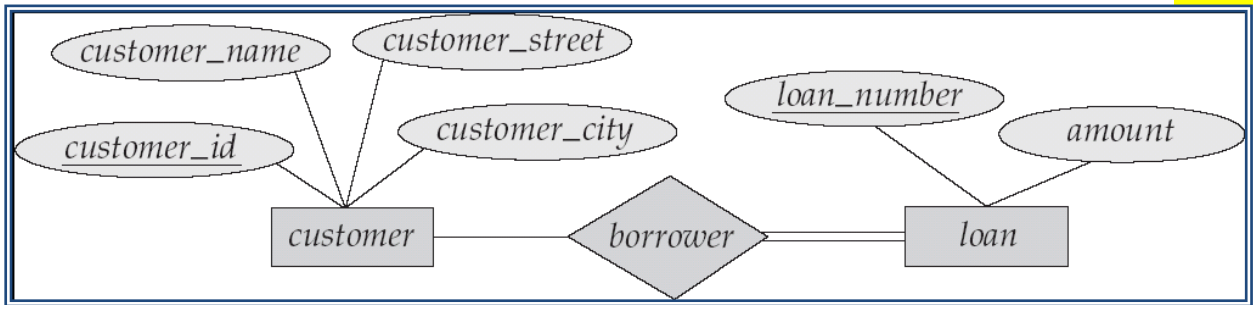
[1] Null
[2] Total participation

- مشارکت یک نوع موجودیت در یک نوع ارتباط را اجباری گویند اگر هر موجودیت در مجموعه موجودیت حداقل در یک ارتباط از مجموعه ارتباط مورد نظر باید مشارکت داشته باشد.
- مشارکت اختیاری(جزئی)[1] یعنی برخی موجودیت‌ها ممکن است در هیچ ارتباطی از مجموعه ارتباطی مشارکت نداشته باشد.

مثال۱. مشارکت دانشجو در ارتباط انتخاب درس الزامی است ولی مشارکت دانشجو در ارتباط حذف درس لزوماً اجباری نیست زیرا هر دانشجو موظف به حذف درس نیست. در نمودار E/R مشارکت الزامی با دو خط نشان داده می‌شود.

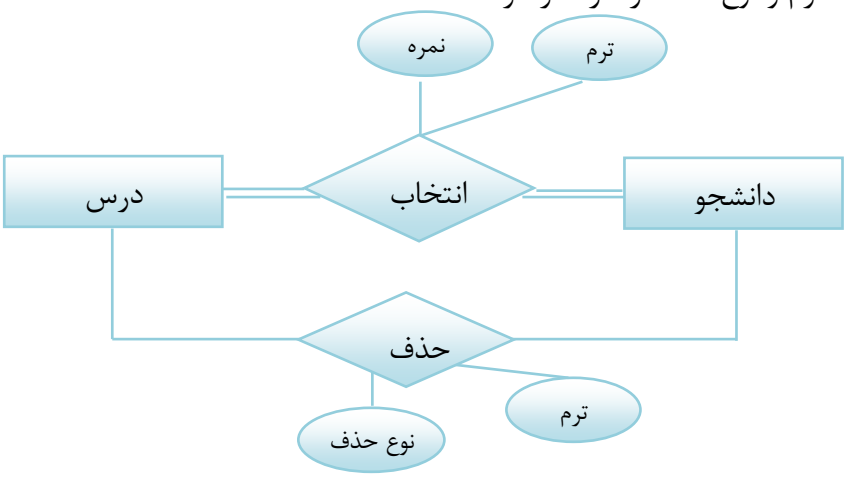


مثال۲. مشارکت مشتری در ارتباط وام گرفتن اختیاری است. [همه مشتریان ممکن است وام نگیرند].



۲-۳-۳-۲. نوع ارتباط به مثابه نوع موجودیت

در یک دید کلی می‌توان گفت نوع ارتباط خود نوعی موجودیت است. زیرا پدیده‌ای است که در دنیای واقعی وجود دارد. با توجه به این تعریف می‌توان گفت چون نوع ارتباط خود نوعی موجودیت است لذا می‌تواند صفت یا صفات خاصه‌ای 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 می‌توان صفت خاصه‌های ترم و نمره و در رابطه حذف صفات‌ترم و نوع حذف را در نظر گرفت.

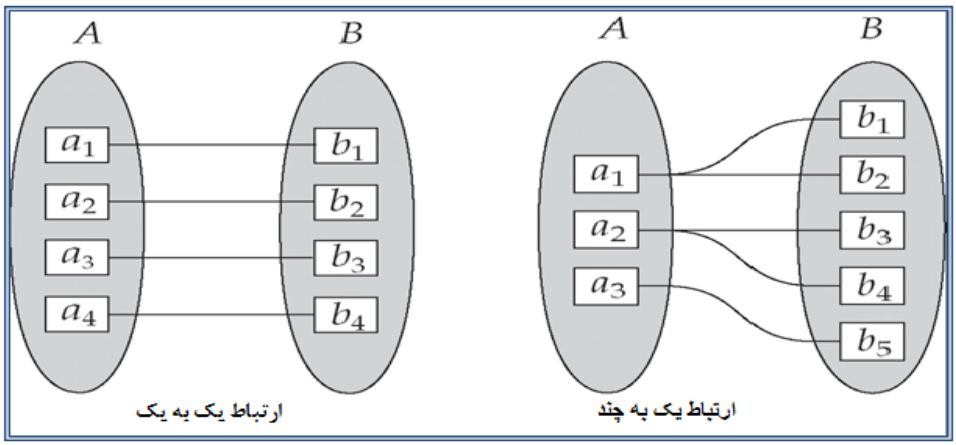


[1] Partial participation

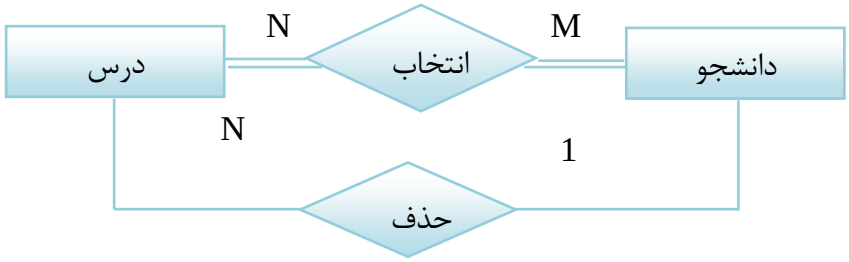
۲-۳-۳-۳. ماهیت نوع ارتباط (اتصال)[1]

چگونگی تناظر بین دو مجموعه نمونه‌های موجودیت را ماهیت ارتباط گویند. می‌دانیم سه نوع تناظر برای ارتباط وجود دارد: تناظر یک به یک ، تناظر یک به چند و تناظر چند به چند. این سه گونه تناظر را چنین نشان می‌دهیم:

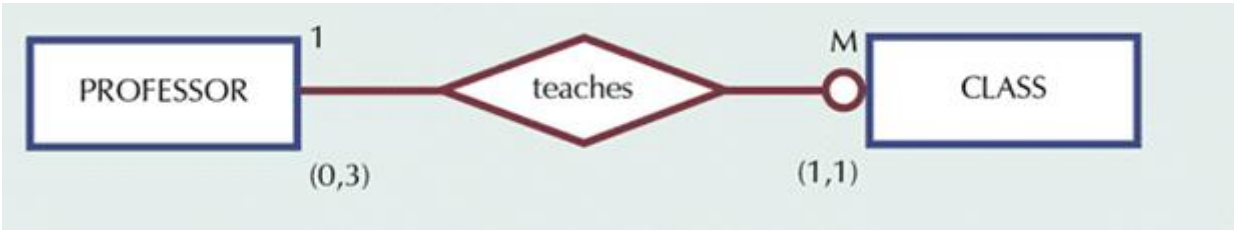
1 : 1 , 1 : N , M : N. اتصال ۱:۱ بسیار کم ظاهر می‌شود. در ارتباط یک به چند بین A و B هر نمونه A با چند B در ارتباط است ولی هر B حداکثر با یک Aارتباط دارد.



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



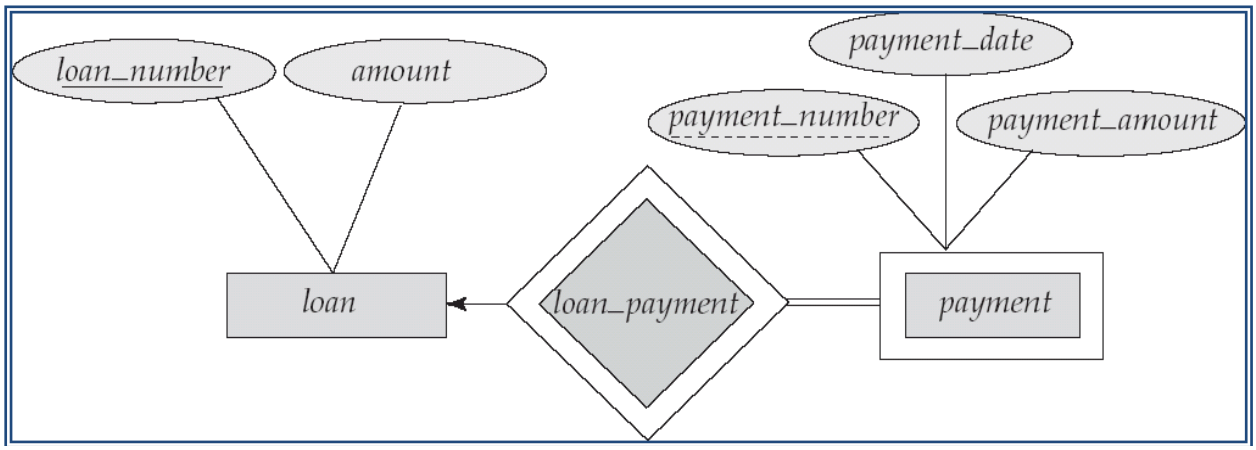
برای نمایش ماهیت ارتباط در نمودار E/R روش دیگری نیز وجود دارد . در این روش به هر مشارکت یک نوع موجودیت در یک ارتباط ، یک زوج عدد صحیح به صورت (min,max) انتساب داده می‌شود. به این معنی که در هر لحظه ، هر نمونه موجودیت e از نوع E باید حداقل در min و حداکثر در max نمونه از ارتباط R شرکت داشته باشد. گر min=0 مشارکت اختیاری و در غیر این صورت مشارکت الزامی است . به این نوع بیان، حد[2] نیز گفته می‌شود. در واقع در این جا حداقل و حداکثر مشارکت نمونه موجودیت را نشان می‌دهد.

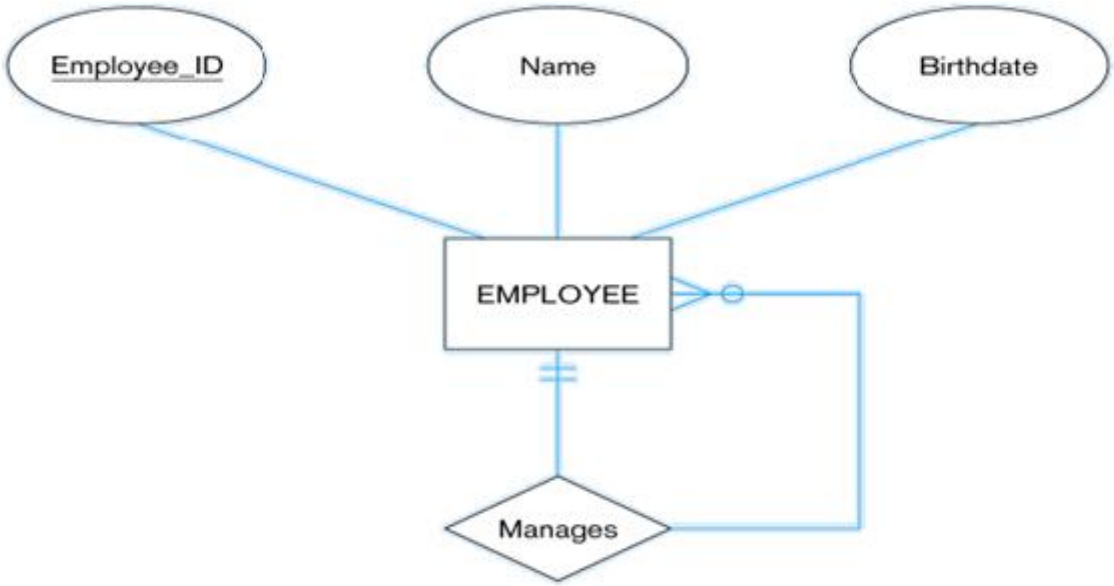


[1] connectivity
[2] cardinality

۲-۳-٤. درجه ارتباط

درجه ارتباط، به تعداد موجودیت‌هایی که در آن ارتباط مشارکت دارند گفته می‌شود. معمولاً درجه ارتباط یک یا دو یا سه است و درجات بالاتر به ندرت دیده می‌شوند. اگر یک نمونه موجودیت با یک نمونه موجودیت از نوع خودش ارتباط داشته باشد ارتباط را بازگشتی یا درجه ۱ گویند. شکل‌های زیر نمونه‌هایی از ارتباطات درجه ۱، درجه ۲ و ۳ را نشان می‌دهد.

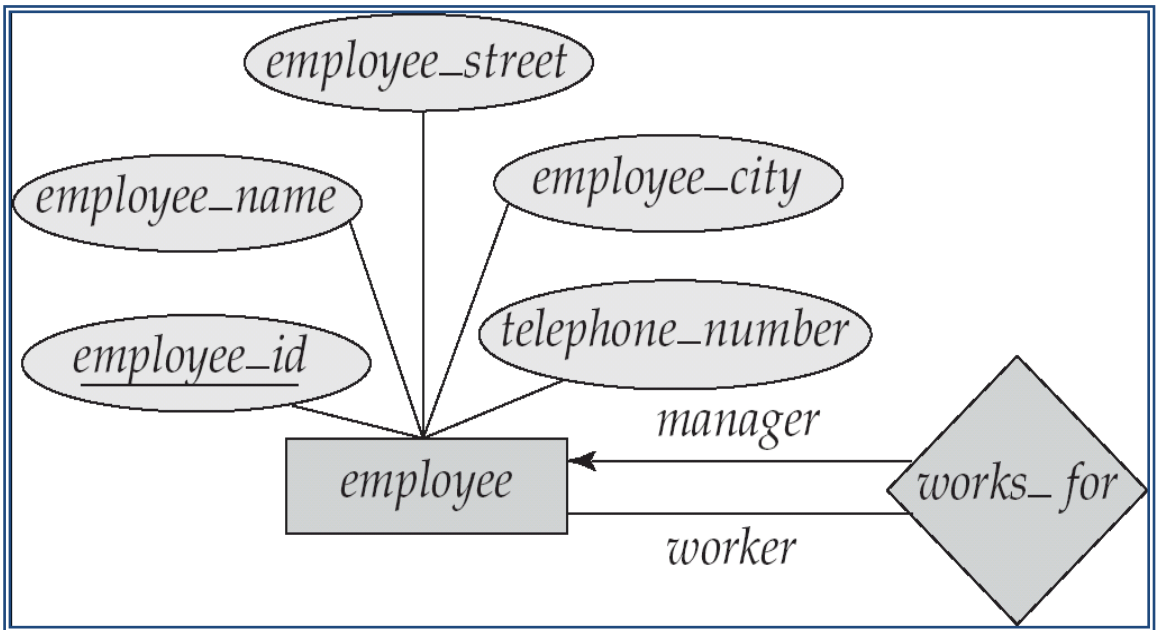




۲-۳-۵. مفهوم نقش[1] در نمودار E/R

در نمودار E/R گاهی مواقع برای وضوح بخشیدن به معنای ارتباط از یکسری برچسب‌ها که بر روی خطوط ارتباطی استفاده می‌شود. به عنوان مثال در شکل زیر که بیان گر یک ارتباط درجه ۱ است کارمند در دو نقش کارمندی و مدیر ظاهر شده است.

[1] Role



۲-۴. موارد افزوده شده به نمودار E/R

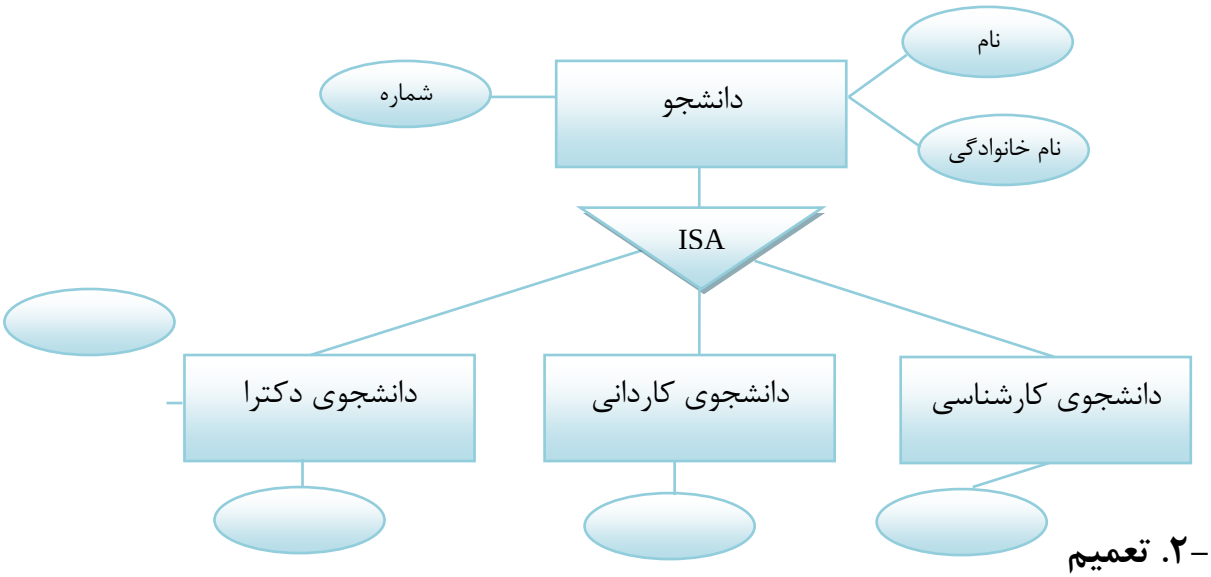
۲-٤-۱. ارتباط سلسله مراتبی

یک موجودیت می‌تواند به طور هم زمان از انواع مختلفی باشد. مثلاً اگر بعضی کارمندان، برنامه نویس باشند و تمام برنامه نو یسان کارمند، آنگاه می‌توان گفت نوع موجودیت برنامه نویس یک زیر نوع از نوع موجودیت کارمند است. در ارتباطات سلسله مراتبی، مجموعه موجودیت‌های سطح پایین‌تر از صفات خاصه مجموعه موجودیت‌های سطح بالاتر ارث می‌برند. به این ترتیب از تکرار بی رویه مجموعه صفات جلوگیری می‌شود.در نمودار E/R به این ارتباط، ارتباط **"گونه‌ای است از..."**، **"هست یک "** یا IS_A گفته می‌شود. در مدل E/R دو دیدگاه در طراحی ارتباطات سلسله مراتبی مطرح است که با توجه به طراحی از بالا به پایین (تخصیص[۱]) و از پایین به بالا (تعمیم [۲]) برگرفته شده است.

۲-۴-۱-۱. تخصیص

مشخص کردن گونه‌های خاص یک شیء را تخصیص گویند. به طور 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 زیر نوع‌هایی داشته باشد. **تخصیص**، یک فرایند از بالا به پایین است. در تخصیص یک موجودیت به گونه‌های مختلف گروه بندی می‌شود. این گروه‌ها به عنوان موجودیت سطح پایین‌تر آن موجودیت منظور می‌گردند. در نمودار E/R تخصیص با یک مثلث حاوی ISA نشان داده می‌شود. یک موجودیت سطح پایین تمام خصوصیات مجموعه موجودیت سطح بالاتر را به ارث می‌برد.

[1] Specialization
[2] Generalization

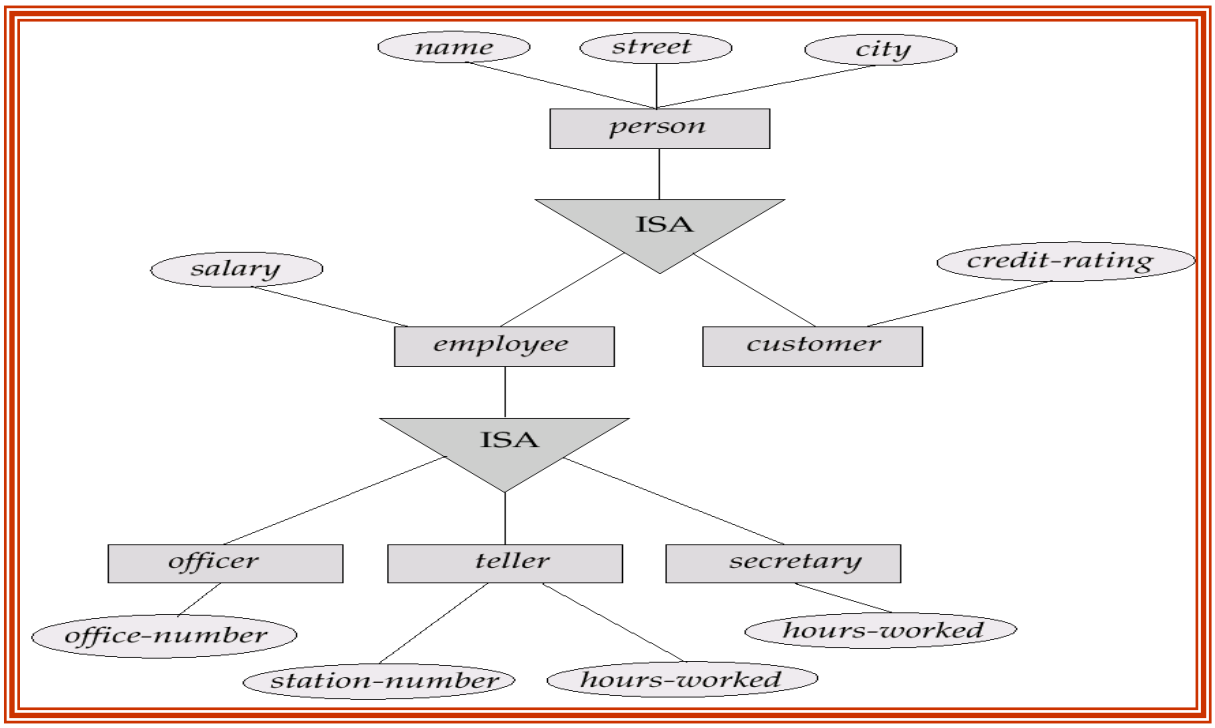


۲-۴-۱-۲. تعمیم

تعمیم، عکس عمل تخصیص است به این معنا که با داشتن زیر نوع‌های خاص، صفات مشترک بین آن‌ها را در یک مجموعه صفات برای یک موجودیت سطح بالاتر در نظر می‌گیریم. تعمیم یک فرایند از پایین به بالاست که تعدادی مجموعه موجودیتی که خصوصیات مشترک دارند را باهم ترکیب می‌کند.

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

شکل زیر مثالی از ارتباط سلسله مراتبی چند سطحی است.



۲-٤-۱-۳. قیود تعریف شده در ارتباط سلسله مراتبی تعمیم/تخصیص

برای مدل سازی دقیق‌تر یک سازمان ، طراح پایگاه داده ممکن است محدودیت‌ها / قیدهایی را در یک تعمیم در نظر بگیرد. یکی از این محدودیت‌ها تعیین شرایط عضویت یک موجودیت سطح بالا به موجودیت سطح پایین است. عضویت یک نمونه موجودیت سطح بالا در مجموعه موجودیت سطح پایین می‌تواند بر اساس شرایط و یا توسط کاربر تعیین گردد.

- **عضویت بر حسب شرط خاص تعریف شده**[1]

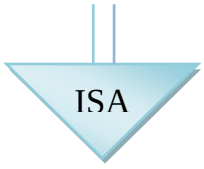
در این حالت، عضویت یک نمونه موجودیت سطح بالا در مجموعه موجودیت سطح پایین بر اساس یک شرط صریح یا گزاره‌ای بیان می‌شود. به عنوان مثال ممکن است در شکل بالا اشخاص با توجه به شرط خاصی جزء مجموعه کارمندان یا مشتریان قرار گیرند.

- **تعریف شده توسط کاربر**[2]

در این حالت، عضویت یک نمونه موجودیت سطح بالا در مجموعه موجودیت سطح پایین توسط کاربر پایگاه داده مشخص می‌شود.

نوع دیگر محدودیت از نظر اینکه آیا نمونه موجودیت‌های هم سطح با هم همپوشانی دارند یا خیر به دو شکل مجزا[3] از هم و همپوشانی[4] وجود دارد. در حالت مجزا بودن، یعنی یک موجودیت می‌تواند فقط به یک مجموعه موجودیت سطح پایین تعلق داشته و نمونه موجودیت‌های هم سطح از هم مجزا هستند که در نمودار E/R با نوشتن عبارت disjoint در کنار مثلث ISA این قید تعریف می‌شود.

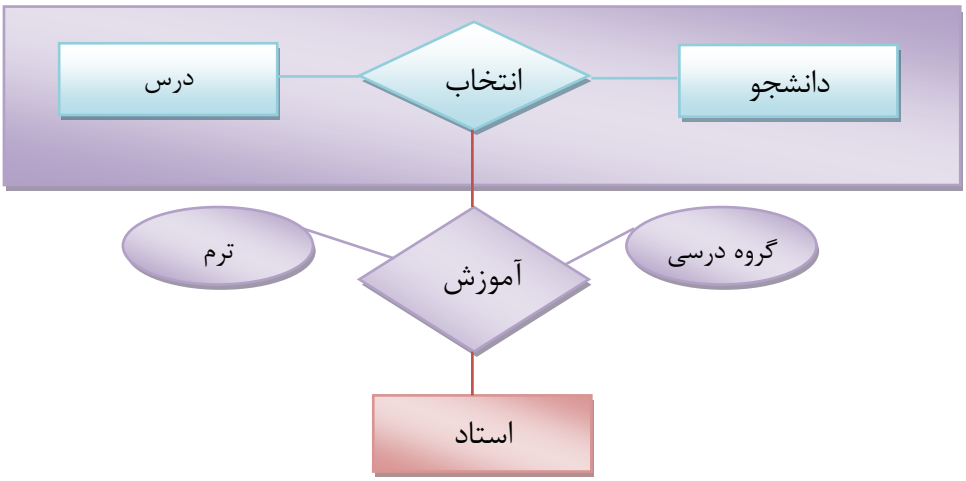
قید دیگری که در تعمیم مطرح شده، قید کامل بودن است که بیانگر این است که آیا یک موجودیت سطح بالاتر باید متعلق به حداقل یکی از مجموعه موجودیت‌های سطح پایین‌تر باشد یا خیر؟ بر این اساس دو نوع قید کامل[5] و جزئی[6] را داریم. در حالت کامل بودن، یعنی یک نمونه موجودیت سطح بالا حتماً باید متعلق به یکی از مجموعه موجودیت‌های سطح پایین‌تر باشد. و در حالت جزئی بودن یعنی یک نمونه موجودیت سطح بالا می‌تواند متعلق به یکی از مجموعه موجودیت‌های 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)



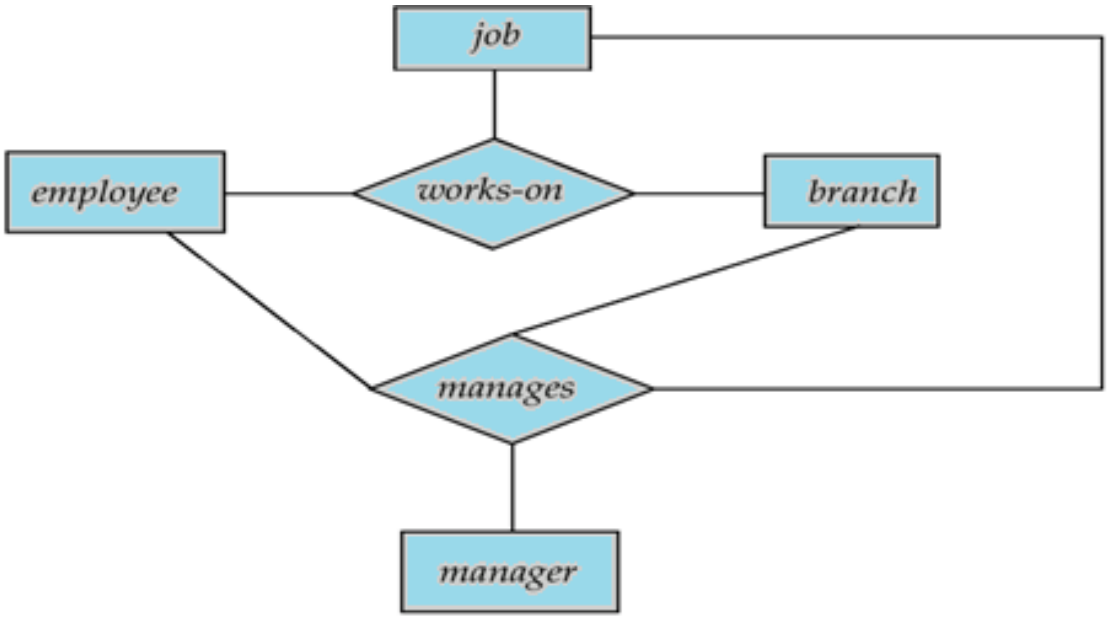
[1] Condition defined
[2] User defined
[3] Disjoint
[4] Overlapping
[5] Total Generalization
[6] Partial Generalization

۲-٤-۲. تجمع[1]

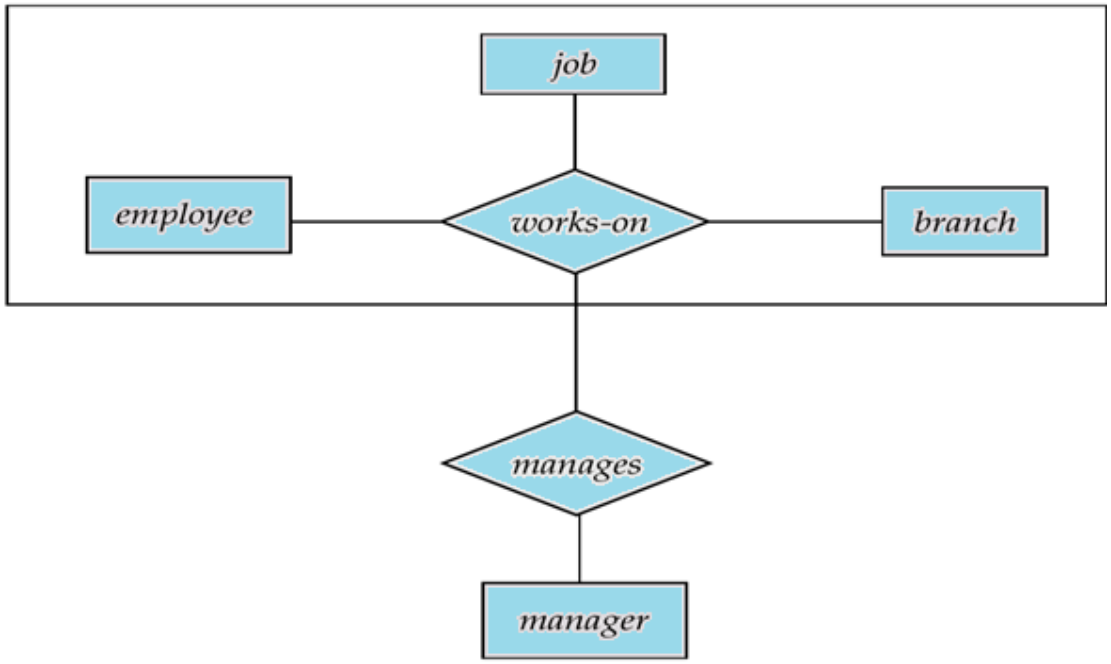
تجمع، یعنی ساختن یک نوع موجودیت جدید و واحد با توجه به وجود دو یا بیش از دو نوع موجودیت,که خود باهم ارتباط دارند. در واقع در تجمع، مجموعه‌ای از موجودیت‌ها و ارتباطات را با هم مجتمع کرده و به عنوان یک نوع موجودیت واحد در نظر می‌گیرند. این نوع موجودیت خود می‌تواند با نوع موجودیت دیگری ارتباط داشته باشد. معمولاً از تجمع استفاده می‌شود که بخواهیم ارتباطی را بین ارتباط‌ها بیان کنیم و یا بخواهیم ارتباطات افزونه را کم کنیم. به عنوان مثال در شکل زیر که ارتباط بین نوع موجودیت‌های دانشجو، درس و استاد را نشان می‌دهد می‌توان همانند شکل زیر مدل سازی نمود.



