پایگاه داده‌ها

فهرست

پیشگفتار

این کتاب با توجه به سر فصل تعیین شده برای دانشجویان دانشگاه پیام‌نور در رشته کامپیوتر تهیه و تنظیم شده است. در تهیه این کتاب سعی بر آن شده است تا مباحثی که برای تدریس درس پایگاه داده‌ها لازم به تدریس است، مطرح گردند. این کتاب مشتمل بر ده فصل می‌باشد.

در ابتدای کتاب لیست سر فصل مطالب قید شده است. در انتهای کتاب مجموعه‌ای از سؤالات تستی به همراه پاسخ نامه ارائه شده است. همچنین به منظور آشنایی بیشتر علاقمندان به ساختار پایه‌ای زبان SQL ، در ضمیمه ۲ کتاب متن کامل SQL BNF ارائه شده است.

نظر به لزوم جاگذاری معادل فارسی کلمات تخصصی برای راحتی فهم دانشجویان دو واژه‌نامه به‌صورت انگلیسی به فارسی و فارسی به انگلیسی در انتهای کتاب آمده است. در صفحه پایانی کتاب لیست منابع و ماخذ نیز برای آگاهی دانشجویان ارائه شده است.

این اثر با دقت نظر فراوان کارشناسان مدیریت تولید مواد و تجهیزات آموزشی مورد ارزیابی قرار گرفت که بدینوسیله از جناب آقای اکبری به نمایندگی از آن عزیزان قدردانی می‌نماییم.

کتاب حاضر بعنوان منبع درسی در دانشگاه پیام‌نور اعلام شده که بعلت کوتاه بودن زمان امکان رفع کلیه ایرادات تایپی و نگارشی میسر نشد. لذا از کلیه صاحب‌نظران، اساتید و دانشجویان محترم تقاضا دارد نظرات پیشنهادی خود را در جهت اصلاح ایرادات این اثر، به آدرس پست الکترونیکی مولفین ارسال فرمایند.

در پایان از آقای مهندس کامیار آهنکوب که در تنظیم و تدوین کتاب همکاری شایانی داشته‌اند سپاسگزاری می‌کنیم.

ناصر آیت
ayat@pnu.ac.ir

احمد فراهی
afaraahi@pnu.ac.ir

تابستان ۱۳۸٦

فصل ۱

مفاهیم اولیه سیستم پایگاه‌داده

هدف کلی

در این فصل ابتدا تاریخچه‌ای در رابطه با بحث پایگاه‌داده‌ها و ذخیره سازی داده‌ها مطرح و سپس انواع سیستم‌های پایگاه‌داده نام برده می‌شوند. در ادامه بعضی از مفاهیم کلیدی در پایگاه‌داده‌ها شرح داده خواهند شد. سپس مفهوم پایگاه‌داده بصورتی دقیق تر مورد بحث و بررسی قرار خواهد گرفت. با تعریف پایگاه‌داده‌ها برای درک بهتر موضوع، مراحل کار در مشی فایلینگ و پایگاه‌داده مورد بحث و بررسی قرار خواهند گرفت و در پایان اجزاء پایگاه‌داده مورد بحث و بررسی قرار گرفته و جایگاه مدیر پایگاه‌داده بررسی خواهد شد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- انواع سیستم‌های پایگاه‌داده
- سیستم مدیریت پایگاه‌داده[1]
- اصطلاح ذخیره و بازیابی اطلاعات (شامل داده، اطلاع، دانش و...)
- تعریف پایگاه‌داده
- مراحل کار در روش فایلینگ
- مراحل کار در روش پایگاهی

1. Database Management System (DBMS)

- اجزاء پایگاه‌داده
- مفهوم داده، سخت‌افزار و نرم‌افزار در پایگاه‌داده‌ها
- انواع کاربران در پایگاه‌داده‌ها
- تعریف مدیر پایگاه‌داده و وظایف آن

۱- تاریخچه

یکی از متداول‌ترین و اصلی‌ترین اصطلاحات در مقوله انفورماتیک اصطلاح پایگاه‌داده‌ها می‌باشد. اصطلاح سیستم مدیریت پایگاه‌داده‌ها در معنای عام، یکی از سیستم‌های ذخیره و بازیابی اطلاعات است. پس از طراحی نسل اولیه سیستم‌های ذخیره و بازیابی اطلاعات، برای تقویت مکانیزم‌ها و الگوریتم‌های مرتبط با ذخیره، بازیابی و پردازش داده‌ها و همچنین به منظور تسهیل در انجام امور فوق، سیستم‌های مدیریت پایگاه‌داده‌ها شکل گرفتند.

تکنولوژی پایگاه‌داده‌ها از اواسط دهه شصت میلادی و به منظور توسعه سیستم‌های فایلینگ‌ایجاد شدند. طول دهه هفتاد مدلهای پایگاه‌های داده سلسله مراتبی و شبکه‌ای توسعه یافته و مورد استفاده زیادی قرار گرفتند. در اوایل دهه هشتاد، شاخه‌ای از آن تکنولوژی به نام سیستم مدیریت پایگاه‌داده‌های رابطه‌ای مورد توجه بیشتری قرار گرفت و به عنوان تکنولوژی برتر شناخته شد و هم اکنون هم بسیار مورد استفاده قرار می‌گیرد. مهمتریت خصیصه سیستم‌های مدیریت پایگاه‌داده‌ها، مستقل شدن برنامه‌های کاربردی از جنبه‌ها و خصوصیات محیط فیزیکی ذخیره سازی است.
از اواسط دهه هشتاد تاکنون، سیستم‌های دیگری هم‌ایجاد و عرضه شد از جمله:

- سیستم فایلینگ معمولی داده‌ها
- سیستم مدیریت داده‌ها و جستجوها
- سیستم مدیریت پایگاه‌داده‌ها[1]

به مرور زمان بواسطه افزایش حجم و نوع اطلاعات نیازهای جدیدتری مطرح و

1. Relational Database Management System (RDBMS)

پایگاه‌های داده متناسب با آنها نیز ارائه گردید که نمونه‌های آن در ذیل آمده است:

- سیستم مدیریت پایگاه دانش [۱]
- سیستم معنایی [۲] مدیریت پایگاه‌داده‌ها
- سیستم هوشمند [۳] مدیریت پایگاه‌داده‌ها
- سیستم مدیریت پایگاه‌داده‌های شیی گرا [٤]
- سیستم مدیریت پایگاه‌داده‌های زمانمند [٥]
- سیستم مدیریت پایگاه‌داده بلادرنگ [٦]
- سیستم داده کاوی [۷]
- سیستم مدیریت چند پایگاهی [۸]

اما باید به این نکته اشاره کرد که بیشتر این سیستم‌ها در کاربردهای خاصی استفاده می‌شوند. امروزه نوع جدیدی از سیستم‌های مدیریت پایگاه‌داده‌ها بنام سیستم مدیریت پایگاه‌داده استنتاجی بصورت آکادمیک در حال طراحی می‌باشد که البته هنوز در محیط‌های تجاری و صنعتی نمودی پیدا نکرده است.

با توجه به آنچه گفته شد شاید بتوان سیستم‌های پایگاهی را به سه نسل زیر تقسیم کرد:

- سیستم‌های پیش رابطه‌ای [۹]
- سیستم‌های رابطه‌ای [۱۰]
- سیستم‌های پس رابطه‌ای [۱۱]

1. Knowledge Base Management System (KBMS)
2. Semantic System
3. Intelligent System
4. Object Oriented Database Management System
5. Temporal Database Management System
6. Real-Time Database Management System
7. Data Mining System
8. Multi Database Management System
9. Pre Relational Systems
10. Relational Systems
11. Post Relational Systems

شاید بتوان گفت علت اصلی تفاوت سیستم مدیریت پایگاه‌داده‌ها با بقیه سیستمها در وجود سیستم حصاری نفوذ ناپذیری به نام سیستم مدیریت پایگاه‌داده است که هر گونه دستیابی به داده‌ها باید از طریق این سیستم انجام شود. سیستم مدیریت پایگاه‌داده در واقع انقلابی در بانکهای اطلاعاتی به شمار می‌آید. در همین اواخر دو تحول دیگر هم در تکنولوژی بانک اطلاعاتی پدید آمد:

- طراحی وایجاد پایگاه‌های داده توزیع شده[1] تحت شبکه‌های مختلف
- طراحی وایجاد سیستم‌های مدیریت پایگاه‌داده برای کامپیوترهای شخصی

برای آشنا شدن ذهن خواننده تعریف چند اصطلاح پایه‌ای در سیستم‌های پایگاه‌داده در ذیل ارائه می‌گردد. لازم به ذکر است که تعاریف مربوط به اصطلاحات ذکر شده، بعضا در کتاب‌های مختلف با کمی‌تغییر نوشته شده‌اند. ولی مفهوم اصلی آنها بسیار نزدیک بهم هستند.

۱-۱ ذخیره و بازیابی اطلاعات

اصطلاح ذخیره و بازیابی اطلاعات در واقع به مجموعه‌ای از الگوریتم‌ها و تکنیکها اطلاق می‌گردد که در طراحی و تولید یک سیستم بکار گرفته می‌شود و به کاربر امکان می‌دهد تا اطلاعات (اسناد، مدارک، متون، اصوات و تصاویر و...) خود را ذخیره سازی، بازیابی و پردازش کند. این گونه داده‌ها ممکن است ساختمند، نیم ساختمند و یا حتی ناساخته باشند.

۱-۲ داده

اصطلاح داده در مفهوم کلی عبارت است از نمایش ذخیره شده کلیه موجودیتها، واقعیات و رخدادها که در تصمیم گیری بکار می آیند.

۱-۳ تعریف داده از دیدگاه ANSI

استاندارد ANSI برای اصطلاح داده تعاریف زیر را ارائه کرده است:

1. Distributed Data Base

- هر نمایشی که توسط انسان یا یک سیستم مکانیکی خودکار معنایی به آن قابل انتساب باشد.
- نمایش واقعیات، پدیده‌ها، مفاهیم یا شناخته‌ها به طرزی صوری و مناسب برای برقراری ارتباط، تفسیر یا پردازش توسط انسان یا هر دستگاه خودکار.
- به طور کلی می‌توان گفت که داده ارزشهای واقعی هستند که از طریق مشاهده و تحقیق بدست می آیند.

۱-٤ اطلاع

هر نوع داده پردازش شده (ساخت یافته) را اطلاع می‌نامند. این تعریف یک تعریف بسیار ساده است که بیانگر تفاوت بین دو اصطلاح داده و اطلاع است. ولی بطور کلی می‌توان گفت اطلاع مجموعه داده‌هایی است که در تصمیم گیری بکار می‌روند و اساساً کمیتی است نسبی و وابسته به وضعیت مشخص، زمان مشخص و نیز خود شخص (یا سیستم) تصمیم گیرنده. در واقع، شخص تصمیم گیرنده با تفسیر داده‌ها در یک وضعیت مشخص اطلاعات لازم برای تصمیم گیری را به دست می‌آورد.

۱-۵ تعریف اطلاع از دیدگاه ANSI

استاندارد ANSI برای اصطلاح اطلاع تعریف زیر را ارائه کرده است:

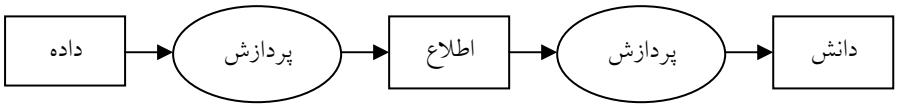
معنایی که انسان از طریق توافقات و قراردادهای شناخته شده‌ای به داده منتسب می‌کند.

۱-٦ دانش

اصطلاح دانش عبارت است از نمایش نمادین بخش‌هایی از دنیای واقعی. به بیانی دیگر، دانش نوعی شناخت است که از یک مجموعه از اطلاعات، براساس یک مجموعه از قواعد مشخص بدست می‌آید.

نکته۱: بعضی از تئوریسین‌ها داده را همان مقدار واقعا ذخیره شده و اطلاع را معنای آن می‌دانند، بنابراین اطلاع و داده با هم فرق دارند، اطلاع دارای خاصیت ارتباط دهندگی و انتقال دهندگی است درحالیکه داده مجرد این خاصیت را ندارد.

نکته۲: داده‌ها حالت منفرد و مجزا دارند و لزوما اطلاعی از آنها بدست نمی‌آید مگر اینکه به نحوی بهم مرتبط شوند و معنایی به آنها منتسب شود و باید دانش را نوعی اطلاع سطح بالاتر دانست در واقع هم اطلاع و هم دانش حاصل عملیاتی روی داده هستند، ولی نوع عملیات لازم برای حصول آنها متفاوت است. با این اوصاف رابطه بین سه مفهوم داده، اطلاع و دانش بصورت زیر قابل نمایش می‌باشد:



شکل ۱-۱ رابطه نمادین بین داده، اطلاع و دانش

با توجه به توضیحاتی که در بالا ارائه شد، اکنون توضیحی دقیقتر از مفهوم پایگاه‌داده را ارائه می‌دهیم.

۲-تعریف پایگاه‌داده

پایگاه‌داده‌ها با توصیفی جامع‌تر، مجموعه‌ای است از داده‌ها که بصورت مجتمع و تا حد ممکن بصورت مرتبط بهم و با کمترین افزونگی ذخیره شده‌اند که این مجموعه تحت مدیریت یک سیستم کنترل متمرکز برای استفاده یک یا چند کاربر قرار گرفته‌اند.

شاید در نگاه اول تعریف ارائه شده در مورد پایگاه‌داده‌ها کمی‌مبهم به نظر برسد. در تشریح کلی سیستم پایگاه‌داده‌ها می‌توان گفت که یک سیستم پایگاه‌داده مجموعه‌ای از داده‌های بهم وابسته است که از افزونگی بی‌حاصل و مضر مبرا است و برای کاربردهای گوناگون استفاده می‌شود. داده‌ها به گونه‌ای ذخیره شده‌اند که از برنامه‌هایی که آنها را به فرمت می‌گیرند مستقل هستند و راه یافت مشترک کنترل شده برای درج، حذف، تغییر و بازیابی داده‌های موجود استفاده می‌شود و داده‌ها به‌گونه‌ای ساخت یافته است که پایه‌ای برای توسعه برنامه‌های کاربردی آینده فراهم می‌سازد. این داده‌ها توسط یک سیستم مدیریت پایگاه‌داده‌ها مدیریت می‌شود. با این وصف می‌توان دریافت که هر

مجموعه‌ای از فایلها یا هر مجموعه‌ای از اطلاعات ذخیره شده لزوما یک پایگاه‌داده‌ها نیست.

با توجه به مطالب فوق می‌توان چنین نتیجه گرفت که برای ایجاد پایگاه‌داده‌ها وجود حداقل یک سیستم مدیریت پایگاه‌داده‌ها به عنوان سیستم واسطه الزامی است. بارزترین برتری یک پایگاه‌داده نسبت به سیستم بانک‌های داده قبلی (سیستم‌های فایلینگ)، سیستم مدیریت پایگاه‌داده می‌باشد. با این وصف لازم است بین اصطلاحاتی مانند بانک داده، بانک اطلاعاتی، پایگاه‌داده و پایگاه اطلاعاتی تفاوت قائل شویم.

در بررسی محیط یک پایگاه‌داده‌ها لازم است به این نکته توجه شود که محیط واحد، مجتمع و مشترک ذخیره سازی لزوما به این معنا نیست که چنین محیطی از نظر فیزیکی و محل جغرافیایی واحد و یکپارچه است. بلکه محیط پایگاه‌داده‌ها از لحاظ منطقی، یکپارچگی دارد. وضع پایگاه‌داده‌ها در سطح فیزیکی بستگی به معماری سیستم پایگاه‌داده‌ها دارد، پایگاه‌داده‌ها می‌تواند در عین واحد مجتمع و مشترک بودن در سطح منطقی، از نظر فیزیکی نامتمرکز و توزیع شده باشد.

۲-۱ تفاوت‌های بین روش فایلینگ و روش پایگاه‌داده‌ها

با توجه به تعاریف بالا ممکن است این سئوال پیش بیاید که اساساً استفاده از روش فایلینگ در طراحی برنامه‌ها به چه صورت بوده و چه تفاوتی با روش پایگاهی دارد. برای درک بهتر موضوع ابتدا روش فایلینگ برای طراحی برنامه‌ها را بصورت کلی بیان می‌کنیم.

۲-۱-۱ مراحل کار در روش فایلینگ

سیستم مورد درخواست برای مکانیزه کردن انتخاب و مورد بررسی و پردازش قرار می‌گیرد تا مشخصات نیازها تعیین گردد.

- مراحل اولیه لازم برای طراحی و پیاده سازی سیستم مورد نظر انجام می‌گیرد.
- مشخصات سیستم و زیر سیستم‌های احتمالی و وظایف هر کدام از آنها تعیین می‌گردد.

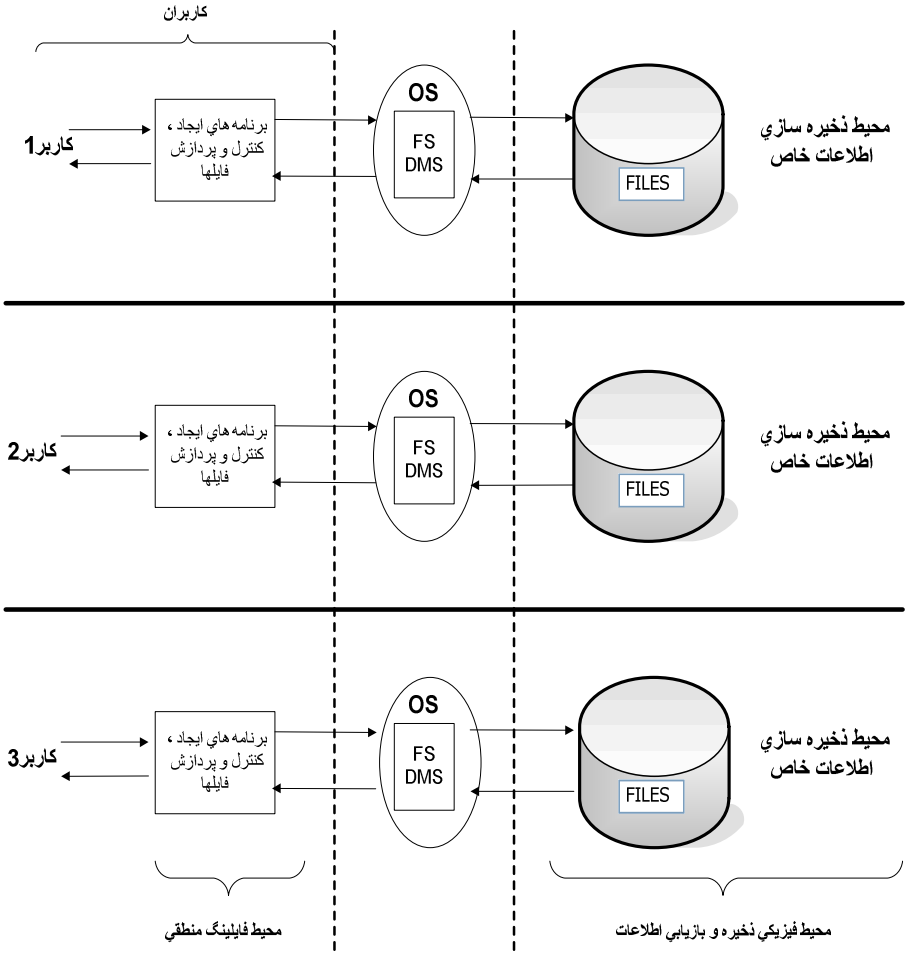
- تعدادی فایل اطلاعاتی برای ذخیره سازی فایلها طراحی می‌گردد. این فایلها معمولاً به صورت منفرد در سطح سیستم طراحی می‌گردند.
- برنامه مورد درخواست با استفاده از یک زبان برنامه نویسی تهیه می‌گردد. برنامه مورد نظر به همراه برنامه‌های مرتبط با فایل‌ها اطلاعاتی، مجموعه نرم‌افزاری سیستم را پوشش می‌دهند.
- یک مجموعه کامل شامل نرم‌افزار و سخت‌افزار و احتمالا مکانیزمی‌جهت برقراری ارتباط بین چند سخت‌افزار فراهم می‌گردد.
- مجموعه‌ای از تست‌ها جهت بررسی هر زیر سیستم و در نهایت بررسی کل سیستم اعمال می‌گردد.

با این وصف به راحتی می‌توان دریافت که در روش فایلینگ، داده‌ها در واقع چند مجموعه مجزا و نامجتمع (از لحاظ منطقی و فیزیکی) و تا حدود زیادی نامرتبط با هم و بدون مدیریت متمرکز خواهند بود. نمایی از روش فایلینگ در شکل ۱-۲ آمده است.

۲-۱-۲ مراحل کار در روش پایگاهی

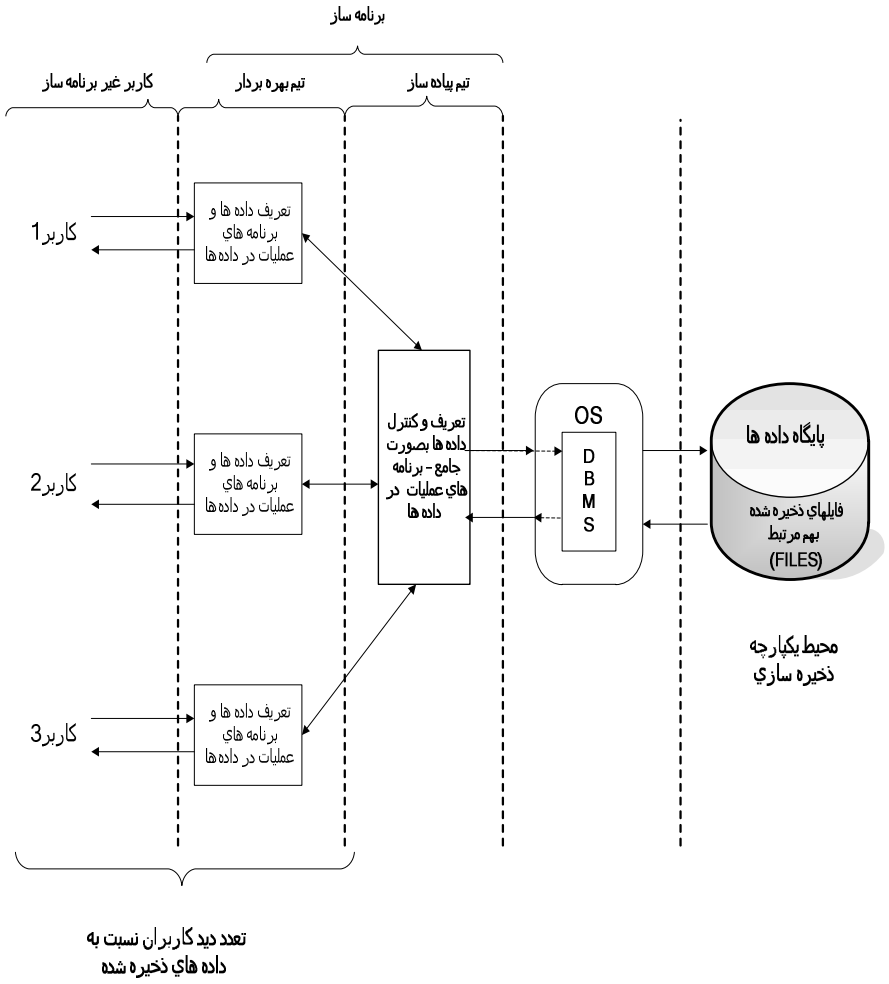
با عنایت به روش کار در سیستم فایلینگ، در ادامه مراحل کار در روش پایگاهی معرفی می‌گردد.

- کلیه نیازهای اطلاعاتی و پردازشی مجموعه مورد نظر بصورت یکپارچه مورد مطالعه و تحلیل قرار می‌گیرد. داده‌های مورد نظر، مدل سازی می‌گردند و مشخصات سیستم و وظایف آن بصورت جامع تعیین می‌گردند.
- یک یا چند پایگاه‌داده بعنوان سیستم مدیریت متمرکز انتخاب می‌گردد. طراحی‌های لازم در سطوح مختلف پایگاه‌داده تعریف و طراحی می‌گردند.
- واسط‌های مورد نظر برای کاربران واحدهای مختلف (و بعضا مدیران سیستم) طراحی می‌گردند.
- تست‌های لازم بر روی قسمت‌های مختلف سیستم اعمال می‌گردند.



شکل ۱-۲ نمای کلی از روش فایلینگ

با این وصف می‌توان مشاهده کرد که اساس روش فایلینگ بر مبنای ساختار اطلاعاتی یکپارچه طراحی شده است و مبنای کار مدیریت یک پارچه و متمرکز داده‌ها می‌باشد. برای درک بهتر موضوع شمایی از روش پایگاهی در شکل ۱-۳ آمده است.



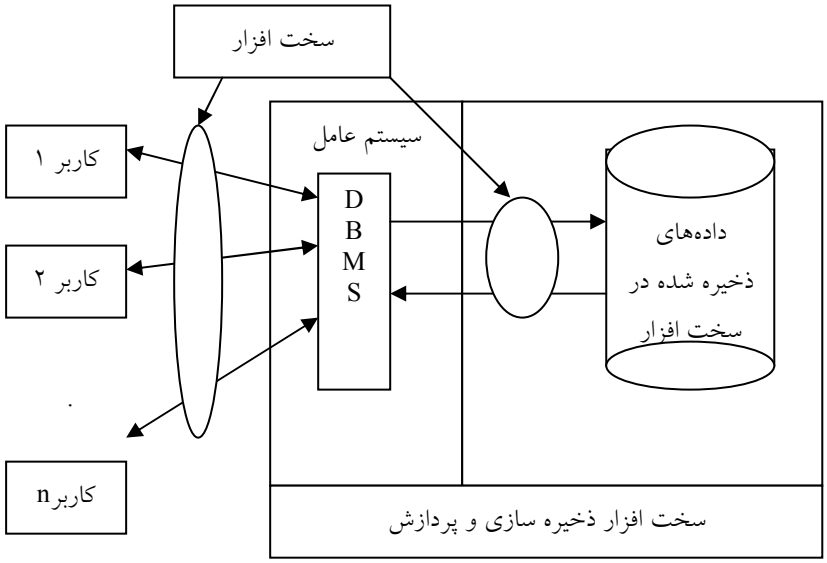
شکل ۱-۳ شمایی از روش پایگاهی

۲-۲ اجزاء پایگاه‌داده

همانگونه که در شکل زیر مشاهده می‌شود هر سیستم پایگاه‌داده از چهار جزء اساسی تشکیل می‌شود:

- داده‌ها
- سخت‌افزار

- نرم‌افزار
- کاربر



شکل ۱-٤ شمای تصویری ارتباط بین اجزاء پایگاه‌داده

شکل ۱-٤ یک نمای کلی از اجزاء معرفی شده به همراه نحوه ارتباط بین آنها را نشان می‌دهد. در ادامه هر یک از اجزاء مذکور مورد بحث و بررسی قرار خواهند گرفت.

۲-۲-۱داده‌ها

یک پایگاه‌داده گنجینه‌ای از داده‌ها است که در کل مجتمع شده[1] و به اشتراک گذاشته شده[2] است. منظور از مجتمع شدگی، اتحاد چندین فایل داده به صورت مجموعه‌ای است که آن را به نام پایگاه‌داده می‌شناسیم و به همین علت است که تمام آن افزونگی‌ها که در سیستم پردازش فایلها وجود داشت از میان می‌رود. منظور از مشترک بودن پایگاه‌داده‌ها این است که اطلاعات موجود در پایگاه‌داده‌ها بین

1. Integrated
2. Shared

استفاده کنندگان مختلف به اشتراک گذاشته می‌شود. داده‌های ذخیره شده در یک سیستم پایگاهی عبارتند از:

- داده‌های کاربران
- داده‌های سیستمی

۲-۲-۲ سخت‌افزار

یک پایگاه‌داده جهت استقرار به مجموعه مناسبی از تجهیزات سخت‌افزاری نیاز دارد. سخت‌افزارها به سه دسته تقسیم می‌گردند:

- سخت‌افزار ذخیره سازی داده‌ها
- سخت‌افزار پردازشگر
- سخت‌افزار برقرار کننده ارتباط

در ادامه به شرح هر یک از انواع سخت‌افزارها در محیط پایگاه‌داده‌ها خواهیم پرداخت:

۱- سخت‌افزار ذخیره سازی داده‌ها

منظور همان رسانه‌های ذخیره سازی خارجی است ولی باید دانست که رسانه اصلی ذخیره سازی دیسک است، سایر رسانه‌های ذخیره سازی مانند نوار مغناطیسی در محیط پایگاه‌داده‌ها کاربرد دارد ولی نه به عنوان رسانه اصلی، بلکه به صورت رسانه کمکی برای تولید نسخه‌های پشتیبان و فایلهای ثبت تراکنش‌ها یا فایلها رویدادنگاری.

۲- سخت‌افزار پردازشگر

منظور خود کامپیوتر است. لازم به ذکر است که برای پایگاه‌های داده با معماری خاص و یا بسیار حجیم، از انواع خاصی از کامپیوترها با سخت‌افزارهای خاص و نوع پردازش خاص استفاده می‌گردد.

۳- سخت‌افزار برقرار کننده ارتباط

منظور از سخت‌افزار برقرار کننده ارتباط، سخت‌افزار ارتباطی بین کامپیوتر و دستگاه‌های جنبی و نیز بین کامپیوترهاست. این امکانات به دو دسته تقسیم می‌شوند:

- امکانات محلی: برای ایجاد ارتباط بین کامپیوتر و دستگاه‌های جنبی آن در یک سایت به کار می‌رود.
- امکانات شبکه‌ای: درایجاد سیستم پایگاه‌داده‌های با معماری نا متمرکز به کار می‌رود.

چنانچه بخواهیم پایگاه‌های داده را بر اساس نگاه ارتباطی دسته بندی کنیم، انواع معماری پایگاه‌داده‌ها به شرح زیر خواهد بود:

- معماری مشتری- خدمت گزار[1]
- معماری متمرکز[2]
- معماری توزیع شده[3]
- معماری با پردازش موازی[4]
- معماری چند پایگاهی[5]
- معماری موبایل[6]

۲-۲-۳ نرم‌افزار

بین داده‌هایی که به صورت فیزیکی روی دستگاه‌های ذخیره سازی مناسب استقرار می‌یابد و پایگاه‌داده‌ها را به وجود می‌آورند و استفاده کنندگان یک لایه نرم‌افزاری قرار می‌گیرد که آن را سیستم مدیریت پایگاه‌داده (DBMS) می‌نامند. تمام تقاضا برای استفاده از اطلاعات پایگاه‌داده از طریق این سیستم سیر می‌شود و بازیابی داده‌ها روی سخت‌افزارهایی صورت می‌گیرد. نرم‌افزارها خود به دو دسته تقسیم

1. Client-Server Architecture
2. Centralized Architecture
3. Distributed Architecture
4. Parallel Processing Architecture
5. Multi Database Architecture
6. Mobile Architecture

می‌شوند:

- نرم‌افزار کاربردی
- نرم‌افزار سیستمی

در ادامه به شرح هر یک از انواع نرم‌افزارها خواهیم پرداخت:

۱- نرم‌افزار کاربردی

نرم‌افزاری است که کاربر باید برای تماس با سیستم بانک اطلاعاتی آماده کند. این نرم‌افزار به کمک یک زبان سطح بالا و یک زبان داده‌یی[1] و برخی تسهیلات نرم‌افزاری برای تماس با بانک ساخته می‌شود.

۲- نرم‌افزار سیستمی

این نوع نرم‌افزار از دو قسمت نرم‌افزار سیستمی خاص بانک که در اینجا به آن DBMS می‌گوییم و نرم‌افزار سیستمی‌عمومی (سیستم عامل) تشکیل شده است. (DBMS) در یک تعریف ساده، سیستمی‌است که به کاربران امکان می‌دهد عملیات مورد نظرشان را (مانند تعریف داده‌ها، بازیابی داده‌ها، ذخیره سازی داده‌ها) انجام دهند. DBMS که نرم‌افزاری پیچیده است میهمان یک سیستم عامل است و از امکانات سیستم عامل در انجام وظایفش استفاده می‌کند.

۲-۲-٤ کاربر

هر استفاده کننده از سیستم پایگاه‌داده‌ها را کاربر می‌گویند. کاربران پایگاه‌داده را می‌توان به سه گروه اساسی و متفاوت تقسیم نمود:

- برنامه نویسان کاربردی
- کاربران واقعی یا نهایی
- مدیر پایگاه‌داده‌ها

1. Data Language

۱- برنامه نویسان کاربردی [۱]

افرادی هستند که با اطلاعاتی که در مورد پایگاه‌داده پیدا می‌کنند می‌توانند برنامه‌های مناسبی جهت بروز کردن اطلاعات و یا استفاده از اطلاعات موجود در پایگاه‌داده تهیه نمایند.

۲- کاربران واقعی یا نهایی [۲]

افرادی هستند که با استفاده از امکاناتی که پایگاه‌داده در اختیار آنها قرار می‌دهد می‌توانند امور مربوط به خود و موسسه و سازمان را انجام دهند.

۳- مدیر پایگاه‌داده [۳]

مدیر پایگاه‌داده مسئولیت کنترل متمرکز سازمان بر داده‌های عملیاتی را بر عهده دارد. اگر بخواهیم وظایف DBA را به طور جزئی طراحی کنیم عبارتند از:

- تصمیم گیری در مورد داده‌هایی که در پایگاه‌داده نگهداری می‌شوند.
- تصمیم گیری در مورد ذخیره سازی و روش دستیابی است.
- ارتباط با کاربران برای حصول اطمینان از برآورده شدن نیازهای آنان.
- تعریف بررسی‌های مربوط به امنیت[۴] وجامعیت[۵] (صحت عملیات داده‌ها).
- تعریف استراتژی اخذ نسخه پشتیبانی و ترمیم[۶].
- نظارت بر عملکرد سیستم و پاسخگویی به نیازهای در حال تغییر.

لازم به ذکر است که امروزه با پیچیده تر شدن پایگاه‌های داده و حجیم تر شدن داده‌ها، معمولاً چند مدیر پایگاه‌داده (Junior DBA) تحت نظر یک مدیر اصلی پایگاه‌داده‌ها (Senior DBA) قرار گرفته و هر یک مسئولیت یک قسمت را بر عهده دارند.

---

1. Application Programmer
2. End User (Real User)
3. Database Administrator
4. Security
5. Integrity
6. Recovery

تمرینات

۱. مفاهیم داده، اطلاع و دانش را توضیح دهید ؟
۲. پایگاه‌داده را تعریف نمایید ؟
۳. عناصر اساسی یک پایگاه‌داده را نام ببرید ؟
٤. انواع تجهیزات سخت‌افزاری مورد نیاز برای یک پایگاه‌داده را در حالت کلی نام برده و توضیح دهید ؟
۵. انواع کاربران استفاده کننده از یک سیستم پایگاه‌داده را در حالت کلی نام برده و توضیح دهید ؟

فصل ۲

مدل سازی معنایی داده‌ها

هدف کلی

در این فصل با مفهوم مدل سازی معنایی داده‌ها آشنا خواهیم شد و دو روش مدل سازی موجودیت- ارتباط (ER)[1] و موجودیت- ارتباط توسعه یافته (ERR)[2] را مورد بحث و بررسی قرار خواهیم داد. در بحث مدل سازی به روش ER، با مفاهیم موجودیت، صفات و ارتباط آشنا شده و سپس نحوه رسم نمودار را در این روش مورد بررسی قرار خواهیم داد و پس از آن مشکلات روش ER را شرح خواهیم داد. در ادامه مدل سازی به روش ERR را توضیح داده و مطالبی در خصوص نحوه پوشش مباحث شی گرایی در این روش را برخواهیم شمرد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- مدل سازی معنایی داده‌ها
- مدل سازی به روش ER
- نوع موجودیت
- نمونه موجودیت
- حالات یک موجودیت

1. Entity Relationship (ER)
2. Extended (Enhanced) Entity Relationship (EER)

- صفات یک موجودیت
- ارتباط
- نوع ارتباط
- نمودار ER
- درجه و ماهیت نوع ارتباط
- حد کاردینالیتی
- مشکلات روش ER
- مدل سازی با روش EER
- تجزیه و ترکیب
- تخصیص و تعمیم
- زیر نوع‌های همپوشا و مجزا
- دسته بندی و وراثت
- تجمع

۱- مدلسازی معنایی داده‌ها[1]

کاربران پایگاه‌داده به طور معمول با داده‌های ذخیره شده در پایگاه‌داده سر و کار دارند که اصطلاحا به آنها داده‌های عملیاتی می‌گویند. یکی از نکات مهم در ذخیره سازی داده‌ها مدل سازی معنایی آنها می‌باشد. یعنی داده‌های ذخیره شدنی در پایگاه‌داده‌ها ابتدا باید در بالاترین سطح انتزاع مدل سازی معنایی شوند. حال ممکن است این سئوال به ذهن برسد که مدل سازی معنایی به چه معنی است.

مدل سازی معنایی داده‌ها عبارت است از ارائه مدلی از محیط عملیاتی[2] به کمک مفاهیمی‌مستقل از موضوعات مربوط به نمایش منطقی و فیزیکی داده‌ها. مدل سازی معنایی را در بعضی از کتب طراحی ادراکی[3] نیز می‌نامند. برای مدل سازی معنایی روش‌های مختلف وجود دارد. روش‌های کلاسیک رایج عبارتند از:

1. Semantic Data Modeling
2. Operational Environment
3. Conceptual Design

- روش موجودیت- ارتباط
- روش موجودیت- ارتباط توسعه یافته

در این بین روش موجودیت- ارتباط (ER) که از ابتدا به عنوان روش مدل سازی معنایی در پایگاه‌های داده رابطه‌ای مورد استفاده قرار گرفته است بیشتر مورد بررسی قرار خواهد گرفت.

۲- مدلسازی به روش ER

مدل سازی به روش ER یکی از ابزارهای مدل سازی معنایی در پایگاه‌داده است که در سال ۱۹۷٦ توسط آقای Chen در MIT ارائه گردید و به مرور زمان این ابزار پیشرفت کرد. تعریف Chen از بانک اطلاعات عبارت بود از تعدادی پدیده[۱] (موجودیت) دارای صفات[۲] مشخص و ارتباط[۳] بین پدیده‌ها. در این روش، سه مفهوم معنایی[٤] زیر وجود دارد:

- نوع موجودیت[٥]
- صفت[٦]
- نوع ارتباط[۷]

در ادامه هر یک از مفاهیم مذکور را مورد بحث و بررسی قرار خواهیم داد.

۲-۱ نوع موجودیت

نوع موجودیت عبارت است از مفهوم کلی هر آنچه که می‌خواهیم در مورد آن اطلاعی جمع‌آوری کنیم و دانش خود را در موردش افزایش دهیم. گاه به نوع موجودیت، نوع شیی هم می‌گوییم. لازم بذکر است که تشخیص انواع موجودیت‌ها

1. Entity
2. Attribute
3. Relation
4. Semantic Concept
5. Entity Type
6. Attribute
7. Relationship Type

در یک محیط کاری دشوار می‌باشد. به طور کلی یک نوع موجودیت دارای خصوصیات زیر می‌باشد یا به بیانی دیگر لازم است اطلاعات زیر در مورد هر نوع موجودیتی بدست آید:

- نام موجودیت (یا پدیده)
- معنای مشخص
- مجموعه‌ای از صفات
- مجموعه‌ای از نمونه‌ها
- حالت کنش گری یا کنش پذیری
- عدم وابستگی و یا وابستگی به یک نوع دیگر

برای مثال در یک سیستم اطلاعات اتومبیل انواع موجودیت‌ها می‌توانند شامل موجودیت اتومبیل، کشور سازنده (یا مصرف کننده) و... باشند. در یک سیستم آموزشی انواع موجودیت‌ها می‌توانند شامل موجودیت‌های درس، دانشجو، استاد، کلاس و... باشند.

۲-۱-۱ نمونه موجودیت [1]

تمام نمونه‌های مشخص (در مواردی متمایز) هر نوع موجودیت از یک محیط مشخص، مجموعه‌ای به نام مجموعه نمونه‌های[2] آن موجودیت را تشکیل می‌دهند هر نوع موجودیت خود می‌تواند دارای نمونه‌های مختلفی باشد که این نمونه‌ها از مشخصات نوع خود تبعیت می‌کنند. برای مثال موجودیت اتومبیل می‌تواند دارای نمونه‌هایی مانند پیکان، پژو، سمند، بنز و... باشد که هر کدام از اینها در واقع یک نمونه از موجودیت اتومبیل هستند.

۲-۱-۲ حالات یک موجودیت

یکی از نکات بسیار مهم در تعیین موجودیت‌ها تعیین مستقل یا وابسته بودن موجودیت‌ها می‌باشد. یک موجودیت ممکن است به دو صورت قوی (مستقل) یا

1. Entity Instance
2. Instances Set

ضعیف (وابسته) باشد. در ادامه به شرح حالات مذکور خواهیم پرداخت:

موجودیت قوی یا مستقل[1]

موجودیتی است که مستقل از هر نوع موجودیت دیگر و به خودی خود در یک محیط مشخص مطرح باشد. این نوع موجودیت وابستگی خاصی به سایر موجودیت‌های محیط عملیاتی ندارد.

موجودیت ضعیف یا وابسته[2]

موجودیت ضعیف موجودیتی است که وجودش وابسته به یک نوع موجودیت دیگر (موجودیت قوی) است. لازم بذکر است که اگر موجودیت قوی از مدل معنایی حذف گردد، وجود موجودیت ضعیف بی معنا بوده و موجودیت ضعیف نیز حذف می‌گردد.

نکته: مستقل بودن یا وابسته بودن موجودیت‌ها در محیط عملیاتی که می‌خواهیم برای آن پایگاه‌داده‌ای طراحی کنیم، تعیین می‌گردد و این موضوع ارتباطی به وابستگی و یا استقلال موجودیت در دنیای واقعی ندارد.

۲-۲ صفت

هر نوع موجودیت شامل مجموعه‌ای از صفات (مشخصات) مربوط به آن موجودیت است که حالت یا وضع آن موجودیت را توصیف می‌کند. این صفات خود دارای رده‌بندی‌های مختلفی هستند که در ذیل آمده است:

رده بندی صفت

صفات یک موجودیت بر حسب مفهوم آنها به دسته‌های زیر تقسیم می‌گردند:

- ساده[3] یا مرکب[4]
- تک مقداری[1] یا چند مقداری[2]

1. Strong
2. Weak
3. Single
4. Composite

- شناسه یا ناشناسه
- هیچ مقدار پذیر[3] یا هیچ مقدار ناپذیر
- ذخیره شده (واقعی[4] یا مبنا) یا مشتق[5]

در ادامه هر یک از رده بندی صفات (بر اساس آنچه که در بالا آمده است) مورد بحث و بررسی قرار می‌گیرند.

۲-۲-۱ صفت ساده یا مرکب

صفت ساده

صفتی است که مقدار آن تجزیه نشدنی می‌باشد، به این معنا که اگر مقدار آنرا به اجزائی تجزیه کنیم، مقادیر هر جزء فاقد معنا می‌باشد. مانند اسم درخت.

صفت مرکب

صفتی که از چند صفت ساده تشکیل شده و تجزیه شدنی می‌باشد. مانند صفت آدرس که می‌تواند شامل نام کشور، استان، شهر، منطقه و... باشد. لازم به ذکر است که می‌توان بر حسب نوع نیاز بجای استفاده از یک صفت مرکب، صفت مذکور را به چندین صفت ساده شکست.

۲-۲-۲ صفت تک مقداری یا چند مقداری

صفت تک مقداری

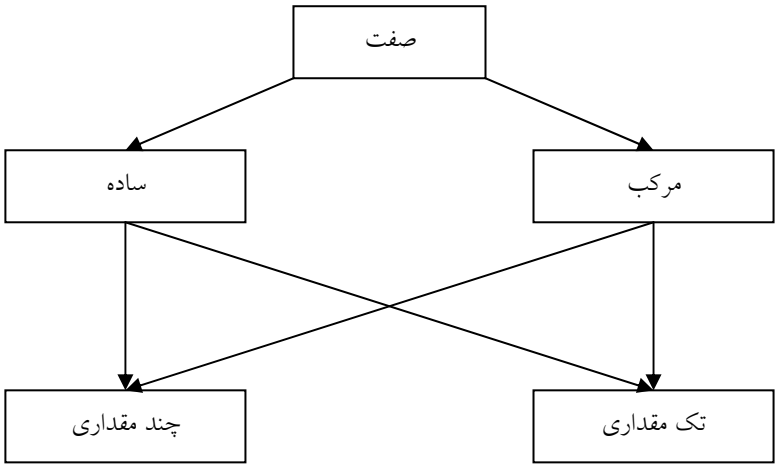
صفت تک مقداری صفتی است که حداکثر یک مقدار از میدان مقادیر را برای یک نمونه از یک نوع موجودیت می‌گیرد. به بیانی دیگر مقدار آن صفت برای نوع موجودیت مورد نظر، یک مقدار مشخص از میدان مقادیر مربوط به آن صفت می‌باشد. مثلا برای هر شخص یک کد ملی وجود دارد.

---

1. Single Valued
2. Multi Valued
3. Null Value
4. Real
5. Derived

صفت چند مقداری

صفتی است که بیش از یک مقدار از میدان مقادیر را برای حداقل یک نمونه از نوع موجودیت در بر می‌گیرد. مانند صفت مدرک تحصیلی برای یک شخص که ممکن است چند مقداری باشد، لیسانس، فوق لیسانس و....



شکل ۲-۱ نمودار ارتباط دهنده رده‌های مختلف صفات

۲-۲-۳ شناسه

صفت شناسه موجودیت، صفتی است که باید یکتایی مقدار داشته باشد و حتی الامکان طول مقادیرش کوتاه باشد.

مفهوم مقدار هیچ (هیچ مقدار)

این مفهوم از مفاهیم مدل رابطه‌ای است، مقدار هیچ یعنی مقدار ناشناخته، مقدار تعریف نشده. ممکن است مقدار یک صفت برای برخی از نمونه‌های یک نوع موجودیت، ناشناخته باشد. لازم به ذکر است که صفت شناسه موجودیت نمی‌تواند هیچ مقدار پذیر باشد.

۲-۲-٤ صفت واقعی یا مشتق

صفت واقعی (ذخیره شده)

صفت واقعی آن صفتی است که مقادیرش در پایگاه‌داده‌ها ذخیره شده باشد. باید توجه داشت که جنانچه صفت بعنوان شناسه نباشد، می‌تواند مقدار هیچ را نیز داشته باشد.

صفت مشتق

صفتی است که مقادیرش در پایگاه‌داده‌ها ذخیره شده نباشد. این صفت وجود خارجی ندارد. ولی از روی دیگر صفات قابل محاسبه است. مانند سن افراد که از روی تاریخ تولد قابل محاسبه است. هر صفت جنبه‌های زیر را دارد:

- نام
- معنا
- میدان (دامنه) مقادیر
- نوع مقدار
- طول مقدار (صریح یا صفتی)
- یک یا چند محدودیت ناظر به صفت

۲-۳ ارتباط[1]

یکی از مفاهیم بسیار مهم در مدل سازی معنایی داده‌ها مفهوم ارتباط یا بستگی است، به همین منظور به تعریف نوع ارتباط می‌پردازیم:

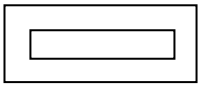
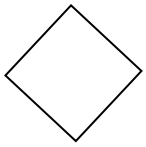
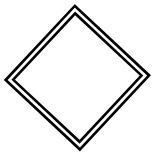
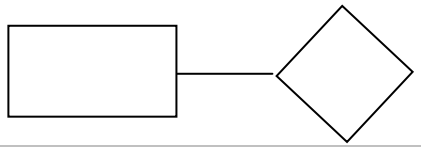
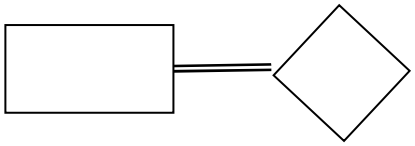
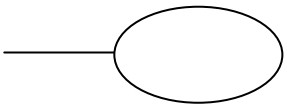
نوع ارتباط

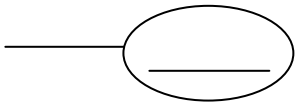
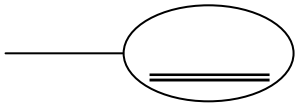
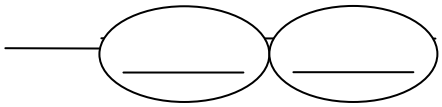
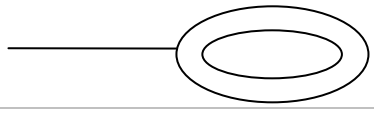
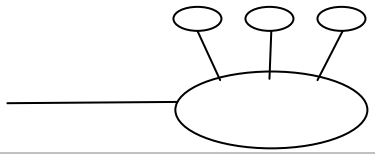
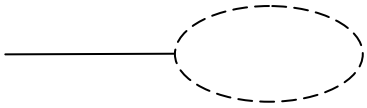
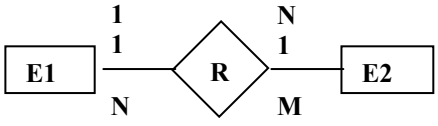
ئوع ارتباط عبارت است از تعامل[2] بین n نوع موجودیت ($n \geq 1$) و ماهیتا نوعی بستگی بین انواع موجودیت‌هاست. به تعبیری دیگر عملی است که بین انواع موجودیت‌ها جاری بوده، هست یا خواهد بود. هر نوع ارتباط یک معنای مشخص داشته و با یک نام بیان می‌شود.

1. Association
2. Interaction

۳- نمودار ER

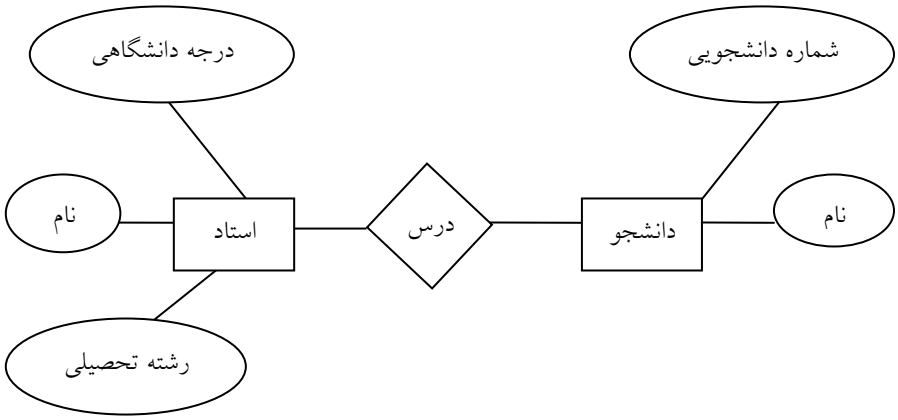
نمودار ER در واقع نموداری است که در آن سه مفهوم اساسی مدل ER یعنی موجودیت، صفت و نوع ارتباط نمایش داده می‌شوند. در واقع نمودار ER اولین طرح پایگاه‌داده‌ها و مدل کلی آن در بالاترین سطح انتزاع می‌باشد.

مفهوم نماد در مدل ER	نماد در مدل ER
نوع موجودیت: پدیده‌های موجود که وجود خارجی دارند (Entity) را با مستطیل نمایش می‌دهند.	
نوع موجودیت ضعیف (وابسته)	
نوع ارتباط (Relationship): عامل ارتباط موجودیت‌ها را با لوزی نمایش می‌دهند.	
نوع ارتباط موجودیت ضعیف با قوی	
مشارکت نوع موجودیت در نوع ارتباط	
مشارکت الزامی	
صفت: صفت‌های هر موجودیت را توسط اشکال بیضی به آن متصل می‌نماییم.	

صفت شناسه اول	
صفت شناسه دوم (در صورت وجود)	
صفت شناسه مرکب (مثلا درختی)	
صفت جند مقداری	
صفت مرکب	
صفت مشتق (مجازی یا محاسبه شدنی)	
چندی ارتباط	

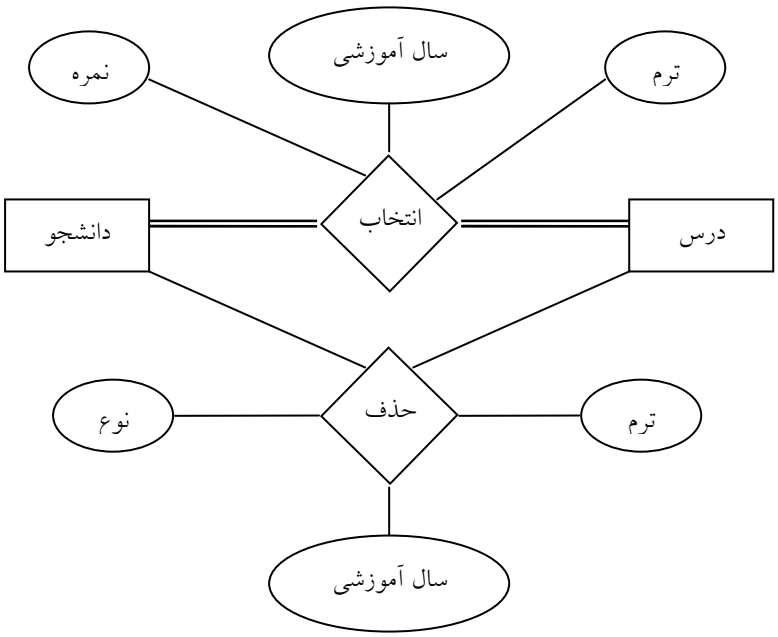
شکل ۲-۲ نمادهای رسم نمودار ER

هر نمودار ER پاسخگوی مجموعه مشخصی از نیازهای کاربران است و بدیهی است که این نمودار با تغییر و رشد نیازهای کاربران تغییر یافته و توسعه داده می‌شود. برای رسم این نوع نمودار نیاز به نمادهایی است که این نمادها در جدول ذیل آمده است. لازم به ذکر است که تمامی‌نمادهای مورد استفاده برای رسم نمودار ER در رسم نمودار EER نیز معتبر بوده و مورد استفاده قرار می‌گیرند. برای درک بهتر نحوه استفاده از نمادها در رسم نمودار ER، به نمودار زیر که بیانگر وضعیت یک سیستم دانشجویی است توجه نمایید:



شکل ۲-۳ یک نمونه از نمودار ER برای نمایش سیستم دانشجویی

یکی از نکات مهم در رسم نمودار ER، معنای ارتباط بین عناصر است. بدین صورت که معنای ارتباط ((انتخاب)) با معنای ارتباط ((حذف)) فرق دارد. در ادامه نمونه‌ای از معنای ارتباط بین موجودیت‌ها ترسیم شده است.



شکل ۲-٤ نمونه‌ای از معنای ارتباط بین عناصر

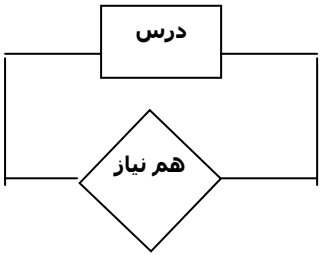
۳-۱ درجه نوع ارتباط

درجه ارتباط بستگی به تعداد موجودیتها مرتبط به هم دارد. به بیانی دیگر تعداد شرکت کنندگان در یک نوع ارتباط را درجه آن ارتباط می‌گویند.

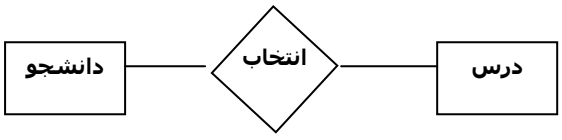
تعداد موجودیت‌های شرکت کننده در ارتباط	درجه	
	لاتین	فارسی
۱	Unary	یگانی
۲	Binary	دوگانی
۳	Ternary	سه گانی
…	…	…
N	n-ary	چند گانی

در ادامه هر یک از درجات نوع ارتباط بصورت شماتیک رسم شده و توضیح داده می‌شوند:

یگانی[1]: زمانی که یک نوع ارتباط بین یک نوع موجودیت و خودش برقرار باشد.

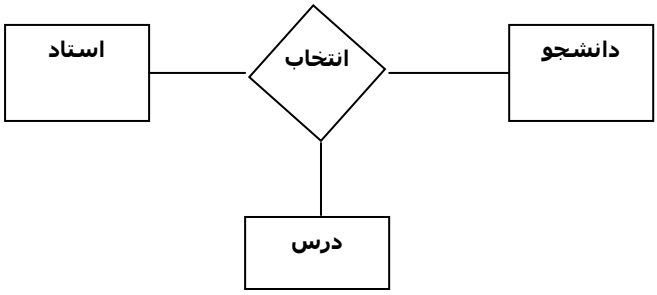


دوگانی[2]: ارتباط بین دو موجودیت

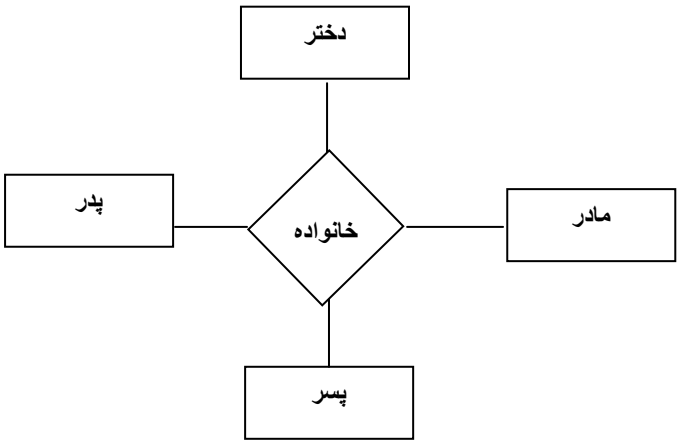


1. Unary
2. Binary

سه گانی[1]: ارتباط بین سه موجودیت



چند گانی: ارتباط بین چند موجودیت



توجه: چون معمولاً مادر و پدر و خواهر و برادر یک عضو محسوب می‌شوند، این نوع ارتباط منطقاً درست نیست. شکل مذکور فقط بعنوان نمونه است.

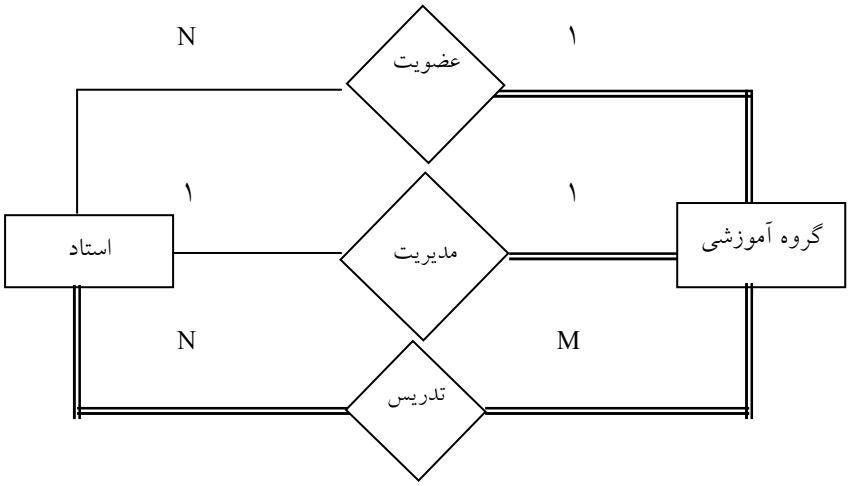
۳-۲ ماهیت نوع ارتباط (اتصال)

اگر دو موجودیت A وB را در نظر بگیریم این دو موجودیت به یکی از سه حالت زیر با هم ارتباط دارند.

- **یک به یک**[2]: یک نمونه از موجودیت A حداکثر با یک نمونه از موجودیت B ارتباط دارد و بر عکس. به اختصار 1:1 نمایش می‌دهند.

1. Ternary
2. One To One

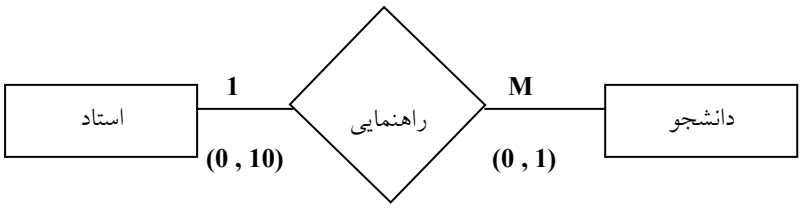
- **یک به چند**[1]: یک نمونه از A با n نمونه از B ارتباط دارد ولی یک نمونه از B حداکثر با یک نمونه از A ارتباط دارد (n>1). به اختصار 1: N نمایش می‌دهند.
- **چند به چند**[2]:یک نمونه از A با n نمونه از B ارتباط دارد و بر عکس (n>1). به اختصار M: N نمایش می‌دهند.



شکل ۲-۵ نمایش چندی ارتباط

۳-۳ حد کاردینالیتی[3]

حد کاردینالیتی، حداقل و حداکثر ارتباط بین دو موجودیت را می‌رساند. برای درک بهتر این مفهوم مثال زیر ارائه می‌گردد:



شکل ۲-٦ نمونه‌ای از نمایش حد کاردینالیتی

1. One To Many
2. Many To Many
3. Cardinality

عبارات ذکر شده در بالا و پایین ارتباط بین عناصر، هر یک دارای مفاهیمی‌هستند. در ادامه دو عبارت (10 , 0) و (1 , 0) مورد بررسی قرار گرفته‌اند:

- (۱۰و۰) بدین معنی است که یک استاد ممکن است حداکثر ۱۰ دانشجو را راهنمایی کند و یا هیچ دانشجویی برای راهنمایی نداشته باشد (این استاد پروژه ارائه نکرده است).
- (۱و۰) بدین معنی است که یک دانشجو حداکثر می‌تواند یک استاد راهنما داشته باشد و اگر پروژه اخذ نکرده باشد هیچ استاد راهنمایی ندارد.

٤- مشکلات روش ER

در نتیجه درک نادرست و تفسیر ناصحیح از معنای بعضی ارتباطات در مدل سازی داده‌ها مشکلاتی موسوم به دامهای پیوندی[1] نمایان می‌گردد. این دامهای پیوندی عبارتند از:

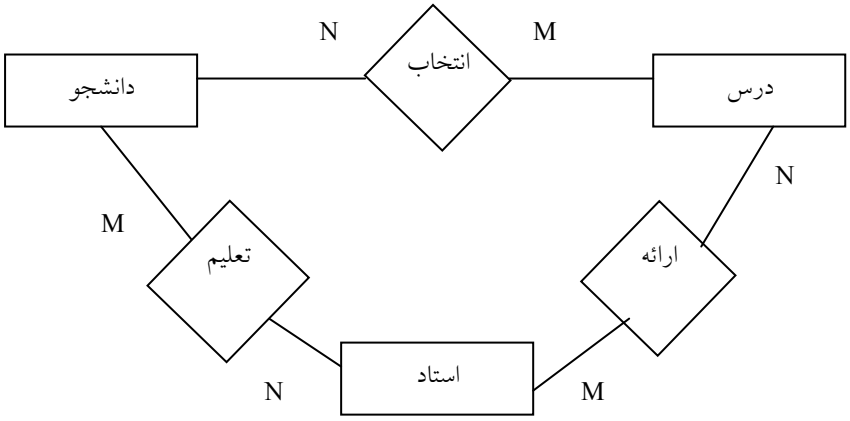
- دام حلقه‌ای[2]
- دام چند شاخه (چتری)[3]
- دام گسل (شکاف)[4]

در ادامه برای درک بهتر مفهوم دام ها، هر یک از انواع دام ها در روش ER با ذکر مثال شرح داده خواهند شد.

٤-١ دام حلقه‌ای

این دام وقتی ایجاد می‌شود که با داشتن مثلا سه ارتباط دو موجودیتی، وجود یک ارتباط سه موجودیتی را نتیجه گیری کنیم، ولی این استنتاج درست نباشد.

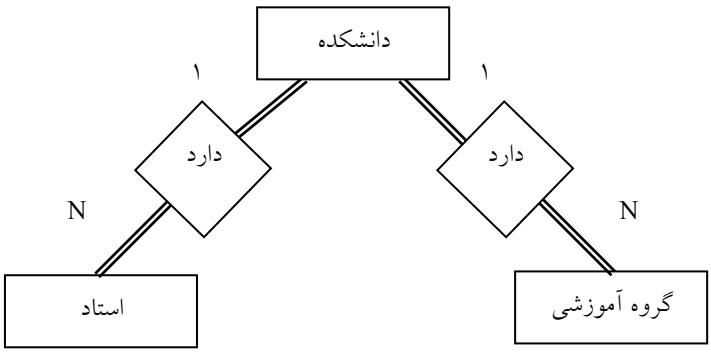
1. Connection Traps
2. Loop Trap
3. Fan Trap
4. Chasm Trap



شکل ۲-۷ نمونه‌ای از دام حلقه‌ای

٤-۲ دام چند شاخه‌ای

این نوع دام وقتی‌ایجاد می‌شود که بین یک نوع موجودیت E و موجودیت‌های F و G ارتباط 1: N با مشارکت الزامی‌وجود داشته باشد، ولی ارتباط بین F و G دیده نشده باشد. در این صورت نمی‌توان وجود ارتباط بین F و G را بدست آورد.



شکل ۲-۸ نمونه‌ای از دام شاخه‌ای

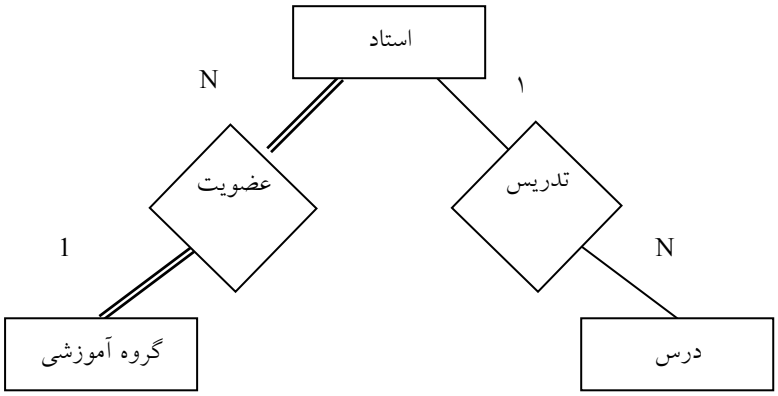
٤-۳ دام گسل

این نوع دام وقتی‌ایجاد می‌شود که بین دو نوع موجودیت E و F یک ارتباط 1: N و مشارکت الزامی‌وجود داشته باشد، ولیF با نوع موجودیت G ارتباط 1:N با مشارکت

غیر الزامی‌داشته باشد. در این شرایط نمی‌توان تمام اطلاع‌های دو موجودیتی بین E و G را بدست آورد. اگر چنین فرضی در نظر گرفته شود، دچار دام گسل شده‌ایم.

۵- مدل سازی با روش EER

همانطور که می‌دانیم روش ER دارای نقاط ضعفی بود. این نقاط ضعف بیشتر زمانی نمایان می‌شد که می‌خواستیم یک سیستم شیء گرا را مدل سازی نماییم. بطور کلی نقاط ضعف روش ER که در روش EER رفع شده‌اند به شرح ذیل بودند.



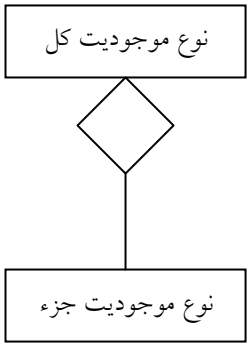
شکل ۲-۹ نمونه‌ای از دام گسل

۵-۱ تجزیه[1] و ترکیب[2]

فرایند تقسیم یک شیء کل به اجزاء تشکیل دهنده آن را تجزیه گویند. بدیهی است که در فرایند تجزیه، شیء کل و اجزاء آن هر یک دارای صفات، ساختار و رفتار خاص خود می‌باشند.

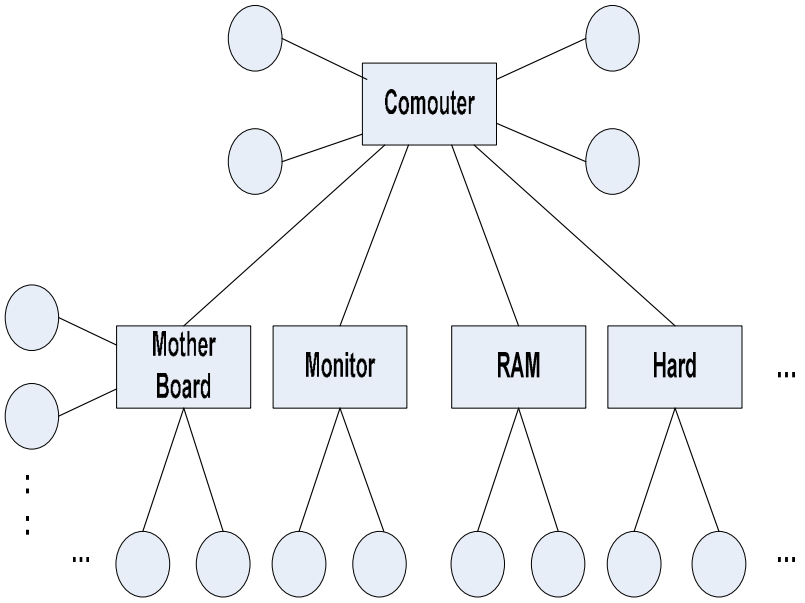
ترکیب، عکس عمل تجزیه است که در آن با داشتن تعدادی نوع موجودیت، یک نوع موجودیت جدید راایجاد می‌کنیم.

1. Decomposition
2. Composition



شکل ۲-۱۰ نماد ارتباط " جزئی است از... "

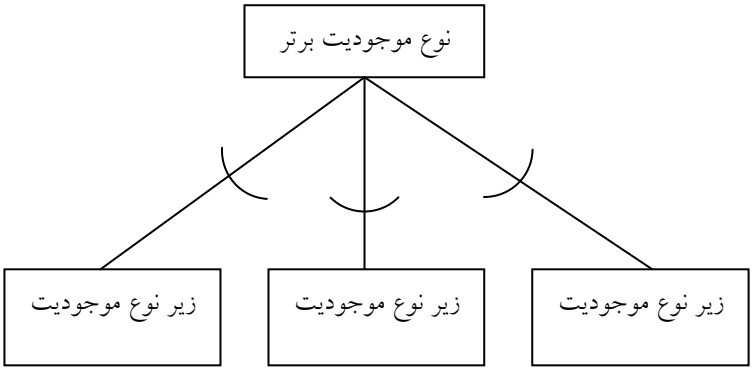
در روش ERR ارتباط بین شیء کل و اشیاء جزء را ارتباط " ***جزئی است از...*** " گویند. نماد مورد استفاده برای نمایش ارتباط نوع موجودیت کل و نوع موجودیت جزء به شکل زیر می‌باشد:



شکل ۲-۱۱ نمونه‌ای از تجزیه و ترکیب موجودیت‌ها

۲-۵ تخصیص[1] و تعمیم[2]

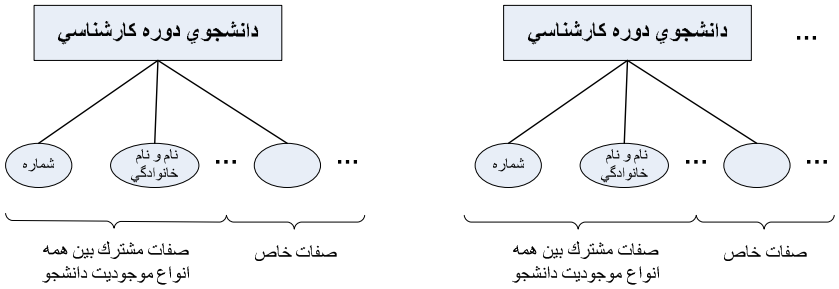
تخصیص فرایندی است که طی آن نمونه‌های یک نوع موجودیت برتر [3] (زبر نوع) را بر اساس یک یا چند صفت خاصه آن موجودیت برتر تشخیص می‌دهیم. لازم به ذکر است که یک نوع موجودیت می‌تواند دارای یک یا چند زیر نوع موجودیت [4] نیز باشد. ارتباط بین موجودیت برتر و زیر نوع‌های آن را ارتباط "***گونه‌ای است از...*** " می‌نامیم. نماد مورد استفاده برای نمایش این نوع ارتباط در شکل زیر آمده است.



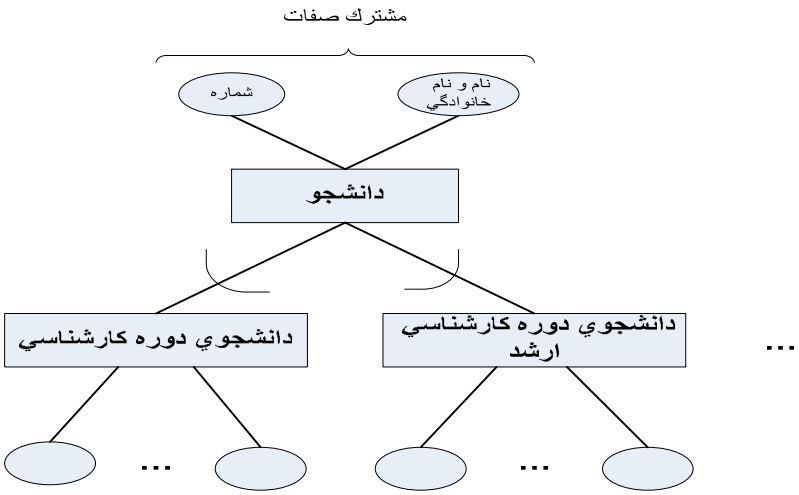
شکل ۲-۱۲ نماد ارتباط " گونه‌ای است از... " (تخصیص)

تعمیم عکس عمل تخصیص است که در آن با داشتن زیر نوع‌های یک نوع موجودیت و تعیین صفات مشترک بین آنها، یک مجموعه صفات را برای نوع موجودیت برتر در نظر می‌گیریم.

1. Specialization
2. Generalization
3. Super Entity Type
4. Sub Entity Type



شکل۲-۱۳ (الف) نمونه‌ای از حالت تعمیم در سیستم‌های دانشجویی

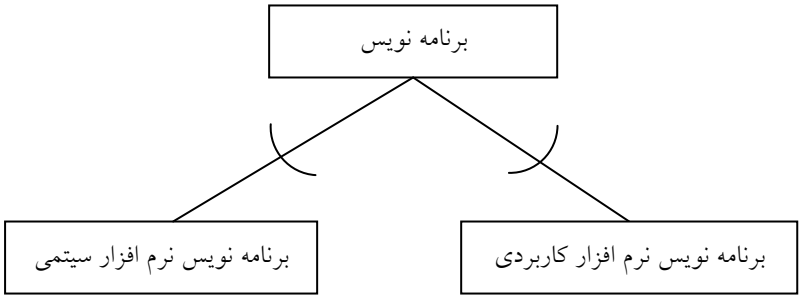


شکل۲-۱۳ (ب) نمونه‌ای از حالت تعمیم در سیستم‌های دانشجویی

۵-۳ زیر نوع‌های همپوشا[1] و مجزا[2]

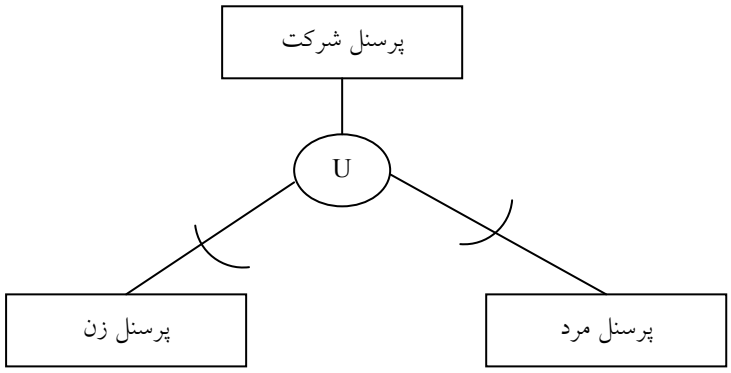
نمونه‌های یک نوع موجودیت برتر بر اساس صفات خاصه مشترک و مجزا در دسته‌های خاص خود دسته بندی می‌شوند. حال ممکن است یک نمونه موجودیت در دو دسته قابل دسته بندی باشد در چنین شرایطی به این نوع موجودیت‌ها، موجودیت‌های همپوشا (مشترک) گویند.

1. Overlap
2. Disjoint



شکل ۲-۱٤ نمونه‌ای از زیر نوع‌های همپوشا

در عین حال ممکن است یک نوع موجودیت فقط در یک دسته بندی از زیر نوع موجودیت‌ها قابل دسته بندی باشد. به این زیر نوع موجودیت‌ها، موجودیت‌های مجزا می‌گویند.