و یا در مثال زیر می‌خواهیم ارتباطی را بین موجودیت مدیر و ارتباط سه موجودیت کارمند، شغل و شعبه و ایجاد کنیم. این کار می‌تواند یا از طریق ارتباطات افزونه ایجاد شود یا به صورت تجمع بیان گردد. به شکل‌های زیر توجه کنید.



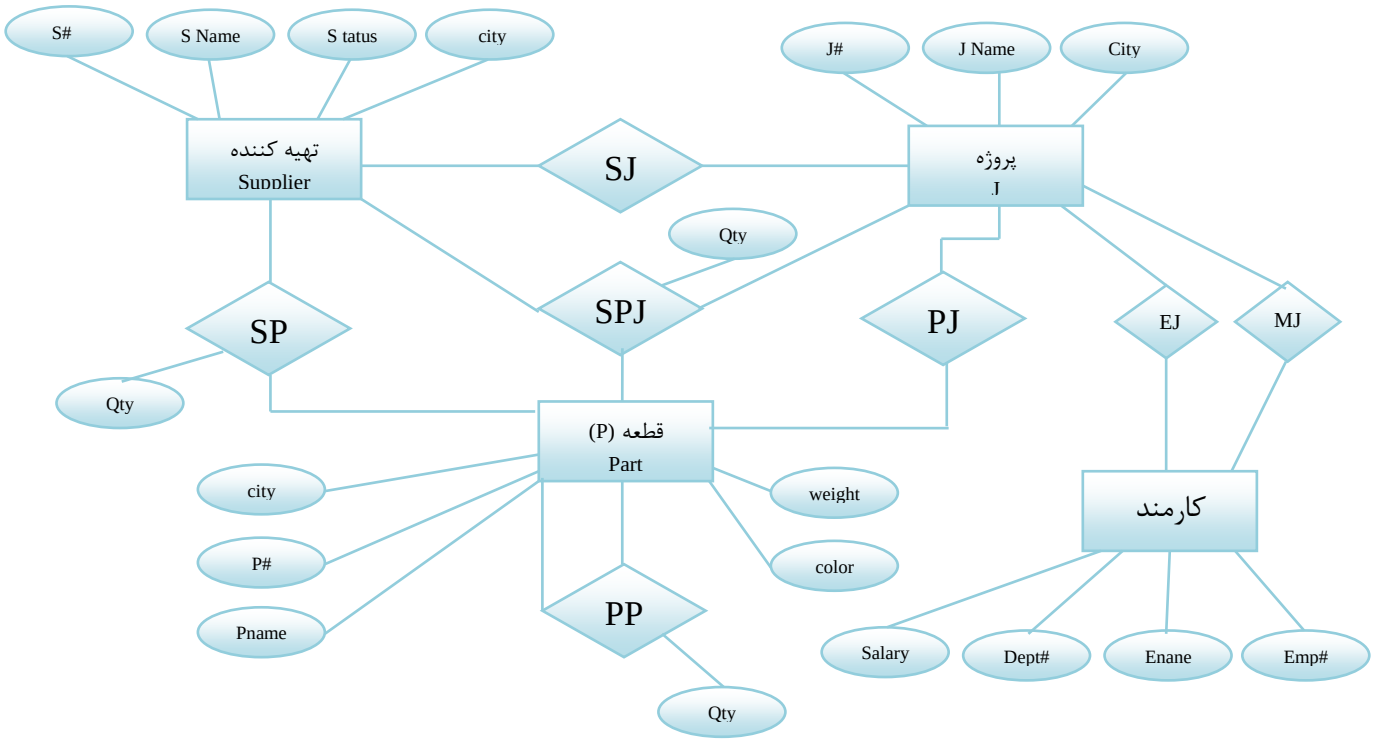
[1] Aggregation



- **مثالی از نمودار E/R در یک محیط عملیاتی.**

سازمان یا شرکتی را در نظر می‌گیریم که پروژه‌هایی را در دست اجرا دارد.در پروژه‌ها از قطعاتی در کار ساخت استفاده می‌شود و تهیه کنندگانی این قطعات را تأمین می‌کنند. قطعات در پروژه‌ها استفاده می‌شوند. هر تهیه کننده در یک شهر دفتر دارد. هر قطعه می‌تواند در ساخت قطعه دیگر نیز بکار رود. کارمند مدیر پروژه است و یا در پروژه کار می‌کند. [برگرفته از کتاب پایگاه داده نویسنده. سی جی دیت].

یک نمودار ساده E/R می‌تواند به فرم زیر باشد. ارتباط ممکن است مابین بیش از دو موجودیت باشد(SPJ) . اطلاعاتی که از این ارتباط بین سه موجودیت به دست می‌آید همیشه لزوماً همان اطلاعاتی نیست که از ارتباط دو به دوی موجودیت‌ها بدست می‌آید. به عنوان مثال در نظر بگیرید:



۱ – تهیه کننده S_1 قطعه P_1 را تهیه می‌کند.
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 نمی‌توان اطلاع ٤ را نتیجه گرفت.

مفهوم دام پیوندی. اگر از ارتباط بین دو به دوی بین موجودیتها نتیجه گرفته شود که حتماً ارتباط بین سه موجودیت یا بیشتر از آن وجود دارددر این صورت طراح گرفتار **دام پیوندی** (تله ارتباطی) شده است.

۲-۵. طراحی پایگاه داده‌ها

طراحی یک پایگاه داده مستلزم مراحلی است که در هر مرحله فعالیت‌هایی انجام می‌شود. طراحی پایگاه داده‌ها مشابه با سیستم‌های نرم افزاری به مراحلی نظیر امکان سنجی، بررسی نیازها و محدودیت‌ها، تجزیه و تحلیل راه حل‌ها و غیره نیاز دارد. شکل زیر نمودار ساده شده مراحل طراحی پایگاه داده‌ها را نشان می‌دهد. در فاز تحلیل نیازها، امکان سنجی ، شناخت نیازهای اطلاعاتی صورت می‌گیرد. در طراحی مفهومی شناخت بخش‌های مختلف و ارتباط بین آن‌ها و سپس طراحی شمای پایگاه داده با استفاده از مدل‌های داده‌ای نظیر E/R یا UML انجام می‌شود. در طراحی منطقی با توجه به DBMS و مدل انتخابی طراحی شمای مفهومی به ساختارهای مدل انتخابی تبدیل شده و پایگاه داده با استفاده از زبان مناسب کد می‌شود. در طراحی فیزیکی، طراحی فایل‌ها و شاخص‌ها روی رسانه با استفاده از ساختار فایل مناسب انجام می‌گردد. در سیستم‌های جدید مدیریت پایگاه داده ابزارهای کمکی (CASE TOOLS) مطرح هستند که برای طراحی پایگاه داده‌ها مورد استفاده قرار می‌گیرند. این ابزارها به طراح پایگاه داده کمک می‌کنند تا در مراحل مختلف مدل سازی 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 به وجود می‌آورند. به طور کلی مزایای استفاده از این ابزارها را می‌توان به صورت زیر نام برد:

۱ -سادگی فرایند ایجاد نمودارها
۲ -تولید خودکار جملات SQL برای تعریف جدول‌ها ، محدودیت‌ها ، اند یس‌ها و دیگر اشیاء مدل رابطه‌ای.
۳ -امکان مستند سازی هر موجودیت ، صفت خاصه ، رابطه و محدودیت.

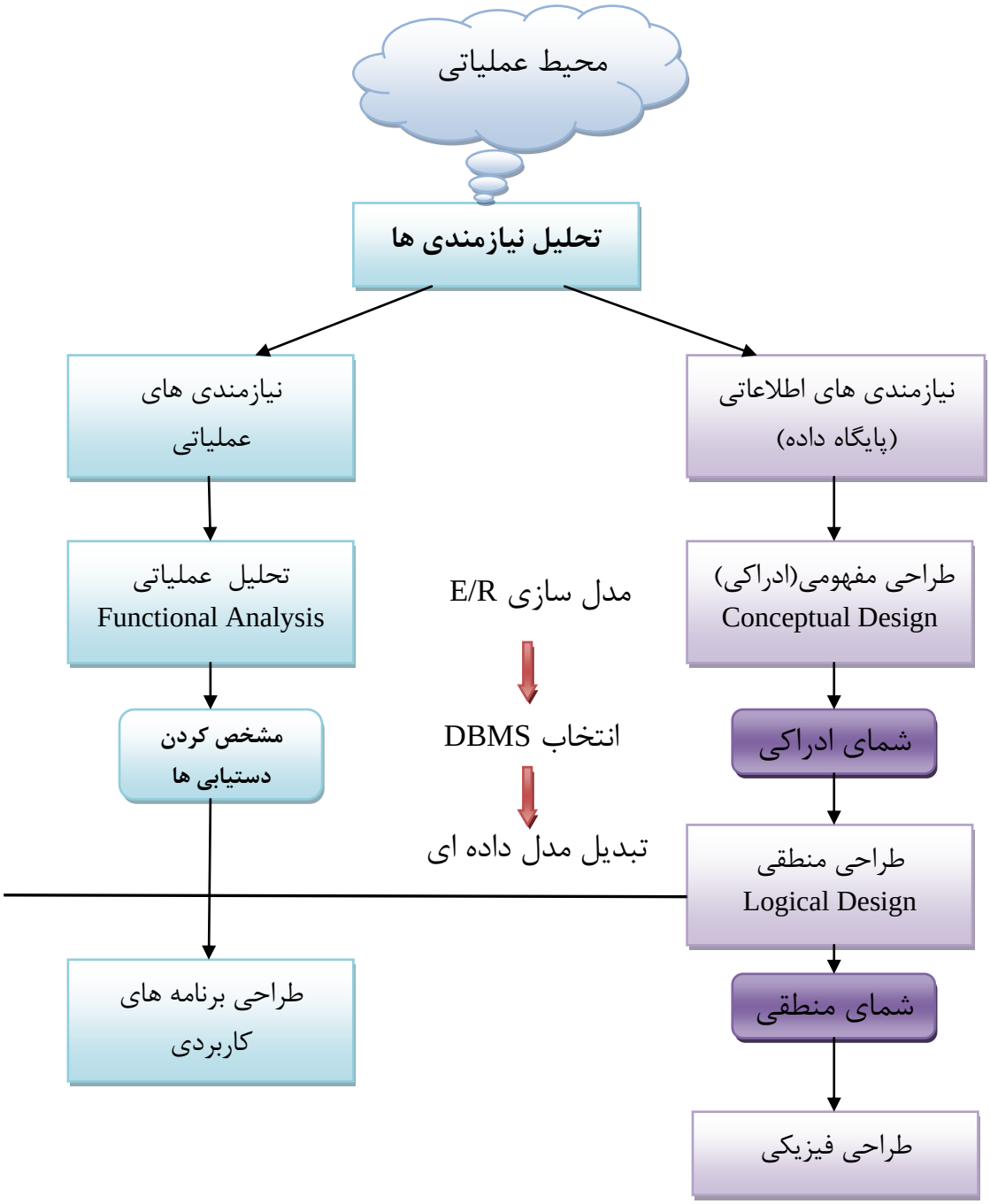
برخی ابزارهایی که در حال حاضر استفاده می‌شوند عبارتند از:

- ER STUDIO برای مدل‌سازی E/R بانک اطلاعاتی
- DB Atrisan برای مدیریت پایگاه داده‌ها و امنیت آن
- Oracle Developer 2000 & Designer 2000 برای مدل سازی پایگاه داده و توسعه برنامه‌های کاربردی.

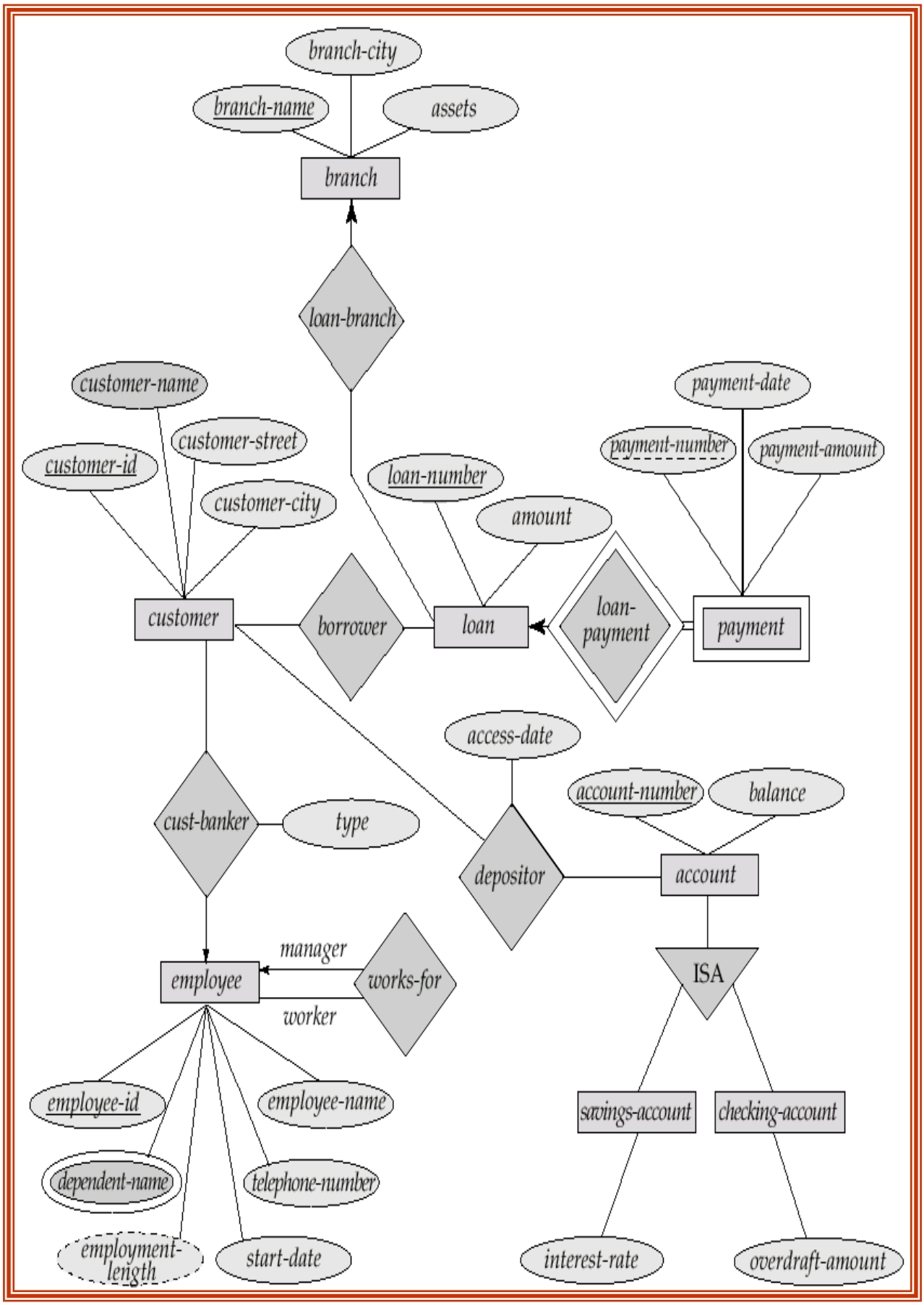
- Platinum Enterprise Modeling suite : ER Win , BpWin برای مدل سازی داده‌ها و پردازش‌ها
- RW Metro برای تبدیل از مدل شی گرایی به مدل رابطه‌ای.
- Rational Rose برای مدل‌سازی UML و تولید برنامه‌های کاربردی به زبان جاوا و C++.
- Visio Enterprise Visual Basic برای مدل سازی داده‌ها و طراحی مهندسی مجدد.
- X Case برای مدل سازی مفهومی. http://www.xcase.com
- Case Studio برای مدل سازی E/R بانک اطلاعاتی.

مقایسه‌ای از ابزارهای مختلف طراحی پایگاه داده‌ها در آدرس اینترنتی http://en.wikipedia.org/wiki/Comparison_of_database_tools آورده شده است.

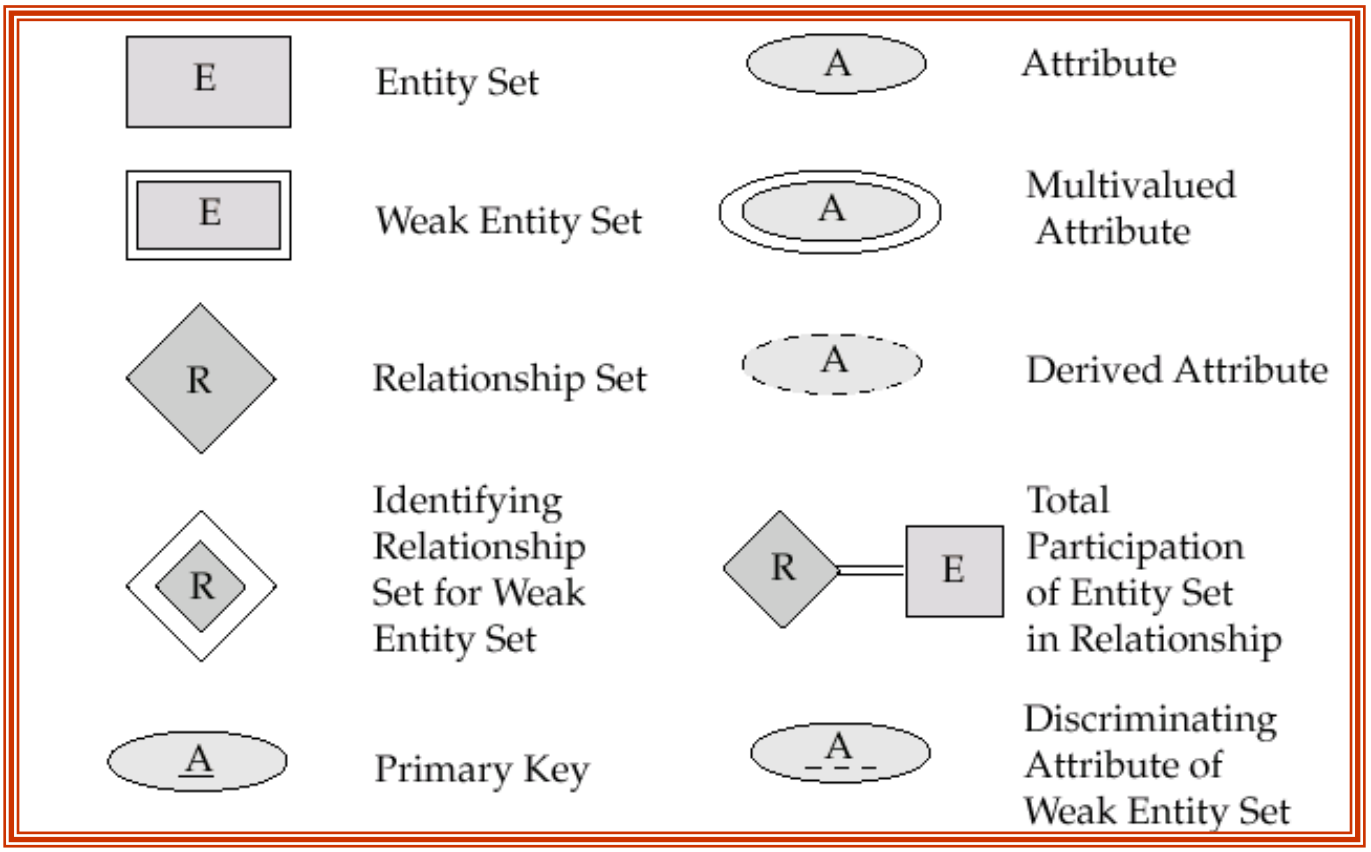
در ادامه این فصل نمونه‌ای از نمودار E/R در محیط بانک آورده شده‌اند.

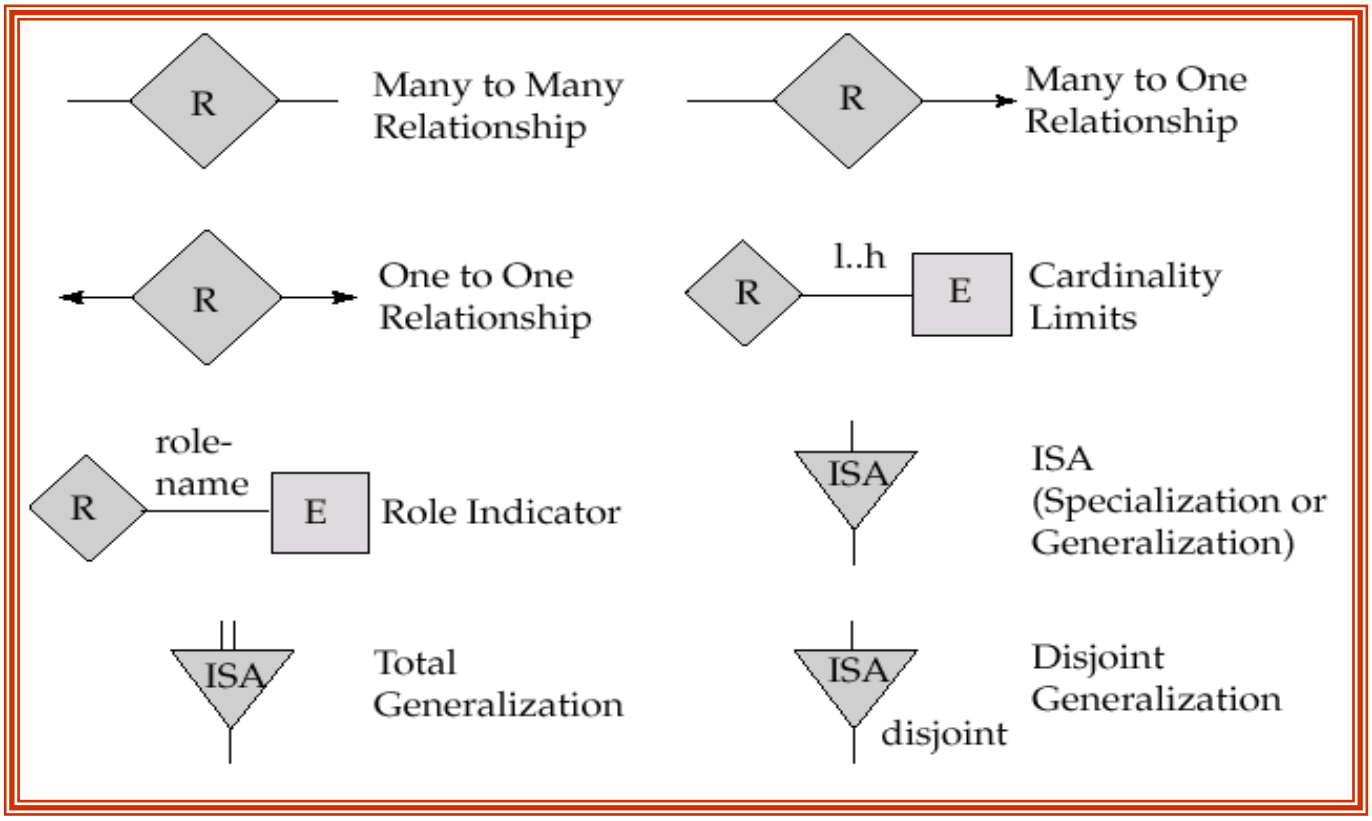


مثال : نمودار E/R یک سیستم بانکی:

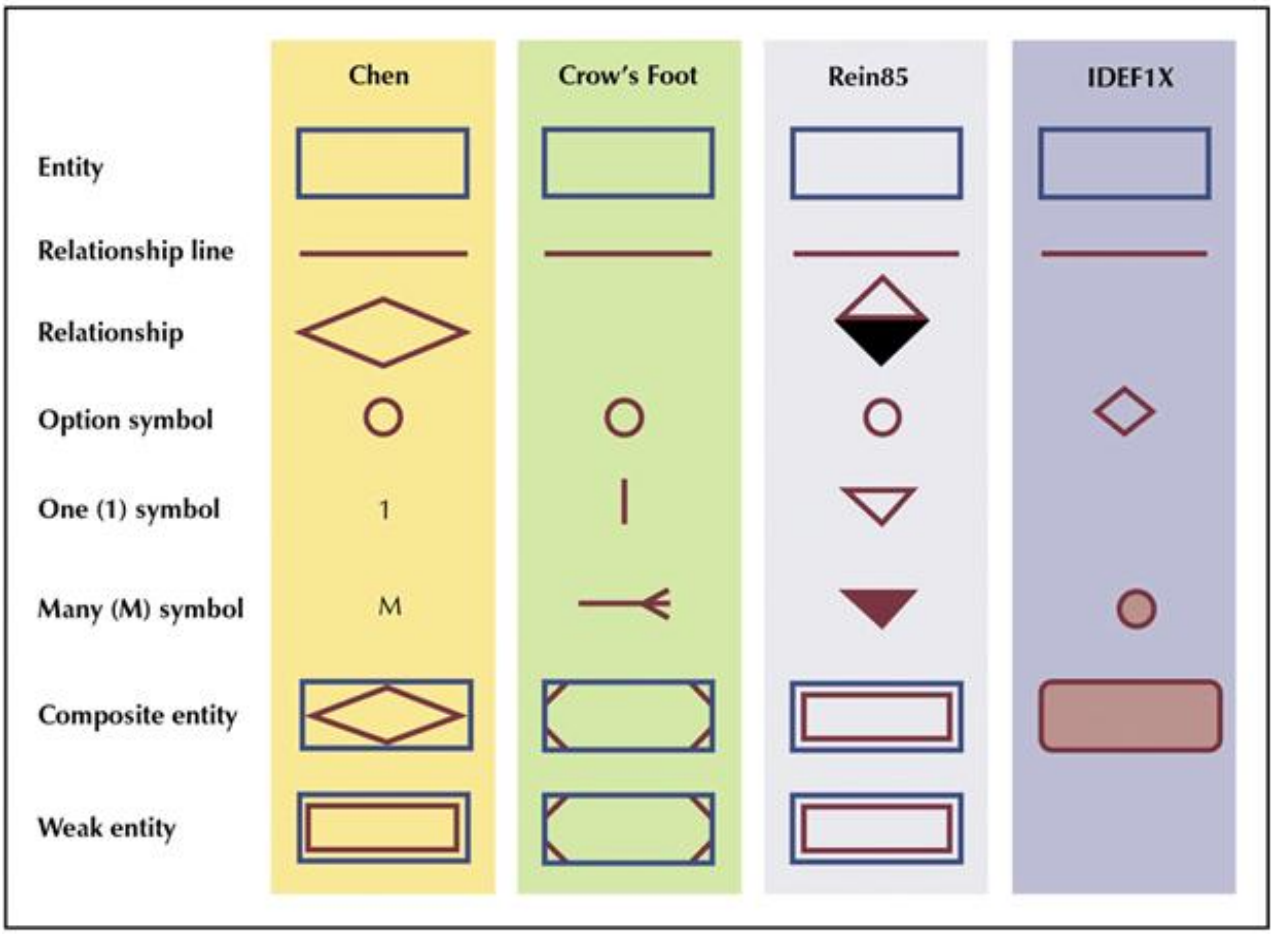


خلاصه شکل‌های بکاررفته در نمودار E/R





مقایسه نمادهای به کار رفته در نمودار E/R



تمرین‌های این فصل.

۱ -فرض کنید می‌خواهیم یک پایگاه داده درباره دانشگاه ایجاد کنیم. اطلاعات مورد نیاز درباره اساتید (دارای شناسه کد استاد) و دروس (دارای شناسه کد درس) است. استادان درس‌های مختلفی را تدریس می‌کنند. وضعیت‌های زیر در محیط حاکم است. برای هر کدام از وضعیت‌های زیر یک نمودار E/R رسم کنید.

الف) استادان می‌توانند یک درس مشخص را درترم های مختلف ارائه کنند و اطلاعات پیشنهادی آن‌ها در همه ترم‌ها بایستی در سیستم لحاظ گردد.

ب) استادان می‌توانند یک درس مشخص را درترم های مختلف ارائه کنند ولی فقط آخرین پیشنهاد ارائه درس آن‌ها ثبت می‌شود.

ج) هر استاد بایستی چندین درس ارائه کند.

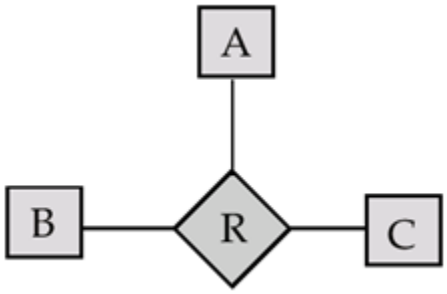
د) هر استاد فقط یک درس را ارائه می‌کند.

ه) هر استاد فقط یک درس را ارائه می‌دهد و هر درس نیز فقط توسط یک استاد ارائه می شود.

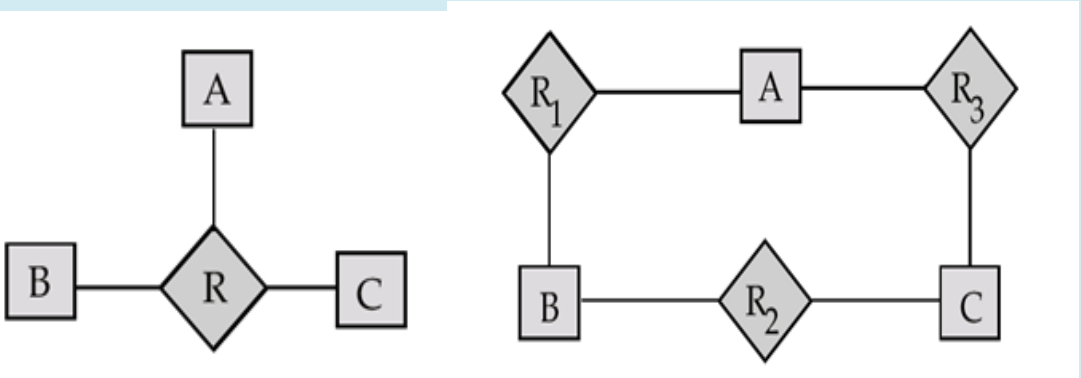
۲ -مفهوم تجمع را با ذکر دو مثال به غیر مثال‌های مطرح شده در کلاس توضیح دهد.

۳ -یک نمودار E/R می‌تواند به صورت یک گراف دیده شود. اگر گراف یک گراف غیر متصل باشد در اینجا به چه معناست؟ اگر گراف دارای دور نباشد در نمودار به چه معنی است؟

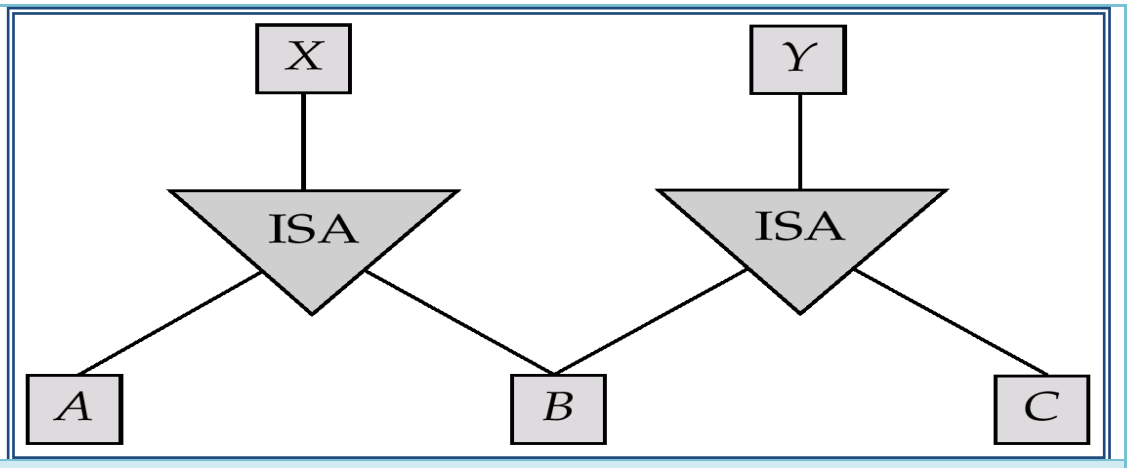
۴ -هر ارتباط سه تایی در نمودار E/R را می‌توان با چندین ارتباط دوتایی نمایش داد. توضیح دهید این کار چگونه انجام می‌شود. مثال زیر را در نظر بگیرید.



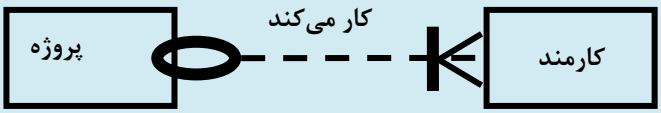
۵ -آیا تبدیل ارتباط سه تایی نمودار E/R زیر به شکل سمت راست درست است؟ در مورد پاسخ خود توضیح دهید.



۶ -نمودار زیر دو ارتباط سلسله مراتبی را نشان می‌دهد. برای مجموعه موجودیت‌های A,B,C صفات خاصه موجود در X,Y را به ارث می‌برند. اگر در X صفت خاصه هم نام با Y باشد چه اتفاقی می افتد؟



۷- با توجه به نمودار E/Rمقابل هر یک از جملات زیر را تفسیر کنید آیا درست یا نادرست



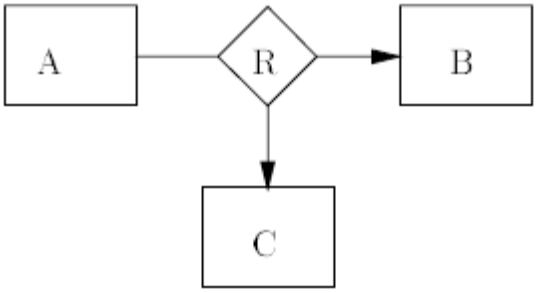
است؟

الف) در یک پروژه ممکن است چندین کارمند کار کنند.

ب) ممکن است پروژه‌ای کارمند نداشته باشد.

ج) هر کارمند حداکثر در یک پروژه کار می‌کند.

۸- نمودار E/R زیر چگونه تفسیر می شود؟ هر نمونه A با چند نمونه B,C ارتباط دارد و چگونه؟



فصل سوم :

مدل رابطه‌ای

۳-۱. مقدمه

مدل رابطه‌ای اولین بار توسط ریاضی دانی به نام کاد[1] به عنوان الگو ساختاری برای طراحی سطوح انتزاعی پایگاه داده مطرح گردید. تأثیر این مدل تا حدی است که بسیاری از افراد متخصص در پایگاه داده آن را در قالب مدل رابطه‌ای می‌شناسند و پایگاه داده را مجموعه‌ای از رابطه(جدول) می‌بینند. در این بخش به شرح مفاهیم مبنایی مدل رابطه‌ای پرداخته می‌شود.

۳-۲. تعریف رابطه

میدان[2]، مجموعه تمام مقادیر صفات خاصه است. به عنوان مثال میدان مقادیر اعداد صحیح در کامپیوتر مجموعه‌ای است که محدود به اعداد ۳۲۷۶۸- تا ۳۲۷۶۷ است. فرض کنید مجموعه دامنه‌های $D_1, D_2, \dots D_n$ داده شده‌اند یک رابطه زیر مجموعه‌ای از ضرب دکارتی چند میدان یعنی در اینجا $D_1 \times D_2 \times \dots D_n$ است. بنا بر این هر رابطه مجموعه‌ای از n تایی $(a_1, a_2, \dots, a_n)$ است که هر $a_i \in D_i$ است. فرض کنید میدان مقادیر زیر را برای موجودیت مشتری داشته باشیم :

customer_name = {Ali, Reza, Hamid, Bardia}

customer_street = {Khayyam, Azadi, Naderi}

customer_city = {Karaj, Qazvin, Tehran}

در صورتی که مجموعه r را به صورت زیر داشته باشیم:

r = { (Ali, Khayyam, Karaj) , (Reza, Azadi, Tehran) , (Bardia,Khayyam, Qazvin}

در این صورت r رابطه‌ای از *customer_name* × *customer_street* × *customer_city* است.

در بحث پایگاه داده، هر رابطه از دو مجموعه تشکیل شده است، یکی موسوم به مجموعه عنوان[3] رابطه و دیگری مجموعه بدنه[4]. **عنوان**، مجموعه‌ای ثابت از صفات خاصه $A_1 \dots\dots A_n$ است که این صفات خاصه هر یک مقادیرشان را از یک میدان می‌گیرند. مجموعه **بدنه** مجموعه‌ای است متغیر در زمان از چند تایی مقادیر صفات

---

[1] Cod
[2] Domain
[3] heading
[4] body

خاصه بنام تاپل[1]. معمولاً در مدل رابطه‌ای از اصطلاح جدول بجای رابطه نیز استفاده می‌شود. یعنی جدول امکانی است برای نمایش مفهوم رابطه به خاطر تا مین و وضوح کاربردی. معمولاً رابطه را به صورت r (R) نمایش می‌دهند که به R شمای رابطه نیز گفته می‌شود. R = (A1,A2,A3,…) در جدول زیر ارتباط بین مفاهیم تئوری و نمایش جدولی طراحان پایگاه داده از مدل رابطه‌ای را نشان می‌دهد.

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

مجموعه عنوان ←

customer_name	***customer_street***	***customer_city***
Ali	Khayyam	**Karaj**
Reza	Azadi	**Tehran**
Bardia	Khayyam	**Qazvin**
Hamid	Naderi	**Karaj**

مجموعه بدنه ←

۳-۲-۱. درجه رابطه

تعداد صفات خاصه رابطه را درجه آن گویند. اگر درجه رابطه یک باشد رابطه یگانی[2]، رابطه درجه دو را دوتایی[3] , رابطه با درجه سه را رابطه سه تایی[4] و رابطه با درجه N را Nتایی گویند.

۳-۲-۱.کاردینالتی

به تعداد تاپلهای رابطه در یک لحظه از حیات آن کاردینالیتی رابطه گویند. کاردینالیتی رابطه در طول حیات رابطه متغیر است. به عنوان مثال، رابطه بالا از درجه ۳ با کاردینالیتی ۴ است.

۳-۲-۱.خصوصیات رابطه

با توجه به تعریف رابطه، هر رابطه دارای ویژگی‌های زیر است:

۱. هر رابطه به کمک یک ساختار ساده به نام جدول قابل نمایش است.

۲. در رابطه، تاپل تکراری 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 می‌شود.

---

[1] Tuple
[2] Unary
[3] Binary
[4] Ternary

٤_ صفات خاصه در رابطه نظم ندارند. این خاصیت نیز از مجموعه بودن عنوان رابطه نتیجه می‌شود

٥_ عناصر تشکیل دهنده تاپل ساده[1] یعنی تجزیه نشدنی می‌باشند. به عبارتی گوئیم یک فقره داده تجزیه نشدنی است اگر نتوان آن را به مقادیر دیگر تجزیه کرد. به عنوان مثال، مقادیر نوع داده تاریخ، ماهیتی مرکب 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 داده‌ها قائل می‌شود بستگی دارد.

۳-۳. مفهوم میدان و نقش آن در عملیات روی بانک

می‌دانیم میدان، مجموعه‌ای از مقادیر است که یک یا بیش از یک صفت خاصه از آن مقدار می‌گیرند. میدان در عملیات روی بانک مزایایی دارد که عبارتند از:

أ. سبب ساده‌تر شدن و کوتاه‌تر شدن شمای پایگاه داده می‌گردد. (از نظر تعداد احکام) . زیرا لازم نیست که در تمام رابطه‌ها , هر بار مشخصات صفات را بدهیم.

ب. کنترل مقداری عملیات در پایگاه.

مقادیر یک صفت خاصه در طول حیات رابطه, از مقادیر میدان برگرفته می‌شوند. به عبارت دیگر باید در میدان وجود داشته باشند. بنابراین به کمک مفهوم میدان می‌توان عملیات روی بانک را از نظر مقادیر صفات خاصه کنترل کرد. به طور مثال اگر میدان مقادیر STAUS به صورت STATUS = {10,20,30,40,50} باشد امکان درج مقدار ۶۰ در بانک میسر نمی‌باشد.

ت. مکانی 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 مقایسه‌اند و سیستم می‌تواند با انجام مقایسه‌های لازم , به پرسش کاربر پاسخ دهد. اما این دو صفت به لحاظ مفهومی غیر قابل مقایسه‌اند، زیرا روی دو میدان ماهیتا به متفاوت تعریف شده‌اند.

ث. پاسخگویی به بعضی از پرس و جوها را آسان می‌کند.

اگر امکان تعریف میدان وجود داشته باشد این تعریف وارد کاتالوگ سیستم به عنوان بخشی از شمای ادراکی پایگاه می‌شود و در شرایطی برخی از کاربران می‌توانند از آن استفاده کنند. به عنوان مثال، در چه رابطه‌هایی از پایگاه اطلاعاتی از تهیه کنندگان وجود دارد؟ اگر میدان‌ها تعریف شوند برای پاسخگویی به این پرس و جو فقط مراجعه به کاتالوگ کفایت می‌کند .

۴-۳. مفهوم کلید در مدل رابطه‌ای

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

- ابر کلید

[1] Atomic

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

مفهوم ابر کلید [1]

مجموعه‌ای از یک یا چند صفات خاصه را که دارای یکتایی مقدار[2] باشند ابر کلید گویند. به بیان دیگر، هر ترکیبی از صفات خاصه رابطه که در هیچ دو تاپل مقدار یکسان نداشته باشد. این کلید تنها نوعی است که کاهش ناپذیر[3] نیست یعنی زیر مجموعه‌ای از آن هم ممکن است کلید باشد.

کلید(نامزد)کاندید[4]

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

۱-یکتایی مقدار

به این معنا که در هر لحظه از حیات رابطه مقدار (A_i , A_j ..A_k) یکتا باشد.

۲- کاهش ناپذیری

به این معنی است که از نظر تعداد اجزاء در حداقل باشد در عین حال که یکتایی محفوظ بماند. گوییم زیر مجموعه‌ای 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 می‌بینیم هر ابر کلید لزوماً کلید کاندید نیست اما هر کلید کاندید جزء مجموعه‌های ابر کلید رابطه هست.

نکته ۱. هررابطه 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.

تعریف رابطه **تمام کلید** (ALL KEY) .رابطه ای که مجموعه **عنوان** رابطه تنها کلید کاندیدرابطه باشد.

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

---

[1] Super Key
[2] Uniqueness
[3] Minimality
[4] Candidate Key

کلید اصلی[1]

کلید اصلی، یکی از کلیدهای کاندید است که طراح انتخاب می‌کند. دو معیار 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 حاصل از ترکیب صفات خاصه.

کلید اصلی شناسه تا پل است و بایستی به نوعی به سیستم معرفی شود که معمولاً در سیستم‌های رابطه‌ای با عبارت Primary Key (Attributes) تعریف می‌شود.

نام کلید	توضیح
ابر کلید	مجموعه‌ای از یک یا چند صفات خاصه را که دارای یکتایی مقدار باشند ابر کلید گویند.
کلید کاندید	ابر کلیدی که خاصیت کاهش ناپذیری داشته باشد کلید کاندید گویند.
کلید اصلی	یکی از کلیدهای کاندید است که طراح انتخاب می‌کند.

کلید فرعی(بدیل)[2]

هر کلید کاندید غیر از کلید اصلی کلید فرعی نامیده می‌شود.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.

کلید خارجی[3]

کلید خارجی، خاصیت یکتایی مقدار نداشته وصفتی در یک رابطه است که در رابطه دیگر کلید اصلی (فرعی) باشد و برای برقراری ارتباط بین دو رابطه استفاده می‌شود. به عبارت دیگر، هر صفت خاصه‌ای از رابطه مانند R_j (تکی یا چند صفتی) مانند A_i که در رابطه‌ای دیگر مثلاً R_i کلید اصلی باشد کلید خارجی R_j نامیده می‌شود.

مثال۱. پایگاه داده زیر را که در رابطه با ماشین، راننده و مالک ماشین است در نظر بگیرید.

CAR(CarNo, Sno, Name , Color)
Driver(DriverNo,DriverName, Type, Age)
Owner(CarNo, DriverNo, Date)

در رابطه مالک فیلدهای CarNo, DriverNo کلید خارجی رابطه هستند. این فیلدها هر کدام در رابطه‌های خود کلید اصلی هستند.

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

---

[1] Primary Key
[2] Alternate Key
[3] Foreign Key

مثال۲. دو رابطه زیر را که درباره دپارتمان و کارمند است در نظر بگیرید:

Department (Dept # , Dname , manager - Emp #, budget)
Employe (Emp # , Ename, Dept # , Salary)

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

توجه: لزومی ندارد R_j از R_i متمایز باشد. در رابطه های بازگشتی زیر، هر کارمند دارای یک مدیر است و هر مدیر چندین کارمند را مدیریت می کند. صفت Manager-Emp# کلید خارجی رابطه است که به فیلد Emp# مراجعه می کند.

Employe (Emp # , Ename, Manager -Emp # , Salay).

سوال: نقش کلید خارجی چیست؟

کلید خارجی امکانی است برای ایجاد ارتباط بین تاپل ها. به عنوان مثال وجود کلیدهای خارجی CarNo, DriverNo در رابطه Owner نمایشگر ارتباطی است که بین تاپل‌های رابطه Car و تاپل‌های رابطه Driver 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. به عنوان مثال در دو رابطه استاد و دانشجو ممکن است هر دو دارای صفت City باشند. در این صورت صفت City در رابطه امکان ارتباط بین دو موجودیت را به وجود می‌آورد در صورتی که صفت City نه کلید خارجی استاد و نه کلید خارجی دانشجو است.

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

FOREIGN KEY (Attributes) REFERENCE RelationName

۳-۵. تبدیل مدل E/R به مدل رابطه‌ای

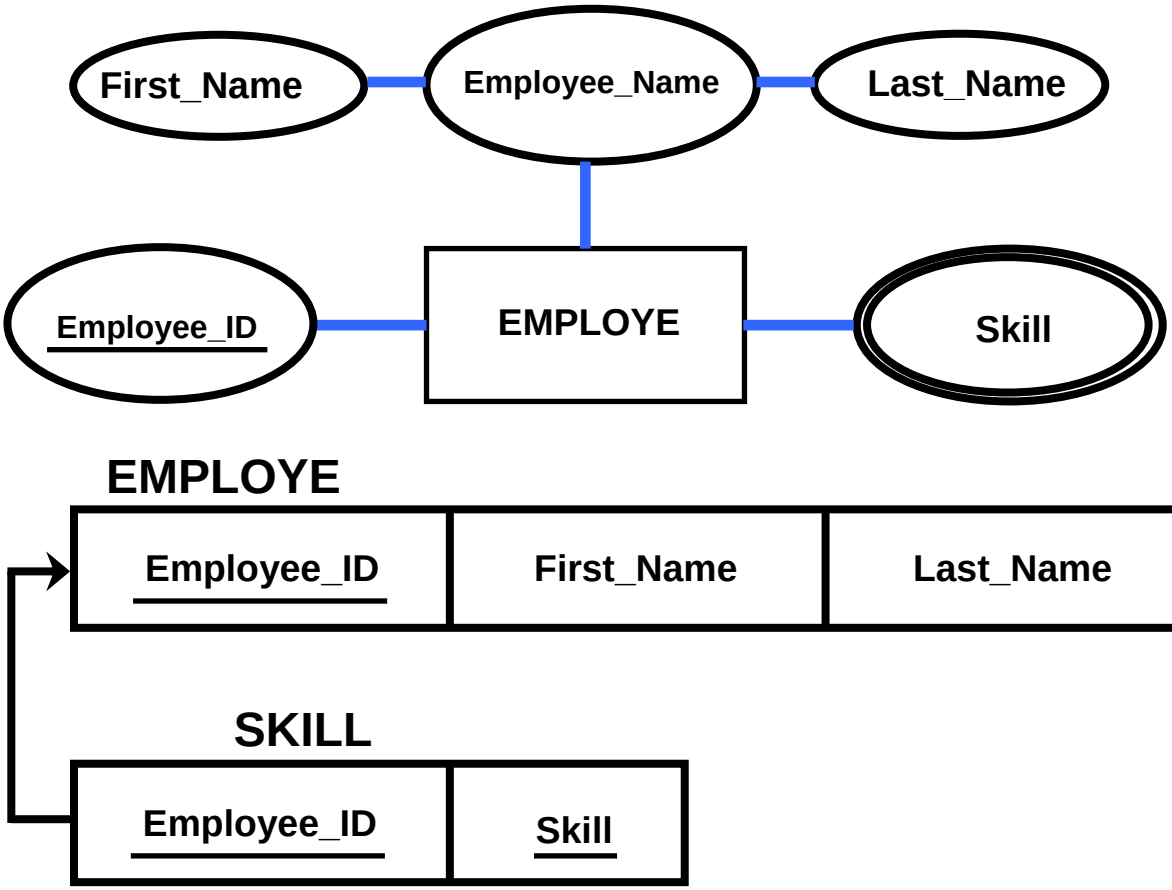
یک پایگاه داده طراحی شده بر اساس مدل موجودیت/ارتباط می‌تواند توسط مجموعه‌ای از جدول‌ها نمایش داده شود. برای تبدیل مدل موجودیت/ارتباط به مدل رابطه‌ای از یک سری قوانین استفاده می‌شود که در ادامه آورده شده‌اند.

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

صفات مرکب در مدل رابطه‌ای وجود نداشته بلکه فقط صفات تشکیل دهنده صفات مرکب به طور مجزا در همان جدول قرار می‌گیرند.

- هر صفت چند مقداری مانند M از موجودیت E با یک جدول مجزای EM نشان داده می‌شود. این جدول دارای ستون‌های متناظر با کلید اصلی Eو صفت M خواهد بود.

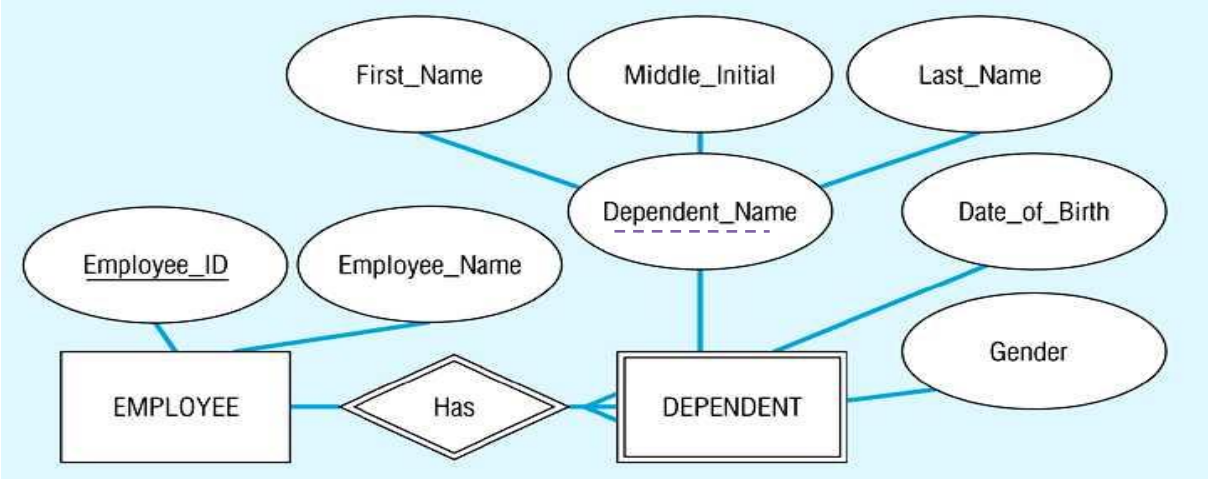
مثال : موجودیت Employee با صفت چند مقداری Skill



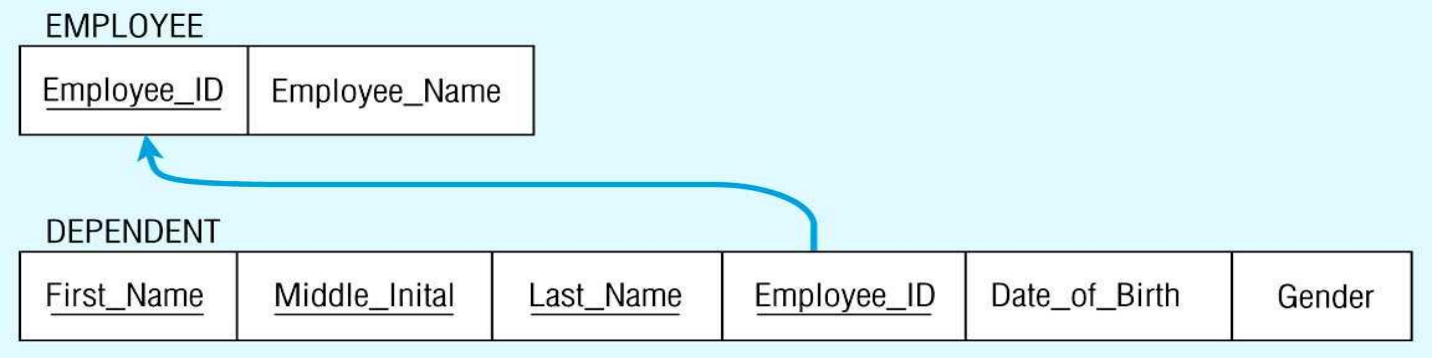
قاعده ۲. هر موجودیت **ضعیف** توسط یک جدول با همان صفات مورد نظر به همراه کلید اصلی موجودیت قوی (کلید خارجی این جدول) نمایش داده می‌شود.

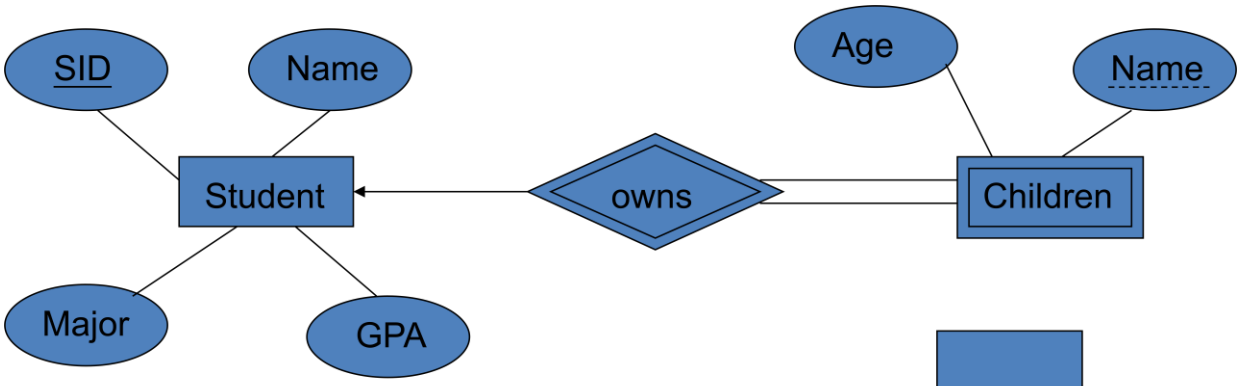
- کلید اصلی جدول جدید، ترکیبی از شناسه موجودیت ضعیف و کلید اصلی موجودیت قوی مورد نظر خواهد بود.

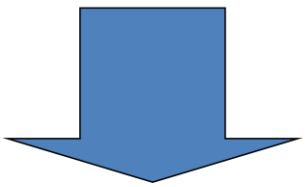
مثال : موجودیت ضعیف DEPENDENT در نمودار زیر را در نظر بگیرید:



شمای رابطه به صورت زیر است.





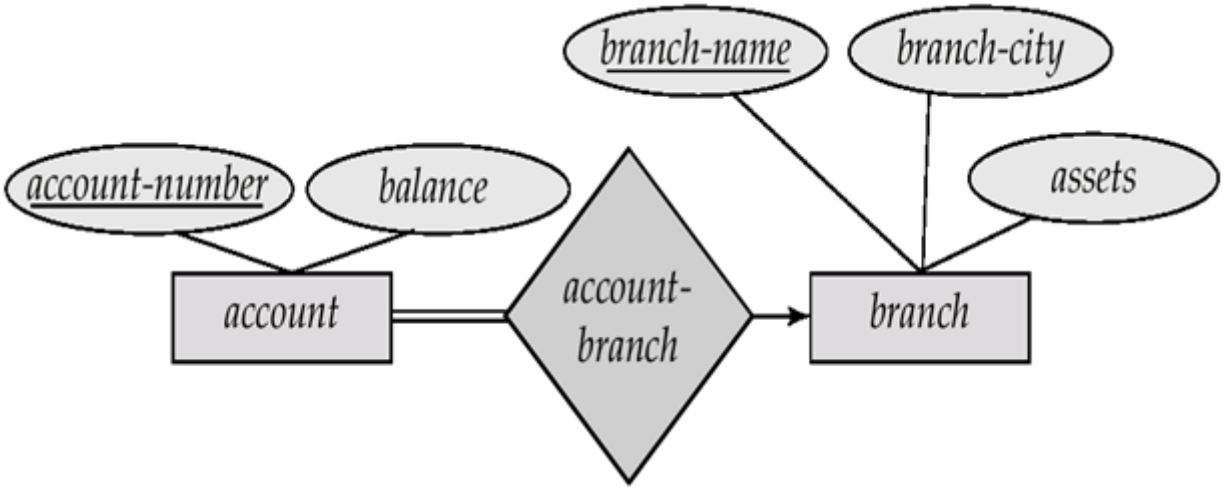


Age	Name	Parent_SID
10	Bart	1234
8	Lisa	5678

قاعده ۳ : ارتباطات درجه ۲.

- هر ارتباط درجه دو چند به چند با یک جدول جداگانه نمایش داده می شود که ستونهایش ترکیبی از کلیدهای اصلی دو موجودیت مشارکت یافته در ارتباط به همراه صفات دیگر مجموعه ارتباط هستند.
- درارتباطات چند به یک یا یک به چند، جدول جداگانه ایجاد نمی شود بلکه کلید اصلی طرف یک به عنوان کلید خارجی طرف چند در نظر گرفته می‌شود. البته بایستی توجه داشت که کلید خارجی اضافه شده بخشی از کلید اصلی نخواهد بود.

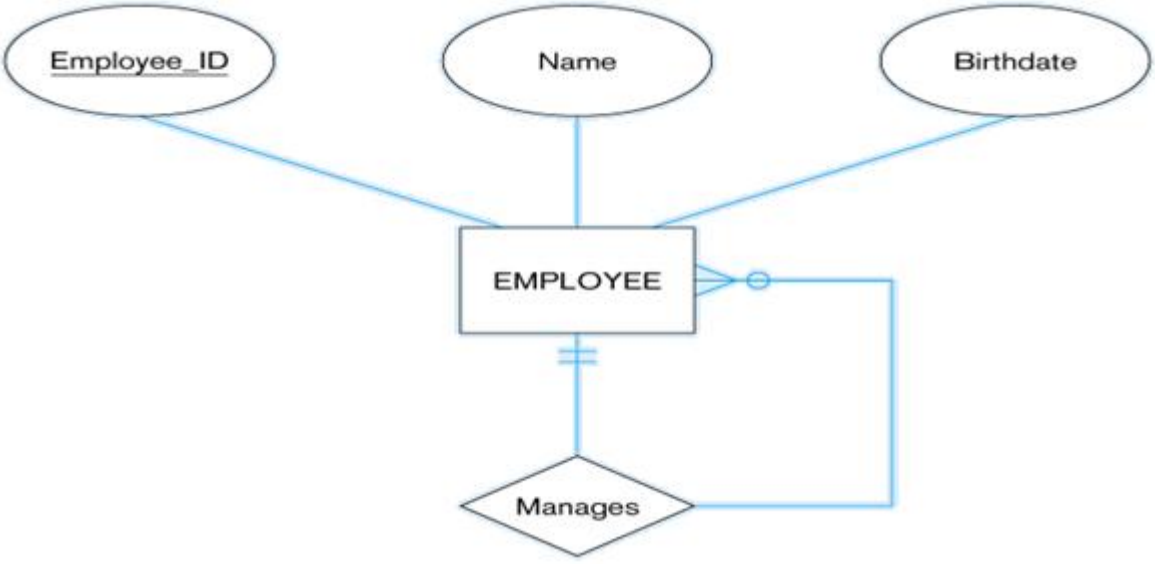
مثال، در نمودار زیر، که بیان گر یک ارتباط یک به چند بین دو موجودیت account وbranch است بجای ایجاد یک جدول برای ارتباط account-branch صفت branchname که کلید اصلی موجودیت branch است به جدول account اضافه می‌شود.



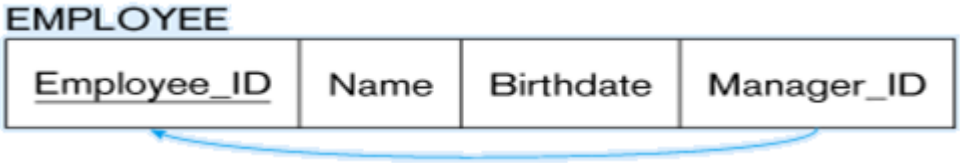
در ارتباطات یک به یک ، هر طرفی می‌تواند به عنوان چند انتخاب شود. بدین معنی که کلید اصلی طرف مهم‌تر به عنوان کلید خارجی طرف کم اهمیت‌تر اضافه می‌شود.

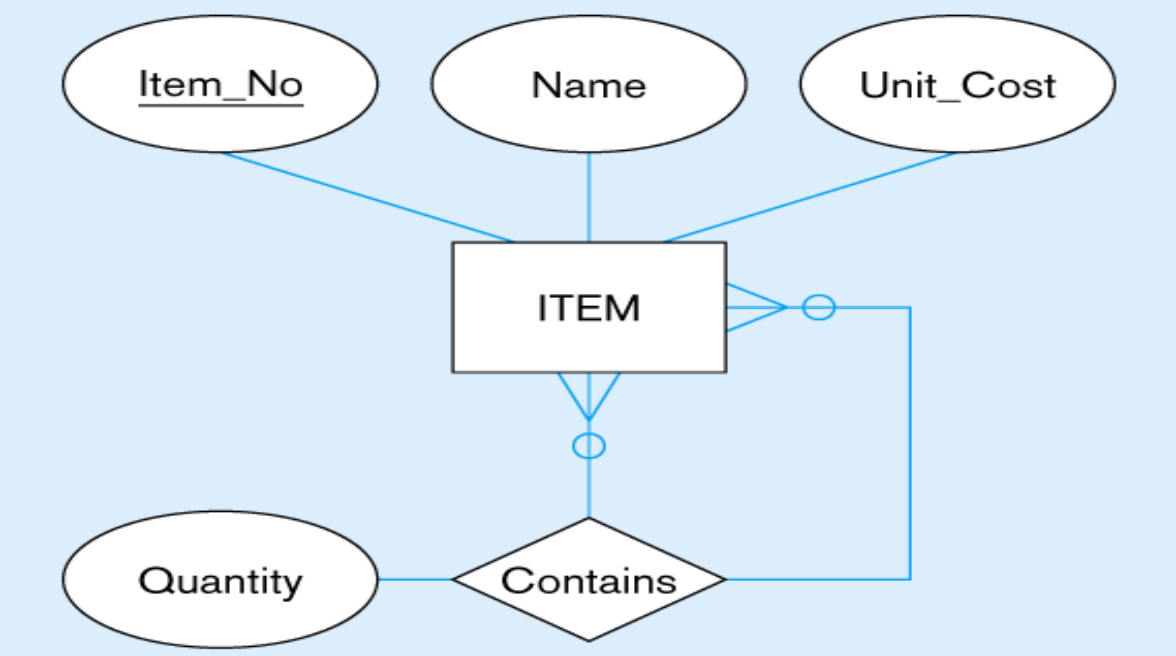
قاعده ٤. ارتباطات درجه یک(بازگشتی).