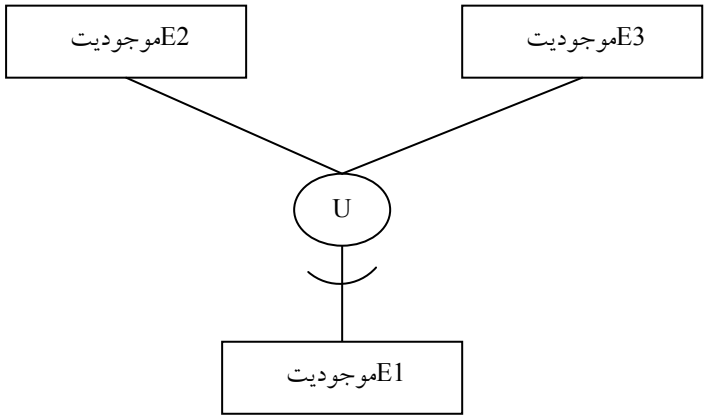
شکل ۲-۱۵ نمونه‌ای از زیر نوع‌های مجزا

٤-۵ دسته بندی [1] و وراثت [2]

یک زیر نوع موجودیت می‌تواند زیر نوع بیش از یک نوع موجودیت برتر باشد. که در این شرایط بعضی از خواص خود را از یک نوع موجودیت برتر و بعضی دیگر

1. Categorization
2. Inheritance

از صفات خاصه خود را از یک نوع موجودیت برتر دیگر به ارث می‌برد. این موجودیت‌های برتر می‌توانند از یک نوع باشند که در این شرایط دارای شناسه‌های یکسان هستند. ولی در شرایطی که موجودیت‌های برتر از یک نوع نباشند، در واقع وراثت چندگانه [1] رخ داده است. به این زیر نوع‌ها در اصطلاح دسته[2] (طبقه) می‌گویند و در بعضی از کتاب‌ها به این زیر نوع اصطلاحا نوع اجتماع [3] می‌گویند. شکل زیر نحوه نمایش وراثت چند گانه را نشان می‌دهد:

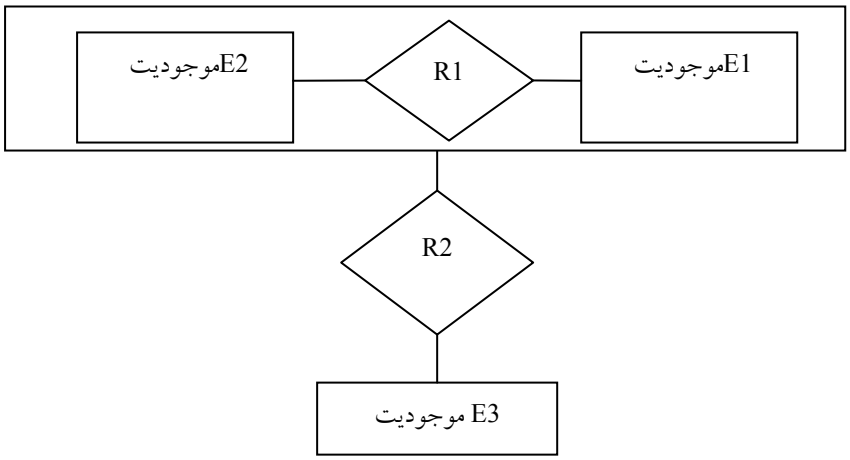


شکل ۲-۱۶ شمای نمایش وراثت چند گانه و دسته بندی

۵-۵ تجمع[4]

تجمع بدین معنا است که یک نوع موجودیت جدید را بر اساس دو یا چند موجودیت مرتبط با یکدیگر، به صورت یکپارچه در یک نوع موجودیت واحد ارائه نماییم. بدیهی است که این نوع موجودیت واحد خود می‌تواند با نوع موجودیت‌های دیگر نیز در ارتباط باشد. شکل زیر نحوه نمایش تجمع را نشان می‌دهد:

1. Multiple Inheritance
2. Category
3. Union Type
4. Aggregation



شکل ۲-۱۷ نمایش تجمع

٦-روش مدل سازی شیء UML[1]

با مطرح شدن الگوهای شیء گرایی که هر موجودیت در جهان واقع را شامل داده‌های آن موجودیت و عملکردهای مرتبط با آن توصیف می‌کرد، طراحان سیستم روش‌های ER و EER را ناکارآمد تشخیص داده و در جستجوی تعریف یک متودولوژی جدید بودند تا بتوانند عملکرد هر موجودیت را نیز به نمایش در آورند. از اینرو گروه OMG[2] روشی استاندارد بنام UML را برای‌ایجاد سیستم‌های کاربردی معرفی کردند و این روش را درایجاد سیستم‌های کاربردی مورد استفاده قرار دادند. مهمترین خصوصیت این روش، ایجاد امکانی برای‌ایجاد و نمایش اشیاء جهان واقع بصورت تصویری است. یکی از نکات بسیار مهم در استفاده موثر از روش UML، وجود دانش کافی در مفاهیم شیء گرایی است. به بیانی دیگر بدون دانستن مفاهیم شیء گرایی، امکان استفاده موثر و بجا از روش UML میسر نخواهد شد.

٦-١ مفاهیم اصلی

اساس روش UML بر مبنای نمودار (دیاگرام) می‌باشد. این روش از نمودارهای

1. Object Modeling
2. Object Management Group

برای نمایش مدل سازی و طراحی نرم‌افزار استفاده می‌کند. چند نمونه از مهمترین نمودارها در این روش به شرح زیر می‌باشند:

- نمودار کلاس [۱]
- نمودار چرخه حیات موجودیت[۲]
- نمودار مورد استفاده[۳]
- نمودار فعالیت[۴]
- نمودار پیاده سازی[۵]

مهمترین نمودار در این روش نمودار کلاس می‌باشد. این نمودار مجموعه‌ای از موجودیت‌ها و عملیات (پردازش) مرتبط با موجودیت‌ها را مدل سازی می‌کند. عملیات در واقع رفتار شیء[۶] را نشان می‌دهد و به بیانی دیگر رویداد[۷]های مرتبط با شیء را به نمایش در می‌آورد. مفاهیم اصلی در مدلسازی با این روش عبارتند از:

- کلاس
- صفت
- بستگی[۸]

درروش UML دوگونه ارتباط بین رده‌ها وجود دارد که این دونوع ارتباط عبارتند از

- بستگی
- تجمیع

که در این بین مفهوم بستگی همان ارتباط بین کلاس‌ها است. **تجمیع نیز** عبارت است از ارتباط بین یک شیء کل و شیءهای جزء تشکیل دهنده آن. در ادامه برای درک بهتر تناظر بین مفاهیم در دو روش UML و EER جدول زیر ازائه می‌گردد:

1. Class Diagram
2. Entity Life Cycle Diagram
3. Use Case Diagram
4. Activity Diagram
5. Implementation Diagram
6. Object Behavior
7. Event
8. Association

مفهوم در UML	مفهوم در EER
کلاس	نوع موجودیت
شیء	نمونه موجودیت
صفت	صفت
بستگی	ارتباط
پیوند[1]	نمونه ارتباط
بستگی انعکاسی[2]	ارتباط بازگشتی
بستگی مقید[3]	نوع موجودیت ضعیف
میدان ساختمند[4]	صفت مرکب
صفت پیوند	صفت ارتباط
چندی بستگی[5]	درجه ارتباط

۶-۲ نحوه نمایش مفاهیم

کلاس: برای نمایش یک کلاس از مربع یا مستطیل استفاده می‌شود. در این حالت مربع به سه قسمت مجزا تقسیم شده و در این قسمت‌ها موارد زیر نوشته می‌شوند

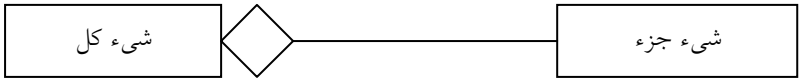
- نام کلاس
- نام صفات کلاس
- نام پردازشهای کلاس

نام کلاس
نام صفات کلاس
نام پردازشهای کلاس

بستگی: بستگی بین دو کلاس بصورت یک خط متصل کننده دو کلاس نمایش داده می‌شودو نام ارتباط روی خط نوشته می‌شود.

1. Link
2. Reflexive Association
3. Qualified Association
4. Structural Domain
5. Multi Association

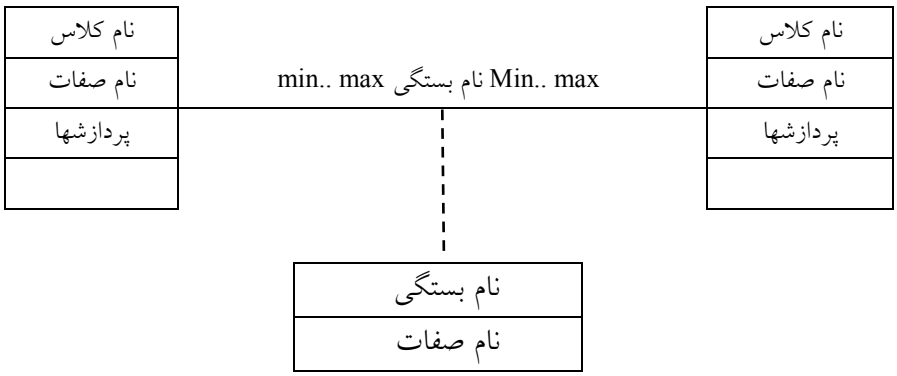
تجمیع: تجمیع بصورت زیر نمایش داده می‌شود



چندی ارتباط: چندی ارتباط بصورت min… max نوشته می‌شود
صفت چند مقداری: صفت چند مقداری به صورت یک کلاس جداگانه نشان داده می‌شود، ولی قسمت پردازش را همراه ندارد.
میدان: نام میدان معمولاً بعد از نام صفت نوشته می‌شود و بین این دو نام از علامت: استفاده می‌شود بصورت زیر

نام میدان: نام صفت

صفت پیوند: صفت **پیوند** در یک مربع یا مستطیل نوشته می‌شود و با خط چین به خط نشان دهنده بستگی متصل می‌شود. نام پیوند و نام صفات پیوند در دو قسمت این مربع نوشته می‌شوند. شکل زیر نحوه نمایش صفت پیوند را نشان می‌دهد



۶-۳ خصوصیات کلی روش مدل سازی معنایی داده‌ها

هر روش مدل سازی معنایی داده‌ها باید حداقل دارای خصوصیات زیر باشد

- گویایی
- سادگی مفاهیم

- ایجاز
- گسترش پذیری
- صوری بودن
- قابلیت نمایش نموداری
- جامع بودن مفاهیم
- قابلیت نمایش ساختار حالت و رفتار نوع موجودیت

باید به این نکته توجه داشت که بعضی از خصوصیات ذکر شده مانند تجمیع و ایجاز ممکن است با یکدیگر مغایرت داشته باشند. در این صورت وجود یکی از این خصوصیات کافی است.

تمرینات

۱. مدل سازی معنایی داده را توضیح دهید ؟
۲. انواع موجودیت‌ها را نام برده و توضیح دهید ؟
۳. صفت را تعریف کرده و رده‌های مختلف آن را نام ببرید ؟
٤. حالات مختلف ارتباط بین دو موجودیت را نام برده و توضیح دهید ؟
۵. مشکل دام حلقه‌ای در روش ER را توضیح دهید ؟
٦. مشکل دام چند شاخه‌ای را در روش ER توضیح دهید ؟
۷. مشکل دام گسل را در روش ER شرح دهید ؟
۸. مفاهیم تجزیه و ترکیب در روش EER را شرح دهید ؟
۹. مفاهیم تخصیص و ترمیم در روش EER را شرح دهید ؟
۱۰. مفاهیم وراثت و دسته بندی در روش EER را شرح دهید ؟
۱۱. مفهوم تجمع در روش EER را شرح دهید ؟

فصل ۳

معماری پایگاه‌داده

هدف کلی

در این فصل معماری سه سطحی پایگاه‌داده مورد بحث و بررسی قرار خواهد گرفت. در این راستا سطوح خارجی، ادراکی و داخلی بصورتی دقیق‌تر مورد ارزیابی قرار خواهند گرفت. در ادامه با سایر اجزاء پایگاه‌داده آشنا خواهیم شد و زبان‌های مختلف مورد استفاده درون پایگاه‌داده نیز بررسی خواهند شد.

هدف رفتاری

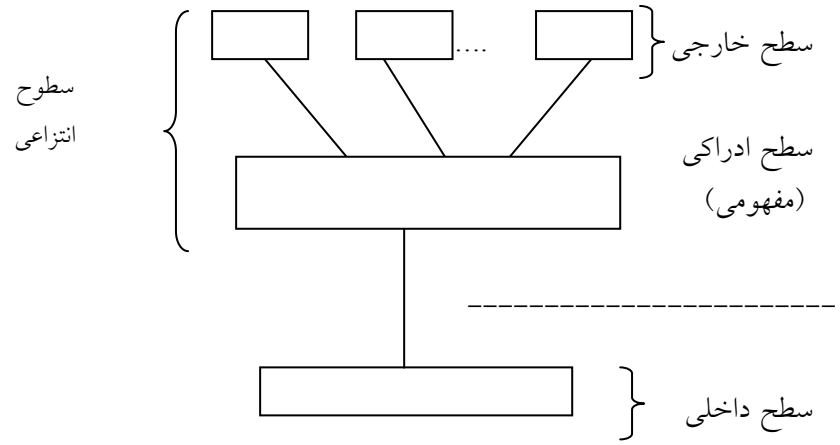
در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- معماری سه سطحی پایگاه‌داده
- دید (نمای) ادراکی (مفهومی)
- دید (نمای) خارجی
- دید (نمای) داخلی
- کاربر
- زبان میزبان
- زبان داده‌ای فرعی
- انواع زبان داده‌ای

۱- معماری سه سطحی پایگاه‌داده

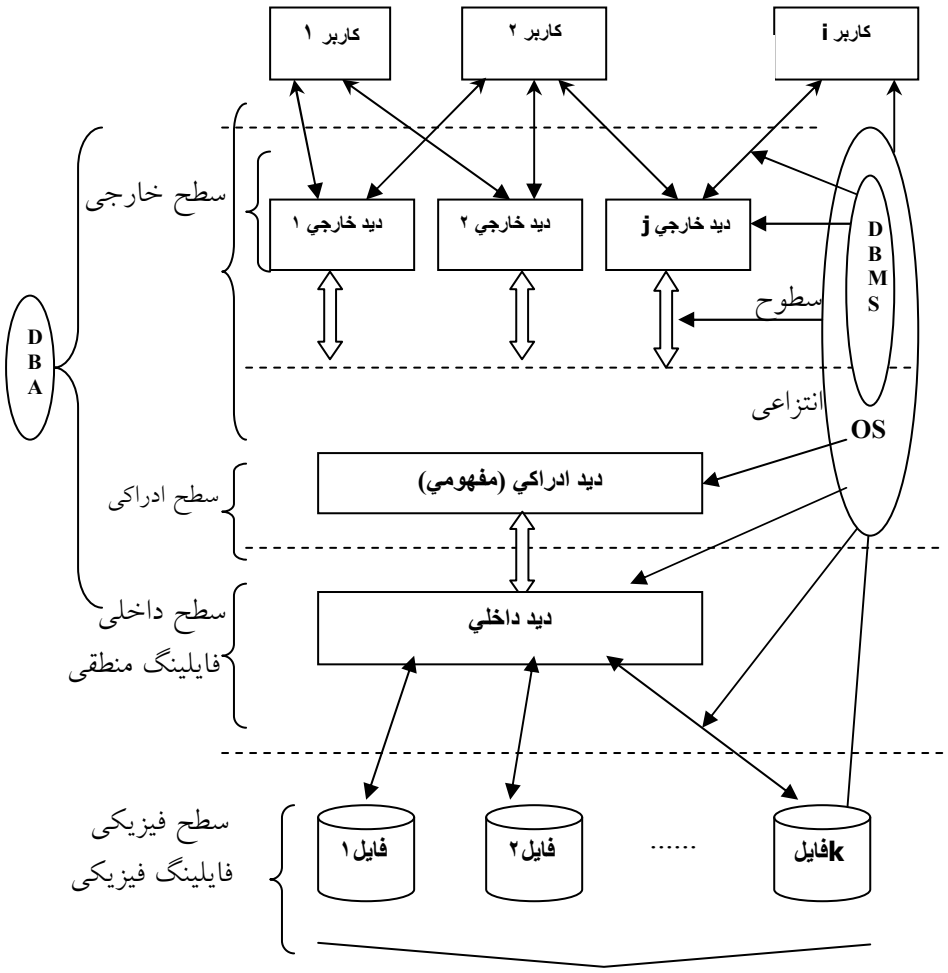
از آنجا که در چگونگی معماری "پایگاه‌داده‌ها" حداقل در سالهای آغازین ایجاد این تکنولوژی، بین کارشناسان اتفاق نظر نبود، لذا ANSI/SPARC با توجه به اهمیت محیط انتزاعی، معماری پایگاه‌داده‌ها را (به عنوان یک معماری استاندارد) پیشنهاد کرد. این معماری به معماری سه سطحی موسوم است.

باید توجه داشت که این سه سطح، اساساً سه سطح "تعریف داده‌ها" هستند: دو سطح خارجی و ادراکی در محیطی انتزاعی هستند و سطح سوم که همان سطح داخلی است در محیط فایلینگ منطقی قرار گرفته است. شکل ۳-۱ نمای ساده‌ای از معماری سه سطحی را نمایش می‌دهد.



شکل ۳-۱ نمای ساده شده معماری سه سطحی

واضح است که سطح فیزیکی یک لایه پایین تر از سطح داخلی قرار می‌گیرد. در بعضی از پایگاه‌های داده، سطح فیزیکی نیز جزئی از معماری محسوب می‌گردد. نمای کاملتر معماری پیشنهادی ANSI برای پایگاه‌داده در شکل ۳-۲ نشان داده می‌شود.



شکل ۳-۲ معماری پایگاه‌داده‌ها

در این معماری علاوه بر سه سطح، اجزای دیگری هم دیده می‌شود که در واقع جزء "سیستم پایگاه‌داده‌ها" هستند. در اینجا سه سطح و نیز اجزاء دیگر را نام می‌بریم:

- کاربر
- زبان میزبان

- زبان داده‌ای فرعی
- دید خارجی
- دید ادراکی
- دید داخلی
- فایلهای فیزیکی
- سیستم مدیریت پایگاه‌داده‌ها
- مدیر پایگاه‌داده‌ها

در ادامه برای درک بهتر موضوع، سطوح سه گانه مورد بررسی قرار می‌گیرند:

۲- شرح سطوح سه گانه

همانطور که در سمت چپ تصویر بالا مشاهده می‌کنید سه سطح (دید) مختلف در معماری پایگاه‌دادهها ارائه شده است. این سطوح عبارتند از:

- دید (نمای) ادراکی (مفهومی)
- دید (نمای) خارجی
- دید (نمای) داخلی

۲-۱ دید (نمای) ادراکی (مفهومی)

- دید یا نمای ادراکی در واقع همان دید طراح پایگاه‌داده‌ها نسبت به داده‌های ذخیره شدنی در پایگاه‌داده است.
- این دید یک دید جامع (سرتاسری) بوده و تمام نیازهای کاربران در محیط عملیاتی را در بر می‌گیرد.
- این دید در یک محیط انتزاعی مطرح است: بنابرین مبتنی است بر یک ساختار داده‌ای مشخص (از یک مدل داده‌ای که انتزاع لازم را تامین می‌کند).
- این دید، با ساتفاده از عناصر ساختاری اساسی همان ساختار داده‌ای، طراحی می‌شود.

- این دید باید (پس از طراحی طبعا) توصیف شود. به وصف یا شرح دید ادراکی، شمای ادراکی می‌گوییم. شمای ادراکی نوعی "برنامه" است حاوی دستورات "تعریف داده‌ها" و "کنترل داده‌ها" (و نه دستورات عملیات در داده‌ها). سطح ادراکی در واقع همین شمای ادراکی است.
- شمای ادراکی به سیستم داده می‌شود و در کاتالوگ سیستم نگهداری می‌شود.

به بیانی ساده، دید ادراکی همان تعریف جدول می‌باشد. برای مثال فرض کنید می‌خواهید یک موجودیت مثل دانشجو را تعریف کنید. تعریف جدول مربوطه به همراه فیلدها (صفات خاصه) آن موجودیت در یک پایگاه‌داده، در بحث دید ادراکی دسته بندی می‌شود.

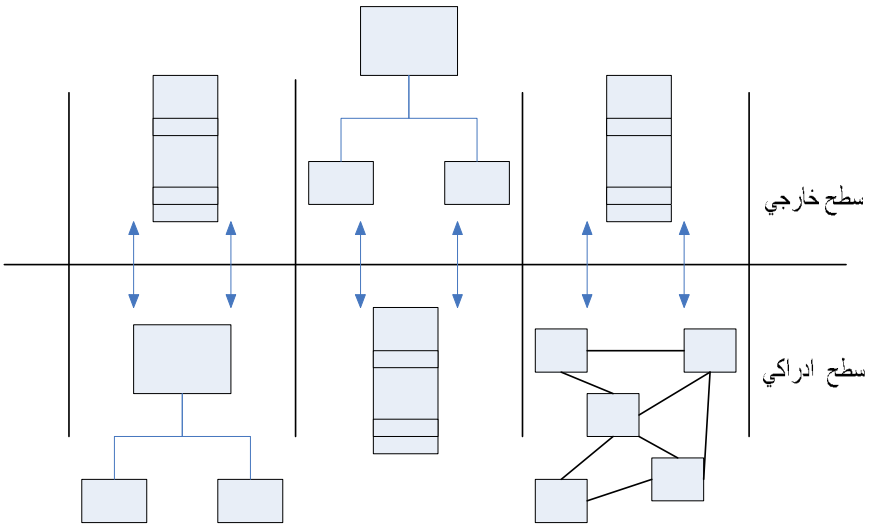
۲-۲ دید (نمای) خارجی

دید یا نمای خارجی در کتب مختلف با عبارات مختلفی توضیح داده شده است. در ادامه چند تعریف از دید خارجی مطرح شده است:

- مفهوم دید یا نمای خارجی در واقع همان مفهومی‌است که در تحلیل و طراحی یک سیستم بکار برده می‌شود. در ادامه تعاریف و نکاتی در مورد دید خارجی آمده است:
- دید کاربر خاص است نسبت به داده‌های ذخیره شده در پایگاه‌داده‌ها.
- این دید جزیی است و نه جامع: نشان دهنده "محدوده‌ای " از پایگاه‌داده‌ها که به نیازهای اطلاعاتی یک کاربر خاص پاسخ می‌دهد.
- این دید هم در سطح انتزاعی مطرح است: بنابرین مبتنی است بر یک ساختار داده‌ای مشخص و معمولاً همان ساختار داده‌ای که دید ادراکی بر اساس آن طراحی و تعریف می‌شود.
- این دید روی دید ادراکی طراحی و تعریف می‌شود (به همین دلیل به جدولهای سطح ادراکی، جدولهای مبنا یا پایه می‌گویند).
- به وصف یا تعریف دید خارجی، شمای خارجی می‌گوییم: نوعی "برنامه " که کاربر سطح خارجی می‌نویسد حاوی دستورات "تعریف داده‌ها " و گاه

"کنترل داده‌ها" در همان سطح خارجی. شمای خارجی هم به سیستم داده می‌شود و در کاتالوگ آن نگهداری می‌شود.

- به تعریف مجموعه دیدهای خارجی کاربر، سطح خارجی گفته می‌شود.



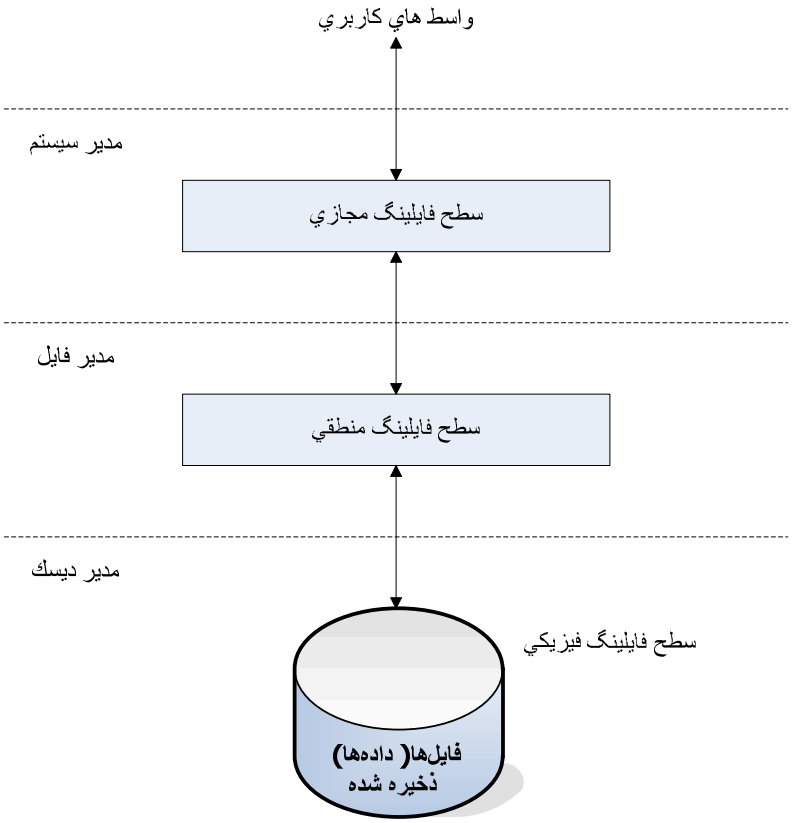
شکل ۳-۳ شمایی از نایکسانی ساختار دادهای در سطوح ادراکی و خارجی

نکته: حداقل از لحاظ نظری می‌توان گفت که ساختار (مدل) داده‌ای در دو سطح انتزاعی یعنی سطح خارجی و سطح ادراکی می‌تواند یکسان نباشد. با این وصف، بدیهی است که یک نرم‌افزار واسط برای تبدیل دو ساختار به یکدیگر لازم است. شکل زیر تفاوت بین ساختارهای داده‌ای دو دید را مشخص می‌کند. عناوین ذکر شده در سطح ادراکی، در مورد ساختارهای داده‌ای است که در فصل بعد مورد بحث و بررسی قرار خواهند گرفت.

۳-۲ دید (نمای) داخلی

سطح داخلی، همان سطح فایلینگ منطقی پایگاه‌داده‌هاست. معنایش این است که DBMS به جنبه‌های فایلینگ فیزیکی پایگاه‌داده‌ها نمی‌پردازد. این کار بر عهده "سیستم فایل فیزیکی" در سیستم عامل است. در واقع، DBMS در محیط فایلینگ

منطقی عمل می‌کند، کم و بیش شبیه برنامه فایل پرداز که در سطح منطقی و نه فیزیکی، کار می‌کند.

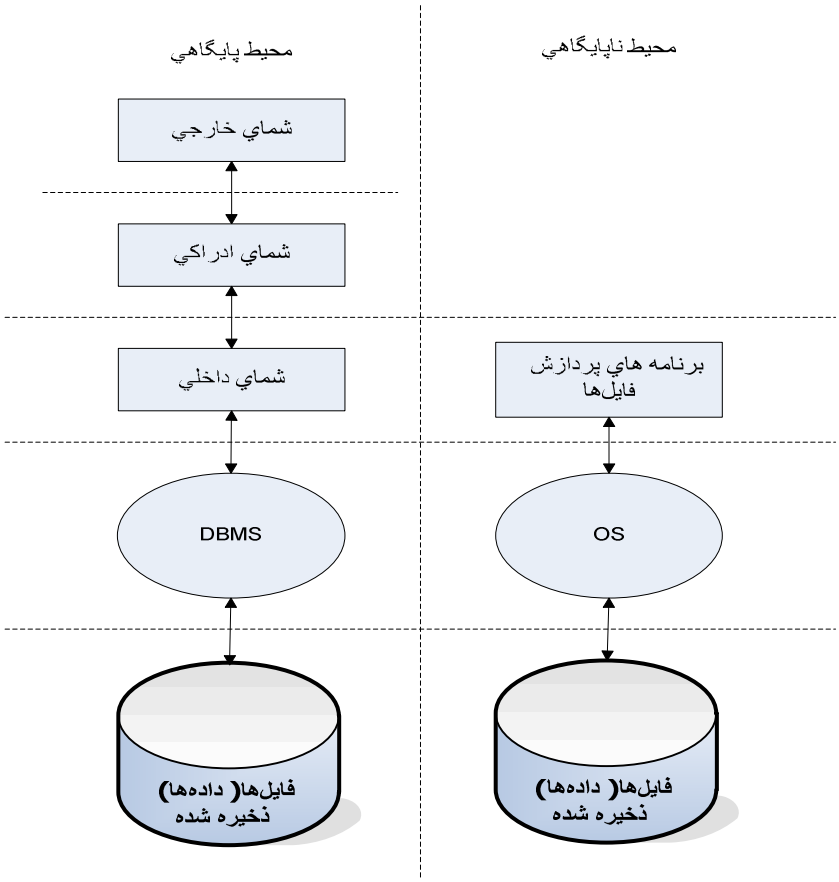


شکل ۳-٤ طرح شماتیک از ذخیره سازی فایلها در محیط DBMS

در ادامه توضیحاتی در خصوص دید داخلی ارائه شده است:

- دید DBMS (و نیز طراح پایگاه‌داده‌ها) است در سطح پایین تر از سطح ادراکی، نسبت به کل دادههای ذخیره شدنی در پایگاه‌داده‌ها.
- این دید در سطح فایلینگ منطقی (و گاه مجازی) پایگاه‌داده‌ها مطرح است.

- این دید مبتنی است بریک (و گاه بیش از یک) ساختار فایل که معمولاً با نظر و دخالت طراح پایگاه‌داده‌ها طراحی می‌شود. این طراحی، اصطلاحا به طراحی فیزیکی موسوم است.
- سطح داخلی پایگاه‌داده‌ها، در واقع سطحی است که در آن فایلهای منطقی پایگاه‌داده‌ها تعریف می‌شود.
- به شرح یا تعریف دید داخلی، شمای داخلی گفته می‌شود: نوعی "برنامه" که توسط خود DBMS (و تا حدی با دخالت طراح پایگاه‌داده‌ها) تولید می‌شود و همانطور که گفتیم شرح یا توصیف فایلینگ منطقی پایگاه است که در واقع همان سطح داخلی است.



شکل ۳-۵ شمایی از تفاوت سطوح معماری بین محیط پایگاهی و ناپایگاهی

نکته: در بعضی از سیستم‌های پایگاهی، DBMS کل فضای پایگاه‌داده‌ها را به صورت مجموعه‌ای از مجموعه صفحات می‌بیند، یعنی نوعی نمای مجازی از داده‌های ذخیره شده در پایگاه دارد. این نمای مجازی بالاتر از سطح فایلینگ منطقی قرار می‌گیرد. در واقع بین سطح نمای مجازی و سطح فایلینگ فیزیکی، یک سطح فایلینگ منطقی واسط است. این سه سطح در شکل ۳-٤ نمایش داده شده است.

نکته: با توجه به دیدهای سه گانه ارائه شده در پایگاه‌داده، اکنون تفاوت سطوح معماری بین محبط پایگاهی و ناپایگاهی بهتر نمایان می‌گردد. شکل زیر این تفاوت را نمایان می‌کند.

۳- سایر اجزاء پایگاه‌داده‌ها

بغیر از سه دید اصلی ارائه شده در بالا، اجزاء دیگری نیز در پایگاه‌داده وجود دارند که عبارتند از:

- کاربر
- زبان میزبان
- زبان داده‌ای فرعی

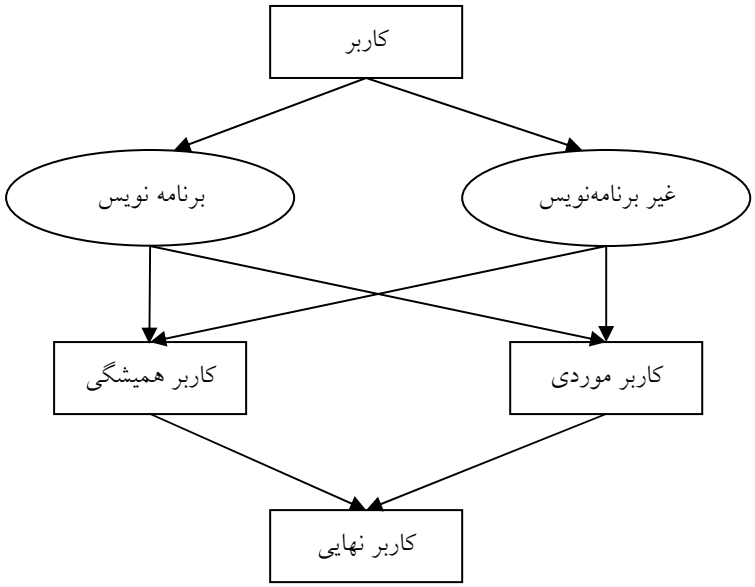
۳-۱ کاربر

در معنای عام، هر استفاده کننده از پایگاه‌داده‌ها را کاربر گوییم. لازم بذکر است که در بعضی از کتب کاربران پایگاه‌داده‌ها بر حسب نوع کار دسته بندی می‌گردند، در اینجا کلمه کاربر بصورت عام استفاده شده است و از ذکر انواع کاربر خودداری شده است.

۳-۲ زبان میزبان

یکی از زبانهای برنامه سازی می‌باشد که قادر به برقراری ارتباط و انتقال دستورات بین خود و پایگاه‌داده باشد. واضح است که هرچه تعداد زبانهای میزبان مورد پذیرش DBMS بیشتر باشد، موارد ذیل تامین می‌گردد:

- تنوع کاربردها امکان پذیر می‌باشد
- تنوع کاربران تامین می‌گردد
- انعطاف پذیری سیستم بیشتر می‌گردد.



شکل ۳-٦ شمایی از انواع کاربران و ارتباط بین آنها

۳-۳ زبان داده‌ای فرعی

این زبان از سه دسته دستور تشکیل شده است:

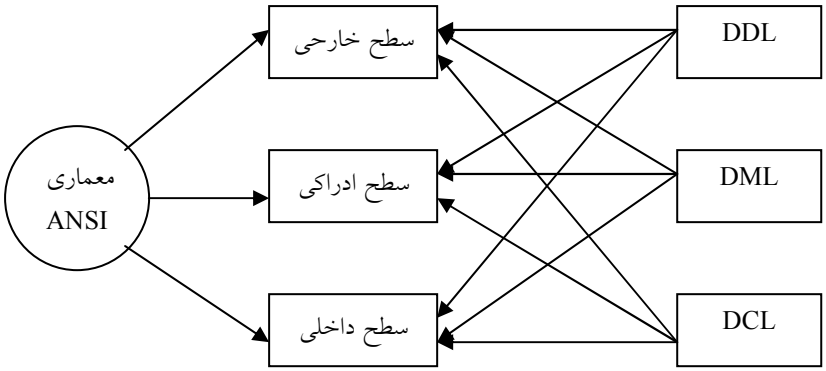
- دستورات تعریف داده‌ها[1]
- دستورات عملیات روی داده‌ها [2]
- دستورات کنترل داده‌ها [3]

این نوع دستورها باید برای هر یک از سطوح معماری پایگاه‌داده وجود داشته باشند. این موضوع در شکل ۳-۷ نشان داده شده است.زبان داده‌ای فرعی از نظر نیاز به

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language (DCL)

زبان میزبان یا عدم نیاز به آن، به دو رده تقسیم می‌شود:

- مستقل یا خود کفا
- ادغام شدنی (ادغام شده)



شکل ۳-۷ ارتباط بین گروه‌های دستوری مختلف و سطوح معماری پایگاه‌داده

زبان مستقل

زبانی است که نیاز به زبان میزبان ندارد و خود به صورت تعاملی (اندر کنشی) استفاده می‌شود و آن را I.DSL گوییم.

زبان ادغام شدنی

زبانی است که دستورهایش (به نحوی) در متن برنامه‌ای به زبان میزبان بکار می‌روند و مستقلا قابل استفاده نیستند و آن را E.DSL گوییم.

نکته: زبان داده‌ای فرعی ممکن است هم مستقل و هم ادغام شدنی باشد(I/E.DSL).

تمرینات

۱. دیدهای سه گانه ارائه شده در معماری پایگاه‌داده‌ها را نام ببرید ؟
۲. دید (نمای) ادارکی را شرح دهید ؟
۳. دید (نمای) خارجی را شرح دهید ؟
٤. دید (نمای) داخلی را شرح دهید ؟
٥. انواع دستورات زبان داده‌ای فرعی را نام ببرید ؟
٦. زبان مستقل به چه معنی است ؟
۷. زبان ادغام شدنی را شرح دهید ؟

فصل ۴

سیستم مدیریت پایگاه‌داده

هدف کلی

در این فصل ابتدا سیستم مدیریت پایگاه‌داده‌ها تعریف و سه بخش اصلی آن مورد بررسی قرار خواهند گرفت. سپس سیستم‌های مدیریت پایگاه‌داده‌ها از جهات مختلف مورد دسته بندی قرار خواهند گرفت. در ادامه اجزاء سیستم مدیریت پایگاه‌داده معرفی شده و کاتالوگ یا دیکشنری پایگاه‌داده‌ها توضیح داده خواهد شد. سپس پارامترهای مختلف مورد نیاز جهت شناخت DBMS مطرح شده و محورهای اصلی مقایسه DBMS شرح داده خواهند شد. در پایان نیز جایگاه مدیر پایگاه‌داده تشریح شده و تیم مدیریت پایگاه‌داده شرح داده خواهد شد.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- سیستم مدیریت پایگاه‌داده‌ها
- بخش ساختاری
- بخش عملیاتی
- بخش جامعیتی
- رده بندی سیستم‌های مدیریت پایگاه‌داده‌ها
- اجزاء سیستم مدیریت پایگاه‌داده‌ها
- نمای بیرونی پایگاه‌داده‌ها

- نمای درونی پایگاه‌داده‌ها
- لایه مدیریت محیط پایگاه‌داده
- لایه هسته پایگاه‌داده
- کاتالوگ سیستم و دیکشنری داده‌ها: متا داده‌ها
- پارامترهای شناخت DBMS
- پارامترهای مربوط به معماری پایگاه‌داده‌ها
- سطح داخلی- فیزیکی
- سطح ادراکی
- سطح خارجی
- پارامترهای مربوط به زبان داده‌ای فرعی
- محورهای اصلی مقایسه DBMSها
- مدیر پایگاه‌داده (DBA)
- تیم DBA، وظایف و مسئولیت‌ها ی آن

۱- تعریف

سیستم مدیریت پایگاه‌داده‌ها یکی از انواع نرم‌افزارهای واسطه بین محیط فیزیکی ذخیره و بازیابی اطلاعات و محیط منطقی برنامه سازی است این نرم‌افزار به کاربر برنامه ساز امکان می‌دهد تا:

- پایگاه‌داده‌های خود را تعریف وایجاد کند.
- در پایگاه‌داده‌های خود عملیات انجام دهد.
- روی پایگاه‌داده‌های خود تا حدی کنترل داشته باشد.

بدیهی است که یک سیستم مدیریت پایگاه‌داده‌ها در محیط یک سیستم عامل عمل می‌کند. شمای ساده‌ای از جایگاه این نرم‌افزار در یک سیستم کامپوتری در شکل زیر نشان داده شده است:

شکل ٤-١ جایگاه DBMS در یک سیستم کامپیوتری

لازم به ذکر است که یک سیستم مدیریت پایگاه‌داده واقعی باید امکان انجام این سه فعالیت اساسی را قبل از هر چیز در محیط انتزاعی به کاربر بدهد. به بیانی دیگر می‌توان گفت که پایگاه‌داده از سه بخش اصلی به شرح ذیل تشکیل شده است:

- بخش ساختاری
- بخش عملیاتی
- بخش جامعیتی

بخش ساختاری

شامل عناصر ساختاری مدل است که همان ساختار داده‌ای اصلی و مفاهیم مرتبط با آن است.

بخش عملیاتی

مجموعه امکاناتی است که به وسیله آنها عملیات مورد نظر کاربر، از جمله بازیابی و ذخیره سازی، در کادر ساختار داده‌ای و مبتنی بر عنصر ساختار اساسی آن انجام می‌شود.

بخش جامعیتی

از مجموعه‌ای از قواعد جامعیتی تشکیل شده است که با استفاده از آن سیستم مدیریت پایگاه‌داده می‌تواند صحت، دقت و سازگاری داده‌ها را در هر لحظه از حیات پایگاه‌داده‌ها، کنترل و تظمین کند.

۲- رده بندی سیستم‌های مدیریت پایگاه‌داده‌ها

سیستم‌های مدیریت پایگاه‌داده را می‌توان به چندین صورت دسته‌بندی نمود. این دسته‌بندی‌ها در ذیل آمده‌اند:

از نظر مدل داده‌ای

- سیستم رابطه‌ای
- سیستم سلسله مراتبی
- سیستم شبکه‌ای

از نظر محیط سخت‌افزاری

- قابلیت اجرا بر روی یک سخت‌افزار خاص
- قابلیت اجرا بر روی سخت‌افزارهای متنوع

از نظر رده بندی کامپیوتر

- قابل اجرا بر روی کامپیوترهای بزرگ[1] و خیلی بزرگ[2]
- قابل اجرا بر روی کامپیوترهای متوسط
- قابل اجرا بر روی کامپیوترهای شخصی
- قابل اجرا بر روی انواع کامپیوترها

از نظر محیط سیستم عامل

- وابسته به یک نوع خاص از سیستم عامل
- عدم وابستگی به سیستم عامل و قابل اجرا بر روی چند سیستم عامل

از نظر نوع معماری سیستم پایگاه‌داده‌ها

- دارای معماری پایگاه‌داده متمرکز
- دارای معماری پایگاه‌داده نا متمرکز

1. Mainframe
2. Super Computer

از نظر معماری مشتری- خدمتگزار[1]

- توانایی در ارائه معماری یک خدمتگزار- چند مشتری
- توانایی در ارائه معماری چند خدمتگزار- چند مشتری

از نظر سیستم فایل

- مستقل از سیستم فایلینگ مربوط سیستم عامل
- وابسته به سیستم فایلینگ سیستم عامل

از نظر متدولوژی زبان پایگاهی

- بدون متدولوژی شیئ گرایی
- دارای متدولوژی شیئ گرایی

از نظر بهینه سازی پرس و جوها و درخواست‌های کاربر

- دارای بهینه ساز متعارف
- دارای بهینه ساز مبتنی بر قاعده، معنایی و...

از نظر نوع تراکنش

- پذیرنده تراکنشهای ساده و تک سطحی
- پذیرنده تراکنشهای با مدل پیشرفته (مثلا تودرتو، زنجیره‌ای و...)

از نظر نوع پردازش

- با قابلیت پردازش بی درنگ
- فاقد این قابلیت

۳- اجزاء سیستم مدیریت پایگاه‌داده‌ها

این نرم‌افزار، مثل هر نرم‌افزار نیمه بنیادی دیگر از واحدهایی تشکیل شده است. تعداد واحدها و حجم هر واحد نرم‌افزاری بستگی به توانایی سیستم در انجام کارها

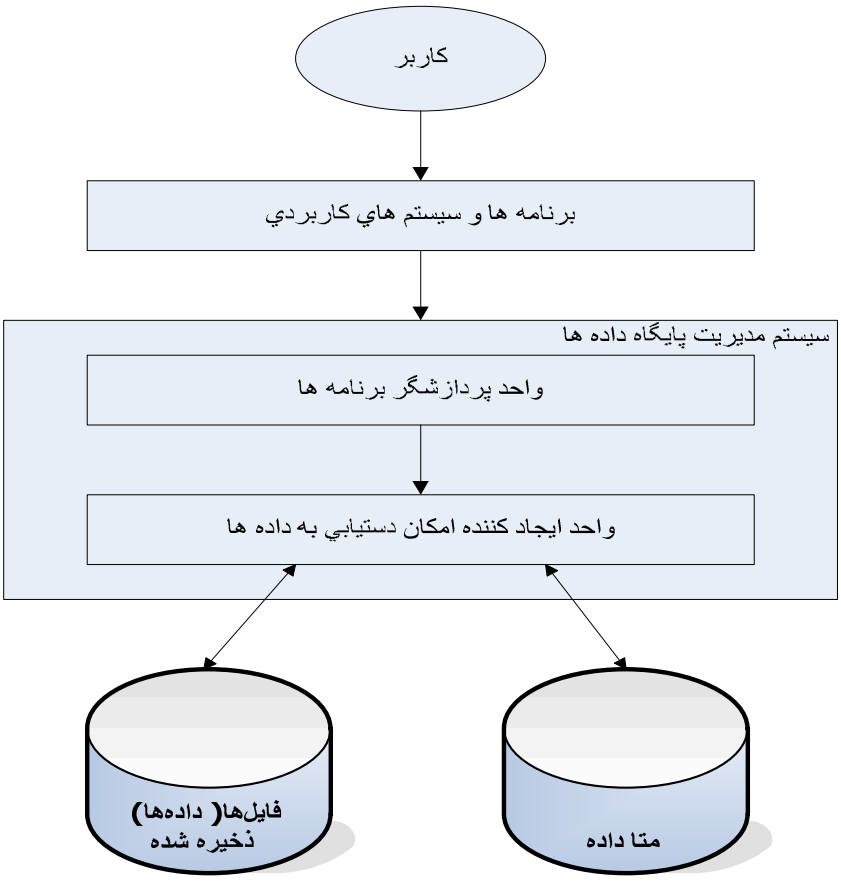
1. Client-Server Architecture

و ارائه خدمات به کاربران دارد. در ادامه واحدهای اصلی سیستم مدیریت پایگاه‌داده‌ها معرفی خواهیم کرد.

۳-۱ نمای بیرونی

نمای بیرونی، از واحدهای اصلی زیر تشکیل شده است:

- واحد پردازشگر "پرسش‌ها"
- واحد پردازشگر "برنامه‌های کاربردی"
- واحدایجاد و مدیریت داده‌های ذخیره شده.



شکل ٤-٢ نمای بیرونی سیستم مدیریت پایگاه‌داده

در اینجا لازم است به این نکته توجه شود که پرسشها نیز در واقع نوعی از برنامه کاربردی و یا در مواردی قسمتی از برنامه کاربردی هستند. تفاوت بین این دو در این نکته است که برنامه‌های کاربردی با یک زبان برنامه نویسی تهیه می‌شوند که به طریقی با پایگاه‌داده ارتباط برقرار می‌کند و دستورات مورد نیاز برای کار با پایگاه‌داده درون زبان انتخاب شده می‌باشد که توسط کامپایلر زبان برای پایگاه‌داده قابل فهم می‌شود. ولی پرسشها با استفاده از دستوراتی که برای پایگاه‌داده قابل فهم می‌باشد، تهیه می‌شوند و در واقع به نوعی وابستگی کامل به پایگاه‌داده مورد استفاده دارد. شکل ٤-٢ نمای بیرونی سیستم مدیریت پایگاه‌داده را بصورتی ساده بیان می‌کند.

۳-۲ نمای درونی

این نرم‌افزار (نمای درونی) از سه لایه تشکیل شده است که هر لایه وظایف خاص خود را بر عهده دارد. لازم به ذکر است که در کتابهای مختلف ممکن است وظایف هر یک از این لایه‌ها متفاوت باشد. در ادامه لایه‌های این قسمت به همراه وظایف هر یک شرح داده شده است:

- لایه مدیریت محیط پایگاه‌داده‌ها [1]
- لایه هسته [2] (سیستم کنترل یا موتور پایگاه‌داده‌ها)
- لایه تسهیلات نرم‌افزاری [3] (ابزارها)

لایه مربوط به تسهیلات نرم‌افزاری معمولاً در سیستم‌های مدیریت پایگاه‌داده‌ها متفاوت می‌باشد و توانایی‌ها و خدمات قابل ارائه در این لایه بعضا بسیار متنوع و یا بسیار ساده است. لذا از بحث پیرامون این لایه اجتناب می‌گردد.

۳-۲-۱ لایه مدیریت محیط پایگاه‌داده

لایه مدیریت محیط پایگاه‌داده وظایف زیر را بر عهده دارد:

1. Database Environment Management Layer
2. Engine Layer
3. Software Facilities

- کنترل جامعیت[1] پایگاه‌داده
- کنترل ترمیم[2] (بازسازی) پایگاه‌داده
- ایمنی و حفاظت پایگاه‌داده[3]
- تولید نسخه‌های پشتیبان[4]
- تولید فایلهای ثبت تراکنش[5]ها

۳-۲-۲ لایه هسته پایگاه‌داده

لایه هسته در واقع لایه اصلی دریافت کننده درخواست‌ها و انجام عملیات مرتبط با آن را بر عهده دارد. مهمترین وظایف این واحد به شرح زیر می‌باشد:

- دریافت درخواست‌ها و پیش کامپایل
- کامپایل درخواست‌ها و بهینه سازی پرس و جو
- مدیریت فایلینگ منطقی و فضای دیسک
- مدیریت بافر (حافظه نهان)
- مدیریت و نظارت بر زمان اجرا دستورات
- مدیریت همزمانی تراکنش‌ها
- مدیریت انتقال داده‌ها
- مدیریت کاتالوگ سیستم

٤- کاتالوگ سیستم و دیکشنری داده‌ها (متا داده‌ها)

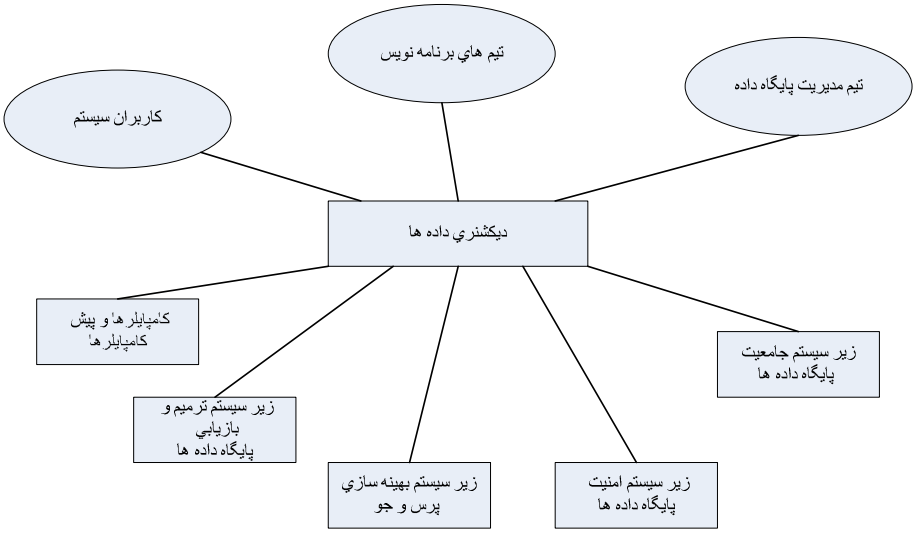
یکی از ویژگی‌های مشی پایگاهی این است که سیستم پایگاهی نه تنها حاوی پایگاه‌داده‌ها است، بلکه تعریف کامل یا توصیف پایگاه‌داده‌ها و قواعد ناظر به آنرا نیز در خود دارد. کاتالوگ حاوی داده‌هایی است در مورد داده‌های ذخیره شده در پایگاه‌داده‌های کاربر. در ادامه نکاتی چند در مورد کاتالوگ سیستم ارائه می‌گردد:

1. Integration Control
2. Repair Control
3. Database Protection
4. Backup Generation
5. Transaction Record

- داده‌های حاوی اطلاعات در مورد داده‌های ذخیره شده در پایگاه‌داده، به متا داده‌ها موسومند.
- در بعضی از کتاب‌ها به کاتالوگ سیستم گاه دیکشنری داده‌ها هم گفته می‌شود، اما در واقع دیکشنری داده‌ها حاوی اطلاعات بیشتری است.
- متا داده‌ها معمولاً از دید کاربر سطح خارجی نهان‌اند، اما مدیر سیستم و یا کاربر مجاز، می‌تواند تا حدی از محتوای کاتالوگ آگاه شود.
- ساختار و محتوای کاتالوگ و دیگشنری داده‌ها در سیستم‌های مختلف یکسان نیست. اما به طور کلی، اطلاعات زیر در آنها ذخیره می‌شود:

دیکشنری داده‌ها معمولاً جزیی از خود سیستم است و اطلاعات زیر در آنها نگهداری می‌شود:

- شماهای خارجی
- شمای ادراکی
- شمای داخلی
- رویه‌های مربوط به تبدیلات بین سه سطح معماری
- شرح ساختار فیزیکی داده‌های ذخیره شده
- مشخصات کاربران و حقوق دستیابی آنها به داده‌های ذخیره شده
- مشخصات برنامه‌های کاربردی تولید شده و ارتباط آنها با درخواست کاربران
- مشخصات پایانه‌های متصل به سیستم و یا کامپیوترهای مشتری و ایستگاههای کاری
- ارتباط بین برنامه‌های کاربردی و داده‌های ذخیره شده
- قواعد مربوط به کنترل صحت و دقت داده‌های ذخیره شده در پایگاه‌داده‌ها
- ضوابط کنترل‌ایمنی داده‌ها
- مشخصات پیکر بندی سخت‌افزاری سیستم و رسانه‌های ذخیره سازی
- اطلاعات متنوع آماری در مورد پایگاه‌داده‌ها، عملیات کاربران، فرکانس اجرای تراکنشها و تعداد دستیابی به اشیاء ذخیره شده
- توابع تعریف شده توسط کاربران



شکل ٤-٣ شمای کلی کاتالوگ سیستم پایگاه‌داده‌ها

۵- پارامترهای شناخت DBMS

امروز مرتبا شاهد عرضه سیستم‌های با نامهای گوناگون تحت عنوان DBMS به بازار مصرف هستیم. عدم اطلاع کافی خریداران و استفاده کنندگان چنین سیستمی‌از پارامترهای انتخاب سبب می‌شود تا اولا انتخاب سیستم مناسب برای یک محیط عملیاتی (سازمان) مخصوصا محیط بزرگ، گاه بر اساس ملاحظات غیر علمی-فنی صورت پذیرد و گاه اساساً نوعی سردرگمی‌در انتخاب بروز کند (البته در انتخاب و خرید یک سیستم ملاحظاتی جز ملاحظات مستقیما مربوط به خود سیستم و پایگاهی که قرار است به کمک آن‌ایجاد شود، نیز می‌تواند مطرح باشد). ثانیا نرم‌افزار خریداری شده منطبق با نیازهای محیط عملیاتی و پاسخگوی احتیاجات اطلاعاتی محیط نباشد. ثالثا از نرم‌افزار به هر حال انتخاب شده بطور بهینه بهره برداری نشود و گاه پیامدهای نامطلوبی بروز کند.

تصمیم گیری در مورد انتخاب سیستم مناسب، مسئله اساسی هر سازمان است. این مسئله مخصوصا در شرایطی که تحولات تکنولوژیک در نرم‌افزار و سخت‌افزار چنان شتابنده است که هر روز تولیدات نرم‌افزاری و سخت‌افزاری جدیدی به بازار

مصرف عرضه می‌شوند، بالاخص برای محیط‌های عملیاتی بزرگ، بسیار استراتژیک می‌نماید. سازمانهایی که با مشکل برنامه ریزی میان مدت و دراز مدت مواجه‌اند، اگر در تصمیم گیری برای انتخاب این تکنولوژی با تخصص وافی و مطالعه همه جانبه و نهایتا با شناخت کافی اقدام نکنند، نه تنها به اهداف تعریف شده در برنامه ریزی نایل نخواهد شد بلکه چه بسا راه به بیراهه خواهد برد و با مشکلات عدیده‌ای مواجه خواهند شد که رفع آنها به آسانی امکان پذیر نخواهد شد.

با توجه به حل بسیاری از مشکلات تکنیکی در ایجاد سیستم‌های توزیع شده ناهمگن، اینک دیگر مسئله انتخاب فقط یک DBMS و یا فقط یک پیکر بندی نرم‌افزاری–سخت‌افزاری می‌تواند چندان مطرح نباشد. با این‌همه انتخاب اجزاء تشکیل دهنده پیکر بندی یک سیستم ناهمگن به هر حال مطرح است تا یک پیکر بندی با بیشترین کارائی، طراحی و تامین شود.

به منظور کسب شناخت تخصصی باید پارامترهای زیادی را در نظر گرفت که این پارامترها را بصورت کلی در پنج رده به شرح زیر دسته بندی می‌کنیم:

- پارامترهای مربوط به توانایی‌ها و کارایی سیستم
- تسهیلات و جنبه‌های دیگر
- مشخصات کلی سیستم
- پارامترهای مربوط به معماری پایگاه‌داده‌ها
- پارامترهای مربوط به زبان داده‌ای فرعی (واسط کار برنامه ساز).

در ادامه هر یک از دسته‌های اصلی را مورد بحث و بررسی قرار خواهیم داد:

۵–۱ پارامترهای مربوط به توانایی‌ها و کارایی سیستم

- توان سیستم در ایجاد و مدیریت پایگاه‌داده‌ها بر اساس معماری پیشنهادی ANSI یعنی سیستم با معماری سه سطحی.
- حداکثر تعداد کاربران همزمان.
- توانایی سیستم در ترمیم داده‌ها و مکانیسم‌های انجام این کار.
- توانایی سیستم در تامین همزمانی، مکانیسم قفل گذاری، اندازه واحد قفل پذیر و وجود استراتژی کارا در حل مشکل بن بست.

- توانایی سیستم در تامین‌ایمنی داده‌ها
- توانایی سیستم در حفاظت داده‌ها وتضمین محرمانگی آنها
- توانایی سیستم درتامین رشد پایگاه‌داده‌ها ودرجه سهولت انجام این کار.
- توانایی سیستم در تضمین جامعیت پایگاه
- تعداد زبانهای میزبان مورد پذیرش سیستم.
- مکانیسم هماهنگی عملکرد سیستم با سیستم عامل خاص پایگاه.
- شرایط بروز تنگناها در عملیات ورودی / خروجی و در فعالیت واحد پردازش مرکزی و مکانیسم رفع آن.
- توانایی سیستم در اشتراکی کردن داده‌ها
- امکانات سیستم در مجاز شماری و سطح یاسطوح استفاده از این امکانات (با توجه به معماری چند سطحی پایگاه).
- آستانه پایداری کارایی سیستم (از نظر تعداد تراکنش‌های اجرا شونده مثلا در ثانیه و با توجه به حجم داده‌های ذخیره شده در پایگاه).
- توانایی سیستم درپردازش پرسشهای موازی.
- توانایی سیستم در پذیرش انواع برنامه‌های کاربردی و پرسشها.
- نحوه پردازش زبان داده‌ای فرعی توسط سیستم (کامپایلری، مفسری).
- توانایی سیستم در بهینه سازی پرسشها و نوع بهینه سازی (اعمال الگوریتم‌های بهینه در اجرای پرسشها) و زمان انجام بهینه سازی استراتژی دستیابی (در زمان کامپایل، در اولین بار اجرای پرسش، در هر بار که پرسش اجرا می‌شود).
- مدلهای تراکنش که می‌پذیرد (بویژه مدلهای پیشرفته).
- قابلیت گسترش پذیری و ارتقاء سیستم.
- توانایی سیستم در سازماندهی مجدد محیط فیزیکی پایگاه و درجه پویایی و خودکار بودن این کار.
- ایجاد سیستم در خدمت رسانی در محیط شبکه‌ای.
- نوع معماری سیستم پایگاه‌داده‌ها که سیستم می‌تواند آن را تامین کند.
- قابلیت تعامل سیستم با سیستم‌های متعارف دیگر (با مدل داده‌ای و زبان داده‌ای یکسان یا متفاوت) و مکانیسم ارتباط با آنها.

- قابلیت تعامل سیستم با گونه‌های غیر متعارف سیستم مدیریت پایگاه‌داده‌ها و مکانیسم و ملزومات همایندی با سیستم‌ها.
- قابلیت تعامل سیستم با سیستم مدیریت پایگاه دانش و مکانیسم همایندی.
- قابلیت تعامل سیستم با سیستم مدیریت پایگاه‌داده‌های شیئ‌گرا و مکانیسم همایندی.
- میزان تغییر کارآیی سیستم از ماشین کلاسیک تا ماشین خاص پایگاه‌داده‌ها.
- امکانات سیستم برای ایجاد پایگاه‌داده‌های زمانبند.
- زبانی که سیستم با آن نوشته شده است.
- میزان رعایت اصول نوین مهندسی نرم‌افزار در طراحی و تولید سیستم.
- میزان رعایت اصول شیئ گرایی در طراحی و پیاده سازی سیستم.
- توانایی سیستم در ایجاد و مدیریت پایگاه‌های بزرگ و خیلی بزرگ (از نظر حجم داده‌ها و تعداد تراکنشها).
- قابلیت سیستم در نهان نگاری داده‌ها
- قابلیت سیستم در پشتیبانی انواع معماری سیستم پایگاه‌داده‌ها.
- قابلیت سیستم در کنترل اشتباهات برنامه سازی.
- قابلیت سیستم در عملیات در محیط پایگاه‌داده‌های وب و یا ادغام در " وب"
- قابلیت سیستم در عملیات در محیط پایگاه‌داده‌های موبایل
- قابلیت جابجایی پذیری سیستم
- متوسط زمان بین دو خرابی پی در پی سیستم
- میزان کار لازم برای پیاده سازی سیستم
- استانداردهای بکار رفته در سیستم
- سیستم عامل مورد نیاز
- قابلیت سیستم در پردازش پیام‌ها.
- تجهیزات لازم برای سیستم از جمله:
- نوع سخت‌افزار
- حداقل حافظه لازم
- حداقل فضای دیسک

- نوع و تعداد سیستم عامل
- تعدد و تنوع رسانه‌های جانبی

۲-۵ تسهیلات و جنبه‌های دیگر

این تسهیلات و جنبه‌ها می‌توانند همراه خود سیستم و یا به نحوی قابل تامین و استفاده در محیط سیستم باشند (به صورت نرم‌افزار جداگانه به کاربران سیستم عرضه شوند). هر چه سیستم از نظر پذیرش انواع تسهیلات و سازگاری و همایندی با آنها غنی تر باشد، مطلوبتر است و بهره برداری بهتری از سیستم امکان پذیر خواهد بود. برخی از این تسهیلات (ابزارها) عبارتند از:

- امکانات تولید نسخه‌های پشتیبان
- امکانات پرسش به کمک مثال و پرسش به کمک فرم
- امکانات بررسی‌ها و تحلیل‌های آماری
- امکانات گرافیکی
- امکانات نگهداری سیستم
- امکانات بهینه سازی برنامه‌های کاربردی
- امکانات یادگیری طرز کار با سیستم و بهره برداری آن
- امکانات کار در محیط وب
- امکانات مدیریت پویای پرسشها
- امکانات نظارت بر عملیات کاربران در پایگاه
- امکانات مدیریتی برای مدیر پایگاه‌داده‌ها
- امکانات پشتیبانی از XML
- امکانات پردازش تحلیلی بر خط
- مولد گزارش
- انواع ویرایشگرها
- ابزارهای‌ایجاد برنامه‌های کاربردی و واسطه‌های کاربردی
- امکانات تعریف گروههای کاربردی و تعیین امتیازات هر گروه (در انجام عملیات در پایگاه‌داده‌ها)

- امکانات دستیابی به داده‌های دوردست
- امکانات تولید خروجی‌های کاربرپسند
- امکانات تنظیم کردن پایگاه
- امکانات تنظیم کردن خود سیستم
- امکانات بارگذاری (گاه موسوم به عمل ورود داده‌ها)، بازبارگذاری، خالی کردن (گاه موسوم به عمل صدور داده‌ها) پایگاه‌داده‌ها
- امکانات تبدیل یک ساختار داده از سطح خارجی معماری ANSI به ساختار داده دیگر از سطح ادراکی همان معماری
- امکانات فعال و غیرفعال کردن پایگاه
- امکانات تهیه آمارهای مربوط به عملیات ذخیره سازی (درج، حذف، بهنگام سازی)
- امکانات سیستم برای مدلسازی داده‌ها و طراحی منطقی و فیزیکی پایگاه
- امکانات سیستم برای گشت زنی (گذارگری) در پایگاه‌داده‌ها
- امکانات استفاده از زبانهای سطح بالا (نه به عنوان زبان میزبان) و مکانیسم پیوند به سیستم از طریق آنها
- امکان‌ایجاد مدولهای اجرایی برای برنامه‌های کاربردی در محیط سیستم به نحوی که خارج از محیط سیستم قابل اجرا باشند
- امکان استفاده از یک سیستم خبره در محیط سیستم
- مولد فرم
- مولد منو
- امکانات شبکه‌ای
- واسط 4GL
- ابزارهای تولید مستندات (در مراحل مختلف تولید یک سیستم کاربردی)

آنچه بر شمرده شد، فهرستی است از تسهیلات و جنبه‌ها، و با توجه به پیشرفت مهندسی نرم‌افزار، امکانات دیگری نیز می‌توانند مطرح باشند. به ویژه که هر روز انواع گوناگونی از این قبیل ابزارها و تسهیلات تولید و به بازار مصرف عرضه می‌شوند. سیستم نه تنها باید بتواند با این امکانات تماس برقرار کند بلکه باید به

سهولت با آنها همایندی داشته باشد.

۵-۳ مشخصات کلی سیستم

- نام نرم‌افزار
- عنوان شرکت سازنده و شرکت فروشنده
- شماره ویراست (نگارش) مورد بررسی-شماره آخرین ویراست
- تاریخ عرضه (اولین نگارش و آخرین نگارش)
- قیمت
- نام کشور سازنده و فروشنده
- شرایط کلی تحویل
- خدمات بعد از تحویل
- کمیت و کیفیت آموزش
- مستندات
- تعداد مشتری‌ها و ماهیت نیازهای اطلاعاتی و پردازشی آنان
- ضمانت (های)شرکت فروشنده
- امکانات شرکت فروشنده در گسترش یا ارتقاء سیستم
- در دسترس بودن فروشنده (هرگاه که لازم باشد)
- وجود تیم فنی پشتیبانی سیستم، در شرکت فروشنده

۵-٤ پارامترهای مربوط به معماری پایگاه‌داده‌ها

از آنجائیکه معماری پایگاه‌داده‌ها در سه سطح داخلی، ادراکی و خارجی در نظر گرفته می‌شود، لذا پارامترهای مربوط به معماری را نیز در این سه سطح بررسی می‌کنیم:

۵-٤-۱ سطح داخلی- فیزیکی

- نوع داده
- نام فیلد
- طول داده (حداکثر طول پیش نهاده)

- حداکثر تعداد رکورد
- حداکثر طول هر رکورد
- ثابت یا متغیر بودن طول رکورد
- حداکثر تعداد فیلد در هر نوع
- حداکثر طول هر فیلد
- ثابت یا متغیر بودن طول فیلد
- حداکثر تعداد فایل‌های باز در یک زمان
- حداکثر تعداد تراکنش‌های عمل کننده در محیط داخلی-فیزیکی
- حداکثر تعداد شاخص‌ها و ساختار درونی هر شاخص و درجه پویایی آن
- اندازه شاخص
- هزینه‌ایجاد شاخص
- ساختار فایلهای موجود
- حداکثر تعداد کلیدها
- حداکثر طول کلید
- نوع کلید
- نحوه انجام عملیات ذخیره سازی در محیط داخلی-فیزیکی و کارایی سیستم در هر عمل
- نحوه جایگیری فایلها روی رسانه خارجی
- نحوه پیاده سازی " هیچمقدار " در این سطح و میزان حافظه مصرفی برای آن
- چگونگی تناظر (نگاشت) بین فایلها و ساخت‌های سطح ادراکی
- چگونگی خوشه واری و موضعی بودن فایلها و رکوردها
- چگونگی فشرده سازی داده‌ها و نسبت فشرده سازی
- فشرده سازی شاخص‌ها و نسبت فشرده سازی
- اندازه و تعداد بافر مورد نیاز
- اندازه بلاک
- اندازه صفحه(ثابت یا متغیر)
- وجود زبان داده‌ای فرعی برای سطح داخلی و امکانات آن

- میزان دخالت مدیر پایگاه‌داده‌ها درایجاد سطح داخلی-فیزیکی و کنترل آن

۵-٤-۲ سطح ادراکی

- وجود سطح ادراکی
- درجه انتزاع سطح ادراکی (که مستقیما به مدل داده‌ای بستگی دارد)
- وجود زبان داده‌ای فرعی در سطح ادراکی و امکانات آن
- امکانات تعریف قواعد (محدودیتهای) جامعیت (به عنوان بخشی از مدل داده‌ای)در بیرون از برنامه‌های عملیاتی (با احکام تعریف داده‌ها)
- امکانات تعریف انواع کلید (کلید کاندید، کلید اصلی، کلید خارجی، کلید دیگرو...)
- میزان تامین استقلال داده‌ای فیزیکی
- میزان دینامیسم رشد پایگاه در سطح ادراکی
- امکانات سازماندهی مجدد پایگاه در سطح ادراکی (تغییر در طراحی منطقی و در پی آن تغییر شما)
- چگونگی مکانیسم نگاشت سطح ادراکی به داخلی و جنبه‌های نگاشت (نام داده، نوع داده، طول داده، کد نمایش داده، واحد داده و...)
- پیامدهای سازماندهی مجدد سطح ادراکی در سطح داخلی-فیزیکی
- میزان افزونگی در سطح ادراکی و مکانیسم کنترل آن و میزان انعکاس آن در سطح داخلی- فیزیکی
- وجود مفهوم میدان و امکانات لازم برای آن در سطح ادراکی
- نحوه برخورد با " هیچمقدار " در سطح ادراکی
- امکانات سیستم در مجازشماری به ویژه در این سطح

۵-٤-۳ سطح خارجی

- وجود مفهوم همگانی دید خارجی
- حداکثر تعداد دیدهای هر کاربر و در مجموع، تنوع دیدها از نظر مکانیسم تعریف آنها

- وجود زبان داده‌ای فرعی خارجی و امکانات آن (برای بازیابی، ذخیره سازیی و کنترل)
- درجه انتزاع تامین شده در این سطح و ارتفاع این انتزاع
- میزان تامین استقلال داده‌ای منطقی
- چگونگی نگاشت سطح خارجی به ادراکی و محورهای این نگاشت (حدود آزادی عمل کاربران در تعیین جنبه‌های داده)
- قدرت سیستم در انجام عملیات بهنگام سازی دیدها (بهنگام سازی به معنای عام یعنی درج، حذف و تغییر) و تنوع دیدهای پذیرای این عملیات
- وجود کلید اصلی در این سطح (برای بعض دیدها به تشخیص مدیر پایگاه‌داده‌ها)

۵-۵ پارامترهای مربوط به زبان داده‌ای فرعی

- میزان پایبندی احکام زبان به عناصر ساختاری مدل داده‌ای در جهت تامین انتزاع هرچه بیشتر
- وجود دستورات DDL، DML، DCL برای هر سه سطح معماری ANSI
- داشتن هر دو جنبه خودکفا (مستقل) بودن و قابل ادغام بودن
- انواع داده‌ای موجود
- گسترش پذیری نوع داده‌ای و وجود نوع داده‌ای تعریف شده توسط کاربر
- پذیرش BLOB و داده‌های از نوع تصویر، ویدئو و متن
- امکان نوشتن رویه ذخیره شده و رهانا
- جهت جابجایی مکان نما (جلو، عقب و هر دو سو)
- واسط‌های محیط برنامه سازی (مثل ODBC و...)
- درجه کمال از نظر ساختار داده‌ای
- درجه کمال برنامه سازی
- سازگاری با 4GL
- رویه‌ای یا نا رویه‌ای بودن
- مفسری یا کامپایلری بودن

- رعایت اصل وحدت احکام برای عمل واحد در دو سطح انتزاعی و نیز در یک سطح مشخص
- وجود امکان کنترل رکوردهای تکراری برای مدیر پایگاه‌داده‌ها
- قدرت در برنامه سازی الگوریتم‌های بهینه برای انجام چهار عمل اصلی (بازیابی، حذف، درج و بهنگام سازی)
- سهولت نوشتن برنامه‌های لازم برای انجام چهار عمل اصلی (با توجه به مفاهیم ساختار داده‌ای)
- سهولت یادگیری و نزدیکی به زبان طبیعی
- وجود امکان معرفی میدان و عملیات روی میدان‌ها
- وجود احکام لازم برای اعمال قواعد جامعیتی
- وجود احکام لازم برای‌ایمنی و حفاظت داده‌ها
- وجود احکام لازم برای اعمال ضوابط مجازشماری
- وجود احکام لازم برای تعیین حدود تراکنش‌ها و کنترل همزمانی آنها
- وجود امکان نوشتن الگوریتم‌های موازی پرسش
- میزان شیئ گرا بودن
- امکانات لازم برای داده‌های زمانبند

٦- محورهای اصلی مقایسه DBMSها

در مقایسه این سیستم‌ها، باید تمام پارامترهای گفته شده در قسمت ۹ این گفتار را در نظر داشت. اما با بررسی این پارامترها می‌توان محورهای اصلی مقایسه را بدست آورد. در زیر محورهای مهمتر را بر می‌شمریم (فرض بر این است که سیستم‌های مقایسه شونده معماری ANSI را پشتیبانی می‌کنند):

- امکانات تعریف داده‌ها
- انواع داده‌ای مورد پذیرش سیستم
- امکانات عملیات در داده‌ها
- نوع تراکنش‌ها (چند پرسشی یا برنامه، تعداد آنها) و طرز مدیریت آنها
- امکانات پردازش پرسش‌ها، بهینه سازی آنها و زمان بهینه سازی

- امکانات‌ایجاد دیکشنری داده‌ها
- وضعیت سطح داخلی- فیزیکی پایگاه‌داده‌ها از نظر ساختار فایلها، شیوه‌های دستیابی (بویژه شاخص بندی) و جنبه‌های دیگر.
- معماری سیستم پایگاهی قابل‌ایجاد و مدیریت
- محیط سخت‌افزاری و حداقل امکانات لازم
- محیط سیستم عامل لازم
- تکنیکهای کنترل همزمانی
- امکانات مدیریت محیط پایگاه‌داده‌ها: کنترل جامعیت، ترمیم، ایمنی و حفاظت.
- تسهیلات طراحی پایگاه‌داده‌ها (طراحی منطقی و طراحی فیزیکی).
- مکانیسم ادغام زبان داده‌ای فرعی در زبان میزبان (وجود پیش کامپایلر یا فراخوانی توابع).
- تنوع دیدهایی که عملیات ذخیره سازی را می‌پذیرند.
- مکانیسم مجازشماری کاربران (نامتمرکز، متمرکز یا نیمه متمرکز).
- امکانات مورد نیاز مدیر پایگاه‌داده‌ها.
- امکانات نهان نگاری داده‌ها.
- طرز انجام عمل بهنگام سازی (درجا یا برون از جا).
- الگوریتم اجرای عملگر پیوند و سایر عملگرهای جبر رابطه‌ای.
- امکانات تعامل با سیستم‌های دیگر (نوع میان افزار و...)
- طرز نمایش نتایج عملیات (گرافیک و گزارش).
- پذیرش یا عدم پذیرش زبانهای نسل چهارم و پنجم.
- امکانات تولید برنامه‌های کاربردی.
- امکانات پشتیبانی تصمیم.
- امکانات لازم برای تولید واسطهای کاربری.
- امکانات عملیاتی در محیط سیستم " توزیع شده ".
- تسهیلات نرم‌افزاری دیگر.

۷- روش مطالعه سیستم

در مطالعه این نرم‌افزار، به منظور کسب آشنایی مقدماتی با آن (ونه چندان تخصصی)، باید موارد زیر بررسی شود:

- بررسی شرکت سازنده، خانواده نرم‌افزارهای پایگاهی مشابه و تاریخچه سیستم
- حداقل پیکره بندی سخت‌افزاری و نرم‌افزاری لازم
- بررسی وجود اجزاء معماری ANSI برای پایگاه‌داده‌ها
- امکانات زبان داده‌ای فرعی سیستم
- امکانات سیستم ازنظر زبان میزبان
- بررسی مولفه‌های مدل داده‌ای و میزان رابطه‌ای بودن سیستم
- اجزاء تشکیل دهنده سیستم
- روند کلی اجرای برنامه توسط سیستم
- نحوه کار با سیستم، برپاسازی، راه‌اندازی و آماده سازی سیستم، ورود به سیستم و کارهای لازم برای‌ایجاد پایگاه و انجام عملیات در آن
- تسهیلات جانبی سیستم از جمله واسط‌های کاربری و....

۸- رویه‌های مستند برای کاربران

برای اینکه کاربران و اعضاء تیم مدیریت پایگاه‌داده‌ها بتوانند از سیستم استفاده کنند، معمولاً مجموعه‌ای از دستورها و قواعد، موسوم به رویه‌های مستند توسط عرضه کنندگان سیستم در اختیار خریداران قرار داده می‌شود. در این رویه‌های مستند چگونگی انجام فعالیتهای زیر مشخص شده است:

- برپاسازی سیستم
- طرز ارتباط با سیستم
- طرز استفاده از سیستم
- طرز استفاده از تسهیلات و امکانات آن
- تولید نسخه‌هایی از پایگاه‌داده‌ها

- طرز تشخیص عیبهای سخت‌افزاری و نرم‌افزاری و چگونگی رفع آنها و ترمیم پایگاه‌داده‌ها
- تغییر ساختار پایگاه‌داده‌ها (سازماندهی مجدد)
- تنظیم سیستم
- بهبود بخشیدن کارایی پایگاه‌داده‌ها
- تولید نسخه‌های پشتیبان

۹- هزینه‌ها

استفاده از تکنولوژی پایگاه‌داده‌ها هزینه‌هایی دارد. برخی از اقلام مهمتر هزینه عبارتنداز:

- هزینه خرید نرم‌افزار اصلی(DBMS)
- هزینه آموزش نرم‌افزار اصلی
- هزینه نگهداری و بهره برداری از آن
- هزینه تبدیل سیستم ناپایگاهی به سیستم پایگاهی
- هزینه تهیه ابزارهای نرم‌افزاری دیگر
- هزینه آموزش امکانات نرم‌افزاری
- هزینه تهیه بسته‌های کاربردی
- هزینه آموزش بسته‌های کاربردی
- هزینه تهیه مستندات خود سیستم
- هزینه تهیه مستندات امکانات نرم‌افزاری و بسته‌های کاربردی
- هزینه تنظیم مستندات سیستم پایگاه‌داده‌ها
- هزینه تهیه سخت‌افزار پردازشگر (کامپیوتر از رده‌های مختلف)
- هزینه تهیه سخت‌افزار ذخیره سازی
- هزینه تامین شبکه‌های لازم (در صورت لزوم)
- هزینه نگهداری و بهره برداری از سیستم کاربردی
- هزینه بهینه سازی و گسترش سیستم
- حقوق و مزایای افراد تیم مدیریت و تیم‌های اجرایی

۱۰- مدیر پایگاه‌داده (DBA)

مدیر پایگاه‌داده‌ها فردی است متخصص در پایگاه‌داده‌ها و با مسئولیت علمی-فنی و نیز اداری در محدوده وظایفی که عهده دار است. این مدیر معمولاً همراه با یک تیم تخصصی کار می‌کند که به آن تیم مدیریت پایگاه‌داده‌ها می‌گویند. هر یک از اعضا این تیم مسئولیت خاصی دارد و در حیطه اختیارات و وظایفش می‌تواند سرپرست یک تیم اجرایی باشد.