- هر ارتباط درجه یک چند به چند با یک جدول مجزا نمایش داده می‌شود که کلید اصلی‌اش ترکیبی از دو صفت خاصه برگرفته از کلید اصلی موجودیت است که با توجه به اینکه امکان صفات تکراری در رابطه وجود ندارد یکی از دو کلید تغییر نام پیدا می‌کند.
- در ارتباط بازگشتی یک به چند، کلید خارجی بازگشتی در همان جدول مربوط به موجودیت (با تغییر نام کلید اصلی) داریم.

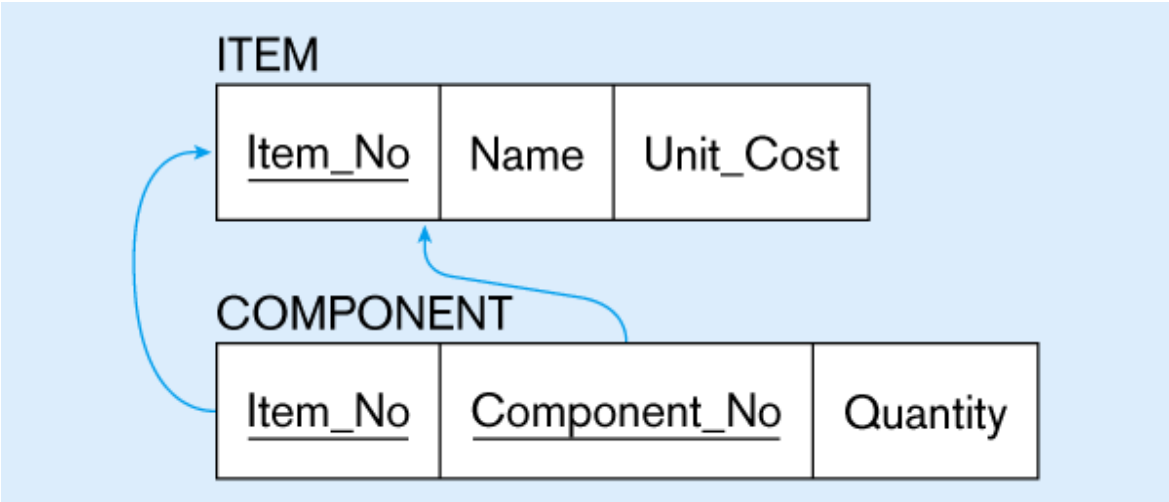


جدول **EMPLOYEE** با کلید خارجی بازگشتی



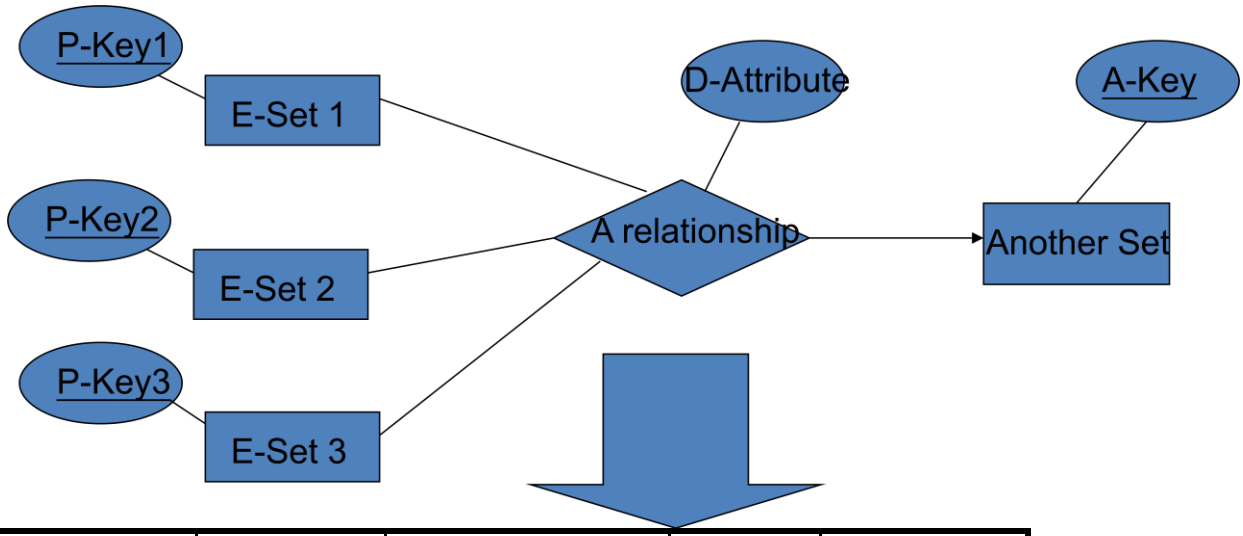


ب) دوجدول بدست آمده



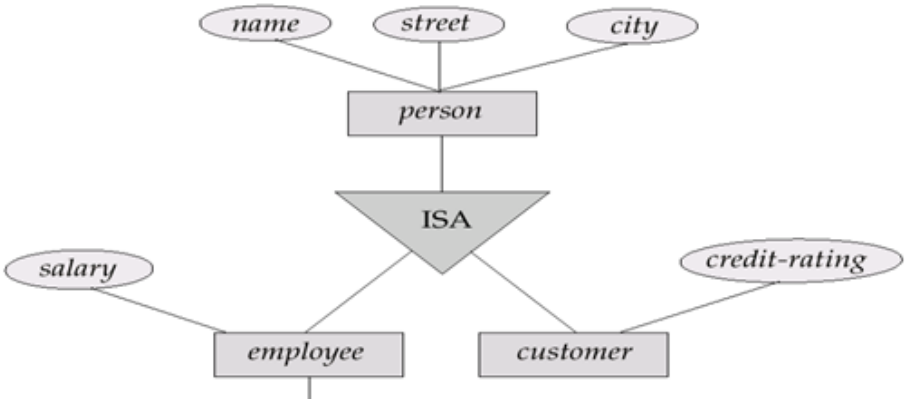
قاعده۵. ارتباطات درجه ۳ و بیشتر.

هر ارتباط nتایی با یک جدول نمایش داده می‌شود که کلید اصلی‌اش n فیلد برگرفته از کلید اصلی موجودیت‌های مشارکت یافته در ارتباط n تایی می‌باشد.



P-Key1	P-Key2	P-Key3	A-Key	D-Attribute
9999	8888	7777	6666	Yes
1234	5678	9012	3456	No

قاعده ٦. ارتباطات تعمیم/تخصیص.



دو روش در این حالت مطرح است:

- روش اول، به ازای 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 اصلی قرار می گیرد.

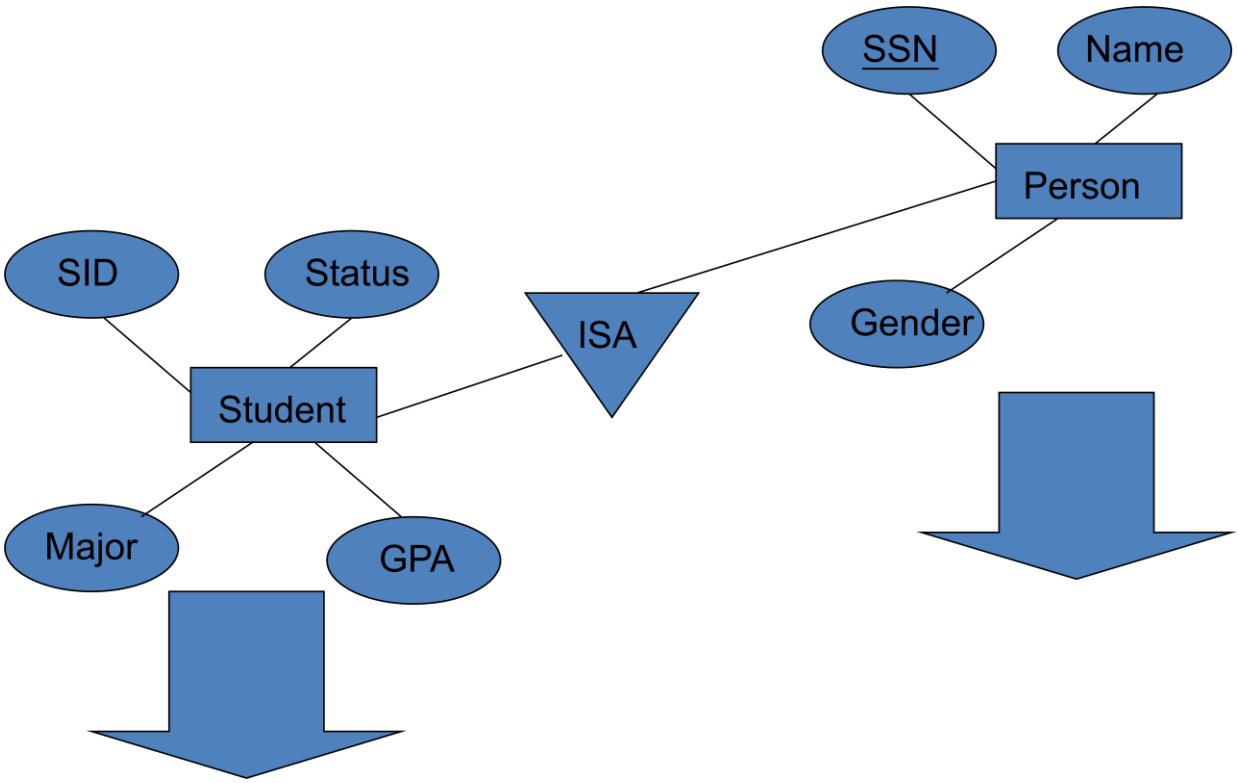
person (name, street, city)

customer (name, credit-rating)
employee(name, salary)

- نکته : اشکال این روش، جمع آوری اطلاعات مستلزم مراجعه به دو جدول است .
- روش دوم : به ازای هر موجودیت صفات کلید اصلی و غیره موجودیت سطح بالاتر در موجودیت سطح پایین‌تر قرار می‌گیرد. 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



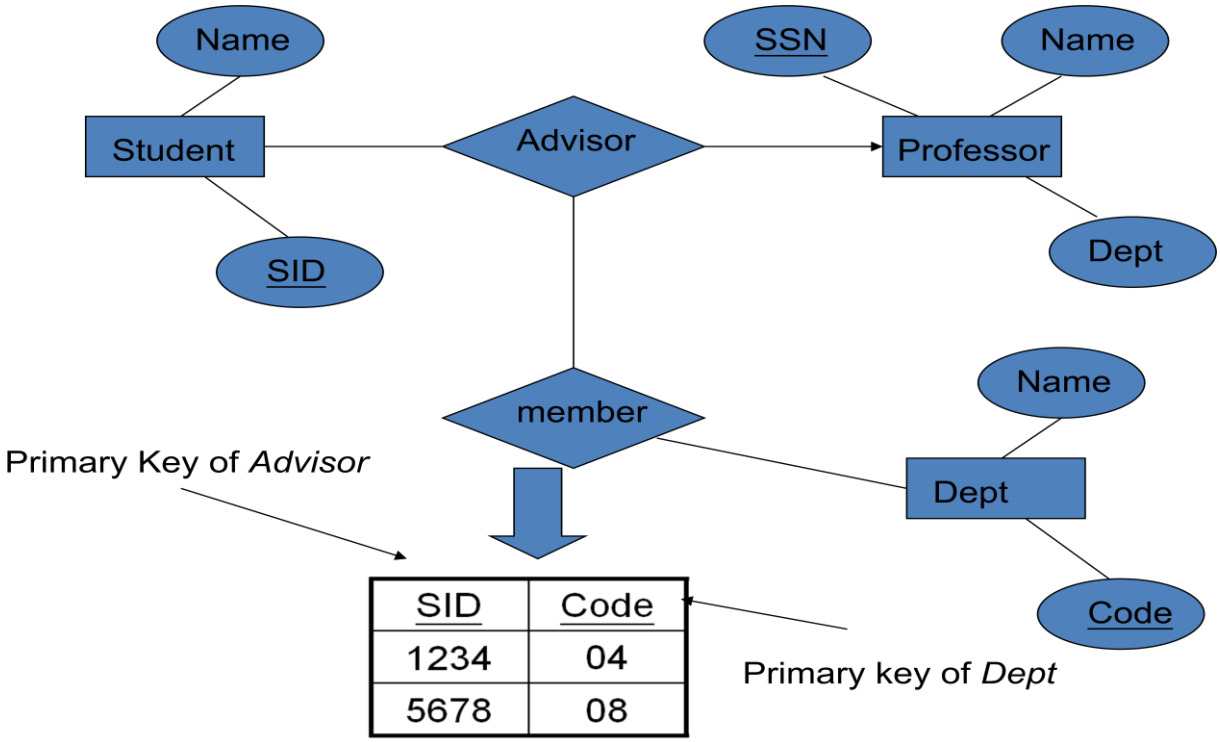
SSN	SID	Status	Major	GPA
1234	9999	Full	CS	2.8
5678	8888	Part	EE	3.6

SSN	Name	Gender
1234	Homer	Male
5678	Marge	Female

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

مثال : رابطه تجمع manages بین ارتباطات manager , Works-on مطرح شده در فصل ۲ به صورت جدول زیر است :

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

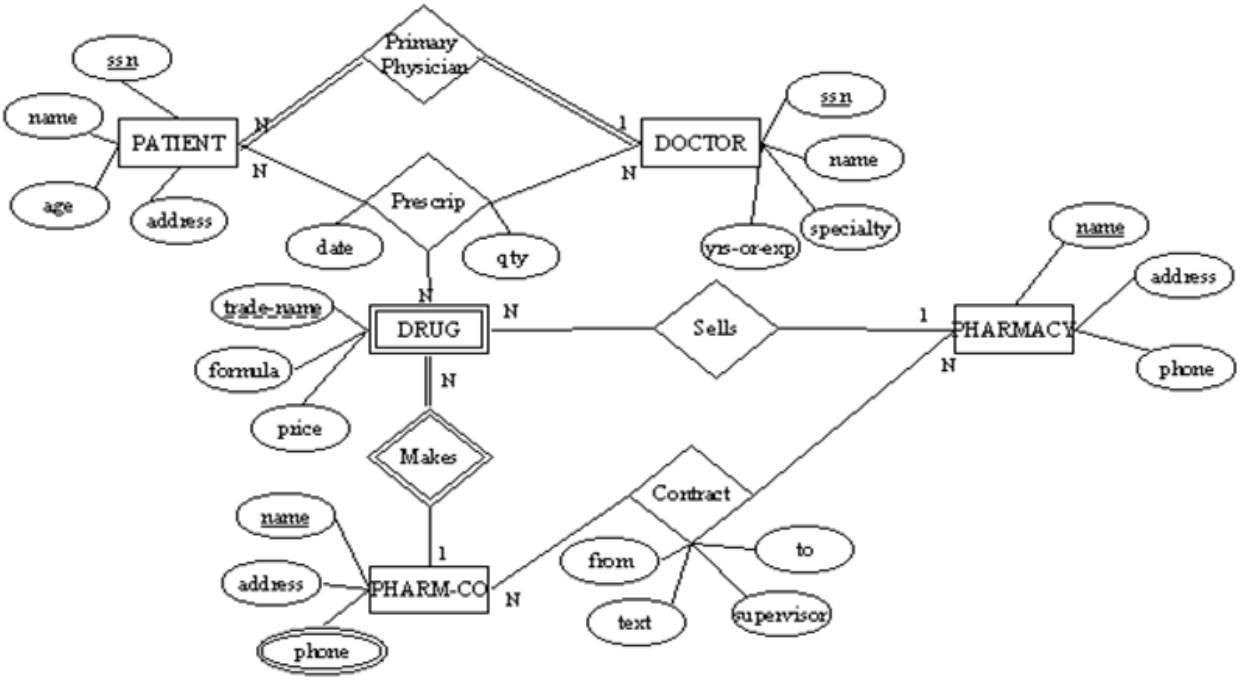


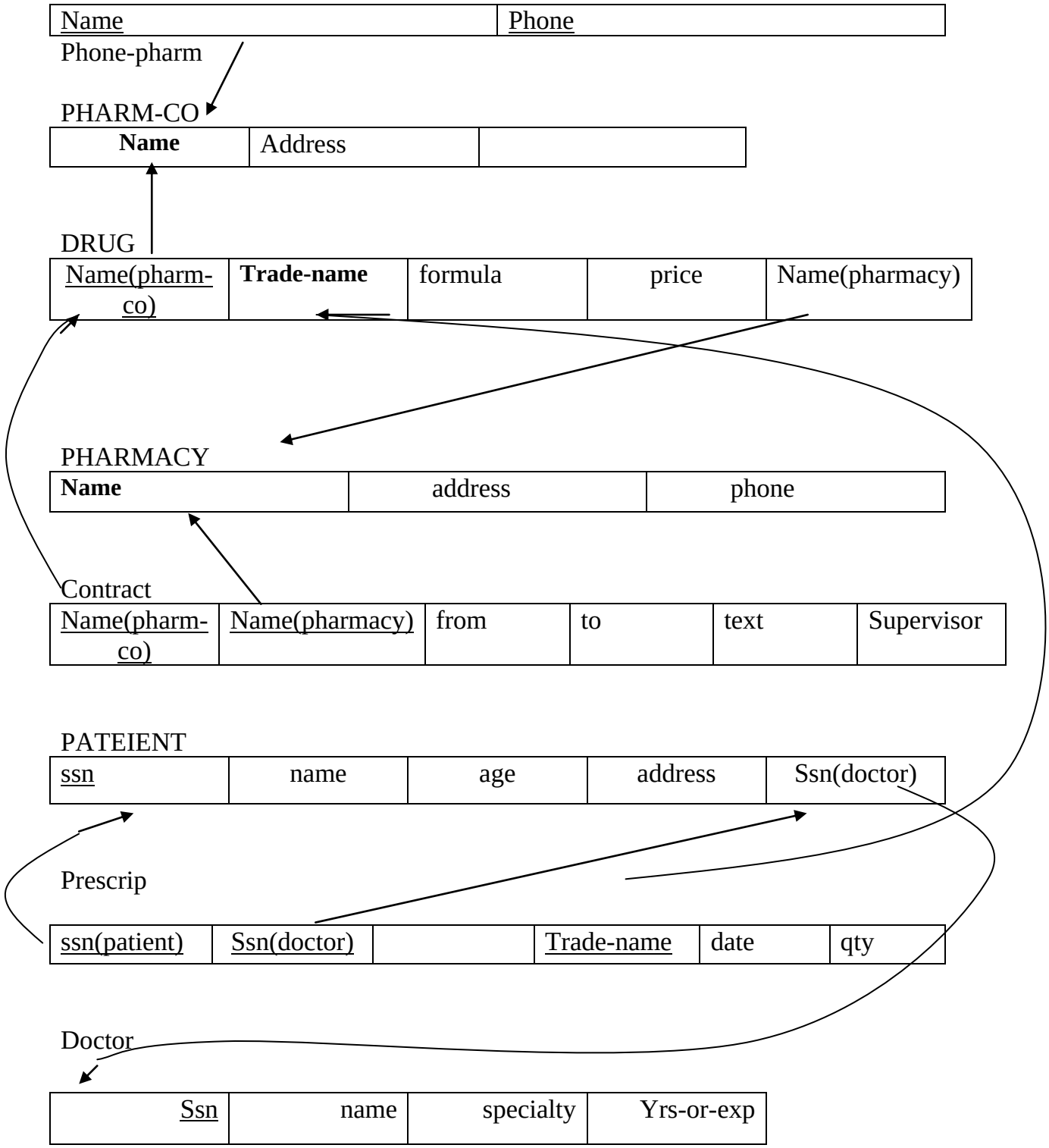
SID	Code
1234	04
5678	08

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

داروخانـه : PHARMACY دارو : DRUG پزشـک : DOCTOR بیمـار : PATIENT

شرکت دارو سازی : PHARM-CO





۳-۶. قوانین جامعیت در سیستم‌های رابطه‌ای

جامعیت در پایگاه داده به معنی، صحت، دقت و سازگاری داده‌های ذخیره شده در پایگاه در تمام لحظات است. هر سیستم مدیریت پایگاه داده‌ها باید بتواند جامعیت پایگاه داده‌ها را کنترل و تضمین نماید. برای کنترل و تضمین جامعیت, نیاز به مجموعه‌ای از قواعد و محدودیت‌هاست که در یک محیط عملیاتی خاص

روی داده‌های همان محیط باید اعمال شوند. این قوانین بنام قوانین جامعیتی خوانده می‌شوند. این قوانین به طور کلی به دو دسته تقسیم می‌شوند.

الف ـ قواعد جامعیت خاص یک محیط (کاربری)

قواعدی هستند که توسط یک کاربر مجاز تعریف می‌شوند. 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 می‌گویند.

به طور مثال، در بانک اطلاعات تهیه کنندگان و قطعات می‌توان قاعده‌ای را به صورت زیر تعریف کرد:

مقدار تعداد(Qty) همیشه بین صفر و ۱۰۰۰۰ باشد. وجود چنین قاعده‌ای در بانک ایجاب می‌کند که سیستم از انجام هرگونه عملی که سبب می‌شود مقدار Qty خارج از طیف مقادیر بشود ، جلوگیری کند.

در سیستم‌های موجود برای معرفی این قواعد از مکانیسم اظهار[1] استفاده می‌شود. اظهار مجموعه‌ای از شرط یا شرایطی است که‌همیشه باید درروی‌عملیات پایگاه رعایت شوند. دستور تعریف اظهار به صورت زیر می‌باشد. در بخش SQL شرح کامل‌تری از این دستور آورده می‌شود.

CREATE ASSERTION <assertion name > CHECK < Predicate>

ب ـ قواعد جامعیت عام

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

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

۱ـ قاعده جامعیت موجودیتی[2] (قاعده C1).

۲ ـ قاعده جامعیت ارجاعی[3] (قاعده C2)

۳ ـ ٦ ـ ۱. قاعده جامعیت موجودیتی: C1

این قاعده که ناظر بر کلید اصلی است می‌گوید، هیچ جز تشکیل دهنده کلید اصلی نباید دارای " مقدار هیچ (NuLL) باشد. این قاعده از طریق معرفی کلید اصلی در تعریف رابطه بر سیستم اعمال می‌شود.

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

هیچ مقدارمی تواند بیان یکی از موارد زیر باشد.

۱ـ مقدار ناشناخته: مثل کارمندی که کد اداره‌اش مشخص نیست و یا خانه‌ای در کوچه‌ای ساخته شد و هنوز پلاک شهرداری ندارد.

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

---

[1] Assertion
[2] Entity Integrity Rule
[3] Referential Integrity Rule

دلیل قاعده جامعیت موجودیتی. با توجه به اینکه 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.

۳-۶-۲. 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 می‌تواند هیچ مقدار داشته باشد در غیر این صورت، حتماً باید مقداری داشته باشد که در رابطه‌ای که در آنجا کلید اصلی است آن مقدار موجود باشد.

به رابطه R_2 رابطه ارجاع دهنده یا رجوع کننده[1] و به رابطه دیگر رابطه مرجع[2] یا مقصد گویند. این قاعده از طریق حکم کلید خارجی و تعریف رابطه به سیستم اعلام می‌شود.

مثال، در رابطه Owner فیلدهای CarNo, DriverNo کلید خارجی رابطه هستند. مقادیر این فیلدها باید چنان باشند که حتماً در هر کدام از رابطه هایCar, Driver وجود داشته باشند.

نکته. در این مثال خاص DriverNo و CarNo در رابطهOwner نمی توانند مقدار Null داشته باشند زیرا هرکدام جزئی از کلید اصلی هستند.

دلیل توجیه کننده قاعده جامعیت ارجاعی. نمی توان به نمونه موجودیتی ارجاع داد که در جهان واقعی وجود ندارد.

۳ـ۶-۳. عوامل نقض جامعیت

در یک پایگاه داده، عوامل مختلفی ممکن لست باعث نقض جامعیت گردند. از جمله این عوامل می توان موارد زیر را نام برد:

- اشتباه در وارد کردن داده ها
- اشتباه در برنامه های کاربردی
- وجود افزونگی کنترل نشده
- خرابی در نرم افزار یا سخت افزار
- اشتباه در یک تراکنش

این عوامل باعث ایجاد شرایطی می شوند که در نهایت منجر به نقض جامعیت یا اعمال نادرست و یا عدم اعمال محدودیت های جامعیتی می شود.

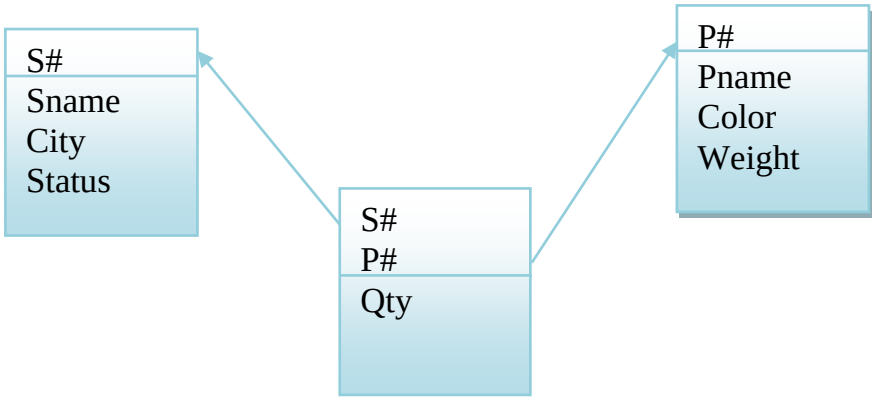
---

[1] Referencing
[2] Reference

۳ـ۶-۴. تبعات قواعد جامعیت

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

بانک اطلاعاتی قطعات و تهیه کنندگان را در نظر بگیرید. در این بانک سه جدول به نام های S برای تهیه کنندگان، P برای قطعات و SP تهیه می کند به شکل زیر وجود دارند.



فرض کنید دستورحذف سطری از جدول S با مقدار "S#='S_2" در خواست شده باشد.

```
Delete From S
Where  S# = 'S2'
```

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

ارجاع SP به S مشکل خواهد داشت زیرا S_2 وجود ندارد. برای حفظ قاعده جامعیت ارجاعی چهار روش مشهور است.

Sمرجع

S#
S_1
S_2
S_3
S_4

SP رجوع کننده

S#	P#	Qty
S_1	P_1	100
S_2	P_1	70
S_1	P_3	200
S_4	P_2	60

الف ـ روش حذف تعویقی(محدود شده[1])

در این روش، اگر درخواست حذف تاپلی از رابطه مرجع شود تا زمانی که تاپلهایی در رابطه رجوع کننده وجود دارند که به این تاپل ارجاع می‌دهند عمل حذف انجام نمی‌شود.

[1] Restricted

ب ـ روش آبشاری یا تسلسلی [1]

در این روش با حذف یک تاپل از رابطه مرجع، تمام تاپلهای رجوع کننده به آن تاپل نیز حذف می‌شوند.
به طور مثال اگر درخواست حذف سطری از جدول S با مقدار "S#='S_2" در خواست شده باشد. باید پس از اجرای آن حکم حذف سطری از جدول SP با مقدار"S#='S_2" نیز باید اجرا شود.

```
Delete FROM S
Where S#='S2'
```

↓

```
Delete FROM  SP
Where S#='S2'
```

این حکم ممکن است به طور اتوماتیک توسط DBMS 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 نمی‌باشد.

ج ـ روش هیچ مقدار گذاری [2]

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

مثال:

```
Delete FROM S
Where S#='S2'
```

↓

```
UPDATE SP
SET S#= NULL
Where S#='S2'
```

توجه. در این مثال نمی توان روش هیچ مقدارگذاری را انجام داد زیرا S# در SP جز کلید اصلی است.

د ـ روش مقدار گذاری با مقدار پیش فرض [3]

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

---

[1] Cascaded
[2] Nullified
[3] SET TO DEFAULT

۳ـ۶ـ۴. راه‌های اعمال قواعد جامعیت:

به طور کلی می توان راه‌های اعمال قواعد جامعیت را به صورت زیر نام برد:

۱ -معرفی کلید اصلی

۲ -اعلام صفت هیچ مقدار ناپذیر

۳ -معرفی کلید خارجی

۴ -اعلان محدودیت‌های مورد نظر در شمای پایگاه داده

۵ - نوشتن راه انداز[1]

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 تا مین گردد. مثلاً در پی انجام یک عمل به هنگام سازی در تاپلی از یک رابطه , تاپل(هایی) از رابطه‌های دیگر نیز به هنگام در می‌آیند.

۶ -معرفی میدان و مقادیر آن

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

۳-۷. مشخصات سیستم‌های رابطه‌ای

درجه یا میزان یا حد رابطه بودن سیستم‌های رابطه‌ای متفاوت است. هر مدل رابطه‌ای حداقل باید دارای سه جنبه باشد. این سه جنبه عبارتند از: ساختار داده‌ای رابطه‌ای، امکانات کار با داده‌ها و قواعد جامعیت. ساختار داده‌ای که همان رابطه‌ها هستند دو امکانات کار با داده‌ها شامل مجموعه‌ای از عمل گر هاست که در کادر مفاهیم ساختار داده‌ای عمل می‌کنند و به گونه هستند که روی رابطه‌ها عمل می‌کنند. از نظر CODD سیستمی در حداقل رابطه‌ای است. اگر دارای دو جنبه زیر باشد:

۱ـ پایگاه داده‌های مبتنی بر رابطه‌ها یا پایگاه جدولی را برای کاربر تا مین کند.

۲ـ سه عمل اساسی از عملیات روی رابطه‌ها را داشته باشد. این سه عمل عبارتند از: عمل گزینش[2]، عمل پرتو[3] و عمل پیوند[4] . اما یک سیستم کاملاً رابطه‌ای باید خصوصیات زیر را داشته باشد:

۱ـ پایگاه رابطه‌ای را ایجاد کند: امکانات کامل تعریف انواع رابطه‌ها را داشته باشد.

۲ـ امکانات کامل کار با رابطه‌ها

۳ـ امکان تعریف مجموعه کاملی از قوانین جامعیت به شرح دیده شده را داشته باشد.

---

[1] Trigger
[2] Selection
[3] Project
[4] Join

تمرین‌های این فصل.

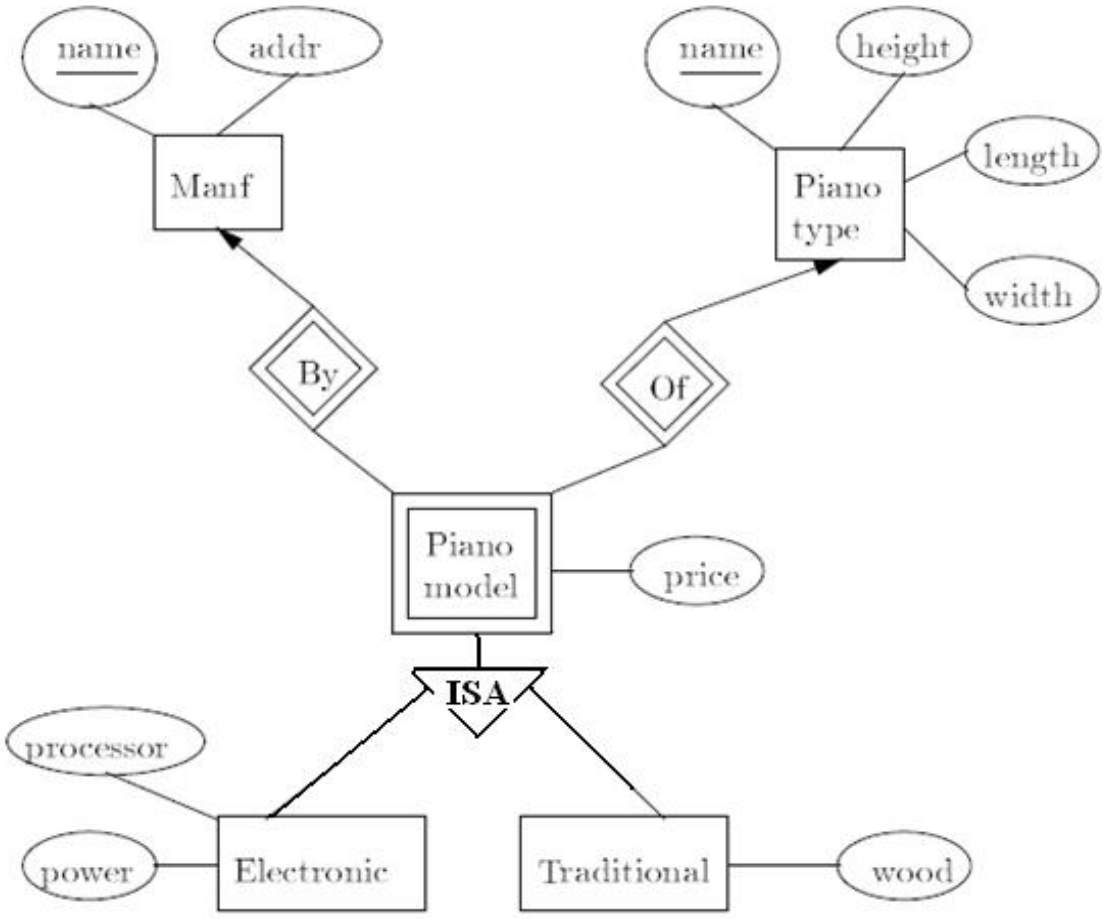
۱ -نحوه تعریف انواع کلید را در یک سیستم رابطه‌ای بررسی نمایید.

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

۳ -هر یک از اصطلاحات زیر را تعریف کنید :

شمای رابطه(Relation Schema) ،شمای پایگاه داده رابطه‌ای(Relational Database Schema) و دامنه(Domain)

۴ -در زیر نمودار E/R درباره پیانوها، کارخانه سازنده و انواع آن دیده می‌شود. به عنوان نمونه کارخانه سازنده می‌تواند Casio, Steinway باشد. اسامی انواع پیانوها عبارتند از : Grand, BabyGrand,Spinet و Keyboard (Electronic) . پیانوها می‌توانند الکترونیکی یا سنتی باشند و برخی جزء هر دو مدل. برای پیانوهای سنتی ما جنس چوبی (Wood) که از آن ساخته شده است را نگهداری می‌کنیم و برای پیانوهای الکترونیکی processor , power.



الف: آیا با توجه به نمودار مطرح شده می‌توان این واقعیت را بیان کرد که Steinway دو مدل مختلف از پیانوهای BabyGrand را می‌سازد. اگر بله توضیح دهید چرا و اگر نه توضیح دهید با چه تغییری در نمودار می‌توان این مفهوم را بیان کرد؟

ب) نمودار Schema ی پایگاه داده را ترسیم کنید. (پس از پاسخ به قسمت الف).

۵ -اگر جدولی دارای n ستون باشد و فقط یک کلید کاندید تک صفتی داشته باشد، تعداد فوق کلیدهای (superkeys) این جدول چند تاست؟

۶ - پایگاه داده زیر را که توسط یک ناشر مورد استفاده قرار می‌گیرد در نظر بگیرید :

فیلد **Position** بیانگر ترتیب نویسنده، ویرایشگر و کلمه کلیدی در کتاب است. به عنوان مثال در کتاب درسی این درس ترتیب به صورت : **Silberschatz, Korth, Sudershan** است. که **Silberschatz** در موقعیت ۱ و **Korth** در موقعیت ۲ و **Sudershan** در موقعیت ۳ می‌باشد. هر نسخه جدید کتاب دارای **ISBN** جدید بوده و لذا **Edition** بخشی از کلید اصلی نیست.

Person (pno, name, street, city, tel_no, postal_code)

Book (isbn, title, date, edition, book_type)

book_author (isbn, pno, position)

book_editor (isbn, pno, position)

book_keywords (isbn, kwd_no, position)

keyword (kwd_no, keyword)

شمای پایگاه داده کتاب را به نمودار E/R معادل با آن تبدیل کنید . تمامی تناظر رابطه‌ها را مشخص نمایید.

فصل چهارم :

عملیات روی رابطه‌ها

۴-۱. مقدمه

کاربران در یک محیط سیستم پایگاه داده برای درخواست اطلاعات مورد نیازشان نیاز به زبان‌های پرس و جو[۱] دارند. در یک تقسیم بندی این زبان‌ها را به دو زبان رویه‌ای یا روشمند[۲] و نا روشمند[۳] تقسیم کرده‌اند. در زبا ن‌های روشمند، برنامه ساز نه تنها به سیستم می‌گوید چه می‌خواهد بلکه رویه (نحوه بدست آوردن آنچه را که می‌خواهد) را نیز بیان می‌کند. از جمله زبان‌های روشمند جبر رابطه‌ای است که در ادامه در باره آن آورده می شود. به طور کلی، 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

جبر رابطه ای که قوی ترین مبنای تئوریک مدل رابطه ای است از دو بخش تشکیل شده است :

الف) نوع داده که در اینجا رابطه است.

ب) عملگرها

آقای کاد در مقاله خود هشت عملگر برای کار با رابطه ها تعریف کرده است که این عملگرها به دو دسته تقسیم می شوند: عملگرهای متعارف در مجموعه ها نظیر: اجتماع[۴] ، اشتراک[۵] ، تفاضل[۶] و ضرب دکارتی[۷] و عملگرهای خاص نظیر: گزینش[۸]، تصویر یا پرتو[۹]، پیوند[۱۰] و تقسیم[۱۱].

---

[1] Query Language
[2] Procedural
[3] Non procedural
[4] Union
[5] Intersect
[6] Minus (difference)
[7] Cartesian Product
[8] Selection
[9] Project
[10] Join
[11] Divide

٤-٢-١. عملگر گزینش یا تحدید (Select)

این عملگر تاپلهایی(سطرهایی) را از یک رابطه گزینش می کند. به 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. نمادی که برای گزینش به کارمی رود به صورت $\sigma_p(r)$ است که در آن، P شرط گزینش بوده و r رابطه (جدول) است.

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

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

<attribute>op<atrribute>or< constant> که op یکی از $=, \neq, >, <, \geq, \leq$ است.

عملکرد گزینش را در شکل زیر می بینید:

A	B	C	D
α	α	1	7
α	β	5	17
β	β	12	10

Relation r

A	B	C	D
α	α	1	7
β	β	23	10

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

٤-٢-٢ 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 $\pi_{A_1,A_2,\ldots A_K}(r)$ که در آن $A_1, A_2 \ldots$ اسامی صفات خاصه و r نام رابطه است. نتیجه این عملگربه صورت 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
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

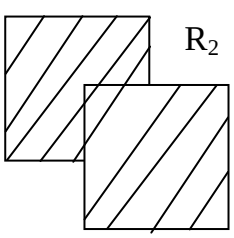
=

A	C
α	1
β	1
β	2

٤-٢-٣ عملگر اجتماع

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

$R = R_1 \text{ union } R_2$



•نماد اجتماع به صورت $r \cup s$ تعریف می شود که در آن $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$ است. $r \cup s$ موقعی معتبر است که :

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

۲ـ حوزه (میدان) صفات خاصه دو مجموعه بایستی سازگار باشند. به عنوان مثال صفت خاصه i ام از هر دو رابطه نوع یکسان داشته باشند.

مثال:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

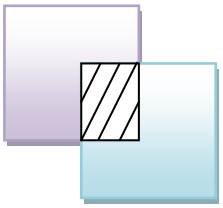
s

$r \cup s =$

A	B
α	1
α	2
β	1
β	3

٤-٢-٤ اشتراک دو رابطه

اشتراک دو رابطه، رابطه ایست که تاپلهایش در هر دو رابطه وجود داشته باشند. نماد $r \cap s$ $R = R1 \cap R2$ به صورت **$r \cap s = \{t \mid t \in r \text{ and } t \in s\}$ است که در آن**، s و r باید از نظر تعدادصفات خاصه برابر و صفات خاصه متناظر از یک میدان برگرفته باشند.



A	B
α	1
α	2
β	1

r1

A	B
α	1
β	3

r2

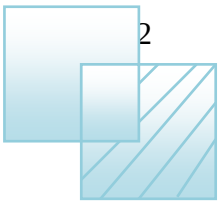
A	B
α	1

r1 ∩ r2

٤-٢-٥ عملگر تفاضل:

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

نماد منها به صورت $r - s = \{t \mid t \in r \text{ and } t \notin s\}$ است. 2



A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
β	1

r – s

٤-٢-٦ حاصلضرب کارتزین :

حاصل ضرب دکارتی دو جدول، جدولی است که ستون هایش شامل همه ستون های دو جدول و سطر هـایش شامل تمام ترکیب های ممکن سطرهای دو جدول است . برای ضرب دو رابطه از نماد $\mathbf{r \times s}$ استفاده می شود که به صورت زیر تعریف می گردد:

- $\mathbf{r \times s = \{(t,q) \mid t \in r \quad and \quad q \in s\}}$

فرض می شود صفات خاصه r(R) و s(S) مجزا باشند به عبارتی : $R \cap S = \varnothing$، اگر صفات خاصـه r(R) و s(S) مجزا نباشند بایستی تغییر نام صورت پذیرد. معمولا برای این کار از کلمه توصیفی نام جدول بـه همـراه نقطـه استفاده می شود.

- مثال :

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
γ	10	b

S

r × s :

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	γ	10	b

٤-٢-٧ ترکیب طبیعی (پیوند) [1]

این عملگر، زیر مجموعه ای از عملگر ضرب دکارتی است که شرط تساوی روی سطرهای آن اعمال شده باشـد. به عبارتی، حاصل رابطه ای است که تاپلهای آن از پیوند (ترکیب) تاپلهایی از دو رابطه با شرط تسـاوی مقـادیر یک یا بیش از یک صفت خاصه مشترک بین دو رابطه به دست می آید.

نکته ۱. عملگر پیوند فقط سطرهایی از جدول ها را کنارهم قرار می دهد که تمام ستون های هم نام آن دو جدول مقادیر مساوی داشته باشند.

نکته ۲. ستون های هم نام در دو رابطه فقط یکبار در خروجی عمل پیوند طبیعی ظاهر می شوند.

برای این عملگر از نماد $r \infty s$ استفاده می شود. به عنوان مثال در نظر بگیرید r رابطه بـا شـمای R و s رابطـه ای با شمای S باشد. نتیجه ترکیب (پیوند)دو رابطه، رابطه ای در شمای $R \cup S$ است که با در نظر گرفتن هر زوج تاپل t_r از r و t_s از s بدست می آید. اگر t_r وt_s دارای مقادیر یکسان در هر یک از صـفات خاصـه $R \cap S$ باشند یک تاپل t به نتیجه اضافه می شود به طوری که : t همان مقادیر t_r در r و t_s در s رادارد.

[1] Natural join

اگر داشته باشیم R = (A,B,C,D) و S = (E,B,D) در این صورت شمای نتیجه به صورت (A,B,C,D,E) خواهد بود و r ∞ s بصورت زیر تعریف می شود:

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

مثال :

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

r ∞ s

- **نکته۱**: پیوند طبیعی، خاصیت شرکت‌پذیری و جابه‌جایی دارد.
- **نکته۲**: اگر دو جدول فیلد همنام نداشته باشند، در این صورت R1 ∞ R2 تبدیل به R1×R2 می‌شود.
- **نکته۳**: اگر تمام فیلدهای دو جدول یکسان باشند، R1 ∞ R2 معادل R1 ∩ R2 خواهد بود.
- **نکته۴**: پیوند طبیعی عملگر گرانی نیست و زمان و فضای کمتری نسبت به ضرب دکارتی می‌گیرد.

٤-۲-۸ عملگر تقسیم.

عمگر تقسیم کاربرد ارزشمندی دارد ولی درزبانهای مرسوم پایگاه داده به طور مستقیم پیاده سازی نشده است. کاربرد این عملگر در مواقعی است که بخواهیم از واژه تمام حالت های یک موضوع استفاده کنیم. به عنوان مثال اسامی دانشجویانی که تمام دروسی را که باید اخذ کنند را اخذ نموده اند.

فرض کنید دورابطه یکی از درجه m + n و دیگری از درجه n که (ستون هایش زیر مجموعه ای از رابطه اولی است) را داشته باشیم در عمل تقسیم، حاصل رابطه ای با درجه mخواهد بود که حاوی مقادیری از صفات خاصه رابطه درجه m + n است که مقادیر صفت خاصه دیگر به تمامی در رابطه درجه n وجود داشته باشند. نماد این عملگر به صورت ÷ بوده و برای پرس و جوهایی که عبارت برای تمام (for all) را دارند مناسب است.

فرض کنید r و s رابطه هایی روی شماهای R و S باشند و $R = (A_1,\ldots,A_m,B_1,\ldots,B_n)$ و $S = (B_1,\ldots,B_n)$

نتیجه تقسیم رابطه r بر s رابطه ایست در شمای R – S: $R - S = (A_1,\ldots,A_m)$ و تعریف آن به صورت زیر است :

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

مثال:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
β	2

r

B
1
2

s

A
α
β

r ÷ s

A	B	C	D	E
α	A	α	a	1
α	A	γ	a	1
α	A	γ	b	1
β	A	γ	a	1
β	A	γ	b	3
γ	A	γ	a	1
γ	A	γ	b	1
γ	A	β	b	1

r

D	E
a	1
b	1

s

A	B	C
α	a	γ
γ	a	γ

r ÷ s

برای بدست آوردن تقسیم دو رابطه می توان از عبارت معادل آن به صورت زیر استفاده کرد:

$$r \div s = \prod_{R\text{-}S}(r) - \prod_{R\text{-}S}((\prod_{R\text{-}S}(r) \times s) - r)$$

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

٤-۳-۱. عملگر تغییر نام[1]

این امکان را می دهد که به یک رابطه با بیش از یک اسم رجوع شود. به عبارتی اگر در پرسشی به یک جدول دو یا بیشتر نیاز باشد بدون ذخیره سازی مجدد جدول می توان با نام گذاری مجدد به دفعات از آن استفاده کرد. در جبر رابطه ای این عملگر به صورت $\rho_x(E)$ نوشته می شود که عبارت E را تحت نام X بر می گرداند. اگر یک عبارت جبررابطه ای E ، n ستون داشته باشد $\rho_{x(A1, \dots, An)}(E)$ 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.

٤-۳-۲. عملگر پرتو تعمیم یافته[2]

عملگری است که برای گسترش عنوان یک رابطه بکار می رود که درآن امکان به کارگیری عملیات ریاضی روی ستون های رابطه فراهم می شود. این عمگر را به صورت $\prod_{F_1, F_2, \dots, F_n}(E)$ نشان می دهند که هر یک از

[1] Rename
[2] Generalized Projection

$F_{1...}F_n$ عملیات ریاضی به کار رفته روی صفات خاصه است. معمولا صفت خاصه اضافه شده مقادیر آن با ارزیابی یک عبارت محاسباتی مشخص بدست می آید.

٤-٣-٣.عملگر های جمعی[1]

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. انواع این عملگر ها عبارتند از:

Min , Max , Avg , Sum , Count

این عملگرها به شکل $\mathcal{G}_{F_1(A_1),F_2(A_2,...,F_n(A_n)}(E)$ تعریف می شوند که در آن E عبارت جبر رابطه ای و $G_1,G_2,...G_n$ لیست ستونهایی است که برروی آن ها گروه بندی صورت می گیرد و می تواند وجود نداشته باشد. و هر یک از $F_1,F_2,...F_n$ همان توابع جمعی هستند که روی ستونهای $A_1..A_n$ تعریف شده اند.

مثال :

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

$g_{\text{sum(c)}}(r)$

sum(c)
27

٤-٣-٤. عملگر انتساب[2]

در صورت طولانی بودن پرسش ها با استفاده از عملگر انتساب می توان بخشی از جدول را در جایی ذخیره نمود و در ادامه کار مورد استفاده قرار داد. نماد این عملگر ← بوده و این عملگر یک روش مناسب برای بیان پرس و جوهای پیچیده است. به عنوان مثال تقسیم را می توان به صورت زیر بیان کرد:

$temp1 \leftarrow \prod_{R-S}(r)$
$temp2 \leftarrow \prod_{R-S}((temp1 \times s) - r)$
$result = temp1 - temp2$

[1] Aggregate Operator
[2] Assignment

٤-٣-٥. عملگر نیم پیوند[1]

این عملگر گونه ای دیگر از پیوند طبیعی است که درآن, تنها تاپلهای پیوند شدنی از رابطه سمت چپ در رابطه جواب وارد می شوند. به عبارت دیگر مشابه پیوند طبیعی است با این تفاوت که فقط ستونهای جدول اول را می‌دهد. این عملگر در پایگاه داده های توزیع شده کاربرد دارد.

$R1 \propto R2 = \prod_{Attribute(R1)} (R1 \infty R2)$

٤-٣-٥. عملگر نیم تفاضل[2]

این عملگر بصورت زیر تعریف شده است :

$R1\ SEMIMINUS\ R2 = R1\ MINUS\ (R1 \propto R2)$

نکته: در عملگر تفاضل، دو عملوند باید هم‌نوع باشند ولی در نیم تفاضل، این مساله لازم نیست.

٤-٣-٦. عملگر فراپیوند (پیوند بیرونی)[3]

نوع دیگری از عملگر پیوند که معمولاً به شرط تساوی بکار میرود. این عملگر توسعه یافته عملگر پیوند است که در آن از دست رفتن اطلاعات جلوگیری می کند. این عملگرها، به سه دسته تقسیم می‌شوند:

- فراپیوند چپ (Left Outer Join) با نماد $\supset\!\bowtie$

مانند پیوند طبیعی است با این تفاوت که علاوه بر تاپل‌های پیوند شدنی از دو رابطه، تاپل‌های پیوند نشدنی از رابطه چپ هم، پیوند شده با مقادیر NULL، در جواب وارد می‌شوند.

- فراپیوند راست (Right Outer Join) با نماد $\bowtie\!\subset$

تاپل‌های پیوند نشدنی از رابطه راست هم، پیوند شده با مقادیر NULL، در جواب وارد می‌شوند.

- فراپیوند کامل (Full Outer Join) با نماد $\supset\!\bowtie\!\subset$: هر دو حالت راست و چپ را شامل می‌شود.

٤-٤-مجموعه کامل عملگرها در جبر رابطه ای:

از میان عملگرهای مطرح شده تاکنون، 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 :

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

٤-٥. برخی خواص عملگرها :

اگر R,S,T رابطه باشند داریم :

1- $R \infty S = S \infty R$
2- $(R \infty S) \infty T = R \infty (S \infty T)$

---

[1] Semi Join
[2] Semi Minus
[3] Outer join

3- $\prod A_1..A_p (\prod A_i..A_j) (R) = \prod A_1..A_p (R)$
4- $б_{Ai='a'} (R) U б_{Ai='a'}(S) = б_{Ai='a'} (RUS)$
5- $б_{Ai='a'} (R) \cap б_{Ai='a'}(S) = б_{Ai='a'} (R \cap S)$
6- $б_{Ai='a'} (R) - б_{Ai='a'}(S) = б_{Ai='a'} (R-S)$
7- $\prod A_1..A_p (R) U \prod A_1..A_p (S) = \prod A_1..A_p (R U S)$

۴-۶. بهینه‌سازی پرس و جو

در جبر رابطه‌ای برای یک سؤال ممکن است چند پاسخ وجود داشته باشد ولی همه انها از نظر فضای مصرفی و زمان معادل نیستند. اینکه کدام روش بهینه‌تر است موضوع مهمی است. در برخی از DBMS ها، بخشی با بهینه‌ساز پرس‌و جو (query optimizer) وجود دارد، که قبل از کامپایل کردن پرس‌و جوها، آن‌ها را بهینه می کند. جهت بهینه‌سازی پرس‌و جوها می‌توان از قواعد زیر استفاده کرد:

۱-گزینش را هرچه ممکن است زودتر انجام دهید. به عبارتی اولویت بیشتری دارد.

۲- شرط‌های ترکیبی را به شرط‌های متوالی تبدیل کنید.

۳- عملگر پرتو را دیرتر از گزینش و زودتر از سایر عملگرها انجام دهید.

۴- از نظر ریاضی، عملگر پیوند طبیعی خاصیت جابه‌جایی و شرکت‌پذیری دارد ولی از نظر کامپیوتری، ممکن است زمان و حافظه مورد نیاز عملیات $A\infty B$ خیلی بیشتر از $B\infty A$ باشد.

۵- در چند عمل پرتو متوالی، فقط پرتو آخری مؤثر است و بقیه قابل حذف هستند.

- $\Pi_{L1}(\Pi_{L2}(\ldots(\Pi_{Ln}(R))\ldots))=\Pi_{L1}(R)$

۴-۷. چند مثال از جبر رابطه ای.

مثال۱. 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 .

$$б_{city = 'تهران'} (J)$$

۱ - اسامی تهیه کنندگان قطعه P2 را بدهید :

$$\prod_{Sname} (б_{P\# = 'P2'}(S \infty SPJ$$

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

$$\prod_{Sname} ((бcolor = 'آبی' P) \infty SPJ) \infty S$$

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

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

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

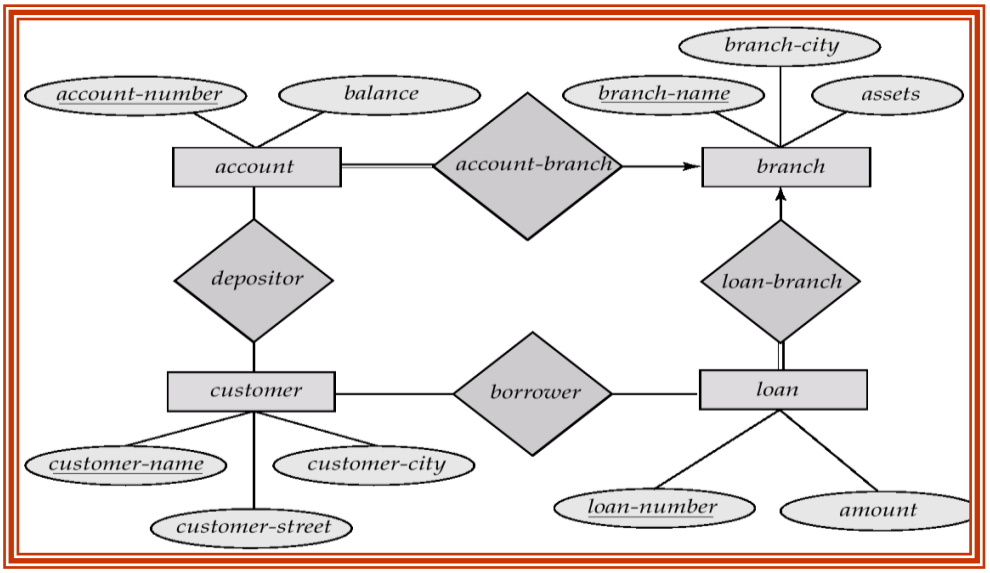
$$\prod_{sname} (\prod_{S\#} (S) - (\prod_{S\#} (б_{P\# = 'P2'}(SPJ)) \infty S$$

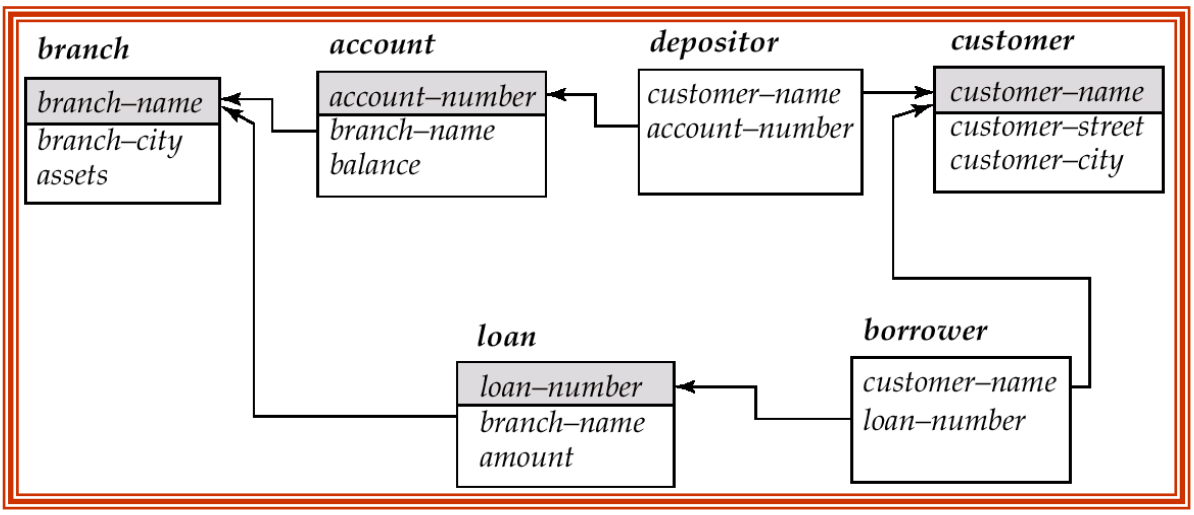
۵ - شماره قطعاتی را مشخص کنید که توسط یک تهیه کننده در شهر تهران یا پروژه ای در شهر تهران عرضه می شود:

$\Pi_{P\#}$ (SPJ ∞ (бcity = ' تهران ' (J)) $\quad\cup\quad$ $\Pi_{P\#}$ (SPJ ∞ (бcity = ' تهران ' (S))

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)	branch (branch_name , branch_city , assests)
مشتری (نام مشتری، خیابان و شهر)	customer (customer_name , cust_street , custr_city)
حساب (شماره حساب ، نام شعبه، موجودی)	account (account_number , branch_name , balance)
وام (شماره وام، نام شعبه، مقدار وام)	loan (loan_number , branch_name , amount)
صاحب حساب(نام مشتری، شماره حساب)	depositor (customer_name , account_number)
وام گیرنده (نام مشتری، شماره وام)	borrower (customer_name , loan_number)





- نمونه پرسش ها و پاسخ آنها با استفاده از جبر رابطه ای:

۱ -تمامی وامهای بیش از ۱۰۰۰۰۰ریال را بدهید :

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

۲ -شماره وامهایی را که بیش از ۱۰۰۰۰۰ریال می باشند بدهید:

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

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

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

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

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

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

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

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

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

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

۴-۸. عملگرهای کار برروی داد ه ها:

هر کاربر در محیط پایگاه داده برای انجام عملیات نیازمند، عملگر های زیر می باشد:

- عملگر درج
- عملگر بهنگام سازی
- عملگر حذف

این عملگرها در واقع امکانات حذف، اضافه و اصلاح اطلاعات را برای کاربر فراهم می آورند .در ادامه به شرح هریک خواهیم پرداخت.

٤-۸-۱. عمل درج

این عملگر امکان اضافه کردن اطلاعات به جداول اطلاعاتی را فراهم می سازد. این عملگر در حالت کلی امکان افزودن داده ها به دوصورت زیر را دارد:

۱ -افزودن یک سطر به جدول

۲ -افزودن چندین سطر از جدولی دیگر به این جدول

شکل کلی دستور به صورت $r \leftarrow r \cup E$ است. که در آن r رابطه و E *عبارت* جبر رابطه ایست که می خواهد به r اضافه شود. در واقع عملگر انتساب است که نتیجه را به همان جدول r نسبت می دهد و ذخیره می کند.

مثال۱. حسابی با شماره A-973 برایAli در شعبه Azadi ایجادکنید:

$account \leftarrow account \cup \{(\text{“Azadi”}, \text{A-973}, 1200)\}$
$\text{depositor} \leftarrow depositor \cup \{(\text{“Ali”}, \text{A-973})\}$

٤-٨-٢. حذف

این عملگر جهت حذف سطرهایی از اطلاعات درون جداول بکـار مـی رود. در جبـر رابطـه ای عمـل حـذف بـه صورت $r \leftarrow r - E$ تعریف می شود.

مثال۱: سطرهایی از جدول حساب مربوط به شعبه Center را حذف کنید.

$$account \leftarrow account - \sigma_{branch\_name = \text{“Center”}} (account)$$

مثال۲: اطلاعات وام های کمتر از ۱۰۰۰۰ تومان را حذف کنید.

$$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 1000} (loan)$$

مثال۳: تمام اطلاعات مربوط به حساب های شهر قزوین را حذف کنید.

$$r_1 \leftarrow \sigma_{branch\_city = \text{“Qazvin”}} (account \bowtie branch)$$
$$r_2 \leftarrow \prod_{branch\_name,\ account\_number,\ balance} (r_1)$$
$$r_3 \leftarrow \prod_{customer\_name,\ account\_number} (r_2 \bowtie \text{depositor})$$
$$account \leftarrow \text{account} - r_2$$
$$depositor \leftarrow \text{depositor} - r_3$$

٤-٨-٣. بهنگام سازی

این عملگر امکان تغییر اطلاعات ذخیره شده در یک جدول را برای کاربر فراهم می آورد. شکل کلی دسـتوربروز رسانی در جبر رابطه ای به صورت $r \leftarrow \prod_{F_1, F_2, \ldots, Fn,} (r)$ است.

مثال. تمام حساب ها را ۵ درصد سود دهید.

$$account \leftarrow \prod_{account\_number,\ branch\_name,\ balance * 1.05} (account)$$

۴-۹. محاسبات رابطه ای

نوع دیگری از امکانات انجام عملیات در رابطه ها، حساب رابطه ای است که از نظر منطقی معادل جبـر رابطـه ای است. محاسبات رابطه ای امکان دیگری برای انجام عملیات روی رابطه ها و کار با بانک رابطه ای است که حالت ناروشمند (غیر رویه ای) دارد یعنی در آن عملیات لازم برای اشتقاق رابطه ای از رابطه های دیگرتصریح 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 [1]

۲ -محاسبات روی میدان ها [2]

---

1 Tuple Relational Calculus

2 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.

٤-٩-١. محاسبات رابطه ای تاپلی

در حساب رابطه ای تاپلی، یک مفهوم مهم به نام متغیر طیفی(تاپلی) وجود دارد. متغیر تاپلی متغیری است که مقادیرش از یک رابطه برگرفته می شود وبه 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. گرامر زبان مبتنی بر محاسبات رابطه ای را می‌توان درکتابهای مختلف بانک اطلاعاتی از جمله کتاب آقای دیت مشاهده کرد که خارج از بحث این درس می باشد. در اینجا به طور خلاصه برخی مفاهیم اساسی موجود در محاسبات رابطه ای بیان می شود و سپس به ذکر چند مثال بسنده می کنیم. در محاسبات رابطه ای هر پرس وجو به صورت $\{t \mid P(t)\}$ بیان می گردد که شامل مجموعه ای از تاپلهای t است که شرط P برروی آن صدق می کند. t یک متغیر تاپلی است که t[A] مقدار تاپل را روی ستون A نشان می دهد. $t \in r$ بیانگر این است که تاپل t عضوی از رابطه r است و P فرمولی است که برپایه منطق گزاره ها بیان می شود.