۱۰-۱ اصطلاح تیم DBA

در یک محیط کاری برخوردار از سطح مطلوب دانش و تکنولوژی و عمل کننده بر اساس دیسیپلین‌ها و استانداردهای علمی‌و مهندسی و دارای مدیریت پویا، وجود این تیم تخصصی اجتناب ناپذیر است. بعلاوه این تیم باید از مشاورانی در زمینه‌های دیگر مهندسی نرم‌افزار و مهندسی سخت‌افزار استفاده کند و حتی مطلوب این است که بعضی از مشاوران به نحوی عضو خود تیم باشند. برخی از مسئولیت‌های اصلی در این تیم تخصصی عبارتند از:

- مدیر پایگاه‌داده‌ها
- مدیر داده‌ها
- مدیر امور پژوهشی- توسعه
- مدیر سیستم (های) کاربردی
- مسئول تیمهای برنامه سازی
- مسئول کنترل کارایی DBMS
- مسئول کنترل کارایی خود سیستم پایگاهی
- مسئول نظارت بر عملیات روی پایگاه‌داده‌ها و انجام فعالیت‌های آماری مربوطه
- مسئول تماس با کاربران زیر محیط‌های سازمان
- مسئول تنظیم مستندات

اصطلاح تیم DBA به دو معنا (در محیط‌های کاری) مطرح است:

- **تیم DBA در معنای محدود:** با این معنا، تیم DBA تیمی‌است که پایگاه‌داده‌های سازمان را، پس ازایجاد توسط گروه دیگری از متخصصین، تحویل می‌گیرد و پس از تحویل، وظیفه نگهداری، بهره برداری و گاه بهینه سازی و احتمالا گسترش "سیستم" را بر عهده دارد.
- **تیم DBA در معنای گسترده:** با این معنا، تیم DBA، خود همه مراحل لازم برای‌ایجاد پایگاه‌داده‌های سازمان را انجام می‌دهد و سپس نگهداری، بهره برداری و بهینه سازی و گسترش آن را برعهده می‌گیرد.

۱۰-۲ مسئولیت‌ها

مدیر داده‌ها فردی است با دانش و تجربه در مدیریت و آشنا با دانش و تکنولوژی پایگاه‌داده‌ها. برای اطلاع از وظایف مدیر داده‌ها به منابع درس‌های "مهندسی نرم‌افزار"، "تحلیل و طراحی سیستم‌ها" و نیز درس "سیستم اطلاعات مدیریت" مراجعه شود.

۱۰-۳ وظایف

تیم DBA در معنای گسترده وظایفی دارد که اهم آنها را ذکر می‌کنیم. توجه داریم که ترتیب انجام این وظایف لزوما همان نیست که در زیر آمده است (روشن است که در اجرای هر پروژه، نیاز به یک طرح زمان بندی فعالیتهاست که باید آماده شود). همچنین برخی از این وظایف می‌توانند بطور همزمان انجام شوند.

- مشارکت در تفهیم اهمیت و نقش داده به مدیریت سازمان
- مشارکت در تفهیم اهمیت و مزایای تکنولوژی پایگاه‌داده‌ها.
- مشارکت در تصمیم گیری در مورد استفاده یا عدم استفاده از این تکنولوژی.
- مشارکت در توجیه علمی-فنی تصمیم استفاده از این تکنولوژی.
- مطالعه دقیق و همه جانبه محیط عملیاتی و برآورد خواسته‌ها و بر آورد نیازهای کاربران (انجام اصولی مهندسی نیازها).

- بررسی روند داده‌ها و روند رویدادها در محیط و رسم نمودار روند داده‌ها و روند رویدادها (یک یا هر دو بسته به شیوه مدلسازی سیستم مورد نظر) و تهیه و تنظیم مستندات لازم.
- مدلسازی معنایی داده‌ها (با مراحلی که دیده شد از جمله رسم نمودار EER).
- تخمین حجم داده‌های ذخیره شدنی در پایگاه‌داده‌ها.
- تصمیم گیری دو مورد تعیین معماری سیستم پایگاه‌داده‌ها و تعیین مشخصات سیستم (های) کاربردی مورد نیاز.
- مشارکت در انتخاب DBMS (ها) و پیکر بندی سخت‌افزاری و نرم‌افزاری لازم در صورت لزوم (اگر انتخاب نشده باشد).
- تصمیم گیری در انتخاب و انتساب اعضاء تیمهای اجرایی.
- تصمیم گیری در انتخاب ابزارهای نرم‌افزاری دیگر برای تولید و گسترش سیستم مورد نظر.
- تصمیم‌گیری درمورد زبان(های) برنامه‌سازی مورد نیاز و متناسب با هر کاربرد.
- طراحی سطح ادراکی پایگاه‌داده‌ها (طراحی منطقی).
- نوشتن شمای ادراکی و برنامه‌های لازم برای‌ایجاد پایگاه‌داده‌ها.
- تعیین مجموعه قواعد (محدودیتهای) جامعیت ناظر به پایگاه‌داده‌ها.
- نظارت بر تعیین دیدهای خارجی و نوشتن شماهای خارجی.
- تصمیم گیری در مورد مشخصات ساختار سطح داخلی پایگاه‌داده‌ها و تعیین ساختار فایل‌های مناسب، استراتژی‌های دستیابی کارا و نوشتن شمای داخلی.
- انجام طراحی توزیع (در صورت معماری توزیع شده).
- طراحی "برنامه‌های کاربردی"، تراکنش‌های لازم و رویه‌های عملیاتی لازم. (توجه داشته باشیم که هر برنامه‌ای، نیاز به طراحی دارد و تنها پس از طراحی اصولی برنامه، می‌توان برنامه سازی کرد)، وایجاد ارتباط دائم با تولید کنندگان " سیستم کاربردی".
- طراحی واسطه‌های کاربردی.
- ایجاد نمونه نخست سیستم پایگاهی و بارگذاری پایگاه با داده‌های تستی.

- نوشتن برنامه‌های لازم برای کنترل پایگاه‌داده‌ها بویژه اعمال محدودیتهای جامعیتی.
- نوشتن برنامه‌های لازم برای بهره برداری از پایگاه‌داده‌ها.
- ایجاد سیستم پایگاهی واقعی (و منطبق با نیازهای کاربران).
- نظارت بر وارد کردن داده‌ها (در حجم محدود).
- انتخاب استراتژی‌های تست مناسب و تست کردن " سیستم" با داده‌های تستی و نیز با داده‌های واقعی در حجم محدود (انجام دو مرحله تست).
- نظارت بر وارد کردن داده‌های واقعی سازمان.
- تست کردن " سیستم" با داده‌های واقعی و در حجم واقعی
- تنظیم دقیق قسمتهای مختلف سیستم و کل سیستم یکپارچه.
- تعیین ضوابط دستیابی کاربران به داده‌ها.
- نظارت در تهیه مستندات لازم در همه مراحل کار.
- وضع استانداردهای لازم در همه مراحل کار و نظارت بر اعمال آنها.
- تصمیم گیری در مورد چگونگی ترمیم پایگاه‌داده‌ها و در صورت لزوم تهیه یا توسعه ابزارهای این کار و انجام ترمیم پایگاه.
- کنترل مداوم کارایی DBMS و تلاش در افزایش کارایی.
- نظارت و کنترل دائم بر عملیاتی که در پایگاه‌داده‌ها انجام می‌شود.
- کنترل جامعیت پایگاه‌داده‌ها.
- تضمین محرمانگی داده‌ها.
- تصمیم گیری در مورد چندی و چگونگی رشد (گسترش) پایگاه‌داده‌ها.
- اتخاذ تدابیر لازم برای‌ایمنی و حفاظت داده‌ها و اعمال این تدابیر.
- مدیریت کاربران پایانی (ایجاد و تعریف کاربران، گذر واژه‌ها، امتیازها و...).
- تماس دائم با کاربران و شناخت نیازهای جدید آنها.
- تولید نسخه‌های پشتیبان بطور متناوب (با تناوب مناسب).
- تعیین الگوهای استفاده از داده‌ها و بسامد (فرکانس) استفاده از داده‌ها.
- تصمیم گیری در مورد چگونگی سازماندهی مجدد پایگاه‌داده‌ها.

- انجام تبدیل و انتقال داده‌ها از" سیستم‌های موجود" به پایگاه‌داده‌های جدید و انجام تبدیل برنامه‌های کاربردی موجود به گونه‌ای که قابل اجرا در"سیستم کاربردی" جدید باشند.
- تلاش در جهت ارتقاء سطح دانش و فن اعضاء تیم و کاربران (بویژه در زمینه تکنولوژی اطلاعات و سیستمهای اطلاعاتی).
- تلاش در جهت شناسایی امکانات جدید، گسترش، ارتقاء و کاراتر کردن سیستم با استفاده از این امکانات.
- تهیه و تنظیم انواع آمارها و گزارشات کنترلی و مدیریتی در مورد سیستم پایگاه‌داده‌ها و کاربران.
- تضمین انجام و اتمام "پروژه پایگاهی" در مدت زمان پیش بینی شده و با توجه به محدودیت بودجه.

تمرینات

۱. با توجه به مطالعه گفتار، وظایف اساسی یک DBMS را فهرست کنید.
۲. برای طراحی و تولید یک RDBMS، حداقل چه اجزایی لازم است ؟
۳. یک RDBMS را انتخاب و کارهای لازم برای برپاسازی آن را انجام دهید.
٤. در انتخاب و استفاده از یک DBMS، چه هزینه‌های دیگری را باید در نظر داشت (غیر از هزینه‌های برشمرده در گفتار) ؟
۵. چه تفاوت (هایی) بین متا داده‌ها و دیکشنری داده‌ها وجود دارد ؟
٦. پارامترهای اصلی شناخت یک DBMS را نام ببرید ؟
۷. مدیر پایگاه‌داده را تعریف کنید ؟
۸. تیم DBA به چه معنا است ؟
۹. کدامیک از وظایف DBA، مستقل از DBMS، قابل انجام است ؟
۱۰. کدامیک از موارد بر شمرده در فهرست وظایف تیم DBA، مستقیما به مرحله پیاده سازی پایگاه‌داده‌ها مربوط می‌شود ؟

فصل ۵

مدل‌ها و ساختارهای داده‌ای پایگاه‌داده‌ها

هدف کلی

دراین فصل ابتدا با مفهوم محیط انتزاعی یک پایگاه‌داده و بخش‌های مختلف آن آشنا خواهیم شد. در ادامه سه ساختار داده‌ای اصلی پایگاه‌های داده که شامل ساختار داده‌ای سلسله مراتبی، شبکه‌ای و رابطه‌ای می‌باشند را معرفی کرده و عناصر اصلی، عملیات اصلی و نحوه نمایش ارتباط موجودیت‌ها و ویژگی‌های هر یک از ساختارهای داده‌ای شرح داده خواهند شد.

هدف رفتاری

دراین فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- محیط انتزاعی
- بخش ساختاری محیط انتزاعی
- بخش عملیاتی محیط انتزاعی
- بخش جامعیتی محیط انتزاعی
- آشنایی با ساختار داده‌ای سلسله مراتبی
- عناصر ساختاری
- طراحی پایگاه سلسله مراتبی
- عملیات در پایگاه سلسله مراتبی
- برخی ویژگی‌های ساختار (و مدل) داده‌ای سلسله مراتبی

- آشنایی با ساختار داده‌ای شبکه‌ای
- تعریف ساختار شبکه
- عناصر ساختاری
- طراحی پایگاه شبکه‌ای
- طرز نمایش ارتباط "یک به چند"
- طرز نمایش ارتباط بازگشتی
- طرز نمایش ارتباط "چند به چند"
- عملیات در پایگاه شبکه‌ای
- برخی ویژگی‌های ساختار (و مدل) داده‌ای شبکه‌ای

۱- محیط انتزاعی[1]

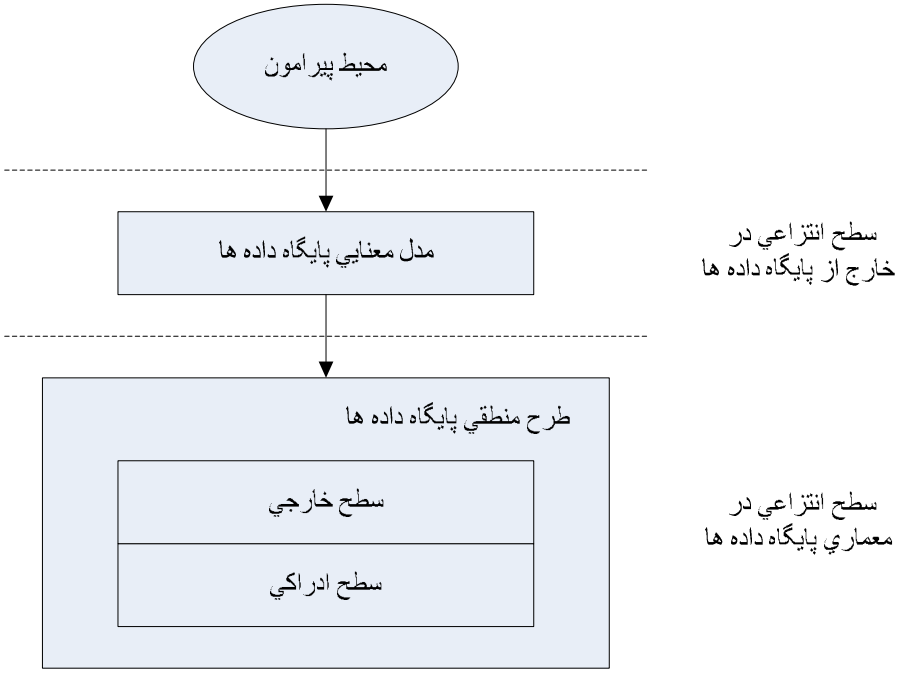
در مفاهیم پایگاه‌داده‌ها منظور از محیط انتزاعی، محیطی است که مابین سطح داخلی (محیط فایلینگ فیزیکی و فایلینگ منطقی) و سطح ادراکی است. طبیعتااین محیط باید از سطح فایلینگ مستقل عمل کند.این محیط منطقاً خود می‌تواند دارای چند سطح باشد. شکل ۵-۱ نمای کلی از محیط انتزاعی را نشان می‌دهد.

همانطور که در مدل سازی داده‌ها نیاز به امکانی برای نمایش واقعیات داریم، برای طراحی منطقی پایگاه‌داده‌ها نیز نیاز به یک مدل داده‌ای شامل یک ساختار داده‌ای داریم. مدل داده‌ای امکانی است برای طراحی منطقی پایگاه‌داده‌ها، تعریف و کنترل آن و انجام عملیات در آن. کاربر با استفاده از مدل داده‌ای می‌تواند هر سه عمل مذکور را در یک محیط انتزاعی انجام دهد. بنابراین می‌توان گفت که مدل داده‌ای تامین کننده محیط انتزاعی پایگاه‌داده‌ها است و از سه بخش اساسی تشکیل شده است:

- بخش ساختاری[2]
- بخش عملیاتی[3]
- بخش جامعیتی[4]

1. Abstract Environment
2. Structural
3. Manipulative
4. Integrity

بخش ساختاری، شامل عناصر ساختاری مدل است که همان ساختار داده‌ای اصلی و مفاهیم مربوط به آن است.
بخش عملیاتی، مجموعه امکاناتی است که به وسیله آنها عملیات مورد نظر کاربر در مجدوده ساختار داده‌ای و مبتنی بر عنصر ساختاری اساسی آن انجام می‌شود.
بخش جامعیتی، از مجموعه‌ای از قواعد (محدودیت‌های) جامعیتی تشکیل شده است که با استفاده از آنها سیستم مدیریت پایگاه‌داده‌ها می‌تواند صحت، دقت و سازگاری داده‌ها را در هر لحظه از حیات پایگاه‌داده، کنترل و تضمین کند.



شکل ۵-۱ سطوح مختلف محیط انتزاعی

با این توصیف هر ساختار داده‌ای نیز فقط بخضی از یک مدل داده‌ای است و حداقل یک عنصر ساختار اساسی دارد که به کمک آن نوع موجودیت یا نوع ارتباط و یا هر دو نمایش داده می‌شوند. در ادامه دلایلی که استفاده از ساختار داده‌ای را بیان می‌کند ارائه شده است:

- ساختار داده‌ای تامین کننده محیط انتزاعی پایگاه‌داده‌ها و چار چوب طراحی منطقی پایگاه است.
- ساختار داده‌ای مبنا (و چارچوب) طراحی دستورات تعریف داده‌ها، دستورهای عملیات در پایگاه‌داده‌ها و دستورهای کنترل داده‌ها است.
- ساختار داده‌ای مبنای طراحی DBMSهاست، یعنی با توجه به سه نوع مدل، سه رده DBMS داریم که عبارتند از:RDBMS، HDBMS وNDBMS، به نحوی که هویت "هر سیستم مدیریت پایگاه‌داده‌ها" همان مدل داده‌ای آن است.
- ساختار داده‌ای ضابطه‌ای است برای مقایسه DBMSها و نیز ارزیابی آنها.
- ساختار داده‌ای مبنایی است برای ایجاد و گسترش تکنیکهای طراحی پایگاه‌داده‌ها.
- ساختار داده‌ای، مبنای پژوهش در دانش و تکنولوژی پایگاه‌داده‌ها است.

پایگاه‌داده‌ها در محیط انتزاعی، مجموعه‌ای است از نمونه‌های متمایز عناصر ساختاری یک ساختار داده‌ای. در مفاهیم پایگاه‌داده، نمونه‌های کلاسیک ساختار داده‌ای عبارتند از:

- ساختار داده‌ای سلسله مراتبی[1]
- ساختار داده‌ای شبکه[2]
- ساختار داده‌ای رابطه‌ای[3]

با توجه به اینکه ساختارهای داده‌ای سلسله مراتبی و شبکه‌ای از رده خارج شده‌اند، لذا این دو ساختار جهت آشنایی خوانندگان محترم بصورت کلی توضیح داده خواهد شد. ساختار داده‌ای رابطه‌ای در فصل‌های بعدی بصورت کامل مورد بحث و بررسی قرار خواهد گرفت.

1. Hierarchical Data Structure
2. Network Data Structure
3. Relational Data Structure

۲- آشنایی با ساختار داده‌ای سلسله مراتبی

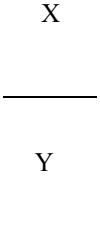
این ساختار قدیمی‌ترین ساختار داده‌ای برای طراحی منطقی پایگاه‌داده‌ها (در سطح انتزاعی) است. انسان از دیرباز با مفهوم سلسله مراتب آشنا بود و آن را برای رده بندی پدیده‌ها (در معنای عام) در جهان واقع، استفاده می‌کرد. در اواسط دهه ۱۹۶۰ میلادی، وقتی که طراحان سیستم‌های نگهداری داده- داده پردازی دراندیشه یافتن نوعی "رکورد ساختمند" بودند (به جای "رکورد خطی مسطح") همین مفهوم آشنای "سلسله مراتب" را برای نمایش داده‌های مورد نظرشان، مناسب یافتند.

۲-۱ عناصر ساختاری

سلسله مراتب در اساس نوعی درخت است که مفهوم آشنایی است. در ساختار داده‌ای سلسله مراتبی، دو عنصر ساختاری اساسی وجود دارد:

- نوع رکورد
- نوع پیوند پدر–فرزندی

بطوریکه خواهیم دید، نوع رکورد برای نمایش نوع موجودیت به کار می‌رود. بین هر دو نوع رکورد بلافصل از یک مسیر سلسله مراتب، پیوند پدر–فرزندی وجود دارد و با این پیوند ارتباط بین دو نوع موجودیت نمایش داده می‌شود. این نوع پیوند، طبق ماهیت مفهوم سلسله مراتب، ارتباط یک به چند را می‌تواند نمایش دهد. در سوی "یک" ارتباط (۱)، نوع رکورد پدر و در سوی "چند" (N) آن، نوع رکورد فرزند قرار دارد.

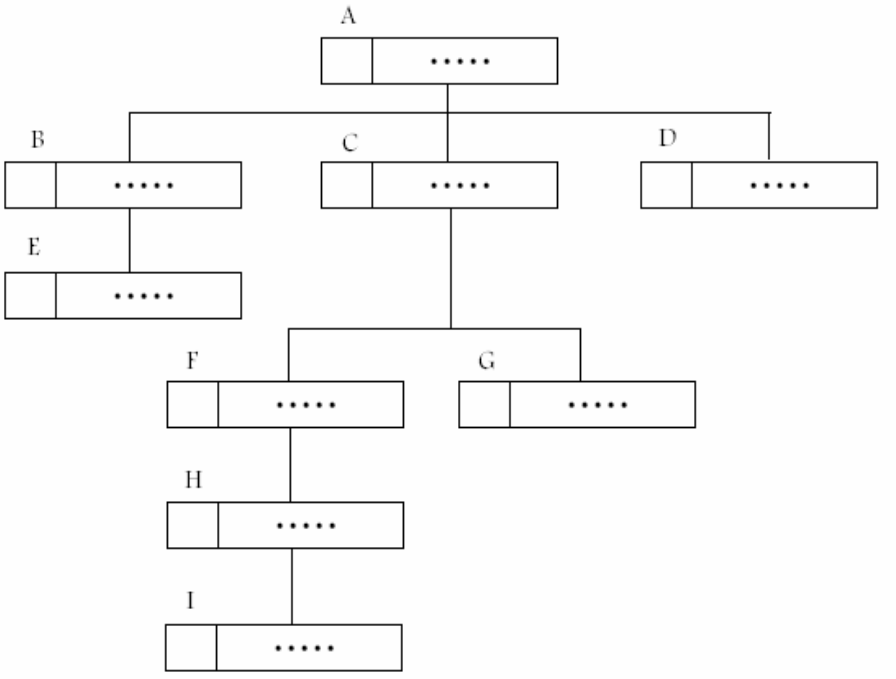


شکل ۵-۲ مفهوم سلسله مراتب

در شکل ۵-۲، رکورد نوع Y فرزند و رکورد نوع X، پدر است. هر نوع رکورد (از هر سطح سلسله مراتب)، می‌تواند از صفر تا n نوع رکورد فرزند (در سطح بلافاصله پائین تر و در مسیرهای مختلف طبعا) داشته باشد. پس هر نوع رکورد فرزند از یک سطح، خود می‌تواند پدر انواعی از رکوردها در سطح بلافاصله پائین تر و در چند مسیر باشد. بدینسان، سلسله مراتبی از انواع رکوردها در سطوح و مسیرهای مختلف ایجاد می‌شود.

- ریشه سلسله مراتب، رکورد نوع پدر در بالاترین سطح است. بقیه انواع رکوردها، وابستگان ریشه در سطوح مختلف، هستند.
- ریشه در حالت خاص ممکن است اصلاً فرزند نداشته باشد و دراین صورت سلسله مراتب را " فقط ریشه" می‌گوییم.

برای مثال یک نوع سلسله مراتب با تعداد ۹ نوع رکورد در شکل ۵-۳ نشان داده شده است.



شکل ۵-۳ نمایش ارتباطی نه نوع رکورد در ساختار سلسله مراتبی

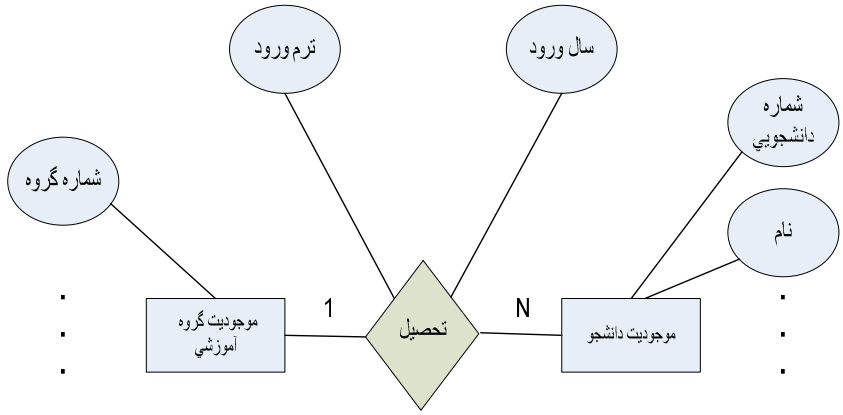
لازم به ذکر است که ترتیب پر شدن زیر مجموعه‌ها بر اساس نیاز اطلاعاتی بوده و هیچ نوع نظم خاصی را ندارد. همچنین لازم بذکر است که هر نوع رکورد فرزند، تنها یک نوع رکورد پدر دارد.

توجه: یک نوع سلسله مراتب، از لحاظ نظری می‌تواند هر تعداد مسیر داشته باشد (عرض) و عمق هر مسیر هم محدودیتی ندارد. اما در سیستم سلسله مراتبی عرض و عمق یک نوع سلسله مراتب معمولاً یک سقف حداکثر (مثلا ١٦ سطح) دارد.

۲-۲ طراحی پایگاه سلسله مراتبی

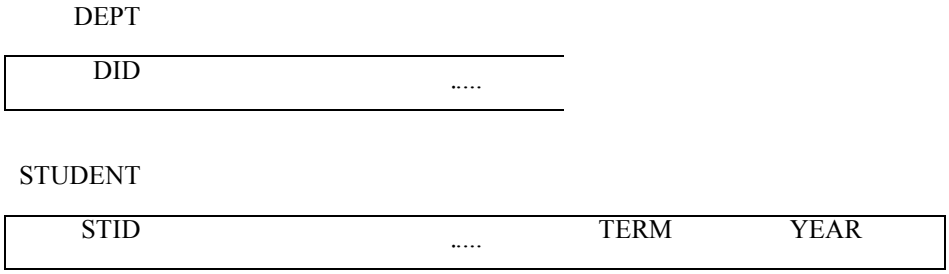
در سطح انتزاعی (از دید کاربر) پایگاه‌داده سلسله مراتبی، مجموعه‌ای است منطقاً "منظم" از نمونه‌های متمایز یک یا بیش از یک نوع سلسله مراتب. (روشن است که در سطح فیزیکی، نهایتا مجموعه‌ای است از فایلهای ذخیره شده بهم مرتبط بر اساس همان پیوندهای منطقی و با ساختار مشخص).

لازم به ذکر است که در هر نمونه سلسله مراتب، یک نمونه متمایز از ریشه داریم (عامل تمایز، کلید ریشه است). هر نمونه از یک رکورد از یک سطح می‌تواند از صفر تا n نمونه از هر نوع رکورد فرزند بلافصل، داشته باشد. نمونه‌های هر نوع رکورد فرزند ذیل یک نمونه پدر مشخص هم از یکدیگر متمایزند. عامل تمایز، همان کلید رکورد فرزند است. به عنوان اولین مثال، نمودار ER شکل ٥-٤ را در نظر می‌گیریم.این نمودار نوع ارتباط بین موجودیت‌ها دانشجو، و گروه آموزشی را نشان می‌دهد:



شکل ٥-٤ نمودار ER مثال ۱

فرض می‌کنیم که یک دانشجو فقط در یک گروه آموزشی تحصیل می‌کند (وابسته به یک گروه آموزشی است). چندی ارتباط "تحصیل کردن " دراین مثال، یک به چند است، بنابراین یک نوع سلسله مراتب به صورت شکل ۵-۵ طراحی می‌کنیم:



شکل ۵-۵ سلسله مراتب نوع DEST

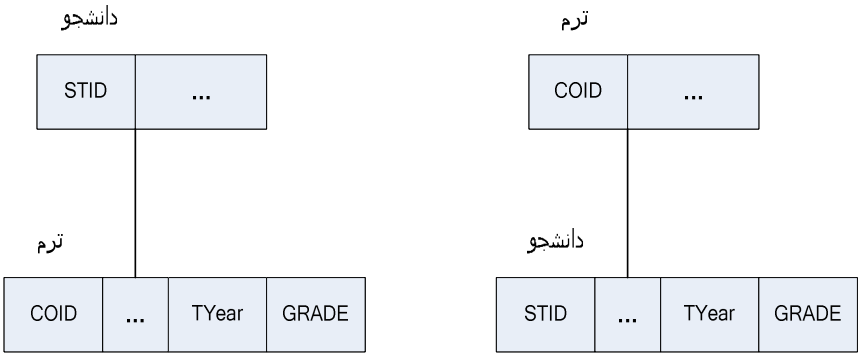
در این طرح ساده، رکورد نوع DEPT ریشه- پدر و رکورد نوع STUD وابسته بلافصل (فرزند) است. فقط فیلد صفت شناسه دو نوع موجودیت رانشان داده‌ایم. صفت ارتباط، فیلدی از نوع رکورد فرزند می‌شود. پس پایگاه‌داده سلسله مراتبی مجموعه‌ای است منطقاً منظم از نمونه‌های متمایز یک یا بیش از یک نوع سلسله مراتب.

مثال ۲: در مثال قبل، چندی ارتباط "یک به چند" است. دراین مثال، یک ارتباط با چندی "چند به چند" را مطرح می‌کنیم.

نمودار "دانشجو- درس" را در نظر می‌گیریم. می‌دانیم که چندی ارتباط "انتخاب"، چند به چند است. اگر ارتباط چند به چند را ارتباط یک به چند دو سویه بدانیم برای نمایش‌این ارتباط با عناصر ساختار داده‌ای سلسله مراتبی، منطقاً به دو نوع سلسله مراتب نیاز داریم. دو روش برای طراحی وجود دارد:

روش۱: طراحی دو نوع سلسله مراتب جداگانه

در این طراحی، یک نمونه رکورد فرزند، می‌تواند تکرار شود: مثلا در سلسله مراتب نوع STCO، یک نمونه رکورد نوع درس ذیل هر نمونه رکورد دانشجو که آن درس انتخاب کرده باشد، می‌آید. چنین است برای رکورد نوع STUD در سلسله مراتب نوع COST.



شکل ۵-۶ نمایش دو نوع سلسله مراتب جداگانه برای ارتباط چند به چند

این تکرار، در واقع همان پدیده نامطلوب افزونگی است.این افزونگی البته ماهیتا منطقی یا ادراکی است. اما اگر منجر به بروز افزونگی در محیط فیزیکی ذخیره سازی شود می‌تواند:

- سبب بروز ناسازگاری داده‌ای شود (به یاد داشته باشیم که افزونگی کنترل نشده و ناسازگاری داده‌ای پشت و روی یک سکه‌اند !).
- سبب بروز فزونکاری (بیشکاری)در سیستم در انجام عملیات در پایگاه‌داده‌ها شود.

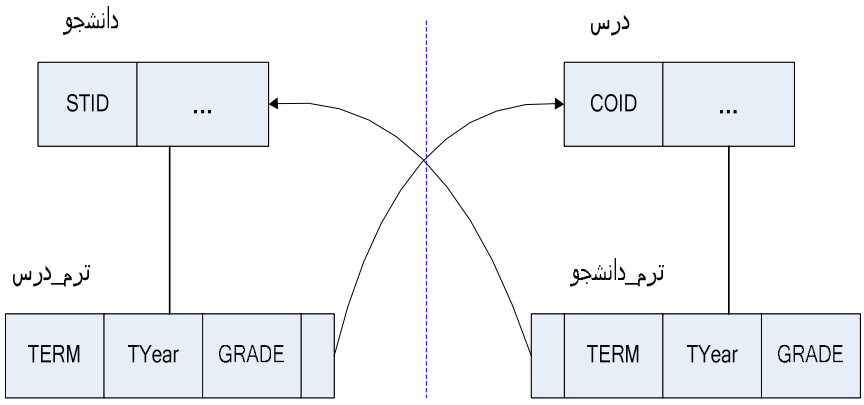
توجه: می‌توان ارتباط با چندی M:N را با یک نوع سلسله مراتب هم طراحی کرد، مثلا فقط نوع سلسله مراتب STCO را داشت. اما چون منطق طبیعی در سلسله مراتب، با توجه به ماهیت پیوند پدر- فرزندی، ورود از ریشه و طی کردن مسیر مناسب برای رسیدن به نمونه فرزند مورد نظر است، برای پاسخگویی به پرسشی که در آن کلید نمونه فرزند داده شده باشد و نمونه (های) پدر مورد درخواست باشد، دیگر نمی‌توان منطقاً بر اساس رویه مبتنی بر پیوند پدر- فرزندی عمل کرد. بعلاوه با داشتن فقط یک نوع سلسله مراتب برای ارتباط M:N و (عدم وجود نوع سلسله مراتب قرینه)، مشکلاتی ناشی از ماهیت پیوند پدر-فرزندی در عملیات ذخیره سازی در پایگاه‌داده‌ها بروز می‌کند (از جمله عدم امکان درج نمونه فرزند بدون وجود نمونه پدر و یا لزوم انجام بهنگام سازی منتشر شونده)

روش ۲: طراحی دو نوع سلسله مراتب به هم بسته: تکنیک VPCL[1]

دراین روش، از یک نوع رکورد مجازی (حاوی نشان نما: اشاره گر و صفات ارتباط) استفاده می‌شود. به این نوع رکورد اصطلاحا "فرزند مجازی" گفته می‌شود. هر نمونه از فرزند مجازی به یک نمونه " پدر مجازی " اشاره می‌کند. چنین پیوندی را پیوند پدر- فرزندی مجازی می‌نامیم. در شکل ۵-۷ این روش دیده می‌شود.

در این طرح نوع رکورد COPTR، فرزند مجازی نوع رکورد COURSE واین نوع رکورد، پدر مجازی نوع رکورد COPTR است و نوع رکورد STPTR، فرزند مجازی نوع رکورد STUD و این نوع رکورد، پدر مجازی نوع رکورد STPTR است. اطلاعاتی که هم به پدر و هم به فرزند مربوط می‌شود (صفات ارتباط)، در نوع رکورد اشاره گر مجازی گنجانده می‌شود: دراینجا صفات TR، YRYR و GRADE.

در این طراحی، نمونه‌های دو نوع رکورد STUD و COURSE دیگر تکرار نمی‌شوند. این کاهش افزونگی، البته به قیمت پیچیده‌تر شدن طراحی و پیاده سازی پایگاه‌داده‌ها تمام می‌شود (به ویژه اگر سلسله مراتب مورد نیاز، بزرگ باشد (در عرض و عمق)). شایان ذکر است که از دیدگاه نظری، این تکنیک ربطی به ماهیت ساختار داده‌ای سلسله مراتبی ندارد، بلکه امکانی است برای نمایش بهم بستگی دو یا بیش از دو نوع سلسله مراتب و از آنجا، امکانی است برای نمایش ارتباط چند به چند.

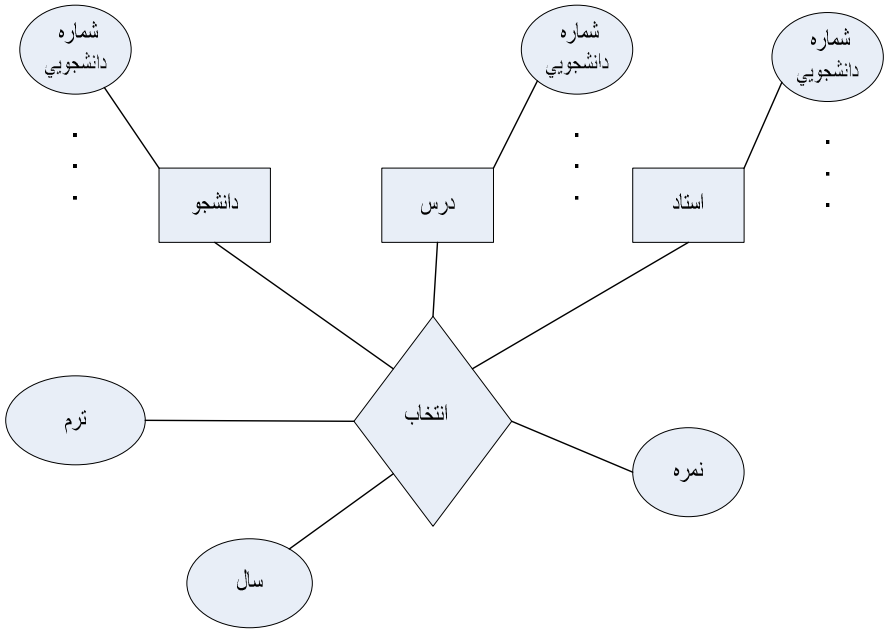


شکل ۵-۷ پیوند پدر- فرزندی مجازی

1. Virtual Parent Child Link

توجه داشته باشیم که PCL و VPCL از نظر مفهومی، یکسان هستند، تفاوت در طرز پیاده سازی آنها است: PCL با استفاده از توالی سلسله مراتبی پیاده سازی می‌شود و VPCL با استفاده از نشانه رو فیزیکی حاوی آدرس و یا نشانه رو منطقی حاوی کلید، از رکورد فرزند مجازی به رکورد پدر مجازی.این تکنیک سبب افزایش کارایی سیستم در پاسخ گویی به بعض پرسشها می‌شود.

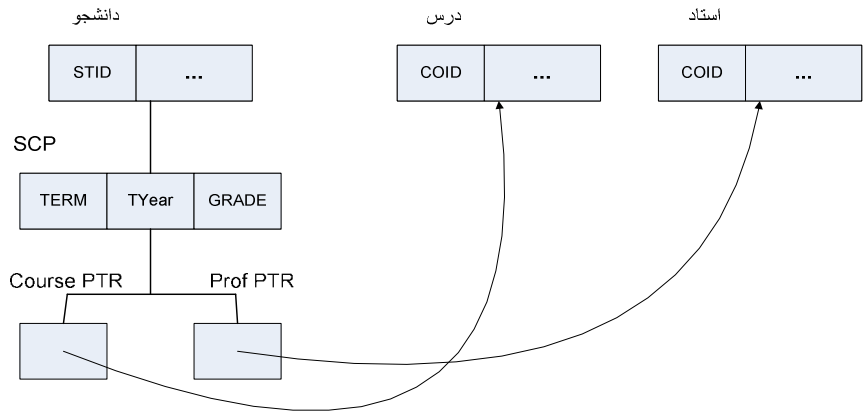
مثال ۳: دراین مثال، یک ارتباط سه موجودیتی را در نظر می‌گیریم: ارتباط بین انواع موجودیت‌های دانشجو، درس و استاد که در بحث مدل سازی معنایی داده‌ها دیدیم. نمودار ER مربوطه در شکل ۵–۸ نشان داده شده است:



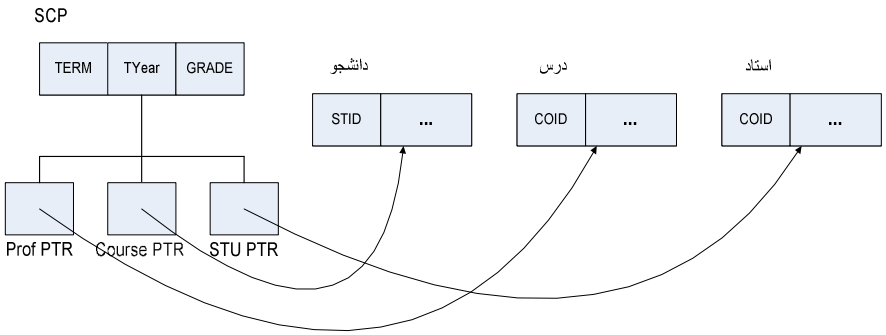
شکل ۵–۸ نمودار ER ارتباط بین سه موجودیت

برای تبدیل این نمودار به طراحی پایگاه سلسله مراتبی، دو روش ممکن به صورت زیر خواهند بود:

روش اول:



روش دوم:



شکل ۵-۹ روش‌های برای نمایش ارتباط سه موجودیتی با ساختار سلسله مراتبی

۲-۳ عملیات در پایگاه سلسله مراتبی

با توجه به عناصر ساختاری اساسی ساختار داده‌ای سلسله مراتبی، برخی از دستورات لازم، بطور کلی چنین‌اند:

- دستور بازیابی یک نمونه رکورد نوع ریشه.
- دستور بازیابی نمونه ریشه بعدی.
- دستور بازیابی یک نمونه فرزند از یک نمونه پدر (بلافصل یا از نمونه پدران تا ریشه).

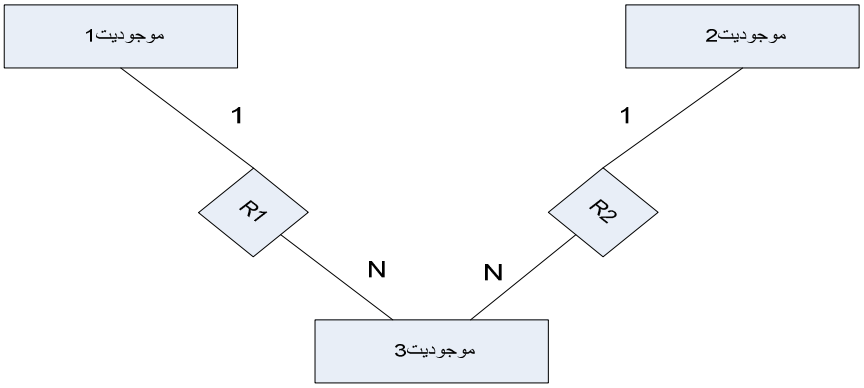
- دستور بازیابی نمونه فرزند بعدی از یک نمونه پدر.
- دستور درج یک نمونه فرزند زیر یک نمونه پدر (مشخصات مسیر، در صورت نیاز باید داده شود).
- دستور حذف یک (یا چند) نمونه فرزند از یک نمونه پدر(مشخصات مسیر، در صورت نیاز باید داده شود).
- دستور بهنگام سازی یک (یا چند) نمونه فرزند از یک نمونه پدر (مشخصات مسیر، در صورت نیاز باید داده شود).

۲-٤ برخی ویژگی‌های ساختار (و مدل) داده‌ای سلسله مراتبی

در ادامه برخی ویژگی‌های ساختار داده‌ای سلسله مراتبی به اختصار بیان شده است:

- مبنای ریاضی ندارد (حداقل در سیستم موجود) و ازاین رو میزان انتزاع آن به‌اندازه انتزاع ساختار جدولی نیست (البته گراف یک سویه، بی‌گسست و نابازگشتی می‌تواند مبنای ریاضی مناسبی برای این ساختار داده‌ای باشد).
- دو عنصر ساختاری اساسی دارد (و نه یکی).
- برای نمایش ارتباط یک به چند مناسب است (محدودیت ساختار).
- در عملیات ذخیره سازی مشکلاتی دارد.
- در نمایش ارتباط "چند به چند" دشواری دارد و نیز در نمایش ارتباط سه گانی یا با درجه بیشتر و همچنین وقتی که دو یا بیش از دو ارتباط با چندی N: 1 داشته باشیم که در آنها، در طرف N ارتباط، یک نوع موجودیت واحد وجود داشته باشد، به صورت شکل زیر.

لازم به ذکر است که دراین‌جا دو نوع موجودیت E1 وE2 ممکن است یکسان هم باشند، به بیان دیگر دو یا بیش از دو نوع ارتباط بین دو نوع موجودیت وجود داشته باشد).



شکل ۵-۱۰ نمودار توع ارتباط ۱ به چند

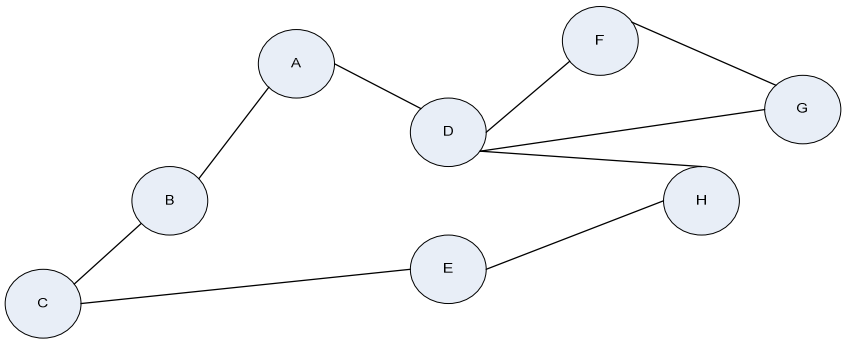
۳- آشنایی با ساختار داده‌ای شبکه‌ای

این ساختار (که گاه به آن ساختار پلکس هم می‌گویند)، نخستین بار در سال ۱۹٦٦ توسط یک گروه کاری به نام DBTG وابسته به ANSI پیشنهاد شده است. در سال ۱۹۷۱ اولین سیستم مدیریت پایگاه‌داده‌های شبکه‌ای (NDBMS) مورد تائید ANSI، به نام IDMS توسط همان گروه کاری طراحی شد.این سیستم گاه به سیستم کوداسیل و مدل داده‌ای شبکه‌ای به مدل داده‌ای کوداسیل نیز موسوم است. سیستم‌های شبکه‌ای دیگر عبارتند از:

- VAX- DBMS
- IMAGE
- .1100-DMS

۳-۱ تعریف ساختار شبکه

شبکه نوعی گراف جهت دار است که در آن گره‌ها به کمک یال‌هایی بهم بسته‌اند. در شکل ۵-۱۰ یک شبکه با هشت گره دیده می‌شود.این ساختار را می‌توان گسترش یافته ساختار سلسله مراتبی دانست بهاین معنا که در آن هر نوع گره فرزند می‌تواند بیش از یک نوع گره پدر داشته باشد و بنابرین ازاین نظر، محدودیت عدم تقارن ساختار سلسله مراتبی را ندارد.



شکل ۵-۱۱ نمایش کلی ساختار شبکه

۳-۲ عناصر ساختاری

در ساختار داده‌ای شبکه‌ای دو عنصر ساختاری اساسی وجود دارد:

- نوع رکورد
- نوع مجموعه

نوع رکورد: برای نمایش نوع موجودیت به کار می‌رود (مثل ساختار سلسله مراتبی). در یک نوع رکورد طبعا صفاتی وجود دارد که می‌توانند ساده تکرار شونده (بردار در اصطلاح شبکه‌ای) یا مرکب تکرار شونده (گروه تکرار شونده در اصطلاح شبکه‌ای) هم باشند.

نوع مجموعه: در اساس برای نمایش ارتباط 1:N بین دو (چند) نوع موجودیت بکار می‌رود. نوع مجموعه (که به آن مجموعه کوداسیل هم می‌گویند)، از سه جزء تشکیل شده است:

- نام مجموعه
- یک نوع رکورد مالک
- یک نوع رکورد عضو

هر چند این عنصر ساختاری در اساس برای نمایش ارتباط "یک به چند" بین دو یا بیش از دو نوع موجودیت بکار می‌رود، اما می‌توان ارتباط "چند به چند" را هم با آن نمایش داد. در اینجا لازم است به نکات زیر توجه گردد:

- مالک یک نوع مجموعه، می‌تواند مالک در نوع مجموعه دیگر باشد.
- مالک یک نوع مجموعه می‌تواند عضو در نوع مجموعه دیگر باشد.
- عضو یک نوع مجموعه می‌تواند مالک در نوع مجموعه دیگر باشد.
- عضو یک نوع مجموعه می‌تواند عضو در نوع مجموعه دیگر باشد. به این عضو، عضو مشترک یا پیوند دهنده می‌گوئیم و در نمایش ارتباط با چندی N:M بکار می‌آید.

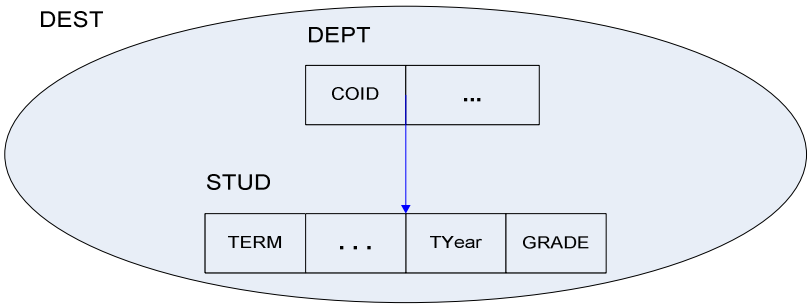
یک نوع مجموعه، می‌تواند نمونه‌هایی داشته باشد. در هر نمونه مجموعه، یک نمونه مشخص و متمایز از رکورد نوع مالک و $n \geq 0$ نمونه متمایز از رکورد نوع عضو وجود دارد. پایگاه‌داده‌های شبکه‌ای در محیط انتزاعی (از دید کاربر) مجموعه‌ای است منطقاً "منظم" از نمونه‌های یک (یا چند) نوع مجموعه.

۳-۳ طراحی پایگاه شبکه‌ای

پایگاه‌داده شبکه‌ای روش خاص خود جهت نمایش ارتباط بین موجودیت‌ها را دارا است. با توجه به اهمیت ارتباط‌های از نوع یک به چند و چند به چند، نحوه نمایش و پیاده سازی این دو نوع ارتباط در زیر توضیح داده شده‌اند.

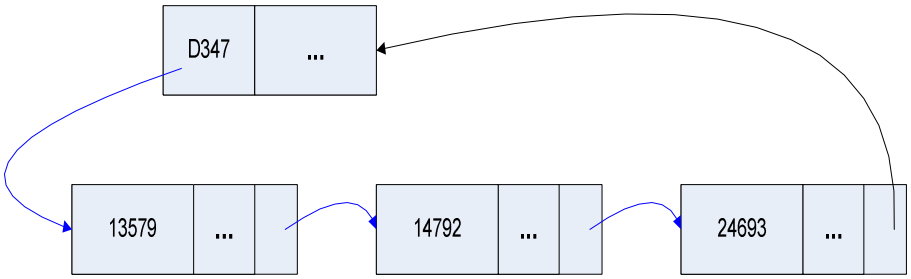
۳-۳-۱ طرز نمایش ارتباط " یک به چند"

در این نوع حالت نوع رکورد سوی " یک " را مالک و نوع رکورد سوی " چند " را عضو در نظر می‌گیریم.



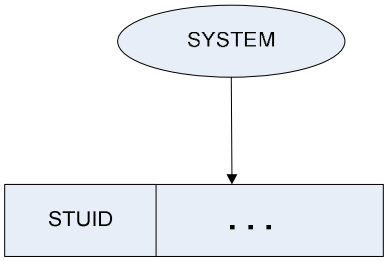
شکل ۵-۱۲ نوع مجموعه کوداسیل DEST

در این نمودار که به آن نمودار باخمن[1] هم می‌گویند، نوع رکورد DEPT، مالک، و نوع رکورد STUD، عضو است.این نمودار یک نوع مجموعه کوداسیل به نام DEST را نمایش می‌دهد. می‌بینیم که در حالت ارتباط یک به چند، صفت (صفات) ارتباط، فیلد(هایی) از نوع رکورد عضو هستند. در شکل ۵-۱۳ نمونه‌ای از نوع مجموعه DEST دیده می‌شود.



شکل ۵-۱۳ دو نمونه از نوع مجموعه DEST

یک نوع مجموعه، می‌تواند چند نوع رکورد عضو داشته باشد که البته یک حالت خاص است و در سیستم‌های شبکه‌ای موجود پذیرفته نیست. رکورد نوع مالک در یک نوع مجموعه می‌تواند مالک در چند نوع مجموعه باشد. گونه خاصی از نوع مجموعه وجود دارد که رکورد نوع مالک ندارد و در واقع " سیستم " خود مالک آن است. شکل ۵-۱٤ نمونه‌ای از رکورد تحت مالکیت سیستم را نشان می‌دهد.

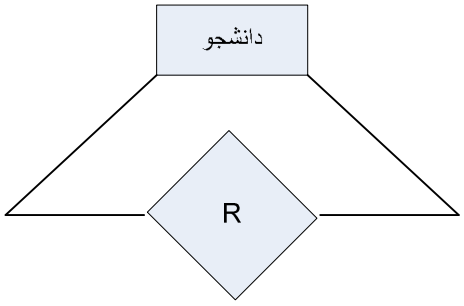


شکل ۵-۱٤ نوع رکورد تحت مالکیت " سیستم "

1. Bachman Diagram

ارتباط بازگشتی

نوع مجموعه ممکن است بازگشتی باشد. دراین گونه نوع مجموعه، یک نوع رکورد هم مالک است و هم عضو (این نوع مجموعه غالبا در سیستمهای شبکه‌ای مجاز نیست). شکل ۵-۱۵ ارتباط موجودیت E با خودش را نشان می‌دهد:



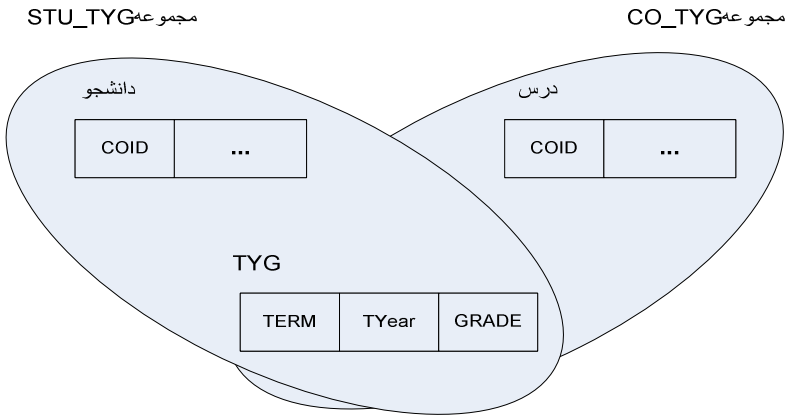
شکل ۵-۱۵ ارتباط موجودیت با خودش

۳-۳-۲ طرز نمایش ارتباط "چند به چند"

برای نمایش ارتباز چند به چند، از یک نوع رکورد پیوند دهنده استفاده می‌کنیم که عضو مشترک بین دو یا بیش از دو نوع مجموعه است (تعداد نوع مجموعه‌ها همان تعداد نوع موجودیتهای شرکت کننده در ارتباط است. به بیان دیگر به تعداد درجه ارتباط، نوع مجموعه با عضو مشترک داریم).این نوع رکورد (که می‌تواند بدون یا دارای فیلد (های) داده‌ای باشد) را گاه رکورد ساختگی می‌نامیم. فیلد(های) این رکورد (در سطح طراحی)، همان صفت (صفات) ارتباط N:M است. برای نمونه در شکل۵-۱٦ ارتباط " انتخاب " بین دانشجو و درس نشان داده شده است. این ارتباط به کمک دو نوع مجموعه نمایش داده شده است که در آنها رکورد TYGR رکورد پیوند دهنده است.

حال برای درک بهتر موضوع می‌خواهیم پایگاه‌داده شبکه‌ای را در سطح انتزاعی توصیف کنیم. پایگاه‌داده شبکه‌ای مجموعه ایست منطقاً منظم از نمونه‌های متمایز یک یا چند مجموعه کوداسیل. در اینجا ذکر این نکته اهمیت دارد که ساختار چند حلقه‌ای

بویژه گونه‌های کاملترش می‌تواند ساختار مناسبی باشد و می‌توان این ساختار را با ساختار شاخص و یا ساختار مستقیم از طریق درهمسازی، ترکیب کرد.



شکل ۵-۱۶ نمایش ارتباط " چند به چند " بین دانشجو و درس

۳-٤ عملیات در پایگاه شبکه‌ای

با توجه به عناصر ساختاری ساختار داده‌ای شبکه‌ای، برخی دستورات بطور کلی عبارتند از:

- دستور بازیابی یک نمونه از یک نوع رکورد مالک
- دستور بازیابی نمونه بعدی از یک نوع رکورد مالک
- دستور بازیابی یک نمونه از یک نوع رکورد عضو با داشتن نمونه مالک (ها)
- دستور بازیابی نمونه بعدی از یک نوع رکورد عضو با داشتن نمونه مالک (ها)
- دستور بازیابی یک نمونه مالک از یک نوع مجموعه با داشتن یک نمونه عضو از آن
- دستور درج یک نمونه نوع رکورد مالک یا نوع رکورد عضو
- دستور حذف یک نمونه نوع رکورد مالک یا نوع رکورد عضو
- دستور بهنگام سازی یک نمونه نوع رکورد مالک یا نوع رکورد عضو

۳-۵ برخی ویژگی‌های ساختار (و مدل) داده‌ای شبکه‌ای

در ادامه برخی ویژگی‌ها و محدودیت‌های ساختار داده‌ای شبکه را شرح خواهیم داد:

- مبنای ریاضی ندارد (حداقل در سیستم موجود)، ازاین رو میزان انتزاع آن در حد ساختار داده‌ای رابطه‌ای (جدولی) نیست (البته گراف می‌تواند مبنای ریاضی مناسبی برای این ساختار داده‌ای باشد).
- دو نوع عنصر ساختاری اساسی دارد.
- ماهیتا خاص نمایش ارتباطات " یک به چند" نیست، یعنی محدودیتی در نمایش ارتباطات با چندی‌های دیگر، ندارد.
- ساخت منطقی رویه بازیابی آن پیچیده تر از ساختارهای دیگر است و ناوش غیر اتوماتیک است.
- خطر بروز ناسازگاری داده‌ها نسبت به ساختار سلسله مراتبی، کمتر است.
- قواعد جامعیت ذاتی دارد.
- بعض آنومالی‌های مدل سلسله مراتبی در عملیات ذخیره سازی را ندارد.
- فزون کاری احتمالی ناشی از افزونگی که در ساختار سلسله مراتبی می‌تواند وجود داشته باشد، دراین ساختار وجود ندارد اما به علت حجم زیاد نشان نماها (اشاره گرها)، ایجاد یا اصلاًح آنها می‌تواند سبب بروز فزون کاری در سیستم شود.

تمرینات

۱. دلایل استفاده از ساختار داده‌ای را شرح دهید ؟
۲. عناصر اصلی ساختار داده سلسله مراتبی را شرح دهید ؟
۳. ویژگی‌ها و محدودیت‌های ساختار داده سلسله مراتبی را شرح دهید ؟
٤. عناصر اصلی ساختار داده شبکه‌ای را شرح دهید ؟
۵. ویژگی‌ها و محدودیت‌های ساختار داده شبکه‌ای را شرح دهید ؟
٦. نحوه پیاده سازی ارتباط بازگشتی را در ساختار داده‌ای شبکه‌ای توضیح دهید ؟

فصل ۶

پایگاه‌داده رابطه‌ای

هدف کلی

دراین فصل ابتدا تاریخچه‌ای در مورد بانک اطلاعات رابطه‌ای بیان شده و ساختار داده‌ای رابطه‌ای به تفصیل مورد بحث و بررسی قرار خواهد گرفت. در این راستا مفهوم رابطه شرح داده شده و تناظری بین مفهوم رابطه و جدول بیان خواهد شد. سپس ویژگی‌های رابطه شرح داده شده و در مورد مفاهیمی‌مانند کلیدها، نوع رابطه و مفهوم دید صحبت خواهد شد. در ادامه مفهوم قواعد جامعیت مطرح شده و انواع قواعد جامعیت توضیح داده خواهند شد.

هدف رفتاری

دراین فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- بانک اطلاعات رابطه‌ای Relational DB
- ساختار داده‌ای رابطه‌ای
- تعریف رابطه
- تناظر بین مفاهیم رابطه‌ای و مفاهیم جدولی
- تعریف صوری جدول
- ویژگی‌های رابطه
- انواع کلید در مدل رابطه‌ای
- انواع رابطه

- مفهوم دید
- قواعد کاربری
- متا قواعد
- قواعد جامعیت در مدل رابطه‌ای

۱- بانک اطلاعات رابطه‌ای[1]

پس از پایگاه‌های داده‌ای سلسله مراتبی و شبکه‌ای، که هر یک دارای ضعف‌هایی بودند، متخصصان در جستجوی مدلی بودند که دارای ساختار داده‌ای با انتزاع قوی باشد. مدل رابطه‌ای در سال ۱۹۷۰ توسط E.F.Codd ابداع گردید. این مدل دارای ساختار داده‌ای با انتزاع قوی بوده و اساساً ساختار داده‌ای در آن بر اساس یک مفهوم ریاضی بنام ***رابطه***[2] استوار است. در اینجا لازم است به این نکته توجه شود که مفهوم رابطه از دیدگاه کاد با مفهوم ریاضی آن تاحدودی متفاوت است. در اینجا برای ارائه ساده‌تر موضوعات، در مباحث این کتاب مفهوم رابطه را با مفهوم ریاضی آن مساوی در نظر می‌گیریم.

۲- ساختار داده‌ای رابطه‌ای

برای طراحی پایگاه‌داده‌ها در سطح انتزاعی پایین تر از سطح مدل سازی، به یک ساختار داده‌ای از یک مدل داده‌ای نیاز است و اساساً همین مدل داده‌ای تامین کننده محیط انتزاعی است.

در پایگاه‌داده رابطه‌ای بالاخص در محیط انتزاعی مورد استفاده کاربر، رابطه نمایشی جدولی دارد و اساساً پایگاه‌داده رابطه‌ای مجموعه‌ای است از تعدادی نوع جدول. مفاهیم ساختار جدولی عبارتند از:

- جدول
- سطر

1. Relational Database (Data Bank)
2. Relation

- ستون

هر جدول از نظر محتوای داده‌ای مجموعه‌ای است از نمونه‌های متمایز از انواع سطرها و هر سطر نیز مجموعه‌ای از مقادیر است که هر کدام از یک مجموعه برگرفته شده‌اند. به هر یک از عناصر سطر یک ستون گویند. لازم به ذکر است که در ساختار جدولی، تنها عنصر ساختاری اساسی همین مفهوم نوع جدول است.

۳- تعریف رابطه

۳-۱ تعریف رابطه از دیدگاه کاد

با فرض وجود n مجموعه S1، S2، ...، Sn، رابطه R تعریف شده روی این n مجموعه، مجموعه‌ایست از n تایی‌ها (که به آن تاپل[1] می‌گویند) به نحوی که جزء اول هر n تایی از S1، جزء دوم از S2 و... مقدار بگیرد.

به بیانی دیگر R عبارتست از زیرمجموعه‌ای از ضرب کارتزین S1*S2*...*Sn. و می‌گوییم رابطه R از درجه n است. هر یک از مجموعه‌های S1، S2 و... میدان یا دامنه[2] نامیده می‌شوند. دامنه، مجموعه مقادیری است که یک صفت از رابطه مشخص می‌توانند داشته باشند.

D1={1,2}
D2={T,F,V}

D2 ✻D1 = R1

D1	D2
1	T
1	F
1	V
2	T
2	F
2	V

شکل ٦-۱ نمایش رابطه R1=D1* D2

1. Tuple
2. Domain

برای درک بهتر موضوع مثال بالا که نشان دهنده دو مجموعه و حاصل ضرب بین آن دو است، توجه کنید. شاید این مثال تا حدودی شفافیت لازم را برای خواننده نداشته باشد. اکنون مثال دیگری را با جزئیات بیشتر بیان می‌کنیم. فرض کنید مجموعه‌های زیر را داشته باشیم:

- S1 مجموعه مقادیر شماره دانشجویان
- S2 مجموعه اسامی‌دانشجویان
- S3 مجموعه مقادیر سطوح تحصیلی در دانشگاه
- S4 مجموعه مقادیر رشته‌های تحصیلی
- S5 مجموعه مقادیر شماره گروه‌های آموزشی

در این صورت STT با پنج صفت خاصه مرتبط با پنج مجموعه بالا، یک رابطه بوده و بصورت زیر نوشته می‌شود:

STT(STID,STNAME,STDEG,STMJR,STDEID)

جدول ٦-٢ نمایش جدولی رابطه STT

STT

STID	STNAME	STDEG	STMGR	STDEID
1	Reza	Bs	computer	V532
2	Hamid	ms	math	V135
…	…	…	…	…

۳-۲ تعریف رابطه

با فرض وجود n میدان D1، D2، ...، Dn (نه لزوما متمایز)، رابطه R از دو قسمت تشکیل شده است:

۱- مجموعه عنوان: مجموعه‌ای نامدار است از n صفت به صورت Ai: Di که در آن هر Ai نام یک صفت است و هر Di نام میدان صفت است. لازم به ذکر است که Aiها از یکدیگر متمایز هستند. به این مجموعه شمای رابطه[1] نیز می‌گویند که به صورت شماتیک چنین است:

1. Relation Schema

$$\{ < A_1 : D_1 > , < A_2 : D_2 > , \dots , < A_n : D_n > \}$$

۲- مجموعه بدنه: مجموعه‌ایست از m تاپل t بنحوی که t خود مجموعه‌ایست از n عنصر بصورت Ai: vi که در آن vi مقداری است از نوع (میدان) Di.

$$\{ < A_1 : v_{i1} > , < A_2 : v_{i2} > , \dots , < A_n : v_{in} > \}$$
$$(i = 1 , 2 , 3 , \dots , m)$$

- پیکر رابطه را گاها بسط[1] رابطه یا حالت رابطه[2] هم می‌گویند.
- مقدار n را درجه[3] رابطه یا آریتی[4] می‌گویند.
- مقدار m را کاردینالیتی[5] رابطه می‌گویند.

۳-۳ تناظر بین مفاهیم رابطه‌ای و مفاهیم جدولی

همانطور که می‌دانید برای پیاده سازی مدل رابطه‌ای در محیط انتزاعی از رابطه استفاده می‌شود. برای پیاده سازی مدل رابطه‌ای در ساختار پایگاه‌داده از مفهوم جدول استفاده می‌شود.

جدول ٦-۳ تناظر بین اجزاء دو مفهوم رابطه و جدول

اجزاء مفهوم رابطه	اجزاء مفهوم جدولی
رابطه	جدول
تاپل	سطر
صفت	ستون
میدان	مجموعه مقادیر ستون
درجه	تعداد ستون‌ها
کاردینالیتی	تعداد سطرها

مفاهیم جدول و رابطه تا حدود زیادی به یکدیگر نزدیک بوده و در مواردی به

1. Extension
2. Relation State
3. Degree
4. Arity
5. Cardinality

اشتباه بجای یکدیگر مورد استفاده قرار می‌گیرند. در جدول ٦-۳ تناظر بین دو مفهوم رابطه و جدول نشان داده شده است.

۳-٤ تعریف صوری جدول

فرض می‌کنیم U مجموعه‌ای از نمادها موسوم به صفات رابطه‌ای باشد و برای هر$A \in U$ مجموعه‌ای از مقادیر به نام Domain(A) وجود داشته باشد. نماد نمایشگر ستون جدول است و Domain(A) مقادیری هستند که در ستون A ظاهر می‌شوند.
فرض می‌کنیم $H = \{ A_1,A_2,...,A_n\}$ مجموعه‌ای متناهی از صفات رابطه‌ای باشد. می‌دانیم که ضرب کارتزین

$$\text{Domain } (A_1) * \ldots * \text{Domain } (A_n)$$

از تاپلهایی به صورت $t = (a_1,a_2,...,a_n)$ تشکیل شده است که در آن:

$$1 <= i <= n \text{ , } a_i \in \text{Domain } (A_i)$$

تاپل T را می‌توان به گونه دیگری هم تعریف کرد:

$$t = \{A_1,A_2,...,A_n\} \rightarrow U \{\text{Domain } (A_i), 1 <= i <= n\}$$

به نحوی که

$$T(A_i) \in \text{Domain } (A_i)$$

ضرب کارتزین $\text{Domain } (A_1) * \ldots * \text{Domain } (A_n)$ را با tupl(H) نمایش می‌دهیم، به بیان دیگر:

$$\text{Tupl(H)} = \{t \mid t: H \rightarrow U_{A \in H} \text{Domain } (A), t(A) \in \text{Domain } (A) \text{ , for } A \in H \}$$

عناصر Tupl(H) را تاپلهای روی H می‌نامیم.

رابطه R روی میدانهای $\text{Domain } (A_1), \ldots, \text{Domain } (A_n)$ زیر مجموعه‌ای است از Tupl(H). مجموعه همه این گونه رابطه‌ها را با rel(H) نمایش می‌دهیم. حال می‌گوییم جدول عبارتست از سه تایی:

$$= (T , H , \rho)\text{ٹ}$$

که در آن: T نمادی است به عنوان جدول و $H = \{ A_1, A_2 , \ldots A_n\}$ مجموعه‌ای است از صفات رابطه‌ای که به آن سرآیند ٹ گوییم و ρ یک رابطه است، $\rho \in rel (H)$ و به آن گسترده ٹ می‌گوییم.

٤- ویژگی‌های رابطه

رابطه به عنوان تنها عنصر ساختاری اصلی در مدل رابطه‌ای برای نمایش انواع موجودیتها و انواع ارتباطات بکار می‌رود. در واقع در مدل رابطه‌ای هم نوع موجودیت و هم نوع ارتباط با مفهوم رابطه نمایش داده می‌شوند و در نتیجه هم نمونه موجودیت و هم نمونه ارتباط با مفهوم تاپل نشان داده می‌شوند. رابطه دارای چهار ویژگی به شرح ذیل می‌باشد:

- رابطه تاپل تکراری ندارد.
- زیرا بدنه رابطه مجموعه است و مجموعه نمی‌تواند عنصر تکراری داشته باشد.
- تاپل‌ها نظم ندارند.
- زیرا بدنه رابطه مجموعه است و مجموعه در حالت کلی فاقد نظم است.
- صفات رابطه نظم مکانی (از چپ به راست) ندارند.
- زیرا سرآیند رابطه مجموعه است و مجموعه در حالت کلی فاقد نظم است.
- تمام صفات تک مقداری (تجزیه نشدنی) هستند.
- زیرا در نمایش جدولی رابطه، در تقاطع هر سطر و ستون، باید یک مقدار وجود داشته باشد. به بیانی دیگر در هر تاپل، دقیقا یک مقدار برای هر صفت وجود دارد.

در ادامه برای درک بهتر موضوع مثالی ارائه شده است. در این مثال رابطه‌ای بنام S نشان داده شده است.

مثال ۱: رابطه S را که به صورت جدول ٦-٤ است در نظر بگیرید.

جدول ٦ -٤ نمایش جدولی رابطه S

S#	S NAME	STATUS	CITY
S1	S	20	CONDON
S2	J	10	PARIS
S3	B	30	PARIS
S4	C	20	LONDON
S5	A	30	ATHENS

همانگونه که مشاهده می‌کنبد در جدول بالا:

- S# نشان دهنده کلید اصلی است. (در مورد مبحث کلید در ادامه توضیح خواهیم داد).
- (S# , SNAME , STATUS , CITY) این ٤ مشخصه نشان دهنده attribute هستند.
- دامنه CITY شامل (LONDON، PARIS، ATHENS) است.
- (LONDON، PARIS، PARIS، LONDON، ATHENS) نیز نشان دهنده تعداد سطرهای موجود یا Cardinality می‌باشند.
- (S5 , A , 30 , ATHENS)نشان دهنده تعداد صفات درجه Degree است.

۵- انواع کلید در مدل رابطه‌ای

در مدل رابطه‌ای چند مفهوم اساسی در بحث کلید وجود دارد که در ادامه بررسی می‌شوند:

۵-۱ ابر کلید[1]

هر ترکیبی از صفات جدول را که یکتایی مقدار را داشته باشد، ابر کلید گویند (به صورت دلخواه). به بیانی دیگر هر زیر مجموعه عنوان رابطه که یکتایی مقدار در گسترده (بدنه) رابطه داشته باشد. تعریف دیگر ابر کلید عبارت است از هر ترکیبی از اسامی‌صفات رابطه که در هیچ دو تاپل مقدار یکسان نداشته باشد.

۵-۲ کلید کاندید[2]

کلید کاندید امکانی است برای ارجاع به "تک تاپل" در رابطه. مجمعه صفات K از

1. Super key
2. Candidate

رابطه R، یک کلید کاندید می‌باشد (مانند#S وSNAME)اگر دارای خواص زیر باشند:

- خاصیت منحصر بفرد بودن (یکتایی مقدار):

هیچ دو tuple مجزائی از رابطه R دارای مقدار یکسانی برای K نباشند.(به طور مثال هیچ دو سطر #S یکی نیست ولیCITY تکرار دارد، status، city هم نمی‌توانند با هم با شند زیرا دو فروشنده، در لندن 20 هستند).

- خاصیت غیر کاهشی(minimal):

هیچ زیر مجموعه مناسبی از K وجود ندارد که دارای خاصیت منحصر بفرد باشد.

به بیانی دیگر هر زیر مجموعه از مجموعه عنوان رابطه که دو خاصیت بالا را داشته باشد، کلید کاندید رابطه است.

در مثال فوق #s و sname کلید انحصاری هستند. خاصیت دوم را ندارند زیرا #s به تنهایی منحصر بفرد است و می‌توان اینها رو شکست مثلا #s به تنهایی هم می‌تواند کلید کاندید بشود.(هر چیزی را نمی‌توان کلید کاندید به حساب آوریم زیرا کلید کاندید نباید بشکند).

توجه:هر رابطه حداقل دارای یک کلید کاندید می‌باشد (ممکن است تکی باشد یا دو صفت و یا...).

۵-۳ کلید اصلی[1]

یکی از کلیدهای کاندید رابطه که شرایط ذیل را داشته باشد:

- شناسایی کننده نوع موجودیت (تک تاپل) در رابطه باشد. مانند شماره دانشجویی برای هر دانشجو.
- از نظر سایز دارای طول کوتاه تر باشد. بدین معنی که مثلا بین دو کلید کاندید که یکی از نوع رشته ۲۰ تایی و دیگری یک عدد ٤ بایتی است، کلیدی که دارای نوع عدد ٤ بایتی است، برای کلید اصلی بودن بهتر است.

۵-٤ کلید بدیل[1]

1. Primary Key

هر کلید کاندید بغیر از کلید اصلی را کلید بدیل گویند.

۵-۵ کلید خارجی[۲]

دو رابطه R1 و R2 را در نظر بگیرید. هر زیر مجموعه از صفات رابطه R2 که هر مقدار معلومش با یک مقدار از کلید کاندید R1 برابر باشد، کلید خارجی در رابطه R2 است.

نقش کلید خارجی برای نمایش ارتباطات بین انواع موجودیت‌ها (و در نتیجه بین نمونه‌های آنها) بکار می‌رود. برای درک بهتر موضوعات کلید به مثال زیر توجه فرمایید:

مثال ۲: کلیدی که ارتباط جدول مختلف را مشخص و برقرار می‌نماید.

جدول ٦-۵ جدول مثال ۲

C#	CNAME	STATUS	NO….
10	Pen	Q	5
10	Pen	1	10
10	Pen	2	7
10	Ruler	Q	

C# و status می‌توانند کلید کاندید باشند کلید کاندیدی که می‌تواند کلید اصلی بشود ولی به تنهایی نمی‌توانند کلید کاندید باشند.

جدول فروشنده‌ها (سازنده کالا)

جدول ٦-٦ جدول فروشنده‌ها

S#	SNAME	C#	Tell#
100	a	10	
105	b	11	

1. Alternate Key
2. Foreign key

توجه: در جدول فوق #s کلید کاندید می‌باشد.

جدول کالا

جدول ٦-٧ جدول کالا

C#	CNAME	STATUS	NO…
10	Pen	Q	5
10	Pen	1	7
10	Pen	2	10
11	ruler	Q	

نکته: #C، کلید خارجی برای ارتباط بین دو جدول کالا و فروشنده می‌باشد.
نکته: #C، در جدول فروشنده کلید خارجی و در جدول کالا کلید اصلی می‌باشد.

نکاتی در مورد کلید خارجی

فرض کنید که رابطه R2 یک رابطه مبنا باشد آنگاه یک کلید خارجی در رابطه R2 زیر مجمعه‌ای از مجموعه صفات R2 مانند FR می‌باشد به نحوی که:

- رابطه مبنایی به نام R1 با کلید کاندید CKوجود داشته باشند.
- برای تمامی‌مواقع هر مقدار از FK در مقدار فعلی R2 با مقدار CK در بعضی از Tupleها در مقدار قبلی یکسان باشند.

نکته: اگر صفتی بتواند مقدار تهی (null) بگیرد، نمی‌تواند کلید باشد (مانند #tell (شماره تلفن)).

از ویژگی‌های RDBMS در پایگاه‌داده رابطه‌ای در مورد جداول این است که:

- داده‌ها توسط کاربر و به صورت جداول دریافت می‌شوند.
- عملکردهایی که کاربر می‌تواند از آنها استفاده نماید که آنها به کاربر این اجازه را می‌دهند تا بتوانند جداول جدیدی را ایجاد نمایند.

مثال: بخشی از جداول یک سیستم اداری به صورت زیر است:

کارمند EMP

جدول ٦-٨ جدول Employee

EMP#	SNAME	DEPT#	SALARY
E1	LOP	D1	40
E2	CHEN	D1	42
E3	FIN	D2	30
E4	SAI	D2	35

بخش‌ها DEPT

جدول ٦-٩ جدول Department

DEPT# #سازمان	DNAME نام سازمان	بودجه
D1	MARKETING	40
D2	DEVELOPMENT	12
D3	RESEARCH	5

SELECT * FROM DEPT WHERE (بودجه > 8)

نکته: در عبارت بالا علامت * نمایانگر همه صفت‌ها است.

در نتیجه خروجی حاصل از عبارت بالا به صورت جدول زیر خواهد بود. همانگونه که مشاهده می‌شود داده‌ها به صورت جدول وارد سیستم شده‌اند و گزارش خواسته شده نیز بر اساس یک سری از عملگرهای استاندارد ارائه شده و نتیجه به صورت جدول (بر اساس نوع عملگر) گزارش می‌گردد.

جدول ٦-١٠ خروجی حاصل از عبارت ذکر شده

#سازمان	نام سازمان	بودجه
D1	MARKETING	10
D2	DEVELOPMENT	12

نکته: خروجی‌ها از همان نوع ورودی‌ها می‌باشند.

نکته: خروجی هر عملگر می‌تواند به عنوان ورودی عملگر دیگر بکار رود. این نوع

عبارت را عبارت تو در تو[1] می نامند.

٦- انواع رابطه

انواع رابطه عبارتند از

رابطه نامدار[2]

رابطه‌ایست که با یک نام به سیستم معرفی شده باشد.

رابطه مبنا[3]

نوعی رابطه نامدار است که استقلال وجودی دارد و مشتق از رابطه‌های دیگر نیست و داده‌های ذخیره شده متناظر دارد.

رابطه مشتق[4]

رابطه‌ایست که به کمک یک عبارت رابطه‌ای بر حسب رابطه‌های نامدار دیگر و نهایتا برحسب رابطه‌های مبنا تعریف می‌شود.

مفهوم دید

نوعی رابطه نامدار که مشتق از رابطه‌های دیگر است و ماهیتا رابطه‌ای مجازی[5] است، یعنی داده‌های ذخیره شده خاص خود را ندارد.

انواع دیگری از رابطه‌ها در کتابهای دیگر نامبرده شده‌اند که در ذیل به آنها اشاره می‌گردد. لازم به ذکر است که روابط مذکور حالتهایی از روابط ذکر شده در بالا هستند:

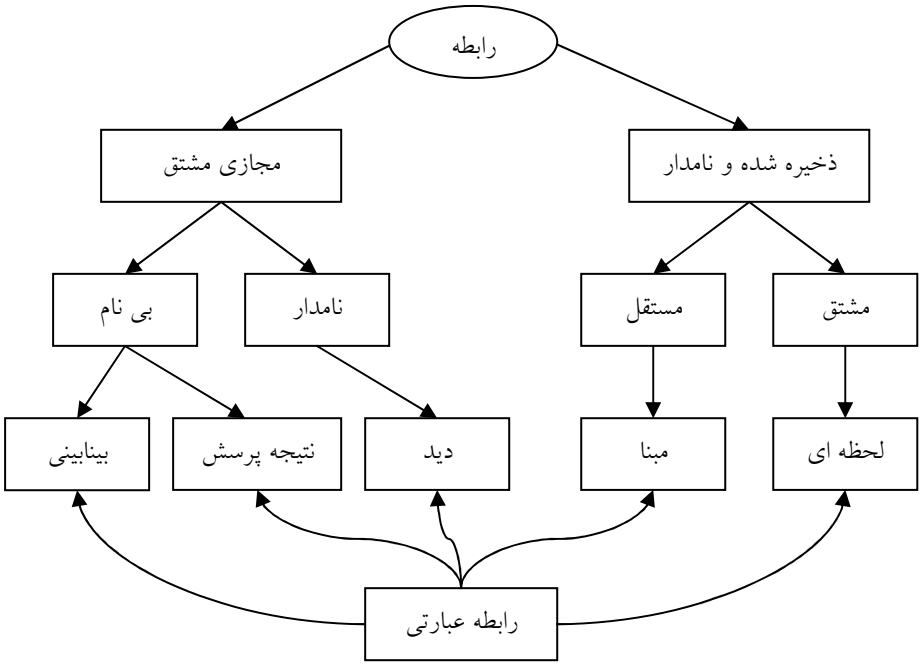
- رابطه لحظه‌ای[6]
- رابطه عبارتی[7]
- رابطه نتیجه پرسش[8]

---

1. Nested Expression
2. Named Relation
3. Base Relation
4. Derived Relation
5. Virtual Relation
6. Snapshot
7. Expression Relation
8. Query Result Relation

- رابطه بینابینی[1]
- رابطه ذخیره شده [2]

شکل زیر ارتباط و دسته بندی انواع رابطه را نشان می دهد:



شکل ٦-١١ نمایش نحوه ارتباط بین رابطه‌ها

۷- قواعد جامعیت[3] در مدل رابطه‌ای

در هر محیط عملیاتی همیشه مجموعه‌ای از قواعد معنایی[4] یا محدودیت‌های جامعیتی[5] وجود دارند. اگر داده‌های ذخیره شده در پایگاه‌داده همواره این

1. Intermediate Relation
2. Stored Relation
3. Integrity Rules
4. Semantic Rules
5. Integrity Constraints

محدودیت‌ها را رعایت کنند، جامعیت پایگاه‌داده تامین می‌باشد.

جامعیت پایگاه‌داده‌ها بمعنی صحت، دقت و سازگاری داده‌های ذخیره شده در پایگاه‌داده‌ها در تمام لحظات است. هر سیستم پایگاه‌داده باید بتواند جامعیت پایگاه‌داده را کنترل و تضمین کند. زیرا ممکن است عواملی مانند آنچه که در ذیل نام برده شده‌اند، باعث نقض جامعیت شوند:

- اشتباه در برنامه‌ها
- اشتباه در ورود اطلاعات از سوی کاربران
- مشکلات سخت‌افزاری و نرم‌افزاری مرتبط با داده‌ها
- عدم انجام کامل فرایندها بر روی داده‌ها
- وجود افزونگی و تعدد در داده‌های تکراری

برای کنترل و تضمین جامعیت، قواعدی لازم است تا سیستم مدیریت بتواند بر اساس آنها عمل کرده و باعث انطباق محتوای پایگاه‌داده با واقعیات باشد. این قواعدرا قواعد جامعیتی و یا محدودیت‌های جامعیتی گویند.

۷-۱ انواع قواعد جامعیت

قواعد جامعیت در مدل رابطه‌ای به دو رده کلی تقسیم می‌شوند که به شرح هریک خواهیم پرداخت:

- قواعد کاربری [1]
- متا قواعد [2]

۷-۱-۱ قواعد کاربری

قواعد کاربری که گاه به آنها قواعد محیطی یا قواعد وابسته به داده نیز می‌گویند، قواعدی هستند که توسط کاربر مجاز و برای یک پایگاه‌داده خاص تعریف می‌شوند. این قواعد وابسته به واقعیات محیط هستند و در بعضی کتب به آنها محدودیت‌های

1. User Defined Rules
2. Meta Rules

جامعیتی معنایی[1] نیز می‌گویند. قواعد کاربری در مدل رابطه‌ای به چهار دسته به شرح زیر تقسیم می‌شوند:

- **محدودیت میدانی**[2]: محدودیتی است ناظر بر میدان هر صفت خاصه و مقادیر مجاز آنرا مشخص می‌کند.
- **محدودیت صفتی**[3]: محدودیتی است ناظر بر یک صفت و نوع آن را مشخص می‌کند.
- **محدودیت رابطه‌ای**[4]: محدودیتی است ناظر بر یک رابطه و مقادیر مجاز یک متغیر رابطه‌ای را مشخص می‌کند.
- **محدودیت پایگاهی**[5]: محدودیتی است ناظر بر دو یا چند متغیر رابطه‌ای به نحوی که آنها را بهم مرتبط می‌کند.

۷-۱-۲ متا قواعد

متا قواعد قواعدی هستند که باید توسط هر سیستم رابطه در هر پایگاه‌داده رابطه‌ای اعمال گردند. لذا در بعضی کتب بدانها قواعد عام نیز می‌گویند. این قواعد به دو گروه کلی به شرح ذیل تقسیم می‌شوند:

- قاعده جامعیت موجودیتی[6]
- قاعده جامعیت ارجاعی[7]

۷-۱-۲-۱ قاعده جامعیت موجودیتی

این قاعده ناظر بر کلید اصلی است و به شرح ذیل می‌باشد:

- هیچ جزء تشکیل دهنده کلید اصلی رابطه نمی‌تواند "مقدار هیچ" داشته باشد.

دلیل توجیه کننده این قاعده این است که هر مقدار یک کلید اصلی، در واقع شناسه

1. Semantic Integrity Constraint
2. Domain Constraint
3. Attribute Constraint
4. Relation Constraint
5. Database Constraint
6. Entity Integrity Rule
7. Referential Integrity Rule

آن تاپل در رابطه است و عامل تمییز نمونه‌های موجودیت (تاپل‌ها) در رابطه است و بدیهی است که عامل تمییز خود نمی‌تواند مقدار هیچ یا ناشناخته داشته باشد.

۷-۱-۲-۲ قاعده جامعیت ارجاعی

این قاعده ناظر بر کلید خارجی بوده و به شرح ذیل می‌باشد:

- اگر صفت خاصه Ai (ساده یا مرکب) در رابطه R2 کلید خارجی باشد در این صورت Ai در R2 می‌تواند مقدار هیچ داشته باشد (به شرطی که جزئی از کلید R2 نباشد)، در غیر اینصورت باید حتما مقداری باشد که در رابطه مرجع R1 وجود دارد. به عبارت دیگر مقدار کلید خارجی نمی‌تواند در رابطه مرجع وجود نداشته باشد.

دلیل توجیه این قاعده این است که کلید خارجی عامل ارجاع از یک نمونه موجودیت به نمونه موجودیت دیگر است و بدیهی است که نمی‌توان به نمونه موجودیت ناموجود ارجاع داد.

تمرینات

۱. عناصر اصلی ساختار جدولی را نام ببرید؟
۲. رابطه را تعریف کنید؟

۳. رابطه را از دیدگاه کاد تعریف کنید؟
٤. ویژگی‌های رابطه را نام ببرید؟
۵. انواع کلید را در مدل رابطه‌ای نام برده و توضیح دهید؟
٦. مفهوم کلید خارجی را توضیح دهید؟
۷. انواع رابطه را نام ببرید ؟
۸. قاعده جامعیت موجودیتی را توضیح دهید؟
۹. قاعده جامعیت ارجاعی را توضیح دهید؟

فصل ۷

عملیات در پایگاه رابطه‌ای

هدف کلی

در این فصل ابتدا تعریفی از پایگاه‌داده رابطه‌ای بیان شده و سپس مبحث جبر رابطه‌ای مورد بحث و بررسی قرار خواهد گرفت. در این راستا در ابتدا عملگرهای جبر رابطه‌ای معرفی شده و توانایی این عملگرها در الحاق و جداسازی اطلاعات و نمایش آنها شرح داده خواهد شد. سپس عملگرهای کار بر روی داده‌ها مطرح شده و انواع این عملگرها مورد بررسی قرار خواهند گرفت. در خاتمه نیز مروری مختصر بر روی مبحث بهینه سازی عبارات جستجو در پایگاه‌داده‌ها خواهیم داشت.

هدف رفتاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- تعریف پایگاه‌داده رابطه‌ای
- جبر رابطه‌ای
- عملگرهای جبر رابطه‌ای
- عملگر گزینش
- عملگر پرتو
- عملگر تغییر نام
- عملگر جایگزینی
- عملگرهای مجموعه

- عملگر ضرب دکارتی
- عملگر اجتماع
- عملگر اشتراک
- عملگر تفاضل
- عملگر تقسیم
- عملگر پیوند
- عملگرهای کار برروی داده‌ها
- عملگرهای درج، بهنگام سازی، حذف
- بهینه سازی پرس و جوها

۱- تعریف

پایگاه‌داده رابطه‌ای از دید کاربر در محیط انتزاعی مجموعه‌ای از رابطه‌های نرمال است. بدیهی است که جهت انجام عملیات در این محیط انتزاعی نیاز به یک مدل ریاضی است. پس کاربر تعریف شده در پایگاه‌داده نیاز به یک مدل ریاضی، که می‌تواند مجموعه‌ای از عملگرهای صوری باشد، دارد. این امکانات در واقع بخشی از مدل رابطه‌ای بوده و طبعا بدون وجود آنها، مدل رابطه‌ای کامل نخواهد بود. برای این منظور امکانات زیر وجود دارد:

- جبر رابطه‌ای[1]
- حساب رابطه‌ای
- محاسبات رابطه‌ای دامنه [2]
- محاسبات رابطه‌ای سطری [3]

لازم به ذکر است که در تئوری رابطه‌ها، هر سه تئوری فوق ارزش یکسانی دارند. در ادامه جبر رابطه‌ای مورد بررسی قرار می‌گیرد.

1. Relational algebra
2. Calculate Relational Domain
3. Calculate Relational Tuple

۲- جبر رابطه‌ای (RA) و عملگرهای آن

جبر رابطه‌ای ارائه شده توسط کاد (codd) شامل هشت عملگر در دو گروه چهار تایی می‌باشد. در بعضی کتب تعدادی عملگر دیگر نیز به این عملگرها اضافه شده‌اند. کلیه این عملگرها برای کار بر روی مفهوم رابطه‌ها تعریف و طراحی شده‌اند. در جبر رابطه‌ای داده‌ها همان رابطه‌ها می‌باشند. مجموعه عملگرهای جبر رابطه‌ای در عبارت زیر مشخص شده است، بدین ترتیب که:

$$RA = \{\{Relations\}, \{ Б, \prod, \ell, \longleftarrow, X, U, -, \cap, \div, \infty, X\theta, \theta\alpha \}\}$$

در بعضی از کتب تعدادی از این عملگرها تحت عنوان عملگر **set** و تعدادی دیگر تحت عنوان عملگر **join** معرفی شده‌اند که این دو گروه نیز توضیح داده خواهند شد. عملگرهای جبر رابطه‌ای در رابطه بالا به ترتیب عبارتند از:

- Б = عملگر گزینش[1]
- $\prod$= عملگر پرتو[2]
- ℓ = عملگر تغییر نام[3]
- $\longleftarrow$ = عملگر جایگزینی
- Set = عملگر مجموعه، این عملگرها بر روی مجموعه‌ای از رابطه‌ها عمل می‌کنند و شامل عملگرهای
- ضرب دکارتی[4] (X)
- اجتماع[5] (U)
- اشتراک[6] ($\cap$)
- تفاضل[7] (–)
- تقسیم (÷)

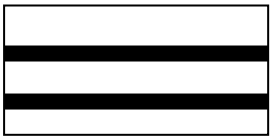
1. select
2. project
3. rename
4. Cartesian product
5. union
6. intersect
7. difference

- Join = عملگر پیوند (الحاقی) شامل عملگرهای
- پیوند طبیعی[1] (∞)
- نیم پیوند[2] ($\theta\alpha$)
- پیوند شرطی ($X\theta$)

در ادامه با ذکر مثال هایی به شرح هر یک از این عملگرها خواهیم پرداخت.

عملگرهای اصلی Select (Б)

انتخاب سطرهایی از یک رابطه بر اساس شرط مشخص (مشخصات مثلا یک فرد در یک سطر (افرادی که معدلشان مساوی x است)).



عملگر اصلی Project (Π)

انتخاب دیگر ستونهای مشخص از یک رابطه و حذف دیگر ستونها

نام		تلفن	

الحاق (Join)

دارای انواع مختلف بوده و رابطه‌ای را بر می‌گرداند که شامل تمام Tupleهای ترکیب شده از دو جدول می‌باشد. برای درک بهتر موضوع در ادامه مثالی از عملگرهای الحاقی ارائه شده است.

1. Natural Join
2. Semi Join

R1	R2
a1	b1
a2	b2
a3	b3

R1 ∞ R2		
a1	b1	c1
a2	b2	c2
a3	b3	c3

تمرین:

با استفاده از عملگر (÷) نتیجه R1 ÷ R2 = ? (R3) را بدست آورید:

R1

a	X
a	y
a	z
b	X
c	y

R2

x
y

R3

?

نکته: یکی از مهمترین خواص، خاصیت بسته بودن (بستار یا closure) می‌باشد. بدین معنا که خروجی هر عملگر رابطه‌ای خود یک رابطه است. بطور مثال می‌توان گفت که اجتماع دو رابطه (دو جدول) یک رابطه (یک جدول) می‌گردد.

عملگرتغییر نام (ρ)

این عملگر یک رابطه را گرفته و نسخه دیگری از آن را که ممکن است به بعضی از صفات اسامی‌دیگری داده شده باشد را می‌دهد:

S RENAME city AS Scity

تغییر نام صفت city به Scity در رابطه S

اگر بیشتر از یک تغییر انجام گیرد در پرانتز قرار می‌دهیم. بطور مثال:

S RENAME (city AS Scity , S# AS SNUM)

S#	SNAME	STATUS	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Black	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

با استفاده از دستور RENAME جدول فوق به جدول ذیل تبدیل می‌گردد.

SNUM	SNAME	STATUS	Scity
S1	Smith	20	London
S2	Jones	10	Paris
S3	Black	30	Paris
S4	Clark	20	London
S5	Adoms	30	Athens

مثال:

A

S#	SNAME	STATUS	City
S1	Smith	20	London
S4	Clark	20	London

B

S#	SNAME	STATUS	City
S1	Smith	20	London
S2	Jones	10	Paris

با توجه به دو جدول فوق داریم:

A ∪ B

S#	SNAME	STATUS	city
S1	Smith	20	London
S4	Clark	20	London
S2	Jones	10	Paris

A ∩ B

S#	SNAME	STATUS	City
S1	Smith	20	London

B-A

S#	SNAME	STATUS	City
S4	Clark	20	London

A-B

S#	SNAME	STATUS	city
S2	Jones	10	Paris

توجه:

$(A \cup B) \cup C \equiv A \cup (B \cup C) = A \cup B \cup C$
$(A \cap B) \cap C \equiv A \cap (B \cap C) = A \cap B \cap C$
$(A \times B) \times C \equiv A \times (B \times C) = A \times B \times C$

دستور (Б)

این دستور همان عملگر گزینش (Select) است و در واقع تعدادی از سطرها را انتخاب می‌کنیم. شکل کلی دستور به یکی از دو صورت ذیل می‌باشد:

(نام رابطه) Б (در کامپیوتر اجرا نمی‌شود).
شرط یا شرطها
شرط یا شرطها where نام رابطه (در کامپیوتر اجرا نمی‌شود (SQL)).

مثال:

Б (S) یا S where city = London

city = London

S#	SNAME	STATUS	CITY
S1	smith	20	London
S4	clark	20	london

مثال:

Б (S)

CITY = " London" AND SNAME = " smith"

S#	SNAME	STATUS	CITY
S1	smith	20	London

دستور (Π)

این دستور همان عملگر پرتو (Project) می‌باشد که در واقع ستون‌هایی از رابطه را در بر می‌گیرد. شکل کلی دستور به یکی از دو صورت ذیل می‌باشد:

۱- Π (نام رابطه)
نام صفت

۲- [نام صفتها] نام رابطه

مثال:

Π (s) یا s [city]

City

City
London
Paris
Athens

مثال:

Π (S) یا S [S# , SNAME]

S# و SNAME

S#	SNAME
S1	Smith
S2	Jones
S3	Black
S4	Clark
S5	Adams

تمرین: نام افرادیکه شهرشان لندن (london) است ؟

Б (s)
Π city = " London"
Name

تمرین: #S افرادیکه وضعیتشان برابر ۳۰ است.

Б (s)
Π STATUS = 30
S#

(S WHERE CITY = " London") S#
Name

عملگرهای پیوند (join)

ضرب دکارتی[1]

رابطه‌ای است که ستونهایش مجموع ستونهای دو رابطه و سطرهایش برابر با حاصلضرب (تمامی‌ترکیبهای ممکن) سطرهای آن دو رابطه (تکراری نیز خواهیم داشت) است.

خواص عملگر پیوند

- اطلاعات کامل

1 . Cartesian Product

- حجم یا فضای زیادی اشغال می‌کند (حتی الامکان انجام نمی‌دهیم).
- a ❋ b = b ❋ a

کاربردهای ضرب دکارتی(X)

یکی از کاربردهای ضرب دکارتی مواقعی است که بین جداول عامل ارتباط (صفتهای مشخص) نداشته باشیم (راه حل خوبی نیست زیرا فضای زیادی اشغال می‌کند).

a	b	c	d

*

x	y

بخاطر پر هزینه بودن ضرب، joinهای دیگر ارائه گردیدند (اصل join با ضرب شروع می‌شود).

الحاق طبیعی Natural join (∞)

نتیجه حاصل از این عملیات شامل جدولی است که ستونهایش، ستونهای دو جدولی (بدون تکرار) و سطرهایش، سطرهایی از جدول است که مقادیر همه ستونهای هم نامش مساوی هستند.

A

a	b
1	2
2	4

B

b	c
2	4
5	10
6	8
4	3
2	1

=

B ∞A
=
A join B

a	b	C
1	2	4
1	2	1
2	4	3

در مباحث پیوند رابطه‌ها نکات زیر باید مورد توجه قرار گیرد:

$$(A \infty B) \infty C = A \infty (B \infty C)$$
$$A \infty B = B \infty A$$

اگر دو جدول A و B دارای صفات مشترکی نباشند ∞ تبدیل به X می‌شود.

$$(X\ \theta\ (\theta - \text{join})\ \text{teta join})$$

و این عبارت معادل ضرب دکارتی است که شرطی نیز به آن افزوده شده است.

$$R1 * R2 \quad \text{یا} \quad (R1 * R2)\ \text{WHERE شرط}$$

کاربرد: مواقعی که نیاز به join دو رابطه بر اساس شرط خاصی به غیر از برابری آنها باشد.

مثال:

A

a	b
x	15
y	20

B

c	d
10	K
25	l

A * B
A.b > B.c

a	b	c	d
x	15	10	K
y	20	10	K

α = semi join

همان θ X است که فقط شامل ستونهای جدول اول می‌باشد (بدون ستونهای تکراری).

$$R1\ \alpha\ R2$$
شرط

کاربرد: در پایگاه‌های داده توزیع یافته (Distributed DB) بکار می‌رود تا از انتقال اطلاعات زیادی جلوگیری کند.

مثال:

$B\ \alpha\ A$

A	B

c.B < b.A

x	15
y	20

عملگر تقسیم[1]

فرض کنید

$A\ (X_1$ و X_2 و ... و X_n و Y_1 و Y_2 و ... و $Y_n)$

و

$B\ (Y_1$ و Y_2 و ... و $Y_n)$

در تقسیم دو رابطه، ستونهای باقی مانده و سطرها، سطرهایی است که شرط در همه آنها صادق است.

$A\ \text{DIVIDE By}\ B = A \div B = (X_1$ و X_2 و ... و $X_n)$

لازم به ذکر است که اگر Y2 را نداشته باشیم X2 را به ما نمی‌دهد، اگر Yn را نداشته باشیمXn را به ما نمی‌دهد.

ممکن است این نکته به ذهن برسد که اساساً این عملگر چه کاربردهایی می‌تواند داشته باشد. در پاسخ به این پرسش باید گفت که یکی از ساده ترین موارد استفاده از این عملگر پرس و جوهایی است که با استفاده از آنها دنبال مقادیری بگردیم که حاوی یک کلمه خاص باشد. (مثلا تمام Xهایی را بده که Yهایشان برابر □ است).

مثال:

S (S# و SNAME و STATUS و CITY) سازنده‌ها

P (P# و PNAME و WEIGHT و CITY) محصولات

SP (S# و P# و Qty) ترکیبی از محصولات و سازنده

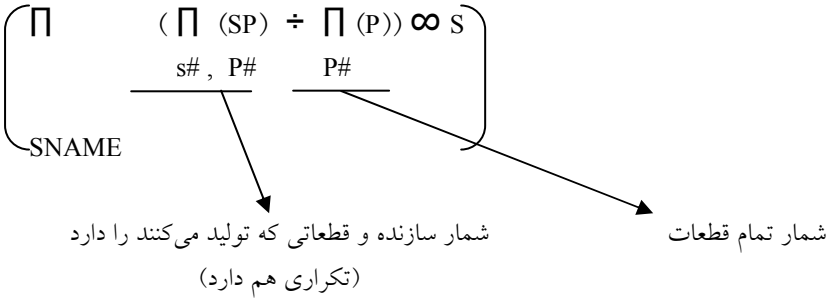
تعداد

سوال:

فروشندگانی که همه (تمام) قطعات را تولید می‌کنند یا نام تولید کنندگان همه

1. Divide - Division

قطعات؟



جواب: S# که همه قطعات را تولید کند.

از آنجائیکه در جستجوس نام قطعه هستیم، پس جواب را با جدول S، join می‌کنیم که در آخر نام آن تولید کنندگان را بدست آوریم.

سوال:

شماره فروشندگانی که حداقل تمام قطعات عرضه شده توسط S2 را عرضه می‌کنند؟

$$\Pi_{S\#}(s) \div \left(Б_{S\#='S2'}(sp) \right)$$

سوال:

جداول زیر را در نظر گرفته به پرسش‌ها پاسخ دهید.

Stud (S# و SNAME و city و avg و clg #)

prof (pname و affice و esp و degree و clg#)

crs (c#و cname و unit و clg#)

sec (sec#و c#و s# و term و pname و score)

clg (clg# و clgname و city و pnam)

الف) شماره دانشجویی، نام، کد دانشکده و معدل دانشجویی که میانگین نمرات آنها بالای ۱۵ می‌باشد.

ب) لیست نام اساتیدی که رئیس دانشکده نیستند.

ج) مشخصات کامل روسای دانشکده‌ها.

د) دانشجویانی که همه درسهای استاد x را گرفته‌اند.
ه) درسهایی که توسط همه دانشکده‌ها ارائه می‌شود.
و) مشخصات دروس ٤ واحدی که در نیمسال اول ۷۸ ارائه شده‌اند.

۳- عملگرهای کار برروی داده‌ها

هر کاربر در محیط پایگاه‌داده برای انجام عملیات نیازمند، عملگرهای زیر می‌باشد:

- عملگر درج[1]
- عملگر بهنگام سازی [2]
- عملگر حذف[3]

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

۳-۱ عملگر درج

این عملگر امکان اضافه کردن اطلاعات به جداول اطلاعاتی (رابطه‌ها) را فراهم می‌آورد. این عملگر در حالت کلی امکان افزودن داده‌ها به دوصورت زیر را دارد.

- افزودن مقادیر مشخص بعنوان یک سطر اطلاعاتی به جدول
- افزودن تعدادی سطر اطلاعاتی از جدولی به جدول دیگر

شکل کلی دستور به شرح ذیل می‌باشد:

INSERT source INTO Target

جدول مقصد

همانطور که مشاهده می‌گردد عبارت Source می‌تواند بصورت مجموعه‌ای مشخص از مقادیر که منطبق با جدول مقصد هستند، باشد و یا اینکه خروجه حاصل از یک

1. Insert
2. Update
3. Delete

جدول و یا خروجی حاصل از اعمال عملگرهای جبر رابطه‌ای بر روی یک یا چند جدول باشد.

مثال:

INSERT (S WHERE city = 'london') INTO TEMP

آن سطرهایی که city = 'London' است را در رابطه S، را به جدول temp اضافه می‌کند.

۳-۲ عملگر بهنگام سازی

این عملگر امکان اصلاًح اطلاعات ذخیره شده در یک جدول را برای کاربر فراهم می‌آورد. شکل کلی دستور به شرح ذیل می‌باشد:

UPDATE target Comma list-assignment
یک انتساب

مثال:

UPDATE P city = 'Paris' WHERE color = 'Red'

عبارت بالا بدین معنی است که از رابطه Pاندسته از سطرهایی را که مقدار صفت خاصه رنگ در آنها قرمز است، مقدار صفت خاصه شهر را برابر با پاریس کنیم. این عملگر را بصورت نمادین (نمایش) می‌توان به صورت زیر نشان داد.

(P)
WHERE color = 'Red'
City = 'Paris'

لازم بذکر است که عبارت بالا در بعضی از کتب بصورت کامل تر بیان می‌شود که در بحث فعلی مدنظر نیست.

۳-۳ عملگر حذف

این عملگر جهت حذف سطرهایی از اطلاعات درون جداول بکار می‌رود. شکل کلی دستور به شرح ذیل می‌باشد:

DELETE target R ← R –E

مثال:

DELETE S WHERE STATUS < 20

معنی دستور مذکور بدین صورت است: حذف کن از رابطه S آنهایی را که STATUS آن کمتر از ۲۰ است (پس یک سطر حذف می‌گردد).

توجه: رابطه R و E باید دارای شرط Same arity باشد (تعداد و نوع ستون‌هایشان یکسان است).

٤- کامل بودن جبر رابطه‌ای

همانطور که می‌دانید می‌توان عبارات جبر رابطه‌ای را با استفاده از ترکیب عملگرها نوشت و حاصل هر عبارت معتبر جبر رابطه‌ای باز هم یک رابطه است. لذا می‌توان چنین ارزیابی کرد که هر رابطه معتبر از مجموعه رابطه‌ها را می‌توان به کمک یک عبارت جبر رابطه‌ای نوشت. با این وصف می‌توان گفت که جبر رابطه‌ای از نظر رابطه‌ای کامل است یا به بیانی دیگر جبر رابطه‌ای کمال رابطه‌ای را دارد. بدین علت می‌توان جبر رابطه‌ای را معیار تشخیص کمال رابطه‌ای برای زبان‌های رابطه‌ای دانست. یعنی می‌توان گفت زبانی دارای کمال رابطه‌ای است که حداقل هر رابطه‌ای که با عبارت جبر رابطه‌ای قابل تعریف باشد، توسط آن زبان هم تعریف شدنی باشد.

٥- حساب رابطه‌ای

نوع دیگری از امکانات انجام عملیات در رابطه‌ها، حساب رابطه‌ای استکه منطقاً معادل جبر رابطه‌ای می‌باشد. یعنی برای هر عبارت جبر رابطه‌ای، یک عبارت معادل در حساب رابطه‌ای وجود دارد و بالعکس. دراینجا لازم است به این نکته توجه شود که جبر رابطه‌ای، دستوری بوده و شبیه به زبان برنامه نویسی است و حساب رابطه‌ای بصورت توصیفی بوده و به زبان طبیعی نزدیک‌تر است. حساب رابطه‌ای خود دارای دو شاخه می‌باشد:

- حساب تاپلی[1]
- حساب میدانی[2]

در ادامه به شرح هریک از این دو نوع حساب خواهیم پرداخت:

۵-۱ حساب رابطه‌ای تاپلی

در حساب رابطه‌ای تاپلی، یک مفهوم مهم به نام متغیر تاپلی وجود دارد. متغیر تاپلی متغیری است که تنها مقادیر مجازش، تاپل‌های رابطه هستند. این متغیر در شکل کلی به صورت زیر تعریف می‌شود

<range var definition>:: RANGEVAR <range var name> RANGEOVER
<Relational expression commalist>

برای درک بهتر موضوع به عبارات زیر توجه نمایید:

1-RANGEVAR STUD RANGE OVER STT;
2-RANGEVAR STUD RANGEOVER
(STUD WHERE STUD.STMJR = ‘’comp.eng)

۵-۱-۱ شکل کلی عبارت حساب تاپلی

اگر t یک متغیر تاپلی روی رابطه $R(A_1,A_2, \dots , A_N)$ باشد، در اینصورت شکل کلی عبارت حساب تاپلی بصورت زیر خواهد بود:

(target-items(s)) [WHERE f]

که در آن (target-items(s)) به مفهوم فهرستی از صفات متغیر تاپلی T است:

$T. A_1 , T. A_2 , \dots , T. A_N$

برای درک بهتر موضوع به مثال‌های زیر توجه نمایید:

ST.STID
ST.STID, ST.STDEID AS W
ST.STID WHERE ST.STDEID = ‘123’

1. Tuple Oriented
2. Domain Oriented

۵-۱-۲ سور وجودی و سور همگانی

در حساب رابطه تاپلی دو سور وجود دارد:

- سور وجودی[1]
- سور همگانی[2]

سور وجودی

سور وجودی را با ∃ نمایش داده و عبارت آنرا بصورت زیر می‌نویسیم

∃ T (f)

عبارت مذکور بدین معنا است که حداقل یک مقدار برای متغیر T وجود دارد به نحوی که f به مفهوم "درست" ارزیابی شود.

سور همگانی

سور همگانی را با ∀ نمایش داده و عبارت آنرا بصورت زیر می‌نویسیم

∀ T (f)

عبارت مذکور بدین معنا است که به ازاء تمام مقادیر متغیر T ، F به مفهوم "درست" ارزیابی می‌شود.

این دو سور به روش‌های زیر قابل تبدیل به یکدیگر می‌باشند:

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))
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)

۵-۱-۳ عبارت مطمئن

وقتی از سور وجودی، سور همگانی و سور نفی در نوشتن یک عبارت حساب

1. Existential
2. Universal

رابطه‌ای استفاده می‌کنیم، باید اطمینان حاصل کنیم که عبارت حسابی نوشته شده دارای معنا باشد. یک عبارت حسابی را عبارت مطمئن[1] گوییم هرگاه نتیجه ارزیابی آن، تعداد محدودی از تاپلها باشد. در غیر اینصورت آن عبارت را عبارت نامطمئن[2] می‌گوییم.

۵-۲ حساب رابطه میدانی

در این نوع حساب، بجای متغیر تاپلی، متغیر میدانی[3] داریم. متغیر میدانی متغیری است که از یک میدان مقدار می‌گیرد. تفاوت اصلی حساب میدانی با حساب تاپلی دراین است که در حساب میدانی، یک شرط اضافی به نام شرط عضویت[4] وجود دارد. شرط عضویت بصورت زیر نوشته می‌شود:

$$R\,(A_1{:}v_1\,,\,A_2{:}v_2,\,\ldots\,,\,A_n{:}v_n)$$

که در آن R نام رابطه ، Ai نام صفت و vi یک مقدار از میدان و یا یک لیترال است. این شرط به مفهوم "درست" ارزیابی می‌شود اگر و تنها اگر تاپلی در R وجود داشته باشد که مقادیر داده شده برای صفات را داشته باشد (تاپل عضوی از مجموعه بدنه رابطه باشد). برای مثال به عبارت زیرتوجه نمایید:

STT (STID: ‘345’,stdeg:’xwe’)

در عبارت فوق این شرط به مفهوم "درست" ارزیابی می‌شود اگر و فقط اگر تاپلی در STT با شماره دانشجویی ’345‘ و رشته کارشناسی ’xwe‘ وجود داشته باشد.

٦- بهینه سازی پرس و جوها[5]

یکی از مباحثی که در پایگاه‌های داده مطرح است بحث بهینه سازی عبارت پرس و

1. Safe expression
2. Unsafe expression
3. Domain variable
4. Membership condition
5. Optimization Query

جو می‌باشد. از آنجائیکه مبحث بهینه سازی پرس و جو مربوط به دوره کارشناسی نیست، لذا در اینجا صرفا توضیحات مختصری در اینباره ارائه خواهیم کرد.
تمامی‌تلاش طراحان پایگاه‌داده‌ها در کوتاه کردن مدت زمان ارسال پاسخ به متقاضی است. در عین حال خروجی یکسان در همه حال مد نظر می‌باشد. بعنوان مثال دو عبارت زیر را در نظر بگیرید:

۱- $\prod_{A,B}$ Б$_{A=B}$ (R)

۲- Б$_{A=B}$ $\prod_{A,B}$ (R)

۱- ابتدا از رابطه R آن‌هایی که A = B است را جدا می‌کند (سطری)، بعد از کل آنها ستون‌ها را بیرون می‌کشد.
۲- ابتدا از رابطه R دو ستون B و A را جدا می‌کند، بعد آنهایی را که A = B است را جدا می‌کند (سطرهایی که A = B).

در اینجا ممکن است سئوالاتی برای کاربر مطرح گردد. مانند:

- آیا همه پاسخ‌ها معادل اند ؟ بلی
- آیا همه پاسخ‌ها هم سرعت‌اند ؟ خیر

بنابرین جهت افزایش سرعت سیستم پایگاه‌داده سعی بر آن است بهترین روش بکار گرفته شود و یا به عبارت بهتر سعی می‌کنیم تا آنجا که می‌توانیم هر چه سریعتر جداول را کوچک کرده و روی آنها کار کنیم.

یکی از سئوالاتی که مطرح می‌گردد این است که سرعت Quaryهای متفاوت چقدر تفاوت دارد ؟ در پاسخ باید گفت که: بسیار زیاد، زیرا داده‌ها و جداول در حافظه جانبی قرار دارند. در این راستا ممکن است بگوییم که کاربر باید آنقدر تبهر داشته باشد تا بتواند عبارت پرسو جوی خود را بهینه نماید. ولی آیا واقعا چنین کاری ممکن است؟ در پاسخ باید گفت: خیر. زیرا:

- کاربر متخصص نیست.

- از سایر جداول و چگونگی آنها مطلع نمی‌باشد.
- اکثریت کاربران از ساختار و مکانیزم درونی پایگاه‌داده بی اطلاع هستند.

پس بهترین راه حل این است که بگوییم DBMS باید Quary را بهینه سازی نماید. طبیعی است که باید مکانیزم مشخصی برای اینکار وجود داشته باشد. هرچند که بحث بهینه سازی پرسو جو در این کتاب نمی‌گنجد، ولی برای ارائه یک شمای ساده از این عمل الگوریتم ذیل را (که بسیار ساده و مختصر در نظر گرفته شده است) ارائه می‌نماییم.

٦-١ تبدیل Q به o.Q

برای بهینه سازی پرس و جوها لازم است تا رویه مشخصی در نظر گرفته شود. در ادامه قواعد مربوط به بهینه سازی پرس و جوها مطرح شده است:

قاعده اول: Б را هر چه سریعتر (زودتر) انجام دهیم.

قاعده دوم: شرطهای ترکیبی را به شرطهای سوالی تبدیل نمایید (شرطهای دارای AND، OR و... را تا حد امکان بشکنید).

$$Б_{p1 \cap p2}(R) \longrightarrow Б_{p1}\left(Б_{p2}(R)\right)$$

قاعده سوم: Π را زودتر انجام دهیم (پس از Б).

٦-٢ سایر قواعد و عملگرهای بهینه سازی

قاعده اول:

$$a \infty b = b \infty a$$

قاعده دوم:

$$(a \infty b) \infty c = a \infty (b \infty c)$$

توجه: لازم بذکر است که عبارات این دو عبارات قاعده از لحاظ زمان با هم برابر نیستند.

مثال: مشخصات کامل دروس و گروه‌های درسی آنها را بدست آورید.

در اینجا این سئوال مطرح می‌شود که کدام یک از دو عبارت زیر سریعتر پاسخ می‌دهند؟

crs ∞ sec

یا

sec ∞ crs

این دو دستور از نظر سرعت با یکدیگر بسیار متفاوتند. زیرا سایز دو جدول کاملا متفاوت می‌باشد جدول crs (فقط مشخصات درس‌ها می‌باشد) معمولاً کوچک است و جدول sec بسیار بزرگتر از آن، زیرا هم گروههای مختلف دروس را شامل می‌شود و هم دانشجویان آنها را. فرض کنید جدول crs در حافظه cash جا بگیرد، در این صورت الگوریتم‌های دو راه حل بالا را بررسی می‌کنیم:

crs ∞ sec

الگوریتم (۱): برای هر سطر جدول crs **{**

برای هر سطر جدول sec **{**

مقایسه کن**}**

انتخاب کن **}**

در الگوریتم یک باید سطرهای بسیار زیاد جدول sec را به دفعات وارد حافظه اصلی کنیم و مقایسه و انتخاب را انجام دهیم به عبارت دیگر تعداد دستیابی به دیسک (منظور حافظه جانبی) به‌اندازه حاصلضرب سایز دو جدول است.

sec ∞ crs

الگوریتم (۲): برای هر سطر جدول sec **{**

برای هر سطر جدول crs **{**

مقایسه کن **}**

انتخاب کن **}**

در الگوریتم دو، هر سطر sec را فقط یکبار به حافظه اصلی می‌آوریم زیرا جدول crs به طور کامل در حافظه cash قرار دارد و همه مقایسه‌ها یکبار انجام می‌گیرد و تعداد دستیابی به دیسک به‌اندازه سایز جدول sec است. بطور مثال اگر سایز جدول crs را

۱۰۰ و جدول sec را ۱۰۰۱ فرض کنیم الگوریتم اول ۱۰۰۰۰۰۰ بار بیشتر به دیسک دستیابی پیدا می‌کند.

تمرینات

۱. چه امکاناتی برای انجام عملیات در مدل رابطه‌ای وجود دارند ؟

۲. عملگرهای جبر رابطه‌ای را نام ببرید ؟
۳. عملگرهای Select و Project چه تفاوتی با یکدیگر دارند ؟
٤. عملگر پیوند را توضیح داده و خواص آنرا نام ببرید؟
۵. عملگرهای کار بر روی داده‌ها را نام ببرید؟

فصل ۸

آشنایی با زبان رابطه‌ای SQL

هدف کلی

در این فصل ابتدا تاریخچه‌ای از زبان SQL و دلائل گستردگی آن ارائه خواهد شد. سپس دستورات تعریف داده‌ها مانند تعریف شما و جدول و... مطرح شده و در ادامه دستورات مربوط به پرس‌وجوی داده‌ها بیان خواهد شد. پس از توضیح این گروه از دستورات، دستورات کار با داده‌ها شرح داده خواهند شد.پس از آن جستجوهای پیشرفته‌تر مطرح و در این راستا عملگرهای پیوند و سایر عملگرهای مرتبط با جستجوهای پیشرفته بیان خواهند شد. در پایان نیز دربازه دستورات کنترل مجوزهای دسترسی صحبت خواهد شد.

هدف رفتاری

دراین فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- مقدمه ای بر SQL
- دلائل گستردگی SQL
- دستورات تعریف داده‌ها DDL
- تعریف پایگاه‌داده
- تعریف شما
- تعریف جدول
- دستورات پرس و جو (مشاهده) داده‌ها

- دستورات کار بر روی داده‌ها
- جستجوهای پیشرفته با عملگر پیوند
- عملگر گروه بندی
- عملگر مرتب سازی
- توابع تجمعی
- استفاده از پرس و جوهای تودرتو
- استفاده از زیرپرسش‌ها (پرسش‌های فرعی)
- دستورات کنترل مجوز دسترسی

۱ – مقدمه‌ای بر SQL

یک زبان رابطه‌ای قبل از هر چیز باید بتواند در محیط انتزاعی رابطه عمل نماید و طبیعتا باید بر اساس ریاضیات رابطه‌ای مانند جبر رابطه‌ای استوار باشد. در این فصل زبان SQL[1] بعنوان یک زبان رابطه‌ای استاندارد[2] شده و شاید بعنوان بهترین زبان رابطه‌ای مورد بحث و بررسی قرار می‌گیرد.

اولین نسخه این زبان بعنوان قسمتی از سیستم R[3] در اواخر دهه ۱۹۸۰ توسط شرکت IBM طراحی شد. از آنجائیکه پس از مدتی شرکت‌ها و اشخاص مختلف هر کدام اقدام به‌ایجاد یک نسخه از زبان SQL با اسامی‌مختلف کردند، لذا در سال ۱۹۸٤ موسسه استانداردهای ملی آمریکا (ANSI) و سازمان استانداردهای بین المللی (ISO) نسخه استاندارد زبان SQL را ارائه کردند و در سال ۱۹۸۷ اولین نسخه استاندارد این زبان بنام ANSI/ISO SQL-89 را که اصطلاحا به آن SQL1 می‌گفتند، ارائه کردند. پس از گذشت چند سال و با توجه به توسعه توانایی‌ها و انتظارات از این زبان، نسخه SQL-92 به نام SQL2 توسط این موسسه‌ها ارائه گردید. با ورود مفاهیم شیء گرایی به محیط پایگاه‌های داده رابطه‌ای، زبان SQL-99 که به SQL3 معروف است، توسط موسسه‌های مذکور ارائه گردید. امروزه نسخه جدیدی از زبان استاندارد SQL بنام SQL-2003 نیز

1. Standard Query Language
2. Standard Relational Language
3. R System

ارائه شده است که هنوز در پایگاه‌های داده رابطه‌ای مورد استفاده قرار نگرفته است. مفاهیم اصلی زبان SQL در کلیه زبان‌های مذکور یکسان هستند.

۲ - دلایل گستردگی SQL

شرکت‌های فعال در زمینه پایگاه‌های داده رابطه‌ای با توجه به نیاز خود اقدام به طراحی زبانهای رابطه‌ای کرده‌اند که اصولا همگی از استانداردهای ANSI/ISO پشتیبانی می‌کنند. این زبان‌ها مفاهیم زبان SQL را بصورت کامل پشتیبانی کرده و علاوه بر آن دستورات دیگری نیز برای ساختار خود دارند. امروزه اکثریت پایگاه‌های داده رابطه‌ای زبان SQL2 را بعنوان استاندارد زبان پایگاه‌داده خود انتخاب کرده‌اند. با توجه به اینکه پیاده سازی زبان SQL3 بعنوان یک زبان رابطه‌ای شیء گرا دارای پیچیدگی‌های خاص خود می‌باشد، لذا امروزه از بین پایگاه‌های داده رابطه‌ای تنها پایگاه‌داده اراکل[1] محصول شرکت اراکل، قادر به پشتیبانی از زبان SQL3 می‌باشد. بصورت کلی می‌توان اهم دلایل گستردگی استفاده از زبان SQL را به شرح زیر عنوان کرد:

- مورد قبول همه است (همه متخصصین از آن استفاده می‌کنند و آنرا قبول می‌کنند).
- شبه رویه‌ای است (زبان پرس و جوئی است).
- به صورت تعبیه شده (توکار[2]) در دیگر محیط‌های برنامه سازی است. مانند محیط Access، Delphi، Oracle و....

برخی امکانات مهمتر زبان SQL به شرح ذیل می‌باشند که بعضی از آنها در زبان SQL1 وجود نداشته و به مرور به زبان‌های SQL2 و SQL3 افزوده شده‌اند:

- دستورات تعریف داده‌ها
- دستورات پرس و جو (مشاهده) داده‌ها
- دستورات کار بر روی داده‌ها

1. ORACLE
2. Embedded

- دستورات کنترل مجوز دسترسی
- دستورات نوشتن رویه‌ها و توابع
- دستورات کنترل جامعیت
- دستورات کنترل تراکنش‌ها

با توجه به اینکه چهار دسته اول دستورات ذکر شده تقریبا در همه پایگاه‌های داده موجود می‌باشد لذا در ادامه چهار سری دستورات اول را به تفصیل مورد بحث و بررسی قرار خواهیم داد. بعضی از پایگاه‌های داده و یا زبان‌های برنامه نویسی (که زبان SQL را درون خود دارند)، مجموعه دستورات نوشتن رویه‌ها و توابع، کنترل جامعیت و کنترل تراکنش‌ها را یا ندارند و یا در اختیار کاربر قرار نمی‌دهند. لذا این نوع دستورات در این کتاب توضیح داده نمی‌شوند.

۳ – دستورات تعریف داده‌ها DDL[1]

این دستورات در حالت کلی شامل تعریف پایگاه‌داده، تعریف شما[2]، تعریف کاربر[3] و تعریف جداول[4] اطلاعاتی و دیده‌ها می‌باشند.

۳ –۱ تعریف پایگاه‌داده

شکل کلی دستور به شرح ذیل می‌باشد:

```
CREATE DATABASE database -name
```

با استفاده از این دستور پایگاه‌داده مورد نظر را ایجاد می‌نمائیم که خود پایگاه‌داده می‌تواند شامل جداول متعددی باشد. لازم به ذکر است که تعریف پایگاه‌داده اساساً دارای پارامترهای بسیاری است که موضوع این کتاب نیست.

1. Data Definition Language
2. Schema
3. User Definition
4. Table Definition

۳ -۲ تعریف شما

این تعریف در حالت کلی شامل تعریف کاربر، تعریف جداول و تعریف دیدها و بسیاری مفاهیم دیگر می‌باشد و ساختار آن بسته به پایگاه‌داده انتخاب شده متفاوت می‌باشد. شکل کلی دستور (که در آن صرفا تعاریف Table و View آمده است) به شرح ذیل می‌باشد.

```
Schema::= CREATE SCHEMA
    AUTORIZATION   User
    { Schema -element -list }
Schema -element::= base -table definition
     | View definition
     | grant -operation
```

جهت حذف یک شما از عبارت زیر استفاده می‌شود:

```
DROP SCHEMA Name [ Restrict | Cascade ]
```

۳ -۳ جدول

۳ -۳ -۱ تعریف جدول

با این دستور امکان تعریف جداول اطلاعاتی برای کاربر فراهم می‌شود. شکل کلی دستور به صورت زیر می‌باشد:

```
Base -table definition::= CREATE TABLE base -table
(base -table -element commalist)
Base -table -element::= column -definition
| Unique constraint -definition
Column -definition::= column data -type [ NOT NULL [ UNIQUE ] ]
Unique Constraint -definition::= UNIQUE (Column -Cammalist)
```

برای مثال فرض کنید می‌خواهیم یک جدول شامل شماره و نام قاره‌های مختلف جهان درست کنیم. عبارت ایجاد کننده جدول مذکور به شرح زیر می‌باشد:

```
CREATE TABLE Continent
(CntId int ,
CntName char(30))
```

حال چنانچه بخواهیم عبارت مذکور را کامل تر کنیم لازم است محدودیت‌هایی را به آن اضافه نماییم.

```
CREATE TABLE Continent
(CntId int Unique,
CntName char (30) NOT NULL)
```

لازم به ذکر است که در SQL -2 کلمات کلیدی دیگری به عبارت ایجاد جدول افزوده شده‌اند بگونه‌ایکه امکان افزودن مقدار اولیه به یک صفت خاصه و یا امکان افزودن محدودیت‌ها به جدول در هنگام ایجاد جدول فراهم شده است. برای مثال با استفاده از کلمه کلیدی Default در جلوی یک صفت خاصه می‌توان مقدار پیش فرض را برای آن ستون تعیین نمود. در عبارت زیر مقدار پیش فرض ستون سال برابر با ۱۳۸٤ در نظر گرفته شده است:

```
CREATE TABLE Info
(InfoId int unique,
InfoYear int default 1384,
InfoName char (50) NOT NULL)
```

تمرین:

سیستم پایگاه‌داده دانشگاه را به صورت زیر در نظر بگیرید، طراحی به زبان SQL آنرا کامل نمائید:

```
Stud (S#, Sname, S -add, S -tel)
Prof (P#, Pname, P -add, P -off)
Crs (C#, Cname, unit)
Enroll(S#, C#, Sec#, term, score)
Sec (C#, Sec#, term, time, place, P#)
```

۳ -۳ -۲ اصلاح ساختار جدول

یکی از ملزومات اساسی در مورد جداول، امکان اصلاًح ساختار جداول و حذف و اضافه کردن ستونهای مربوط به آن می‌باشد. بدیهی است که در مواقعی لازم می‌شود تا ستونهای جدیدی (صفات خاصه) به ساختار جدولی که از قبل طراحی کرده‌ایم اضافه گردد یا ستون‌هایی از جدول مذکور حذف گردد. همچنین این امکان نیز

وجود دارد که در مواردی تشخیص دهیم که نوع داده مربوط به یک ستون باید تغییر یابد. برای این منظور از دستور ALTER TABLE استفاده می‌شود که ساختار کلی این دستور در ذیل آمده است:

```
ALTER TABLE tablename
{ ADD [COLUMN] column definition }
| { ALTER [COLUMN] column name
{ SET DEFAULT default option } | { DROP DEFAULT }}
| { DROP [COLUMN] column name }
| { ADD table constraint definition }
| { DROP CONSTRAINT constraint name };
```

همانطور که در بالا مشاهده می‌شود لیست تغییرات قابل اعمال بر روی یک جدول عبارتند از:

- افزودن یک ستون به جدول
- حذف یک ستون از جدول
- افزودن یک محدودیت به جدول
- حذف یک محدودیت از جدول
- افزودن مقدار اولیه به یک ستون
- حذف مقدار اولیه یک ستون

ممکن است در اینجا این سئوال به ذهن برسد که چگونه می‌توان نوع داده‌ای یک ستون را تغییر داد. در اینجا لازم است به این نکته اشاره شود که تغییر نوع داده‌ای یک ستون اساساً در بسیاری از پایگاه‌های داده در نظر گرفته نشده است و این مهم بواسطه نوع قرار گرفتن مقادیر مربوط به ستون‌ها در فضای فیزیکی حافظه می‌باشد بگونه‌ایکه حتی در بعضی پایگاه‌های داده امکان تغییر ترتیب ستون‌ها در یک جدول وجود ندارد. امکان تغییر نوع داده مربوط به یک ستون فقط در بعضی از پایگاه‌های داده و بصورت منطقی طراحی شده است. حال در ادامه به منظور آشنایی بهتر با این نوع دستور چند مثال مختلف را نمایش می‌دهیم. فرض کنید می‌خواهیم ستونی به نام fname و از نوع رشته‌ای را به جدولی به نام SalesPeople اضافه نماییم. عبارت SQL مربوطه به شرح ذیل خواهد بود:

```
ALTER TABLE SalesPeople ADD fname char(10);
```

همانطور که مشاهده می‌نمایید کلمه کلیدی COLUMN بعد از کلمه ADD نوشته نشده است. در بعضی از پایگاه‌های داده استفاده از کلمه کلیدی COLUMN در این شرایط اجباری نمی‌باشد، حال آنکه در بعضی دیگر اجباری است. حال به مثال دیگری توجه نمایید که در آن مقدار پیش فرض برای یک ستون پایگاه‌داده تعریف می‌شود.

```
ALTER TABLE Salespeople
ALTER COLUMN city
ADD DEFAULT 'London';
```

۳ -۳ -۳ تغییر نوع داده‌های یک ستون جدول

همانطور که در بالا اشاره شد تغییر نوع داده یک ستون از جدول در هر پایگاه‌داده‌ای امکان پذیر نیست. با توجه به اینکه این دستور در بعضی پایگاه‌های داده فراهم شده است لذا شکل دستور مذکور در ذیل آمده است.

```
ALTER  TABLE  tablename
MODIFY (< col -name > < new type >)
```

مثال: این دستور تعداد کاراکترهای صفت درس را به 40 تغییر می‌دهد.

```
ALTER TABLE crs
MODIFY (Cname char(40));
```

توجه: تغییر نوع داده در اکثر SQLها غیر مجاز است و در بعضی از پایگاه‌های داده قبل از تایید عمل مذکور اقدام به بررسی محتوای داده‌های ستون مورد نظر می‌گردد و در صورت قابل قبول بودن عمل تبدیل نوع این درخواست تائید می‌گردد. بعنوان مثال تغییر نوع داده‌ای از کاراکتر به عدد فقط درصورتیکه تمامی‌مقادیر یک ستون از نوع عدد باشند امکان پذیر است و چنانچه حتی یک مقدار شامل حروف باشد، انجام این کار غیر ممکن است.

۳ -۳ -٤ حذف یک جدول

در مواردی ممکن است لازم باشد یک جدول از پایگاه‌داده حذف گردد. برای انجام

اینکار دستور زیر موجود می‌باشد. لازم به ذکر است که در هنگام حذف یک جدول باید بررسی گردد که چنانچه جداول دیگری به این جدول مرتبط باشند، چه عملی باید برروی جداول مرتبط انجام گیرد و آیا اساساً امکان حذف جدول وجود دارد یا خیر. شکل کلی دستور مذکور به شرح زیر می‌باشد:

```
DROP TABLE tablename [ RESTRICT | CASCADE ];
```

٤- دستورات پرس و جو (مشاهده) داده‌ها

یکی از مهمترین خواسته‌های کاربران هر پایگاه‌داده، امکان بازیابی داده‌های ثبت شده در جداول می‌باشد. به این منظور دستوراتی ارائه شده‌اند که امکان بازیابی سطرها یا ستون‌هایی از یک جدول و یا چند جدول اطلاعاتی را فراهم می‌آورند که به آنها اصطلاحا زبان پرس و جوی داده‌ها[1] می‌گویند. مهمترین دستور از این نوع دستورات دستور SELECT می‌باشد. این دستور یکی از پیچیده ترین دستورات در زبان SQL می‌باشد و خود به تنهایی شامل دستورات متنوعی می‌باشد. شکل کلی دستور به شرح ذیل می‌باشد:

```
SELECT [DISTINCT]
{ value expression
[ AS column name ] } , … }
| { qualifier.*}
| *
FROM { { table name [ AS ] [ correlation name ] } , …
[ WHERE predicate ]
[ GROUP BY [ table name | correlation name ].column name
[ COLLATE collation name ] ]
[ HAVING predicate ]
[ { UNION | INTERSECT | EXCEPT } [ ALL ]
Select statement | { TABLE table name}
| table value constructor ]
[ ORDER BY { { out put column | positive integer } [ ASC | DESC] , … } ] ;
```

همانطور که در عبارت بالا مشاهده می‌گردد این دستور بسیار پیچیده و کلی می‌باشد و اساساً دارای زیر دستوراتی می‌باشد که در بحث این کتاب نمی‌گنجد. برای

1. Data Query Language (DQL)

راحتی بیشتر خوانندگان، سعی می‌کنیم در ابتدا مثالهای ساده تر در مورد این دستور را مورد بررسی قرار دهیم و در ادامه همین فصل در خصوص حالات پیچیده تر این دستور و طریقه بازیابی داده‌ها از چندین جدول اطلاعاتی را شرح خواهیم داد. حالت ساده شده دستور مذکور بصورت زیر قابل نمایش است که به آن اصطلاحا SFW نیز می‌گویند که حروف اول هر قسمت می‌باشد.

```
SELECT column name , …
FROM table name
WHERE criteria
```

که در آن column name نام ستون‌ها، table name نام جدول و criteria شرط مورد نظر برای انتخاب و بازیابی رکوردها می‌باشد.

در حالت بسیار ساده می‌توان فرض کرد که می‌خواهیم رکوردهای یک جدول (مثلا جدول اطلاعات پرسنلی EMP) را بازیابی نماییم. فرض کنیم جدول EMP دارای ساختار زیر باشد و بغیر از صفت خاصه EMPNO که کلید اصلی می‌باشد سایر مقادیر می‌توانند مقدار NULL را بپذیرند:

Name	نام صفت خاصه
EMPNO	شماره پرسنلی
ENAME	نام پرسنل
JOB	شغل
MGR	شماره مدیر
HIREDATE	تاریخ استخدام

در عبارت ذیل می‌خواهیم کلیه رکوردهای جدول را بازیابی نماییم. عبارت مورد نظر به شرح ذیل می‌باشد:

```
Select *
From EMP;
```

همانطور که در عبارت بالا مشاهده می‌شود علامت * به معنای انتخابی تمامی‌ستون‌های مربوط به سطرها می‌باشد. این دستور می‌تواند به شرح ذیل نیز باشد:

```
Select EMP.*
From EMP ;
```

همانطور که مشاهده می‌گردد در این دستور قبل از استفاده از علامت *، نام جدول نیز نوشته شده است که در این مورد بعدا توضیح خواهیم داد. حال در ادامه فرض کنیم می‌خواهیم فقط ستون‌های شماره پرسنلی و نام پرسنل را برای تمامی‌پرسنل بازیابی نماییم. عبارت مذکور به شرح ذیل می‌باشد:

```
Select empno , ename
From EMP ;
```

می‌توانستیم قبل از اسامی‌ستونها، نام جدول را نیز بکار ببریم که از این کار اجتناب کردیم. حال فرض کنیم می‌خواهیم بدانیم که اگر حقوق هر یک از پرسنل را ۱۰٪ افزایش دهیم چه میزانی حقوق برای هر یک از پرسنل خواهیم داشت. عبارت مذکور به شرح زیر خواهد بود:

```
Select empno,ename,sall * 0.1
From emp;
```

چنانچه بخواهیم عبارت را واضح تر بیان کنیم لازم است که برای عبارت sall * 0.1 یک نام مستعار مانند NEWSALL در نظر بگیریم که عبارت مذکور به شرح زیر خواهد بود.

```
Select empno,ename,sall * 0.1 as newsall
From emp;
```

حال برای مشخص شدن عبارت WHERE (بیان کننده شرط) مثال دیگری را مطرح می‌کنیم. فرض کنیم می‌خواهیم اطلاعات کسانی را مشاهده کنیم که میزان حقوق آنها از ۱۰۰۰۰ تومان بیشتر باشد. به عبارت زیر توجه نمایید.

```
SELECT  *
FROM  emp
WHERE sall > 10000 ;
```

در این عبارت جلوی کلمه WHERE عبارت شرط مطرح شده است. چنانچه صرفا بعضی از صفات خاصه مطرح باشند می‌توان بجای استفاده از علامت *، اسامی‌ستونها (صفات خاصه) را نوشت.

یکی از نکات بسایر مهم این است که لازم است ترتیب سه کلمه SELECT و FROM و WHERE رعایت شود. چنانچه ترتیب مذکور رعابت نگردد سیستم با پیغام خطا مواجهه می‌گردد. همچنین می‌توان در عبارات، کلمه WHERE را در صورت نبودن شرط بکار نبرد، ولی حتما باید کلمات کلیدی SELECT و FROM در عبارت باشند.

تذکر: در بعضی شرایط امکان دارد که که کاربر بخواهد مقداری را در سیستم محاسبه نماید که اصلاً ارتباطی با هیچ جدولی ندارد. از آنجاییکه لازم است حتما بعد از کلمه FROM نام جدول ذکر گردد، لذا در بعضی از پایگاه‌های داده یک جدول فرضی تعریف شده است. مثلا در پایگاه‌داده ORACLE نام جدول فرضی DUAL می‌باشد. به عبارت ذیل توجه نمایید:

```
SELECT  50 * 147
FROM DUAL ;
```

تا این مرحله از دستور SELECT بصورت بسیار ساده مورد بررسی قرار گرفت. قسمت‌های پیشرفته تر این دستور در ادامه همین فصل مورد بررسی قرار خواهد گرفت. در ادامه دستورات کار بر روی داده‌ها را مورد بررسی قرار خواهیم داد.

۵ – دستورات کار بر روی داده‌ها

یکی از بدیهی ترین نیازها برای کاربران، وجود امکان کار بر روی داده‌ها می‌باشد که متناسب با این نیاز دستوراتی تحت عنوان زبان کار با داده‌ها [1] طراحی شده است. این عملیات می‌تواند شامل موارد زیر باشد:

- INSERT –افزودن رکورد (هایی) به یک جدول
- UPDATE –اصلاًح اطلاعات موجود در یک جدول
- DELETE –حذف رکورد (هایی) از جدول

هر یک از این عملیات دارای دستورات خاص خود می‌باشند که در ادامه به شرح

1. Data Manipulation Language (DML)

هر یک خواهیم پرداخت.

۵-۱ دستور INSERT

این دستور به منظور وارد کردن (افزودن) رکورد یا رکوردهایی به یک جدول تعریف شده است. شکل کلی دستور به شرح ذیل می‌باشد:

```
INSERT INTO table name
[ (column name , …) ]
{ VALUES (value , …) }
| sub query ;
```

همانطور که در دستور بالا مشاهده می‌گردد، در انتهای دستور از عبارت sub query استفاده شده است که این دستور به منظور انتخاب چندین رکورد از یک جئول و افزودن یکباره آنها به جدول مذکور می‌باشد. در ابتدا برای آشنایی خوانندگان با این دستور، افزودن یک رکورد به جدول را مورد بررسی قرار می‌دهیم. در دستور بالا در ابتدا نام جدول را که می‌خواهیم مقادیری را به آن بیافزاییم وارد می‌کنیم و سپس اسامی‌ستونهایی از رکورد را که می‌خواهیم مقادیری را در آنها وارد نماییم را نوشته و سپس مقادیر مرتبط با هر ستون را وارد می‌نماییم. برای مثال فرض کنیم می‌خواهیم در جدولی بنام SalesPeople که دارای چهار ستون می‌باشد، یک رکورد با مقادیر مشخصی ذیل وارد نماییم. عبارت مذکور به این شرح می‌باشد:

```
INSERT INTO Salespeople
VALUES (1001,'Peel','London', 12);
```

حال فرض کنیم مثلا نمی‌دانیم که شخص مذکور در شهر لندن اقامت دارد و می‌خواهیم رکورد مذکور را وارد نماییم بدون آنکه کقداری برای صفت خاصه شهر در نظر بگیریم. عبارت مذکور به شرح زیر می‌باشد:

```
Insert Into salespeople
Values (1001,'Peel', NULL, 12);
```

توجه: برای آندسته از ستونهایی که بعنوان کلید اصلی هستند و یا اینکه حتما باید مقداری (غیر از NULL) را داشته باشند، لازم است مقداری در نظر گرفته شود، در غیر اینصورت امکان افزودن رکورد جدید وجود ندارد.

لازم به ذکر است که ترتیب وارد کردن مقادیر به ترتیب تعریف ستونها فرض شده است. چنانچه بخواهیم ترتیب ورود مقادیر برای ستونها را تغییر دهیم و یا اینکه صرفا بخواهیم بجای قرار دادن مقدار NULL برای یک ستون، اصلاً نام آن ستون را بکار نبریم، لازم است نام ستونها به ترتیب مورد نظر در جلوی نام جدول وارد شده و سپس به همان ترتیب، مقادیر مرتبط وارد شوند. برای توضیح بیشتر فرض کنیم جدولی بنام Customers داریم که فقط می‌خواهیم در ستون‌های city و cname و cnum آن جدول مقادیری را وارد نماییم. به عبارت ذیل توجه نمایید:

```
INSERT INTO Customers (city, cname, cnum)
                VALUES ('London','Hoffman', 2001);
```

تا این مرحله صرفا یک رکورد به جدول وارد شده است. چنانچه بخواهیم تعدادی رکورد را از جدول دیگری انتخاب و به جدول مذکور وارد نماییم، بجای استفاده از کلمه Values، لازم است تا یک عبارت Select را وارد نماییم. برای روشن شدن این مطلب به عبارت زیر توجه نمایید:

```
INSERT INTO LondonStaff
SELECT *
FROM salespeople
Where city = 'London';
```

حال فرض کنید می‌خواهیم صرفا دو ستون بنام‌های FDate و Total از جدولی بنام DayTotals را با مقادیر انتخاب شده از جدول دیگری پر کنیم. مثال زیر بیانگر این مطلب می‌باشد:

```
Insert Into daytotals (fdate,total)
            Select odate,income
From orders ;
```

نکته بسیار مهم در نوشتن این عبارت این است که تعداد ستونهای انتخاب شده در عبارت Select باید برابر با تعداد ستون‌های جدول اصلی باشد.

۲-۵ دستور UPDATE

یکی از مهمترین نیازهای کاربران، توانایی اعمال تغییرات در داده‌هایی است که قبلا در سیستم وارد کرده‌اند. این نیاز ناشی از تغییر داده‌ها و یا وادر کردن مقدار برای ستونهایی است که قبلا هیچ مقداری برای آن در نظر گرفته نشده بود. دستور UPDATE به منظور اعمال تغییرات در مقادیر رکوردهای ثبت شده در جداول طراحی شده و شکل کلی آن به شرح ذیل می‌باشد:

```
UPDATE table name
SET { column name = { value expression
            | NULL
    | DEFAULT } } , …
[ { WHERE predicate } ] ;
```

در ادامه برای تشریح بهتر دستور مذکور مثالهایی آورده شده است که حالات مختلف استفاده از دستور را نشان می‌دهد. در ساده ترین حالت فرض کنید می‌خواهیم مقدار صفت خاصه rating از جدول Customers را برابر با ۲۰۰ کنیم.

```
UPDATE customers
        SET rating = 200;
```

همانطور که در بالا مشاهده می‌کنید هیچ نوع شرطی برای انتخاب رکوردها در نظر گرفته نشده است. لذا با اجرا دستور بالا مقدار صفت خاصه rating برای تمامی‌رکوردها برابر با ۲۰۰ می‌شود. در مثال زیر یک شرط را جهت بروز رسانی جدول اضافه می‌کنیم:

```
UPDATE customers
        SET rating = 200
        WHERE snum = 1001;
```

تا این مرحله صرفا یک صفت خاصه از جدول را بروز رسانی کرده‌ایم. در ادامه طریقه بروز رسانی چند صفت خاصه بصورت همزمان را مشاهده خواهیم کرد.

```
UPDATE Salespeople
SET sname = 'Gibson', city = 'Boston', comm. = 0.1
WHERE snum = 1004;
```

همانطور که مشاهده می‌کنید هر یک از صفات خاصه با یک کاما از دیگری جدا شده است. تا این قسمت از کار معمولاً مقدار یک صفت خاصه را تغییر داده‌ایم. ولی

این امکان نیز وجود دارد که مقدار جدید، عبارتی شامل صفات خاصه نیز باشد. برای مثال ممکن است بخواهیم مقدار یک صفت خاصه را دو برابر کنیم.

```
Update salespeople
    SET comm = comm * 2;
```

همانطور که مشاهده می‌شود مقدار جدید در واقع از پردازش بر روی مقدار قبلی همان صفت خاصه بدست آمده است. این امکان وجود دارد که از مقادیر سایر صفات خاصه نیز استفاده شود. حال فرض کنید می‌خواهیم مقدار یک صفت خاصه را تبدیل به NULL (مقدار هیچ) نماییم. عبارت زیر طریقه انجام این کار را نشان می‌دهد.

```
UPDATE customers
    SET rating = NULL
    WHERE city = 'London';
```

عمل تبدیل یک مقدار به NULL در واقع حذف مقدار یک ستون از رکورد می‌باشد که در اینجا به آن بروز رسانی می‌گوییم. مفهوم حذف یک مقدار از یک ستون با مفهوم حذف یک رکورد (سطر) متفاوت می‌باشد. در ادامه دستور DELETE را جهت حذف رکوردها توضیح خواهیم داد.

۵-۳ دستور DELETE

ممکن است در مواقعی کاربران بخواهند رکورد (هایی) را از جدول اطلاعاتی حذف نمایند. در واقع کل اطلاعات یک رکورد (و نه فقط بعضی ستون‌ها) حذف خواهد شد. برای این منظور دستور DELETE طراحی شده است. شمای کلی این دستور در زیر آمده است:

```
DELETE FROM table name
        { WHERE predicate } ;
```

برای درک بهتر دستور، مثالهایی در ذیل آمده است. در ابتدا لازم بذکر است چنانچه در هنگام استفاده از دستور DELETE، شرطی آورده نشود بدین معنا است که کلیه اطلاعات موجود در آن جدول حذف شود. برای مثال دستور ذیل بیانگر حذف کلیه رکوردهای موجود در جدول Customers می‌باشد.

```
DELETE FROM customers;
```

حال در مثالی دیگر یک عبارت شرطی جهت تعیین رکوردهای مورد نظر جهت حذف نشان داده شده است.

```
DELETE FROM salespeople
WHERE snum = 1003;
```

لازم به ذکر است که عبارت شرط می‌تواند یک یا تعدادی از رکوردها را در برگیرد. با این وصف دستور DELETE توانایی حذف یک یا چند رکورد یا تمامی‌رکوردهای یک جدول را دارد.

٦- جستجوهای پیشرفته با عملگر پیوند[1]

تا بحال کلیه مثال‌های ارائه شده در این فصل بر اساس یک جدول طراحی شده بودند. یکی از انتظارات بسیار مهم کاربران یک پایگاه‌داده، امکان بازیابی اطلاعات از جداول مرتبط با هم می‌باشد. به این منظور عملگر پیوند بین جداول در زبان SQL در نظر گرفته شده است. این عملگر در واقع جزئی از دستور SELECT می‌باشد. در واقع عملگر پیوند باعث ایجاد ارتباط بین جداول بر اساس کلید اصلی و کلید خارجی بین آنها می‌باشد. البته در مواردی نیز ممکن است که ارتباط بین جداول بر اساس یک عبارت خاص یا یک ترکیب شرطی باشد. در حالت کلی عملگر پیوند به دو صورت می‌تواند باشد:

- عملگر پیوند داخلی[2]
- عملگر پیوند خارجی[3]

این عملگرها هریک دارای انواع مختلفی می‌باشد که در ادامه به شرح هر یک خواهیم پرداخت:

---

1. Join
2. Inner Join
3. Outer Join

۶-۱ عملگر پیوند داخلی

این عملگر در واقع متداول ترین نوع پیوند بین دو جدول می‌باشد که بر اساس آن تمامی‌رکوردهایی از دو جدول که در شرط تعریف شده در پیوند صدق می‌کنند، مورد بررسی قرار می‌گیرند. به بیانی دیگر اگر جدول (رابطه) سمت چپ در پیوند را R1 و جدول سمت راست در پیوند را R2 بنامیم، صرفا تمامی رکوردهایی از دو جدول که شرط تعریف شده در پیوند در مورد آنها صادق باشد، مورد بازیابی قرار خواهند گرفت.

```
R1 JOIN R2 ON criteria
OR
R1 INNER JOIN R2 ON criteria
```

استفاده از عبارت Inner Join به این علت است که در بعضی از زبان‌ها این نوع پیوند را پیوند داخلی می‌نامند. برای درک بهتر دو جدول اطلاعات پرسنل (EMP) و ساختار سازمانی (DEPT) را به صورت آنچه که در زیر آمده است در نظر می‌گیریم:

جدول ۸ -۱ جدول اطلاعات پرسنل EMP

EMP			
EMPNO	**ENAME**	**SAL**	**DEPTNO**
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7566	JONES	2975	20
7654	MARTIN	1250	30
7698	BLAKE	2850	30
7782	CLARK	2450	10
7788	SCOTT	3000	20
7839	KING	5000	10
7844	TURNER	1500	30
7876	ADAMS	1100	20
7900	JAMES	950	30

7902	FORD	3000	20
7934	MILLER	1300	10

جدول ۸ -۲ جدول اطلاعات ساختار سازمانی DEPT

DEPT		
DEPTNO	**DNAME**	**LOC**
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

حال فرض کنیم می‌خواهیم دو جدول مذکور را بر اساس صفت خاصه Deptno به یکدیگر پیوند داده و ببینیم هر یک از پرسنل در کدام واحد مشغول بکار هستند. عبارت مذکور به شرح ذیل می‌باشد:

```
Select Emp.ename, Emp.deptno, Dept.dname
From Emp  Join  Dept
On emp.deptno = dept.deptno
```

همانطور که ملاحظه می‌گردد دو جدول Emp و Dept بر اساس صفت خاصه Deptno که در جدول Dept بعنوان کلید اصلی و در جدول Emp بعنوان کلید خارجی می‌باشد، به یکدیگر پیوند داده شده‌اند. با این وصف خروجی حاصل از عبارت مذکور به صورت زیر خواهد بود:

جدول ۸ -۳ خروجی حاصل از عبارت SQL

ENAME	DEPTNO	DNAME
SMITH	20	RESEARCH

ALLEN	30	SALES
WARD	30	SALES
JONES	20	RESEARCH
MARTIN	30	SALES
BLAKE	30	SALES
CLARK	10	ACCOUNTING
SCOTT	20	RESEARCH
KING	10	ACCOUNTING
TURNER	30	SALES
ADAMS	20	RESEARCH
JAMES	30	SALES
FORD	20	RESEARCH
MILLER	10	ACCOUNTING

توجه داشته باشید که می‌توان عبارت بالا را بدون استفاده از عملگر پیوند نیز نوشت. نحوه نوشتن عبارت مذکور بدون عملگر پیوند به شرح ذیل می‌باشد:

```
Select Emp.ename, Emp.deptno, Dept.dname
From Emp , Dept
Where emp.deptno = dept.deptno
```

همانطور که مشاهده می‌کنید می‌توان اسامی‌جداول مورد نظر را در جلوی کلمه FROM بکار برد. با این وصف می‌توان بدون نیاز به استفاده از عملگر پیوند اقدام به برقراری ارتباط بین دو یا چند جدول نمود. ولی بهتر است از عملگر پیوند استفاده گردد. در هر حال هنگامیکه می‌خواهیم از بیش از یک جدول استفاده نماییم، لازم است نام جداول را قبل از نام ستون‌ها (صفات خاصه) مورد نظر بکار ببریم. این وضعیت بیشتر زمانی اهمیت پیدا می‌کند که صفات خاصه همنام در جداول مورد پیوند داشته باشیم.

حال فرض کنید می‌خواهیم بجای اسامی‌جداول از نام‌های مستعار استفاده نماییم. مثال زیر بیانگر نحوه پیاده سازی این وضعیت می‌باشد.

```
Select E.ename, E.deptno, D.dname
From Emp E  Join  Dept  D
```

```
On E.deptno = D.deptno
```

این وضعیت بیشتر زمانی نمود پیدا می‌کند که بخواهیم چند جدول با اسامی‌مختلف و یا نسبتا طولانی را به یکدیگر پیوند بزنیم و یا اینکه در مواردی بخواهیم یک جدول را با خودش پیوند بزنیم.

عملگر پیوند طبیعی[1]

حال می‌خواهیم عبارت مذکور را بر اساس عملگر Natural join بنویسیم. لازم به ذکر است که نوع پیوند Natural Join در بعضی از پایگاه‌های داده طراحی شده است و ممکن است در بعضی از پایگاه‌های داده این نوع پیوند پشتیبانی نشود. نوع Natural Join بر این اساس عمل می‌کند که کلید ارتباطی را بصورت اتوماتیک از دیکشنری پایگاه‌داده می‌خواند. به بیانی دیگر در این نوع از پیوند نیازی به تعریف شرط پیوند نیست. مثال زیر بیانگر این وضعیت می‌باشد:

```
Select Emp.ename, Emp.deptno, Dept.dname
From Emp Natural Join Dept
```

٦ -٣ عملگرهای پیوند خارجی[2]

در بسیاری از موارد ممکن است بعضی مقادیر ستونهایی که می‌خواهیم به یکدیگر پیوند بزنیم، دارای مقدار NULL باشند. در چنین شرایطی چنانچه از عملگر پیوند بصورت عادی استفاده نماییم، تعدادی از رکوردها را نمی‌توانیم بازیابی نماییم. حال چنانچه بخواهیم بگوییم که در پیوند مذکور تمامی رکوردهای اطلاعاتی مورد نظر از یک جدول نمایش داده شود و از جدول دوم‌اندسته از رکوردهایی که در شرط بین دو جدول صدق می‌کنند نمایش داده شوند، لازم است تا از پیوند خارجی استفاده گردد. پیوند خارجی به صورت‌های زیر قابل پیاده سازی می‌باشد:

- پیوند خارجی چپ[3]
- پیوند خارجی راست[1]

1. Natural Join
2. Outer Join
3. Left Outer Join

- پیوند خارجی کامل[2]

در ادامه هر یک از پیوندهای مذکور مورد بررسی قرار خواهند گرفت.

٦ -٣ -١ پیوند خارجی چپ

هر گاه بخواهیم در خروجی حاصل از ارتباط بین دو جدول، تمامی‌رکوردهای مورد نظر از جدول سمت چپ (جدول اول) را انتخاب کرده و از جدول سمت راست صرفااندسته از رکوردهایی که شرط ارتباط بین دو جدول را دارند نمایش بدهیم، از پیوند خارجی چپ استفاده می‌کنیم. بدیهی است در چنین شرایطی برای بعضی از رکوردهای جدول سمت چپ، هیچ مقداری از جدول سمت راست نمی‌توان یافت. با این اوصاف برای‌اندسته از ستونهای انتخاب شده از جدول سمت راست که مقداری ندارند، مقدار NULL در نظر گرفته خواهد شد.

برای درک بهتر موضوع همان دو جدول EMP و DEPT که کمی‌قبل تر معرفی شدند را مورد بررسی قرار می‌دهیم. بدین منظور با استفاده از عبارت زیر مقدار Deptno یکی از رکوردهای جدول EMP را برابر با NULL قرار می‌دهیم.

```
Update emp
set deptno = NULL
where ename = 'KING';
```

حال به عبارت زیر که در آن جدول DEPT بعنوان جدول سمت راست پیوند و جدول EMP بعنوان جدول سمت چپ پیوند می‌باشند توجه نمایید:

```
Select e.ename , e.deptno , d.dname
 LEFT JOIN   DEPT d   From   EMP e
ON e.deptno = d.deptno
```

و یا

```
Select e.ename , e.deptno , d.dname
EMP e   LEFTOUTER JOIN   DEPT d   From
ON e.deptno = d.deptno
```

---

1. Right Outer Join
2. Full Outer Join

دو عبارت مذکور از لحاظ معنایی با یکدیگر برابر هستند و خروجی حاصل از عبارت بالا به شرح جدول ۸-٤ خواهد بود. همانطور که مشاهده می‌کنید با توجه به اینکه مقدار deptno از جدول EMP برابر با NULL شده است، لذا هیچ رکورد متناظری با جدول DEPT وجود نداشته و مقدار جایگزین برای ستونهای مربوطه برابر با NULL می‌باشد.

ممکن است این سئوال مطرح گردد که چرا عبارت مذکور به دو شکل مختلف نوشته شده است. در پاسخ به این سئوال باید گفت که در زبان SQL ارائه شده از سوی ANSI از کلمه Left Outer Join استفاده شده است. ولی در بعضی از پایگاه‌های داده این عبارت به صورت Left Join نوشته شده است. حتی در بعضی از پایگاه‌های داده از علامت " + " بجای این عبارت استفاده شده است.