٤-٩-١-١. عملگر ها

دو عملگر در محاسبات رابطه ای تعریف شده اند که به آنها سورگویندکه عبارتند از:

- سور وجودی[1] به معنای وجود دارد و به صورت $\exists t \in r\,(Q(t))$ نوشته می شود که بیان می کند حداقل یک تاپل t وجود دارد که در آن شرط $Q(t)$ صدق می کند.
- سور همگانی[2] به معنای برای همه است و به صورت $\forall t \in r\,(Q(t))$ نوشته می شود که بیان می کند برای همه تاپل ها شرط $Q(t)$ برقرار است.

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

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

```
FORALL X(P) =NOT EXISTS X (NOT P)
```

این دو سور به روش های زیر قابل تبدیل به یکدیگر می باشند:

```
FORAll T(f) = NOT EXISTS T(NOT f)
EXISTS T(f) = NOT (FORALL T(NOT f))
FORALL T((f) AND (g)) = NOT EXISTS T(NOT (f) OR NOT (g))
FORALL T((f) OR (g)) = NOT EXISTS T(NOT (f) AND NOT (g))
```

---

1 Exist quantifiers

2 FOR ALL quantifire

EXISTS T((F) OR (g)) = NOT FORALL T(NOT (f) AND NOT (g))
EXISTS T((f) AND (g)) = NOT FORALL T(NOT (f) OR NOT (g))
FORALL T(f) => EXISTS T(f)
NOT EXISTS T(f) => NOT FORALL T(f)

٤-٩-١-٢. استفاده در حساب محمولات در فرموله کردن پرس و جو ها:

بانک اطلاعاتی قبلی در مورد بانک و حساب و وام را در نظر بگیرید. مـی خـواهیم نحـوه فرمولـه کـردن پـرس وجوها را با کمک محاسبات رابطه ای ببینیم.

سوال	اطلاعات شامل شماره وام، نام شعبه و میزان وام وام های بالای ۱۲۰۰۰۰ تومان را بدهید.
پاسخ	$\{t \mid t \in loan \wedge t\,[amount\,] > 120000\}$
سوال	شماره وام های بالای ۱۲۰۰۰۰ تومان را بدهید.
پاسخ	$\{t \mid \exists\, s \in \text{loan}\ (t\,[loan\_no] = s\,[loan\_\ no\,] \wedge s\,[amount\,] > 1200)\}$
سوال	اسامی مشتریانی را بدهید که یاوام یا حساب یا هردو را در بانک دارند.
پاسخ	$\{t \mid \exists s \in borrower\ (\ t\,[customer\_name\,] = s\,[customer\_name\,]) \vee \exists u \in depositor\ (\ t\,[customer\_name\,] = u\,[customer\_name\,])$
سوال	اسامی مشتریانی را بدهید که هم وام و هم حساب در بانک دارند.
پاسخ	$\{t \mid \exists s \in borrower\ (\ t\,[customer\_name\,] = s\,[customer\_name\,]) \wedge \exists u \in depositor\ (\ t\,[customer\_name\,] = u\,[customer\_name]\,)$
سوال	اسامی مشتریانی را بدهید که از شعبه center وام گرفته و شهر محـل اقامتشـان بـا شـهر شعبه یکی است.
پاسخ	$\{t \mid \exists s \in loan\ (s\,[branch\_name\,] = \text{“center”} \wedge \exists u \in borrower\ (u\,[loan\_no] = s\,[loan\_no\,] \wedge\ t\,[customer\_name\,] = u\,[customer\_name\,]) \wedge \exists\, v \in customer\ (u\,[customer\_name\,] = v\,[customer\_name\,] \wedge\ t\,[customer\_city\,] = v\,[customer\_city\,]))\}$
سوال	اسامی مشتریانی را بدهید که در تمام شعبات شهر Qazvin حساب دارند.
پاسخ	$\{t \mid \exists\, r \in customer\ (t\,[customer\_name\,] = r\,[customer\_name\,]) \wedge (\ \forall\, u \in branch\ (u\,[branch\_city\,] = \text{“Qazvin”} \Rightarrow \exists\, s \in depositor\ (t\,[customer\_name\,] = s\,[customer\_name\,] \wedge \exists\, w \in account\ (\ w[account\_number\,] = s\,[account\_number] \wedge\ (\ w\,[branch\_name\,] = u\,[branch\_name\,])))))\}$

- **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.

٤ -زبانهای مبتنی بر محاسبات رابطه ای به دلیل ناروشمند بودن سطحشان بالاتر است.

۴-۱۰. تمرینات این فصل:

بانک اطلاعاتی زیر را در نظر بگیرید :

Employee (employee_name ,stat , city,street)	کارمند (نام , وضعیت ,شهر,خیابان)
Works (employee_name ,company_name , salary)	کار (نام کارمند ,نام شرکت , میزان حقوق)
Company (company_name ,city)	شرکت (نام ,شهر)
Manages (employee_name, manager_name)	مدیر(نام کارمند ,نام مدیر)

با توجه به بانک مذکور به سوالات زیر به زبان جبر رابطه ای پاسخ دهید :

الف : اسامی وآدرس خیابان و شهر محل اقامت تمام کارمندانی را بدهید که در شرکت آفتاب رایانه کارمی کنند و حقوق بالای ۱۰۰/۰۰۰ تومان می گیرند. (۱/۵نمره)

ب: مشخصات تمام کارمندانی را بدهید که در همان شهری که شرکت کارشان قرار دارد کارمی کنند نیز اقامت دارند.۱/۵ نمره

ج) اسامی کارمندانی را بدهید که در همان شهر و خیابانی که مدیرانشان زندگی می کند قرار دارند.۱نمره

د) حقوق مدیران شرکت ایزیران را ۱۰درصد افزایش دهید. (۱ نمره)

فصل پنجم :

آشنایی با زبان SQL

۵-۱ 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. SQL تا تبدیل شدن به یک زبان رابطه ای کامل فاصله زیادی دارد. با این همه SQL استاندارد است و تقریبا توسط هر محصول نرم افزاری بانک اطلاعات پشتیبانی می شود و هر فرد حرفه ای نیازمند آن است. امروزه نسخه جدیدی از زبان استاندارد به نام Sql-2003 نیز ارائه شده است که هنوز در پایگاه های داده رابط های مورد استفاده قرار نگرفته است. زبان SQL از دو بخش عمده تشکیل شده است: زبان تعریف داده ها[1](DDL) و زبان کار با داده ها[2](DML).

۵-۲. احکام تعریف داده ها (DDL) در SQL

این احکام شامل تعریف جداول، شاخص، حذف جداول، شاخص و تغییرات آن ها می باشد. در تعریف جدول ها باید نام، صفت ها، دامنه مقادیر آنها، کلیدهای جدول و کنترل های لازم روی ستون های آن مشخص شود. انواع دامنه ها در SQL به شرح زیر است:

- Char (n) یک رشته کاراکتری با طول ثابت که توسط کاربر n مشخص می شود .
- Varchar (n) رشته های با طول متغیر حداکثر به طول n
- در Oracle نوع Varchar2 برای رشته های با طول متغیر و حداکثر ۲۰۰۰کاراکتر تعریف شده است.
- Int اعداد صحیح
- Small int اعداد کوچک
- Numeric (p,d) اعداد اعشاری p حداکثرتعداد رقمها و dتعداد رقم های اعشاری .
- Double , real

[1] Data Definition Language
[2] Data Manipulation Language

- Float(n) اعداد اعشاری با دقت حداقل n رقم .
- Not null می تواند در انتهای تعریف فیلد به کار رود که در این صورت، 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 . time1)
- 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 name type
[DEFAULT]
[CONSTRAINT]
CHECK(…)
```

مثال: دامنه مقادیر *Dollars* و *AccountType* به صورت زیر تعریف شده اند.

```
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

برای تعریف جدول از دستور زیر استفاده می شود:

Create table r ($A_1D_1, A_2D_2, ...A_nD_n ...,$
(integrity–constraint$_k$))

* r نام رابطه، هر Ai نام صفت خاصه در شمای r است و Di نوع داده ای در دامنه صفت خاصه Ai است.

مثال از تعریف جدول شعبه بانک:

```
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 ($A_1,...A_n$)

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

دستور Unique برای تعریف کلید های کاندید به کار می رود.

دستور Check (P) برای کنترل مقدار فیلدها به کار می رود که شرط کنترلی P است.

- FOREGIN Key (A_1...) References

دستور FOREIGN KEY کلید خارجی را مشخص می کندو با قید References تعیین می کند که ارجاع بـه کدام جدول صورت می گیرد.

مثال:

create table *customer*
(*customer_name* **char(20),**
customer_street **char(30),**
customer_city **char(30),**
primary key (*customer_name* **))**

create table *branch*
(*branch_name* **char(15),**
branch_city **char(30),**
assets **numeric(12,2),**
primary key (*branch_name* **))**

create table *account*
(*account_number* **char**(10)**,**
branch_name **char**(15),
balance **integer**,
primary key (*account_number),*
foreign key (*branch_name*) **references** *branch*)

create table *depositor*
(*customer_name* **char**(20)**,**
account_number **char**(10)**,**
primary key (*customer_name, account_number),*
foreign key (*account_number*) **references** *account,*
foreign key (*customer_name*) **references** *customer*)

۵-۲-۲. 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 برای تغییر در ستون های جدول از دستور عبارت alter table استفاده می شود که تغییر می تواند اضافه کردن و یا حـذف

صفات خاصه در جدول باشد. 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.

دستور 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.

دستور Alter table r Modify A NewType برای تغییر نوع یک صفت خاصه به کار می رود.

۵ -۳. احکام کار با داده ها در SQL

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

- SELECT
- UPDATE
- DELETE
- INSERT

۵-۳-۱. 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 $A_1, A_2, ..., A_n$
from $r_1, r_2, ..., r_m$
where ***P***
[Group by columns **]**
[Having P **]**
[ORDER BY A_1, A_2,**]**

A_i ها صفات خاصه و r_i ها رابطه ها و P شرط است.در صورت استفاده از سه قسمت اول پرس و جو معادل جبر رابطه ای زیر می باشد:

$$\prod_{A_1,A_2,\ldots,A_n}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$$

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 A_1,...
From r_i

- 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

مثال. شماره وام هایی که مبلغ آنها بین ۹۰۰۰۰ تا ۱۰۰۰۰۰ است را بدهید.

```
select loan_number
    from loan
    where amount between 90000 and 100000
```

- اگر چند جدول جلوی عبارت from آورده شود به منزله حاصلضرب دکارتی رابطه هاست.

مثال. حاصلضرب بین رابطه های load,borrower

```
select *
    from borrower, loan
```

مثال. اسامی مشتریان، شماره وام آنها و مبلغ وامشان را بدهید که در شعبه center وام گرفته اند.

```
select customer_name, borrower.loan_number, amount
    from borrower, loan
    where borrower.loan_number = loan.loan_number and
        branch_name = 'Center'
```

- در SQL امکان نام گذاری مجددرابطه و یا صفات خاصه با استفاده از عبارت as وجود دارد. قالب کلی دستور بصورت زیر است:

Oldname as new name
و یا Oldname newname

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

مثال. اسامی مشتریان، شماره وام آنها و مبلغ وامشان را بدهید.

```
select customer_name, T.loan_number, S.amount
    from borrower as T, loan as S
    where T.loan_number = S.loan_number
```

مثال. استفاده از متغیر تاپلی برای حاصلضرب یک رابطه در خودش. اسامی شعباتی را بدهید که میزان دارایی شان از برخی شعبات شهر کرج بیشتر است.

```
select distinct T.branch_name
    from branch as T, branch as S
    where T.assets > S.assets and S.branch_city = 'کرج'
```

- عبارت **ORDER BY** به معنی آن است که کاربر می خواهد جواب را به طور منظم روی صفت خاصه مورد نظرش ببیند. در صورتی که در انتها عبارت desc بیاید ترتیب نزولی و اگر asc بیاید ترتیب صعودی است. مثال. اسامی مشتریانی را که در شعبه مرکزی وام گرفته اند به ترتیب حروف الفبا بدهید.

```
select distinct customer_name
        from    borrower, loan
        where borrower loan_number = loan.loan_number and
                branch_name = 'مرکزی'
        order by customer_name
```

تطابق رشته ها

- 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 به کار می رود و علامت underline _ برای تطابق هر کاراکتری به کار می رود.

مثال.

```
Select  *
        From   customer
        Where          CustomerName  like  " W %"
```

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

```
Select  *
        From   customer
        Where          CustomerName  like  " _ _c %"
```

اسامی مشتریانی که حداقل سه حرف داشته باشد و سومین حرف ازسمت چپ c باشد.

```
Select  *
        From   customer
        Where          CustomerName  like  " _ _c "
```

اسامی دقیقا سه حرف داشته باشد و سومین حرف ازسمت چپ c باشد.

- 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 می‌گیرد.

عملیات مجموعه ای

عملیات اجتماع و اشتراک و تفریق نیز در SQL وجود دارند. عبارت union برای اجتماع و عبارت Intersect برای اشتراک و عبارت except برای عملگر منها به کار می روند. 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 مانند .

مثال. اسامی مشتریانی را بدهید که یا در بانک وام گرفته اند یا در بانک حساب دارند یا هردو.

```
(select customer_name from depositor)
union
(select customer_name from borrower)
```

مثال. اسامی مشتریانی را بدهید که هم در بانک وام گرفته اند وهم حساب دارند.

(**select** *customer_name* **from** *depositor*)
intersect
(select *customer_name* **from** *borrower)*

مثال. اسامی مشتریانی را بدهید که یا در بانک حساب دارند ولی وام نگرفته اند.

(**select** *customer_name* **from** *depositor*)
except
(select *customer_name* **from** *borrower)*

توابع جمعی در 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:

- avg برای محاسبه میانگین
- max , min مینیمم، ماکزیمم
- sum مجموع مقادیر
- count تعداد مقادیر

مثال. میانگین موجودی حسابهای شعبه مرکزی را بدهید.

select avg *(balance)*
from *account*
where *branch_name* = ‘Perryridge’

مثال. تعداد تاپل های رابطه Customer را بدهید.

select count (*)
from *customer*

مثال. تعداد مشتریانی که در بانک حساب دارند رابدهید.

select count (**distinct** *customer_name)*
from *depositor*

Group By

فرض کنید می خواهیم رکوردهای) سطرها (یک جدول را بر اساس مقادیر یک یا چند ستون آن جدول گروه بندی نماییم بگونه ایکه رکوردهای موجود در هر گروه دارای یک مقدار مشترک از ستونهای مورد نظر در گروه بندی باشند. برای انجام این کار از دستور Group By استفاده می شود.

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. تعداد مشتریان در هر شعبه را بدهید.

select *branch_name*, **count (distinct** *customer_name)*
from *depositor, account*
where *depositor.account_number = account.account_number*
group by *branch_name*

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

نکته. **درصورت داشتن شرط در گروه از عبارت having استفاده می شود.**

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

مثال. اسامی شعبات و میانگین موجودی آن ها رابدهید که میانگین موجودی از ۱۰۰۰۰۰ بیشتر باشد.

```
select branch_name, avg (balance)
        from account
        group by branch_name
        having avg (balance) > 100000
```

- 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 NUL L استفاده شود آنگاه سیستم با مقدار NULL برخورد می کند.
- نتیجه هر عمل ریاضی روی NULL نیز NULL است.
- تمام توابع جمعی بجز Count(*) تاپلهای با مقادیر NULL در آن صفت خاصه را نادیده می گیرند.

مثال. شماره وام هایی که مبلغ آنها مشخص نیست را بدهید.

```
select loan-number
        from loan
        where amount is null
```

- در برخی نسخه های SQL , توابع دیگری نیز وجود دارندکه از جمله می توان توابع زیر را نام برد:

توابع مثلثاتی	توابع آماری	توابع رشته ها	توابع ریاضی	توابع تاریخ
TANH	VARIANCE	TO_NUMBER	LN	SYSDATE
COSH	STTDEV	LENGTH	SIGN	NEXT-DAY
TAN		REPLACE	FLOOR	
COS		RTRIM, LTRIM	MOD	
SIN		LPAD , RPAD	EXP	
SINH		CONCAT	POWER	
		TO_CHAR	CEIL	

پرس وجوهای فرعی

یکی از نیازهای کاربران امکان طرح پرسشی درون پرسش دیگر می باشد .هر گاه بخواهیم پرسشی را درون پرسشی دیگر مطرح نماییم، به آن پرسش فرعی می گویند. پرس و جوهای فرعی یکی از تواناییهای مهم درSQL می باشدکه یک عبارت پرس و جوی Select –from-where است که در داخل یک پرس و جو به کار برده می شود.

معمولاً در پرسش های فرعی شرایط بگونه ایست که کاربر می خواهد مقداری را از بین یک مجموعه مقادیر بدست آورد. برای این منظور از عملگرIN استفاده می گردد.

مثال. اسامی مشتریانی را بدهید که هم در بانک وام گرفته اند وهم حساب دارند.

```
select distinct customer_name
        from borrower
        where customer_name in (select customer_name
                                from depositor )
```

یکی از نکات کلیدی در استفاده از عملگر مذکور این است که بایدپرسش فرعی جلوی عملگر IN حتما دارای یک ستون همنوع با ستون مورد پرسش باشد و نمی توان در پرسش فرعی از هر ستونی به عنوان خروجی استفاده کرد.

مثال. اسامی مشتریانی را بدهید که هم حساب و هم وام در شعبه مرکزی دارند.

```
select distinct customer_name
from borrower, loan
where borrower.loan_number = loan.loan_number andbranch_name = ‘center’
and   (branch_name, customer_name) in
                                    (select branch_name, customer_name
                                     from depositor, account
                                    where depositor.account_number =
                                     account.account_number )
```

اگر نتیجه پرس و جوی فرعی تک مقداری باشد می توان از عملگرد = > < نیزاستفاده 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 );
```

مثال. اسامی شعباتی را بدهید که دارایی شان از هر یک از شعبات شهر قزوین بیشتر باشد.

```
select branch_name
        from branch
        where assets >( select  max(assets)
                        from branch
                         where branch_city = ‘Qazvin’)
```

مقایسه مجموعه ای در SQL با استفاده از عملگر های SOME و ALL

برای مقایسه مجموعه ای در پرسش های فرعی عملگر های Some و ALL نیز تعبیه شده است که همراه با عملگرهای مقایسه ای <>, >=, =<, <,> به کار می روند. Some برای مقایسه با برخی عناصر مجموعه و All برای مقایسه همه به کار می رود.

مثال. اسامی شعباتی را بدهید که دارایی شان از بعضی از شعبات شهر قزوین بیشتر باشد.

```
select branch_name
        from branch
        where assets >some
                        ( select  assets
                          from branch
                          where branch_city = 'Qazvin')
```

قبلا این پرسش را به صورت زیر نوشته بودیم.

```
select distinct T.branch_name
        from branch as T, branch as S
        where T.assets > S.assets and S.branch_city = 'Qazvin'
```

مثال. اسامی شعباتی را بدهید که دارایی شان از همه شعبات شهر قزوین بیشتر باشد.

```
select branch_name
        from branch
        where assets >all
                        ( select  assets
                          from branch
                          where branch_city = 'Qazvin')
```

توجه داریم: $(\neq \textbf{all}) \equiv \textbf{not in}$ و $(= \textbf{some}) \equiv \textbf{in}$

اما $(\neq \textbf{some}) \not\equiv \textbf{not in}$ و $(= \textbf{all}) \not\equiv \textbf{in}$

بررسی برای تهی یا غیر تهی بودن جواب های پرسش فرعی با EXISTS

در SQL می توان از سور وجودی EXISTS برای بررسی تهی بودن جواب های پرسش های فرعی استفاده می شود. عبارت EXISTS مقدار درست T برمی گرداند اگر آرگومان پرسش فرعی غیر تهی باشد. و عبارت NOT EXISTS مقدار T برمی گرداند اگر آرگومان پرسش فرعی تهی باشد.

exists $r \Leftrightarrow r \neq \emptyset$

not exists $r \Leftrightarrow r = \emptyset$

مثال : اسامی تهیه کنندگان قطعه P 2را 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 وفقط 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 EXISTSهمیشه کاراتر از شبیه سازی عمل پیوند است.

مثال مهم از NOT EXISTS. اسامی مشتریانی را بدهید که در تمام شعبات شهر قزوین حساب دارند.
بایستی توجه داشته که اگر داشته باشیم $X \subseteq Y \Leftrightarrow X - Y = \emptyset$.

```
select distinct S.customer_name
    from depositor as S
    where not exists (
            (select branch_name
            from branch
            where branch_city = 'Qazvin')
            except
            (select R.branch_name
            from depositor as T, account as R
            where T.account_number = R.account_number and
                  S.customer_name = T.customer_name ))
```

بررسی تکراری بودن با نبودن تاپل های پرسش فرعی با UNIQUE

در SQL می توان از دستور Unique برای بررسی اینکه جواب های پرسش های فرعی دارای تاپل های تکراری است یا خیر استفاده می شود. عبارت Unique مشابه عبارت EXISTS بوده و در آن ارتباط بین پرسش بیرونی و درونی بایستی ذکر شود.

مثال. اسامی مشتریانی را بدهید که حداکثر یک حساب در شعبه Center دارند.

```
select distinct 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 = 'Center')
```

مثال. اسامی مشتریانی را بدهید که حداقل دو حساب در شعبه Center دارند.

```
select distinct T.customer-name
  from depositor as 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 = 'center')
```

عملگر پیوند (درونی و خارجی)

در زبان SQL برای عمل پیوند عملگر Join در نظر گرفته شده است. عملگرپیوند درونی، متداول ترین نوع پیوند بین دو جدول می باشد که بر اساس آن تمامی رکوردهایی از دو جدول که در شرط تعریف شده در پیوند صدق می کنند، مورد بررسی قرار می گیرند. دستور عملگر پیوند با توجه به شکل زیر است:

Join types		*Join Conditions*
inner join		**natural**
left outer join		**on** <predicate>
right outer join		**using** $(A_1, A_1, \ldots, A_n)$
full outer join		

در برخی مواقع ممکن است مقادیر ستون هایی که می خواهند در عمل پیوند ترکیب شوند مقدار Null داشته باشند. در چنین شرایطی عملگر پیوند داخلی برخی سطر ها را حذف می کند. چنانچه بخواهیم در عمل پیوند تمامی اطلاعات سطرهای مورد نظر نیز نمایش یابند لازم است از پیوند خارجی (فراپیوند استفاده کنیم). پیوند خارجی به سه شکل پیوند خارجی از چپ، پیوند خارجی از راست و پیوند خارجی کامل وجود دارد. برای بررسی عملکرد این عملگر ها به مثال زیر توجه کنید.

مثال. جداول زیر را در نظر بگیرید.

loan_number	branch_name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

loan

customer_name	loan_number
Jones	L-170
Smith	L-230
Hayes	L-155

borrower

عملکرد عملگر پیوند داخلی(درونی) به صورت زیر است:

loan **inner join** *borrower* **on**
loan.loan_number = borrower.loan_number

loan_number	branch_name	amount	customer_name	loan_number
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230

عملکرد عملگر فراپیوند از چپ به صورت زیر است:

loan **left outer join** *borrower* **on**
loan.loan_number = borrower.loan_number

loan_number	branch_name	amount	customer_name	loan_number
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230
L-260	Perryridge	1700	*null*	*null*

عملکرد عملگر فراپیوند از راست به صورت زیر است:

loan **Right outer join** *borrower* **on**
loan.loan_number = borrower.loan_number

loan_number	branch_name	amount	customer_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

عملکرد عملگر پیوند خارجی کامل به صورت زیر است:

loan **full outer join** *borrower* **on**
loan.loan_number = borrower.loan_number

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes

۵-۳-۲. احکام تغییر بانک اطلاعاتی:

الف. 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. اطلاعات حساب های شعبه Azadi را حذف کنید. البته بایستی توجه داشت این حذف در صورتی اجرا می شود که قانون جامعیت ارجاعی را نقض نکند.

delete from *account*
where *branch_name* = 'Azadi'

مثال. اطلاعات حساب های شعبه های شهر Alvand را حذف کنید.

delete from *account*
where *branch_name* **in** (**select** *branch_name*
from *branch*
where *branch_city* = 'Alvand')

مثال. اطلاعات حساب هایی که از میانگین موجودی کمتر هستند را حذف کنید.

در این مثال بایستی توجه داشت که مقدار میانگین فقط یک بار محاسبه و پس از آن مقادیر حساب ها با آن مقایسه می شوندو در طول اجرای پرسش مقدار میانگین محاسبه مجدد نمی شود.

delete from *account*
where *balance* < (**select avg** (*balance*)
from *account*)

۵-۳-۲-۲. اضافه کردن تاپل و یا تاپلهای جدید یک رابطه (جدول)

با استفاده از دستورINSERT 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 *ACCOUNT*
VALUES ('A-9732', 'AZADI',1200)

INSERT INTO *ACCOUNT* (*BRANCH_NAME, BALANCE, ACCOUNT_NUMBER*)
VALUES ('AZADI', 1200, 'A-9732')

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: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’
```

مثال. 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
```

نکته. ترتیب جملات بالا مهم است. هرچند می توان با استفاده از جمله CASE بطور ساده تر پرس و جو را نوشت که ترتیب اهمیت نداشته باشد.

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

۵ -۴.SQL و سطح خارجی

نما (دید)، عبارتی نامدار از جبر رابطه ای است که مشتق از رابطه های دیگر است و ماهیتا رابطه ای مجازی است، یعنی داده های ذخیره شده خاص خود را ندارد. در SQL امکان تعریف رابطه مجازی که با استفاده از یک پرسش

تعریف می شود وجود دارد. هر چنین رابطه ای که به عنوان بخشی از مدل منطقی پایگاه داده نبوده و برای کاربران به صورت یک رابطه مجازی دیده می شود را نما(دید) گویند. معمولا از مفهوم نما(دید) برای پنهان نگه داشتن برخی داده از نظر کاربران استفاده می شود. درSQL برای تعریف دید از دستور زیر استفاده می شود .

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

تمام قدرت دستور Select در خدمت دید است و در واقع مکانیزم اشتقاق دید یک پرس و جو است که از طریق Select داده می شود. پرس و جوی مربوطه در زمان تعریف دید 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 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 )
```

با حکم بالا یک دید بنام *all_customer* تعریف می شود که شامل همه مشتریان بانک اعم از وام گیرندگان یا صاحبان حساب است.

۵-٤-۱. عملیات در دید

الف) بازیابی :

عمل بازیابی روی دید ها از نظر تئوری مشکلی ندارد هر چند در بعضی سیستم ها محدودیت هایی در این زمینه وجود دارد چون view از نظرماهیتی خودش یک جدول است(البته مجازی). لذا همان حکم Select نیز برای آن عمل می کند.

مثال.

```
select customer_name
        from all_customer
        where branch_name = 'Center'
```

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 all_customer
```

دید کاربر را به عینیت[1] 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.

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 درآوری 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 سیستمR

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.

```
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 = '****'
```

[1] materialized

```
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 status، city پدیده هیچ مقدار 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 هیچ مقدار مشکلی ندارند.

مثال .

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

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

هردو دید بالا روی یک جدول تعریف شده اند و هر دو دید ماهیتاً 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 79-1978 مفهوم کلید اصلی تعریف نشده بود لذا وی آن را مطرح نکرد.

مثال. 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 درآوردن 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 رابطهSY,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 بیک بین سطرهاو 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 این صورت دید تعریف شده روی یک رابطه که فاقد کلید کاندید(اصلی) رابطه مبنا باشد، عملیات ذخیره سازی را نمی پذیرد لذا دو حالت زیر را غیر پذیرا در نظر می گیریم:

الف) دید پرتوی فاقد کلید:

ب) حاصل عمل پیوند(ترکیب) FK-FK:

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

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 درview 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. درview 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. فرض کنید دید V5بصورت زیر تعریف شده باشد:

```
CREATE      VIEW      V5
        AS   Select   S#,Status,city
                From      S
             Where   Status > 15
       With      check   option;
```

S

S#	Status
S1	16
S2	10
S3	17
S4	15

V5

S#	Status
S1	16
S3	17

تهیه کننده 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 گزینهWith 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.

مزایای دید :

۱ -تامین استقلال داده ای
۲ -تامین تعدد دید کاربران
۳ -تامین وضوح درک کاربران از داده های ذخیره شده
۴ -آسان شدن کار با داده ها در بازیابی
۵ -تامین مکانیسم خودکار ایمنی
۶ -پنهان نگه داشتن دادهای خارج از دید کاربر.

۵-۵. امکانات امنیتی SQL:

امنیت به معنی حفاظت داده ها در قبال دستیابی غیر مجاز، تغییر غیرمجازیا تخریب آنها ست. ایمنی یعنی جلوگیری از هر رویدادی که ممکن است به داده ها، برنامه ها آسیب بزند. مفهوم ایمنی با مفهوم جامعیت در عین حال که شباهت هایی با هم دارند از یکدیگر متفاوتند. در ایمنی، مساله حفاظت داده ها در برابر کاربران

غیر مجاز است و حصول اطمینان از اینکه کاربران به انجام عملیات مجازند. حال آنکه در جامعیت، نوعی حفاظت داده ها در برابر عملیات کاربر مجاز است و حصول اطمینان از اینکه اقدام کاربر مجاز صحیح بوده و صحت و دقت داده ها را خدشه دار نمی کند. 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 می توان 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}]
```

- **امتیاز ها**

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

Grant <لیست مجوز ها>

on <relation name or view name> to <user list>

در قسمت مجوز ها از یکی از ترکیبات زیر استفاده می شود:

Select, insert, update,delete:

توجه: دادن مجوز به یک دید باعث دادن مجوز به جدول های درون دید نمی شود.

مثال. برای دادن مجوز select برروی جدول branch به کاربران u1,u2,u3 به صورت زیر عمل می شود.

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

دستور لغو مجوز.

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

مثال.

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

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. معمولا" دستورات تراکنش با دستورشروع تراکنش (begin transaction) آغاز و با یک عمل commit ویا undo پایان می پذیرد. در خصوص تراکنش چند نکته وجود دارد :

۱ -طراحی صحیح [۱] . برنامه نویس باید عملیات اجرایی یک تراکنش را بصورت واحد یکپارچه طراحی کند و این به خود dbmsربطی ندارد.

۲ -خواندن اطلاعات. هر مورد اطلاعاتی مورد نیاز تراکنش باید فقط یکبار خوانده شود. به عبارت دیگر در داخل یک تراکنش یک رکورد دوبار خوانده نشود.

۳ -نوشتن اطلاعات. هر مورد اطلاعاتی مورد عمل در تراکنش در صورت تغییر فقط یک بار 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(یکپارچگی)[۲]. تراکنش ها، ساده و غیر قابل تجزیه هستند. به عبارتی کلیه عملیات هر تراکنش یا تماما" اجرا می شوند و یا هیچکدام اجرا نمی گردند.

ب: ویژگی سازگاری[۳]. تراکنش ها سازگاری پایگاه داده را حفظ می کنند. به عبارت دیگر تراکنش پایگاه داده را از یک حالت سازگار به حالت سازگار دیگری تبدیل می کند.

ج : ویژگی جداسازی[۴]. تراکنش ها از یکدیگر مجزا هستند یعنی اثر مخرب روی یکدیگر ندارند.

د: ویژگی های پایداری[۵]. 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

---

[1] correctness
[2] Atomicity
[3] consistency
[4] Isolation
[5] Durability

3.write(A)
4.read(B)
5.B:=B+50000
6.write(B)

در اینصورت با توجه به خواص تراکنش ها داریم :

الف)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.

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

ج)جداسازی: اگر بین مراحل ۳و۶ یک تراکنش دیگر اجازه دستیابی به تغییرات در بانک را داشته باشد باعث ناسازگاری در بانک خواهد گردید(مجموع A+Bکمتر از مقدار اصلی خواهد شد) لذا نبایستی امکان اجازه تراکنش های دیگر برای به هنگام سازی پایگاه را بوجود آورد.