جدول ۸ -٤ خروجی حاصل از عبارت SQL پیوند خارجی چپ

ENAME	DEPTNO	DNAME
SMITH	20	RESEARCH
ALLEN	30	SALES
WARD	30	SALES
JONES	20	RESEARCH
MARTIN	30	SALES
BLAKE	30	SALES
CLARK	10	ACCOUNTING
SCOTT	20	RESEARCH
KING		
TURNER	30	SALES
ADAMS	20	RESEARCH
JAMES	30	SALES
FORD	20	RESEARCH
MILLER	10	ACCOUNTING

٦ -٣ -٢ پیوند خارجی راست

هر گاه بخواهیم در خروجی حاصل از ارتباط بین دو جدول، تمامی‌رکوردهای مورد نظر از جدول سمت راست (جدول دوم) را انتخاب کرده و از جدول سمت چپ صرفااندسته از رکوردهایی که شرط ارتباط بین دو جدول را دارند نمایش بدهیم، از پیوند خارجی راست استفاده می‌کنیم. بدیهی است در چنین شرایطی برای بعضی از رکوردهای جدول سمت راست پیوند، هیچ مقداری از جدول سمت چپ نمی‌توان یافت. با این اوصاف برای‌اندسته از ستونهای انتخاب شده از جدول سمت چپ پیوند که مقداری ندارند، مقدار NULL در نظر گرفته خواهد شد.

برای درک بهتر موضوع همان دو جدول EMP و DEPT که کمی‌قبل تر معرفی شدند را مورد بررسی قرار می‌دهیم. همانطور که می‌دانید در جدول DEPT یک رکورد با deptno = 40 وجود دارد. ولی هیچ رکوردی در جدول EMP وجود ندارد که دمقدار صفت خاصه deptno در آن برابر با ٤٠ باشد. حال به عبارت زیر که در آن جدول DEPT بعنوان جدول سمت چپ پیوند و جدول EMP بعنوان جدول سمت راست پیوند می‌باشند توجه نمایید:

```
Select e.ename , e.deptno , d.dname
 RIGHT JOIN    DEPT d    From    EMP e
ON e.deptno = d.deptno
```

و یا

```
Select e.ename , e.deptno , d.dname
EMP e    RIGHT OUTER JOIN    DEPT d   From
ON e.deptno = d.deptno
```

خروجی حاصل از عبارت بالا به شرح ذیل خواهد بود:

جدول ۸ –۵ خروجی حاصل از عبارت SQL پیوند خارجی راست

ENAME	DEPTNO	DNAME
CLARK	10	ACCOUNTING
MILLER	10	ACCOUNTING
SMITH	20	RESEARCH
SCOTT	20	RESEARCH
ADAMS	20	RESEARCH
FORD	20	RESEARCH

JONES	20	RESEARCH
ALLEN	30	SALES
JAMES	30	SALES
TURNER	30	SALES
BLAKE	30	SALES
MARTIN	30	SALES
WARD	30	SALES
		OPERATIONS

همانطور که مشاهده می‌کنید با توجه به اینکه مقدار deptno از جدول EMP برابر با NULL شده است، لذا هیچ رکورد متناظری با جدول DEPT وجود نداشته و مقدار جایگزین برای ستونهای مربوطه برابر با NULL می‌باشد.

٦ -٣ -٣ پیوند خارجی کامل

قبل از توضیح این نوع از پیوند لازم است به این نکته مهم توجه شود که این نوع پیوند فقط در بعضی از پایگاه‌های داده طراحی شده است و در بسیاری از پایگاه‌های داده اصلاً چنین پیوند وجود ندارد.

هر گاه بخواهیم در خروجی حاصل از ارتباط بین دو جدول، تمامی‌رکوردهای مورد نظر از هر دو جدول چه آنهاییکه در شرط پیوند هستند و چه آنهاییکه در شرط پیوند صادق نیستند، در خروجی حاصل بیایند، از این نوع پیوند استفاده می‌گردد. بدیهی است که هر یک از مقادیر ستونهایی که در شرط پیوند صادق نباشند، با مقدار NULL جایگزین خواهند شد. برای درک بهتر موضوع همان دو جدول EMP و DEPT که کمی‌قبل تر معرفی شدند را مورد بررسی قرار می‌دهیم. بدین منظور با استفاده از عبارت زیر مقدار Deptno یکی از رکوردهای جدول EMP را برابر با NULL قرار می‌دهیم.

```
Update emp
set deptno = NULL
where ename = ‘KING’;
```

حال به عبارت زیر که در آن جدول DEPT بعنوان جدول سمت راست پیوند و جدول EMP بعنوان جدول سمت چپ پیوند می‌باشند توجه نمایید:

```
Select e.ename , e.deptno , d.dname
EMP e    FULL OUTER JOIN    DEPT d   From
ON e.deptno = d.deptno
```

خروجی حاصل از عبارت بالا به شرح ذیل خواهد بود. همانطور که مشاهده می‌کنید با توجه به اینکه مقدار deptno از جدول EMP برابر با NULL شده است، لذا دو رکورد در خروجی حاصل دارای مقادیر NULL می‌باشند.

جدول ۸ –٦ خروجی حاصل از عبارت SQL پیوند خارجی کامل

ENAME	DEPTNO	DNAME
SMITH	20	RESEARCH
ALLEN	30	SALES
WARD	30	SALES
JONES	20	RESEARCH
MARTIN	30	SALES
BLAKE	30	SALES
CLARK	10	ACCOUNTING
SCOTT	20	RESEARCH
TURNER	30	SALES
ADAMS	20	RESEARCH
JAMES	30	SALES
FORD	20	RESEARCH
MILLER	10	ACCOUNTING
		OPERATIONS
KING		

۷ – سایر عملگرها در جستجوهای پیشرفته

۷ –۱ عملگر گروه بندی [1]

1. Group By

فرض کنید می‌خواهیم رکوردهای (سطرها) یک جدول را بر اساس مقادیر یک یا چند ستون آن جدول گروه بندی نماییم بگونه‌ایکه رکوردهای موجود در هر گروه دارای یک مقدار مشترک از ستونهای مورد نظر در گروه بندی باشند. برای انجام این امر از عملگر GroupBy استفاده می‌شود. برای مثال فرض کنید می‌خواهیم رکوردهای جدول EMP را بر اساس مقادیر صفت خاصه (ستون) deptno گروه بندی نماییم. عبارت مورد نظر به شرح ذیل خواهد بود:

```
Select *
From EMP
Group By deptno
```

۷ -۲ عملگر مرتب سازی

همانطور که مشاهده می‌گردد عملگر Group By اقدام به دسته بندی رکوردها بر اساس مقادیر یک (یا چند) ستون می‌نماید. ولی این عملگر منطقاً الزام به مرتب سازی مقادیر بصورت صعودی یا نزولی نمی‌کند و صرفا گروه بندی را انجام می‌دهد. حال اگر لازم باشد بگونه‌ای اقدام به مرتب سازی خروجی بنماییم لازم است از عملگر ORDERBY استفاده گردد. این عملگر توانایی مرتب سازی رکوردها به صورت صعودی و نزولی دارد. برای انتخاب نوع مرتب سازی یکی از دو عملگر زیر مورد استفاده قرار می‌گیرند:

ASC : رکوردها را بصورت صعودی مرتب سازی می‌کند

DESC : رکوردها را بصورت نزولی مرتب سازی می‌کند

به عبارت زیر که رکوردهای جدول EMP را بر اساس صفت خاصه ename و بصورت صعودی مرتب می‌کند توجه نمایید:

```
Select *
From EMP
ORDER BY ename ASC
```

حال فرض کنید می‌خواهیم رکوردهای جدول مذکور را بر اساس دو صفت خاصه مرتب سازی نماییم. عبارت زیر بیانگر نحوه انجام این موضوع می‌باشد:

```
Select *
```

```
From EMP
ORDER BY deptno DESC , ename ASC
```

همانطور که مشاهده می‌نمایید در عبارت فوق ابتدا رکوردها بر اساس مقدار صفت خاصه deptno بصورت نزولی مرتب شده و سپس بر اساس صفت خاصه ename بصورت صعودی مرتب می‌گردند.

۷ -۳ توابع تجمعی

یکی از نیازهای معمول کاربران انجام محاسباتی بر روی بعضی ستونها می‌باشد. بعنوان مثال فرض کنید می‌خواهید میانگین نمره دانش آموزان در یک کلاس برای یک درس را بدست آورید. بدیهی است که این عمل بصورت ستونی انجام می‌گیرد و با سایر دستوراتی که تا بحال گفته‌ایم متفاوت می‌باشد. توابعی که بر روی مقادیر ستونها عمل می‌کنند را توابع تجمعی می‌نامیم. لیستی از معروفترین این توابع به شرح ذیل می‌باشند:

COUNT : تعداد مقادیر در یک ستون را ارائه می‌دهد.
SUM : حاصل جمع مقادیر یک ستون را ارائه می‌دهد.
AVG : میانگین مقادیر یک ستون را ارائه می‌دهد.
MAX : بالاترین مقدار در بین مقادیر یک ستون را ارائه می‌دهد.
MIN : پایین ترین مقدار در بین مقادیر یک ستون را ارائه می‌دهد.

نکته‌ای که باید بدان توجه کرد این است که خروجی حاصل از توابع مذکور بصورت یک مقدار می‌باشند. لازم به ذکر است که هر پایگاه‌داده‌ای علاوه بر توابع مذکور، تعدادی تابع دیگر نیز طراحی کرده است. برای مثال در بعضی از پایگاه‌های داده توابعی جهت محاسبه واریانس و یا انحراف معیار طراحی شده‌اند. برای درک بهتر موضوع، جدول EMP را در نظر بگیرید. فرض کنید می‌خواهیم تعداد پرسنلی که مقدار صفت خاصه deptno در آنها برابر با ۱۰ باشد را، بدست آوریم. عبارت مذکور به شرح ذیل خواهد بود:

```
SELECT Count (*)  as Tedad
FROM EMP
WHERE deptno = 10 ;
```

خروجی حاصل از عبارت مذکور به شرح ذیل خواهد بود:

TEDAD
3

حال فرض کنید می‌خواهیم بالاترین و پایین ترین میزان حقوق پرسنلی که مقدار صفت خاصه deptno در آنها برابر با ۱۰ باشد را، بدست آوریم. عبارت مذکور به شرح ذیل خواهد بود:

```
SELECT Max(sal) as maxval , Min(sal) as minval
From EMP
Where deptno =10 ;
```

خروجی حاصل از عبارت مذکور به شرح ذیل خواهد بود:

MAXVAL	MINVAL
5000	1300

حال فرض کنید می‌خواهیم حداقل میانگین حقوق را برای پرسنل تعریف شده در جدول EMP بر حسب صفت خاصه deptno بدست آوریم. در این شرایط لازم است علاوه بر استفاده از توابع تجمعی از Group By نیز استفاده نمود. به عبارت زیر که بیانگر این درخواست می‌باشد توجه کنید:

```
Select deptno , Min(sal) as MinVal
From emp
Group by deptno
```

خروجی حاصل از عبارت مذکور به شرح ذیل خواهد بود:

DEPTNO	MINVAL
10	1300
20	800
30	950

۷ -٤ عملگر HAVING

همانطور که می‌دانید تابحال از عملگر WHERE برای تعیین شرایط استفاده می‌کردیم. ولی عملگر Where فقط قادر بود بر روی ستونها عمل نماید، حال آنکه گاهی لازم است عملگر تعیین شرایط برروی سطرها عمل نماید. این وضعیت بیشتر زمانی رخ می‌دهد که بخواهیم از Group By استفاده نماییم و یا اینکه بخواهیم از توابع تجمعی که اساساً بر روی سطرها عمل می‌کنند، استفاده نماییم. در چنین شرایطی بجای استفاده از کلمه Where از کلمه Having استفاده می‌کنیم. برای درک بهتر موضوع به مثال زیر توجه نمایید:

```
Select deptno
From emp
Group By deptno
Having COUNT (*) > 4
```

خروجی حاصل از عبارت بالا به شرح ذیل خواهد بود:

DEPTNO
20
30

همانطور که مشاهده می‌نمایید از آنجائیکه تعداد رکوردهایی که در آنها مقدار deptno برابر ۱۰ باشد، کمتر از ٤ تا هستند، لذا رکوردی که بیانگر مقدار ۱۰ در ستون deptno باشد در خروجی مشاهده نمی‌گردد. یکی از نکات مهم در زبان SQL این است که عملگرهای Having و Where می‌توانند بصورت ترکیبی در یک عبارت SQL مورد استفاده قرار گیرند.

۷ -۵ عملگر BETWEEN

یکی از ویژگی‌های مورد نیاز کاربران، امکان انتخاب رکوردها بر اساس مقادیری است که در یک بازه قرار داشته باشند. برای مثال فرض کنید لیست پرسنلی را بخواهیم که حقوق آنها بین ۱۰۰۰ و 2000 باشد. عبارت زیر نحوه استفاده از دستور Between را نشان می‌دهد:

```
Select  ename ,Sal
From EMP
Where sal  Between 1000  and  2000 ;
```

خروجی حاصل از عبارت مذکور به شرح زیر خواهد بود:

ENAME	SAL
ALLEN	1600
WARD	1250
MARTIN	1250
TURNER	1500
ADAMS	1100
MILLER	1300

۷ -٦ عملگر LIKE

در بعضی موارد کاربران نیاز دارند تا عبارتی را در بین مقادیر یک صفت خاصه جستجو نمایند. این وضعیت بیشتر زمانی رخ می‌دهد که کاربران در حال کار با مقادیر رشته‌ای باشند. در این شرایط از عملگر Like استفاده می‌نماییم.

برای مثال فرض کنید می‌خواهیم در جدول EMP به دنبال افرادی بگردیم که در صفت خاصه ename آنها کلمه 'AM' باشد. عبارت زیر نحوه نوشتن چنین درخواستی را نشان می‌دهد:

```
Select ename ,deptno, sal
From emp
' ; % AM % Where ename LIKE '
```

همانطور که مشاهده می‌کنید علامت " ٪ " نشان می‌دهد که کلمه " AM " می‌تواند در بین مقادیر موجود باشد و لازم نیست حتما در ابتدا یا انتهای آن مقادیر باشد. خروجی حاصل از عبارت بالا به شرح زیر خواهد بود.

ENAME	DEPTNO	SAL
ADAMS	20	1100
JAMES	30	950

۸ – استفاده از پرس و جوهای تودرتو[1]

در بسیاری از موارد مقادیر مورد نظر در عبارت where خود بصورت مشخص نبوده و بر اساس یک عبارت پرس و جوس دیگر بدست می ایند. در چنین شرایطی لازم است تا بجای استفاده از عبارات جداگانه، از عبارات تودرتو استفاده کرد تا خروجی هر یک از عبارات درونی بعنوان مقادیر مورد استفاده در عبارت بیرونی تر در نظر گرفته شود. در ادامه انواعی از عبارات تودرتو را نشان خواهیم داد.

۸ –۱ استفاده از زیرپرسش‌ها [2] (پرسش‌های فرعی)

یکی از نیازهای کاربران امکان طرح پرسشی درون پرسش دیگر می‌باشد. هر گاه بخواهیم پرسشی را درون پرسشی دیگر مطرح نماییم، به آن زیر پرسش می‌گویند. بدیهی است که هر پرسشی چه بصورت پرسش بیرونی باشد و یا بصورت پرسش درونی، لازم است که با عبارت Select آغاز گردد. زیر پرسش‌ها معمولاً بعنوان قسمتی از عبارت Where بکار می‌روند.

عباراتی که بصورت زیرپرسش باشند معمولاً می‌توانند بصورت تودرتو[3] نوشته شوند. در چنین شرایطی برای اجرای هر عبارت بیرونی لازم است ابتدا عبارت جستجوی درونی محاسبه شده و سپس عبارت بیرونی مورد بررسی قرار گیرد. به بیانی دیگر مسیر حل در عبارات تودرتو از درونی ترین عبارت به سمت بیرونی ترین عبارت خواهد بود. عبارت زیر بعنوان نمونه‌ای از این نوع پرسش‌ها می‌باشد:

```
Select ename,deptno,sal
From EMP
Where Sal > (Select AVG (sal) as Xval
From EMP)
```

خروجی حاصل از عبارت بالا به شرح ذیل می‌باشد:

---

1. nested Queries
2. Sub query
3. Nested

ENAME	DEPTNO	SAL
JONES	20	2975
BLAKE	30	2850
CLARK	10	2450
SCOTT	20	3000
KING	10	5000
FORD	20	3000

۸ -۲ عملگر IN

معمولاً در زیر پرسش‌ها شرایط بگونه‌ایست که کاربر می‌خواهد مقداری را از بین یک مجموعه مقادیر بدست آورد. برای این منظور از عملگر IN استفاده می‌گردد. این عملگر تقریبا شبیه به عملگر Between می‌باشد با این تفاوت که مقداری را درون یک مجموعه مقادیر مورد جستجو قرار می‌دهد. مثال زیر نمونه‌ایست از عباراتی که بدین صورت نوشته می‌شوند:

```
Select ename,sal
        From emp
                Where empno IN
                                (Select empno
                                        From emp
                                                Where sal >2000)
```

خروجی حاصل از عبارت بالا به صورت جدول زیر می‌باشد:

ENAME	SAL
JONES	2975
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
FORD	3000

ممکن است این سئوال به ذهن برسد که عبارت مذکور را می‌توان بصورتی ساده‌تر نیز نوشت که در آن نیازی به استفاده از عملگر IN نباشد. در اینجا لازم است گفته شود که عبارت مذکور فقط جهت نشان دادن نحوه استفاده از عملگر IN می‌باشد.

یکی از نکات کلیدی در استفاده از عملگر مذکور این است که باید زیرپرسش جلوی عملگر IN حتما دارای یک ستون همنوع با ستون مورد پرسش باشد و نمی‌توان در زیرپرسش از هر ستونی بعنوان خروجی استفاده کرد.

۸ -۳ عملگر EXISTS

یکی دیگر از عبارات مورد استفاده در پرس و جوهای تودرتو، استفاده از کلمه کلیدی EXISTS می‌باشد. این دستور نیز تا حدودی شبیه به کلمه کلیدی IN می‌باشد. معمولاً از عملگر EXISTS به ندرت استفاده می‌شود. در هنگام استفاده از این عملگر می‌توان در قسمت Where عبارت درونی از مقادیر عبارت بیرونی استفاده کرد. عملگر EXISTS تا حدودی با سایر عملگرهای مربوط به عبارات پرس و جوی تودرتو متفاوت می‌باشد. زیرا ترکیبی از مقادیر مربوط به عبارات بیرونی و درونی درون شرط عبارات درونی قرار می‌گیرد. برای مثال به عبارت زیر توجه کنید. همانطور که مشاهده می‌کنید مقدار صفت خاصه STID از رابطه بیرونی در قسمت WHERE عبارت درونی مورد مقایسه قرار گرفته است.

```
SELECT STNAME
FROM STT
WHERE EXISTS (SELECT *
FROM STCOT
WHERE STCOT.STID = STT.STID
AND COID = '1234')
```

۹ - دستورات کنترل مجوز دسترسی

دستورات کنترل کننده مجوز دسترسی در پایگاه‌های داده به دو صورت کلی زیر تقسیم می‌شود:

- دسترسی به داده‌ها
- دسترسی به امکانات مدیریتی

با توجه به اینکه بحث دسترسی به امکانات مدیریتی در این قسمت از کتاب مطرح نیست، لذا صرفا کنترل دسترسی به داده‌ها مورد بررسی قرار می‌گیرد. دستورات کنترل کننده دسترسی به داده‌ها به شرح زیر می‌باشد:

- دستور اعطا[1] اختیارات
- دستور لغو[2] اختیارات

۹ -۱ دستور اعطا اختیارات

برای اعطا امتیاز از دستور GRANT استفاده می‌شود. شکل کلی این دستور به شرح زیر می‌باشد:

```
GRANT Privileges | ALL Privileges
ON Object
TO Users [WITH GRANT OPTION];
```

در قسمت Privileges از یکی یا ترکیبی از کلمات زیر استفاده می‌گردد:

```
INSERT
DELETE
UPDATE
SELECT
```

در قسمت Object یکی از اشیاء موجود در Schema یک کاربر انتخاب شده و در قسمت Users نام یکی از کاربران پایگاه‌داده انتخاب می‌گردد.

عبارت WITH GRANT OPTION در مواردی مورد استفاده قرار می‌گیرد که بخواهیم جدا از اعطا اختیارات مذکور به یک کاربر، به او اجازه دادن همان اختیارات به کاربر دیگری را نیز بدهیم. برای درک بهتر موضوع به عبارات زیر توجه کنید:

```
Grant delete,update,insert
ON EMP
TO ALI;

Grant select
On dept
TO PERSON[With Grant Option];
```

1. Grant
2. Revoke

۹ -۲ دستور لغو اختیارات

برای لغو امتیاز از دستور REVOKE استفاده می‌شود. شکل کلی این دستور به شرح زیر می‌باشد:

```
REVOKE  Privileges | ALL Privileges
ON Object
FROM Users
```

در قسمت Privileges از یکی یا ترکیبی از کلمات زیر استفاده می‌گردد:

```
INSERT
DELETE
UPDATE
SELECT
```

در قسمت Object یکی از اشیاء موجود در Schema یک کاربر انتخاب شده و در قسمت Users نام یکی از کاربران پایگاه‌داده انتخاب می‌گردد. برای درک بهتر موضوع به عبارات زیر توجه کنید:

```
Revoke select
On DEPT
FROM ALI;
```

۱۰- امکانات و ویژگی‌های SQL2 و SQL3

با توجه به توضیحات ارائه شده در خصوص زبان SQL، در ادامه درخصوص امکانات و ویژگی‌های دو زبان استاندارد SQL2 و SQL3 به اختصار مطالبی بیان شده است.

۱۰ -۱ امکانات SQL2

این نسخه از SQL در سال ۱۹۹۲ میلادی به عنوان SQL استاندارد معرفی گردید. این نسخه در واقع نسخه تصحیح شده و کاملتر نسخه SQL1 بود. در این دستور امکاناتی به نسخه SQL1 افزوده شده و یا بعضی از دستورات SQL1 کامل تر شدند. در ادامه تغییرات اعمال شده به اختصار آمده است:

- افزودن دستورات مربوط به ایجاد، اصلاًح و حذف Domain
- تکمیل دستورات مربوط به ایجاد و حذف جدول
- افزودن دستور اصلاًح جدول
- افزودن دستور تعریف جدول موقت
- امکان نوشتن Select.از یک Select دیگر
- امکان نوشتن Select بعد از WHERE
- تکمیل دستور JOIN
- اصلاًح و تکمیل عملگرهای خاص مانند فرااجتماع، اسکالر و...
- افزودن امکانات جامعیتی و ایمنی و...

۱۰-۲ امکانات SQL3

این نسخه از SQL برای محیط‌های ORDBMS[1] تولید شده است. در واقع پژوهشگران به این نتیجه رسیدند که تکنولوژی‌های RDBMS و OODBMS هیچیک به تنهایی کامل نبوده و صرفا ترکیب این دو تکنولوژی پاسخگوی نیازهای آتی خواهد بود. لذا تصمیم گرفته شد نسخه ای از SQL ارائه گردد تا پاسخگوی نیازهای آتی باشد. دراین نسخه امکانات زیر افزوده شده‌اند.

- ایجاد امکان تعریف انواع داده‌های جدید. مدل داده ای قوی تر برای نمایش داده‌هایی که امکان نمایش آنها با ساختار جدولی وجود ندارد.
- رویه ای شدن زبان. به نحوی که SQL را از نظر برنامه سازی و محاسباتی کامل و انرا به یک زبان برنامه سازی تبدیل می‌کند.
- ایجاد امکان راه‌انداز. راه‌انداز[2] قاعده (محدودیت) و یا قواعدی است که قبل یا بعد از بروز یک رویداد در پایگاه‌داده‌ها و یا بجای یک رویداد باید اعمال گردد. این قاعده یا محدودیت در سطح برنامه سازی ، بصورت رویه ای از پیش تعریف شده است که بصورت شرطی یا غیر شرطی، قبل یا بعد از انجام یک عمل در پایگاه‌داده‌ها بصورت اتوماتیک اجرا می‌شود

1. Object Relational Database Management System
2. Trigger

تمرینات

تمرین ۱: رابطه‌های زیر را در نظر بگیرید

Student (SID, Name, Major, Gradelevel, Age)
Class (Name, Time, Room)
Enrollment (Student Number, Class Name, Polition Number)
Junior (Snum, Name, Major)
Honor-Student (Number, Name, Interset)
Facuity (FID, Name, Department)

STUDENT

SID	Name	Major	Gradelevel	Age
100	J	Hestory	G	21
150	P	Acount	S	19
200	B	Math	G	50
250	G	Hestory	S	50
300	B	Acount	S	41
350	RU	Math	J	20
400	R	Acount	F	18
450	J	Hestory	S	24

Enrollment

Student Num	Class Name	Polition Num
100	BD445	1
150	BA200	1
200	BD445	2
250	CS250	1
300	CS150	1

350	BA200	2
400	BF410	1
450	CS250	2

Class

Name	Time	Room
BA200	F9	110
BD445	F3	213
BF410	F8	213
CS150	F3	314
CS250	F12	210

خروجی حاصل از عبارات زیر را بدت آورید؟

```
SELECT    Name,Major,Age
FROM    Student
WHERE Major = ' Acount'
ORDER   by Name

SELECT    Name,Major,Age
FROM    Student
WHERE   Gradelevel   IN ['F', 'S']

SELECT   Count (*)
FROM    Student

SELECT    Count (DISTINCT Major)
FROM    Student

SELECT   Major, Count(*)
FROM    Student
Order   By  Major

SELECT   Major, Count(*)
FROM    Student
GROUP  By  Major
HAVING   Count (*)  >   2

SELECT   Major, AVG(Age)
FROM    Student
WHERE   Gradelevel  = 'S'
GROUP  By  Major
HAVING   COUNT (*)  >   1
```

```
SELECT    Name
FROM    Student
WHERE   SID   IN
          (SELECT    Student Number
            FROM    Enrollment
            WHERE   Class Name = '445' )
```

تمرین ۲:

بانک اطلاعاتی عرضه کنندگان قطعات و پروژه‌ها به شرح زیر تنظیم شده است ، بر اساس آنها به سوالات مطرح شده پاسخ دهید ؟

```
S { S# , SNAME , STATUS , CITY }
PRIMARY KEY  { S# }
P { P# , PNAME , COLOR , WEIGHT , CITY }
PRIMARY KEY  { P# }
J { J# , JNAME , CITY }
PRIMARY KEY  { J# }
SPJ { S# , P# , J# , QTY }
PRIMARY KEY  { S# , P# , J# }
FOREIGN KEY { S# } REFRENCES   S
FOREIGN KEY { P# } REFRENCES   P
FOREIGN KEY { J# } REFRENCES   J
```

شماره قطعاتی را مشخص کنید که بیش از یک عرضه کننده آنها را عرضه می‌کنند؟

اسامی‌عرضه کنندگانی که قطعه P2 را عرضه نمی‌کنند ؟

فصل ۹

نرمال سازی

هدف کلی

در این فصل ابتدا تعریفی از رابطه نرمال را ارائه کرده و معایب و مزایای رابطه‌های نرمال و غیرنرمال را بیان خواهیم کرد. سپس با مفهوم تئوری وابستگی و در ادامه مفاهیم تئوری وابستگی تابعی، وابستگی تابعی و حالتهای وابستگی آشنا خواهیم شد. در ادامه مفهوم نرمال سازی و انواع صورت‌های نرمال رابطه‌ها را به تفصیل مورد بحث و بررسی قرار خواهیم داد.

هدف رفتاری

دراین فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- تعریف رابطه نرمال
- مزایا و معایب رابطه‌های نرمال
- مزایا و معایب رابطه‌های غیر نرمال
- مفاهیم تئوری وابستگی و انواع وابستگی‌ها
- حالتهای وابستگی
- نمودار وابستگی‌های تابعی
- صورت‌های نرمال
- فرایند نرمال سازی

۱- تعریف رابطه نرمال

یکی از مهم ترین مباحث در پایگاه‌های داده رابطه‌ای، بحث نرمال سازی رابطه‌ها است. حال این سئوال به ذهن می‌رسد که اصولا رابطه نرمال چیست. در ادامه تعاریفی از رابطه نرمال را ارائه می‌کنیم.

- **تعریف اول:** رابطه نرمال رابطه‌ای است که مقادیر تمام صفاتش تجزیه نشدنی باشند.
- **تعریف دوم:** رابطه نرمال رابطه‌ای است که مقادیر هیچیک از میدانهایش، یک مقدار رابطه‌ای با کاردینالیتی بزرگتر از یک نباشد.

در توصیف تعاریف بالا می‌توان گفت یک مقدار ساده تجزیه نشدنی را می‌توان یک مقدار رابطه‌ای از درجه یک و کاردینالیتی یک دانست. به بیانی دیگر در بدنه رابطه نرمال، صفت ساده یا مرکب چند مقداری وجود ندارد و اصطلاحا می‌گوییم رابطه مسطح است. با توجه به تعاریف بالا ***رابطه غیر نرمال*** را می‌توان بصورت زیر توصیف کرد:

> رابطه‌ای که در آن مقادیر حداقل یک صفت، خود مقادیر رابطه‌ای باشند را رابطه غیر نرمال گوییم. به بیانی دیگر رابطه‌ای که حداقل یک صفت ساده یا مرکب چند مقداری داشته باشد، رابطه غیر نرمال است.

۱-۱ دلایل لزوم نرمال بودن رابطه

همانطور که می‌دانید نرمال بودن رابطه دلیل ریاضی ندارد و می‌توان گفت صرفا جنبه سادگی آن مطرح است. سادگی عنصر اساسی ساختار رابطه‌ای خود سادگی‌های زیر را در بر دارد:

- سادگی در نمایش ظاهری رابطه
- سادگی دستورات زبان
- سادگی اجرای عملیات در پایگاه‌داده

۱-۲ معایب رابطه نرمال

رابطه نرمال معایبی دارد که به شرح ذیل آمده است:

- بروز پدیده افزونگی که می‌تواند فیزیکی هم باشد.
- طولانی تر شدن کلید رابطه
- عدم امکان نمایش داده‌های پیچیده
- دشواری در نمایش طبیعی مفهوم سلسله مراتب
- دشواری در نمایش مفهوم وراثت

۱-۳ مزایا و معایب رابطه غیر نرمال

رابطه غیر نرمال نیز دارای مزایایی به شرح ذیل می‌باشد

- کاهش میزان افزونگی
- کوتاه شدن کلید
- امکان نمایش داده‌های پیچیده
- دشواری کمتر در نمایش مفهوم سلسله مراتب
- دشواری کمتر در نمایش مفهوم وراثت
- افزایش سرعت عملیات (در بعضی موارد) در بازیابی داده‌ها

بدیهی است که در کنار این مزایا، رابطه غیر نرمال خود نیز دارای معایبی می‌باشد که این معایب در ذیل آمده است:

- پیچیدگی در نمایش داده‌ها
- پیچیدگی در نگارش دستورات
- پیچیدگی در اجرای دستورات

۲- مفاهیم تئوری وابستگی[1]

در بحث نرمال سازی رابطه‌ها، به مفاهیم تئوری وابستگی نیاز داریم. در این قسمت با برخی مفاهیم وابستگی آشنا می‌شویم.

1. Dependency

۲-۱ انواع وابستگی‌ها

وابستگی دارای حالات مختلفی است که در ذیل آمده است:

- وابستگی تابعی[1]
- وابستگی پیوندی[2]
- وابستگی چند مقداری[3]

۲-۱-۱ وابستگی تابعی

مهمترین نوع وابستگی وابستگی تابعی است. در حالت کلی وابستگی تابعی را می‌توان اینگونه توصیف کرد:

اگر A و B دو صفت در شمای R باشند ، آنگاه وابستگی تابعی $A \rightarrow B$ برقرار است به شرطی که برای تمامی‌رابطه‌ها در R به ازای هر مقدار A فقط یک مقدار B وجود داشته باشد.

وابستگی تابعی را می‌توان بدین صورت نیز توصیف کرد:

فرض می‌کنیم که R یک متغیر رابطه‌ای و A و B دو زیر مجموعه دلخواه از مجموعه عنوان R یعنی H باشند. می‌گوییم B با A وابستگی تابعی دارد و چنین نمایش می‌دهیم: $A \rightarrow B$ اگر و فقط اگر در هر مقدار ممکن از متغیر رابطه‌ای R ، به هر مقدار A فقط یک مقدار B متناظر باشد. به بیان دیگر اگر t1 و t2 دو تاپل متمایز از R باشند، در اینصورت

If t1.A = t2.A then t1.B = t2.B

۲-۱-۲ وابستگی تابعی کامل

$A \rightarrow B$ را وابستگی تابعی کامل [4] گویند اگر B به هیچ زیرمجموعه از A وابسته نباشد. وابستگی تابعی از ویژگی‌های معنایی صفات است و به بیان دیگر هر

1. functional Dependency
2. Join Dependency
3. Multi Value Dependency
4. Full Functional Dependency

وابستگی تابعی بین دو صفت، نمایشگر یک قاعده سمنتیک در جهان واقعی است. مثلا وقتی می‌گوییم:

STID→STMJR

معنایش این است که یک دانشجو فقط دریک رشته تحصیل می‌کند.

۲-۲ حالتهای وابستگی

وابستگی دارای حالات مختلفی می‌باشد که در ذیل نام برده شده‌اند:

- وابستگی به کلید
- وابستگی به بخشی از کلید
- وابستگی غیرکلید به غیرکلید
- وابستگی معکوس
- کلید یا بخشی از آن به صفت یا صفتهای دیگر وابسته باشد

۲-۳ تعریف + F

اگر F یک مجموعه از وابستگی‌های تابعی باشد ، آنگاه مجموعه تمام وابستگی‌های تابعی قابل استخراج از آنرا مجموعه پوششی F می‌نامند و با +F نشان می‌دهند.

روش یافتن+ F:

برای یافتن +F عملیات زیر را به ترتیب انجام می‌دهیم:

- بازتاب

IF $B \subseteq A$ THEN $A \rightarrow B$

- افزایشی

IF $A \rightarrow B$, C THEN $AC \rightarrow BC$

((توجه: C صفت می‌باشد))

- انتقال

IF $A \rightarrow B$, $B \rightarrow C$ THEN $A \rightarrow C$

چون اعمال این قواعد مشکل می‌باشد ، آنها را به صورت ساده تر بیان می‌کنیم.

- اجتماع

IF A → B AND A → C THEN A → BC

- تجزیه

IF A → BC THEN A → B AND A → C

- ترکیب

IF A → B , C → D THEN AC → BD

با اعمال این قواعد سعی می‌کنیم وابستگی‌های تکراری را حذف نموده و به مجموعه وابستگی کمینه یا بهینه برسیم.

مثال:

$F^+ = \{ A \to B , A \to C , B \to C , AB \to C , AC \to D \}$

$\min (F^+) = \{ A \to B , B \to C , A \to D \}$

مثال:

R=(U ,V,W,X,Y,Z)

F={ U → XY , X → Y , XY → ZV }

$F^+ = \{ U \to XY , X \to Y , XY \to ZV , U \to ZV \}$

$F^+ = \{ U \to X, U \to Y , X \to Y, XY \to ZV , U \to ZV \}$

$F^+ = \{ U \to X, U \to Y , X \to Y, X \to ZV , U \to ZV \}$

$F^+ = \{ U \to X, U \to Y , X \to Y, U \to Z , U \to V, X \to Z, X \to V \}$

تمرین:

R = (U ,V,W,X,Y,Z,O,P,Q)

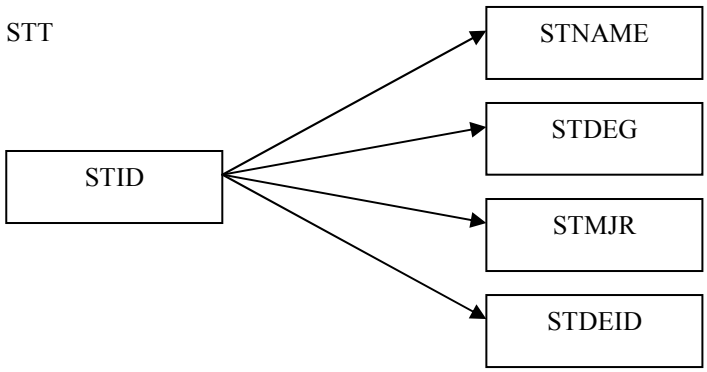
F={ U → VXQ , UVP → O , OQ → YZ, UP → XY }

۲-٤ نمودار وابستگی‌های تابعی [1]

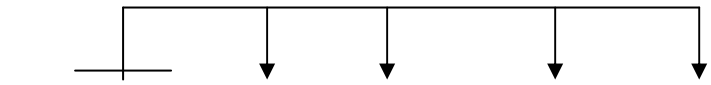
جهت درک بهتر از وابستگی‌ها ، نمودار وابستگی‌های تابعی را رسم می‌کنیم. نحوه نمایش نمودار وابستگی تابعی برای رابطه STT به یکی از دو شکل زیر قابل ترسیم است.

1. Functional Dependency Diagram

روش اول



روش دوم



STT (STID, STNAME, STDEG, STMJR, STDEID)

۳- صورت‌های نرمال

کاد واضع مدل رابطه‌ای ، در آغاز سه سطح نرمال را تعریف کرد. این سه سطح عبارت هستند از:

- صورت نخست نرمال[۱]
- صورت دوم نرمال[۲]
- صورت سوم نرمال[۳]

پژوهشگران و متخصصین بعدا سه نوع دیگر از صور نرمال را به شرح زیر تعریف کردند:

- صورت نرمال بویس- کاد[٤]
- صورت چهارم نرمال[٥]

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. Boyce-Codd Normal Form
5. Forth Normal Form

- صورت پنجم نرمال[1]

امروزه دو نوع دیگر از صورت‌های نرمال تعریف شده‌اند که البته در شرایط بسیار خاص رخ می‌دهند. این دو صرت نرمال عبارتند از:

- صورت نرمال میدان- کلیدی
- صورت نرمال تحدید- اجتماع

در اینجا لازم است به نکات زیر توجه شود:

- چنانچه رابطه‌ای در هر یک از حالات نرمال، بصورت غیرنرمال باشد، می‌گوییم رابطه در آن سطح دارای آنومالی است.
- در واقع هشت صورت نرمال مذکور هر یک از سطح قبلی خود نرمال تر هستند (بغیر از BCNF که شرایط خاصی دارد).
- همچنین لازم به ذکر است که صورت‌های BCNF و 4NF و 5NF دارای شرایط خاصی بوده و تقریبا چنانچه رابطه در یکی از این سه سطوح نرمال باشد، در دو مورد دیگر نیز نرمال خواهد بود(این موضوع همیشگی نیست).
- حالات بالاتر از 3NF شرایط خاصی هستند که در خیلی از مواقع اصلاً رخ نمی‌دهند. لذا چنانچه رابطه‌ای را تا سطح 3NF نرمال کنیم، بطور معمول می‌گوییم نرمال سازی را انجام داده‌ایم. با توجه به تعاریف بالا می‌توان سطوح نرمال سازی را بصورت ذیل نشان داد:

DKNF ⊂ 5NF ⊂ 4NF ⊂ BCNF ⊂ 3NF ⊂ 2NF ⊂ 1NF

٤- فرایند نرمال سازی[2]

با توجه به داده‌های عملیاتی و ارتباط بین موجودیتها، لازم است تا قبل از پیاده سازی جداول طراحی شده، آنها را نرمال سازی نماییم. مراحل نرمال سازی رابطه‌ها به شرح ذیل می‌باشند:

1. Fifth Normal Form
2. Normalization

- مراحل نرمال سازی
- مشخص کردن جداول
- شناسائی داده‌ها و ارتباطشان
- رسم نمودار وابستگی
- طراحی جداول اولیه
- در نظر گرفتن وابستگی‌ها
- اعمال فرمهای نرمال

در ادامه رابطه‌های نرمال را مورد بحث و بررسی قرار خواهیم داد:

٤-١ رابطه 1NF

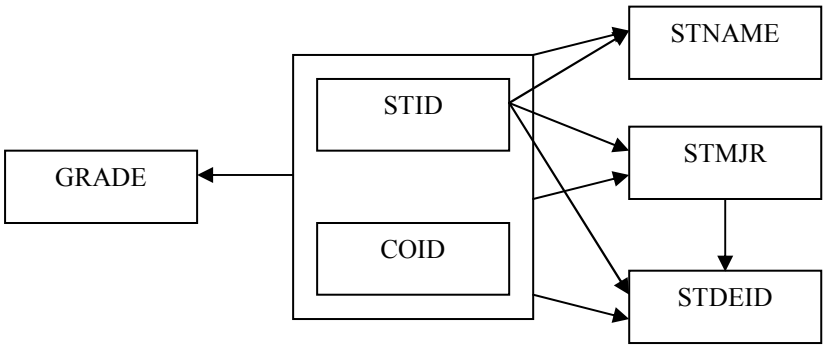
رابطه‌ای 1NF است که:

- همه کلیدهای آن تعریف شده باشد
- همه صفت‌های آن به کلید اصلی وابسته باشند
- صفت‌های ترکیبی نداشته باشیم

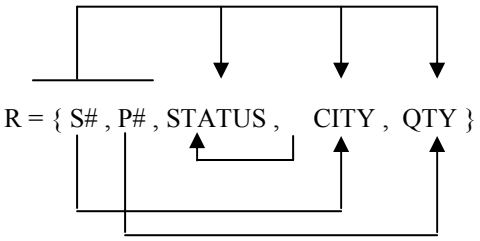
به بیانی دیگر رابطه‌ای 1NF است اگر و فقط اگر هر صفت آن در هر تاپل، تک مقداری باشد (صفت چند مقداری نداشته باشد).

مثال ۱: رابطه STCOR را که در آن STID و COID تواما کلید اصلی هستند را در نظر بگیرید:

STCOR (STID, COID, STNAME, GRADE, STMJR, STDEID)



مثال ۲:



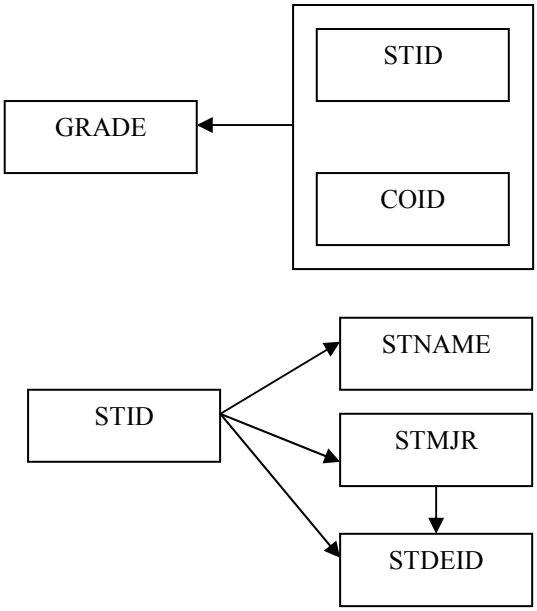
٤-۲ رابطه 2NF

رابطه‌ای 2NF است که

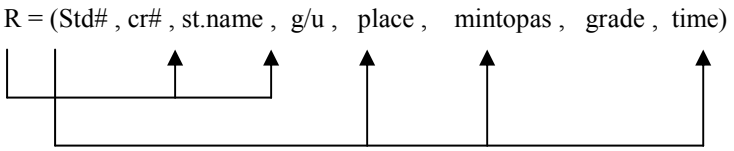
- 1NF باشد
- صفت‌های آن به زیر مجموعه‌های کلید اصلی وابسته نباشد

به بیانی دیگر رابطه‌ای 2NF است که اولا 1NF باشد و ثانیا تمام صفات غیر کلید با کلید اصلی وابستگی تابعی کامل داشته باشند.

مثال ۱:



مثال ۲:



1NF می‌باشد برای 2NF بودن:

R1=(Std# , cr# , grade)
R2=(Std# , st.name , g/u)
R3= (cr# , place , mintopas , time)

بعد از اینکه جداول شکسته شد ، باید 1NF و 2NF بودن هرکدام از جداول بدست آمده را بررسی کنید (هر سه جدول بدست آمده فوق هم 1NF وهم 2NF می‌باشد).

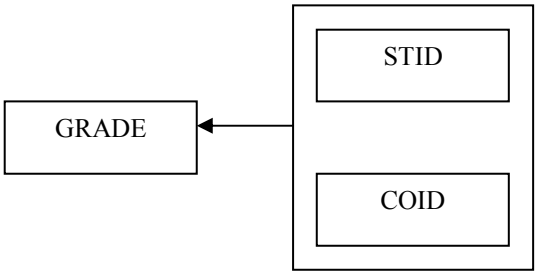
٤-۳ رابطه 3NF

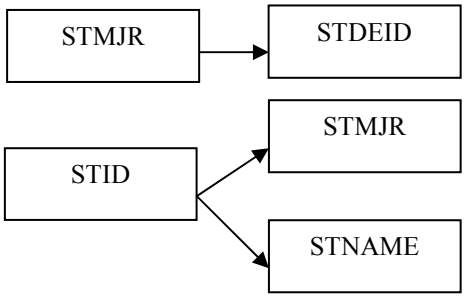
جدولی 3NF است که

- در 2NF باشد
- وابستگی انتقالی نداشته باشد (وابستگی غیر کلید به غیر کلید)

به بیانی دیگر رابطه‌ای 3NF است اگر و فقط اگر 2NF باشد و هر صفت غیرکلید با کلید اصلی، وابستگی تابعی بی واسطه داشته باشد.

مثال ۱:





٤-٤ رابطه BCNF

رابطه BCNF حالت خاصی از رابطه نرمال است و دارای تعاریف متعددی می‌باشد که در ذیل آمده است:

تعریف اول: جدولی BCNF است که

- در 3NF باشد
- وابستگی معکوس نداشته باشد (وابستگی کلید به غیر کلید)

تعریف دوم: رابطه‌ای BCNF است اگر و فقط اگر در آن هر دترمینان، کلید کاندید باشد.

تعریف سوم: رابطه R(H)، با مجموعه وابستگی‌های تابعی F، در BCNF است اگر برای هر وابستگی تابعی F+ به صورت A→B که در آن A ⊂ R(H) و B ⊂ R(H) است، حداقل یکی از دو حالت زیر برقرار باشد:

- $A \rightarrow B$ یک وابستگی غیر مهم باشد یعنی $B \subset A$
- A سوپر کلید رابطه R باشد

تعریف چهارم: رابطه‌ای BCNF است اگر و فقط اگر سمت چپ هر وابستگی تابعی مهم و کاهش ناپذیر، کلید کاندید رابطه باشد.

٤-٥ رابطه 4NF

رابطه‌ای 4NF است اگر

- BCNF باشد
- در آن وابستگی چند مقداری مهم وجود نداشته باشد.

مشکل آنومالی در سطح 4NF بدین صورت است که گاهی دو رابطه‌ای که از یکدیگر مجزا هستند را به اشتباه و صرفا بواسطه اینکه فرض کرده‌ایم می‌توانیم تحت یک رابطه قرار دهیم، با یکدیگر ترکیب می‌کنیم و تحت عنوان یک رابطه در پایگاه‌داده قرار می‌دهیم. پس از ورود اطلاعات در رابطه جدید، این رابطه را به دو رابطه قبلی تجزیه می‌کنیم و طبیعتا مقادیر رابطه کلی در د رابطه تقسیم می‌شوند. حال چنانچه بخواهیم دو رابطه را با هم ترکیب کنیم، با حذف بعضی از تاپل‌ها مواجه می‌شویم. البته این شرایط بسیار خاص بوده و امکان رخداد آن نیز بسیار کم است. برای مثال رابطه زیر را که اشتباها نرمال شده است را در نظر بگیرید:

PSR(PRID,STID,RNUM)

PRID	STID	RNUM
Pr22	S2	R1
Pr22	S1	R1
Pr22	S3	R1
Pr22	S2	R2
Pr22	S1	R2
Pr22	S3	R2
Pr33	S2	R1
Pr33	S6	R1

درج تاپلی مانند (Pr22,S8) عملا به مفهوم درج دو تاپل در جدول PSR بصورت زیر خواهد بود:

(Pr22,S8,R1)
(Pr22,S8,R2)

حال چنانچه رابطه PSR را عملا از ابتدا بصورت دو رابطه PS و PR مانند زیر در نظر بگیریم دیگر دچار چنین مشکلاتی نخواهیم شد.

PS(PRID,STID)
PR(PRID,RNUM)

٤-٦ رابطه 5NF

در فرایندهای نرمال سازی تا سطح 4NF، الگوریتم عمومی‌درهر مرحله عبارت بود از تجزیه رابطه به دو رابطه نرمالتر به گونه‌ایکه با پیوند دو رابطه، همان رابطه اصلی بدست می‌آید. اماممکن است در وضع خاصی رابطه را به دو رابطه تجزیه کنیم ولی با پیوند دو رابطه، رابطه اولیه بدست نیاید. به مثال زیر توجه کنید:

PRCODE (PRID, COID, DEID)

PRID	COID	DEID
Pr11	C1	D4
Pr11	C3	D1
Pr22	C1	D1
Pr11	C1	D1

حال این رابطه را به دو رابطه زیر تجزیه می‌کنیم. پس رابطه‌های زیر از تجزیه رابطه PRCODE بدست می‌آیند:

PRCO (PRID, COID)
CODE (COID, DEID)

PRCO

PRID	COID
Pr11	C1
Pr11	C3
Pr22	C1

CODE

COID	DEID
C1	D4
C3	D1
C1	D1

حال چنانچه دو رابطه را با یکدیگر ترکیب کنیم در خروجی حاصل یک رکورد اضافی مشاهده می‌گردد.

Proc JOIN Code = XPRCODE

PRID	COID	DEID
Pr11	C1	D4
Pr11	C1	D1
Pr11	C3	D1
Pr22	C1	D4
Pr22	C1	D1

→ رکورد اضافه

به این شرایط آنومالی در سطح 5NF می‌گویند. رفع این مکل صرفا تا قبل از ورود داده‌ها ممکن خواهد بود.

با این وصف، تعریف رابطه 5NF به شرح زیر خواهد بود:

رابطه‌ای 5NF است اگر تمام وابستگی‌های پیوندی آن ناشی از کلیدهای کاندید آن باشد. با این وصف اگر بتوانیم یک وابستگی پیوندی در رابطه R پیدا کنیم که در پرتوهایش کلید کاندید R وجود نداشته باشد، رابطه 5Nf نیست.

توجه: آنومالی در سطوح 4NF و 5NF صرفا در شرایط خاص و برای محیطهای خاص رخ می‌دهد و در حالت عادی این نوع آنومالی‌ها در طراحی ساختار اطلاعاتی سیستم‌ها رخ نمی‌دهد. لذا بطور معمول اگر بتوانیم ساختار اطلاعاتی سیستم مورد نظر را حداکثر تا سطح 3NF و یا BCNF نرمال کنیم، اصطلاحا می‌گوییم نرمال سازی کامل انجام شده است.

تمرینات

۱. رابطه نرمال را تعریف کنید.
۲. مزایا و معایب رابطه نرمال را نام ببرید
۳. رابطه غیر نرمال دارای چه مزایا و معایبی خواهد بود
٤. انواع وابستگی را نام ببرید
٥. وابستگی تابعی را توضیح دهید
٦. حالت‌های وابستگی را نام ببرید
۷. صورت‌های نرمال را نام ببرید
۸. صورت نرمال اول را توضیح دهید
۹. صورت نرمال سوم به چه صورت است
۱۰. صورت نرمال BCNF راشرح دهید
۱۱. صورت نرمال 5NF چگونه است

فصل ۱۰

معماری‌های مختلف سیستم پایگاه‌داده

هدف کلی

در این فصل به موضوع معماری پایگاه‌داده‌ها می پردازیم. ابتدا معماری‌های متمرکز پایگاه‌داده‌ها را شرح داده و سپس وارد بحث معماری‌های غیر متمرکز پایگاه‌داده‌ها می‌شویم. در بحث معماری پایگاه‌های داده نامتمرکز، معماری‌هایی مانند معماری مشتری-خدمتگزار، معماری توزیع شده، معماری با پردازش موازی، معماری چند پایگاهی و سیستم پایگاهی همراه را با ذکر توضیحات در مورد هر یک از انواع معماری‌های مذکور مورد بحث و بررسی قرار خواهیم داد.

هدف رفتاری

دراین فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- معماری متمرکز
- معماری نامتمرکز
- معماری سیستم پایگاهی مشتری- خدمتگزار
- معماری سیستم پایگاهی توزیع شده
- معماری با پردازش موازی
- معماری با حافظه مشترک
- معماری با دیسک‌های مشترک
- معماری بی اجزا مشترک

- معماری سلسله مراتبی
- معماری چند پایگاهی
- سیستم پایگاهی همراه

۱- مقدمه

یکی از مباحث مهم و کلیدی در معماری پایگاه‌داده، اجزاء تشکیل دهنده و پیکر بندی یا طرز ترکیب اجزاء سیستم و چگونگی تعامل اجزاء با یکدیگر است. در معماری پایگاه‌داده‌ها حداقل یک پایگاه‌داده‌ها، یک سیستم مدیریت پایگاه‌داده‌ها، یک سیستم عامل، یک کامپیوتر با دستگاه‌های جانبی و تعدادی برنامه کاربردی و کاربر وجود دارند.

معماری پایگاه‌داده‌ها بستگی به دو عنصر اصلی سیستم یعنی سخت‌افزار و نرم‌افزار مدیریت پایگاه‌داده‌ها دارد. البته عوامل دیگری هم در طراحی این معماری دخالت دارند که اهم این عوامل عبارتند از:

- موقعیت جغرافیایی کاربران
- نیازهای کاربران
- ماهیت پردازش‌ها و تراکنش‌ها
- تعداد تراکنش‌ها
- حجم داده‌های ذخیره شدنی
- موقعیت مکانی داده‌ها و ارتباطات بین آنها
- ماهیت کاربردهای مورد نظر

۲- انواع معماری

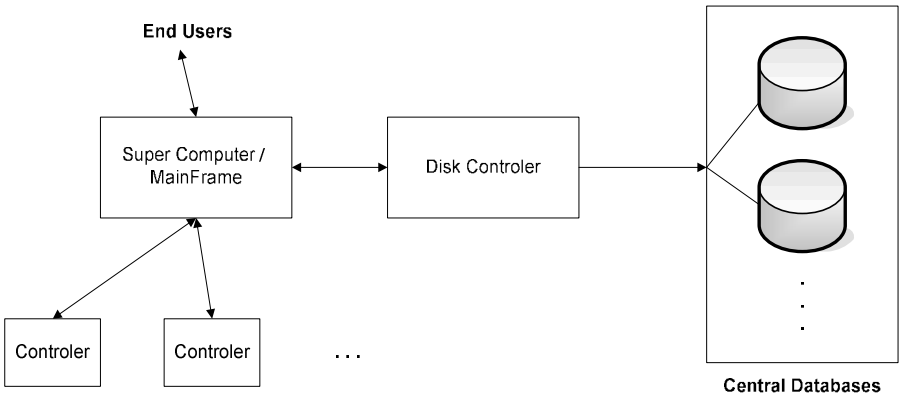
در اساس دو نوع معماری برای سیستم پایگاهی وجود دارد که‌این دو معماری عبارتند از:

- معماری متمرکز
- معماری نا متمرکز

در ادامه هر یک از انواع معماری مورد بحث و بررسی قرار خواهند گرفت.

۲-۱ معماری متمرکز

در این معماری یک پایگاه‌داده‌ها روی یک سیستم کامپیوتری و بدون ارتباط با سیستم کامپیوتری دیگر، ایجاد می‌شود. سخت‌افزار این سیستم می‌تواند کامپیوتر شخصی متوسط و یا بزرگ باشد و طبعا قدرت، توانش و کارایی سیستم نیز متفاوت است.



شکل ۱۰-۱ نمایشی از معماری متمرکز

سیستم با معماری متمرکز که روی یک کامپیوتر شخصی ایجاد می‌شود، تک کاربری، برای کاربردهای کوچک و با امکانات محدود است و سیستم مدیریت نیز توانش چندانی ندارد. اما سیستم معماری متمرکز روی کامپیوترهای متوسط و به ویژه بزرگ متصل به تعداد زیادی پایانه، می‌تواند سیستم کارایی باشد. شکل ۱۰-۱ طرحی از معماری متمرکز ارائه شده است.

۲-۲ معماری نامتمرکز

معماری نا متمرکز، خود دارای انواع مختلفی است از جمله:

- معماری مشتری- خدمتگزار
- معماری توزیع شده

- معماری چند پایگاهی
- معماری با پردازش موازی
- معماری موبایل

در ادامه هر یک از انواع معماری‌های غیر متمرکز را مورد بحث و بررسی قرار خواهیم داد.

۲-۲-۱ معماری سیستم پایگاهی مشتری- خدمتگزار

۲-۲-۱-۱ تعریف

معماری مشتری-خدمتگزار تعریف واحد و مورد پذیرش همگان ندارد. در معنای عام می‌توان این معماری را چنین تعریف کرد:

> هر معماری که در آن قسمتی از پردازش را یک برنامه، سیستم یا ماشین انجام دهد و انجام قسمت دیگر از پردازش را از برنامه، سیستم یا ماشین دیگر بخواهد، معماری مشتری-خدمتگزار نامیده می‌شود.

در واقع وظایفی که باید " سیستم" انجام دهد به دو دسته تقسیم می‌شوند:

- دسته‌ای که انجام آن بر عهده خدمتگزار است
- دسته‌ای که توسط مشتری انجام می‌شود.

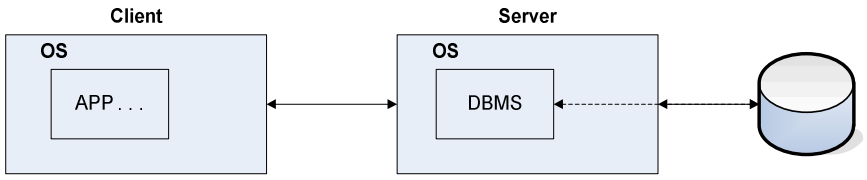
بدین ترتیب یک معماری دو سطحی داریم که برخورد با پیچیدگی سیستم‌های (DBMS) جدید و نیز مشکل توزیع را تسهیل می‌کند. ماشین در این تعریف در معنای عام آن آمده است (فیزیکی یا منطقی). با این تعریف، در این معماری یک ماشین (یا سیستم یا برنامه) خدمتی را به ماشین (یا سیستم یا برنامه) دیگر ارائه می‌کند، از این رو به این ماشین (یا سیستم یا برنامه)، خدمتگزار می‌گویند. بنابراین، منظور از معماری مشتری-خدمتگزار آن نوع از معماری است که در آن مسئولیت‌ها به طور منطقی تقسیم شده است.

توجه: با توجه به‌این تقسیم کار، ممکن است این موضوع به ذهن برسد که حتما حداقل دو کامپیوتر در این معماری وجود دارد. حال آنکه لزوما چنین نیست، دو

برنامه، دو زیر سیستم از یک سیستم کامپیوتری و... می‌توانند با هم تقسیم کار داشته باشند.... اما در محیط‌های جدید، معمولاً با تعدادی کامپیوتر شخصی، ایستگاه کار، چاپگر، خدمتگزار فایل و تجهیزات دیگر، ایجاد می‌شود.

۲-۲-۱-۲ معماری پایگاهی مشتری-خدمتگزار

در این معماری معمولاً دو گروه کامپیوتر داریم: گروه مشتری و گروه خدمتگزار. تمام داده‌ها در کامپیوترهای خدمتگزار ذخیره می‌شوند و تمام برنامه‌های کاربردی در کامپیوترهای مشتری اجرا می‌شوند. در شکل ۱۰-۲ مثالی از طرح ساده شده این معماری با یک مشتری و یک خدمتگزار دیده می‌شود (گاه موسوم به معماری دو ردیفی. البته معماری سه ردیفی (و از نظر تئوریک، n ردیفی، $n \geq 2$) هم وجود دارد).

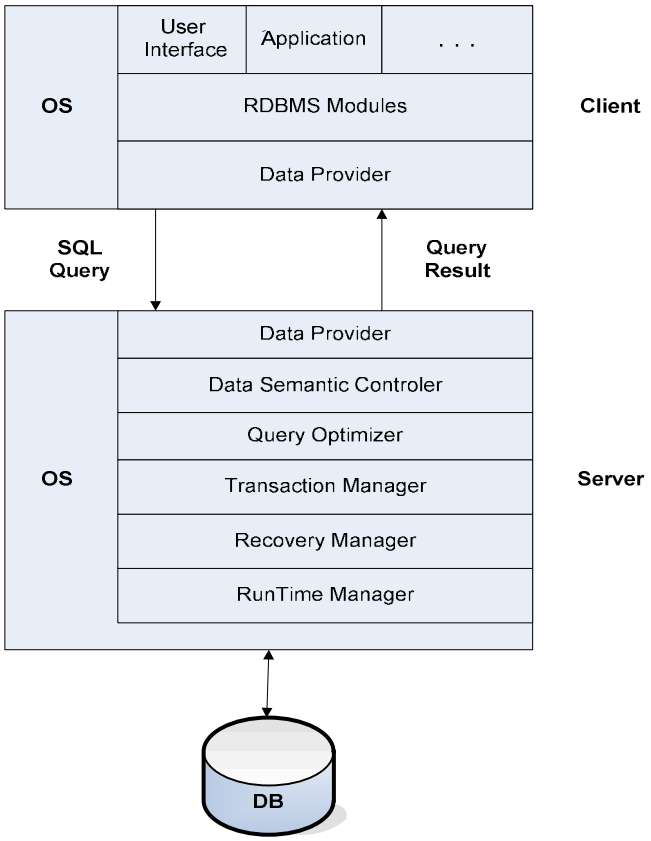


شکل ۱۰-۲ معماری مشتری- خدمتگزار

کاربر در ماشین مشتری می‌تواند از طریق واسط زمانی (مثلا زبان داده‌ای فرعی) یا واسط‌های دیگر مثل واسط گرافیکی، واسط فرمی‌و... عمل کند. در شکل ۱۰-۳، عناصر محیط نرم‌افزاری این معماری با فرض وجود یک RDBMS دیده می‌شود. در این شکل، واحدهایی از RDBMS در ماشین مشتری و بقیه واحدهای آن در ماشین خدمتگزار قرار دارند.

برای تسهیل تماس بین مشتری و خدمتگزار، ابزارهایی مثل ODBC و JDBC و... تولید شده‌اند. این ابزارها در اساس واسط برنامه کاربردی هستند که به مشتری‌ها امکان می‌دهند تا پرسش‌ها به صورت احکام SQL تولید شده و به ماشین خدمتگزار فرستاده شوند. با استفاده از این واسط استانده، هر ماشین مشتری می‌تواند با هر ماشین خدمتگزار متصل باشد و نیاز نیست که محیط سیستمی‌دو ماشین، همگن باشد. در

بعضی از کاربردها، ممکن است واسطهای خاص در ماشین مشتری وجود داشته باشد و حتی خود ماشین مشتری هم می‌تواند ماشین خاصی باشد.



شکل ۱۰-۳ یک طرح ممکن از اجزاء نرم‌افزاری معماری مشتری–خدمتگزار

در بعضی از سیستم‌ها، تماس مشتری با خدمتگزار از طریق فراخوان دور انجام می‌شود. تمام فراخوان‌های دور از سوی ماشین مشتری، در یک (یا چند) ترکنش در ماشین خدمتگزار جای داده می‌شوند تا در این ماشین اجرا شوند، به گونه‌ای که اگر تراکنش به هر دلیلی، طرد شود، خدمتگزار می‌تواند تاثیرات اجرای هر یک از فراخوان‌های دور را از بین برده، پایگاه‌داده‌ها را به وضع صحیح قبل از اجرای تراکنش برگرداند.

۲-۲-۱-۳ طرحهای معماری

۲-۲-۱-۳ -۱ از نظر تعداد مشتری و خدمتگزار

از نظر تعداد مشتری و خدمت گذار طرحهای زیر وجود دارند:

- یک مشتری- یک خدمتگزار
- چند مشتری- یک خدمتگزار
- یک مشتری- چند خدمتگزار
- چند مشتری- چند خدمتگزار

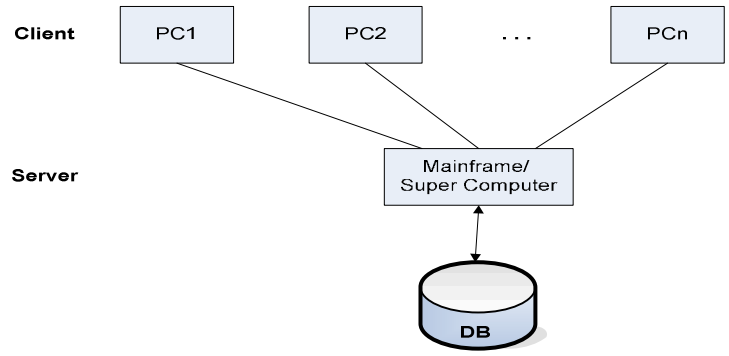
۲-۲-۱-۳ -۲ از نظر پیکر بندی سخت‌افزاری

از نظر پیکربندی سخت‌افزاری ، دو نوع معماری به شرح زیر وجود دارد:

- معماری حول کامپیوتر بزرگ
- معماری حول شبکه

معماری حول کامپیوتر بزرگ

در این طرح، ماشین خدمتگزار یک کامپیوتر بزرگ است و پایگاه‌داده‌ها روی همین کامپیوتر ایجاد و مدیریت می‌شود و تعدادی کامپیوتر شخصی، از خدمات پایگاهی این کامپیوتر بزرگ استفاده می‌کنند. مثالی از طرح این معماری در شکل 10-٤ نشان داده شده است.



شکل ۱۰-٤ معماری مشتری- خدمتگزار حول کامپیوتر بزرگ

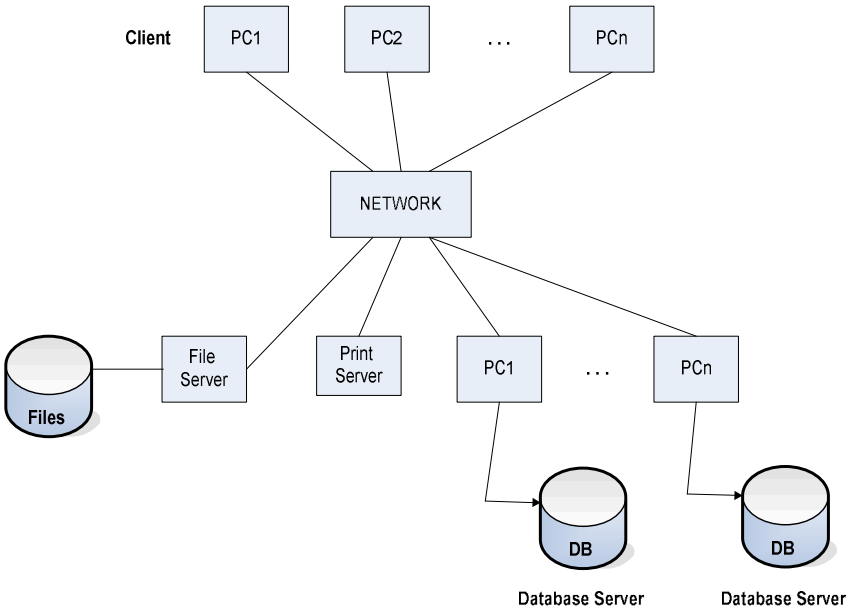
معماری حول شبکه

در این طرح، تعدادی کامپیوتر شخصی به عنوان خدمتگزار و تعدادی دیگر به عنوان مشتری، از طریق شبکه بهم مرتبط‌اند. یک (یا بیش از یک) کامپیوتر شخصی، خدمتگزار پایگاه‌داده‌هاست و خدمتگزاران دیگری هم می‌توانند وجود داشته باشند. مثالی از طرح این معماری در شکل 10-۵ دیده می‌شود.

۲-۲-۱-٤ مزایای معماری مشتری- خدمتگزار

در مقایسه با معماری متمرکز، این نوع معماری مزایای زیر را دارد:

- تقسیم پردازش
- کاهش ترافیک شبکه (در معماری حول شبکه)
- استقلال ایستگاههای کار
- اشتراک داده‌ها



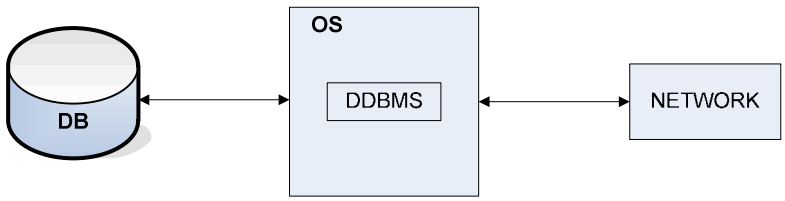
شکل ۱۰-۵ معماری مشتری- خدمتگزار حول شبکه

۲-۲-۲ معماری سیستم پایگاهی توزیع شده

۲-۲-۲-۱ تعریف

مجموعه‌ای از داده‌های ذخیره شده، که منطقاً به یک سیستم تعلق دارند، ولی در کامپیوترهای مختلف که در یک یا بیش از یک شبکه توزیع شده‌اند، قرار گرفته‌اند. می‌توان گفت که در این معماری تعدادی پایگاه‌داده‌های ذخیره شده روی کامپیوترهای مختلف داریم که از نظر کاربران، پایگاه واحدی هستند. به بیان دیگر، این معماری مجموعه‌ای است از چند پایگاه‌داده منطقاً یکپارچه (مجتمع)، ولی به طور فیزیکی توزیع شده روی یک شبکه کامپیوتری. توجه داشته باشید که در این معماری، در سطح طراحی منطقی پایگاه، در آغاز یک پایگاه‌داده‌های یکپارچه داریم که طراح بر اساس یک استراتژی توزیع و یک طرح تخصیص مشخص، داده‌هایش را در چند کامپیوتر توزیع می‌کند. کامپیوترها با یکدیگر چنان همکاری دارند که هر کاربر می‌تواند به داده‌های مورد نیازش در هر کامپیوتر دستیابی داشته باشد به گونه‌ای که انگار داده‌ها در کامپیوتر خودش ذخیره شده باشند.

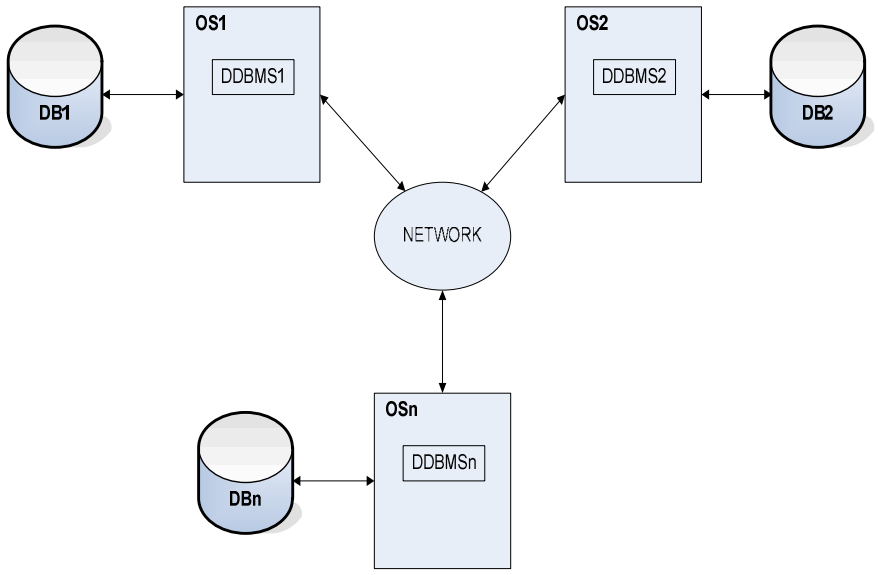
در این معماری هر کامپیوتر خود یک سیستم پایگاه‌داده‌هاست یعنی: پایگاه‌داده‌ها، سیستم مدیریت پایگاه‌داده‌ها و مدیر انتقال داده‌ها دارد. اصطلاحا می‌گوییم تعدادی DBMS محلی داریم و برای ایجاد هماهنگی بین این سیستم‌های محلی، عنصر نرم‌افزاری خاصی که نوعی گسترش DBMS است، لازم است. در واقع در هر کامپیوتر، یک DDBMS داریم، یعنی یک DBMS با توانش ایجاد و مدیریت پایگاه‌داده‌های توزیع شده. بنابرین نمای کلی هر کامپیوتر به صورتی است که در شکل ۱۰-۶ دیده می‌شود.



شکل ۱۰-۶ نمای یک کامپیوتر در معماری توزیع شده

کامپیوترها از طریق امکانات شبکه‌ای (محلی، گسترده و یا متحرک) به هم مرتبط‌اند و هر کامپیوتر اجزا و عناصر سخت‌افزاری و نرم‌افزاری خود را دارد. این اجزا و عناصر می‌توانند همگن یا ناهمگن باشند. مثالی از طرح کلی این معماری درشکل۱۰-۷ دیده می‌شود. با توجه به تعریف این نوع معماری و طرح کلی آن، ویژگیهای این سیستم را می‌توان چنین برشمرد:

- مجموعه‌ای است از داده‌های منطقاً مرتبط و اشتراکی
- بعض بخشها ممکن است بطور تکراری (در چند نسخه) در کامپیوترها ذخیره شده باشند.
- کامپیوترها از طریق یک شبکه بهم مرتبط‌اند.
- داده‌های ذخیره شده در هر کامپیوتر تحت کنترل یک DBMS است.
- DBMS در هر کامپیوتر، می‌تواند برنامه‌های کاربردی محلی، را بطور اتوماتیک اجرا کند.
- هر DBMS حداقل در اجرای یک برنامه کاربردی سرتاسری مشارکت دارد.

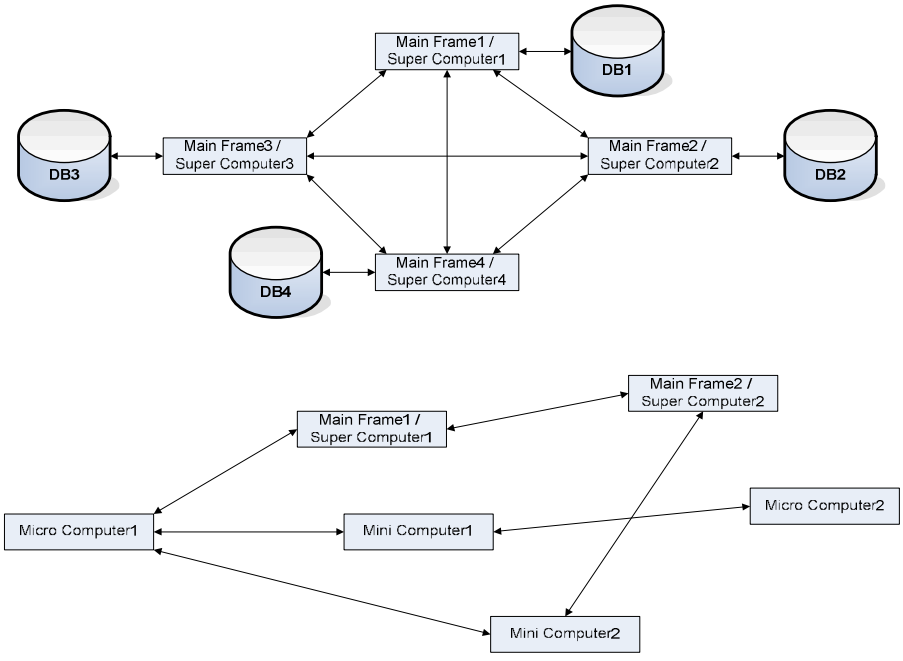


شکل ۱۰-۷ معماری توزیع شده با سه کامپیوتر

۲-۲-۲-۲ پیکربندی سخت‌افزاری

گفتیم که اجزاء تشکیل دهنده‌این معماری می‌توانند همگن یا نا همگن باشند، به ویژه از نظر سخت‌افزاری، پیکربندی‌های گوناگونی پنداشتنی است. در مثال شکل ۱۰-۸ دو پیکربندی دیده می‌شود. در این معماری اجزاء زیر می‌توانند همگن یا نا همگن باشند:

- سخت‌افزار پردازشگر
- سیستم عامل
- سیستم مدیریت پایگاه‌داده (بویژه از نظر مدل داده‌ای و زبان پایگاهی)
- پروتوکلهای شبکه
- سخت‌افزار ارتباط
- سخت‌افزار ذخیره سازی
- تسهیلات و ابزارهای (نرم‌افزاری) جانبی



شکل ۱۰-۸ دو پیکربندی سخت‌افزاری در معماری توزیع شده

۲ -۲-۲-۳ قواعد

اصل مهم در معماری سیستم پایگاهی توزیع شده‌این است که "سیستم" باید جنان عمل کند که کاربران دقیقا مثل محیط پایگاه‌داده‌های متمرکز معمولی از آن استفاده کنند (و این ویژگی، یکی از تفاوت‌های این معماری با معماری مشتری-خدمتگزار است). برای رعایت این اصل مهم، در هر سیستم پایگاهی توزیع شده قواعدی باید رعایت شوند که در واقع اهداف این سیستم هم هستند. برخی از این عوامل عبارتند از:

- خود مختاری محلی (داخلی)
- تداوم عملیات
- ناوابستگی کامپیوترها به کامپیوتر اصلی
- ناوابستگی برنامه‌ها به مکان ذخیره سازی داده‌ها
- ناوابستگی برنامه‌ها به طرز جایدهی داده‌ها در کامپیوترها.
- پردازش پرسشها به گونه‌ای توزیع شده
- ناوابستگی برنامه‌ها به سخت‌افزار
- ناوابستگی برنامه‌ها به سیستم عامل
- ناوابستگی برنامه‌ها به سیستم مدیریت پایگاه‌داده‌ها
- ناوابستگی برنامه‌ها به شبکه

۲-۲-۲-٤ مزایا

برخی از مزایای این معماری عبارتست از:

- سازگاری و هماهنگی با ماهیت سازمان‌های نوین
- کارایی بیشتر در پردازش داده‌ها به ویژه در پایگاه‌داده‌های بزرگ
- دستیابی بهتر به داده‌ها
- اشتراک داده‌ها
- افزایش پردازش موازی
- کاهش هزینه ارتباطات
- تسهیل گسترش سیستم

- استفاده از پایگاه‌داده‌های از قبل موجود

۲-۲-۲-۵ معایب

این معماری معایبی هم دارد از جمله:

- پیچیدگی طراحی سیستم
- پیچیدگی پیاده سازی
- کاهش کارایی در برخی موارد
- هزینه بیشتر
- مصرف حافظه بیشتر

۲-۲-۳ معماری با پردازش موازی

این معماری با ساخت و گسترش ماشین‌های موازی، برای ایجاد پایگاه‌داده‌های خیلی بزرگ و نیز در سیستم‌هایی که چند هزار تراکنش در ثانیه را پردازش می‌کنند ، مورد توجه قرار گرفت. گونه گسترش یافته معماری توزیع شده است و برای تامین کارایی بیشتر، دستیابی پذیری بالا و گسترش پذیری سریع طراحی می‌شود. در اینگونه سیستم‌ها معمولاً تعداد زیادی تراکنش در ثانیه (حدود چند هزار) و بطور موازی اجرا می‌شوند. مطالعه ماشین‌های موازی از حدود این درس خارج است، تنها اشاره می‌شود که ماشین موازی بر اساس یکی از دو طرح کلی زیر طراحی و تولید می‌شود:

- چند پردازنده قوی (دو یا چهار در حال حاضر و گاه بیشتر)
- تعدادی پردازنده کوچک (گاه چند سطر یا حتی بیشتر)

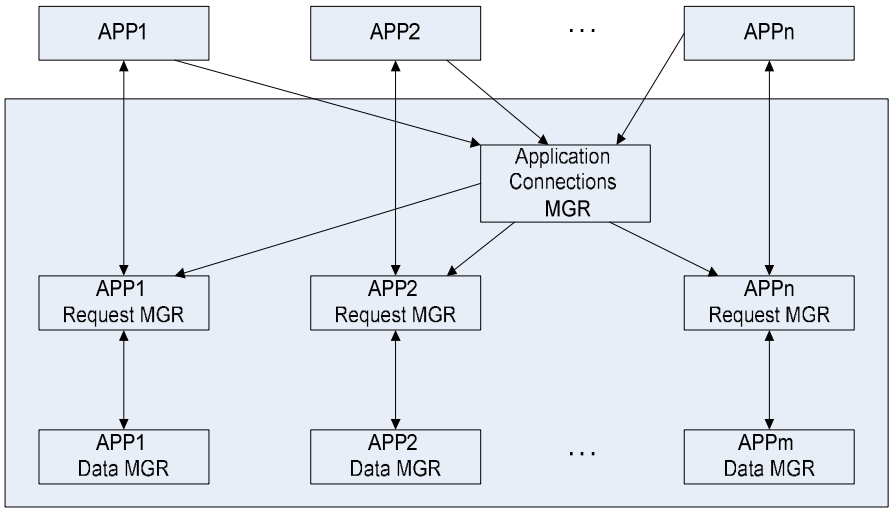
در ماشین با پردازش موازی دو هدف اساسی وجود دارد:

- افزایش سرعت (کارایی)، یعنی انجام یک کار در زمان کمتر
- افزایش مقیاس کار (گسترش کار) یعنی انجام کارهای بزرگ تر در زمان کمتر.

۲-۲-۳-۱ طرح کلی معماری

این معماری به صورت کلی در شکل ۱۰-۹ دیده می‌شود. می‌بینیم که در چنین سیستمی، در اساس سه واحد اصلی وجود دارد:

- مدیر تماس‌های اجرایی کاربران
- مدیر درخواست‌ها
- مدیر داده‌ها



شکل ۱۰-۹ طرح کلی معماری موازی

۲-۲-۳-۲ طرح‌ها

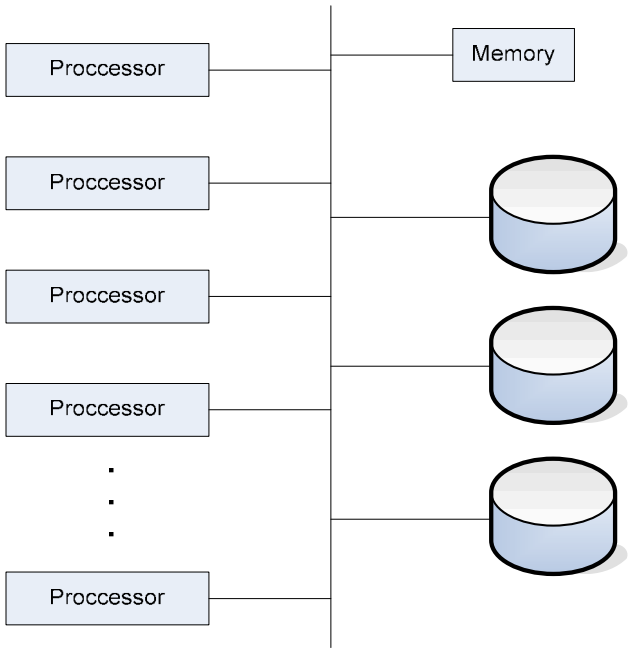
برای ایجاد پایگاه‌داده‌ها با معماری پردازش موازی، به طور کلی، چهار مدل یا طرح وجود دارد:

- معماری با حافظه مشترک
- معماری با دیسک مشترک
- معماری بی اجزاء مشترک
- معماری سلسله مراتبی

برای ایجاد پایگاه‌داده‌های موازی، علاوه بر استفاده از یکی از این معماری‌ها، روش‌هایی نیز برای جایدهی داده‌ها روی دیسک‌ها و یا در حافظه‌های نهان و نیز بهینه سازش پرسش‌های کاربران به کار گرفته می‌شوند، بنابرین صرف وجود معماری موازی برای ایجاد پایگاه‌داده‌های موازی کافی نیست.

۲-۲-۳-۲-۱ معماری با حافظه مشترک

در این طرح، پردازنده‌ها به حافظه مشترک دستیابی دارند. مزیت این طرح این است که ارتباط بین پردازنده‌ها بطور کارا انجام می‌شود. زیرا پیام‌های بین پردازنده‌ها با نوشتن در حافظه مبادله می‌شود که زمان آن کمتر از میکروثانیه است. داده‌های ذخیره شده در حافظه در اختیار همه پردازنده‌ها قرار دارند.



شکل ۱۰-۱۰ معماری با حافظه مشترک

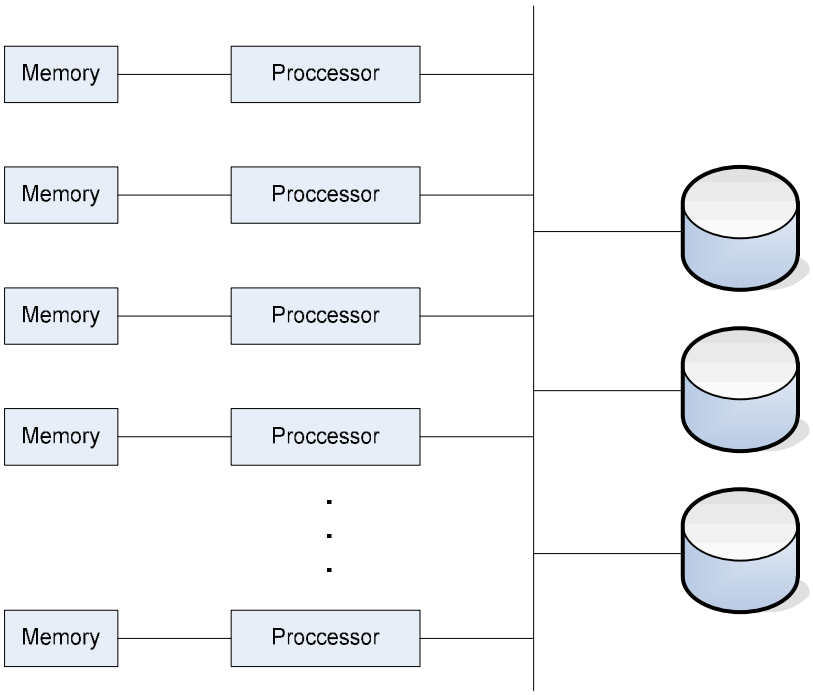
عیب این معماری در این است که نمی‌توان بیش از ۳۲ یا ٦٤ پردازنده داشت، زیرا احتمال بروز تنگنا در بازسای حافظه‌ای یا شبکه ارتباطی افزایش می‌یابد. البته اگر

در هر پردازنده حافظه نهان بااندازه بزرگ وجود داشته باشد، دفعات مراجعه به حافظه اصلی کاهش می‌یابد، هر چند نمی‌توان همه داده‌ها را در این بافرها جای داد. طرح کلی این معماری در شکل ۱۰-۱۰ نشان داده شده است (در این طرح و طرح‌های بعدی معماری موازی، P به جای پردازنده و M به جای حافظه است).

۲-۲-۳-۲-۲ معماری با دیسک‌های مشترک

در این طرح، تمام پردازنده‌ها به تمام دیسک‌ها از طریق شبکه ارتباطی دستیابی دارند. هر پردازنده، حافظه اختصاصی خود را دارد. مزایای این نوع معماری به شرح ذیل می‌باشد:

- عدم بروز تنگنا در باس‌های حافظه
- تسهیل تحمل خرابی، زیرا در صورت خراب شدن یک پردازنده یا حافظه، پردازنده دیگر می‌تواند کار را ادامه دهد. می‌توان از سیستم RAID هم استفاده کرد.

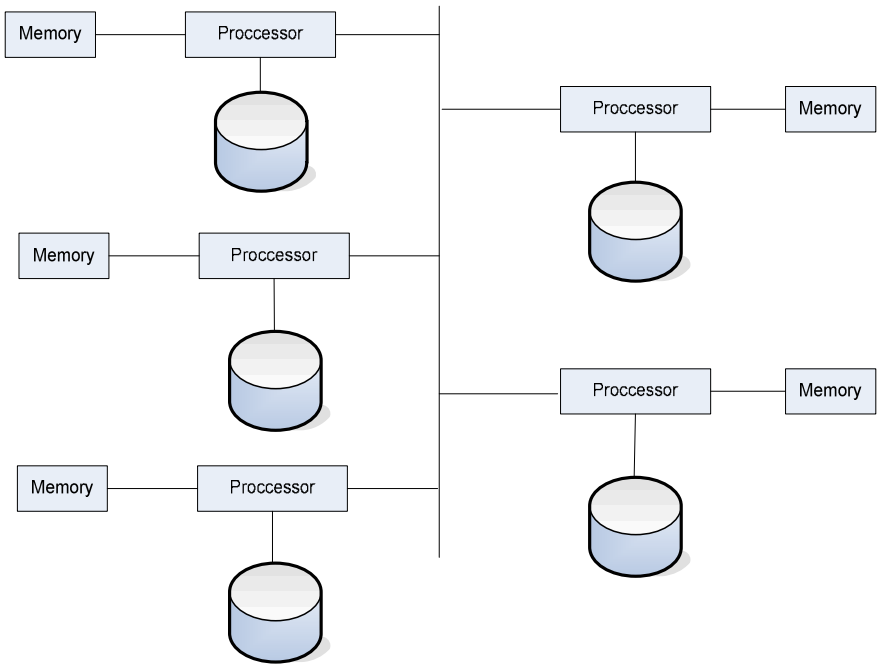


شکل ۱۰-۱۱ معماری با دیسک‌های مشترک

این معماری دارای عیوبی نیز است که مهمترین عیب آن دشواری در گسترش سیستم است، زیرا با افزایش دیسک‌ها و پردازنده‌ها، در ارتباط بین اجزا، تنگنا ایجاد می‌شود و سرعت ارتقا بین آنها کاهش می‌یابد. طرح کلی این سیستم در شکل ۱۰-۱۱ نشان داده شده است.

۲-۲-۳-۲-۳ معماری بی اجزا مشترک

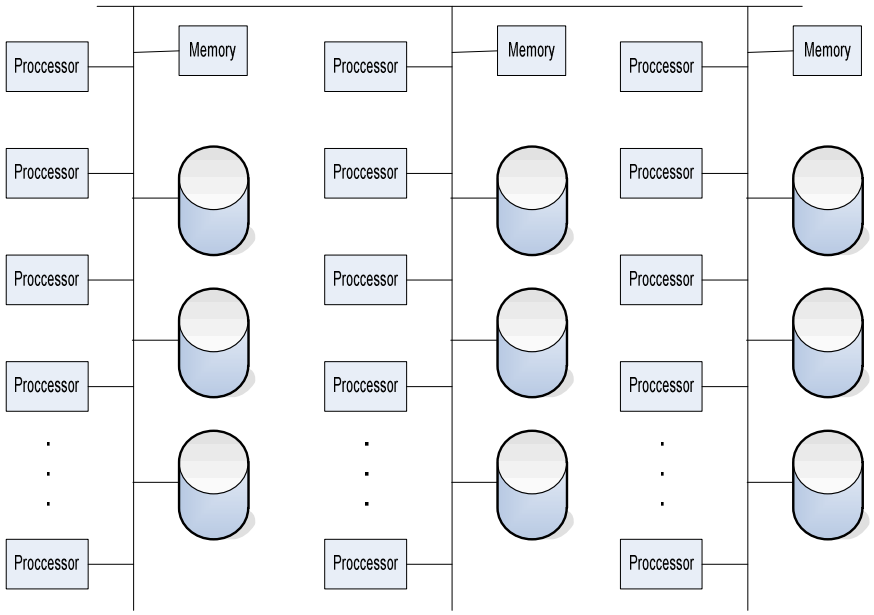
در این طرح، هر ماشین، پردازنده، حافظه و دیسک (های) خود را دارد. یک شبکه ارتباطی با سرعت بالا این ماشین‌ها را به هم مرتبط می‌کند. هر ماشین نوعی خدمت گزار پایگاهی است. در شکل ۱۰-۱۲، طرح کلی این معماری دیده می‌شود. مزیت این نوع معماری در تسهیل گسترش است. ولی عیب این نوع معماری در این است که هزینه ارتباط و دستیابی‌های غیر محلی زیاد است.



شکل ۱۰-۱۲ معماری بی اجزاء مشترک

۲-۲-۳-۲-٤ معماری سلسله مراتبی

در این طرح، ویژگی‌های سه طرح پیش با هم وجود دارد. در بالاترین سطح سیستم، تعدادی گره با شبکه ارتباطی به هم مرتبط‌اند و اجزا مشترک ندارند. پس در این سطح، معماری بی اشتراک داریم. هر گره خود می‌تواند تعداد کمی‌پردازنده با حافظه مشترک داشته باشد یا یک سیستم با دیسک‌های مشترک باشد. بنابرین یک سلسله مراتب از معماری‌های پیش گفته‌ایجاد می‌شود. طرح کلی این معماری در شکل ۱۰-۱۳ دیده می‌شود.



شکل ۱۰-۱۳ معماری سلسله مراتبی

۲-۲-٤ معماری چند پایگاهی

این معماری نوعی معماری نامتمرکز است که در آن، کامپیوترها خود مختاری کامل دارند. در سالهای اخیر گرایش زیادی به این معماری ایجاد شده است. در این معماری، معمولاً چند سیستم با معماری توزیع شده و یا چند سیستم با معماری متمرکز از قبل وجود دارد و هر سیستم، روی عملیات محلی خود کنترل

کامل دارد. اما برای آنکه کاربران بتوانند نیازهای اطلاعاتی خود را تامین کنند بی‌آنکه مستقیما با اجزاء معماری مرتبط باشند و در واقع بتوانند از یک سیستم چند پایگاهی بهره برداری کنند، به یک لایه نرم‌افزاری خاصی نیاز است. این لایه نرم‌افزاری امکان می‌دهد تا در هر سیستم، مدیر پایگاه‌داده‌ها روی داده‌های پایگاه خود کنترل کامل داشته باشد و نیازی به کنترل متمرکز نباشد و در عین حال خدمات لازم به کاربران داده شود.

این لایه نرم‌افزاری در سطح بالاتر از DBMSها و احیانا سیستم‌های فایلینگ موجود در پیکر بندی سیستم چند پایگاهی، قرار دارد. این لایه چنان عمل می‌کند که کاربران سیستم‌های مختلف، مجموعه پایگاه‌ها را به صورت یک پایگاه واحد می‌بینند. واضح است که در سیستم چند پایگاهی، یک شمای ادراکی جامع (سراسری) و در هر یک از DBMSها، یک شمای ادراکی محلی وجود دارد (و نیز شماهای دیگر، طبق آنچه که در معماری ANSI دیدیم. در چنین سیستمی‌طبعا به یک کاتالوگ جامع نیاز است. این معماری علاوه بر مشکلات تکنیکی موجود در یک سیستم توزیع شده، مشکلات خاص خود را نیز دارد که در اینجا از ذکر آنها خودداری می شود.

۲-۲-۵ سیستم پایگاهی همراه

۲-۲-۵-۱ تعریف

با رشد سریع تکنولوژی ارتباطات، اینک دیگر انسان می‌تواند از هر جا، به هر داده ذخیره شده در هر جای دیگر دستیابی داشته باشد، هر چند هنوز محدودیت‌هایی در این "داده فرستی- داده گیری" وجود دارد. اما تا آنجا که به حیطه دانش و تکنولوژی پایگاه‌داده‌ها مربوط می‌شود، نوع نوینی از سیستم پایگاه‌داده‌ها پدید آمده و در حال گسترش یافتن است: سیستم پایگاه‌داده‌های همراه (جابجا شونده). ما در قسمتهای قبل انواع معماری‌های سیستم پایگاهی را دیدیم، از جمله معماری مشتری-خدمتگزار. اما این معماری و حتی گونه گسترده تر آن، یعنی معماری پایگاه توزیع شده، هنوز مشکلاتی از جمله "عملی نبودن در همه جا"، ایمنی سیستم و داده‌ها، هزینه و... دارند. سیستم پایگاه‌های همراه راه حلی است برای برخی از این دشواری‌ها، به ویژه در کاربردهای شخصی.

۲-۲-۵-۲ معماری سیستم پایگاهی همراه

در این معماری، یک یا بیش از یک کامپیوتر متوسط یا بزرگ، نقش خدمتگزار پایگاهی را ایفا می‌کند. این کامپیوتر را خدمتگزار میزبان می‌نامند. هر کاربر، کامپیوتر کوچک همراه خود را دارد (با خود حمل می‌کند: نوعی دفترکار همراه). در این کامپیوتر داده‌های عملیاتی و برنامه‌های کاربردی مورد نیازش ذخیره شده‌اند. پایگاه‌داده‌های ذخیره شده در کامپیوتر خدمتگزار میزبان، حاوی داده‌های پویا است. بدین ترتیب کاربر می‌تواند از هر جا، با زیر سیستم خدمتگزار پایگاهی مرتبط شده و پردازش‌های مورد نظرش را انجام دهد. در این سیستم لزومی‌ندارد که پایگاه‌های همراه و پایگاه میزبان در یک شبکه باشند. ارتباط بین "همراه" و میزبان در بازه‌های زمانی نامنظم و معمولاً کوتاه برپا می‌گردد. بین کامپیوترهای همراه، مگر در شرایط خاص، چندان ارتباط برقرار نمی‌گردد.

دو جنبه اساسی این سیستم که باید در طراحی و پیاده سازی به آن توجه شود عبارتند از: مدیریت کارای پایگاههای همراه و ایجاد ارتباط پویا و سریع بین پایگاههای همراه و پایگاه میزبان.

تمرینات

۱. دو نوع اصلی معماری پایگاه‌داده‌ها را نام ببرید؟
۲. معماری متمرکز را شرح دهید ؟
۳. انواع معماری نامتمرکز را نام ببرید ؟
٤. سیستم پایگاهی مشتری-خدمت گذار را توضیح دهید ؟
٥. معماری سیستم پایگاهی توزیع شده را توضیح دهید ؟
٦. مزایای معماری سیستم پایگاهی توزیع شده را نام ببرید ؟
۷. چهار مدل معماری با پردازش موازی را نام ببرید ؟
۸. معماری پایگاه‌داده همراه را توضیح دهید ؟

ضمیمه ۱

مجموعه سئوالات

تست های سری ۱

۱- کدامیک از موارد زیر بیانگر اختلاف بین پایگاه‌داده‌ها و پایگاه بصیرت (Knowledge Base) می‌باشد ؟ (کارشناسی ارشد- آزاد۷۲)

۱. هر دو پایگاه از یک نوع بوده و فرقی با هم ندارند.
۲. پایگاه بصیرت دینامیک ولی پایگاه‌داده‌هاایستا (static) می‌باشد.
۳. پایگاه‌داده‌ها اغلب با زبان SQL و پایگاه بصیرت اغلب با زبان QUEL کد می‌شود.
٤. هیچکدام

۲- اطلاعات (Information) به چه معنی می‌باشد ؟ (مسابقات آموزشکده‌های فنی)

۱. داده‌هایی که روی آنها عمل مقایسه‌ای صورت گیرد.
۲. داده‌هایی که روی آنها عمل محاسبات صورت گیرد.
۳. داده‌هایی که به نمایش در آمده باشند.
٤. داده‌هایی که روی آنها عمل پردازش صورت گیرد.

۳- داده + ویژگی +موجودیت =....(مسابقات آموزشکده‌های فنی -۸۰)

۱. اطلاع
۲. پردازش
۳. پرونده

٤. دانش

٤- کدام گزینه در رابطه با نسل دوم ذخیره و بازیابی اطلاعات نادرست است ؟

۱. این نسل به نسل شیوه‌های دستیابی معروف است.

۲. تکرار ذخیره سازی دراین نسل بسیار کاهش یافته.

۳. تغییر در رسانه‌های ذخیره سازی بر روی برنامه‌های کاربردی تاثیر چندانی ندارد.

٤. دراین نسل از رسانه‌های با دستیابی مستقیم مثل دیسک استفاده شده است.

۵- کدام تعریف در رابطه با ((داده)) درست است ؟

۱. نمایش واقعیات، پدیده‌ها، مفاهیم یا معلومات به صورتی صوری و مناسب برای برقراری ارتباط، تفسیر یا پردازش توسط انسان یا امکانات خودکار

۲. از نظر ساختاری و از دیدگاه بانک اطلاعاتی، داده عبارت است از مقادیر صفات خاصه انواع موجودیتها

۳. هر دو گزینه ۱ و ۲

٤. اطلاعات کاراکتری که وارد سیستم دستی شده و به آن معنایی منتسب می‌شود.

٦- کدام عبارت درست است ؟

۱. داده همان اطلاعات است.

۲. داده اطلاعات پردازش شده است.

۳. اطلاعات، داده پردازش شده است.

٤. اطلاعات مقادیر خام است.

۷- Attribute عبارت است از.....

۱. همان موجودیت است.

۲. صفات خاصه توضیحی برای هر موجودیت است.

۳. همان رکورد است.

٤. سطرهای یک جدول است.

۸- File System چیست ؟

۱. همان DBMS است.

۲. به ساختار کلی نامگذاری، ذخیره سازی و سازماندهی فایلها در یک سیستم عامل گفته می‌شود.

۳. انباره‌ای برای یک مجموعه از فایلهای داده‌ای است.

٤. هر سه گزینه.

۹- کدام گزینه از ویژگی های نسل اول ذخیره و بازیابی اطلاعات نیست ؟

۱. نسخه‌های متعددی از یک فایل نگهداری می‌شود.

۲. ساختار فایل ها ترتیبی است.

۳. پردازش در محیط های بلادرنگ و Online می‌تواند انجام شود.

٤. طراحی ساختار فیزیکی فایلها هم، بر عهده کاربر است.

۱۰- کدام گزینه در رابطه با بانک های اطلاعاتی و بانکهای معرفت صحیح است ؟

۱. بانک معرفت حاوی تعداد زیادی واقعیت ساده است که به طور صریح بیان شده‌اند، همراه با تعداد کمی‌قواعد عام که به طور ضمنی بیان می‌شوند.

۲. بانک معرفت حاوی مجموعه‌ای از واقعیت های ساده و قواعد عام که به طور صریح بیان شده‌اند.

۳. بانک اطلاعاتی مجموعه‌ای است از تعداد زیادی واقعیت ساده که به طور صریح بیان شده‌اند.همراه با تعداد کمی‌قواعد عام که به طور ضمنی بیان می‌شوند.

٤. ۳و۲

۱۱- جزء داده‌ها (Data Element) چیست ؟ (مسابقات آموزشکده‌های فنی-۷٦)

۱. قسمتی از اطلاعات است.

۲. محلی موقت برای یک مقدار اطلاعات است.

۳. مقداری است که در فیلد ذخیره می‌شود.
٤. هر مقداری می‌تواند باشد.

پاسخ تست های سری ۱

۱- (۲) پایگاه بصیرت به تدریج بر بصیرتهایش (Knowledge Base) می‌افزاید.
۲- (٤) داده ورودی سیستم است که پس از پردازش تبدیل به اطلاعات شده و از سیستم خارج می‌شود.
۳- (۱)
٤- (۲) در نسل دوم تکرار ذخیره سازی هنوز در حد نسبتا بالایی وجود دارد.
۵- (۳) گزینه ٤ کامل نیست. داده هر نمایشی اعم از کاراکتری یا کمیتهای آنالوگ و یا غیره است.یعنی فقط به اطلاعات کاراکتری داده گفته نمی‌شود و داده می‌تواند وارد هر سیستمی‌شود و نه فقط سیستم دستی.
٦- (۳)
۷- (۲)
۸- (۲)
۹- (۳) گزینه ۳ از ویژگی های نسل دوم است.
۱۰- (٤)
۱۱- (۲)

تست های سری ۲

۱- شکل زیر در نمودار ER چه مفهومی‌را می‌رساند ؟

۱. هر استاد فقط و فقط یک درس (نه کمتر و نه بیشتر) و هر درس توسط یک استاد ارائه می‌شود.
۲. هر استاد یک درس و هر درس توسط یک استاد ارائه می‌شود. ممکن است استادی درسی ارائه نکند.
۳. هر استاد یک درس و هر درس توسط یک استاد ارائه می‌شود.
٤. یک استاد یک درس و یک درس توسط یک استاد ارائه می‌شود. ولی ممکن است استاد آن درس تغییر کند.

۲- در نمودار EER کدام گزینه درست است ؟

۱. صفت مشتق را با زیر خط، صفت کلیدی را با ۲ خط و صفت چند مقداری را با خط چین نشان می‌دهیم.
۲. صفت مشتق را با خط چین، صفت مرکب را با زیر خط و صفت وابسته را با ۲ خط نمایش می‌دهیم.
۳. صفت مشتق را با زیر خط، صفت چند مقداری را با ۲ خط و صفت کلیدی را با خط چین نشان می‌دهیم.
٤. صفت مشتق را با خط چین، صفت کلیدی را با زیر خط و صفت چند مقداری را با ۲ خط نشان می‌دهیم.

۳- به ارتباط بین موجودیتها.... گویند.

۱. Entity
۲. Attribute
۳. Tuple
٤. ER

٤-در مدل رابطه‌ای کدام نوع صفت را نداریم ؟

۱. صفت کلید
۲. صفت چند مقداری

۳. صفت مشتق

٤. هر سه را داریم

۵- کدام گزینه در رابطه با نمودار EER زیر صادق است ؟

۱. به محض حذف دانشجو از بانک اطلاعاتی، وابستگان او نیز حذف خواهند شد.

۲. وابستگان یک موجودیت قوی می‌باشند.

۳. موجودیت دانشجو، کلید اصلی موجودیت وابستگان را به ارث می‌برد.

٤. ۲ و ۱

٦- کدام گزینه نادرست است ؟

۱. در نمودار ER هر ارتباطی بین موجودیتها دارای سمانتیک خاصی است.

۲. صفت مشتق صفتی است که به کمک صفتهای دیگر می‌توان آن را محاسبه کرد.

۳. معنای نمودار ER زیر آن است که ارتباط استاد اختیاری است.

٤. هیچکدام

۷- شکل زیر نمایانگر چه نوع ارتباطی است ؟

۱. existence dependency
۲. is-a
۳. cardinality
٤. connectivity

۸- ارتباط بین موجودیتها کدام است ؟

۱. Attribute
۲. record
۳. ER
٤. entity

۹- منظور از Connection trap چیست ؟

۱. استنتاج نادرست از ارتباط بین موجودیتها

۲. ارتباطات دو طرفه نادرست

۳. ارتباط نادرست میان سه entity

٤. هر سه گزینه

۱۰- کدام موجودیت از نوع weak می‌باشد ؟

۱. استاد

۲. درس

۳. خانواده استاد

٤. دانشجو

۱۱- کدام مورد جزء انواع صفات خاصه نمی‌باشد ؟

۱. ذخیره شده یا مشتق

۲. تک مقداری یا چند مقداری

۳. ساده یا مرکب

٤. معرفه یا نکره

۱۲-کدام گزینه ویژگی موجودیت نمی‌باشد ؟

۱. یک نوع موجودیت معمولاً بیش از یک صفت دارد.

۲. موجودیت تک صفتی وجود ندارد.

۳. موجودیت تک نمونه‌ای وجود دارد.

٤. یک نوع موجودیت معمولاً نمونه‌های متمایز از یکدیگرند.

۱۳- کدام گزینه در رابطه با کلید (صفت شناسه) درست می‌باشد ؟

۱. می‌تواند یکتا نباشد.

۲. می‌تواند هیچ مقدار (NULL) باشد.

۳. می‌تواند مرکب باشد.

٤. می‌تواند مشتق باشد.

۱٤- در مسابقه بسکتبال ارتباط موجودیتهای گل و بازیکن چگونه است ؟

۱. ارتباط بازیکن با گل به صورت n: 1 می‌باشد.

۲. ارتباط بازیکن با گل به صورت n: n می‌باشد.

۳. ارتباط بازیکن با گل به صورت 1: 1 می‌باشد.
٤. نامشخص است.

پاسخ تست های سری ۲

۱- (۱) اگر علامت نقطه داخل مستطیل موجودیت باشد یعنی شرکت در ارتباط اجباری است.یعنی استاد ها نمی‌توانند درس ندهند.

۲- (٤)

۳- (٤)

٤- (۲) مثلا صفت مدرک برای استاد چند مقداری است چرا که ممکن است یک استاد چند مدرک داشته باشد.

۵- (۱) ممکن است وجود یک پدیده وابسته به وجود پدیده دیگری باشد، یعنی در صورت حذف عضوی از آن پدیده، عضوهای وابسته هم لازم باشد به طور خودکار حذف شوند.این نوع وابستگی را وابستگی وجودی و پدیده وابسته را موجودیت ضعیف weak entity می‌نامند. پدیده وابسته باید کلید پدیده اصلی را که به آن وابسته است به ارث ببرد. پدیده وابسته را با دو مستطیل تودرتو نمایش می‌دهیم.

٦- (۳) مشارکت یک نوع موجودیت در یک نوع ارتباط را الزامی(یا کامل) می‌گوئیم اگر تمام نمونه‌های آن نوع موجودیت در آن نوع ارتباط شرکت کنند. در غیراینصورت مشارکت غیر الزامی(اختیاری یا ناکامل) است.

اجباری_ الزامی_ کامل = Total participation

اختیاری _ غیر الزامی_ناکامل = Partial participation

در بعضی کتابها مشارکت الزامی‌را با دو خط موازی که موجودیت را به ارتباط متصل می‌کند نمایش می‌دهند. پس مشارکت استاد در تدریس الزامی‌است یعنی هر استاد حتما حداقل باید یک درس را تدریس کند.

۷- (۲) ارتباطاتی که از نماد مثلث (مفهوم ارث بری) استفاده می‌کنند. ارتباطات ((هست)) یا ((is-a)) نامیده می‌شوند.

۸- (۳)

۹- (۱)

۱۰- (۳) با حذف استاد اطلاعات خانواده او نیز حذف می‌گردد.

۱۱- (٤)

۱۲- (۲) موجودیت تک صفتی می‌تواند وجود داشته باشد مثلا جدولی که فقط یک ستون دارد.

۱۳- (۳) فیلد کلید نمی‌تواند مقدار نداشته باشد یعنی نمی‌تواند NULL باشد.

۱٤- (۱) یک بازیکن می‌تواند چند گل بزند ولی هر گل توسط یک بازیکن زده می‌شود.

تست های سری ۳

۱- مزایای یک پایگاه‌داده‌ها (Data Base) نسبت به فایلهای متعارف چیست ؟ (کارشناسی ارشد -آزاد ۷۲)

۱. ۲و٤
۲. کنترل حساب شده مقدار افزونگی در پایگاه‌داده‌ها
۳. اطمینان از صحت داده‌ها (Data Validation) کمتر مورد نیاز خواهد بود.
٤. دستیابی مشترک به داده‌ها

۲- کدامیک از موارد زیر جزء وظایف DBA نمی‌باشد ؟(کارشناسی ارشد-آزاد۷۲)

۱. نوشتن Data Dictionary برای پایگاه‌داده‌ها
۲. نظارت بر عملکرد پایگاه‌داده‌ها (Performance Monitoring)
۳. تهیه رویه‌ها و استراتژی تهیه Backup و نحوه احیای پایگاه‌داده‌ها
٤. تهیه schema برای پایگاه‌داده‌ها

۳- در یک سیستم مدیریت پایگاه‌داده‌ها (DBMS) کدامیک از امکانات زیر جزء عناصر اصلی تشکیل دهنده DBMS محسوب نمی‌شوند: (کارشناسی ارشد-دولتی۷۲)

۱. امکان پردازش زبان طبیعی برای کار با پایگاه
۲. امکان کار با داده‌ها به کمک یک DSL (Data Sublanguage)
۳. امکان تامین جمعیت و بی نقصی (integrity) پایگاه
٤. امکان تامین‌ایمنی پایگاه

٤- دو مرحله از مراحل طراحی یک بانک اطلاعاتی عبارتند از طراحی ادراکی (Conceptual Design) و طراحی منطقی (Logical Design) این دو چه تفاوت اساسی با هم دارند ؟ (کارشناسی ارشد-دولتی ۷۵)

۱. طراحی ادراکی به مدل خاصی مربوط می‌شود و پس از انتخاب مدل صورت می‌گیرد ولی طراحی منطقی به مدل خاصی بستگی ندارد.
۲. طراحی ادراکی مکمل طراحی منطقی است و پس از آن انجام می‌گیرد.

۳. طراحی منطقی به صورت کلی به سیستم می‌نگرد و با روشهایی مانند ER انجام می‌گیرد.
٤. طراحی منطقی به مدل خاصی مربوط می‌شود و پس از انتخاب مدل صورت می‌گیرد ولی طراحی ادراکی به مدل خاصی بستگی ندارد.

۵- فرق زبانهای پرس و جوی (Query) رویه‌ای (Procedural) و نارویه‌ای (Non-procedural) این است که:

۱. در زبانهای نارویه‌ای کاربر (user) چگونگی به عینیت (materalize) در آوردن دیدش را تصریح می‌کند و به طور ضمنی آنچه را که می‌خواهد، بیان می‌کند ولی در زبان های رویه‌ای بر عکس است.
۲. در زبان های نارویه‌ای کاربر فقط آنچه را که می‌خواهد تصریح می‌کند، در حالی که در زبان های رویه‌ای به علاوه باید چگونگی به عینیت در آوردن دیدش را نیز بیان نماید.
۳. اساساً با هم فرقی ندارند.
٤. در زبانهای رویه‌ای کاربر فقط آنچه را که می‌خواهد، تصریح می‌کند در حالیکه در زبانهای نارویه‌ای به علاوه باید چگونگی به عینیت در آوردن دیدش را نیز بیان نماید.

٦- کدامیک از موارد زیر از عناصر اصلی بانک اطلاعاتی است ؟ (مسابقات آموزشکده‌های فنی-۷٦)

۱. اشتراک منابع
۲. امنیت داده‌ها
۳. پروژه‌ها
٤. سخت‌افزار

۷- DBMS چیست ؟ (مسابقات آموزشکده‌های فنی-۷٦)

۱. رابطی است بین بانک اطلاعاتی و کاربر و نیز زبانی است برای تعریف بانک اطلاعاتی
۲. نرم‌افزار مدیریت و توصیف بانک اطلاعاتی و حفاظت و بازیابی داده‌هاست.
۳. همان بانک اطلاعاتی است.
٤. یک برنامه کاربردی است که در زبان اطلاعاتی داده‌ها را تعریف می‌کند.

۸- عناصر تشکیل دهنده محیط بانک اطلاعاتی کدام است ؟ (مسابقات آموزشکده‌های فنی-۷۷)

۱. موجودیت- صفت خاصه- داده- فایل
۲. سخت‌افزار- نرم‌افزار- فایل- داده
۳. نرم‌افزار-کاربر- داده-فایل
٤. سخت‌افزار- نرم‌افزار- کاربر- داده

۹- کدام یک از ویژگیهای یک بانک اطلاعاتی نیست ؟ (مسابقات آموزشکده‌های فنی)

۱- کنترل متمرکز
۲- کاهش میزان افزونگی
۳- پدیده ناسازگاری داده‌ها
٤- استفاده اشتراکی و همزمان داده‌ها

۱۰-DML چیست ؟ (مسابقات آموزشکده‌های فنی-۷۷)

۱- یک نوع بانک اطلاعاتی خاص است
۲- وظیفه تعریف فیلد ها را در بانک اطلاعاتی دارد
۳- دستورهایی است برای تعریف داده‌ها
٤- عهده دار اعمال هر گونه تغییر در بانک اطلاعاتی است

۱۱- کدامیک از موارد زیر در یک سیستم بانک اطلاعاتی جزو الزامات کارآیی است ؟ (مسابقات آموزشکده‌های فنی -۷۷)

۱- گزارش از کارکنان یک بخش سازمان

۲- ثبت نام بخش های یک سازمان
۳- محدودیت تعداد کارمندان یک بخش
٤- ثبت و گزارش مقدار ساعات اضافه کاری هر کارمند

۱۲- کدام تعریف برای DBMS کاملتر است ؟ (مسابقات آموزشکده‌های فنی۷۷ و ۸۰)
۱- نرم‌افزار سیستم مدیریت بانک اطلاعاتی جهت کار و انجام عملیات روی داده‌ها
۲- نرم‌افزاری برای ارتباط بین کاربران و فایلها در بانک اطلاعاتی
۳- سیستم عامل مخصوص بانکهای اطلاعاتی
٤- نرم‌افزاری برای مدیریت داده‌ها در بانک اطلاعاتی

۱۳-کدامیک تعریف دیکشنری داده‌ها (catalog) در بانک اطلاعاتی می‌باشد ؟ (مسابقات آموزشکده‌های فنی -۷۷ و ۸۰)
۱- یک بانک شامل مشخصات کاربران بانک اطلاعاتی
۲- یک بانک که مشخصات سایر بانکهای اطلاعاتی در آن وجود دارد
۳- یک بانک شامل مشخصات داده‌های بانک اطلاعاتی
٤- یک بانک شامل مشخصات صفات خاصه در یک بانک اطلاعاتی

۱٤- DDL چیست ؟ (مسابقات آموزشکده‌های فنی ۷۷)
۱- زبانی است برای نوشتن بانک اطلاعاتی
۲- زبانی است برای تعریف داده‌ها
۳- زبانی است برای کار در بانک اطلاعاتی
٤- زبانی است برای دستکاری داده‌ها

۱۵- کدامیک از موارد زیر از اجزای معماری سیستم بانک اطلاعاتی ارائه شده توسط ANSI نیست ؟ (مسابقات آموزشکده‌های فنی-۷۸)
۱- سیستم مدیریت بانک اطلاعات (DBMS)
۲- زبان فرعی داده‌ای (DSL)

۳- پیش کامپایلر (Precompiler)

٤- زبان میزبان (HL)

۱٦- در پایگاه‌داده‌ها، با کاهش افزونگی داده‌ها نتیجه حاصل کدام است ؟ (مسابقات آموزشکده‌های فنی -۷۸)

۱- استقلال داده‌ها

۲- امنیت داده‌ها (security)

۳- کاهش ناسازگاری داده‌ها (Inconsistency)

٤- اشتراک داده‌ها

۱۷- کدامیک از موارد زیر جزء وظایف DBMS محسوب نمی‌شود ؟ (مسابقات آموزشکده‌های فنی-۷۸)

۱- تبدیل احکام / شمای ادراکی به سطح داخلی و بالعکس

۲- تعیین و محاسبه طول بلاک، آدرس فیزیکی رکوردها واندازه بافر

۳- شناسایی کاربر و تشخیص اجازه دسترسی آن به پایگاه‌داده (DB)

٤- بررسی درخواست کاربر از نظر صحت گرامری (Syntax) آن

۱۸- یک سیستم بانک اطلاعاتی شامل چه مواردی می‌باشد ؟ (مسابقات آموزشکده‌های فنی -۷۸)

۱- داده‌های ذخیره شده- کاربران- DBMS و سخت‌افزار

۲- داده‌های ذخیره شده- DBMS و DBA

۳- داده‌های ذخیره شده- کاربران- AM

٤- داده‌های ذخیره شده- کاربران- AM و سخت‌افزار

۱۹-integrity به عنوان یکی از امتیازات database به چه معناست ؟ (مسابقات آموزشکده‌های فنی-۷۹)

۱- کاهش افزونگی داده‌ها و پایگاه‌داده‌ها

۲- صحت داده‌ها و پردازش ها و پیروی از قواعد سیستم

۳- محافظت داده‌ها در برابر خطرات

٤- پرهیز از ناهمخوانی داده‌ها در پایگاه‌داده‌ها

۲۰- طراح بانک، از داده‌های ذخیره شده در بانک چگونه دیدی در بالاترین سطح انتزاعی دارد ؟ (مسابقات آموزشکده‌های فنی -۸۰)

۱- ادراکی

۲- خارجی

۳- داخلی

٤- فیزیکی

۲۱- کدامیک از موارد زیر جزء ویژگی های بانک است ؟ (مسابقات آموزشکده‌های فنی -۸۰)

۱- امکان استفاده از بازیابی استنتاجی

۲- امکان پردازش دستورات کاربر به زبان طبیعی

۳- استفاده همزمان و اشتراکی از داده‌ها

٤- پدیده ناسازگاری داده‌ها

۲۲- کدام گزینه از اجزاء معماری سیستم بانک اطلاعاتی در ANSI نمی‌باشد ؟

۱- Mapping
۲- MSD
۳- HL
٤- DSL

۲۳- منظور از CASE در پایگاه‌داده‌ها چیست ؟

۱- ابزار کمکی در طراحی و پیاده سازی بانک اطلاعاتی

۲- حالات مختلف پیاده سازی بانک اطلاعاتی

۳- یکی از ساختارهای رابطه‌ای، شبکه‌ای یا سلسله مراتبی

٤- حالت خاصی که ممکن است بانک به حالت آنومالی برسد

۲٤- چهار کنترلی که باید روی تمامی‌تراکنش های بانک اطلاعاتی اعمال گردد تا صحت و جامعیت آن تضمین شود، کدامند ؟

۱- data dictionary -isolation-security atomicity-
۲- isolation -anomaly-integrity durability-
۳- atomicity -isolation-consistency durability-
٤- consistency -data dictionary-security anomaly-

۲۵- در کاتالوگ سیستم کدام گزینه وجود دارد ؟
۱- نام موجودیتها و ارتباطات بین آنها
۲- شمای ادراکی
۳- شماهای خارجی کاربران-ساختار فیزیکی بانک
٤- هر سه

۲٦-منظور از DML چیست ؟
۱- احکام کار با (پردازش) دادهها
۲- احکام کنترلی
۳- احکام تعریف دادهها
٤- هیچکدام

۲۷- تصویر ادراکی یعنی:
۱- دید DBA از کل بانک
۲- دید کاربر از کل بانک
۳- دید DBA از بخشی از بانک
٤- دید کاربر از بخشی از بانک

۲۸- کدام گزینه از معایب بانک اطلاعاتی نیست ؟ (آزمایشی- کاوشکران- ۸۰)
۱- در صورت عدم کنترل مناسب ممکن است جامعیت دادهها به خطر افتد.
۲- سربارهای کارایی ممکن است زیاد باشد.
۳- برنامه نویسی پیچیده می‌باشد.
٤- در صورت عدم کنترل مناسب ممکن است امنیت به خطر بیفتد.

۲۹- کدام گزینه در رابطه با تراکنش ها نادرست است ؟

۱- تراکنش ها اتمی‌هستند یعنی‌اینکه یا به طور کامل اجراء می‌شوند یا اصلاً اجراء نمی‌شوند حتی اگر سیستم در وسط کار خراب شود.
۲- تراکنش یک واحد منطقی از کار است و معمولاً شامل چندین عمل بانک اطلاعاتی است.
۳- عمل COMMIT پیام موفقیت انجام تراکنش را ارسال می‌کند.
٤- عمل ROLLBACK باعث می‌شود عملیات از اول دوباره تکرار شود.

۳۰- کدام گزینه از وظایف DDL است ؟
۱- ساختار داده‌ها
۲- تعریف فیلد ها
۳- محل فایلها
٤- همه گزینه‌ها

۳۱- مدیران داده (DA)وظیفه شان چیست ؟
۱- کنترل database
۲- کنترل DBMS و داده‌ها
۳- اجرای database
٤- استفاده از database

۳۲- کدام گزینه برای توضیح دیتابیس کاملتر است ؟
۱- بیان کننده فایلهای داده‌ای است
۲- بیان کننده جداول و صفات خاصه است
۳- یک نوع زبان پردازشی جهت مدیریت داده‌ای است
٤- بیان کننده موجودیتها، صفات خاصه و ارتباط بین موجودیتها است

۳۳- کدام گزینه تعریف کاملتری از DBMS را ارائه می‌کند ؟
۱- همان دیتابیس است.
۲- رابط بین دیتابیس و کاربر است.

۳- جهت تعریف و کار با دیتابیس استفاده می‌شود.

٤- ۲و۳

۳٤- منظور از DBMS Environment کدام است ؟

۱- Procedure-People -Data -Software -Hardware

۲- User -Software -Hardware

۳- Database -Software - Hardware

٤- Hardware-Software -People -Data

۳۵- DML چیست ؟

۱- زبانی جهت پرس و جوی داده‌ها

۲- زبانی جهت گزارش گیری

۳- ۲و۱

٤- زبانی جهت تعریف داده‌ها

۳٦- DDL چیست ؟

۱- برای انجام اعمال محاسباتی استفاده می‌شود.

۲- برای درج داده‌ها و به روز رسانی آنها استفاده می‌گردد.

۳- صفات مشخصه و خصوصیات دیتابیس را تعریف می‌کند.

٤- همان DML می‌باشد.

۳۷- کدام گزینه نادرست است ؟

۱- یکی از مزایای DBMS ایجاد محدودیتهای امنیتی است.

۲- Database مجموعه‌ای از چند رکورد است.

۳- External view دید طراح بانک است.

٤- End user افرادی هستند که از دیتابیس استفاده می‌کنند.

۳۸-در مدل ANSI / SPARC کدام دید مربوط به داده‌ها و نگرشی بر محدودیتهای سیستم است ؟

۱- External Level

۲- Conceptual level

۳- Internal
۴- 1,3

۳۹- کدام گزینه توسط سطح ادراکی ارائه می‌گردد ؟

۱- مشخص ساختن داده‌ها
۲- تعریف دیتابیس
۳- تعریف سطرهای جداول
۴- مشخص ساختن رسانه ذخیره سازی

۴۰- DBA کیست ؟

۱- برنامه نویس دیتابیس
۲- پیاده سازی تصمیمات اداره کننده داده‌ها (DA)
۳- اجراء کننده دیتابیس
۴- طراح دیتابیس

۴۱- کدام گزینه نادرست است ؟

۱- در استقلال فیزیکی داده‌ها اگر تغییری در ذخیره سازی داده‌ها انجام گیرد، برنامه‌های کاربردی هیچ تغییری نمی‌کنند.
۲- در استقلال منطقی داده‌ها تغییر تصویر ادراکی بانک از دید کاربران و برنامه‌های آنها مخفی می‌ماند.
۳- CASE یک ابزار کمکی است که در طراحی و پیاده سازی بانک مورد استفاده قرار می‌گیرد.
۴- ناسازگاری و ناایمن بودن داده‌ها، جامعیت بانک را خدشه دار می‌کند ولی گسست پیوندهای سمانتیک برای جامعیت مشکلی‌ایجاد نمی‌کند.

۴۲- در پایگاه‌داده‌ها با کاهش افزونگی، نتیجه حاصل کدام است ؟

۱- استقلال داده‌ها
۲- security
۳- کاهش Inconsistency
۴- اشتراک داده‌ها

٤٣- کدام گزینه نادرست است ؟

۱- یک سیستم بانک اطلاعاتی شامل موارد روبرو است: داده‌های ذخیره شده، کاربران، DBMS، سخت‌افزار

۲- DBMS زبانی است برای تعریف بانک اطلاعاتی

۳- DBMS نرم‌افزار سیستم مدیریت بانک اطلاعاتی جهت کار و انجام عملیات روی داده‌ها می‌باشد.

٤- DBMS میهمان سیستم عامل است.

٤٤- در کدامیک از انواع ارتباط بین DBMS با سایر عناصر نرم‌افزاری محیط، امکان اشتراکی
شدن داده‌ها برای برنامه‌ها وجود ندارد ؟

۱- یک DBMS برای هر برنامه کاربردی وجود داشته باشد.

۲- یک DBMS تحت کنترل سیستم عامل وجود داشته باشد.

۳- برنامه‌ها تحت کنترل DBMS اجراء شوند.

٤- هیچکدام.

٤٥- کدامیک از موارد زیر از وظلیف DBMS نمی‌باشد ؟

۱- برقراری امنیت

۲- تامین امکان تعریف وایجاد بانک

۳- بازیابی و بهنگام سازی بانک

٤- تامین امکان کنترل جامعیت بانک

٤٦- DSL چه نوع زبانی است ؟

۱- Procedural
۲- Declarative
۳- OOP
٤- Non-structured

٤٧- الگوی ANSI / SPARC چیست ؟

۱- رابطه بین موجودیتها را بیان می‌کند.

۲- اجتماعی از دیدگاههای مجزای کاربران است.
۳- اجتماعی از دیدگاههای انبارشی و اجتماع دیدگاه کاربران و دیدگاههای مجزای کاربران است.
٤- اجتماعی از دیدگاههای انبارشی و مجزای کاربران است.

٤۸- مجتمع بودن عبارت است از:
۱- جمع پذیری نرم‌افزارهای بانک اطلاعاتی
۲- وحدت چندین فایل اطلاعاتی مجزا با امکان تکرار تکنیکی اطلاعات
۳- وحدت چندین فایل اطلاعاتی مجزا بودن هرگونه تکراری
٤- وحدت چندین رکورد حذف شده

٤۹- schema یک بانک اطلاعاتی عبارت است از:
۱- تشریح DBMS
۲- تشریح دیتابیس برای DBMS
۳- تشریح داده‌ها
٤- معرفی کاربران

۵۰- منظور از View در مدل ANSI چیست ؟
۱- داده‌های دیده شده در بانکهای اطلاعاتی
۲- نمائی از اطلاعات مورد نیاز طراح
۳- مشاهده داده‌ها به همان شکل که در هر سطح مشاهده می‌شود.
٤- نمائی از داده‌های مورد نیاز در DBMS

۵۱- دید داخلی عبارت است از:
۱- شمای مربوط به استفاده از سیستم مدیریت بانک اطلاعاتی
۲- شمای نحوه ذخیره داده‌ها در بانک اطلاعاتی
۳- شمای مربوط به کاربران
٤- شمای مربوط به فایلها، سکتورها و شیارها

۵۲- کدام گزینه در رابطه با فرهنگ داده‌ها درست است ؟

۱- DBMS امکانات فرهنگ داده‌ها را فراهم می‌سازد و یک بانک اطلاعاتی برای سیستم است.

۲- DBMS امکانات فرهنگ داده‌ها را فراهم می‌سازد و یک بانک اطلاعاتی برای کاربران است.

۳- فرهنگ داده‌ها توضیحاتی در مورد جداول به کاربر می‌دهد.

٤- فرهنگ داده‌ها یکی از امکاناتی است که DBA در اختیار دیتابیس قرار می‌دهد.

۵۳- کدام گزینه نادرست است ؟

۱- در اکثر نرم‌افزارهای پایگاه‌داده‌ها برای ذخیره سازی اطلاعات یک entity از جدول استفاده می‌شود.

۲- END USER نگهدارندگان محیط DBMS می‌باشند.

۳- DBMS امکانات فرهنگ داده‌ها را فراهم ساخته و یک بانک اطلاعاتی برای سیستم می‌باشد.

٤- از مزایای پایگاه‌داده نسبت به فایلهای معمولی می‌توان کنترل حساب شده مقدار افزونگی واشتراک داده‌ها را نام برد.

٥٤-کدام گزینه درست است ؟

۱- مجتمع بودن یعنی وحدت چندین فایل اطلاعاتی مجزا بودن هر گونه تکراری

۲- کوچکترین واحد داده‌های مفهومی‌را ((داده‌های تجزیه پذیر)) گویند.

۳- دیتابیس مجموعه‌ای از رکوردهای تکراری است.

٤- محدودیتهای جامعیتی، شمای هر رابطه و اطلاعات امنیتی توسط DDL تعیین می‌گردد.

۵۵- کدام گزینه درست است ؟

۱- سیستم مدیریت پایگاه اطلاعاتی از محل فیزیکی رکوردها بر روی رسانه ذخیره سازی خبر دارد.

۲- Pre-compiler در بانکهای اطلاعاتی دستورهای بانکی را به دستورهای زبان برنامه نویسی تبدیل می‌کند.

۳- کلیه اطلاعات سیستمی‌از دید خارجی، ادراکی و داخلی درون DBMS قرار دارد.

٤- کاتالوگ سیستم فقط با دستورات DML قابل تغییر است.

٥٦-کدام گزینه نادرست است ؟

۱. External view سطح مشاهدات مجزای کاربران را بیان می‌کند.
۲. host language یک زبان بانک اطلاعاتی است.
۳. نگاشت خارجی / مفهومی‌کلید استقلال منطقی داده‌هاست.
٤. در استقلال فیزیکی طراح در تغییرات سطح داخلی / ادراکی آزاد است.

۵۷- استقلال منطقی شامل کدام مورد نمی‌باشد ؟

۱. تعریف یک رابطه جدید در شما
۲. تبدیل یک رابطه به دو رابطه کوچکتر
۳. جایگزینی یک رابطه از شما
٤. ۱و۲

۵۸- خاصیت isolation در تراکنش ها به چند مفهوم می‌باشد ؟

۱. هر تراکنش اگر به تنهایی اجراء شود بانک را از حالتی صحیح به حالت صحیح دیگری منتقل می‌سازد.
۲. اثر تراکنش های هم روند روی یکدیگر چنان است که انگار هر کدام در انزوا انجام می‌شود.
۳. کنترل isolation توسط واحد recovery management صورت می‌گیرد.
٤. تراکنش ها ممکن است اثر مخرب بر روی هم داشته باشند.

۵۹- دو نوع پایان برای تراکنش ها عبارتند از:

۱. Abort , Commit
۲. Run , ready
۳. Abort , Retry

٤. Commit , Success

٦٠- همخوانی یا سازگاری (Consistency) برای تراکنش ها به چه معناست ؟

١. هر تراکنش اگر به تنهایی اجراء شود بانک اطلاعاتی را از حالتی صحیح به حالت صحیح دیگر منتقل می‌کند.

٢. در تراکنش باید تمامی‌قوانین جامعیت بانک اطلاعاتی را رعایت کند.

٣. ١و٢

٤. هیچکدام

٦١- خاصیت اتمی‌بودن برای تراکنش ها به چه مفهوم می‌باشد ؟

١. تمام دستورات یک تراکنش یا باید اجراء شود و یا هیچکدام از آنها نباید اجراء شود.

٢. تراکنش می‌تواند وسط کار لغو گردد و نیازی به ترمیم داده‌ها دراینحالت نیست.

٣. تراکنش باید کامل اجراء شود و هیچگاه نمی‌تواند وسط کار لغو گردد.

٤. تراکنش در کل بانک باید صورت گرفته و نیازی به لغو کردن آن نیست.

٦٢- کدام اعمال توسط واحد Recovery management صورت می‌گیرد ؟

١. یکپارچگی

٢. پایائی

٣. همخوانی

٤. ١و٢

٦٣- خاصیت پایائی (durability) در تراکنشها به چه معناست ؟

١. تراکنشهائی که به مرحله Commit رسیده‌اند به صورت اتفاقی حذف نمی‌شوند.

٢. تراکنشهائی که به مرحله Commit رسیده‌اند اثرشان ماندنی می‌باشد.

٣. تراکنشهائی که به مرحله Commit رسیده‌اند حتی در صورت زلزله یا آتش سوزی نبایستی از بین بروند.

٤. هر سه گزینه.

٦٤-مشخصات اصلی یک تراکنش نسبت به یک برنامه معمولی (در محیط غیر بانکی) کدام است ؟

١. تراکنش نمی‌تواند به تعویق بیافتد.
٢. تراکنش به DBMS وابسته نیست.
٣. هدف از کنترل تراکنش ها بالابردن سرعت است.
٤. تراکنش توسط DBMS کنترل می‌شود.

٦٥- در محیط پایگاه‌داده کدام نرم‌افزار وجود ندارد ؟

١. نرم‌افزار DBMS
٢. رویه‌های ذخیره شده
٣. نرم‌افزار سیستم عامل
٤. نرم‌افزار برنامه‌های کاربردی

٦٦- درون کاتالوگ سیستم چه اطلاعاتی ذخیره نمی‌گردد ؟

١. لغت نامه داده‌ها
٢. مقادیر داده‌های درون جداول
٣. تعداد نسخه‌های هر فایل و ترتیب زمانی آنها
٤. حق دستیابی افراد به داده‌های مختلف

٦٧- در رابطه با دید داخلی کدام گزینه درست می‌باشد ؟

١. این دید مبتنی بر یک یا بیش لز یک ساختار فایل است.
٢. این دید در سطح فیزیکی فایلینگ مطرح می‌گردد.
٣. این دید به طراحی منطقی معروف است.
٤. هر سه گزینه

٦٨- کدام گزینه درست است ؟

١. زبان DSL می‌تواند مستقل از HL نباشد.

۲. زبان DSL می‌تواند مستقل از HL باشد.
۳. زبان HL یکی از زبانهای برنامه نویسی متعارف است.
٤. هر سه گزینه

٦۹- گزینه صحیح کدام است ؟
۱. یک کاربر می‌تواند چند دید داشته باشد.
۲. چند کاربر می‌توانند یک دید مشترک داشته باشند.
۳. مجموعه دیدهای کاربران را سطح خارجی می‌نامند.
٤. هر سه گزینه

۷۰- چه نوع سخت‌افزاری در پایگاه‌داده‌ها وجود دارد ؟
۱. ذخیره سازی
۲. ارتباطی
۳. پردازشی
٤. هر سه گزینه

پاسخ تست های سری ۳

۱- (۱) در بانک اطلاعاتی اطمینان از صحت داده‌ها بسیار مهم است. کاهش افزونگی و دستیابی مشترک از مزایای بانک اطلاعاتی است.

۲- (۱) Data Dictionary توسط خودDBMS ساخته و به روز در می‌آید. (به صورت خودکار)

۳- (۱)

٤- (۱) در طراحی منطقی مدل مورد استفاده (رابطه‌ای- شبکه‌ای- سلسله مراتبی) مهم نیست و شمای کلی مثلا با نمودار ER ترسیم می‌شود. اما طراحی ادراکی بستگی به مدل مورد استفاده دارد.

۵- (۲) در زبانهای غیر رویه‌ای کاربر فقط می‌گوید چه می‌خواهد ولی شیوه و الگوریتم انجام آن کار را بیان نمی‌کند. در زبانهای رویه‌ای کاربر باید الگوریتم انجام کارش را نیز بیان کند.

٦- (٤) اشتراک منابع و امنیت داده‌ها از مزایای بانک اطلاعاتی هستند. عناصر اصلی بانک اطلاعاتی عبارتند از: نرم‌افزار- سخت‌افزار- داده‌ها و کاربر

۷- (۲)

۸- (٤)

۹- (۳) برعکس در پایگاه‌داده‌ها ناسازگاری داده‌ها نباید وجود داشته باشد.

۱۰- (٤)DML بخشی از زبان DSL است و به دستورات کار با داده‌ها (Data Manipulation Language = DML) گفته می‌شود.

۱۱- (۳) برای آنکه کارایی را ثابت نگه داریم باید تعداد کارمندان را محدود کنیم.

۱۲- (۱)

۱۳- (۳) به کاتالوگ فراداده (Meta data) یا داده‌هایی در مورد داده‌ها (Data about data) نیز گفته می‌شود.

۱٤- (۲) DDL بخشی از زبان DSL است و به دستورات تعریف داده‌ها (Data Definition Language = DDL) گفته می‌شود.

۱۵- (۳) اجزاء معماری ANSI عبارت است از: دید داخلی- دید خارجی- دید ادراکی-تبدیلات بین سطوح- زبان فرعی داده‌ای (DSL)- زبان میزبان (HL) و سیستم مدیریت بانک اطلاعاتی (DBMS)

۱٦- (۳) با کاهش افزونگی داده‌ها، داده مشابه کمتری وجود دارد و لذا احتمال ناسازگاری‌این داده‌ها کاهش می‌یابد. مثلا اگر آدرس فردی تغییر کند باید تمامی‌جداولی که آدرس آن شخص را دارند تغییر کند. بدیهی است هر چقدر تعدادین جداول بیشتر باشد تغییر سخت تر انجام می‌گیرد و احتمال‌ایجاد ناسازگاری بیشتر می‌شود.

۱۷- (۲)

۱۸- (۱) عناصر اصلی سیستم بانک اطلاعاتی عبارت است از: سخت‌افزار- نرم‌افزار- داده‌ها و کاربران

۱۹- (۲)

۲۰- (۱)

۲۱- (۳)

۲۲- (۲)

۲۳- (۱) CASE به معنای Computer Aided Software Engineering می‌باشد.

۲٤- (۳)

۲۵- (٤)

۲٦- (۱)

۲۷- (۱)

۲۸- (۳) در بانک اطلاعاتی برنامه نویسی ساده تر است.

۲۹- (٤) عمل ROLLBACK عدم موفقیت اجرای تراکنش را گزارش می‌کند و به مدیر تراکنش می‌گوید که اشکالی پیش آمده است. بانک اطلاعاتی ممکن است در حالت ناسازگار باشد و تمام بهنگام سازی ها که توسط آن واحد کاری انجام شده است باید لغو یا رد شود.

۳۰- (٤)

۳۱- (۲)

۳۲- (٤)

۳۳- (٤)

۳٤- (۱)

۳۵- (۳)

۳٦- (۳)

۳۷- (۳) دید طراح بانک Conceptual view می‌باشد. دید خارجی سطح مشاهدات مجزای استفاده کنندگان را بیان می‌کند.

۳۸-(۲)

۳۹- (۱)

٤۰- (۲)

٤۱- (٤) عدم وجود ارتباطات بین موجودیتها (گسست پیوندهای سمانتیک) نیز باعث خدشه دار شدن جامعیت بانک می‌شود.

٤۲- (۳) (ناسازگاری داده‌ها = Inconsistency) با کاهش افزونگی داده‌ها، داده ی مشابه کمتری وجود دارد و لذا احتمال ناسازگاری‌این داده‌ها کاهش می‌یابد.

٤۳- (۲)

٤٤- (۱)

٤٥- (۱) برقراری امنیت بر عهده DBA می‌باشد.

٤٦- (۲) در یک زبان بیانی (declarative یا Non-procedural) کاربر می‌گوید چه می‌خواهد ولی رویه انجام کار را بیان نمی‌کند. ولی مثلا در C و پاسکال که رویه‌ای (procedural) هستند کاربر باید رویه و الگوریتم کار را نیز بیان کند.

٤۷- (۳)

٤۸- (۲)

٤۹- (۲)

٥۰- (۳)

٥۱- (۲)

٥۲- (۱)

٥۳- (۲) END USER افرادی هستند که از دیتابیس استفاده می‌کنند.

٥٤- (٤) گزینه ۱: مجتمع بودن یعنی وحدت چندین فایل اطلاعاتی مجزا با هر گروه تکراری

گزینه ۲: کوچکترین واحد داده‌های مفهومی‌را مقادیر غیر قابل تجزیه گویند.
گزینه ۳: دیتابیس مجموعه‌ای از چند رکورد است.
۵۵- (۳) گزینه ۱: DBMS از نوع فیلدها و ساختار فایلها خبر دارد ولی از محل فیزیکی رکوردها بر روی هارد دیسک خبر ندارد.
گزینه ۲: Pre-compiler دستورهای زبان برنامه نویسی را به دستورات بانکی تبدیل می‌کند.
گزینه ٤: کاتالوگ سیستم با برخی احکام DML و برخی احکام DDL قابل تغییر است.
٥٦- (۲) host language یک زبان برنامه نویسی است که امکان استفاده و ارتباط با بانک اطلاعاتی را فراهم می‌سازد.
(۳) -٥٧
٥٨- (۲) گزینه ۱ خاصیت consistency است. کنترل isolation توسط واحد concurrency control انجام می‌گیرد. بر طبق خاصیت انزوا همروندی تراکنش ها باید کنترل شود تا اثر مخرب بر روی هم نداشته باشند.
(۱) -٥٩
(۳) -٦۰
(۱) -٦۱
(٤) -٦۲
(٤) -٦۳
(٤) -٦٤
(۳) -٦٥
(۲) -٦٦
٦۷- (۱)این دید در سطح منطقی فایلینگ مطرح شده و به طراحی فیزیکی معروف است.
(٤) -٦۸

۶۹- (٤)

۷۰- (٤)

تست های سری ٤

۱- در کدام مدل پایگاه‌داده‌ها، مجموعه‌ای مرتب از درختها داریم که در آن فرزندی بدون پدر وجود ندارد ؟ (مسابقات آموزشکده‌های فنی-۸۰)

۱. رابطه‌ای
۲. سلسله مراتبی
۳. شبکه‌ای
٤. لیست معکوس

۲- کدامیک از موارد زیر، جزو ویژگیهای ساختار داده‌ای مدل شبکه‌ای است ؟ (مسابقات آموزشکده‌های فنی-۷۹)

۱. اصل وحدت عملگر، در یک عمل واحد رعایت می‌شود.
۲. از دید کاربر دارای وضوح است.
۳. برای محیط های دارای ارتباطات یک به چند دوسویه، مدل مناسبی نمی‌باشد.
٤. در عملیات ذخیره سازی آنومالی ندارد.

۳- کدامیک از موارد زیر جزء مدلهای پایگاه اطلاعاتی نیستند ؟(مسابقات آموزشکده‌های فنی و حرفه‌ای- ۷۸)

۱. مدل رابطه‌ای
۲. مدل سلسله مراتبی
۳. مدل اطلاعاتی
٤. مدل شبکه‌ای

٤- شکل زیر از کدام مدل برگرفته شده است ؟ (مسابقات آموزشکده‌های فنی و حرفه‌ای- ۷۷)

۱- رابطه‌ای
۲- سلسله مراتبی
۳- شبکه‌ای
٤- بانکی

۵- کدام گزینه نادرست است ؟ (مسابقات آموزشکده‌های فنی-۷۷)

۱- در مدل رابطه‌ای بانک داده‌ها از دید کاربر از تعدادی جدول تشکیل شده است.

۲- در مدل سلسله مراتبی مسیر منطقی همیشه از ریشه و از بالا به پایین است.

۳- در مدل شبکه‌ای هر گره فرزند می‌تواند دارای بیش از یک پدر باشد.

٤- در مدل شبکه‌ای از مفاهیم ریاضی مجموعه‌ها برای نمایش داده‌ها و ارتباط آنها استفاده می‌شود.

٦- کدامیک از موسسات زیر روی استانداردسازی زبانهای پایگاه‌داده‌ها کار نکرده‌اند ؟ (کارشناسی ارشد- آزاد ۷۲)

۱- CODASYL DBTG
۲- ISO
۳- ANSI/SPARC
٤- 2,3

۷- کدامیک از موارد زیر بیانگر فرق بین یک مدل شبکه‌ای داده (Network) و یک مدل Entity Relationship(ER) می‌باشد ؟ (کارشناسی ارشد- آزاد ۷۲)

۱- در مدل شبکه‌ای هرEntity فقط یک مولد (parent) دارد در مدل ER هر Entity می‌تواند بیشتر از یک مولد داشته باشد.

۲- در مدل ER روابط M:M پشتیبانی می‌شود در مدل شبکه‌ای خیر.

۳- در مدل ER یک Entity می‌تواند از نوع ضعیف (weak)باشد در رابطه شبکه کلید Entity از نوع قوی (strong) می‌باشد.

٤- ۳و۲

۸- می‌خواهیم برای یک دانشکده یک بانک اطلاعاتی آموزشی طراحی کنیم. از بین موجودیتهای اساسی، موجودیت درس را در نظر می‌گیریم و ارتباط زیر را:
در مدل شبکه‌ای برای نشان دادن ارتباط فوق در سطح ادراکی:

۱- حداقل دو مجموعه به معنایی که در CODASYL مطرح است، لازم است.

۲- در نظر گرفتن یک مجموعه به معنایی که در CODASYL مطرح است، کافی است.

۳- بیش از دو مجموعه به معنایی که در CODASYL مطرح است لازم نیست.

٤- در نظر گرفتن یک مجموعه به معنایی که در CODASYL مطرح است کافی است ولی بهتر است طراحی را با دو مجموعه انجام داد.

۹- کدام گزینه در رابطه با مدل سلسله مراتبی درست نمی‌باشد ؟

۱- هر گره پدر می‌تواند چندین فرزند داشته باشد ولی هر فرزند فقط یک پدر دارد.

۲- این مدل در عملیات حذف و بهنگام سازی آنومالی دارد.

۳- اگر پرس و جوئی را قرینه کنیم همواره رویه پاسخگویی نیز به همان صورت قرینه می‌شود.

٤- این مدل در عملیات درج دارای آنومالی است.

۱۰- کدام گزینه در رابطه با مدل شبکه‌ای درست نمی‌باشد ؟

۱- دراین ساختار یک نمونه رکورد عضو می‌تواند عضو دو مالک متمایز باشد.

۲- عمل بازیابی ساده تر از مدل سلسله مراتبی است.

۳- این مدل در عملیات درج و حذف آنومالی ندارد.

٤- این مدل در عملیات بهنگام سازی آنومالی ندارد.

۱۱- کدام گزینه بیانگر تصویر ادراکی عام است ؟

۱- NIAM , EER

۲- شبکه‌ای- سلسله مراتبی- جدولی

۳- رابطه- جدول

٤- معماری ANSI

۱۲- بانک اطلاعاتی رابطه‌ای عبارت است از:

۱- بانکی است که در آن هر گونه تکرار همراه با جداول دیده می‌شود.

۲- بانکی است که در مقادیرش قابل تجزیه‌اند.

۳- بانک اطلاعاتی است که به عنوان مجموعه‌ای از روابط و جداول در نظر گرفته می‌شود.

٤- هیچکدام.

۱۳- کدام گزینه درست است ؟

۱- سیستم رابطه‌ای دارای ساختار فیزیکی است.

۲- سیستم رابطه‌ای دارای ساختار منطقی است.

۳- سیستم رابطه‌ای دارای ساختار منطقی و فیزیکی است.

٤- هیچکدام.

۱٤- کدام گزینه جزو مدلهای داده‌ای است ؟

۱- Entity Relation Ship
۲- Record- based data Model
۳- Networking Model
٤- Semantic Model

۱٥- کدام عبارت نادرست است ؟

۱- برای مدلینگ ارتباطات یک به چند یکسویه بین انواع موجودیتها ساختار سلسله مراتبی مناسبتر است.

۲- در ساختار سلسله مراتبی، آنومالی در عملیات ذخیره سازی وجود دارد.

۳- زمان پاسخگویی به پرس و جوها در مدل رابطه‌ای بسیار کم است.

٤- ساختار سلسله مراتبی دارای یک رویه پاسخگویی واحد برای پرس و جوهای قرینه نمی‌باشد.

۱٦- کدام گزینه نادرست است ؟

۱- اصل وحدت عملگر، در یک عمل واحد (مثلا درج) در مدل شبکه‌ای رعایت می‌شود.

۲- در مدل سلسله مراتبی مسیر منطقی همیشه از ریشه و از بالا به پایین است.

۳- در مدل رابطه‌ای بانک داده‌ها از دید کاربر از تعدادی جدول تشکیل شده است.

٤- در مدل شبکه‌ای هر فرزند می‌تواند دارای بیش از یک پدر باشد.

۱۷- کدام گزینه در رابطه با خصوصیات مدل سلسله مراتبی نادرست است ؟

۱- خاص محیط هایی است که در آنها ارتباط های یک به چند یک سویه وجود دارد.

۲- قدیمی‌ترین ساختار داده یی در سطح انتزاعی برای طراحی بانک است.

۳- در عملیات ذخیره سازی آنومالی دارد.

٤- برای پاسخگوئی به پرس و جوهای قرینه رویه‌های پاسخگویی قرینه دارد.

۱۸- کدام گزینه در رابطه با خصوصیات مدل شبکه‌ای نادرست است ؟

۱- عملگر بازیابی خاصیت تقارن دارد.

۲- در عملیات ذخیره سازی آنومالی دارد.

۳- از مفهوم ریاضی مجموعه ی کوداسیلی استفاده میکند.

٤- برای محیط های دارای ارتباط یک به چند دوسویه مدل مناسبی است.

۱۹- دو مرحله از مراحل طراحی یک بانک اطلاعاتی عبارتند از طراحی ادراکی Conceptual Design و طراحی منطقیLogical Designاین دو چه تفاوت اساسی با هم دارند ؟ (کارشناسی ارشد- دولتی ۷۵)

۱- طراحی ادراکی به مدل خاصی مربوط می‌شود و پس از انتخاب مدل صورت می‌گیرد ولی طراحی منطقی به مدل خاصی بستگی ندارد.

۲- طراحی ادراکی مکمل طراحی منطقی است و پس از آن انجام می‌گیرد.

۳- طراحی منطقی به صورت کلی به سیستم می‌نگرد و با روشهایی مانند ER انجام می‌گیرد.

٤- طراحی منطقی به مدل خاصی مربوط می‌شود و پس از انتخاب مدل صورت می‌گیرد ولی طراحی ادراکی به مدل خاصی بستگی ندارد.

۲۰- جهت پیاده سازی یک بانک اطلاعاتی کدام فعالیت در اولویت قرار دارد ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱- طراحی مدل داده‌ها به صورت منطقی
۲- طراحی مدل داده‌ها به صورت فیزیکی
۳- طراحی مدل داده‌ها به صورت فیزیکی و منطقی
٤- هیچکدام

۲۱- چرا مدل سلسله مراتبی آنومالی دارد ؟
۱- چون در هنگام اصلاًح عمل بهنگام سازی باید منتشر شونده باشد.
۲- چون هنگام درج باید رکورد پدر را داشته باشیم تا بتوانیم فرزند را درج کنیم.
۳- چون در عمل حذف، ممکن است اطلاعات ناخواسته دیگری نیز حذف گردد.
٤- هر سه گزینه

پاسخ تست های سری ٤

۱- (۲)

۲- (٤) مدل شبکه‌ای در عملیات درج، حذف و بهنگام سازی آنومالیهای مدل سلسله مراتبی را ندارد.این مدل از دید کاربر واضح نیست. مدل شبکه‌ای برای نمایش ارتباطات دوسویه مناسب می‌باشد.

۳- (۳)

٤- (۲) مدل سلسله مراتبی مشابه شکل یک درخت می‌باشد.

۵- (٤) مفاهیم مجموعه در مدل رابطه‌ای استفاده می‌گردد.

٦- (۲)

۷- (٤) ممکن است وجود یک پدیده وابسته به وجود پدیده‌ای دیگر باشد. مثلا به محض حذف دانشجو از بانک دانشگاه (بر اثر فارغ التحصیلی یا اخراج شدن) وابستگان اونیز (مثل همسر و فرزند)از سیستم کمک هزینه باید حذف شوند. دراین حال پدیده وابسته را موجودیت ضعیف (weak entity) می‌نامند.

۸- (۲)

۹- (۳) منظور از آنومالی (anomaly) یعنی وجود دشواری در انجام یک عمل خاص و یا عدم امکان انجام عمل و یا بروز عوارض نامطلوب در پی انجام عمل.

۱۰- (۲) عمل بازیابی در مدل شبکه‌ای پیچیده تر از مدل سلسله مراتبی می‌باشد.

۱۱- (۱)

۱۲- (۳)

۱۳- (۲)

۱٤- (۲)

۱۵- (۳)

۱٦- (۱) اصل وحدت عملگر در مدل شبکه‌ای رعایت نمی‌شود. مثلا برای درج از دو عملگر connect و store می‌توان استفاده کرد. store برای درج نمونه رکورد و connect برای برقراری پیوند بین نمونه رکورد می‌باشد.

۱۷- (٤) مدل سلسله مراتبی برای پاسخگوئی به پرس و جوهای قرینه، رویه‌های پاسخگویی قرینه ندارد.

۱۸- (۲) مدل شبکه‌ای در عملیات ذخیره سازی آنومالی ندارد.

۱۹- (۱)در طراحی منطقی مدل مورد استفاده (رابطه‌ای- شبکه‌ای- سلسله مراتبی) مهم نیست و شمای کلی مثلا با نمودار ER ترسیم می‌شود. اما طراحی ادراکی بستگی به مدل مورد استفاده دارد.

۲۰- (۱)

۲۱- (٤)

تست های سری ۵

۱- کلید اصلی چیست ؟ (مسابقات آموزشکده‌های فنی- ۷٦)

۱. برای مرتب سازی اطلاعات بانک اطلاعاتی استفاده می‌شود.
۲. ترکیبی از اجزای داده‌ای است که مهم و کلیدی می‌باشد.
۳. جزء داده‌ای است که محل دقیق رکورد را مشخص می‌کند.
٤. کاربران بانک اطلاعاتی بوسیله آن با بانک ارتباط برقرار می‌کنند.

۲- کلید اصلی چیست ؟ (مسابقات آموزشکده‌های فنی- ۷۷)
۱. جزء داده‌ای است که محل رکورد مورد نظر ما را مشخص می‌کند.
۲. ترکیبی از چند جزء داده است.
۳. با آن می‌توان اطلاعات را مرتب سازی و sort نمود.
٤. رابطه‌ای است بین کاربر و بانک اطلاعاتی.

۳- با توجه به مفاهیم کلید در سیستم های رابطه‌ای کدام گزینه مصداق ندارد ؟ (مسابقات آموزشکده‌های فنی- ۷۸)
۱. کلید کاندید یکی از کلیدهای اصلی است که طراح آن را به عنوان کلید کاندید بر می‌گزیند.
۲. کلید اصلی یکی از کلید های کاندید است که طراح آن را به عنوان کلید اصلی بر می‌گزیند.
۳. کلید کاندید هر زیر مجموعه‌ای از عناصر رابطه است که یکتا بوده و کهینگی یاایجاز (minimality) داشته باشد.
٤. کلید خارجی کلیدی است که در رابطه دیگر کلید اصلی باشد.