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

```
begin atomic
  …
end
```

۵-۷. دستور assertion

مکانیسم اظهار(اعلان شرط) در پایگاه داده، شرط یا شرایطی است که می خواهیم روی پایگاه داده برقرار باشد. این مکانیسم به نوعی قانون جامعیت خاص است که روی پایگاه داده تعریف می شود. در زبان SQL برای تعریف آن از ساختار زیر استفاده می شود.

create assertion <assertion-name> **check** <predicate>

وقتی می خواهیم شرط روی تمامی تاپل ها اعمال شود نیاز به دستوری به معنای برای همه for all خواهیم داشت. برای این منظور از دستور معادل آن با استفاده از NOT EXISTS استفاده می شود.

for all X, $P(X)$ = not exists X such that not $P(X)$

مثال. در هر شعبه بانک، وام به شخصی تعلق می گیرد که حداقل یک حساب با موجودی ۱۰۰۰۰۰۰ داشته باشد.

```
create assertion balance_constraint check
(not exists (
    select *
        from loan
   where not exists (
       select *
    from borrower, depositor, account
    where loan.loan_number =
borrower.loan_number
      and borrower.customer_name =
depositor.customer_name
```

مثال. مجموع وامهای هر شعبه باید همیشه از موجودی حسابهای آن شعبه کمتر باشد.

```
create assertion sum_constraint check
(not exists (select *
            from branch
                where (select sum(amount )
                        from loan
                         where loan.branch_name =
                         branch.branch_name )
                    >= (select sum (amount )
                         from account
                          where loan.branch_name =
                           branch.branch_name )))
```

۵-۸. تمرینات این فصل :

بانک اطلاعاتی زیر را در نظر بگیرید :

Employee (employee_name ,stat , city,street)	کارمند (نام , وضعیت , خیابان,شهر)
Works (employee_name ,company_name , salary)	کار (نام کارمند ,نام شرکت , میزان حقوق)
Company (company_name ,city)	شرکت (نام ,شهر)
Manages (employee_name, manager_name)	مدیر (نام کارمند ,نام مدیر)

با توجه به بانک مذکور به سوالات زیر به زبان SQL پاسخ دهید :

الف : اسامی وآدرس خیابان و شهر محل اقامت تمام کارمندانی را بدهید که در شرکت آفتاب رایانه کارمی کنند و حقوق بالای ۱۰۰/۰۰۰ تومان می گیرند. (۱/۵نمره)

ب: مشخصات تمام کارمندانی را بدهید که در همان شهری که شرکت کارشان قرار دارد کارمی کنند نیز اقامت دارند.۱/۵ نمره

ج: اسامی کارمندانی را بدهید که در همان شهر و خیابانی که مدیرانشان زندگی می کند قرار دارند.۱نمره

د: اسامی کارکنانی را بدهید که حقوقشان از میانگین حقوق کارکنان شرکت بیشتر است . جواب خود را در قالب View بیان کنید(۱/۵ نمره)

ه: اسامی شرکتهایی را بدهید که میانگین حقوقی کارمندانشان بیشتر و یا مساوی میانگین حقوق کارکنان شرکت ایزیران می باشد. (از group by استفاده می شود.) (۱/۵ نمره)

و) حقوق مدیران شرکت ایزیران را ۱۰درصد افزایش دهید. (۱ نمره)

۲-پایگاه داده زیر را که توسط یک ناشر مورد استفاده قرار می گیرد در نظر بگیرید: فیلد Position بیانگر ترتیب نویسنده، ویرایشگر و کلمه کلیدی در کتاب است. به عنوان مثال در کتاب درسی این درس ترتیب بصورت : Silberschatz, Korth, Sudershan. است. که Silberschatz در موقعیت ۱ وKorth در موقعیت ۲ و Sudershan در موقعیت ۳ می باشد. هر نسخه جدید کتاب دارای ISBN جدید بوده و لذا Edition بخشی از کلید اصلی نیست.

Person (<u>pno</u>, name, street, city, tel_no, postal_code)

Book (<u>isbn</u>, title, date, edition, book_type)

book_author (<u>isbn, pno</u>, position)

book_editor (<u>isbn, pno</u>, position)

book_keywords (<u>isbn, kwd_no</u>, position)

keyword (<u>kwd_no</u>, keyword)

به سوالات زیر با استفاده از مورد خواسته شده پاسخ دهید: (SQL/ RA/), RA: Relational Algebra

الف) Pno همه نویسندگانی که حد اقل در ۵ کتاب کار کرده اند را بدهید [SQL]

ب) عناوین کتابهایی را که در آنها کلمه کلیدی database وجود دارد را بدهید. [RA,SQL]

SQL =

RA=

ج) آدرس(street,city , postal_code) دومین نویسنده کتابهایی که با عنوان "All abaut Database " را بدهید. [sql]

د) شماره دو بدو نویسندگانی (دو Pno) را بدهید بطوریکه نویسندگان روی یک کتاب کار کرده و دارای کد پستی یکسانی باشند..[sql]

و) شمای پایگاه داده کتاب را به نمودار E/R معادل با آن تبدیل کنید. تمامی تناظر رابطه ها را مشخص نمایید.

فصل ششم:

نرمال سازی رابطه ها

٦-١ مقدمه:

در طراحی یک سیستم بانک اطلاعاتی رابطه‌ای، همیشه این سوال برای طراح مطرح است که با توجه به داده‌های محیط و ارتباط بین موجودیت‌ها چه رابطه‌هایی باید طراحی کرد و در هرکدام چه صفات خاصه‌ای باید وجود داشته باشد. این موارد سوالاتی هستند که طراح بانک رابطه‌ای باید بتواند پاسخ‌های منطقی و صحیح برای آنها داشته باشد. نکته‌ای که در سیستم‌های رابطه‌ای وجود دارد این است که در به کارگیری آنها نیازی نیست که بانک رابطه‌ای به یکباره و به صورت کامل و درست طراحی گردد، بلکه این امکان را به طراح می‌دهند که بتواند هسته اصلی بانک را طراحی کرده و سپس آن را رشد و گسترش دهد. بدین ترتیب نیازی نیست تمامی رابطه‌ها از ابتدا مشخص و صحیح طراحی گردند بلکه رفته رفته اصلاح و گسترش می‌یابند.

۶-۲. تعریف نرمال سازی

عبارت است از فرایند اعمال یک سری قوانین برای تضمین بهینه بودن ساختار بانک اطلاعاتی. در واقع با اعمال هر صورت از نرمال سازی، به طراحی بهتری از بانک اطلاعاتی نسبت به شکل قبل دست می‌یابیم. به طورکلی به مجموعه ای از روش هایی که در طراحی پایگاه داده موجب کاهش افزونگی اطلاعات شده و بر مبنای آن روابط بین داده ها اداره شوند نرمال سازی گفته می شود. ایده اصلی نرمال تر سازی رابطه ها بر مبنای رفع ناهنجاری[1] های رابطه ها ست. 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 (فزون کاری) در عملیات باشد. برای رفع آنومالی ها باید رابطه ها نرمال تر شوند. به طور کلی اهداف نرمال سازی عبارتند از:

۱-کاهش برخی از آنومالی ها

۲- کاهش افزونگی

۳- تامین طرح بهتر برای پایگاه داده قابل درک تر

٤- اعمال برخی قواعد جامعیتی ناشی از وابستگی تابعی

[1] anomaly

در ابتدا برای درک بهتر مفهوم آنومالی رابطه ی مانند SPC را به صورت زیر در نظر می گیریم:

S#	P#	QTY	CITY
S1	P1	100	C2
S1	P2	200	C2
S1	P3	150	C2
S2	P1	100	C3
S2	P2	80	C3
S3	P1	90	C3

این رابطه عناصرش اتمیک (ساده) می باشند که به آن رابطه نرمال 1NF نیز می گویند. اما این رابطه در عملیات آنومالی دارد:

۱- در عمل درج: درج کن اطلاع $\langle s7, c7 \rangle$ یعنی S7 ساکن C7 است. این درج ناممکن است تا وقتیکه ندانیم چه قطعه ای را تهیه کرده است.

۲- در عمل به هنگام سازی: شهر s1را عوض کنید. عمل منطقاً تاپلی به عملی مجموعه ای تبدیل می شوند و به نوعی به هنگام سازی منتشر شونده[1] 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) به طور غیر لازم در یکدیگر مخلوط شده اند بعبارتی اطلاع در مورد تهیه کننده و شهرش با اطلاع قطعه مخلوط شده است.

مثالی دیگر در این زمینه مثال در پایگاه داده بانک است. فرض کنیم جدولی به صورت زیر را طراحی کرده باشیم:

Depositor_Account

CUST_NAME	ACCOUNT-NO	BRANCHNAME	BALANCE
ALI	100	CENTER	**500000**
REZA	200	KHAYAM	**300000**
HAMID	100	CENTER	**500000**
MOHSEN	300	KHAYAM	**700000**
SAEED	400	AZADI	**600000**
PARSA	500	AZADI	**200000**

آنومالی های رابطه

۱ -در درج: درج کن حساب شماره ۱۰۰ با شعبه AZADI و موجودی ۳۲۰۰۰۰۰ . این درج تا وقتی که ندانیم صاحب حساب کیست غیر ممکن است.

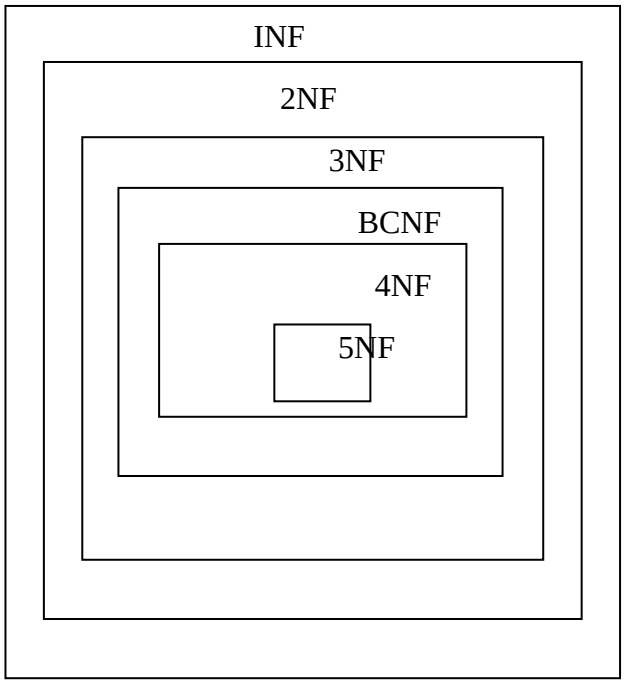
[1] Propagating Updating

۲ -شعبه حساب ۱۰۰ به Cenetral تغییر کند. این عمل منطقا تاپلی به مجموعه ای تبدیل می شود.

۳ -حساب مربوط بهREZA را حذف کنید. منجر به حذف اطلاع ناخواسته حساب ۲۰۰ مربوط بـه شـعبه KHAYAM و مبلغ 300000 می شود.

۶-۳. شکل های نرمال (سطوح مختلف نرمال)

سطوح مختلف نرمال را می توان بصورت زیر بیان نمود:



INF –

2NF–

3NF–

BCNF–

4NF–

5NP–

DK/NF–

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

٦-۳-۱. مفهوم وابستگی تابعی[1]

صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد و می‌نویسـیم $R.X \rightarrow R.Y$ اگـر و فقط اگر در طول حیات رابطه به ازای هر مقدار از X در رابطه R دقیقا یک مقدار Y از رابطـه R متنـاظر باشـد. دراین تعریف، X و Y می‌توانند صفات خاصه مرکب باشند، اصطلاحا می‌گوییم صفت خاصه X صفت خاصه Y را تعیین می‌کند[2]. به X تعیین کننده(دترمینان) و به Y وابسته گویند.

مثال مقدماتی : فرض کنید رابطه مقابل را داریم

X	Y	Z
X_1	Y_1	Z_1
X_1	Y_2	Z_1
X_1	Y_3	Z_1
X_2	Y_1	Z_2
X_2	Y_2	Z_2

در اینصورت داریم:

$X \rightarrow Z$

$Z \rightarrow X$

$X /\rightarrow Y$

[1] Functional Dependency
[2] determines

تعریف دوم وابستگی تابعی:

صفت خاصه Y از رابطه R با صفت خاصه X از R وابستگی تابعی دارد اگر و فقط اگر هر وقت در دو تاپـل از R یک مقدار X وجود داشته باشد که مقدار Y نیز در آن دو تاپل یکسان باشد. به عبارتی دیگر داشته باشیم:

$$t_1[x] = t_2[x] \Rightarrow t_1[y] = t_2[y]$$

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. به عنوان مثال در جدول زیر وابستگی های زیـادی دیـده مـی شود که در واقع صحیح نیست.

کلاس	ترم	درس	استاد
۱۰۱	۸۹/۱	اسمبلی	حمیدی
۱۰۲	۸۹/۲	مبانی کامپیوتر	شریفی
۱۰۵	۸۸/۱	مدار الکتریکی	رحیمی
۲۰۱	۸۸/۲	مدار منطقی	افتخاری

استاد → کلاس
کلاس → درس
استاد → درس
درس → استاد

۲ -وابستگی تابعی برای تعریف محدویتهای پایگاه داده نیز بکار می رود. یک وابستگی تابعی ممکن اسـت 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.

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

(s# ,p#) → Qty
(s# ,p#) → city
(s#) → city
(s# ,p#) → s#
(s# ,p#) → (city ,Qty)

برای رابطه دوم نیز وابستگی های زیر مفروضند:

(CUST_NAME ,ACCOUNT_NO)→ BRANCH_NAME
(CUST_NAME ,ACCOUNT_NO)→ BALANCE
(ACCOUNT_NO)→ BRANCH_NAME
(ACCOUNT_NO)→ BALANCE

۶-۳-۲. مفهوم وابستگی تابعی کامل (FFD)

صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی کامل دارد اگر Y با X وابستگی تابعی داشته باشد ولی با هیچ یک از زیرمجموعه‌های Xوابستگی تابعی نداشته باشد. صفت خاصه X الزاما باید مرکب باشد. و آن را بصورت R.Y ⇒ R.X نشان می دهیم به عنوان مثال در رابطه SPC داریم :

$$(S \neq, P \neq) \rightarrow Qty$$

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

$$s \neq \Rightarrow Qty$$

$$p \neq \Rightarrow Qty$$

نکته قابل اشاره این است که اگر X کلید کاندید و به ویژه کلید اصلی رابطه R باشد در این صورت هر صفت خاصه دیگر این رابطه مثلا Y الزاما با X وابستگی تابعی دارد.

تعریف ابر کلید(فوق کلید).

اگر برای تمام صفات خاصه Yدر R داشته باشیم Y → X در اینصورت X را ابر کلید R می نامند و بصورت X → R نمایش می دهند . اگر این وابستگی از نوع FFD باشد آنگاه X کلید کاندید R است.

تعریف وابستگی تابعی بدیهی

اگر Y زیر مجموعه ای از X باشد آنگاه Y → X 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. به عنوان مثال، از وابستگی تابعی $(s \neq, p \neq) \rightarrow (city, Qty)$ می توان دو وابستگی $(s \neq, p \neq) \rightarrow Qty, (s \neq, p \neq) \rightarrow city$ را نتیجه گرفت.
- اگر F مجموعه ای از FD های رابطه F باشد، مجموعه تمام FD هایی که F قابل استنتاج هستند را بستار[1] مجموعه F (مجموعه پوششی)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.

[1] Closure

٦-٣-٣. اصول آرمسترانگ

۱ -قاعده بازتابی[1]: اگرB زیر مجموعه ای از A باشد در اینصورت $A \rightarrow B$

۲ - قاعده افزایش[2] : اگر $A \rightarrow B$ در اینصورت $(A,C) \rightarrow (B,C)$

۳ - قاعده تعدی[3] : اگر $A \rightarrow B$ و $B \rightarrow C$ در اینصورت $A \rightarrow C$

۴ - قاعده تجزیه پذیری[4]: اگر $A \rightarrow (B,C)$ در اینصورت $A \rightarrow C, A \rightarrow B$

۵ - قاعده اجتماع[5] : اگر $A \rightarrow B$ و $A \rightarrow C$ در اینصورت $A \rightarrow (B,C)$

۶ - قاعده شبه تعدی[6] : اگر $A \rightarrow B$ و $(C,B) \rightarrow D$ در اینصورت $(A,C) \rightarrow D$

۷ - قاعده ترکیب[7]: اگر $A \rightarrow B$ و $C \rightarrow D$ در اینصورت $(A,C) \rightarrow (B,D)$

٦-٣-٤. بستار یک مجموعه از صفات خاصه

اگر F مجموعه ای از FD هاباشد، 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 نمائیم 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 FDهایی که در آن A تعیین کننده (دترمینان) است بدست آورد. الگوریتم محاسبه بستار به صورت زیر است:

Algorithm to compute A^+, the closure of a A under *F*
    result := A;
    while (changes to *result*) **do**
        for each $\beta \rightarrow \gamma$ **in** *F* **do**
            begin
              if $\beta \subseteq$ *result* **then** *result* := *result* $\cup \gamma$
            end
    end
End

مثال. رابطه R = (A, B, C, G, H, I) با وابستگی های تابعی زیر مفروض است. مطلوبست +(AG)

F = {A → B, A → C, CG → H, CG → I, B → H}

1. *result* = *AG*
2. *result* = *ABCG* (*A* → *C* and *A* → *B*)
3. *result* = *ABCGH* (*CG* → *H* and *CG* ⊆ *AGBC*)
4. *result* = *ABCGHI* (*CG* → *I* and *CG* ⊆ *AGBCH*)

---

[1] reflexivity
[2] augmentation
[3] transitivity
[4] decomposition
[5] union
[6] pseudo transitivity
[7] composition

مثال. فرض کنید متغیر رابطه ای R با صفات خاصه A,B,C,D,E,F, زیر داده شده است نشان دهید وابستگی تابعی F→(A,D) برای R برقرار است.

R=(A,B,C,D,E,F)
FD={ A → (B,C), B → E, (C,D) → (E,F)}
1) A → (B,C)
2) A →C تجزیه
3) (A,D) → (C,D) بسط پذیری
4) (C,D) → (E,F)
5) (A,D) → (E,F)
6) (A,D) → 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→ (X,Y), X → Y, (X,Y) → (Z,V)}
F^+ = { U → X , U → Y , X → Y , (X, Y) → (Z,V), U → (Z,V) }

تمرین: فرض کنید A→D CD→B ,AD→C نشان دهید AD ابر کلید است ولی کلید کاندید نیست.

٦-٣-٥. 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 گردد.

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 → (X,Y) , X →Y , (X,Y) → (Z,V) , U → (Z,V) }

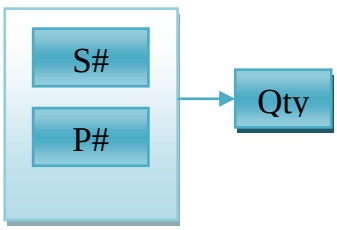
- U → (x,y) ⇒ u →x, u→ Y
- U → (Z,V) ⇒ U → Z , U → V
- (X,Y) → (Z,V) ⇒ X→Y, X→ (Z,V)
- X →(Z,V) ⇒ X→Z, X → V
- F_{OPT} = { u → x, u→ Y, U→ Z, U→ V, X→Y, X→Z, X → V}

٦-٣-٦. نمودار وابستگی تابعی

می توان وابستگی تابعی را با استفاده از نمودار نشان داد. این کار به فهم بهتر کمک می کند. 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(S#,P#,Qty) رادر نظر می گیریم. فرض کنیم وابستگی های زیر مفروضند. نمودار وابستگی به صورت زیر است:

(S#,P#) → Qty



۶-۴. نرمال سازی[1]

با استفاده از نرمال سازی می توان جداول مناسبی را طراحی کرد و یا جداول موجود را بهتر کرد. در نرمال سازی بایستی یکسری مراحل انجام شوند:

۱ -شناسایی داده ها و ارتباطات وابستگی جدول

۲ -رسم نمودار وابستگی

۳ -تبدیل جدول به فرم نرمال در سطح لزوم

نرمال سازی به معنای پیروی از یکسری قوانین که منجر به تجزیه جدول می گردد.

٦-٤-١ رابطه نرمال سطح یک (1NF)

رابطه ای را 1NF گویند اگر مقادیر تمام صفات خاصه اش تجزیه ناپذیر باشند. فرم سطح یک نرمال ملزومات یک مدل رابطه ای رابیان می کند که تمام صفت های یک رابطه به کلید اصلی وابستگی تابعی دارند.

مثال. رابطه Account_Depositor دیده شده در قبل یک رابطه سطح اول نرمال است. دیدیم این رابطه با وجود سطح اول نرمال دارای آنومالی بود. این رابطه بایستی به دو رابطه تجزیه شود.

در تجزیه یک رابطه r(R) به دو رابطه $r_1(R_1), r_2(R_2)$ بایستی شرایط زیر رعایت شود:

۱ - $R = R_1 \cup R_2$

۲ -تجزیه بدون گمشدگی اطلاعات[2] باشد یعنی: $r = \prod_{R1}(r) \bowtie \prod_{R2}(r)$

مثال. رابطه زیر را در نظر می گیریم:

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

FD ها رابطه بصورت زیر است:

(S#,P#) → Qty , (S#,P#) → CITY , (S#,P#) → STATUS

(S#) → CITY هر تهیه کننده ای در یک شهر ساکن است

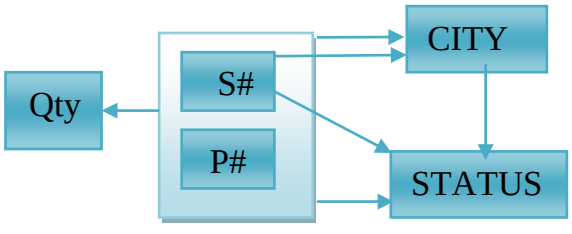
(S#) → STATUS هر تهیه کننده ای یک مقدار وضعیت دارد

CITY → STATUS تمام تهیه کنندگان ساکن یک شهر یک وضعیت دارد

نمودار FD و نمایی از داده های این رابطه به شکل زیر است.

First

S#	P#	QTY	CITY	STATUS
S1	P1	100	C1	**10**
S1	P2	120	C1	**10**
S1	P3	80	C1	**10**
S2	P1	90	C2	**15**
S3	P1	100	C2	**15**
S4	P1	60	C1	**10**



[1] normalization
[2] Lossless-join decomposition

- **آنومالیهای این رابطه:**

۱- درج کن اطلاع $\langle s7, c3,14\rangle$ 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:

$SP(s\neq, p\neq, Qty)$ و $Second(s\neq, status, city)$

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

٦-٤-٢. رابطه 2NF

رابطه ای 2NF است که:

۱ -1NF باشد

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

مثال. رابطه DEPOSITOR_ACCOUNT سطح دوم نرمال نیست زیرا وابستگی ناکامل در BRANCHNAME و BALANCE نسبت به کلید اصلی داریم.

(CUST_NAME ,ACCOUNT_NO)→ BRANCHNAME
(CUST_NAME ,ACCOUNT_NO)→ BALANCE
(ACCOUNT_NO)→ BRANCHNAME
(ACCOUNT_NO)→ BALANCE

این رابطه بایستی به دو رابطه ACCOUNT(ACCOUNT_NO, BRANCHNAME , BALANCE) و رابطه DEPOSITOR(ACCOUNT_NO, CUST_NAME) تجزیه شود.

با توجه به تعریف 2NF می بینیم رابطه FIRST نیز 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 $CO\neq \longrightarrow PR\neq$ 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 به طور مناسب استفاده می شودتا وابستگی های ناکامل حذف شوند.

- **آنومالیهای رابطه SECOND**

۱ - در درج : درج کن اطلاع <C5,18> : 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 <S4,10> را حذف کن این حذف منجر به حذف اطلاع <C4,10> می گردد.

۳ -در به هنگام سازی: وضعیت داده شده بـه شـهر c2 را عـوض کن

در اینجا عمل تاپلی به عمل مجموعه ای تبدیل می شود.

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

فـــرض کنـــیم ایـــن رابطـــه بـــه دو رابطـــه $sc(s \neq, city)$ و $cs(city, status)$ تجزیه شود.

S#	CITY	STATUS
S1	C1	**10**
S2	C2	**15**
S3	C2	**15**
S4	C1	**10**

S#	CITY
S1	**C1**
S2	**C2**
S3	**C2**
S4	**C1**

CITY	STATUS
C1	**10**
C2	**15**

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

- **علت آنومالیهای SECOND**

در رابطه SECOND نوعی وابستگی خاص بنام وابستگی با واسطه (ازطریق تعدی) وجود دارد.

تعریف وابستگی با واسطه: در رابطه R(A,B,C) اگر داشته باشیم

$A \rightarrow B \ , B \not\rightarrow A \ B \rightarrow C, \Rightarrow A \rightarrow C$

می گوئیم C به A از طریق B وابسته است.

در مثال قبل داریم: $S\# \rightarrow CITY$ و $CITY \rightarrow STATUS$ می گـوئیم STATUS ضـمن اینکـه خـود مستقیماً بی واسطه با S# وابستگی دارد از طریق CITY نیز به آن وابسته است.

٦-٤-٣. رابطه 3NF

رابطه ای را 3NF گویند هر گاه 2NF بوده وهر صفت خاصه غیر کلید با کلید اصـلی وابسـتگی بـی واسـطه داشـته باشد.

مثال : رابطه های SP وAccount، 3NF هستند.

برای تبدیل یک رابطه 2NF به رابطه 3NF باید صفتی را که وابستگی با واسطه ایجاد کرده است با همه وابسته های آن را در یک رابطه و کلید اصلی را با صفات های باقیمانده در یک رابطه جدا کنیم و سـپس صـفت هـای کلیدی رابه عنوان کلید خارجی به رابطه اول اضافه نمود.

٦-٤-٤. رابطه BCNF

رابطه 3NF با جداولی که دارای حداقل دو کلید کاندیـد باشـد و کلیـد هـای کاندیـد ترکیبـی و دارای صـفت مشترک باشد مشکل دارد. در این صورت باید ایـن نـوع جـدول را تاسـطح بیشـتری نرمـال کـرد. رابطـه ای را

BCNF گویندکه ستون های آن فقط به کلید های کاندید رابطه وابستگی تابع داشته باشند به عبارت دیگر رابطه ای که در آن هر دترمینان کلید کاندید باشد.

مثال : رابطه FIRST ، BCNF نیست زیرا در رابطه داریم : S#→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 نرمال BCNF حتماً 3NF است ولی هر 3NF ای BCNF نیست. بلکه باید بررسی شود. لذا دو حالت را در نظر می گیریم:

الف : رابطه‌هایی با یک کلید کاندید.

در این حالت می‌توان گفت : اگر رابطه 3NF باشد قطعاً BCNF هم هست. مثال: رابطه S و SP

ب : رابطه‌هایی با بیش از یک کلید کاندید :

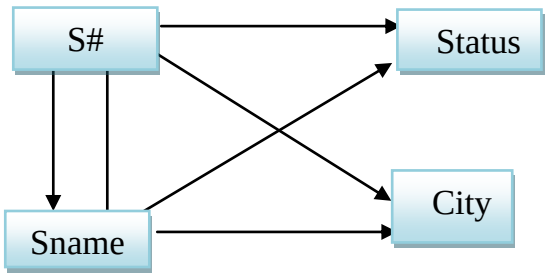
در این حالت نیز می‌توان دو حالت را در نظر گرفت

۱ -عدم وجود همپوشانی در کلیدهای کاندید

۲ -وجود همپوشانی در کلیدهای کاندید

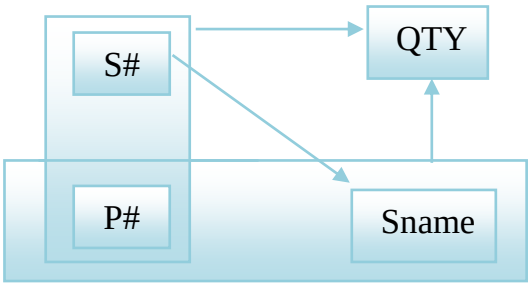
و منظور از همپوشانی این است که اگر داشته باشیم (y,z), (x,y) : R وجود y عنصر مشترک را همپوشانی گویند.

مثال ۱. رابطه S را در نظر می‌گیریم . 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 دترمینان کلید کاندید و 3NF هم می‌باشد.
- 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 (نبود صفت مشترک) اگر 3NF است BCNF نیز می باشد.

مثال۲. رابطه sps را در نظر بگیریم: SPS (s#, p#, Sname, Qty) در اینجا دو کلید کاندید داریم که با هم همپوشانی دارند. نمودار وابستگی تابعی به شکل زیر است:



این رابطه BCNF نیست زیرا S# دترمینان است ولی کلید کاندید نیست. می‌خواهیم بررسی کنیم این رابطه در چندمین صورت نرمال است. این رابطه 1NF است زیرا صفات خاصه‌اش اتمیک هستند. 2NF نیز می‌باشد زیرا وابستگی ناکامل نداریم. البته ظاهراً به نظر می‌رسد وابستگی ناکامل وجود دارد ولیکن اینطور نیست زیرا Sname خود جزئی از کلید کاندید است در حالیکه در تعریف 2NF آمده است هر صفت خاصه غیر کلید و اصلاً عضویت صفت خاصه در کلید کاندید مطرح نیست لذا 2NF می‌باشد. این رابطه 3NF نیز می‌باشد زیرا تعدی وجود ندارد.

می‌بینیم SPS، 3NF است ولی BCNF نیست. نکته جالب تر آنکه رابطه SPS اختلاط اطلاعاتی دارد با این همه با داشتن دو کلید کاندید 3NF است در حالی که معمولاً وجود پدیده اختلاط اطلاعاتی رابطه را در حد 1NF یا حداکثر 2NF نگه می‌دارد.

نتیجه : صرف گفتن رابطه‌ای اختلاط اطلاعاتی دارد لزوماً معنایش این نیست که سطح نرمالیتی آن پایین است. در عمل برای طراحی رابطه‌ها تا سطح BCNF نرمال می‌شوند. سطوح بالاتر بیشتر جنبه تئوریک و پژوهشی دارد و معنایش این است که تقریباً تمام رابطه‌هایی که BCNF هستند عملاً 5NF و ٤NF هستند بعبارت دیگر رابطه‌هایی که BCNF باشد اما 4NF و یا 5NF نباشند بسیار کم‌اند.

مثال ۳ : رابطه‌ای که 3NF هست اما BCNF نیست .

ST#	CO#	PR#
ST_1	C_1	P_1
ST_2	C_1	P_1
ST_1	C_2	P_2
ST_2	C_2	P_3
ST_3	C_2	P_2

SCP

فرض کنید در محیط آموزشی قواعد زیر موجودند:

۱ . یک دانشجو یک درس را فقط با یک استاد اخذ می‌کند.

۲ . یک استاد فقط یک درس تدریس می‌کند.

۳ . درس ممکن است توسط بیش از یک استاد تدریس شود.

در این رابطه دو کلید کاندید داریم: (St#,Co#), (St#,Pr#)

وکلیدهای کاندید با هم همپوشانی دارند. این رابطه BCNF نیست زیرا PR# دترمینان است ولی کلید کاندید نیست اما 3NF هست. PR# → CO# و (ST# , CO#) → PR#.

الگوریتم تجزیه رابطه به BCNF.

```
result := {R};
done := false;
compute F+;
while (not done) do
   if (there is a schema Ri in result that is not in BCNF)        then
    begin
        let α → β be a nontrivial functional dependency that holds on Ri
        such that α → Ri is not in F+,     and α ∩ β = ∅;
          result := (result – Ri ) ∪ (Ri – β) ∪ (α, β );
    end
        else done := true;
end
```

مثال.رابطه $R = (A, B, C)$ با وابستگی های تابعی $F = \{A \rightarrow B, B \rightarrow C\}$ را در نظر بگیرید. ایـن رابطـه در سطح BCNF نیست لذا بایستی آن را به دو رابطه تجزیه کرد. $R_1 = (B, C)$ و $R_2 = (A,B)$.