٤- درجه رابطه در بانک مدل رابطه‌ای چیست ؟ (مسابقات آموزشکده‌های فنی- ۷۸)
۱. به مقادیر یک صفت خاصه گفته می‌شود.
۲. به میدان مقادیر یک صفت خاصه گفته می‌شود.
۳. تعداد تاپلهای رابطه است.
٤. تعداد صفات خاصه موجودیت است.

۵- کدام گزینه در بانک رابطه‌ای صحیح نیست ؟ (مسابقات آموزشکده‌های فنی- ۷۹ و مشابه مسابقات ۸۰)

۱. تاپلها نظم دارند.
۲. تاپل تکراری در رابطه وجود ندارد.
۳. صفات خاصه نظم ندارند.
٤. همه صفات خاصه تجزیه ناپذیرند.

٦- کدام گزینه به مفهوم ((مجموعه مقادیر مجاز یک صفت)) می‌باشد ؟ (مسابقات آموزشکده‌های فنی- ۷۹)

۱. Integrity
۲. Persistent
۳. Tuple
٤. Domain

۷- در رابطه با جامعیت در مدل رابطه‌ای کدام گزینه نادرست است ؟

۱. جامعیت دامنه‌ای یعنی تمام صفات در تمامی‌رابطه‌ها از نوع دامنه خود باشند.
۲. جامعیت درون رابطه‌ای یعنی هر رابطه‌ای ارتباطش با رابطه دیگر صحیح باشد.
۳. جامعیت دامنه‌ای یعنی کلیدهای دارای مقدار تهی (NULL) یا تکراری نباشند.
٤. جامعیت ارجاع یعنی کلید خارجی درست تعریف شده باشد.

۸- کدام تعریف در رابطه با کلیدها نادرست است ؟

۱. Super key یعنی هر ترکیبی از صفت ها که خاصیت کلید داشته باشند ولی‌این ترکیب باید کمینه (minimal) باشد.
۲. Primary key یکی از کلید های کاندید است که توسط مدیر بانک انتخاب می‌شود.
۳. Alternative key کلیدهای کاندید به غیر از کلید اصلی می‌باشند.
٤. Foreign key صفتی است در یک رابطه که در رابطه دیگر کلید اصلی یا فرعی است.

۹-این جمله تعریف چیست ؟ ((زیر مجموعه‌ای از ضرب دکارتی چند دامنه))

۱. Domain
۲. Relation
۳. Tuple
٤. Join

۱۰- کدام گزینه نادرست است ؟

۱. قاعده جامعیت ارجاعی در رابطه با کلید خارجی است.
۲. قاعده جامعیت موجودیتی در رابطه با کلید اصلی است.
۳. Super key کلید اصلی است که یکتایی و minimality دارد.
٤. کلید کاندید غیر از کلید اصلی را Alternative می‌گویند.

۱۱- رابطه عبارت است از:

۱. پل ارتباطی برای فیلدها
۲. مفهوم ریاضی یک جدول
۳. جمع شدن چندین جدول در یک قالب بزرگتر
٤. هیچکدام

۱۲- کدامیک از روشهای زیر بهترین راه برای حصول اطمینان از آن است که کلیدهای اصلی و خارجی از هماهنگی برخوردارند ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. از طریق اعمال پیش فرضهای (default values)
۲. از طریق Referential Integrity
۳. از طریق Entity
٤. هیچکدام

۱۳- کدام گزینه درباره رابطه نادرست است ؟

۱. کاردینالیتی رابطه تعداد صفات خاصه رابطه در یک لحظه از حیات آن است.
۲. در رابطه تاپلها نظم ندارند.
۳. همه مقادیر صفات خاصه در رابطه تجزیه ناپذیر (atomic) می‌باشند.
٤. رابطه از دو مجموعه Heading و body تشکیل یافته است.

۱٤- کلید خارجی چیست ؟

۱. امکان پیوند دادن رابطه‌های بانک اطلاعاتی را می‌دهد.
۲. امکان رجوع از رابطه‌ای به رابطه دیگر را می‌دهد.
۳. امکان خارج شدن از رابطه‌ای را می‌دهد.
٤. ۲و۱

۱۵- کدام گزینه نادرست است ؟

۱. سیستم رابطه‌ای دارای ساختار منطقی است.
۲. جدول مشتق (drived table) رابطه‌ای است که از جدول دیگر اخذ گردیده است.
۳. در رابطه همه مقادیر صفات خاصه قابل تجزیه می‌باشند.
٤. بانک اطلاعاتی رابطه‌ای به عنوان مجموعه‌ای از روابط و جدولها در نظر گرفته می‌شود.

۱٦- چه رابطه‌ای بین کلید های اصلی و خارجی در یک پایگاه اطلاعاتی باید وجود داشته باشد ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. در زمانی که رابطه‌ای بین دو جدول به وجود می‌آید، جدول پدر دراین ارتباط به کلید خارجی بستگی دارد و جدول فرزند به کلید اصلی بستگی خواهد داشت.
۲. در زمانی که رابطه‌ای بین دو جدول به وجود می‌آید، جدول پدر دراین ارتباط به کلید اصلی بستگی دارد و جدول فرزند به کلید خارجی بستگی خواهد داشت.
۳. ستون داده‌های کلید اصلی در جدول شامل ستون کلیدهای خارجی می‌باشد.
٤. هیچ رابطه‌ای بین کلید اصلی و خارجی وجود ندارد.

۱۷- کدامیک از اقلام زیر بهترین تعریف به عنوان یک شخص، مکان چیز یا مفهوم از داده‌های جمع آوری شده‌اند ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. یک رابطه
۲. یک کلید اصلی
۳. یک جدول
٤. یک فیلد

۱۸- جهت پیاده سازی یک بانک اطلاعاتی کدام فعالیت در اولویت قرار دارد ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. طراحی مدل داده‌ها منطقی
۲. طراحی مدل داده‌ها فیزیکی
۳. طراحی مدل داده‌ها به صورت فیزیکی و منطقی
٤. هیچکدام

۱۹- کدامیک از جملات زیر مزیت و برتری یک ستون غیر قابل تجزیه در یک طراحی نیست ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. بهتر می‌توان رابطه کلیدهای اصلی و خارجی را در آنها نگهداری نمود.
۲. بهتر می‌توان از یک ستون Query یا سوال نمود.
۳. بهتر می‌توان صحت اطلاعات (integrity) را بدست آورد.
٤. بهتر می‌توان ستون ها را به روز آورد.

۲۰- در اغلب نرم‌افزارهای پایگاه‌داده‌ها چه چیزی برای ذخیره سازی اطلاعات یک Entity یا موجودیت به کار می‌رود ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. کلید اصلی
۲. فیلد
۳. جدول
٤. بانک اطلاعاتی

۲۱- زمانی که در حال مطالعه مجموعه‌ای از نیازهای بانک اطلاعاتی در حال طراحی هستیم بهترین روش برای مشخص نمودن موجودیت (Entity) ها کدام است ؟ (کارشناسی ناپیوسته- آزاد ۸۰)

۱. به دنبال افعال جملات باشیم و از آنها برای تعریف Entity استفاده کنیم.
۲. به دنبال اسامیو نامهای مختلف باشیم و از آنها برای تعریف Entity ها استفاده کنیم.
۳. به دنبال صفت ها در جملات باشیم و از آنها برای تعریف Entity ها استفاده کنیم.
٤. همه اقلام فوق درست است.

۲۲- کدام گزینه نادرست است ؟

۱. یک DBMS کاملا رابطه‌ای (Fully relational)است اگر ساختار آن جدولی (tabolar) بوده و محدودیتهای جامعیتی را رعایت کند.
۲. در یک رابطه یک صفت غیر کلید، هیچ ارتباطی با کلید اصلی ندارد.
۳. منحصر به فرد بودن و غیر قابل کاهشی از ویژگیهای کلید کاندید است.
٤. اختصاص یک کلید نشانگر یک عامل محدود کننده در جهان واقعی پایگاه‌داده است.

۲۳- کدام گزینه نادرست است ؟

۱. رابطه دارای یک عنوان و یک بدنه است.
۲. رابطه دارای مجموعه‌ای از کلیدهای کاندید است.
۳. در مورد متغیرهای رابطه‌ای پایه (Base Relational)دقیقا یکی از کلیدهای کاندید باید به عنوان کلید اصلی انتخاب شود.
٤. رابطه پایه‌ای بر حسب روابط پایه‌ای دیگر محاسبه می‌شود.

۲٤- هنگام تعریف یک رابطه پایه (Base relation) کدام گزینه درست است ؟

۱. حداقل یک کلید کاندید وجود دارد.

۲. می‌تواند کلید اصلی وجود نداشته باشد.
۳. کلید کاندید وجود ندارد.
٤. باید حتما کلید خارجی وجود داشته باشد.

۲۵- دو رابطه به شرطی سازگارند که....
۱. دارای مجموعه‌ای از اسامی‌و صفات یکسان باشند.
۲. صفات خاصه متناظرشان بر روی دامنه یکسانی تعریف شده باشند.
۳. تعداد سطر و ستونهای یکسانی داشته باشند.
٤. ۲و۱

۲٦- کلید خارجی....
۱. در یک رابطه فیلد معمولی و در رابطه دیگر کلید اصلی است.
۲. فیلدی مشترک بین دو یا چند رابطه که در پیوند رابطه استفاده می‌گردد.
۳. ۲و۱
٤. همان کلید فرعی است.

۲۷- کلید جدول زیر کدام است ؟
۱. (A,C)
۲. (B)
۳. (A)
٤. (B,C)

۲۸- کدام گزینه نادرست است ؟
۱. رابطه زیر مجموعه‌ای از ضرب کارتزین چند دامنه است.
۲. دامنه‌های رابطه لزوما متمایز نمی‌باشند.
۳. در طول حیات جدول، Extention ثابت و Interntion متغیر است.
٤. رابطه از Heading و Body تشکیل یافته است.

۲۹- ((تمام اطلاعات موجود در بانک اطلاعاتی فقط به یک روش نمایش داده می‌شوند، یعنی به صورت مقادیری در موقعیتهای ستونی از سطرهای جدول)).این جمله کدام قاعده است ؟

۱. قاعده جامعیت ارجاعی
۲. قاعده اطلاعات
۳. قاعده درون رابطه‌ای
٤. قاعده برون رابطه‌ای

پاسخ تست های سری ۵

۱- (۳)

۲- (۱)

۳- (۱) توجه کنید کلید اصلی یکی از کلیدهای کاندید است نه‌اینکه کلید کاندید یکی از کلیدهای اصلی باشد.

٤- (٤)

۵- (۱) ترتیب تاپلها (رکوردها) در جدول مهم نیست.

٦- (٤)

۷- (۲) جامعیت درون رابطه‌ای یعنی هر رابطه‌ای به تنهایی صحیح می‌باشد. مثلا عضو تکراری نداشته باشد و کلیدهایش درست باشند. جامعیت دامنه‌ای هر دو گزینه۱ و 3 را در بر می‌گیرد.

۸- (۱) Super key هر ترکیبی از صفتهاست که خاصیت کلید داشته باشد.این تنها نوع کلید است که کمینه نیست یعنی زیر مجموعه‌ای از آن هم ممکن است کلید باشد.

۹- (۲)

۱۰- (۳) ابر کلید یا Super key به هر کلیدی گفته می‌شود که تنها ویژگی یکتایی را داشته باشد. کلید کاندید حالت خاصی از ابر کلید و همچنین کلید اصلی حالت خاصی از کلید کاندید است.

۱۱- (۲)

۱۲- (۲) (قاعده جامعیت ارجاعی = Referential Integrity rule)

۱۳- (۱) تعداد تاپلهای رابطه در یک لحظه از حیات آن کاردینالیتی رابطه نام دارد.

۱٤- (٤)

۱۵- (۳) در رابطه صفات خاصه غیر قابل تجزیه هستند.

۱٦- (۲)

۱۷- (٤) صورت تست در واقع تعریف موجودیت است و هر موجودیت را می‌توان با یک جدول (که همان رابطه است)نشان داده ولی چون رابطه همان جدول است پس احتمالا منظور طراح گزینه ٤ بوده است.

۱۸- (۱)

۱۹- (۱) بدیهی است هنگامی‌که ستون مرکبی داشته باشیم. مثل ستون آدرس که از ستون های شهر- خیابان- کوچه و پلاک تشکیل شده باشد به روز در آوردن ستونهای آن مشکلتر از وقتی است که ستونها ساده و غیر مرکب باشند.

۲۰- (۳) موجودیت فرد یا شیئ یا چیزی است که در مورد آن می‌خواهیم اطلاعاتی داشته باشیم، اطلاعات هر موجودیت را می‌توان در یک جدول خاصی ذخیره کرد.

۲۱- (۲) مثلا در جمله ((دانشجوی مشروطی دارای معدل کمتر از ۱۲ می‌باشد)) دانشجو موجودیت است و برای آن باید یک جدول داشته باسیم که معدل یکی از فیلدهای آن باشد. مشروط بودن دانشجو نیز یک فیلد آن جدول است.

۲۲- (۲) یک صفت غیر کلیدی با کلید اصلی ارتباط دارد.

۲۳- (٤) رابطه پایه‌ای بر حسب روابط پایه‌ای دیگر محاسبه نمی‌شود.

۲٤- (۱)

۲۵- (٤)

۲٦- (۳)

۲۷- (۱) کلید باید یکتا باشد. زیر ستون B دو تا K وجود دارد. زیر ستون A دو تا t وجود دارد. زیر جفت ستون (B,C)دو تا (K,S) وجود دارد.

۲۸- (۳) به بسط یک جدول در هر لحظه Extention و به نام و مجموعه صفات رابطه چکیده رابطه یا Intention گفته می‌شود یعنی اسم جدول به همراه heading برابر intention است. در طول حیات یک جدول Extextion متغیر ولی Intention ثابت است.

۲۹- (۲)

تست های سری ٦

۱- پیوند طبیعی دو جدول زیر چند سطر و چند ستون دارد ؟ (مسابقات آموزشکده‌های فنی- ۷۹)

sp

S#	P#	Qty تعداد
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

s

S#	sname	city
S1	فن آوران	تهران
S2	ایران قطعه	تبریز
S3	پولادین	تبریز

۱. ۱۸ سطر و ٦ ستون
۲. ۱۸ سطر و ۹ ستون
۳. ۹ سطر و ٦ ستون
٤. ٦ سطر و ٥ ستون

۲- عبارت A JOIN B را در نظر بگیرید. اگر تمام صفات خاصه A و B یکسان باشند عبارت فوق معادل کدام عبارت زیر است ؟ (مسابقات آموزشکده‌های فنی- ۷۹)

۱. A UNION B
۲. A MINUS B
۳. A INTERSECT B
٤. A TIMES B

۳- دو جدول زیر را در نظر بگیرید سپس بگوئید در نتیجه الحاق آنها توسط دستور داده شده، کلید اصلی چه می‌شود ؟

DEPT

DEPT #	DNAME	BUDGET
D1	Marketing	10M
D2	Developme nt	12M
D3	Research	5M

EMP

EMP#	ENAME	DEPT#	SALARY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K
E4	Saito	D2	35K

Join:
DEPT and EMP over DEPT#

۱. DEPT#
۲. EMP#
۳. BUDGET
٤. EMP# , DEPT#

٤- در جدول EMP در تست قبلی، کلید خارجی کدام است ؟

۱. EMP#
۲. SALARY
۳. DEPT#
٤. هیچکدام. چون D1 و D2 تکرار شده‌اند، #DEPT نمی‌تواند کلید خارجی باشد.

۵- تقاضای زیر در مورد بانک اطلاعاتی عرضه کنندگان و قطعات چه کاری انجام می‌دهد؟

Result: = ((S join SP) where P# ' p2') { S# , city};

۱. شهر عرضه کننده‌ای را می‌دهد که قطعه p2 را ساخته است.
۲. شماره عرضه کننده و شهر عرضه کننده‌ای را که قطعه p2 را تولید می‌کند، بازیابی می‌کند.
۳. شماره عرضه کننده‌ای را می‌دهد که قطعه p2 در همان شهر آن تولید شده است.
٤. هیچکدام

تذکر: سه جدول تهیه کننده S (S# , Sname , City) و قطعه P (P# , Pname , Color , City) و محموله SP (S# , P# , Qty) را در نظر گرفته و به ٤ سوال زیر پاسخ دهید.

٦- کدام گزینه شماره تهیه کنندگان و قطعات تولیدی آنها را چاپ می‌کند به شرط آنکه قطعه تولیدی آنها بیشتر از ۲۰۰ عدد باشد ؟

۱. Π s#, p# (σ Qty >۲۰۰ (SP))
۲. Π s#, p# (σ Qty >۲۰۰ (S∞ P))
۳. σ Qty >۲۰۰ (Π s#, p# (SP))
٤. 1,3

۷- در رابطه با دو دستور زیر کدام گزینه درست است ؟

الف) σ p# = p1 (S∞ SP) و (ب Π s# , p# (σ Qty >۲۰۰ (SP))

۱. این دو دستور دقیقا معادل یکدیگرند و هیچ ارجحیتی نسبت به هم ندارند.
۲. این دو دستور معادل نیستند.
۳. این دو دستور معادلند ولی ب کاراتر و بهینه تر از الف است.
٤. این دو دستور معادلند ولی الف کاراتر و بهینه تر از ب است.

۸- کدام گزینه اسامی‌تهیه کنندگان قطعه p2 را می‌دهد ؟

۱. Π sname (σp# = p2 (S∞ P))
۲. Π sname (σp# = p2 (S∞ SP))
۳. σp# = p2 (Π sname (P∞ SP))
٤. σp# = p2 (Π sname (P∞ S))

۹- دستور زیر چه می‌کند ؟

Π sname (Π s# ((Π p# (σ color = ' red' (P)) ∞ SP) ∞ S)

۱. شماره اسامی‌تهیه کنندگانی را می‌دهد که قطعات قرمز رنگ را تولید کرده‌اند.
۲. شماره و اسامی‌تهیه کنندگانی را می‌دهد که شماره قطعه آنها موجود بوده و رنگ آنها قرمز می‌باشد.
۳. اسامی‌تهیه کنندگانی را می‌دهد که یک قطعه قرمز رنگ تهیه می‌کنند.

٤. اسامی‌تهیه کنندگانی را می‌یابد که حداقل یک قطعه قرمز رنگ تهیه می‌کنند.

۱۰- کدام عملگر ها را می‌توان بوسیله عملگر های دیگر در جبر رابطه‌ای شبیه سازی کرد؟

۱. تقسیم- اجتماع- پیوند
۲. اشتراک- ضرب دکارتی- جایگزینی
۳. اشتراک- تقسیم- پیوند طبیعی
٤. ضرب دکارتی- تقسیم- اجتماع

۱۱- کدام فرمول در جبر رابطه‌ای نادرست است ؟ (σ نماد عملگر گزینش یا select می‌باشد)

۱. $A \times B = B \times A$
۲. $\sigma p(A \cap B) = \sigma p(A) \cap \sigma p(B)$
۳. $A \cap B = B - (B - A)$
٤. $\sigma p(B - A) = \sigma p(b) - \sigma p(a)$

۱۲- اگر A و B دو رابطه باشند، کلید اصلی A MINUS B چه می‌شود ؟

۱. کلید اصلی A
۲. کلید اصلی B
۳. کلید اصلی $A \cap B$
٤. کلید اصلی A U B

۱۳- اگر Divide by را به صورت سه عملوندی در نظر بگیریم، کدام فرمول در جبر رابطه‌ای صحیح است ؟

۱. (A Divide by B) TIMES B $\geq$ A
۲. (A Divide by B) TIMES B $\neq$ A
۳. (A Divide by B) TIMES B $\leq$ A
٤. (A Divide by B) TIMES B = A

۱٤- رابطه دارای مجموعه‌ای از صفات (عنوان) و مجموعه‌ای از چند تایی ها (پیکر) است. رابطه DEE مجموعه‌ای خالی از چند تایی هاست. رابطه DUM نیز مجموعه‌ای خالی از صفات می‌باشد. حال کدام گزینه درست است ؟

۱. عضو خنثی در حساب معمولی عدد یک است (n * 1 = 1 * n = n). در جبر رابطه‌ای معادل عدد ۱ رابطه DEE است و داریم R TIMES DEE=DEE TIMES R=R
۲. عضو صفر در حساب دارای خاصیت n*0=0*n=0 است در جبر رابطه‌ای معادل عدد صفر رابطه DUM می‌باشد.
۳. هر دو گزینه
٤. هیچکدام

۱۵- کدام عملگر در جبر رابطه‌ای پایه‌ای نمی‌باشد و به کمک سایر عملگر ها می‌توان آن را شبیه سازی کرد ؟

۱. پیوند طبیعی (Join یا∞)
۲. ضرب دکارتی (× یا TIMES)
۳. تفاضل (MINUS یا -)
٤. ۱و۳

۱٦- در جدول زیر

DEPT

D#	D name	Budjet
D1	M	10
D2	D	20
D3	R	5

نتیجه دستور زیر چیست ؟

Project:
DEPT over D# , Budjet

۱. ۲ صفت خاصه و هیچ تاپل
۲. هیچ تاپل و ۲ صفت خاصه
۳. ۲ صفت خاصه و ۳ تاپل
٤. ۲ تاپل و ۳ صفت خاصه

۱۷- کدام عملگر در جبر رابطه‌ای وجود دارد، ولی ممکن است در بعضی از بانکهای اطلاعاتی پیاده سازی نشده باشد ؟

۱. Join
۲. Subtract
۳. Add
٤. هر سه گزینه

۱۸- فرض کنید جدول دانشکده به صورت زیر شامل ((شماره، نام دانشکده و شهر)) باشد:

Clg (clg# , clgname , city)

و جدول درس شامل ((شماره درس، نام درس، تعداد واحد و شماره دانشکده ارائه دهند)) باشد:

Crs (c# , c name , unit , clg#)

حال به کمک دستورات جبر رابطه‌ای، درسهایی که توسط همه دانشکده‌ها ارائه می‌شوند را بدهید.

۱. Clg Join Crs giving temp ; select temp where cname ALL
۲. Project clg [clg#]Giving temp ; Crs Divideby temp
۳. Clg join crs giving temp ; select cname
٤. Project clg [clg#] Giving temp ; clg join crs join temp divideby cname

۱۹- حاصل عبارت زیر کدام است ؟

R1(A , B , C)			DIV	R2 (C)
a1	b1	c1		C1
a1	b1	c3		C3
a2	b2	c2		
a3	b1	c1		
a3	b1	c3		

۱.

R3(A , B)	
b1	a1
b1	a3

۲.

R3(A , B)	
b2	a2

۳.

R3(A , B , C)		
c1	b1	a1
c1	b1	a3
c3	b1	a1
c3	b1	a3

٤. هیچکدام

۲۰- حاصل عبارت زیر کدام گزینه است ؟

R2 (B , C)		DIV	R1(A , B , C)		
c1	b1		c1	b1	a1
c2	b2		c2	b2	a1
c3	b3		c3	b2	a1
			c1	b1	a2
			c2	b2	a2

۱.

R3(A , B , C)		
c1	b1	a2
c2	b2	a2

۲.

R3 (A)
a2

۳.

R3(A)
a1

٤.

R3(A , B)	
c1	b1
c2	b2

21- شماره قطعاتی را بدهید که توسط همه تهیه کنندگان تهیه شده‌اند.

۱. $Temp \leftarrow \Pi\ s\#$
$sp \div Temp$

۲. $Temp \leftarrow \Pi\ s\#(S)$
$\Pi\ p\#\ (\ sp \div Temp)$

۳.
$\Pi\ p\#\ (\ sp \infty S)$

٤. هر سه گزینه

۲۲- اگر DIVIDEBY همان عملگر معرفی شده توسط Codd (یعنی دو عملوندی) باشد کدام رابطه صحیح است ؟

۱. (A TIMES B) DIVIDEBY B=A
۲. (A DIVIDEBY B) TIMES B≤A
۳. (A TIMES B) DIVIDEBY B≠A
٤. ۱و۲

۲۳- با توجه به بسط جداول داده شده دراین فصل، خروجی دستور $S \infty SP \infty P$ چه می‌شود؟

۱.

آهن	قرمز	۳۰۰	P1	تهران	فن آوران	S1
مس	سبز	٤۰۰	P2	تبریز	ایران قطعه	S2

۲.

آهن	قرمز	۳۰۰	P1	تهران	فن آوران	S1
آهن	قرمز	۳۰۰	P1	تبریز	ایران قطعه	S2
آهن	قرمز	۳۰۰	P1	تهران	فن آوران	S1

۳. مس سبز ۲۰۰ P2 تبریز پولادین S3

٤. هیچکدام

۲٤- اگر رابطه A از درجه n باشد، چند تصویر مختلف از A وجود دارد ؟

۱. $(n-1)!$

۲. $n!$

۳. 2^n

٤. $2n$

۲۵- خاصیت بسته بودن (closure) به چه معناست ؟

۱. خروجی هر عملگر رابطه‌ای، یک رابطه است.

۲. الحاق دو رابطه رابطه‌ای جامعتر از هر دو می‌باشد.

۳. ضرب دکارتی دو رابطه مجموعه مادر هر دو می‌باشد.

٤. اجتماع دو رابطه در هر حال امکان پذیر است.

۲٦- کدام عملگرهای جبر رابطه‌ای به عملوندهای سازگار نیاز دارند ؟

۱. U

۲. $-$

۳. ∞

٤. ۱و۲

۲۷- جبر رابطه‌ای....

۱. یک زبان اجرایی برای بانک اطلاعاتی رابطه‌ای است.

۲. همان SQL است.

۳. زبانی از نسل چهارم (4GL) می‌باشد.

٤. به DBMS می‌گوید چگونه یک رابطه جدید از یک یا چند رابطه ساخته شود.

۲۸- زبانی دارای اکمال رابطه‌ای است که....

۱. حداقل همتوان با جبر رابطه‌ای باشد.

۲. حداکثر همتوان با جبر رابطه‌ای باشد.

۳. هر رابطه‌ای که با عبارات جبر رابطه‌ای قابل تعریف است، توسط آم زبان نیز تعریف شدنی باشد.

٤. ۱و۳

۲۹- کدام فرمول نادرست است ؟

۱. $A \cap B = A - (A - B)$

۲. $A \infty B \neq B \infty A$

۳. $A \infty B = \sigma (A \cap b) \times A$

٤. $A \infty B = B \infty A$

۳۰- $X \cup Y$ دو جدول زیر چند سطر و چند ستون دارد ؟

Y

شهر تولد	معدل	شماره	نام
تهران	۱۷	۱۲۳	علی
تبریز	۱۵	٥٧٤	جواد

X

شهر تولد	معدل	شماره	نام
تهران	۱۷	۱۲۳	علی
تهران	۱۷	۹۳٤	مجید

۱. ٤ ستون و ۳ سطر

۲. ۳ ستون و ٤ سطر

۳. ٤ ستون و ٤ سطر

٤. ۳ ستون و ۳ سطر

۳۱- کدامیک از دستورات زیر کاراتر است ؟

الف) (SP ∞ S) where p# = ' p4') [sname]

ب) ((SP where p# = ' p4') ∞ S) [sname]

۱. الف

۲. ب

۳. کارایی هر دو یکسان است.

٤. اتفاقی است و نمی‌توان گفت.

۳۲- اگر A رابطه‌ای با صفات X , Y , B رابطه‌ای دیگر با صفت خاصه X باشد، کدام گزینه معادل عبارت A÷ B می‌باشد ؟

۱. A [Y]− (A− [X])
۲. A [Y]−((A [Y]× B)− A) [Y]
۳. A [X]−((A [X]× B)− A) [X]
٤. A [X]− (A− B [X])

۳۳- با در نظر گرفتن بانک اطلاعاتی تهیه کنندگان و قطعات، کدام دستور مشخصات تهیه کنندگانی را می‌دهد که تمام قطعات را عرضه کرده‌اند ؟

۱. SP (S# , P#) divided by P (P#) Join S
۲. (SP Join S) Minus P
۳. SP (S# , P#) divided by P (P#) Union S
٤. (SP Join P) Minus S

۳٤- با در نظر گرفتن بانک اطلاعاتی تهیه کنندگان و قطعات، کدام دستور مشخصات تهیه کنندگانی را می‌دهد که قطعه P3 را تهیه نمی‌کنند ؟

۱. [Π S# (S ∞ SP)U [σp#< >'p3'(SP)]]
۲. [Π S#(S) − σp#< >'p3'(SP)] ∞ S
۳. [Π S#(S) − (Π S#(σp#= 'p3'(SP)))] ∞ S
٤. هیچکدام

۳۵- دو رابطه x و y زیر را در نظر گرفته و بگوئید x times y چند سطر خواهد داشت ؟

y
e
a
f
g

x
a
b
c
d
m

۱. 20سطر
۲. ۱۵ سطر
۳. ۹ سطر

٤. ۲۵ سطر

۳٦- پیوند طبیعی دو جدول زیر چند سطر و ستون خواهد داشت ؟

P	M	Q
P1	a	x
P2	b	y
P3	c	z
P4	d	u

M	N
a	m
b	n
c	k

۱. 4 , 4
۲. 5 , 4
۳. 3 , 3
٤. 4 , 3

۳۷- کدامیک از عملگرهای زیر در بانک اطلاعاتی بر روی جداول کار می‌کنند ؟(کارشناسی ناپیوسته- آزاد ۸۰)

۱. Join
۲. Add
۳. Subtract
٤. هیچکدام

۳۸- جدول سمت چپی با چه دستوری تبدیل به جدول سمت راستی می‌شود ؟

P

P#	Weight
P1	15
P2	13
P3	18

→

P#	Weight	GRM
P1	15	15000
P2	13	13000
P3	18	18000

۱. extend P Add (weight * 1000) AS GRM
۲. Add P (weight * 1000) AS GRM
۳. P Union weight * 1000 AS GRM
٤. هیچکدام

۳۹- در شکل زیر DEND مقسوم، MED میانجی و DOR مقسوم علیه است. حاصل دستور DEND DIVIDEBY DOR PER MED کدام گزینه است ؟

DEND: _____ **MED:** _____ **DOR:** _____

P#
P2
P4

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

S#
S1
S2
S3
S4
S5

۱.

S#	P#
S1	P1
S2	P3

۲.

P#
P1
P3

۳.

S#
S1
S2

٤.

S#
S1
S4

٤۰- در تست قبلی اگر DOR $\frac{\text{P\#}}{\text{P1}}$ جدول باشد خروجی چه می‌شود ؟

۱.

S#
S1
S4

۲.

S#
S1
S2

۳.

P#
P1
P3

٤.

S#
S1
S2
S3
S4

٤١- در جبر رابطه‌ای $A-(A-B)$ معادل کدام است ؟

۱. $A-(B-A)$
۲. $A \cap B$
۳. $B-(B-A)$
٤. 3,2

پاسخ تست های سری ٦

۱- (٤) پیوند طبیعی‌این دو جدول به صورت زیر می‌شود:

S#	P#	تعداد	Sname	city
S1	P1	300	فن آوران	تهران
S1	P2	200	فن آوران	تهران
S1	P3	400	فن آوران	تهران
S2	P1	300	ایران قطعه	تبریز
S2	P2	400	ایران قطعه	تبریز
S3	P2	200	پولادین	تبریز

۲- (۳) منظور از INTERSECT همان اشتراک ∩ و منظور از TIMES همان ضرب دکارتی × می‌باشد.

۳- (۲) نتیجه الحاق دو جدول داده شده به صورت زیر است:

DEPT#	DNAME	BUDGET	EMP#	ENAME	SALARY
D1	Marketing	10M	E1	Lopez	40K
D1	Marketing	10M	E2	Cheng	42K
D2	Development	12M	E3	Finzi	30K
D2	Development	12M	E4	Saito	35K

با توجه به جدول فوق روشن است که ستون #EMP کلید اصلی است و چون داده‌های #DEPT تکرار شده نمی‌توان کلید باشد.

٤- (۳) در تعریف کلید خارجی یکتائی جزو شروط آن نیست. فقط باید آن ستون در جدول دیگری کلید اصلی باشد.

۵- (۲) نماد به کار رفته مربوط به زبان Tutorial D است که در کتاب دیت استفاده شده است. ستونهای خروجی در علامت آکولاد نوشته می‌شوند.

٦- (۱) توجه کنید که گزینه ۳ نادرست است زیرا هنگامی‌که عملگر Π ستونهای #S و #p را استخراج می‌کند دیگر ستون Qty وجود ندارد که عملگر۲۰۰< Qty σ قابل اجرا باشد.

۷- (۳) در دستور الف ابتدا دو جدول s و sp پیوند خورده سپس سطرهایی از آن انتخاب می‌شود.در دستور ب ابتدا سطرهایی از جدول sp انتخاب شده سپس آن سطرها با جدول s پیوند می‌خورد بدیهی است که پیوند دو جدول کوچکتر زمان و فضای کمتری می‌خواهد.

۸- (۲) برای اسامی‌تهیه کنندگان قطعه p2 از جداول s و sp باید استفاده کنیم. لذااین دو جدول باید به همدیگر پیوند بخورند.

۹- (٤) از داخلی ترین پرانتز دستورات را اجراء کنید.

شماره قطعات قرمز رنگ را می‌دهد: $A \leftarrow \Pi\ p\#\ (\sigma\ color = 'red'\ (p))$

شماره تهیه کنندگان را می‌دهد که حداقل یک قطعه قرمز رنگ تولید کرده‌اند:

$A \leftarrow \Pi\ S\#\ (A \infty SP)$

نام تهیه کنندگانی را می‌دهد که حداقل یک قطعه قرمز رنگ را تولید کرده‌اند:

$\Pi\ Sname\ (B \infty S)$

۱۰- (۳)

۱۱- (٤)

$\sigma\ p\ (a\text{-}b) = \sigma\ p\ (a) - \sigma\ p\ (b)$

ضرب دکارتی در ریاضیات توابع، خاصیت جابه جائی ندارد ولی در مورد جداول از آنجا که جابه جایی ستونها (فیلدها) مجاز است لذا ضرب دکارتی دو جدول خاصیت جابه جایی دارد.

۱۲- (۱) A-B یعنی سطرهایی که در A هست ولی در B نیست.پس A-B زیر مجموعه‌ای از A می‌شود و بدیهی است که کلید اصلی جدول A، کلید اصلی زیر مجموعه آن نیز خواهد بود.

۱۳- (۳)

۱٤- (۱) DUM مثل صفر نیست.

A TIMES DUM = DUM TIMES R = an empty relation with the same heading as R

پس گزینه ۲ اشتباه است.

۱۵- (۱)

۱٦- (۳) خروجی به شکل زیر است:

D#	Budjet
D1	10
D2	20
D3	5

۱۷- (۲)

۱۸- (۲) عموما وقتی در سوال، کلمات ((همه)) یا ((تمام)) دیده می‌شود باید از عملگر تقسیم استفاده کرد.

دستور project clg [clg#] Giving temp ابتدا شماره همه دانشکده‌ها را می‌دهد. سپس Crs divideby temp جدول Crs را بر شماره دانشکده‌ها تقسیم کرده و بدین ترتیب درسهایی که توسط همه دانشکده‌ها ارائه می‌شوند را می‌دهد.

۱۹- (۱)

۲۰- (۳)

۲۱- (۲)

۲۲- (٤) در حساب معمولی ضرب و تقسیم عکس یکدیگرند ولی در جبر رابطه‌ای TIMES و DIVIDEBY عکس یکدیگر نیستند چرا که اولا تقسیم جدید سه عملوندی است، ثانیا تقسیم A بر B و سپس تشکیل ضرب دکارتی از نتیجه حاصل و رابطه B منجر به رابطه‌ای می‌شود که ممکن است همانندA باشد، اما احتمالا زیر مجموعه‌ای از A است (فرمول گزینه ۲). بنابرین عملگر DIVIDEBY در گزینه ۲ خیلی شبیه تقسیم صحیح در حساب معمولی است (یعنی از باقی مانده صرفنظر می‌کند)

۲۳- (۱) توجه کنید که هنگام پیوند S ∞ SP با P باید دو ستون city و #p هر دو یکسان باشند.

۲٤- (۳)این تعداد 2^n مشابه تعداد زیر مجموعه‌های یک مجموعه با n عضو است و شامل تصویر همانی (که نتیجه آن همانند رابطه اصلی A است) و تصویر تهی نیز می‌باشد.

۲۵- (۱)

۲٦- (٤) در اجتماع، اشتراک و تفاضل رابطه‌ها باید سازگار باشند، یعنی از نظر تعداد صفات خاصه و دامنه بایستی با یکدیگر سازگار باشند. اجتماع، اشتراک و تفاضل دو رابطه A و B سازگار است با عنوان A و B و بدنه‌ای شامل مجموعه تاپلهای A و B.

۲۷- (٤)

۲۸- (٤)

۲۹- (۳)

۳۰- (۱)

$X \cap Y$ یک سطر و ٤ ستون خواهد داشت.

$X-Y$ نیز یک سطر و ٤ ستون خواهد داشت.

XUY

شهر تولد	معدل	شماره	نام
تهران	۱۷	۱۲۳	علی
تهران	۱۷	۹۳٤	مجید
تبریز	۱۵	۵۷٤	جواد

تهران، ۱۷، ۱۲۳، علی $\rightarrow X \cap Y$

تهران، ۱۷، ۹۳٤، مجید $\rightarrow Y-X$

۳۱- (۲) ب کاراتر است چرا که گزینش را زودتر از پیوند انجام داده است.

۳۲- (۲)

۳۳- (۱) با دستور SP (S# و p#) divided by P(P#) شماره تهیه کنندگانی بدست می‌آید که تمام قطعات را عرضه می‌کنند. سپس هنگامی‌که نتیجه‌این دستور با S الحاق شود مشخصات‌این تهیه کنندگان بدست می‌آید.

۳٤- (۳)

۳۵- (۱) جدول X دارای ۵ سطر و جدول Y دارای ٤ سطر است پس X times Y دارای ۲۰= ۵×٤ سطر است.

۳٦- (٤) نتیجه پیوند طبیعی به شکل زیر است:

Q	P	N	M
x	P1	m	a
y	P2	n	b
z	P3	k	c

۳۷- (۱) در جبر رابطه‌ای عملگر Join یا پیوند را داریم. همچنین عملگر های اشتراک و اجتماع و تفاضل (Minus یا Subtract) را نیز داریم. ولی عملگر Add یا جمع را نداریم. در اکثر نرم‌افزارهای بانک اطلاعاتی عملگر تفاضل پیاده سازی نشده است ولی عملگر پیوند در تمامی‌آنها وجود دارد.

۳۸- (۱) دستور extend برای اضافه کردن ستون به یک جدول استفاده می‌شود.

۳۹- (٤) یعنی کدام #S ها هم p2 و هم p4 را تهیه کرده‌اند.

٤۰- (۲)

٤۱- (٤)

تست های سری ۷

۱- اگر p و q یک و ف باشند معادل عبارت زیر کدام است ؟

IF p THEN q END IF

۱. P AND (NOT q)
۲. IF NOT p then NOT q
۳. (NOT p) OR q
٤. معادل ندارد.

۲- اگر ST و CO وSTCO متغییرهای تاپلی رابطه‌های STT و COT و STCOT باشند آنگاه عبارت زیر در حساب رابطه‌ای تاپلی چه می‌کند ؟

(ST.STID,ST.STDEG)WHERE EXISTS ST(ST.STID=STCO.STID AND STCO.COID='COM111');

۱. شماره دانشجویی و دوره تحصیلی آنهایی که درس COM111 را انتخاب کرده‌اند، می‌دهد.
۲. شماره دانشجویی و شماره درس افرادی را می‌دهد که درس COM111 را انتخاب کرده‌اند.
۳. سطح تحصیلی دانشجویانی را می‌دهد که درس COM111 را انتخاب کرده‌اند.
٤. این دستور غلط است.

۳- کدام گزینه در حساب رابطه‌ای تاپلی نادرست است ؟

۱. عبارت ST.STID مجموعه تمام شماره دانشجویان در رابطه STT را می‌دهد.
۲. عبارت ST عنوان جدول ST را می‌دهد.
۳. عبارت (STCO.STID,STCO.GRADE) شماره دانشجویان و نمره آنها از رابطه STCOT را می‌دهد.
٤. عبارت ;(CO.COID AS CONUM)WHERE COTYPE='P' شماره درسهای عملی (عملی = 'P') از رابطه COT را می‌دهد.

٤- کدام دستور در حساب رابطه‌ای تاپلی عنوان درسهایی را می‌دهد که دانشجو با شماره ۱۲۳٤ انتخاب کرده است ؟

۱. COTITLE WHERE EXISTS STCO (STCO.COID=CO.COID AND STCO.STID='1234');
۲. COTITLE WHERE STCO.STID='1234';
۳. COTITLE WHERE STCO.COID=CO.COID AND STCO .STID='1234';
٤. هیچکدام

۵- کدام عبارت در حساب رابطه‌ای تاپلی شماره و نام درسهای دو واحدی را می‌دهد ؟

۱. CO.COID,CO.COTITLE WHERE EXISTS CO.CREDIT='2';
۲. CO.COID,CO.COTITLE WHERE CO.CREDIT='2';
۳. CO.CREDIT='2' FORALL WHERE CO. COID,CO.COTITLE;
٤. CO.COID,CO.COTITLE FORALLWHERE CO.CREDIT='2';

٦- عبارت زیر در حساب رابطه‌ای تاپلی نام دانشجویانی را می‌دهد که حداقل یک درس عملی را انتخاب کرده باشند. کدام گزینه فرم معادل پیشوندی (Prenex) آن است؟

ST.STNAME WHERE EXISTS STCO (ST.STID=STCO.STID AND EXISTS CO (CO.COID = STCO.COID AND CO.COTYPE='P'));

١. ST.STNAME EXISTS WHERE (ST.STID=STID AND CO.COID =STCO.COID AND CO.COTYPE='P');
٢. ST.STNAME WHERE NOT FORALL STCO(ST.STID=STCO.STID AND CO(CO.COID=STCO.COID AND CO.COTYPE='P'));
٣. ST.STNAME WHERE EXISTS STCO (EXISTS CO (ST.STID= STCO.STID AND CO. COID=STCO.COID AND CO.COTYPE='P'));
٤. هیچکدام

٧- خروجی عبارت زیر در حساب رابطه‌ای تاپلی چیست ؟

ST.STNAME WHERE NOT EXISTS STCO (STCO.STID=ST.STID AND STCO.COID='COM333')

١. اسامی‌دانشجویانی که درس شماره COM333 را انتخاب نکرده‌اند و شماره آنها در جدول STCOT نیست.
٢. اسامی‌و شماره دانشجویانی که درس شماره COM333 را انتخاب کرده‌اند.
٣. اسامی‌دانشجویانی را می‌دهد که درس شماره COM333 را انتخاب نکرده‌اند.
٤. تمامی‌اسامی‌دانشجویانی که درس شماره COM333 را انتخاب کرده‌اند.

٨- عبارت زیر در حساب رابطه‌ای تاپلی چه می‌کند ؟ (bs یعنی دوره کارشناسی)

CO.COTITLE WHERE FORALL ST (EXISTS STCO(STCO.STID=ST.STID AND STCO.COID=CO.COID AND ST.STDEG='bs'));

١. این عبارت نادرست است وایراد دارد.
٢. عنوان درسهایی را می‌دهد که تمام دانشجویان دوره کارشناسی که شماره‌های یکسان دارند آنها را انتخاب کرده باشند.

۳. شماره درس و شماره دانشجویانی را می‌دهد که تمام درسهای کارشناسی را گذرانده باشند.

٤. عنوان درسهایی را می‌دهد که تمام دانشجویان دوره کارشناسی آنها را انتخاب کرده باشند.

۹- اگر متغیر میدانی CO بر روی فیلد COID تعریف شده باشد، در حساب رابطه‌ای میدانی دستور زیر چه می‌کند ؟

CO WHERE COT (COID: CO,CODEID:'D111');

۱. شماره درسهای گروه آموزشی 'D111' را می‌دهد.

۲. شماره درسها و شماره گروه آموزشی درسهایی را می‌دهد که شماره آنها D111 باشد.

۳. گروه آموزشی را می‌دهد که شماره درس آن D111 باشد.

٤. تمام گروه‌هایی را می‌دهد که درس D111 را ارائه کرده‌اند.

۱۰- اگر متغیر میدانی CO بر روی فیلد COID و متغیر میدانی ST بر روی فیلد STID و متغیر CRED بر روی فیلد CREDIT تعریف شده باشند کدام گزینه در حساب رابطه‌ای میدانی نادرست است ؟

۱. عبارت CO مجموعه شماره همه درسها را نشان می‌دهد.

۲. عبارت ST WHERE STT (STID: ST) مجموعه شماره دانشجویان موجود در رابطه STT را نشان می‌دهد.

۳. دستور زیر شماره درسهای یک واحدی از گروه آموزشی D222 را می‌دهد:

CO WHERE EXISTS CRED (CRED='1' AND COT (COID: CO,CREDIT:CRED,CODEID='D222')

٤. هیچکدام

۱۱- دستور زیر در حساب رابطه‌ای میدانی چه می‌کند ؟

STNAM WHERE EXISTS ST (STT(STID:ST,STNAME:STNAM) AND NOT STCOT (STID: ST,COID:'COM333'))

۱. نام دانشجویانی را می‌دهد که درس COM333 را انتخاب نکرده‌اند.

۲. نام دانشجویانی را می‌دهد که از درس COM333 افتاده‌اند.
۳. مشخصات دانشجویانی را می‌دهد که حداقل درس COM333 را انتخاب کرده‌اند.
٤. نام دانشجویانی را می‌دهد که تمام درسهای هم گروه با COM333 را انتخاب نکرده‌اند.

۱۲- کدام گزینه نادرست است ؟

۱. قدرت جبر رابطه‌ای با حساب رابطه‌ای یکسان است.
۲. حساب رابطه‌ای با حساب رابطه ی تاپلی از قدرت محاسباتی مساوی برخوردارند.
۳. حساب رابطه‌ای مبتنی بر شاخه‌ای از زیاضیات به نام Predicate calculus می‌باشد.
٤. جبر رابطه‌ای توصیفی است در حالی که حساب رابطه‌ای دستوری است.

پاسخ تستهای سری ۷

۱- (۳) در ریاضیات می‌دانیم که $p \rightarrow q = NOT\ p\ OR\ q$

۲- (۱)

۳- (۲) عبارت ST تمام رابطه STT را می‌دهد. (تمام تاپلهای آن را)

٤- (۱)

۵- (۲) توجه کنید که وقتی اطلاعات فقط در یک جدول است از سورهای EXISTS یا FORALL استفاده نمی‌شود.

٦- (۳)

۷- (۳)

۸- (٤)

۹- (۱)

۱۰- (٤)

۱۱- (۱)

۱۲- (٤) حساب رابطه‌ای توصیفی یا نارویه‌ای است ولی جبر رابطه‌ای دستوری یا رویه‌ای است.

تست های سری ۸

۱- SQL چیست ؟

۱. یک زبان قوی است برای‌ایجاد و دسترسی به بانک اطلاعاتی
۲. یک برنامه کاربردی است که از DBMS فرمان می‌گیرد.
۳. قسمتی از بانک اطلاعاتی است و مخصوص دستکاری داده‌ها می‌باشد.
٤. DDL و DML روی هم SQL را می‌سازند.

۲- کدام دستور SQL نمی‌تواند باعث به وجود آمدن ناسازگاری در داده‌ها شود ؟

۱. INSERT
۲. DELETE
۳. UPDATE
٤. SELECT

۳- در بانک اطلاعاتی ((تولید کننده- قطعه- پروژه))دستور SQL زیر چه خروجی تولید می‌کند ؟

```
Select  S# , P# , J#
From S , P , J
Where  S.city=P.city  AND  P.city=J.city
```

۱. شماره تمام تهیه کنندگان- شماره قطعه و شماره پروژه‌هایی را می‌دهد که تهیه کننده قطعه و پروژه دو به دو از یک شهر نیستند.
۲. شماره تمام تهیه کنندگان- شماره قطعه و شماره پروژه‌هایی را می‌دهد که از یک شهر هستند.
۳. شماره تمام تهیه کنندگان- شماره قطعه و شماره پروژه‌هایی را می‌دهد که از یک شهر نیستند.
٤. شماره تمام تهیه کنندگان- شماره قطعه و شماره پروژه‌هایی را می‌دهد که تهیه کننده قطعه و پروژه دوبه دو از یک شهر هستند.

٤- اگر جدول P دارای فیلدهای (P# , Pname , Color , Weight , city) ، جدول S دارای فیلدهای (S# , Sname , Status , City) و جدول SP دارای فیلدهای (S# , P# , City) باشند با زبان SQL به سوال زیر پاسخ دهید:

((تمام جفت شهرهایی (City) را بیایید که تهیه کننده (S) ساکن شهر اول، قطعه‌ای (P) انبار شده و شهر دوم را تهیه کرده باشد.))

۱.

```
Salect city From S , P
Where S.city = P.city
```

۲.

```
Salect S.city , P.city  From S , P
Where S.city = P.city
```

۳.

```
Salect distinct S. city,P.city  From S,P, SP
```

Where S. S# = SP.S# AND SP.P# = P.P#

٤. هیچکدام

۵- منظور از SQL چیست ؟

۱. زبان پرس و جوی ساخت یافته است.

۲. زبان فرعی داده‌ای است.

۳. زبان داده‌ای مستقل است.

٤. همان زبان میزبان است.

٦- دستور مقابل نام چه افرادی را چاپ می‌کند ؟

```
Select name
From T
where name like"٪رضا٪"
```

(فرض کنید T جدولی است که یکی از ستون های آن نام (name) می‌باشد.)

۱. افرادی را چاپ می‌کند که ((رضا)) در نام آنها آمده است.

۲. افرادی را که اسم آنها ((رضا)) باشند چاپ می‌کند.

۳. افرادی را که اسم فامیل آنها ((رضا)) است فقط چاپ می‌کند.

٤. افرادی را که اسم یا اسم فامیل آنها ((رضا)) است چاپ می‌کند.

۷- با توجه به دو جدول S و T کدام دستور باعث نمایش خروجی زیر می‌شود ؟

S:Name	T: Name	Name	خروجی :
Able	Able	Able	
Bravo	Baker	Charlie	
Charlie	Charlie	Exitor	
Decon	Exitor	Goober	
Exitor	Falconer		
Goober	Fuber		
	Goobar		

۱. Select * From S union select * From T ;

۲. Select * From S Minus select * From T ;

۳. Select * From S,T where S.Name = T.name ;
٤. Select * From S Intersect select * From T ;

۸- با توجه به دو جدول S و T کدام دستور خروجی زیر را می‌دهد ؟

S:Name	T: Name	Name	خروجی :
Able	Able	Bravo	
Bravo	Baker	Decon	
Charlie	Charlie		
Decon	Dean		
Exitor	Exitor		
Goober	Falconer		
	Fuber		
	Goobar		

۱. Select * From T Intersect select * From S ;
۲. Select * From T Minus select * From S ;
۳. Select * From S Minus select * From T ;
٤. Select * From T Union select * From S ;

۹- در بانک اطلاعاتی عرضه کنندگان و قطعات نتیجه نهایی دستور شبه SQL زیر چیست؟

```
CREATE VIEW V AS
((S JOIN SP) where P# = 'P2')    {S# , city} ;
Result: = ( V where city = 'london') {s#} ;
```

۱. شماره عرضه کننده را برای عرضه کنندگان لندن که قطعه P2 را تولید می‌کنند، می‌دهد.
۲. شماره عرضه کننده را برای عرضه کنندگان ساکن شهر لندن، می‌دهد.
۳. شماره عرضه کننده را برای عرضه کنندگان تولید کننده قطعه P2، می‌دهد.
٤. شهر و شماره عرضه کنندگانی را می‌دهد که قطعه انبار شده در شهر لندن را تولید کرده‌اند.

۱۰- دستور روبرو در SQL چه می‌کند ؟

```
Select *
From SP
```

۱. تمام جدول SP را چاپ می‌کند.
۲. این دستور غلط است چون قسمت where ندارد.
۳. نام تمام فیلدهای جدول SP را چاپ می‌کند.
٤. کل اطلاعات ستون های اول و آخر جدول را چاپ می‌کند.

۱۱- دستور زیر چه می‌کند ؟

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

۱. تمام سطرها و ستونهای دو جدول S و P را زیر هم چاپ می‌کند.
۲. دو جدول S و P را به طور کامل کنار هم چاپ می‌کند.
۳. ضرب کارتزین دو جدول S و P را می‌دهد.
٤. پیوند طبیعی دو جدول S و P را می‌دهد.

۱۲- دستور زیر چه می‌کند ؟

```
Select S# from S
Where city = (select city from S where S# = S1)
```

۱. شماره تهیه کنندگانی را می‌دهد که تهیه کننده S1 قطعات خود را در آن انبار کرده است.
۲. شماره تهیه کنندگانی را می‌دهد که در همان شهری ساکن باشند که S1 ساکن است.
۳. این طرز نوشتن اصلاً در SQL قابل قبول نیست.
٤. شهر هایی را می‌دهد که تهیه کننده S1 در آن جا شعبه دارد.

۱۳- با کدام دستور SQL زیر می‌توان طول فیلد رشته‌ای را از ۳۰ به ٤۰ افزایش داد ؟

۱. Alter table … Modify…
۲. Alter table … Add …
۳. Alter table … Replace…
٤. اینکار بااین دستور امکان پذیر نمی‌باشد.

۱٤- اگر teacher نام جدول مشخصات اساتید و Sec جدول گروه درس (شامل فیلدهای شماره درس، شماره دانشجو، نام استاد و نمره) و نیز tname نام استاد باشد، آنگاه دستور SQL زیر چه می‌کند ؟

```
Select * from teacher where tname in
( select tname from sec where score = 10 And tname
not in  (" بهتاش "و "امیری ")) ;
```

۱. مشخصات اساتید امیری و بهتاش که نمره ۱۰ نداده‌اند را می‌دهد.

۲. مشخصات اساتیدی را می‌دهد که دانشجویان آنها از تمام دروس ۱۰ گرفته‌اند (به جز بهتاش و امیری)

۳. مشخصات اساتیدی به جز امیری و بهتاش که نمره ۱۰ داده‌اند را نمایش می‌دهد.

٤. مشخصات اساتیدی را می‌دهد که به دانشجویانی درس داده‌اند که نمره ۱۰ گرفته‌اند ولی جزو شاگردان بهتاش و امیری نمی‌باشند.

۱۵- کدام گزینه درست است ؟

۱. در SQL برای انجام یک عمل راه حل های متعددی وجود دارد واین یکی از مهمترین مزایای SQL است.

۲. هرگاه پرس و جوئی را در SQL هم بتوان با پیوند جداول انجام داد و هم با Select متداخل برای بالابردن کارایی بهتر است از Select متداخل استفاده کنیم.

۳. در SQL تابع Count (*) سطرهای NULL را نمی‌شمارد.

٤. در دستور select به جای between می‌توان از AND و به جای IN می‌توان از OR استفاده کرد.

۱٦- کدام گزینه نادرست است ؟

۱. SQL یک زبان مبدل است یعنی یک یه چند رابطه را به عنوان ورودی قبول می‌کند و نتیجه عملیات روی آنها همواره یک رابطه است.

۲. جبر رابطه‌ای خاصیت ((بسته بودن)) دارد. یعنی عملوند ها رابطه است و نتیجه نیز رابطه می‌باشد.

۳. به خاطر بسته بودن SQL و جبر رابطه‌ای می‌توان محاسبات تودرتو در هر دو داشت.

٤. SQL یک زبان procedural می‌باشد.

۱۷- کدامیک از زبان های نسل زیر، در جهت تقلید زبان طبیعی است ؟

۱. 3GL
۲. 4GL
۳. Access
٤. SGL

۱۸- برای ایجاد جدول پایه از چه دستوری در SQL استفاده می‌کنیم ؟

۱. create table
۲. create dbms
۳. struct table
٤. struct dbms

۱۹- دستور زیر در بانک اطلاعاتی تهیه کنندگان و قطعات چه می‌کند ؟

```
Select S#, status from S
Where city = "Tabriz"
Order by status desc
```

۱. شماره تهیه کنندگانی را می‌دهد که در شهر تبریز دارای وضعیت نزولی می‌باشند.

۲. شماره و وضعیت تهیه کنندگانی را می‌دهد که در شهر تبریز هستند، مرتب شده بر اساس وضعیت به صورت نزولی

۳. شماره و وضعیت تهیه کنندگانی را می‌دهد که در شهر تبریز هستند، مرتب شده بر اساس وضعیت به صورت صعودی

٤. وضعیت و شهر شرکتهائی را می‌دهد که در تبریز وضعیت نزولی دارند.

۲۰- دستور زیر در بانک اطلاعاتی تهیه کنندگان و قطعات چه می‌کند ؟

```
Update S Set status=3 * status
Where city = 'Tehran'
```

۱. تهیه کنندگانی که وضعیت سه برابر تولید کنندگان تهران دارند را حذف می‌کند.

۲. وضعیت تولیدکنندگان ساکن شهر تهران را سه برابر می‌کند.

۳. وضعیت تولیدکنندگان ساکن شهر تهران را برابر عدد ۳ می‌کند.

٤. شهرهایی که وضعیت سه برابر تهران دارند را به روز در می‌آورد.

۲۱- اگر S یک دید بر روی جدول پایه‌ای A باشد، برای آنکه هر گونه اصلاًحی روی دید S مجاز باشد کدام محدودیت باید رعایت شود ؟

۱. دیدS باید تمام فیلدهای NULL جدول A را داشته باشد.

۲. دید S باید شامل کلید اصلی جدول A باشد.

۳. دید S باید شامل کلید فرعی جدول A باشد.

٤. هیچ شرط خاصی برای‌این موضوع لازم نمی‌باشد.

۲۲- اگر عبارتی را بخواهیم، که با ABC شروع می‌شوند از کدام گزینه استفاده می‌کنیم؟

۱. starting ABC
۲. like ABC%
۳. like ABC$
٤. like ABC*

۲۳- دو جدول S (x , y , z , w) و S'(x , y , z' , w') را در نظر گرفته و بگوئید پس از اجرای دستور زیر، دید V چه ستونهایی خواهد داشت ؟

```
Create View V AS
(Select x,y from S) union (select x,y from S')
```

۱. V (x , y , z , w , z' , w')
۲. V (z' , w)
۳. V (z , w , z' , w')
٤. V (x , y)

۲٤- در بانک اطلاعاتی تهیه کنندگان و قطعات، کدام دستور شماره قطعه تمام قطعاتی را که توسط بیش از یک تولید کننده تهیه شده‌اند را می‌دهد ؟

۱. select P# from SP group by S# having count (*)>1
۲. select P# from P ,SP where count (*)>1
۳. select P# from SP group by P# having count (*)>1
٤. select P# from S,P,SP where count (*)>1

۲۵- جدول y دارای ستون های price و code می‌باشد. کدام دستور شماره کالائی که بالاترین قیمت را دارد، چاپ می‌کند ؟

۱. select code from y where price = (select max(price)from y)
۲. select code from y where price = max(price)
۳. 2,1

٤. select code from y first , y second where price = max(price)

۲٦- در رابطه با دید تعریف شده زیر کدام گزینه درست می‌باشد ؟ (جدول پایه‌ای R دارای ستونهای x , y , z می‌باشد)

```
Create view K(x , y ,w )
AS select x , y , sum (z)
From R
```

۱. در دید K می‌توان عملیات ذخیره سازی را انجام داد ولی استقلال داده‌ای نقض می‌شود.
۲. چون select قسمت where ندارد عملیات ذخیره سازی عولرض نامطلوب خواهد داشت.
۳. در دید k نمی‌توان عملیات ذخیره سازی را انجام داد.
٤. در دید k می‌توان عملیات ذخیره سازی را انجام داد.

۲۷- دستور زیر در بانک اطلاعاتی تهیه کنندگان و قطعات چه می‌کند ؟

```
Select distinct S.city ,P.city
From (S Join SP using S# ) Join P using P#
```

۱. تمام زوج شهرهایی را می‌دهد که تهیه کننده شهر اول، قطعه‌ای را تهیه می‌کند که در شهر دوم انبار شده است.
۲. تمام زوج شهرهایی را می‌دهد که تهیه کننده و محل انبار یکی است.
۳. تمام زوج شهرهایی را می‌دهد که تهیه کننده شهر دوم، قطعه‌ای را تهیه می‌کند که در شهر اول انبار شده است.
٤. تمام زوج شهرهایی را می‌دهد که تهیه کننده و محل انبار یکی نباشند.

۲۸- دستور زیر چه می‌کند ؟

```
Select sname from S
Where S# not in (select S# from SP)
```

۱. نام تهیه کنندگانی را می‌دهد که تمام قطعات را تهیه می‌کنند.
۲. نام تهیه کنندگانی را می‌دهد که هیچ قطعه‌ای را تهیه نمی‌کنند.
۳. این دستور غلط است.

٤. نام تهیه کنندگانی را می‌دهد که حداقل یک قطعه را تهیه کرده‌اند.

۲۹- کدام گزینه درست است ؟

۱. دستور select S# from SP Group by S# having Count (S#)>3

شماره تهیه کنندگانی را می‌دهد که بیشتر از ۳ قطعه تهیه می‌کنند.

۲. دستور select S# from SP Group by S# having AVG (Qty)>200

شماره تهیه کنندگانی را می‌دهد که میانگین قطعه تهیه شده آنها بیشتر از ۲۰۰ باشد.

۳. هیچکدام

٤. هر دو

۳۰- جدول دانشجو (stud) شامل دو ستون نام (Sname) و معدل (m) می‌باشد. کدام دستور نام دانشجویانی را می‌دهد که معدل آنها از میانگین همه معدل ها بیشتر است ؟

۱. Select Sname from stud where m> AVG (m)
۲. Select Sname from stud where m> (select AVG (m) from stud)
۳. هیچکدام
٤. هردو

۳۱- کدام دستور سطرهایی از جدول S را می‌دهد که ستون city آن ناشناخته نمی‌باشد؟

۱. select * from S where city is not NULL
۲. select * from S where not (city is NULL)
۳. 2,1
٤. select * from S where city = not NULL

۳۲- Group by در SQL پیاده سازی کدام عملگر در جبر رابطه‌ای است ؟

۱. Restrict
۲. Summarize
۳. extend
٤. union

۳۳- عملگر IN در SQL معادل کدام عملگر است ؟

۱. =ANY
۲. <ANY
۳. ALL
٤. = ALL

۳٤- کدام نوع جدول همواره قابل دسترسی است ولی ممکن است بهنگام سازی در آن امکان پذیر نباشد ؟

۱. جدول اصلی
۲. جدول مجازی
۳. جدول موقتی
٤. هرسه گزینه

۳۵- یک سیستم بانک اطلاعاتی را کاملا رابطه‌ای (Fully relational) گوییم اگر:

۱. مبتنی بر مفاهیم و ساختار رابطه‌ای بوده و در DSL آن، همان زبان جبر رابطه‌ای باشد.
۲. مبتنی بر مفاهیم و ساختار رابطه‌ای بوده و در DSL آن، همان زبان محاسبات رابطه‌ای باشد.
۳. مبتنی بر مفاهیم و ساختار رابطه‌ای بوده و در DSL آن حداقل هم توان زبان محاسبات رابطه‌ای باشد.
٤. مبتنی بر مفاهیم و ساختار رابطه‌ای بوده و در DSL آن حداقل هم توان زبان جبر رابطه‌ای باشد.

پاسخ تست های سری ۸

۱- (۱) SQL زبان پرس و جوی ساختیافته و قوی برای‌ایجاد، جستجو و دستکاری داده‌های یک بانک می‌باشد.

۲- (٤) دستور select فقط برای جستجو در بانک است و تغییری را در آن‌ایجاد نمی‌کند. تنها دستوراتی که باعث تغییر در بانک می‌شوند می‌توانند احتمالا باعث ناسازگاری داده‌ها شوند. دستور INSERT برای درج، DELETE برای حذف و UPDATE برای تغییر است.

۳- (۲) شرط جلوی Where می‌گویید شهر هر سه (تولیدکننده-قطعه و پروژه) باید یکی باشد.

٤- (۳)

۵- (۱) البته ظاهرا گزینه ۲ نیز درست می‌باشد.

٦- (۱) به جای علامت ٪ هر کاراکتر یا مجموعه‌ای از کاراکترها می‌تواند قرار گیرد مثل * موجود در DOS

۷- (٤) اگر توجه کنید خروجی اشتراک بین دو ستون Name در دو جدول S و T می‌باشد برای یافتن اشتراک دو جدول از Intersect استفاده می‌کنیم.

۸- (۳) اگر توجه کنید خروجی تفاضل S-T می‌باشد یعنی چه نام هایی در جدول S هست که در جدول T نیست.

۹- (۱) اولین قدم در پردازش‌این تقاضااین است که عبارتی که V را تعریف می‌کند به جای V قرار دهیم:

```
((((S JION SP) where P# = 'P2') {S# , City })
Where city = 'London') {S#}
```

این عبارت به صورت زیر خلاصه می‌شود:

```
 ((S Where city = 'London') Join
(SP Where P# = 'P2')) {S#}
```

۱۰- (۱)

۱۱- (۳)

۱۲- (۲) ابتدا select داخلی اجرا شده و شهر S1 را می‌دهد و سپس select بیرونی اجرا می‌شود و تهیه کنندگانی را می‌دهد که همشهری S1 می‌باشند.

۱۳- (۱) مثلا ALTER table student Modify (Lname char (40))

۱٤- (۳)

۱۵- (٤)

گزینه ۱: اینکه در SQL یک عمل را می‌توان به صورت های مختلف انجام داد یکی از معایب SQL به حساب می‌آید چرا که برنامه نویس را بااین مساله درگیر می‌کند که کدام راه حل کاراتر است.

گزینه ۲: پیوند جداول کاراتر از select متداخل است.

گزینه ۳: Count (*) برای شمارش سطرهای جداول استفاده می‌شود دراین تابع نمی‌توان از Distinct استفاده کرد و سطرهای NULL را هم می‌شمارد.

۱٦- (٤) SQL یک زبان محاوره‌ای و غیر رویه‌ای (non- procedural) است که در آن برنامه نویس تنها می‌گوید چه می‌خواهد ولی روش بدست آوردن جواب را معین نمی‌سازد. در رابطه با گزینه (۱) ممکن است بگوئید خروجی می‌تواند یک عدد تنها باشد. ولی همین عدد تنها به عنوان خروجی یک دستور SQL، در واقع رابطه‌ای است که یک سطر و یک ستون دارد.

۱۷- (۲)

۱۸- (۱)

۱۹- (۲)

۲۰- (۲)

۲۱- (۲)

۲۲- (۲)

۲۳- (٤)

۲٤- (۳)

۲۵- (۱)

۲٦- (۳) اگر در دید K تغییراتی دهیم که ستون W آن (که جمع ستون z جدول پایه‌ای R می‌باشد)تغییر کند، آنگاه DBMS نمی‌تواند تغییرات W را به Z اعمال کند.

۲۷- (۱)

۲۸- (۲)

۲۹- (٤)

۳۰- (۲) باید توجه داشت که گزینه ۱ نادرست است و جواب غلطی می‌دهد. چرا که تا هر جای جدول که پیش می‌رود، مقدار تدریجی AVG(m) عددی غلط را نشان می‌دهد در صورتی که اگر مثلا دانشجوی سطر دهم معدلش ازاین عدد بیشتر باشد، در خروجی ظاهر می‌شود. در گزینه ۲ ابتدا select داخلی اجراء شده و نتیجه آن که یک عدد است استخراج می‌گردد.

۳۱- (۳) توجه کنید که دو عبارت زیر معادل یکدیگرند:

< نام ستون > is not NULL $\Longleftrightarrow$ NOT (< نام ستون > is NULL)

۳۲- (۲)

۳۳- (۱)

۳٤- (۲)

۳۵- (۱)

تست های سری ۹

۱- کدام گزینه در رابطه با ((وابستگی تابعی)) درست است ؟

۱. 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 زیر مجموعه‌های X وابستگی تابعی نداشته باشد.

۲. صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد اگر و فقط اگر در طول حیات رابطه به هر مقدار X حداقل یک مقدار Y متناظر باشد.

۳. هر دو

٤. هیچکدام

۲- اگر رابطه R= {A,B,C,D,E,F,G} دارای FD های زیر باشد:

F= { AF→BE , FC→DE , F→CD , D→E , C→A}

کلید کاندید این رابطه کدام است ؟

۱. (A,F,C,D)
۲. (F,A)
۳. (F,A,G)
٤. (F,G)

۳- کدام مجموعه FD ها کاهش ناپذیرند ؟

۱. F = { A→BC , A→D , A→E }
۲. F = {AB→C , A→B , A→D , A→E }
۳. F = {A→B , A→C , A→D , A→E }
٤. F = {A→A , A→B , A→C , A→D , A→E }

٤- در رابطه R = { A,B,C,D } وابستگی های زیر وجود دارد:

F= {A→BE , B→C , A→B , AB→C , AC→D}

مجموعه کاهش ناپذیر (کمینه) از وابستگی ها کدام است ؟

۱. { A→B , B→C , A→D }
۲. { A→B , A→C , A→D }
۳. { B→C , B→D , A→C }
٤. { A→B , B→D , D→C}

۵- رابطه R = { A,B,C,D,E,F,G} دارای وابستگی های تابعی زیر است:

F= {A→B , BC→DE , AEF→G }

کدام وابستگی از آن قابل استنتاج است ؟

۱. ACF→DG
۲. AC→DE
۳. EF→G
٤. ۱و۲

٦- کدام گزینه در رابطه با دو مجموعه از FDهای زیر برای رابطه R={A,B,C,D,E} درست است ؟

F= {A→B , AB→C, D→AC , D→E }
G= {A→BC , D→AE }

۱. G و F هم ارزند.

۲. G و F هم ارز نیستند.

۳. G مجموعه کمینه F می‌باشد.

٤. F مجموعه بهینه G می‌باشد.

۷- کدام گزینه درست است ؟

۱. برای هر مجموعه از FD ها، حداکثر یک مجموعه هم ارز وجود دارد که کاهش ناپذیر است.

۲. برای هر مجموعه از FD ها، حداقل یک مجموعه هم ارز وجود دارد که کاهش ناپذیر است.

۳. برای هر مجموعه از FD ها، دقیقا و فقط یک مجموعه هم ارز وجود دارد که کاهش ناپذیر است.

٤. هیچکدام

۸- اگر برای رابطه R= { A,B,C,D,E,F,G,H,I,J} وابستگی های زیر را داشته باشیم کدام گزینه درست خواهد بود ؟

F= {ABC→E ,AB→G , B→F , C→J , CJ→I , G→H}

۱. F یک مجموعه کاهش پذیر است و {A,B,C,D} کلید کاندید است.

۲. F یک مجموعه کاهش ناپذیر است و {A,B,C,D} کلید کاندید است.

۳. F یک مجموعه کاهش ناپذیر است و {A,B,C,D,G} کلید کاندید است.

٤. F یک مجموعه کاهش پذیر است و {A,B,C} کلید کاندید است.

۹- رابطه TIMETABLE را با صفات زیر در نظر بگیرید:

دوره زمانی در روز (۱ تا ۸)= P و شماره روز هفته = D

شماره درس = L نام معلم =T شماره کلاس = C

فرض کنید هر درس در دوره خاصی (D,P) ارائه می‌شود و شماره دروس منحصر به فرد می‌باشند. کلید کاندیداین رابطه کدام است ؟

۱. L
۲. DPC
۳. DPT

٤. هرسه گزینه

۱۰- مجموعه کاهش ناپذیر مجموعه وابستگی زیر کدام است ؟

F={A→D , A→C , CD→B , AD→C}

۱. {CD→B , AD→C}
۲. {A→D , A→C , AD→C}
۳. {A→D , CD→B , AD→C}
٤. {A→D , A→C , CD→B }

۱۱- از مجموعه FD زیر کدام رابطه را نمی‌توان استنتاج کرد ؟

F= {A→D, CD→B , AD→C}

۱. C→B
۲. A→B
۳. A→C
٤. A→CD

۱۲- کدام گزینه درست است ؟

۱. به مجموعه تمام وابستگی های استنتاج شده، مجموعه پوششی می‌گوئیم.
۲. اگر مجموعه پوششی دو مجموعه FD با هم برابر باشند آنگاه دو مجموعه وابستگی با هم معادلند.
۳. قوانین ۳ گانه آرمسترانگ برای بدست آوردن مجموعه پوششی کافی است.
٤. هر سه گزینه

۱۳- نمودار وابستگی تابعی:

۱. نموداری است که جداول را تجزیه می‌کند.
۲. نموداری است که شروع کار آن با کلیدهای کاندید است.
۳. نموداری است که شروع کار آن با کلیدهای خارجی است.
٤. نموداری است که وابستگی به کلید خارجی را نمایش می‌دهد.

۱٤- وابستگی تابعی بدیهی کدام است ؟

۱. اگر سمت چپ آن، زیر مجموعه‌ای از سمت راست باشد.
۲. اگر سمت چپ آن قابل تجزیه شدن نباشد.
۳. اگر سمت راست آن فقط یک صفت خاصه باشد.
٤. اگر سمت راست آن، زیر مجموعه‌ای از سمت چپ باشد.

۱۵- در رابطه R = {A,B,C,D,E,F,G} با FD های زیر کلید کاندید کدام است ؟
F= {D→A , E→B , A→B , AB→EFC}

۱. (A,B,D)
۲. (D)
۳. (D,G)
٤. (B,D,E)

۱٦- کدام گزینه نادرست است ؟

۱. در رابطه تمام کلید، بین اجزای کلید می‌تواند وابستگی تابعی وجود داشته باشد.
۲. اگر K سوپر کلید رابطه R باشد آنگاه مجموعه عنوان R با K وابستگی تابعی دارد.
۳. از AB→C در حالت کلی نمی‌توان نتیجه گرفت که B→C و A→C
٤. FD ها محدودیت جامعیت را نشان می‌دهند.

۱۷- رابطه REL(X,Y,Z) را در نظر می‌گیریم. کدام یک از گزاره‌های زیر نادرست است؟

۱. اگر X→Y و Y→X داریم:X→Z
۲. رابطه REL حاصل JOIN دو رابطه REL1(X,Y) و REL2(X,Z) است اگر و فقط اگر داشته باشیم: X→Y
۳. اگر Y→X و Z→X داریم: (Y,Z)→X
٤. اگر X→Y و X→Z داریم: X→(Y,Z)

پاسخ تست های سری ۹

۱- (۱)صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد و می‌نویسیم R.X→R.Y اگر و فقط اگر در طول حیات رابطه به هر مقدار از X در رابطه دقیقا یک مقدار Y از رابطه R متناظر باشد.

۲- (٤) ابتدا FOPT را بدست می‌آوریم:
AF→BE ⇨AF→B , AF→E

$FC \rightarrow DE \Rightarrow FC \rightarrow D , FC \rightarrow E$
$F \rightarrow CD \Rightarrow F \rightarrow C , F \rightarrow D$
$F \rightarrow C , FC \rightarrow DE \Rightarrow F \rightarrow DE \Rightarrow F \rightarrow D , F \rightarrow E$
$F \rightarrow C , C \rightarrow A \Rightarrow F \rightarrow A$
$F \rightarrow A , FA \rightarrow B \Rightarrow F \rightarrow B$
$F \rightarrow A , FA \rightarrow E \Rightarrow F \rightarrow E$
$FOPT = \{F \rightarrow A, F \rightarrow B, F \rightarrow C, F \rightarrow D, F \rightarrow E, D \rightarrow E, C \rightarrow A\}$

پس F همه صفتهای دیگر به جزG را می‌دهد پس (F,G) کلید کاندید است.این کلید منحصر به فرد است زیرا هیچ صفتی F و G را نمی‌دهد. یعنی در هر کلید کاندید دیگر وجوداین دو صفت لازم است.

۳- (۳) در گزینه ۱ در $A \rightarrow BC$ سمت راست یک صفت تنها نیست.

در گزینه ۲: $A \rightarrow B , AB \rightarrow C \Rightarrow A \rightarrow C$ یعنی سمت چپ قابل کاهش است.

در گزینه ٤: $A \rightarrow A$ قابل حذف شدن است و عضو زائدی است.

٤- (۱)

$A \rightarrow BC \Rightarrow A \rightarrow B , A \rightarrow C$
$A \rightarrow B , AB \rightarrow C \Rightarrow A \rightarrow C$
$A \rightarrow C , AC \rightarrow D , A \rightarrow D$
$\Rightarrow F=\{A \rightarrow B , A \rightarrow C , B \rightarrow C , A \rightarrow B , A \rightarrow C , A \rightarrow D\}$
$F=\{A \rightarrow B , A \rightarrow C , B \rightarrow C , A \rightarrow D\}$

از آنجا که:

$A \rightarrow B , B \rightarrow C \Rightarrow A \rightarrow C$

پس $Fmin = \{A \rightarrow B , B \rightarrow C , A \rightarrow D\}$

٥- (٤)

٦- (۱)

مجموعه F: $A \rightarrow B , AB \rightarrow C \Rightarrow A \rightarrow C$

$D \rightarrow AC \Rightarrow D \rightarrow A , D \rightarrow C$
$F= \{A \rightarrow B , A \rightarrow C , D \rightarrow A , D \rightarrow C , D \rightarrow E \}$
$\Rightarrow Fmin = \{A \rightarrow B , A \rightarrow C , D \rightarrow A , D \rightarrow E \}$
مجموعه G $\rightarrow Gmin= \{A \rightarrow B , A \rightarrow C , D \rightarrow A , D \rightarrow E \}$

۷- (۲)

۸- (۱)

مجموعه کاهش پذیر است چرا که $C \rightarrow J , CJ \rightarrow I \Rightarrow C \rightarrow I$

{A,B,C,D,G,J} یک فوق کلید است (مجموعه‌ای از تمام صفاتی که در سمت چپ FD ها وجود دارد). به دلیل وجود $C \rightarrow J$ می‌توانیم J را از مجموعه فوق حذف کنیم. به دلیل وجود $AB \rightarrow G$ می‌توانیم G را از مجموعه حذف کنیم.چون A و B و C وD هیچکدام در سمت راست FD ها ظاهر نشده‌اند پس {A,B,C,D} کلید کاندید است.