مثالی دیگر. رابطه R به صورت زیر با وابستگی های تابع تعریف شده است.

R = (branch-name, branch-city, assets, customer-name, loan-number, amount)
F = {branch-name → assets branch-city, loan-number → amount branch-name}

تجزیه های مختلف عبارتست از:

R_1 = (*branch-name, branch-city, assets*)
R_2 = (*branch-name, customer-name, loan-number, amount*)
R_3 = (*branch-name, loan-number, amount*)
R_4 = (*customer-name, loan-number*)

تجزیه نهایی عبارتست از: R_1, R_3, R_4

٦-٤-٥. رابطه 4NF

گاهی مواقع اتفاق می افتد رابطه ها تا سطح BCNF نرمال شده انـد ولـی هنـوز آنومـالی وجـود دارد. اینگونـه آنومالی ها می توانند ناشی از وابستگی چند مقداری یا وابستگی پیوندی باشد.

- تعریف وابستگی چند مقداری[1]

وابستگی چندمقداری نوعی وابستگی بین دو مجموعه مستقل از صفات خاصه اسـت. وابسـتگی چندمقـداری $(A_1,A_2,...A_n) \rightarrow\rightarrow (B_1,B_2,...,B_m)$ 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.

تعریف. فرض کنید r(R) رابطه ای باشد و داشته باشیم: $\alpha \subseteq R$ و $\beta \subseteq R$ وابستگی چند مقداری به صورت زیر تعریف می شود: $\alpha \rightarrow\rightarrow$:

بازای تمام تاپل هایی نظیر t_1, t_2 داشته باشیم $t_1[\alpha] = t_2[\alpha]$ در اینصورت تاپلهایی مانندt_3 و t_4 وجود دارنـد به گونه ای که :

$$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$$
$$t_3[\beta] = t_1[\beta]$$
$$t_3[R - \beta] = t_2[R - \beta]$$
$$t_4[\beta] = t_2[\beta]$$
$$t_4[R - \beta] = t_1[R - \beta]$$

و نمایش جدولی آن به صورت زیر است:

---

[1] multivalued dependency

	α	β	$R-\alpha-\beta$
t_1	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$a_{j+1} \dots a_n$
t_2	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$b_{j+1} \dots b_n$
t_3	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$b_{j+1} \dots b_n$
t_4	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$a_{j+1} \dots a_n$

مثال. جدول تدریس اساتید را شامل کد استاد، کد دانشکده، شهر دانشکده، کد درس و کتاب درس در نظر می‌گیریم. فرض کنیم دانشکده‌هایی که استاد در آنها تدریس می‌کند و دروسی که درس می‌دهد 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:
دانشکده‌های استاد (100) و نیز دروسی که تدریس می‌کند تکرار شده است (افزونگی) این در حالی است که جدول فوق تا سطح BCNF نرمال‌سازی 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
100	02	قزوین	C3	B3

مثال ۲. رابطه CTX حاوی اطلاعات درس، مدرس و کتاب در نظر می‌گیریم. یک درس می‌تواند توسط هر یک از مدرسین مشخص شده و با استفاده از تمام کتابهای مشخص شده تدریس شود. مثلاً درس C1 می‌تواند توسط t1 و t2 تدریس شود هم با استفاده از کتاب x1 و هم با استفاده از کتاب 2.
در واقع می‌بینیم به یک صفت خاصه مجموعه‌ای از مقادیر متناظر است .

C	T	X
c_1	t_1	x_1
c_1	t_2	x_2
c_2	t_1	x_1
	t_1	x_3
	t_1	x_4

$C \to\to X$ و $C \to\to T$

می‌توان وابستگی چند مقداری را بفرم زیر تعریف نمود:
رابطه R با صفات خاصه A و B و C را در نظر بگیریم.
می‌گوییم B با A وابستگی چندمقداری دارد و چنین نمایش می‌دهیم $A \to\to B$ اگر و فقط اگر مجموعه مقادیر B متناظر مقادیر A و C تنها به A بستگی داشته باشد و به مقدار C بستگی نداشته باشد.

فاگین نشان داد که در رابطه R(A,B,C) وابستگی چندمقداری A →→ B وجود دارد اگر و فقط اگر وابستگی چندمقداری A→→ 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 →→ 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 K → X 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 →→X).اگر CTX را به دو رابطه CT و CX تجزیه کنیم CT و CX رابطه 4NF هستند.

٦-٤-٦. رابطه 5NF (PJNF)

تعریف وابستگی پیوندی[1] :

اگر R یک رابطه و ستونهای هریک از رابطه های $R_1, R_2, \ldots R_n$ زیرمجموعه ای از ستونهای Rباشند ,آنگاه R دارای وابستگی پیوندی روی $R_1, R_2, \ldots R_n$ است اگر و تنها اگر داشته باشیم:

$$R = R_1 \infty R_2 \infty R_3 \ldots\ldots \infty R_n$$

- **تعریف رابطه 5NF**

رابطه R را 5NF گویند اگر و تنها اگر فقط به کلیدهای کاندیدش وابستگی پیوندی داشته باشد. به عبارتی دیگر وجود هر وابستگی پیوندی در آن ناشی از کلیدهای کاندید باشد. از این تعریف این نتیجه بدست می آید 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 BCNF باشد وحداقل یکی ازکلیدهای کاندید آن صفات ساده باشند آن رابطه 4NF است.

اگر رابطه ای 3NF باشد و تمام کلیدهای کاندید آن صفات ساده باشند آن رابطه 5NF است.

[1] Join Dependency

۶-۵. تجزیه خوب و بد

در فرایند نرمالترسازی مواردی وجود دارد که در آنها تجزیه یک رابطه به چند گونه امکان‌پذیر است. طراح بایستی تجزیه خوب و بد را باز شناسد. به عنوان مثال رابطه SECOND را در نظر می‌گیریم:

SECOND(S#, CITY,STATUS)

وابستگی‌های تابعی این رابطه بفرم City → S# ، Status → City ، Status → S# می‌باشد.

قبلاً این رابطه به دو رابطه C(S#, City) S و S(City, Status) C تجزیه شد. این تجزیه تنها تجزیه ممکن نیست بلکه تجزیه‌های دیگری متصور است:

B : SC(S#, City) SS(S#, Status)

C : SS(S#, Status) CS(City, Status)

کدامیک از این سه تجزیه را باید انتخاب کرد؟

تجزیه B مطلوبیت اولیه را ندارد زیرا مشکلاتی در آن وجود دارد. مثلاً نمی‌توان این اطلاع را که شهر خاصی دارای مقدار وضعیت خاصی است در بانک درج کرد تا زمانی که ندانیم چه تهیه‌کننده‌ای در آن شهر ساکن است. از نظر تئوری تجزیه‌ای 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 :

٦-۵-۱. قضیه ریسانن[1]

اگر R1 و R2 دو پرتو مستقل از R باشند، این دو پرتو از یکدیگر مستقلند اگر و فقط اگر

۱. تمام وابستگی‌های تابعی موجود در رابطه R در R1 و R2 با هم وجود داشته باشند و یا از وابستگی‌های موجود در R1 و R2 منطقاً قابل استنتاج باشند.
۲. صفات خاصه مشترک در R1 و R2 اقلاً در یکی از آنها کلید کاندید باشد.

باتوجه به قضیه ریسانن می‌بینیم 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 : در این تجزیه داریم City → S# و S# → Status و نمی‌توان 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 .

٦-۵-۲. تعریف رابطه اتمیک

رابطه‌ای که به عناصر مستقل تجزیه نشود (طبق رابطه ریسانن) 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.

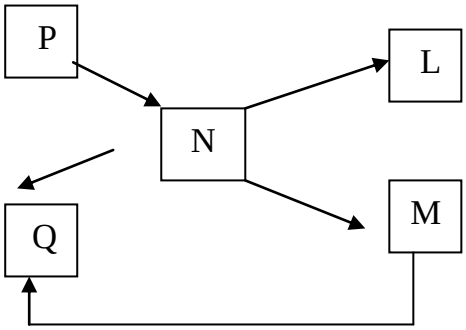
---

[1] *Risanen*

به عنوان مثـال:S(S#, Sname, Status, City) مـی‌توانـد بـه دو رابطـه SX(S#, Sname, Status) و SY(S#, City) تجزیه شود که از نظر نرمالیتی فرقی ندارد و ممکن است بدلایل دیگر تجزیه شده باشد.

۶-۶. نمونه مسائل این فصل

۱ -مجموعه حداقل FD های این رابطه را بدست آورید .



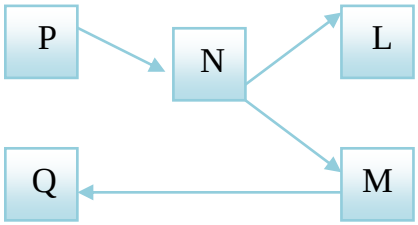
حل : با توجه به نمودار FD ها داریم :

1. P → N
2. N → L
3. P → L
4. N → Q
5. N → (L,M)
6. M → Q

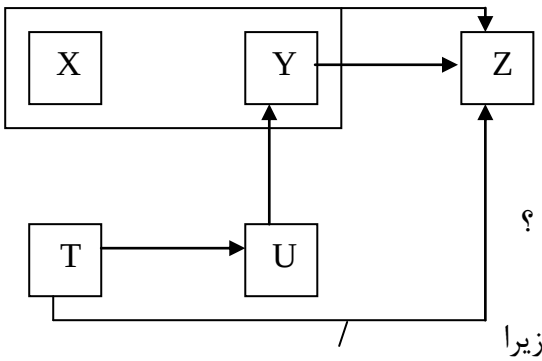
FD شماره 3 افزونه است زیرا منطقاً ازFD های 1 و 2 قابل استنتاج است . از FD شماره 2 و 5 داریم : 7. N → M

با توجه به FD های 7 و 6 ، FD شماره 4 افزونه است ، بنابراین مجموعه حداقل FD ها بصورت زیر است :

P → N و N → L و N → M و M→ Q



۲-در نمودار FD های زیر مجموعه حداقل FD ها را بدست آورید .



1. (X,Y) → Z
2. Y → Z
3. T → U
4. U → Y
5. T → Z

FD شماره ۵ افزونه است. FD شماره ۱ نیز افزونه است. چرا ؟

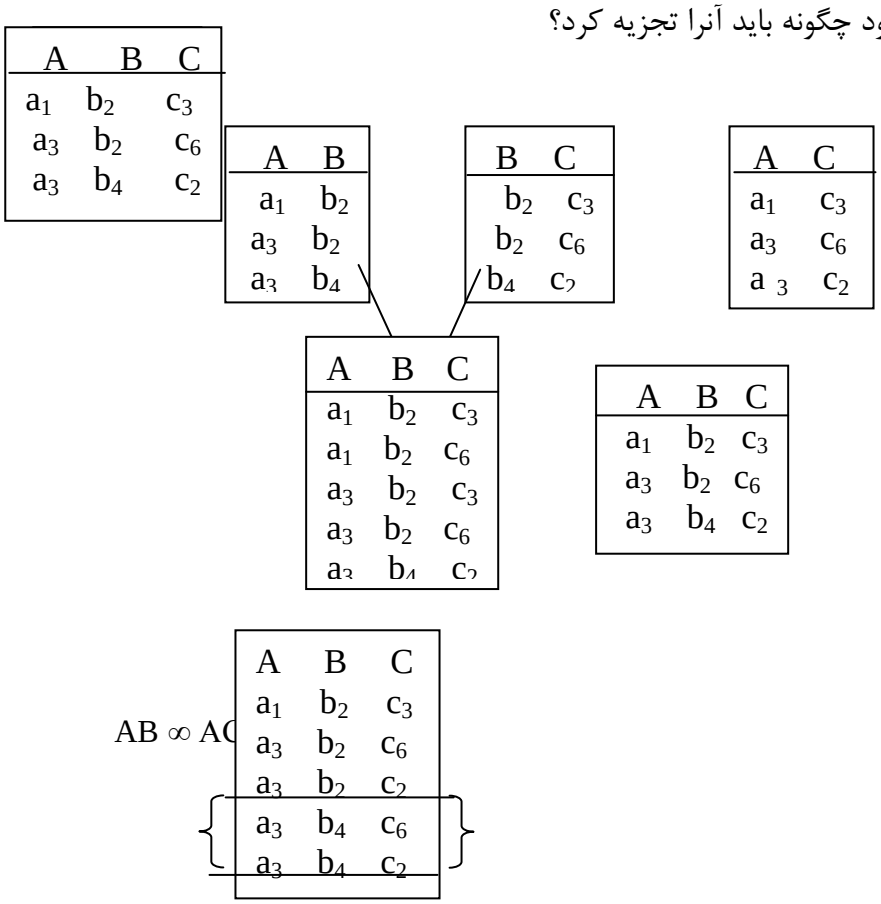
پس داریم : Y → Z , T → U

[چون از Y → Z می‌توان نتیجه گرفت (X,Y) → Z زیرا

اگر (X,Y) → Z یعنی(X1 ,Y1 , Z۱) و

(X1, Y1 , Z۲) و چون (Y1, Z2) , (Y1, Z1) لذا Y→Z خلاف فرض است پس Y→Z]

۳- رابطه R(A,B,C) را در نظر می‌گیریم. 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 تجزیه‌ها تاپل افزونه بروز می‌کند اما در تجزیه R : (BC,AC) این پدیده نامطلوب را در پی ندارد لذا این تجزیه مناسب است.

تمرین‌های این فصل.
۱ -رابطه R(A, B,C,D, E) و وابستگی های تابعی A →B , BC →E , ED→A را در نظر بگیرید. لیست تمام فوق کلید های رابطه R را به دست آورید. آیا رابطه R در سطح 3NF است ؟ آیا BCNF است؟
۲ -

فصل هفتم :

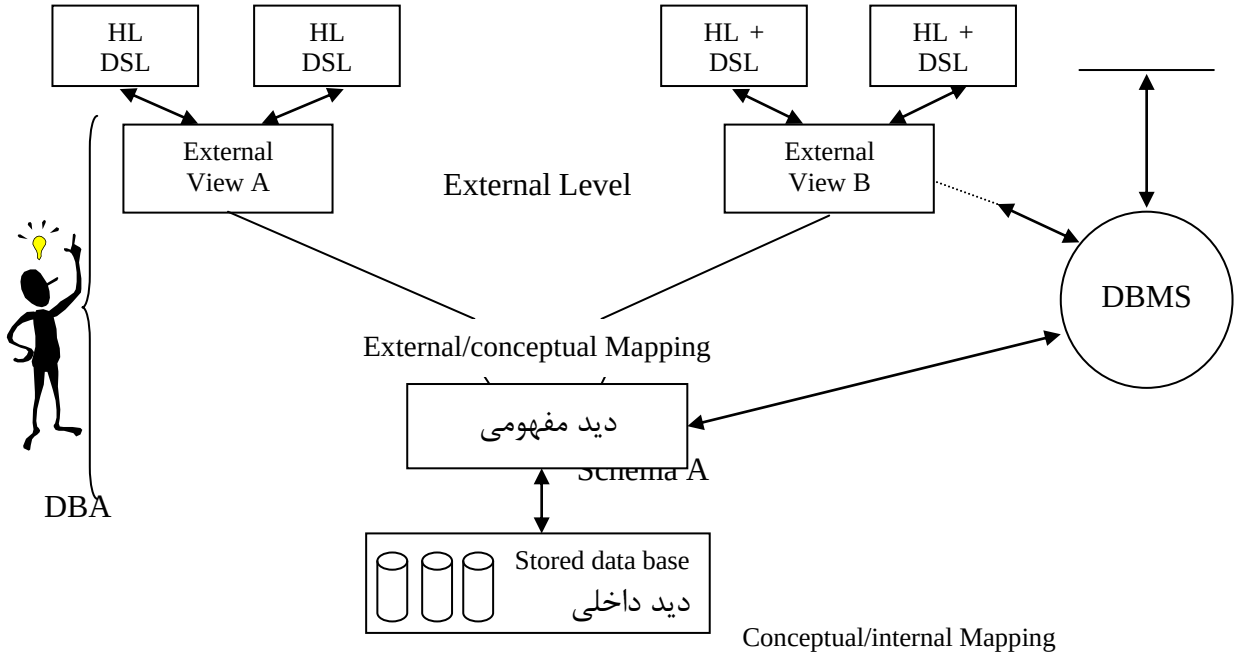
معماری سیستم بانک اطلاعاتی:

۷-۱. 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 آنجایی که 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.

۷-۲- 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 – HL زبان میزبان

۲ – زبان فرعی داده ای

۳ – دید خارجی[1]

٤ – دید مفهومی[2]

٥ – دید داخلی[3]

٦ – تبدیلات[4] بین سطوح

۷ – کاربر

۸ – اداره کننده پایگاه

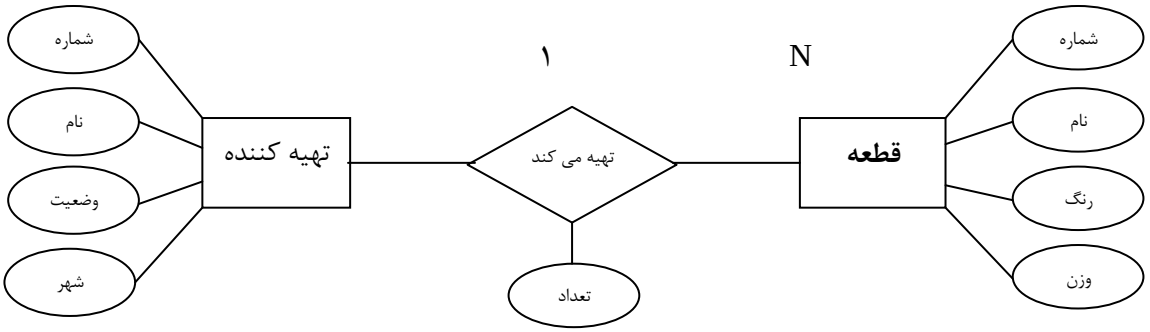
۹ – سیستم مدیریت بانک اطلاعاتی DBMS

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 هااست. 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 است. 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.

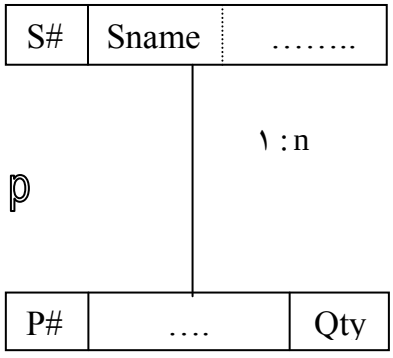
مثال ۱) با توجه به رابطه بین قطعه و تهیه کننده به کمک احکام سلسله مراتبی دید ادراکی پایگاه را تعریف کنید.



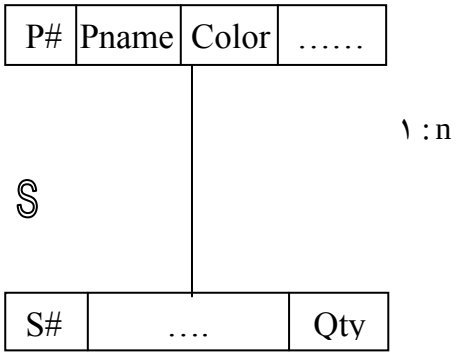
[1] External View
[2] Conceptual View
[3] Internal View
[4] mapping

ساختارسلسله مراتبی نوعی درختواره است که یک ریشه دارد و در سطوح مختلف دارای اعضا یا وابستگانی است وبه دو صورت آن را مدل می کنند.

الف: S type Record



ب: P type Record



توجه: فیلد Qty در سلسله مراتب PS در S قرار دارد و در سلسله مراتب S P در P.

- نمایش دید ادراکی به کمک ساختار 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

S#	Sname	Status	City
S_1	Sn_1	۱۰	تهران
S_2	Sn_2	۱۵	قزوین
S_3	SN_3	۷	شیراز

P

P#	Pname	Color	Weight
P_1	Pn_1	آبی	۱۰
P_2	Pn_2	زرد	۱۵

- برای نمایش ارتباطات بین دو یا بیش از دو موجودیت یک راه و البته رایج تر این 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 P_1 را به تعداد ۱۰۰ تهیه می کند.

SP

S#	P#	Qty
S_1	P_1	۱۰۰
S_2	P_2	۵۰
S_3	P_3	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:

۱) اسم جدول S است

ستونهایش
- S # کاراکتر
- Sname کاراکتر
- Status عدد صحیح
- City کاراکتر

۲) اسم جدول P است

ستونهایش
- P # کاراکتر
- Pname کاراکتر
- Weight عدد اعشاری
- Color کاراکتر

- شناسه جدول S ، S# است.
- شناسه جدول P، P# است

۳) اسم جدول SP است

ستونها
- S# است (همان S# که در S می باشد)
- P# است (همان P# که در P است)
- Qty است

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.

مثال:

u1

Myv3		Myv1	
City	Sname	STA	S#
قزوین	S_2	۱۰	S_1
		۱۵	S_2

•نمونه ساده شده از شمای خارجی

۱ – اسم دید myv_1 است

۲ – این دید روی جدول S تعریف می شود.

۳ – ستونهایش S# و STA است.

٤ – ستون S# از ستون S# مشتق می شود.

٥ – ستون STA از ستون Status مشتق می شود.

٦ – شرط یا شرایط دید C_1 و C_2 ، 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 شاخص هایی وجود دارد و فیلدهای ذخیره شده چگونه نمایش داده می شوند.

۷-۳-٤. زبان میزبان [1]

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.

۷-۳-٥-زبان داده ای فرعی:

هر DBMS یک زبان داده ای فرعی دارد. این زبان، 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:

[1] Host language (HL)

۱ – احکام تعریف داده ها[1] (DDL)

۲ – احکام تعریف کار با داده ها[2] (DML)

۳ – احکام کنترل کار با داده ها[3] (DCL)

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 میزبان 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 به طور کلی به دو صورت ادغام صریح و ادغام ضمنی وجود دارد. در ادغام ضمنی، احکام زبان داده ای به طور 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 منطقاً یک حکم واحد داشته باشد و ترجیحاً همان حکم واحد در سطح خارجی و هم در سطح ادراکی عمل نماید.

۷-۳-۶. نگاشت

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 دیدهای 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.

---

[1] Data Definition Language
[2] Data Manipulation Language
[3] Data Control Language

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 دستیابی 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 – ابزار های ایجاد برنامه های کاربردی

۴ – نرم افزارهای مخصوص محیط شبکه ای

۵ – امکانات گرافیکی

۶ – امکانات دستیابی به داده های دور .

اینکه سیستم مدیریت بانک چگونه درخواستهای کاربران را عملی می سازد بستگی به نوع آن دارد. به طور 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

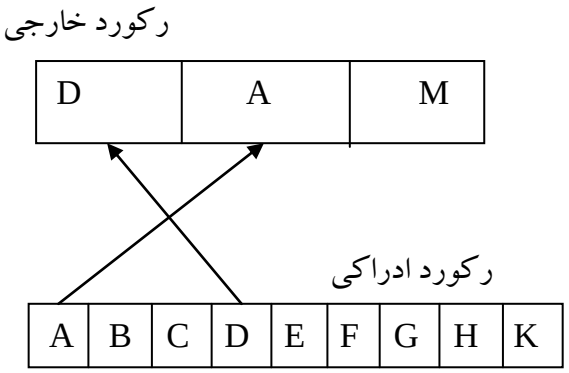
۶ – بررسی شمای داخلی و تبدیل احکام سطح ادراکی به سطح داخلی

۷ – دستیابی به فایلهای فیزیکی و اجرای درخواست کاربر

•اصطلاح به عینیت درآوردن داده[1]

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 نظرکاربر 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.

به طور کلی می توان وظایف DBA را به شرح زیر بیان نمود:

---

[1] materialization

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

٤ – مطالعه دقیق محیط عملیاتی و تشخیص نیازهای کاربران مختلف

٥ – بازشناسی موجودیت ها و ارتباطات بین آنها و تعیین صفات خاصه هر یک از انواع موجودیتها

٦ – 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 افـزاری 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

۱٤ – نظارت در جمع اوری داده ها و ورود داده ها

۱٥ – تست پایگاه با داده های واقعی (آغاز پیاده سازی پایگاه)

۱٦ – 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

۲٤ – 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 بانـک تحت عنوان کاتالوگ سیستم نگهداری می کند. در واقع کاتالوگ جائی است که تمام شماهای مختلف (خارجی

، مفهومی و داخلی) و تمام نگاشتهای متناظر با آنها در آن نگهداری می شوند. به بیان دیگر کاتالوگ شامل اطلاعات تفضیلی (که گاه فرا داده نامیده می شوند) مربوط به اشیاء متعددی است که در خود سیستم قرار دارند .

به طور 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
- شمای ادراکی
- مشخصات فنی کاربران و چگونگی حق دستیابی انها به داده ها و محدوده عملیات مجاز آنها
- رویه های تبدیل بین سطوح مختلف
- تاریخ ایجاد داده ها

۷-٤- دلایل استفاده از بانک اطلاعاتی: ? Why Database

۷-٤-۱- 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

٤ – کاهش میزان افزونگی

٥ – تضمین جامعیت داده ها

٦ – 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

۷-٤-۲- معایب بانک اطلاعاتی (چند کاربری)

۱ –هزینه بالای نرم افزار

۲ – هزینه بالای سخت افزار

۳ -هزینه بیشتر برای برنامه سازی

٤ - هزینه بالا برای انجام مهندسی مجدد به منظور تبدیل سیستم از مشی فایل پردازی به مشی پایگاهی

٥ -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 داردو خدمات پایگاهی به کاربران ارائه میکنند.این معماری بستگی به دو عنصر اصلی یعنی سخت افزار و نرم افزار مدیریت DBMS دارد. به طور کلی چهار معماری برای سیستم پایگاه داده ها وجود دارند :

۱ -معماری متمرکز

۲ -معماری سرویس دهنده – سرویس گیرنده

۳ - معماری توزیع شده

٤ - معماری با پردازش موازی

۷-۵-۱- معماری متمرکز Centeralized

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 دوقسمتی در نظر می‌گیرد : 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.

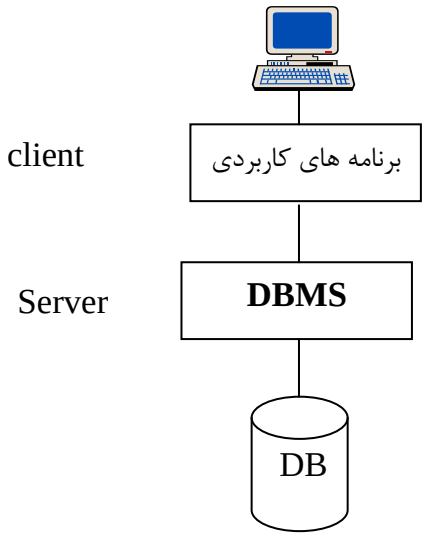
الف- سرویس دهنده:

یک سیستم بانک اطلاعاتی است که از تمام عملیات اصلی یک DBMS پشتیبانی می کند.در واقع می توان نام دیگری برای DBMS را سرویس دهنده نیز به طور 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 بصورتهای 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 نظز 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 (به طور 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

۷-۵-۳-۲. معایب

- پیچیدگی طراحی سیستم
- پیچیدگی پیاده سازی
- هزینه بیشتر

۷-۵-٤- 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 ، به طور 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

۷-٦- 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 می کند. 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

٤- 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 فاز های 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) DSL 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 لزوما باید!) از استفاده از تکنولوژی پایگاه داده ها صرفنظر 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؟

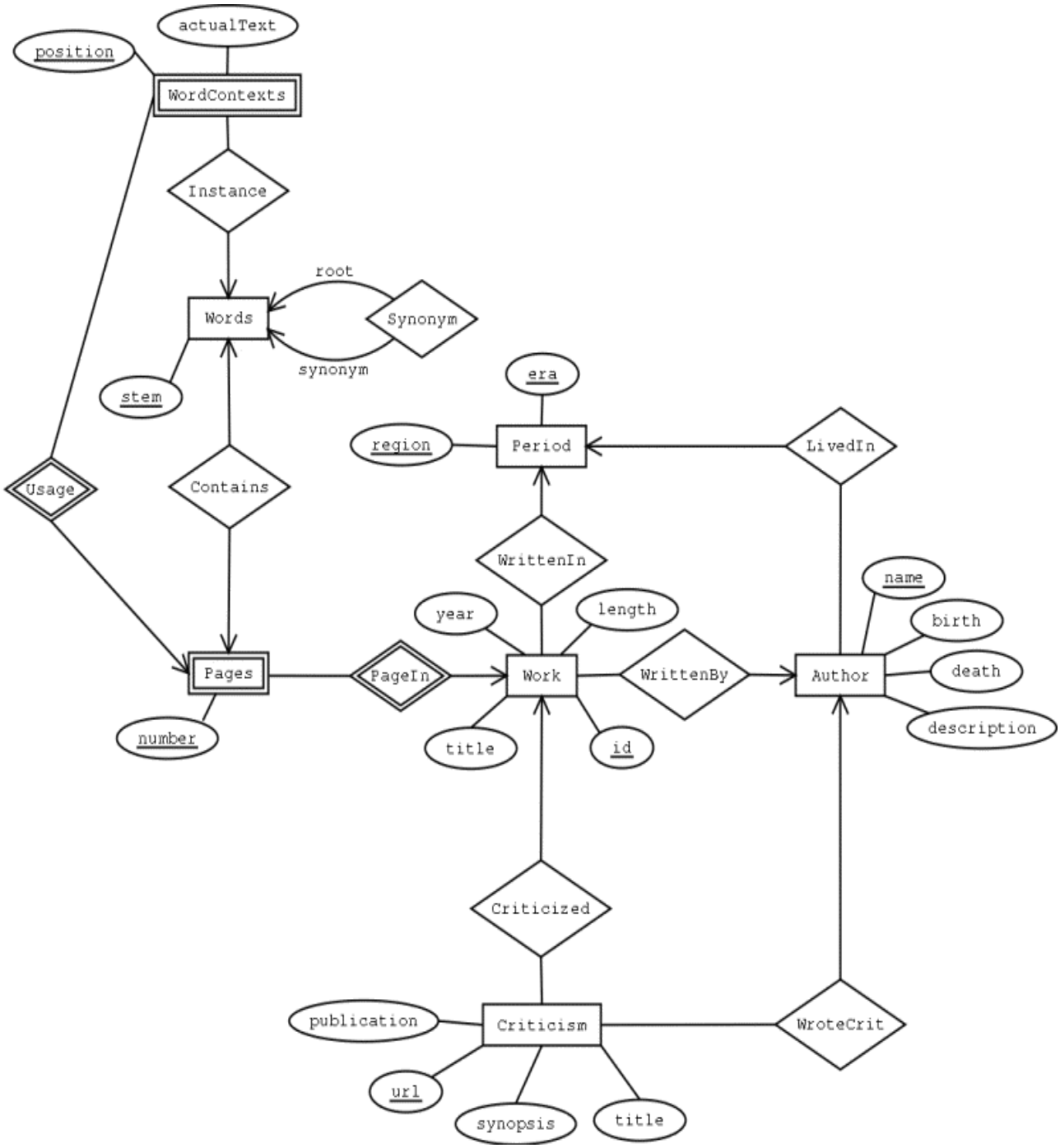
یک مثال عملی :

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.

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 (
  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 indicies 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 indicies 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 databse 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 iteself) 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 (:NE W.id, :NEW.id);
  END ReflexiveSynTrig;

.
run;
```

فهرست منابع ومراجع:

۱- H.Korth and A.silberschatz ,Database System Concepts , Fourth Edition ,Mc Graw Hill ,2006.

۲- C.J.Date , Introduction To Database Systems ,7th Edition , Addison Wesley ,2000.

۳- Elmasri and Navathe, Fundamentals of Database Systems, third edition,Addison Wesley, 1999.

٤- M.kroenke , Database Processing fundementals,Design &Implementation ,8th Edition.2002.

٥- R.Stephens,Teach Yourself SQL In 21 Days. Second Ed.

٦- 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.