۹- (٤)

$L \rightarrow T , L \rightarrow C , L \rightarrow D , L \rightarrow P$

اگر مثلا بگوئیم استاد اکبری سه شنبه ۸ تا ۱۰ صبح، آنگاه شماره درس و شماره کلاس او مشخص می‌گردد پس $DPT \rightarrow CL$. همچنین اگر مثلا بگوئیم سه شنبه ۸ تا ۱۰ صبح کلاس شماره ۲۱۵، آنگاه نام استاد و شماره درس تعیین می‌گردد، پس $DPC \rightarrow TL$

۱۰- (٤)

$A \rightarrow D , AD \rightarrow C \Rightarrow A \rightarrow C$

$Fmin = \{ A \rightarrow D , A \rightarrow C , CD \rightarrow B \}$

۱۱- (۱)

$A \rightarrow D , AD \rightarrow C \Rightarrow A \rightarrow C$

$A \rightarrow D , A \rightarrow C \Rightarrow A \rightarrow CD$

$A \rightarrow CD , CD \rightarrow B \Rightarrow A \rightarrow B$

۱۲- (٤)

۱۳- (۲)

۱٤- (٤) مثلا $AB \rightarrow A$ گزینه‌های ۲ و ۳ ویژگی های وابستگی تابعی کمینه را بیان می‌کنند.

۱۵- (۳)

$D \rightarrow A , A \rightarrow B \Rightarrow D \rightarrow B ,$

$D \rightarrow A , D \rightarrow B \Rightarrow D \rightarrow AB$

$D \rightarrow AB , AB \rightarrow EFC \Rightarrow D \rightarrow E , D \rightarrow F , D \rightarrow C$

۱٦- (۱) در رابطه تمام کلید، بین اجزای کلید وابستگی تابعی وجود ندارد.

۱۷- (۲)

تست های سری ۱۰

۱- فرض کنید لیستی از درس های دانشجویان در اختیار داریم که از ستونهای زیر تشکیل شده است:

تعداد واحد نام درس کد درس نام دانشجو کد دانشجو

هر دانشجو ممکن است چندین درس داشته باشد. اگر بخواهیم این اطلاعات را در یک بانک اطلاعاتی به گونه ای ذخیره کنیم که حداقل تکرار اطلاعات را داشته باشیم حداقل به چند جدول (table) نیاز داریم ؟

۱. ۱

۲. ۳
۳. ۲
٤. ٤

۲- کدامیک از گزاره های زیر در رابطه با ((عمده‌ترین اهداف نرمال ترسازی رابطه‌ها)) نادرست است ؟

۱. کاهش بعضی انواع افزونگی (Redundency)
۲. کاهش سربار (Overhead) سیستم در پاسخگوئی در عمل بازیابی
۳. تسهیل در اعمال بعضی محدودیتهای جامعیتی (integrity constraints)
٤. اجتناب از بعضی از انواع آنومالی در عملیات روی پایگاه

۳- کدام گزینه از اهداف کلی فرایند نرمال سازی (normalization) نیست ؟

۱. تسهیل اعمال بعضی از قواعد جامعیتی (integrity rules)
۲. تسهیل پیاده سازی دید کاربر (user view)
۳. حذف بعضی از انواع قواعد وابستگی تابعی (functional dependency)
٤. کاهش بعضی از انواع افزونگی (redundancy)

٤- کدام گزینه نادرست است ؟

۱. رابطه 1NF را همواره می توان به تعدادی رابطه 2NF تبدیل کرد.
۲. در تبدیل رابطه 1NF به 2NF باید چنان عمل کرد که وابستگی های تابعی غیر کامل موجود در رابطه 1NF از میان بروند.
۳. همیشه پس از تجزیه یک رابطه به دو رابطه و پیوند مجدد روابط حاصله ، لزوما به رابطه نخستین می رسیم.
٤. با انجام عمل پیوند طبیعی بر روی روابط 2NF می توان روابط 1NF را بدست آورد.

۵- کدام گزینه در رابطه با 2NF درست است ؟

۱. فقط در درج ممکن است آنومالی داشته باشیم.

۲. فقط در حذف ممکن است آنومالی داشته باشیم.
۳. در حذف و بهنگام سازی ممکن است آنومالی داشته باشیم.
٤. در درج ، حذف و بهنگام سازی ممکن است آنومالی داشته باشیم.

٦- کدام تعریف در رابطه با صورتهای نرمال نادرست است ؟

۱. رابطه‌ای نرمال است که تمام مقادیر صفات خاصه اش اتومیک باشند.
۲. رابطه‌ای 3NF است اگر و فقط اگر صفات خاصه غیر کلید متقابلا به یکدیگر ناوابسته باشند و همچنین با کلید اصلی رابطه ، وابستگی تابعی کامل داشته باشند.
۳. رابطه‌ای 2NF است اگر و فقط اگر 1NF باشد و همچنین اگر صفات خاصه غیر کلید متقابلا به یکدیگر ناوابسته باشند.
٤. رابطه‌ای 3NF است اگر و فقط اگر 2NF بوده و هر صفت خاصه غیر کلید به طور مستقیم (بی واسطه) با کلید اصلی وابستگی داشته باشد.

۷- کدام گزینه قضیه هیث (Heath) را بیان می کند ؟ (رابطه R را با سه صفت خاصه A و B و C در نظر بگیرید)

۱. اگر R.A→R.B همیشه می توان این رابطه را به دو رابطه R2(A,C) و R1(A,B) تجزیه کرد به نحوی که گمشدگی اطلاعات پیش نیاید.
۲. اگرR.A→R.B و R.A→R.C می توان این رابطه را به رابطه R2(A,C) و R1(A,B) تجزیه کرد.
۳. اگر R.A→R.B همیشه می توان این رابطه را به دو رابطه R2(A,C) و R1(A,B) تجزیه کرد ولی ممکن است گمشدگی اطلاعات پیش آید.
٤. R.A→R.B و R.A→R.C ممکن است نتوان این رابطه را به دو رابطه R2(A,C) و R1(A,B) تجزیه کرد.

۸- در کدام مورد زیر 3NF مطلوبترین صورت نیست ؟

۱. وقتی رابطه دارای چند کلید کاندید باشد.

۲. وقتی که کلیدهای کاندید رابطه مرکب باشند.
۳. وقتی که کلیدهای کاندید با یکدیگر اشتراک صفت خاصه داشته باشند.
٤. هر سه مورد.

9- در کدام حالت جدول 3NF ممکن است مشکل داشته باشد ؟
۱. جدول دارای حداقل دو کلید کاندید باشد.
۲. کلیدهای کاندید ترکیبی باشند.
۳. کلیدهای کاندید ترکیبی ، صفتهای مشترکی داشته باشند.
٤. وقتی که هر سه شرط فوق برقرار باشد.

10- کدام گزینه اهداف کلی نرمال سازی را بیان نمی کند ؟
۱. رعایت قواعد جامعیتی
۲. کاهش افزونگی
۳. کاهش سربار سیستم و ساده کردن دید کاربر
٤. کاهش بعضی از آنومالی ها

۱۱- کدام گزینه درست است ؟
۱. رابطه R در سطح 3NF است اگر و فقط اگر 2NF باشد و هر صفت غیر کلید با کلید اصلی وابستگی تابعی کامل داشته باشد.
۲. رابطهR در سطح 3NF است اگر و فقط اگر 2NF باشد و صفات غیر کلید آن متقابلا به یکدیگر ناوابسته باشند.
۳. رابطهR در سطح 3NF است اگر و فقط اگر 2NF باشد و وابستگی انتقالی (transitive) نداشته باشد.
٤. ۲و۳

12- صفت عمده Prime Attribute کدام است ؟
۱. صفتی که جزء تشکیل دهنده کلید باشد.
۲. همان صفت غیر کلید است.

۳. همان کلید است.

٤. هیچکدام

13- کدام گزینه نادرست است ؟

۱. ممکن است در مرحله 2NF کردن یک رابطه ، رابطه 1NF به بیش از دو رابطه تجزیه شود.

۲. یک صفت غیر کلید ممکن است بتواند با یک صفت غیر کلید دیگر وابستگی تابعی ناتام (غیر کانل)داشته باشد.

۳. در رابطه دو گانی (باینری) BCNF است.

٤. هیچکدام

14- تعریف BCNF کدام است ؟

۱. رابطه‌ای BCNF است اگر در آن هر دترمینان ، کلید کاندید باشد.

۲. رابطه R در BCNF است هرگاه یک وابستگی تابعی به صورت A→B (این وابستگی نامهم نباشد) در مجموعه FD های R وجود داشته باشد ، A سوپر کلید R باشد.

۳. رابطه‌ای BCNF است اگر و فقط اگر سمت چپ هر FD مهم و کاهش ناپذیر ، کلید کاندید رابطه باشد.

٤. هر سه گزینه

۱۵- رابطه R(SID,CID,PID) یعنی (شماره استاد ، شماره درس ، شماره دانشجو)R را در نظر بگیرید. قواعد سمانتیک زیر در این رابطه وجود دارد: الف) یک دانشجو ، یک درس را با یک استاد انتخاب می کند. ب) یک استاد یک درس را تدریس می کند. ج) یک درس توسط چند استاد تدریس می شود. این رابطه در چه سطح نرمال بودن قرار دارد ؟

۱. 2NF
۲. 3NF
۳. BCNF
٤. 4NF

۱۶- رابطه‌ای 2NF است که:

۱. 1NF بوده و صفتهای آن به زیر مجموعه های کلید اصلی وابستگی نداشته باشد.
۲. 1NF بوده و وابستگی انتقالی (تعدی) نداشته باشد.
۳. ۱و۲
٤. هیچکدام

۱۷- در رابطه R={A,B,C,D} ، مجموعه FD های زیر را داریم:

$A \rightarrow D$, $D \rightarrow B$, $A \rightarrow C$

سطح نرمال بودن این رابطه و تجزیه مطلوب آن کدام است ؟

۱. 3NF ، R1(D,B) ، R2(A,D,C}
۲. BCNF ، R1(A,D) ، R2(A,B,C}
۳. 3NF ، R1(A,C,D) ، R2(A,D}
٤. 2NF ، R1(D,B) ، R2(A,C,D}

۱۸- با حذف کلید وابسته ، چه صورتی از فرم نرمال ایجاد می شود ؟

۱. 1NF
۲. 2NF
۳. 3NF
٤. 4NF

۱۹- با حذف ابر کلید ، چه صورتی از فرم نرمال ایجاد می شود ؟

۱. 1NF
۲. 2NF
۳. 3NF
٤. 4NF

۲۰- جامع ترین نرمال سازی بر مبنای وابستگی تابعی توسط کدام صورت نرمال ارائه می شود ؟

۱. 1NF
۲. 2NF
۳. 3NF
٤. BCNF

۲۱- با حذف وابستگی های چند مقداری (MVD) چه صورتی از فرم نرمال ایجاد می شود ؟

۱. 2NF
۲. 3NF
۳. 4NF
٤. BCNF

۲۲- در تبدیل..... به..... آن را چنان به پرتوهایی تجزیه می کنیم که تمام وابستگی های پیوندی که از کلیدهای کاندید ناشی نمی شود حذف گردد.

۱. 3NF-2NF
۲. 4NF-3NF
۳. BCNF-3NF
٤. 5NF-4NF

۲۳- برای اینکه یک پایگاه‌داده ای رابطه‌ای بتواند به درستی در نرم‌افزار متداول طراحی شود ، آن پایگاه‌داده ها باید حداقل بر اساس چه فرم نرمال (Normal Form) قرار گرفته باشد ؟

۱. اول
۲. دوم
۳. سوم
٤. چهارم

۲٤- اگر جدولی یک فیلد داشته باشد که بستگی به قسمتی از کلید اصلی داشته ، این نکته نقض کدامیک از قوانین فرم نرمال است ؟

۱. فرم اول نرمال
۲. فرم دوم نرمال
۳. فرم سوم نرمال
٤. فرم چهارم نرمال

۲۵- اگر ۲ فیلد غیر کلیدی بر یکدیگر بستگی داشته باشند ، این نقض کدامیک از قوانین فرم های نرمال است ؟

۱. اولین فرم نرمال
۲. دومین فرم نرمال
۳. سومین فرم نرمال

٤. چهارمین فرم نرمال

۲٦- به طور کلی چگونه تاثیری از عملیات نرمالیزه کردن بر روی کارکرد بانک اطلاعاتی به وجود می آید ؟

۱. انعطاف کمتری در ذخیره سازی داده ها دارد.
۲. انعطاف بیشتری در ذخیره سازی داده ها دارد.
۳. انعطافی در ذخیره سازی داده ها ندارد.
٤. هیچکدام

۲۷-وقتی در یک جدول تعدادی ستون برای ذخیره سازی موردهای مختلف از یک داده وجود دارد مانند ...,phonenumber 2 , phonenumber1 این نکته کدام یک از موارد فرمهای نرمال را نقض می کند ؟

۱. فرم اول نرمال
۲. فرم دوم نرمال
۳. فرم سوم نرمال
٤. فرم چهارم نرمال

۲۸- چه دلیلی دارد که اطلاعات یک پایگاه اطلاعاتی را بین چندین فایل توزیع نمائیم؟

۱. برای جلوگیری از خرابی داده ها
۲. برای اینکه در بانکهای بزرگ سیستم عامل دچار مشکل نشود.
۳. برای مدیریت بهتر داده ها
٤. برای بهبود سرعت دستیابی به داده ها

۲۹- جدول در.... است که الف) همه کلیدهای آن تعریف شده باشند ب) تمام صفتهای آن به کلید اصلی وابستگی تابعی داشته باشند ج) صفتهای آن از دامنه تودرتو (nested domain) نباشند.

۱. 1NF
۲. 2NF
۳. 3NF
٤. 4NF

۳۰- در کدام فرم نرمال وابستگی انتقالی (transitive) وجود ندارد ؟

۱. 2NF
۲. 3NF,2NF
۳. 4NF,3NF
٤. 2NF

۳۱- جدول دانشجو (Stud) در یک موسسه آموزشی کوچک که دانشجوی همنام ندارد را در نظر بگیرید:

Stud (S# , Sname , address , avg)

S#شماره دانشجو ، Sname نام دانشجو ، Address آدرس دانشجو و avg معدل است.

این جدول حداکثر در چه سطح نرمال قرار دارد ؟

۱. 2NF
۲. 3NF
۳. 1NF
٤. BCNF

۳۲- کدام گزینه در رابطه با نرمال سازی درست است ؟

۱. اگر طراح تشخیص دهد که تجزیه یک جدول هر چند افزونگی هم داشته باشد ، باعث پائین آمدن سرعت اکثر پرس و جوها می شود ، مجاز است از نرمالتر سازی آن صرف نظر کند.
۲. روابط 3NF ی که می توانند تبدیل به BCNF شوند را حتما باید BCNF کرد
۳. در تجزیه رابطه 3NF به روابط BCNF هیچگاه اطلاعات و یا وابستگی‌هایی گم نمی شود.
٤. هر سه گزینه

۳۳- جدول زیر حداکثر در چه سطح نرمالی قرار دارد ؟

نان کتاب	نام درس	شهر دانشکده	نام دانشکده	نام استاد
هربرت شیلد	برنامه سازی	تهران	کامپیوتر	امیری
هربرت شیلد	برنامه سازی	اراک	برق	امیری
لیپ شوتز	ساختمان داده	تهران	کامپوتر	امیری
لیپ شوتز	ساختمان داده	اراک	برق	امیری
دیت	پایگاه‌داده	تهران	کامپیوتر	امیری

دیت پایگاه‌داده اراک برق امیری

۱. 2NF
۲. 3NF
۳. BCNF
٤. 4NF

پاسخ تست های سری ۱۰

۱- (۲) این سه جدول عبارتند از:

جدول ۱: جدول دانشجو: شامل فیلدهای کد دانشجو و نام دانشجو

جدول۲: جدول درس: شامل فیلدهای کد درس ، نام درس و تعداد واحد

جدول ۳: جدول دانشجو- درس: شامل فیلدهای کد دانشجو و کدهای درس

۲- (۲) از آنجا که نرمال تر سازی باعث تجزیه جداول می شود گاهی اوقات برای بازیابی نیاز به پیوند مجدد این جداول داریم که باعث افزایش سربار سیستم می گردد.

۳- (۲) تسهیل پیاده سازی دید کاربر از اهداف کلی فرآیند نرمال سازی زیرا در برخی مواقع این کار باعث مشکل تر شدن پیاده سازی دید کاربر نیز می شود.

٤- (۳) همیشه پس از تجزیه یک رابطه به دو رابطه و پیوند مجدد رابطه های حاصله لزوما به رابطه نخستین نمی رسیم.

۵- (٤) در رابطه 2NF ممکن است آنومالیهایی در درج ، حذف و بهنگام سازی داشته باشیم.

٦- (۳) گزینه های ۱ و ۲ درست می باشند. رابطه‌ای 2NF است اگر و فقط اگر 1NF باشد و هر صفت خاصه غیر کلید با کلید اصلی وابستگی تابعی کامل داشته باشد. توجه کنید تعریف گزینه ٤ برای 3NF معادل تعریف گزینه ۲ می باشد.

۷- (۱)

۸- (٤)

۹- (٤)

۱۰- (۳) نرمال سازی از آنجا که باعث تجزیه جداول می شود دید کاربر را به علت تعدد جداول و ارتباط بین آنها ممکن است مشکل سازد همچنین از آنجا که برای بازیابی بعضی پرس و جوها نیاز به پیوند جدولها خواهد بود ممکن است بار سیستم را افزایش دهد.

۱۱- (٤) در گزینه ۱ وقتی که رابطه در 2NF باشد ، خود به خود هر صفت خاصه غیر کلید با کلید اصلی وابستگی تابعی کامل خواهد داشت.

۱۲- (۱)

۱۳- (٤)

۱٤- (٤) کتابهای مختلف عبارات متفاوتی را برای تعریف حالات نرمال به کار می برند که همگی آنها در معنای نهایی معادل یکدیگر می باشند.

۱۵- (۲)

۱٦- (۳)

۱۷- (٤)

۱۸- (۳)

۱۹- (۲)

۲۰- (٤)

۲۱- (۳)

۲۲- (٤)

۲۳- (۳)

۲٤- (۲)

۲۵- (۳)

۲٦- (۱) نرمال سازی با اعمال محدودیتهایی جلوی یکسری افزونگی و نیز آنومالیها را می گیرد.

۲۷- (٤)

۲۸- (۳) این در واقع همان بحث نرمال سازی است ، ما داه ها را به جای ذخیره در یک جدول در چند جدول ذخیره می سازیم تا آنومالیها و مشکلات را کاهش دهیم. این عمل بر عکس سرعت عملیات را کاهش می دهد. پس گزینه ٤ غلط است.

۲۹- (۱)

۳۰- (۳) جدولی در 3NF است که الف) در 2NF باشد ب) وابستگی انتقالی نداشته باشد. وابستگی انتقالی یعنی وابستگی دو صفت غیر کلیدی به هم زیرا هر یک از آنها به کلید اصلی وابسته‌اند. رابطه‌ای که 4NF باشد حتما 3NF استع و ویژگیهای آن را دارد.

۳۱- (٤) S# و Sname کلید کاندید می باشند و S#→Stud , Sname →Stud پس چون هر دترمینان فقط کلید کاندید می باشد در سطح BCNF قرار دارد.

۳۲- (۱) در بعضی موارد نباید روابط 3NF را به BCNF تبدیل کرد پس گزینه ۲ نادرست است. همچنین مواردی ایت که تجزیه روابط 3NF به روابط BCNF وابستگی ها را از بین می برد پس گزینه ۳ نیز نادرست است.

۳۳- (۳) جدول مذکور در سطح BCNF است زیرا هیچگونه وابستگی به جز وابستگی به کلید اصلی ، که شامل تمام ستونها است ، وجود ندارد. ولی جدول فوق افزونگی دارد ، چرا که دانشکده های استاد امیری و نیز دروسی که تدریس کرده است تکرار شده است. برای رفع این مشکل و تبدیل آن به 4NF می توان آن را به دو جدول زیر تجزیه کرد:

نام کتاب	نام درس	نام استاد
هربرت شیلد	برنامه سازی	امیری
لیپ شوتز	ساختمان داده	امیری
دیت	پایگاه‌داده	امیری

شهر دانشکده	نام دانشکده	نام استاد
تهران	کامپیوتر	امیری
اراک	برق	امیری

ضمیمه ۲

SQL 99 _ BNF

BNF Grammar for ISO/IEC 9075:1999

Database Language SQL (SQL-99)

==========

--p
Note that this version of this file includes the corrections from ISO 9075:1999/Cor.1:2000.
--/p
--p
The plain text version of this grammar is
--## sql-99.bnf .
--/p
--hr
--h2 Key SQL Statements and Fragments
--/h2
--bl
--li ALTER DOMAIN <alter domain statement>
--li ALTER TABLE <alter table statement>
--li CLOSE cursor <close statement>
--li Column definition <column definition>
--li COMMIT WORK <commit statement>
--li CONNECT <connect statement>
--li CREATE ASSERTION <assertion definition>
--li CREATE CHARACTER SET <character set definition>
--li CREATE COLLATION <collation definition>
--li CREATE DOMAIN <domain definition>
--li CREATE FUNCTION <schema function>
--li CREATE PROCEDURE <schema procedure>
--li CREATE SCHEMA <schema definition>
--li CREATE TABLE <table definition>
--li CREATE TRANSLATION <translation definition>
--li CREATE TRIGGER <trigger definition>
--li CREATE VIEW <view definition>
--li Data type <data type>
--li DEALLOCATE PREPARE <deallocate prepared statement>
--li DECLARE cursor <declare cursor> <dynamic declare cursor>
--li DECLARE LOCAL TEMPORARY TABLE <temporary table declaration>
--li DELETE <delete statement: positioned> <delete statement: searched> <dynamic delete statement: positioned>
--li DESCRIBE <describe statement>
--li DESCRIPTOR statements <system descriptor statement>
--li DISCONNECT <disconnect statement>

--li EXECUTE <execute statement>
--li EXECUTE IMMEDIATE <execute immediate statement>
--li FETCH cursor <fetch statement>
--li FROM clause <from clause>
--li GET DIAGNOSTICS <get diagnostics statement>
--li GRANT <grant statement>
--li GROUP BY clause <group by clause>
--li HAVING clause <having clause>
--li INSERT <insert statement>
--li Literals <literal>
--li OPEN cursor <open statement>
--li ORDER BY clause <order by clause>
--li PREPARE <prepare statement>
--li REVOKE <revoke statement>
--li ROLLBACK WORK <rollback statement>
--li SAVEPOINT <savepoint statement>
--li Search condition <search condition> <regular expression>
--li SELECT <query specification>
--li SET CATALOG <set catalog statement>
--li SET CONNECTION <set connection statement>
--li SET CONSTRAINTS <set constraints mode statement>
--li SET NAMES <set names statement>
--li SET SCHEMA <set schema statement>
--li SET SESSION AUTHORIZATION <set session user identifier statement>
--li SET TIME ZONE <set local time zone statement>
--li SET TRANSACTION <set transaction statement>
--li SQL Client MODULE <SQL-client module definition>
--li UPDATE <update statement: positioned> <update statement: searched> <dynamic update statement: positioned>
--li Value expression <value expression>
--li WHERE clause <where clause>
--/bl
--hr
--h2 Identifying the version of SQL in use
--/h2
--p
This material (starting with <SQL object identifier>) is defined in section 6.3 "Object Identifier for Database Language SQL" of ISO/IEC 9075-1:1999 (SQL Framework).
It is used to express the capabilities of an implementation. The package names are identifiers such as 'PKG001', equivalent to 'Enhanced datetime facilities', as defined in the informative Annex B to SQL Framework. Each such package identifies a number of features that are provided when the SQL object identifier claims to provide the package.
--/p
<SQL object identifier>::= <SQL provenance> <SQL variant>
<SQL provenance>::= <arc1> <arc2> <arc3>
<arc1>::=iso | 1 | iso <leftparen> 1 <rightparen>
<arc2>::=standard | 0 | standard <leftparen> 0 <rightparen>
<arc3>::=9075
<SQL variant>::= <SQL edition> <SQL conformance>
<SQL edition>::= <1987> | <1989> | <1992> | <1999>
<1987>::=0 | edition1987 <leftparen> 0 <rightparen>
<1989>::= <1989 base> <1989 package>
<1989 base>::=1 | edition1989 <leftparen> 1 <rightparen>
<1989 package>::= <integrity no> | <integrity yes>
<integrity no>::=0 | IntegrityNo <leftparen> 0 <rightparen>
<integrity yes>::=1 | IntegrityYes <leftparen> 1 <rightparen>
<1992>::=2 | edition1992 <leftparen> 2 <rightparen>
<1999>::=3 | edition1999 <leftparen> 3 <rightparen>
<SQL conformance>::= <level> <parts> <packages>

<level>::= <low> | <intermediate> | <high>
<low>::=0 | Low <leftparen> 0 <rightparen>
<intermediate>::=1 | Intermediate <leftparen> 1 <rightparen>
<high>::=2 | High <leftparen> 2 <rightparen>
<parts>::= <Part 3> <Part 4> <Part 5> <Part 6> <Part 7> <Part 8> <Part 9> <Part 10>
--p
--small
--i
The parenthesized (i) and (n) are italic in the SQL standard. It is not clear exactly what this should look like, despite all the Information. However, it is also not important; this is not really a part of the SQL language per se. Note that the package numbers are PKG001 to PKG009, for example. We still have to devise a mechanism to persuade bnf2yacc.pl to ignore this information.
--/i
--/small
--/p
<packages>::= <Package PKG(i)>...
<Part (n)>::= <Part (n) no> | <Part (n) yes>
<Part (n) no>::=0 | Part-(n)No <leftparen> 0 <rightparen>
<Part (n) yes>::=!! (as specified in ISO/IEC 9075-(n))
<Package PKG(i)>::= <Package PKG(i)Yes> | <Package PKG(i)No>
<Part 3 yes>::= <Part 3 conformance>
<Part 3 conformance>::=3 | sqlcli1999 <leftparen> 3 <rightparen>
<Part 4 yes>::= <Part 4 conformance> <Part 4 module>
<Part 4 conformance>::=4 | sqlpsm1999 <leftparen> 4 <rightparen>
<Part 4 module>::= <Part 4 module yes> | <Part 4 module no>
<Part 4 module yes>::=1 | moduleyes <leftparen> 1 <rightparen>
<Part 4 module no>::=0 | moduleno <leftparen> 0 <rightparen>
<Part 5 yes>::= <Part 5 conformance> <Part 5 direct> <Part 5 embedded>
--p
--small
--i
The original used sqlbindings199x, but the x should clearly be a 9.
--/i
--/small
--/p
<Part 5 conformance>::=5 | sqlbindings1999 <leftparen> 5 <rightparen>
<Part 5 direct>::= <Part 5 direct yes> | <Part 5 direct no>
<Part 5 direct yes>::=1 | directyes <leftparen> 1 <rightparen>
<Part 5 direct no>::=0 | directno <leftparen> 0 <rightparen>

<Part 5 embedded>::= <Part 5 embedded no> | <Part 5 embedded languages>...
<Part 5 embedded no>::=0 | embeddedno <leftparen> 0 <rightparen>
<Part 5 embedded languages>::=
<Part 5 embedded Ada>
| <Part 5 embedded C>
| <Part 5 embedded COBOL>
| <Part 5 embedded Fortran>
| <Part 5 embedded MUMPS>
| <Part 5 embedded Pascal>
| <Part 5 embedded PL/I>
<Part 5 embedded Ada>::=1 | embeddedAda <leftparen> 1 <rightparen>
<Part 5 embedded C>::=2 | embeddedC <leftparen> 2 <rightparen>
<Part 5 embedded COBOL>::=3 | embeddedCOBOL <leftparen> 3 <rightparen>
<Part 5 embedded Fortran>::=4 | embeddedFortran <leftparen> 4 <rightparen>
<Part 5 embedded MUMPS>::=5 | embeddedMUMPS <leftparen> 5 <rightparen>
<Part 5 embedded Pascal>::=6 | embeddedPascal <leftparen> 6 <rightparen>
<Part 5 embedded PL/I>::=7 | embeddedPLI <leftparen> 7 <rightparen>
--hr
--h2 Basic Definitions of Characters Used, Tokens, Symbols, Etc.

--/h2
--p
Most of this section would normally be handled within the lexical analyzer rather than in the grammar proper. Further, the original document does not quote the various single characters, which makes it hard to process automatically.
--/p
<SQL terminal character>::= <SQL language character>
<SQL language character>::= <simple Latin letter> | <digit> | <SQL special character>
<simple Latin letter>::=
<simple Latin upper case letter>
| <simple Latin lower case letter>
<simple Latin upper case letter>::=
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
<simple Latin lower case letter>::=
a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
<digit>::=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<SQL special character>::=
<space>
| <double quote>
| <percent>
| <ampersand>
| <quote>
| <leftparen>
| <rightparen>
| <asterisk>
| <plus sign>
| <comma>
| <minus sign>
| <period>
| <solidus>
| <colon>
| <semicolon>
| <less than operator>
| <equals operator>
| <greater than operator>
| <question mark>
| <left bracket>
| <right bracket>
| <circumflex>
| <underscore>
| <vertical bar>
| <left brace>
| <right brace>
<space>::=!! (See the Syntax Rules)
<double quote>::="
<percent>::=%
<ampersand>::=&
<quote>::='
<leftparen>::=(
<rightparen>::=)
<asterisk>::=*
<plus sign>::=+
<comma>::=,
<minus sign>::=-
<period>::=.
<solidus>::=/
<colon>::=:
<semicolon>::=;
<less than operator>::= <

```
<equals operator>::==
<greater than operator>::=>
<question mark>::=?
<left bracket>::=[
<right bracket>::=]
<circumflex>::=^
<underscore>::=_
<vertical bar>::=|
<left brace>::={
<right brace>::=}
--hr
--h2 Literal Numbers, Strings, Dates and Times
--/h2
--p
--small
--i <SQL-client module definition> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <SQL-client module definition>::=
                              <module name clause>
                              <language clause>
                              <module authorization clause>
                             [ <module path specification> ]
                             [ <module transform group specification> ]
                             [ <temporary table declaration>... ]
                              <module contents>...
 <module name clause>::=
                             MODULE [ <SQL-client module name> ] [ <module character set specification>
]
 <SQL-client module name>::= <identifier>
 <identifier>::= <actual identifier>
 <actual identifier>::= <regular identifier> | <delimited identifier>
 <regular identifier>::= <identifier body>
--p
--small
--i <identifier body> modified per ISO 9075:1999/Cor.1:2000(E).
--/i
--/p
--p
--i <identifier body> also rationalized by removing curly brackets around <identifier part> because they
are unnecessary and inconsistent with other places where '...' modifies a single non-terminal.
--/i
--/small
--/small
--/p
 <identifier body>::= <identifier start> [ <identifier part>... ]
 <identifier start>::= <initial alphabetic character> | <ideographic character>
 <initial alphabetic character>::=!! (See the Syntax Rules)
 <ideographic character>::=!! (See the Syntax Rules)
 <identifier part>::=
                              <alphabetic character>
                |             <ideographic character>
                |             <decimal digit character>
                |             <identifier combining character>
                |             <underscore>
                |             <alternate underscore>
                |             <extender character>
                |             <identifier ignorable character>
```

| <connector character>
<alphabetic character>::=!! (See the Syntax Rules)
<decimal digit character>::=!! (See the Syntax Rules)
<identifier combining character>::=!! (See the Syntax Rules)
<alternate underscore>::=!! (See the Syntax Rules)
<extender character>::=!! (See the Syntax Rules)
<identifier ignorable character>::=!! (See the Syntax Rules)
<connector character>::=!! (See the Syntax Rules)
<delimited identifier>::= <double quote> <delimited identifier body> <double quote>
<delimited identifier body>::= <delimited identifier part>...
<delimited identifier part>::= <nondoublequote character> | <doublequote symbol>
<nondoublequote character>::=!! (See the Syntax Rules)
--p
--small
--i
Note that the two successive double quote characters must have no other character (such as a space) between them. The lexical analyzer would normally deal with this sort of issue.
--/i
--/small
--/p
<doublequote symbol>::= <double quote> <double quote>
<module character set specification>::=NAMES ARE <character set specification>
<character set specification>::=
<standard character set name>
| <implementation-defined character set name>
| <user-defined character set name>
<standard character set name>::= <character set name>
<character set name>::=[<schema name> <period>] <SQL language identifier>
<schema name>::=[<catalog name> <period>] <unqualified schema name>
<catalog name>::= <identifier>
<unqualified schema name>::= <identifier>
<SQL language identifier>::=
<SQL language identifier start> [{ <underscore> | <SQL language identifier part> }...]
<SQL language identifier start>::= <simple Latin letter>
<SQL language identifier part>::= <simple Latin letter> | <digit>
<implementation-defined character set name>::= <character set name>
<user-defined character set name>::= <character set name>
<language clause>::=LANGUAGE <language name>
<language name>::=ADA | C | COBOL | FORTRAN | MUMPS | PASCAL | PLI | SQL
<module authorization clause>::=
SCHEMA <schema name>
| AUTHORIZATION <module authorization identifier>
| SCHEMA <schema name> AUTHORIZATION <module authorization identifier>
<module authorization identifier>::= <authorization identifier>
<authorization identifier>::= <role name> | <user identifier>
<role name>::= <identifier>
<user identifier>::= <identifier>
<module path specification>::= <path specification>
<path specification>::=PATH <schema name list>
<schema name list>::= <schema name> [{ <comma> <schema name> }...]
<module transform group specification>::= <transform group specification>
<transform group specification>::=
TRANSFORM GROUP { <single group specification> | <multiple group specification> }
<single group specification>::= <group name>
<group name>::= <identifier>
<multiple group specification>::= <group specification> [{ <comma> <group specification> }...]
<group specification>::= <group name> FOR TYPE <user-defined type>
<user-defined type>::= <user-defined type name>

<user-defined type name>::= <schema qualified type name>
<schema qualified type name>::=[<schema name> <period>] <qualified identifier>
<qualified identifier>::= <identifier>
<temporary table declaration>::=
DECLARE LOCAL TEMPORARY TABLE <table name>
<table element list>
[ON COMMIT <table commit action> ROWS]
<table name>::= <local or schema qualified name>
<local or schema qualified name>::=[<local or schema qualifier> <period>] <qualified identifier>
<local or schema qualifier>::= <schema name> | MODULE
<table element list>::=
<leftparen> <table element> [{ <comma> <table element> }...] <rightparen>
<table element>::=
<column definition>
| <table constraint definition>
| <like clause>
| <self-referencing column specification>
| <column options>
<column definition>::=
<column name>
{ <data type> | <domain name> }
[<reference scope check>]
[<default clause>]
[<column constraint definition>...]
[<collate clause>]
<column name>::= <identifier>
--hr
--h2 Data Types
--/h2
<data type>::=
<predefined type>
| <row type>
| <user-defined type>
| <reference type>
| <collection type>
<predefined type>::=
<character string type> [CHARACTER SET <character set specification>]
| <national character string type>
| <binary large object string type>
| <bit string type>
| <numeric type>
| <boolean type>
| <datetime type>
| <interval type>
<character string type>::=
CHARACTER [<leftparen> <length> <rightparen>]
| CHAR [<leftparen> <length> <rightparen>]
| CHARACTER VARYING <leftparen> <length> <rightparen>
| CHAR VARYING <leftparen> <length> <rightparen>
| VARCHAR <leftparen> <length> <rightparen>
| CHARACTER LARGE OBJECT [<leftparen> <large object length> <rightparen>]
| CHAR LARGE OBJECT [<leftparen> <large object length> <rightparen>]
| CLOB [<leftparen> <large object length> <rightparen>]
<length>::= <unsigned integer>
<unsigned integer>::= <digit>...
<large object length>::= <unsigned integer> [<multiplier>] | <large object length token>
<multiplier>::=K | M | G
<large object length token>::= <digit>... <multiplier>

```
<national character string type>::=
                    NATIONAL CHARACTER [ <leftparen> <length> <rightparen> ]
          |         NATIONAL CHAR [ <leftparen> <length> <rightparen> ]
          |         NCHAR [ <leftparen> <length> <rightparen> ]
          |         NATIONAL CHARACTER VARYING <leftparen> <length> <rightparen>
          |         NATIONAL CHAR VARYING <leftparen> <length> <rightparen>
          |         NCHAR VARYING <leftparen> <length> <rightparen>
          |         NATIONAL CHARACTER LARGE OBJECT [ <leftparen> <large object
length> <rightparen> ]
          |         NCHAR LARGE OBJECT [ <leftparen> <large object length> <rightparen> ]
          |         NCLOB [ <leftparen> <large object length> <rightparen> ]
<binary large object string type>::=
                    BINARY LARGE OBJECT [ <leftparen> <large object length> <rightparen> ]
          |         BLOB [ <leftparen> <large object length> <rightparen> ]
<bit string type>::=
                    BIT [ <leftparen> <length> <rightparen> ]
          |         BIT VARYING <leftparen> <length> <rightparen>
<numeric type>::= <exact numeric type> | <approximate numeric type>
<exact numeric type>::=
                    NUMERIC [ <leftparen> <precision> [ <comma> <scale> ] <rightparen> ]
          |         DECIMAL [ <leftparen> <precision> [ <comma> <scale> ] <rightparen> ]
          |         DEC [ <leftparen> <precision> [ <comma> <scale> ] <rightparen> ]
          |         INTEGER
          |         INT
          |         SMALLINT
<precision>::= <unsigned integer>
<scale>::= <unsigned integer>
<approximate numeric type>::=
                    FLOAT [ <leftparen> <precision> <rightparen> ]
          |         REAL
          |         DOUBLE PRECISION
<boolean type>::=BOOLEAN
<datetime type>::=
                    DATE
          |         TIME [<leftparen><time precision> <rightparen>] [<with or without time zone>]
          |         TIMESTAMP [ <leftparen> <timestamp precision> <rightparen> ] [ <with or
without time zone> ]
<time precision>::= <time fractional seconds precision>
<time fractional seconds precision>::= <unsigned integer>
<with or without time zone>::=WITH TIME ZONE | WITHOUT TIME ZONE
<timestamp precision>::= <time fractional seconds precision>
<interval type>::=INTERVAL <interval qualifier>
<interval qualifier>::= <start field> TO <end field> | <single datetime field>
<start field>::=
                     <non-second primary datetime field> [ <leftparen> <interval leading field
precision> <rightparen> ]
<non-second primary datetime field>::=YEAR | MONTH | DAY | HOUR | MINUTE
<interval leading field precision>::= <unsigned integer>
<end field>::=
                     <non-second primary datetime field>
          |         SECOND [ <leftparen> <interval fractional seconds precision> <rightparen> ]
<interval fractional seconds precision>::= <unsigned integer>
<single datetime field>::=
                     <non-second primary datetime field> [ <leftparen> <interval leading field
precision> <rightparen> ]
          |         SECOND [ <leftparen> <interval leading field precision> [ <comma> <interval
fractional seconds precision> ] <rightparen> ]
<row type>::=ROW <row type body>
<row type body>::= <leftparen> <field definition> [ { <comma> <field definition> }... ] <rightparen>
```

<field definition>::= <field name> <data type> [<reference scope check>] [<collate clause>]
<field name>::= <identifier>
<reference scope check>::=
REFERENCES ARE [NOT] CHECKED [ON DELETE <reference scope check action>]
<reference scope check action>::= <referential action>
<referential action>::=
CASCADE
| SET NULL
| SET DEFAULT
| RESTRICT
| NO ACTION
<collate clause>::=COLLATE <collation name>
<collation name>::= <schema qualified name>
<schema qualified name>::=[<schema name> <period>] <qualified identifier>
<reference type>::=REF <leftparen> <referenced type> <rightparen> [<scope clause>]
<referenced type>::= <user-defined type>
<scope clause>::=SCOPE <table name>
<collection type>::= <data type> <array specification>
<array specification>::=
<collection type constructor> <left bracket or trigraph> <unsigned integer> <right bracket or trigraph>
<collection type constructor>::=ARRAY
<left bracket or trigraph>::= <left bracket> | <left bracket trigraph>
--p
--small
--i
The trigraphs are strictly sequences of characters, not sequences of tokens. There may not be any spaces between the characters. Normally, the lexical analyzer would return the trigraphs as a simple symbol.
--/i
--/small
--/p
<left bracket trigraph>::= <question mark> <question mark> <leftparen>
<right bracket or trigraph>::= <right bracket> | <right bracket trigraph>
<right bracket trigraph>::= <question mark> <question mark> <rightparen>
<domain name>::= <schema qualified name>
<default clause>::=DEFAULT <default option>
<default option>::=
<literal>
| <datetime value function>
| USER
| CURRENT_USER
| CURRENT_ROLE
| SESSION_USER
| SYSTEM_USER
| CURRENT_PATH
| <implicitly typed value specification>
--hr
--h2 Literals
--/h2
<literal>::= <signed numeric literal> | <general literal>
<signed numeric literal>::=[<sign>] <unsigned numeric literal>
<sign>::= <plus sign> | <minus sign>
<unsigned numeric literal>::= <exact numeric literal> | <approximate numeric literal>
<exact numeric literal>::=
<unsigned integer> [<period> [<unsigned integer>]]
| <period> <unsigned integer>
<approximate numeric literal>::= <mantissa> E <exponent>
<mantissa>::= <exact numeric literal>

<exponent>::= <signed integer>
<signed integer>::=[<sign>] <unsigned integer>
<general literal>::=
<character string literal>
| <national character string literal>
| <bit string literal>
| <hex string literal>
| <binary string literal>
| <datetime literal>
| <interval literal>
| <boolean literal>
<character string literal>::=
[<introducer> <character set specification>]
<quote> [<character representation>...] <quote>
[{ <separator> <quote> [<character representation>...] <quote> }...]
<introducer>::= <underscore>
<character representation>::= <nonquote character> | <quote symbol>
<nonquote character>::=!! (See the Syntax Rules.)
--p
--small
--i
The <quote symbol> rule consists of two immediately adjacent <quote> marks with no spaces. As usual, this would be best handled in the lexical analyzer, not in the grammar.
--/i
--/small
--/p
<quote symbol>::= <quote> <quote>
<separator>::={ <comment> | <white space> }...
<comment>::= <simple comment> | <bracketed comment>
<simple comment>::= <simple comment introducer> [<comment character>...] <newline>
<simple comment introducer>::= <minus sign> <minus sign> [<minus sign>...]
<comment character>::= <nonquote character> | <quote>
<newline>::=!! (See the Syntax Rules)
--p
--small
--i
The <bracketed comment> rule included '!! (See the Syntax Rules)'. This probably says something about the <slash> <asterisk> and <asterisk> <slash> needing to be adjacent characters rather than adjacent tokens.
--/i
--/small
--/p
<bracketed comment>::=
<bracketed comment introducer> <bracketed comment contents> <bracketed comment terminator>
<bracketed comment introducer>::= <slash> <asterisk>
<bracketed comment contents>::=[{ <comment character> | <separator> }...]
<bracketed comment terminator>::= <asterisk> <slash>
<white space>::=!! (See the Syntax Rules)
<national character string literal>::=
N <quote> [<character representation>...] <quote>
[{ <separator> <quote> [<character representation>...] <quote> }...]
<bit string literal>::=
B <quote> [<bit>...] <quote>
[{ <separator> <quote> [<bit>...] <quote> }...]
<bit>::=0 | 1
<hex string literal>::=
X <quote> [<hexit>...] <quote>
[{ <separator> <quote> [<hexit>...] <quote> }...]

```
<hexit>::= <digit> | A | B | C | D | E | F | a | b | c | d | e | f
<binary string literal>::=
                        X <quote> [ { <hexit>  <hexit> }... ] <quote>
                        [ { <separator>  <quote> [ { <hexit>  <hexit> }... ] <quote> }... ]
<datetime literal>::= <date literal> | <time literal> | <timestamp literal>
<date literal>::=DATE <date string>
<date string>::= <quote>  <unquoted date string>  <quote>
<unquoted date string>::= <date value>
<date value>::= <years value>  <minus sign>  <months value>  <minus sign>  <days value>
<years value>::= <datetime value>
<datetime value>::= <unsigned integer>
<months value>::= <datetime value>
<days value>::= <datetime value>
<time literal>::=TIME <time string>
<time string>::= <quote>  <unquoted time string>  <quote>
<unquoted time string>::= <time value> [ <time zone interval> ]
<time value>::= <hours value>  <colon>  <minutes value>  <colon>  <seconds value>
<hours value>::= <datetime value>
<minutes value>::= <datetime value>
<seconds value>::= <seconds integer value> [ <period> [ <seconds fraction> ] ]
<seconds integer value>::= <unsigned integer>
<seconds fraction>::= <unsigned integer>
<time zone interval>::= <sign>  <hours value>  <colon>  <minutes value>
<timestamp literal>::=TIMESTAMP <timestamp string>
<timestamp string>::= <quote>  <unquoted timestamp string>  <quote>
<unquoted timestamp string>::= <unquoted date string>  <space>  <unquoted time string>
<interval literal>::=INTERVAL [ <sign> ] <interval string>  <interval qualifier>
<interval string>::= <quote>  <unquoted interval string>  <quote>
<unquoted interval string>::=[ <sign> ] { <year-month literal> | <day-time literal> }
<year-month literal>::= <years value> | [ <years value>  <minus sign> ] <months value>
<day-time literal>::= <day-time interval> | <time interval>
<day-time interval>::=
                        <days value> [ <space>  <hours value> [ <colon>  <minutes value> [ <colon>
<seconds value> ] ] ]
<time interval>::=
                        <hours value> [ <colon>  <minutes value> [ <colon>  <seconds value> ] ]
            |           <minutes value> [ <colon>  <seconds value> ]
            |           <seconds value>
<boolean literal>::=TRUE | FALSE | UNKNOWN
<datetime value function>::=
                        <current date value function>
            |           <current time value function>
            |           <current timestamp value function>
            |           <current local time value function>
            |           <current local timestamp value function>
<current date value function>::=CURRENT_DATE
<current time value function>::=
                        CURRENT_TIME [ <leftparen> <time precision>  <rightparen>]
<current timestamp value function>::=
                        CURRENT_TIMESTAMP [ <leftparen>  <timestamp precision>  <rightparen> ]
<current local time value function>::=
                        LOCALTIME [ <leftparen>  <time precision>  <rightparen> ]
<current local timestamp value function>::=
                        LOCALTIMESTAMP [ <leftparen>  <timestamp precision>  <rightparen> ]
<implicitly typed value specification>::= <null specification> | <empty specification>
<null specification>::=NULL
<empty specification>::=ARRAY <left bracket or trigraph>  <right bracket or trigraph>
--hr
--h2 Constraints
```

```
--/h2
 <column constraint definition>::=
            [ <constraint name definition> ] <column constraint> [ <constraint  characteristics> ]
 <constraint name definition>::=CONSTRAINT <constraint name>
 <constraint name>::= <schema qualified name>
 <column constraint>::=
                        NOT NULL
            |           <unique specification>
            |           <references specification>
            |           <check constraint definition>
 <unique specification>::=UNIQUE | PRIMARY KEY
 <references specification>::=
                        REFERENCES <referenced table and columns>
                        [ MATCH <match type> ] [ <referential triggered action> ]
 <referenced table and columns>::=
                         <table name> [ <leftparen>  <reference column list>  <rightparen> ]
 <reference column list>::= <column name list>
 <column name list>::= <column name> [ { <comma>  <column name> }... ]
 <match type>::=FULL | PARTIAL | SIMPLE
 <referential triggered action>::=
                         <update rule> [ <delete rule> ]
            |            <delete rule> [ <update rule> ]
 <update rule>::=ON UPDATE <referential action>
 <delete rule>::=ON DELETE <referential action>
 <check constraint definition>::=CHECK <leftparen>  <search condition>  <rightparen>
--hr
--h2 Search Condition
--/h2
 <search condition>::= <boolean value expression>
 <boolean value expression>::=
                        <boolean term>
            |           <boolean value expression> OR <boolean term>
 <boolean term>::=
                        <boolean factor>
            |           <boolean term> AND <boolean factor>
 <boolean factor>::=[ NOT ] <boolean test>
 <boolean test>::= <boolean primary> [ IS [ NOT ] <truth value> ]
 <boolean primary>::=
                        <predicate>
            |           <parenthesized boolean value expression>
            |           <nonparenthesized value expression primary>
 <predicate>::=
                        <comparison predicate>
            |           <between predicate>
            |           <in predicate>
            |           <like predicate>
            |           <null predicate>
            |           <quantified comparison predicate>
            |           <exists predicate>
            |           <unique predicate>
            |           <match predicate>
            |           <overlaps predicate>
            |           <similar predicate>
            |           <distinct predicate>
            |           <type predicate>
 <comparison predicate>::= <row value expression>  <comp op>  <row value expression>
 <row value expression>::= <row value special case> | <row value constructor>
 <row value special case>::= <value specification> | <value expression>
 <value specification>::= <literal> | <general value specification>
```

<general value specification>::=
<host parameter specification>
| <SQL parameter reference>
| <SQL variable reference>
| <dynamic parameter specification>
| <embedded variable specification>
| CURRENT_DEFAULT_TRANSFORM_GROUP
| CURRENT_PATH
| CURRENT_ROLE
| CURRENT_TRANSFORM_GROUP_FOR_TYPE <user-defined type>
| CURRENT_USER
| SESSION_USER
| SYSTEM_USER
| USER
| VALUE
<host parameter specification>::= <host parameter name> [<indicator parameter>]
<host parameter name>::= <colon> <identifier>
<indicator parameter>::=[INDICATOR] <host parameter name>
<SQL parameter reference>::= <basic identifier chain>
<basic identifier chain>::= <identifier chain>
<identifier chain>::= <identifier> [{ <period> <identifier> }...]
<value expression>::=
<numeric value expression>
| <string value expression>
| <datetime value expression>
| <interval value expression>
| <boolean value expression>
| <user-defined type value expression>
| <row value expression>
| <reference value expression>
| <collection value expression>
<numeric value expression>::=
<term>
| <numeric value expression> <plus sign> <term>
| <numeric value expression> <minus sign> <term>
<term>::=
<factor>
| <term> <asterisk> <factor>
| <term> <solidus> <factor>
<factor>::=[<sign>] <numeric primary>
<numeric primary>::=
<value expression primary>
| <numeric value function>
<value expression primary>::=
<parenthesized value expression>
| <nonparenthesized value expression primary>
<parenthesized value expression>::= <leftparen> <value expression> <rightparen>
<nonparenthesized value expression primary>::=
<unsigned value specification>
| <column reference>
| <set function specification>
| <scalar subquery>
| <case expression>
| <cast specification>
| <subtype treatment>
| <attribute or method reference>
| <reference resolution>
| <collection value constructor>
| <routine invocation>

```
            |           <field reference>
            |           <element reference>
            |           <method invocation>
            |           <static method invocation>
            |           <new specification>
<unsigned value specification>::= <unsigned literal> | <general value specification>
<unsigned literal>::= <unsigned numeric literal> | <general literal>
<column reference>::=
                        <basic identifier chain>
            |           MODULE <period> <qualified identifier> <period> <column name>
<set function specification>::=
                        COUNT <leftparen> <asterisk> <rightparen>
            |           <general set function>
            |           <grouping operation>
<general set function>::=
                        <set function type> <leftparen> [ <set quantifier> ] <value expression>
<rightparen>
<set function type>::= <computational operation>
<computational operation>::=AVG | MAX | MIN | SUM | EVERY | ANY | SOME | COUNT
<set quantifier>::=DISTINCT | ALL
<grouping operation>::=GROUPING <leftparen> <column reference> <rightparen>
--hr
--h2 Queries
--/h2
<scalar subquery>::= <subquery>
<subquery>::= <leftparen> <query expression> <rightparen>
<query expression>::=[ <with clause> ] <query expression body>
<with clause>::=WITH [ RECURSIVE ] <with list>
<with list>::= <with list element> [ { <comma> <with list element> }... ]
<with list element>::=
                        <query name>
                        [ <leftparen> <with column list> <rightparen> ]
                        AS <leftparen> <query expression> <rightparen>
                        [ <search or cycle clause> ]
<query name>::= <identifier>
<with column list>::= <column name list>
<search or cycle clause>::=
                        <search clause>
            |           <cycle clause>
            |           <search clause> <cycle clause>
<search clause>::=
                        SEARCH <recursive search order> SET <sequence column>
<recursive search order>::=
                        DEPTH FIRST BY <sort specification list>
            |           BREADTH FIRST BY <sort specification list>
<sort specification list>::= <sort specification> [ { <comma> <sort specification> }... ]
--p
--small
--i <sort specification> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<sort specification>::= <sort key> [ <ordering specification> ]
<sort key>::= <value expression>
<ordering specification>::=ASC | DESC
<sequence column>::= <column name>
<cycle clause>::=
                        CYCLE <cycle column list>
                        SET <cycle mark column> TO <cycle mark value>
```

DEFAULT <non-cycle mark value>
USING <path column>
<cycle column list>::= <cycle column> [{ <comma> <cycle column> }...]
<cycle column>::= <column name>
<cycle mark column>::= <column name>
<cycle mark value>::= <value expression>
<non-cycle mark value>::= <value expression>
<path column>::= <column name>
<query expression body>::= <non-join query expression> | <joined table>
<non-join query expression>::=
<non-join query term>
| <query expression body> UNION [ALL | DISTINCT] [<corresponding spec>] <query term>
| <query expression body> EXCEPT [ALL | DISTINCT] [<corresponding spec>] <query term>
<non-join query term>::=
<non-join query primary>
| <query term> INTERSECT [ALL | DISTINCT] [<corresponding spec>] <query primary>
<non-join query primary>::=
<simple table>
| <leftparen> <non-join query expression> <rightparen>
<simple table>::= <query specification> | <table value constructor> | <explicit table>
<query specification>::=SELECT [<set quantifier>] <select list> <table expression>
<select list>::= <asterisk> | <select sublist> [{ <comma> <select sublist> }...]
<select sublist>::= <derived column> | <qualified asterisk>
<derived column>::= <value expression> [<as clause>]
<as clause>::=[AS] <column name>
<qualified asterisk>::=
<asterisked identifier chain> <period> <asterisk>
| <all fields reference>
<asterisked identifier chain>::= <asterisked identifier> [{ <period> <asterisked identifier> }...]
<asterisked identifier>::= <identifier>
<all fields reference>::= <value expression primary> <period> <asterisk>
<table expression>::= <from clause> [<where clause>] [<group by clause>] [<having clause>]
<from clause>::=FROM <table reference list>
<table reference list>::= <table reference> [{ <comma> <table reference> }...]
<table reference>::= <table primary> | <joined table>
<table primary>::=
<table or query name> [[AS] <correlation name> [<leftparen> <derived column list> <rightparen>]]
| <derived table> [AS] <correlation name> [<leftparen> <derived column list> <rightparen>]
| <lateral derived table> [AS] <correlation name> [<leftparen> <derived column list> <rightparen>]
| <collection derived table> [AS] <correlation name> [<leftparen> <derived column list> <rightparen>]
| <only spec> [[AS] <correlation name> [<leftparen> <derived column list> <rightparen>]]
| <leftparen> <joined table> <rightparen>
<table or query name>::= <table name> | <query name>
<correlation name>::= <identifier>
<derived column list>::= <column name list>
<derived table>::= <table subquery>
<table subquery>::= <subquery>
<lateral derived table>::=LATERAL <leftparen> <query expression> <rightparen>
<collection derived table>::=
UNNEST <leftparen> <collection value expression> <rightparen> [WITH ORDINALITY]

<collection value expression>::= <value expression primary>
<only spec>::=ONLY <leftparen> <table or query name> <rightparen>
<joined table>::= <cross join> | <qualified join> | <natural join> | <union join>
<cross join>::= <table reference> CROSS JOIN <table primary>
<qualified join>::= <table reference> [<join type>] JOIN <table reference> <join specification>
<join type>::=INNER | <outer join type> [OUTER]
<outer join type>::=LEFT | RIGHT | FULL
<join specification>::= <join condition> | <named columns join>
<join condition>::=ON <search condition>
<named columns join>::=USING <leftparen> <join column list> <rightparen>
<join column list>::= <column name list>
<natural join>::= <table reference> NATURAL [<join type>] JOIN <table primary>
<union join>::= <table reference> UNION JOIN <table primary>
<where clause>::=WHERE <search condition>
--p
--small
--i Rules from <group by clause> to <grouping set> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<group by clause>::=GROUP BY <grouping element list>
<grouping element list>::= <grouping element> [{ <comma> <grouping element> }...]
<grouping element>::=
<ordinary grouping set>
| <rollup list>
| <cube list>
| <grouping sets specification>
| <grand total>
<grouping column reference>::= <column reference> [<collate clause>]
<rollup list>::=ROLLUP <leftparen> <grouping column reference list> <rightparen>
<grouping column reference list>::=
<grouping column reference> [{ <comma> <grouping column reference> }...]
<cube list>::=CUBE <leftparen> <grouping column reference list> <rightparen>
<grouping sets specification>::=GROUPING SETS <leftparen> <grouping set list> <rightparen>
<grouping set list>::= <grouping set> [{ <comma> <grouping set> }...]
<grouping set>::=
<ordinary grouping set>
| <rollup list>
| <cube list>
| <grouping sets specification>
| <grand total>
<ordinary grouping set>::=
<grouping column reference>
| <leftparen> <grouping column reference list> <rightparen>
<grand total>::= <leftparen> <rightparen>
<concatenated grouping>::= <grouping set> <comma> <grouping set list>
<having clause>::=HAVING <search condition>
<table value constructor>::=VALUES <row value expression list>
<row value expression list>::= <row value expression> [{ <comma> <row value expression> }...]
<explicit table>::=TABLE <table name>
--hr
--h2 Query expression components
--/h2
<query term>::= <non-join query term> | <joined table>
<corresponding spec>::=
CORRESPONDING [BY <leftparen> <corresponding column list>
<rightparen>]
<corresponding column list>::= <column name list>
<query primary>::= <non-join query primary> | <joined table>

<case expression>::= <case abbreviation> | <case specification>
<case abbreviation>::=
NULLIF <leftparen> <value expression> <comma> <value expression> <rightparen>
| COALESCE <leftparen> <value expression> { <comma> <value expression> }... <rightparen>
<case specification>::= <simple case> | <searched case>
<simple case>::=CASE <case operand> <simple when clause>... [<else clause>] END
<case operand>::= <value expression>
<simple when clause>::=WHEN <when operand> THEN <result>
<when operand>::= <value expression>
<result>::= <result expression> | NULL
<result expression>::= <value expression>
<else clause>::=ELSE <result>
<searched case>::=CASE <searched when clause>... [<else clause>] END
<searched when clause>::=WHEN <search condition> THEN <result>
<cast specification>::=CAST <leftparen> <cast operand> AS <cast target> <rightparen>
<cast operand>::= <value expression> | <implicitly typed value specification>
<cast target>::= <domain name> | <data type>
--p
--small
--i <subtype treatment> to <target subtype> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<subtype treatment>::=
TREAT <leftparen> <subtype operand> AS <target subtype> <rightparen>
<subtype operand>::= <value expression>
<target subtype>::= <user-defined type>
<attribute or method reference>::=
<value expression primary> <dereference operator> <qualified identifier>
[<SQL argument list>]
<dereference operator>::= <right arrow>
<right arrow>::=->
<SQL argument list>::=
<leftparen>[<SQL argument>[{ <comma> <SQL argument> }...]] <rightparen>
<SQL argument>::= <value expression> | <generalized expression> | <target specification>
<generalized expression>::= <value expression> AS <user-defined type>
<target specification>::=
<host parameter specification>
| <SQL parameter reference>
| <column reference>
| <SQL variable reference>
| <dynamic parameter specification>
| <embedded variable specification>
<reference resolution>::=DEREF <leftparen> <reference value expression> <rightparen>
<reference value expression>::= <value expression primary>
<collection value constructor>::= <array value expression>
<array value expression>::= <array value constructor> | <array concatenation> | <value expression primary>
<array value constructor>::= <array value list constructor>
<array value list constructor>::=
ARRAY <left bracket or trigraph> <array element list> <right bracket or trigraph>
<array element list>::= <array element> [{ <comma> <array element> }...]
<array element>::= <value expression>
<array concatenation>::=
<array value expression 1> <concatenation operator> <array value expression 2>
<array value expression 1>::= <array value expression>
<concatenation operator>::=||

<array value expression 2>::= <array value expression>
<routine invocation>::= <routine name> <SQL argument list>
<routine name>::=[<schema name> <period>] <qualified identifier>
<field reference>::= <value expression primary> <period> <field name>
<element reference>::=
<array value expression> <left bracket or trigraph> <numeric value expression> <right bracket or trigraph>
<method invocation>::= <direct invocation> | <generalized invocation>
<direct invocation>::=
<value expression primary> <period> <method name> [<SQL argument list>]
<method name>::= <identifier>
<generalized invocation>::=
<leftparen> <value expression primary>
AS <data type> <rightparen> <period> <method name>
[<SQL argument list>]
--p
--small
--i It is not remotely clear why this was needed in this grammar.
--/i
--i <constructor method selection> added per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<constructor method selection>::= <routine invocation>
<static method invocation>::=
<user-defined type> <double colon> <method name> [<SQL argument list>]
--p
--small
--i
Note that <double colon> must be a pair of characters with no intervening space, not a pair of colon symbols separated by arbitrary white space. Normally, the lexical analyzer would return <double colon> as a symbol.
--/i
--/small
--/p
<double colon>::= <colon> <colon>
<new specification>::=NEW <routine invocation>
<numeric value function>::=
<position expression>
| <extract expression>
| <length expression>
| <cardinality expression>
| <absolute value expression>
| <modulus expression>
<position expression>::= <string position expression> | <blob position expression>
<string position expression>::=
POSITION <leftparen> <string value expression> IN <string value expression> <rightparen>
<string value expression>::= <character value expression> | <bit value expression> | <blob value expression>
<character value expression>::= <concatenation> | <character factor>
<concatenation>::= <character value expression> <concatenation operator> <character factor>
<character factor>::= <character primary> [<collate clause>]
<character primary>::= <value expression primary> | <string value function>
<string value function>::= <character value function> | <blob value function> | <bit value function>
<character value function>::=
<character substring function>
| <regular expression substring function>
| <fold>

```
            |           <form-of-use conversion>
            |           <character translation>
            |           <trim function>
            |           <character overlay function>
            |           <specific type method>
<character substring function>::=
                        SUBSTRING <leftparen>  <character value expression> FROM <start position>
                        [ FOR <string length> ] <rightparen>
<start position>::= <numeric value expression>
<string length>::= <numeric value expression>
--p
--small
--i <regular expression substring function> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<regular expression substring function>::=
                        SUBSTRING <leftparen>  <character value expression> SIMILAR
                         <character value expression> ESCAPE <escape character>  <rightparen>
<escape character>::= <character value expression>
<fold>::={ UPPER | LOWER } <leftparen>  <character value expression>  <rightparen>
<form-of-use conversion>::=
                        CONVERT <leftparen>  <character value expression>
                        USING <form-of-use conversion name>  <rightparen>
<form-of-use conversion name>::= <schema qualified name>
<character translation>::=
                        TRANSLATE <leftparen>  <character value expression>
                        USING <translation name>  <rightparen>
<translation name>::= <schema qualified name>
<trim function>::=TRIM <leftparen>  <trim operands>  <rightparen>
<trim operands>::=[ [ <trim specification> ] [ <trim character> ] FROM ] <trim source>
<trim specification>::=LEADING | TRAILING | BOTH
<trim character>::= <character value expression>
<trim source>::= <character value expression>
<character overlay function>::=
                        OVERLAY  <leftparen>   <character  value  expression>  PLACING  <character
value expression>
                        FROM <start position> [ FOR <string length> ] <rightparen>
<specific type method>::= <user-defined type value expression>  <period> SPECIFICTYPE
<user-defined type value expression>::= <value expression primary>
<blob value function>::= <blob substring function> | <blob trim function> | <blob overlay function>
<blob substring function>::=
                        SUBSTRING <leftparen>  <blob value expression> FROM <start position>
                        [ FOR <string length> ] <rightparen>
<blob value expression>::= <blob concatenation> | <blob factor>
<blob concatenation>::= <blob value expression>  <concatenation operator>  <blob factor>
<blob factor>::= <blob primary>
<blob primary>::= <value expression primary> | <string value function>
<blob trim function>::=TRIM <leftparen>  <blob trim operands>  <rightparen>
<blob trim operands>::=[ [ <trim specification> ] [ <trim octet> ] FROM ] <blob trim source>
<trim octet>::= <blob value expression>
<blob trim source>::= <blob value expression>
<blob overlay function>::=
                        OVERLAY  <leftparen>   <blob  value  expression>  PLACING  <blob  value
expression>
                        FROM <start position> [ FOR <string length> ] <rightparen>
<bit value function>::= <bit substring function>
<bit substring function>::=
                        SUBSTRING <leftparen>  <bit value expression> FROM <start position>
```

[FOR <string length>] <rightparen>
<bit value expression>::= <bit concatenation> | <bit factor>
<bit concatenation>::= <bit value expression> <concatenation operator> <bit factor>
<bit factor>::= <bit primary>
<bit primary>::= <value expression primary> | <string value function>
<blob position expression>::=
POSITION <leftparen> <blob value expression> IN <blob value expression> <rightparen>
<extract expression>::=
EXTRACT <leftparen> <extract field> FROM <extract source> <rightparen>
<extract field>::= <primary datetime field> | <time zone field>
<primary datetime field>::= <non-second primary datetime field> | SECOND
<time zone field>::=TIMEZONE_HOUR | TIMEZONE_MINUTE
<extract source>::= <datetime value expression> | <interval value expression>
<datetime value expression>::=
<datetime term>
| <interval value expression> <plus sign> <datetime term>
| <datetime value expression> <plus sign> <interval term>
| <datetime value expression> <minus sign> <interval term>
<interval term>::=
<interval factor>
| <interval term 2> <asterisk> <factor>
| <interval term 2> <solidus> <factor>
| <term> <asterisk> <interval factor>
<interval factor>::=[<sign>] <interval primary>
<interval primary>::= <value expression primary> | <interval value function>
<interval value function>::= <interval absolute value function>
<interval absolute value function>::=ABS <leftparen> <interval value expression> <rightparen>
<interval value expression>::=
<interval term>
| <interval value expression 1> <plus sign> <interval term 1>
| <interval value expression 1> <minus sign> <interval term 1>
| <leftparen> <datetime value expression> <minus sign>
<datetime term> <rightparen> <interval qualifier>
<interval value expression 1>::= <interval value expression>
<interval term 1>::= <interval term>
<datetime term>::= <datetime factor>
<datetime factor>::= <datetime primary> [<time zone>]
<datetime primary>::= <value expression primary> | <datetime value function>
<time zone>::=AT <time zone specifier>
<time zone specifier>::=LOCAL | TIME ZONE <interval primary>
<interval term 2>::= <interval term>
<length expression>::= <char length expression> | <octet length expression> | <bit length expression>
<char length expression>::=
{ CHAR_LENGTH | CHARACTER_LENGTH } <leftparen> <string value expression> <rightparen>
<octet length expression>::=
OCTET_LENGTH <leftparen> <string value expression> <rightparen>
<bit length expression>::=
BIT_LENGTH <leftparen> <string value expression> <rightparen>
<cardinality expression>::=
CARDINALITY <leftparen> <collection value expression> <rightparen>
<absolute value expression>::=
ABS <leftparen> <numeric value expression> <rightparen>
<modulus expression>::=
MOD <leftparen> <numeric value expression dividend> <comma> <numeric value expression divisor> <rightparen>
<numeric value expression dividend>::= <numeric value expression>
<numeric value expression divisor>::= <numeric value expression>

<row value constructor>::=
<row value constructor element>
| [ROW] <leftparen> <row value constructor element list> <rightparen>
| <row subquery>
<row value constructor element>::= <value expression>
<row value constructor element list>::=
<row value constructor element>[{<comma><row value constructor element>}...]
<row subquery>::= <subquery>
<comp op>::=
<equals operator>
| <not equals operator>
| <less than operator>
| <greater than operator>
| <less than or equals operator>
| <greater than or equals operator>
--p
--small
--i
The <not equals> , <less than or equals operator> and <greater than or equals operator> should be handled by the lexical analyzer as token symbols, not by the grammar. As usual, spaces are not allowed between the two characters.
--/i
--/small
--/p
<not equals operator>::= <less than operator> <greater than operator>
<less than or equals operator>::= <less than operator> <equals operator>
<greater than or equals operator>::= <greater than operator> <equals operator>
<between predicate>::=
<row value expression> [NOT] BETWEEN [ASYMMETRIC | SYMMETRIC]
<row value expression> AND <row value expression>
<in predicate>::= <row value expression> [NOT] IN <in predicate value>
<in predicate value>::= <table subquery> | <leftparen> <in value list> <rightparen>
--p
--small
--i Previously, the expression in curly braces was not in square brackets.
--/i
--i Consequently, every <in value list> had to have at least two items in it.
--/i
--/small
--/p
<in value list>::= <row value expression> [{ <comma> <row value expression> }...]
<like predicate>::= <character like predicate> | <octet like predicate>
<character like predicate>::=
<character match value> [NOT] LIKE <character pattern> [ESCAPE <escape character>]
<character match value>::= <character value expression>
<character pattern>::= <character value expression>
<octet like predicate>::=
<octet match value> [NOT] LIKE <octet pattern> [ESCAPE <escape octet>]
<octet match value>::= <blob value expression>
<octet pattern>::= <blob value expression>
<escape octet>::= <blob value expression>
<null predicate>::= <row value expression> IS [NOT] NULL
<quantified comparison predicate>::= <row value expression> <comp op> <quantifier> <table subquery>
<quantifier>::= <all> | <some>
<all>::=ALL

<some>::=SOME | ANY
<exists predicate>::=EXISTS <table subquery>
<unique predicate>::=UNIQUE <table subquery>
<match predicate>::=
<row value expression> MATCH [UNIQUE] [SIMPLE | PARTIAL | FULL]
<table subquery>
<overlaps predicate>::= <row value expression 1> OVERLAPS <row value expression 2>
<row value expression 1>::= <row value expression>
<row value expression 2>::= <row value expression>
<similar predicate>::=
<character match value> [NOT] SIMILAR TO <similar pattern> [ESCAPE <escape character>]
<similar pattern>::= <character value expression>
--hr
--h2 Regular Expressions for SIMILAR TO
--/h2
--p
These regular expressions are not referenced anywhere else in the document, but define the structure that the <character value expression> used in <similar pattern> must have. Structurally, these regular expressions are similar to 'egrep' expressions, except they use underscore in place of dot, and percent is equivalent to dot star in 'egrep'. The other omission is the use of caret (aka circumflex) to mark the start of the matched text and dollar to mark the end of the matched text.
--/p
<regular expression>::=
<regular term>
| <regular expression> <vertical bar> <regular term>
<regular term>::=
<regular factor>
| <regular term> <regular factor>
<regular factor>::=
<regular primary>
| <regular primary> <asterisk>
| <regular primary> <plus sign>
<regular primary>::=
<character specifier>
| <percent>
| <regular character set>
| <leftparen> <regular expression> <rightparen>
<character specifier>::= <non-escaped character> | <escaped character>
<non-escaped character>::=!! (See the Syntax Rules)
<escaped character>::=!! (See the Syntax Rules)
<regular character set>::=
<underscore>
| <left bracket> <character enumeration>... <right bracket>
| <left bracket> <circumflex> <character enumeration>... <right bracket>
| <left bracket> <colon> <regular character set identifier> <colon> <right bracket>
<character enumeration>::=
<character specifier>
| <character specifier> <minus sign> <character specifier>
<regular character set identifier>::= <identifier>
--hr
<distinct predicate>::=
<row value expression 3> IS DISTINCT FROM <row value expression 4>
<row value expression 3>::= <row value expression>
<row value expression 4>::= <row value expression>
<type predicate>::=
<user-defined type value expression> IS [NOT] OF <leftparen> <type list> <rightparen>

<type list>::=
<user-defined type specification> [{ <comma> <user-defined type specification> }...
]
<user-defined type specification>::=
<inclusive user-defined type specification>
| <exclusive user-defined type specification>
<inclusive user-defined type specification>::= <user-defined type>
<exclusive user-defined type specification>::=ONLY <user-defined type>
<parenthesized boolean value expression>::= <leftparen> <boolean value expression> <rightparen>
<truth value>::=TRUE | FALSE | UNKNOWN
--hr
--h2 More about constraints
--/h2
<constraint characteristics>::=
<constraint check time> [[NOT] DEFERRABLE]
| [NOT] DEFERRABLE [<constraint check time>]
<constraint check time>::=INITIALLY DEFERRED | INITIALLY IMMEDIATE
<table constraint definition>::=
[<constraint name definition>] <table constraint> [<constraint characteristics>]
<table constraint>::= <unique constraint definition> | <referential constraint definition> | <check constraint definition>
--p
--small
--i
The standard documents UNIQUE (VALUE) but there is no explanation of why that is different from the UNIQUE <leftparen> VALUE <rightparen> used here.
--/i
--/small
--/p
<unique constraint definition>::=
<unique specification> <leftparen> <unique column list> <rightparen>
| UNIQUE <leftparen> VALUE <rightparen>
<unique column list>::= <column name list>
<referential constraint definition>::=
FOREIGN KEY <leftparen> <referencing columns> <rightparen> <references specification>
<referencing columns>::= <reference column list>
<like clause>::=LIKE <table name>
<self-referencing column specification>::=
REF IS <self-referencing column name> <reference generation>
<self-referencing column name>::= <column name>
<reference generation>::=SYSTEM GENERATED | USER GENERATED | DERIVED
<column options>::= <column name> WITH OPTIONS <column option list>
<column option list>::=
[<scope clause>] [<default clause>] [<column constraint definition>...] [<collate clause>]
<table commit action>::=PRESERVE | DELETE
--hr
--h2 Module contents
--/h2
<module contents>::=
<declare cursor>
| <externally-invoked procedure>
| <dynamic declare cursor>
<declare cursor>::=
DECLARE <cursor name> [<cursor sensitivity>] [<cursor scrollability>] CURSOR
[<cursor holdability>] [<cursor returnability>] FOR <cursor specification>
<cursor name>::= <local qualified name>

```
<local qualified name>::=[ <local qualifier>  <period> ] <qualified identifier>
<local qualifier>::=MODULE
<cursor sensitivity>::=SENSITIVE | INSENSITIVE | ASENSITIVE
<cursor scrollability>::=SCROLL | NO SCROLL
<cursor holdability>::=WITH HOLD | WITHOUT HOLD
<cursor returnability>::=WITH RETURN | WITHOUT RETURN
<cursor specification>::= <query expression> [ <order by clause> ] [ <updatability clause> ]
<order by clause>::=ORDER BY <sort specification list>
<updatability clause>::=FOR { READ ONLY | UPDATE [ OF <column name list> ] }
--hr
--h2 SQL Procedures
--/h2
<externally-invoked procedure>::=
                        PROCEDURE <procedure name>
                        <host parameter declaration setup>  <semicolon>
                        <SQL procedure statement>  <semicolon>
<procedure name>::= <identifier>
--p
--small
--i <host parameter declaration setup> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<host parameter declaration setup>::= <host parameter declaration list>
<host parameter declaration list>::=
                        <leftparen>  <host parameter declaration>
                        [ { <comma>  <host parameter declaration> }... ] <rightparen>
<host parameter declaration>::=
                        <host parameter name>  <host parameter data type>
            |           <status parameter>
<host parameter data type>::= <data type> [ <locator indication> ]
<locator indication>::=AS LOCATOR
<status parameter>::=SQLSTATE
<SQL procedure statement>::= <SQL executable statement>
<SQL executable statement>::=
                        <SQL schema statement>
            |           <SQL data statement>
            |           <SQL control statement>
            |           <SQL transaction statement>
            |           <SQL connection statement>
            |           <SQL session statement>
            |           <SQL diagnostics statement>
            |           <SQL dynamic statement>
--hr
--h2 SQL Schema Definition Statements
--/h2
<SQL schema statement>::=
                        <SQL schema definition statement>
            |           <SQL schema manipulation statement>
--p
--small
--i <SQL schema definition statement> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<SQL schema definition statement>::=
                        <schema definition>
            |           <table definition>
            |           <view definition>
```

```
            |           <SQL-invoked routine>
            |           <grant statement>
            |           <role definition>
            |           <domain definition>
            |           <character set definition>
            |           <collation definition>
            |           <translation definition>
            |           <assertion definition>
            |           <trigger definition>
            |           <user-defined type definition>
            |           <user-defined cast definition>
            |           <user-defined ordering definition>
            |           <transform definition>
            |           <SQL-server module definition>
<schema definition>::=
                        CREATE SCHEMA <schema name clause> [ <schema character set or path> ]
                        [ <schema element>... ]
<schema name clause>::=
                        <schema name>
            |           AUTHORIZATION <schema authorization identifier>
            |           <schema name> AUTHORIZATION <schema authorization identifier>
<schema authorization identifier>::= <authorization identifier>
<schema character set or path>::=
                        <schema character set specification>
            |           <schema path specification>
            |           <schema character set specification>  <schema path specification>
            |           <schema path specification>  <schema character set specification>
<schema character set specification>::=
                        DEFAULT CHARACTER SET <character set specification>
<schema path specification>::= <path specification>
--p
--small
--i <schema element> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<schema element>::=
                        <table definition>
            |           <view definition>
            |           <domain definition>
            |           <character set definition>
            |           <collation definition>
            |           <translation definition>
            |           <assertion definition>
            |           <trigger definition>
            |           <user-defined type definition>
            |           <schema routine>
            |           <grant statement>
            |           <role definition>
            |           <user-defined cast definition>
            |           <user-defined ordering definition>
            |           <transform definition>
<table definition>::=
                        CREATE [ <table scope> ] TABLE <table name>  <table contents source>
                        [ ON COMMIT <table commit action> ROWS ]
<table scope>::= <global or local> TEMPORARY
<global or local>::=GLOBAL | LOCAL
<table contents source>::=
                        <table element list>
```

```
            |           OF <user-defined type> [ <subtable clause> ] [ <table element list> ]
 <subtable clause>::=UNDER <supertable clause>
 <supertable clause>::= <supertable name>
 <supertable name>::= <table name>
 <view definition>::=
                        CREATE [ RECURSIVE ] VIEW <table name>  <view specification>
                        AS <query expression> [ WITH [ <levels clause> ] CHECK OPTION ]
 <view specification>::= <regular view specification> | <referenceable view specification>
 <regular view specification>::=[ <leftparen>  <view column list>  <rightparen> ]
 <view column list>::= <column name list>
 <referenceable view specification>::=OF <user-defined type>[<subview clause>][<view element list>]
 <subview clause>::=UNDER <table name>
 <view element list>::=
                         <leftparen> [ <self-referencing column specification>  <comma> ]
                         <view element> [ { <comma>  <view element> }... ] <rightparen>
 <view element>::= <view column option>
 <view column option>::= <column name> WITH OPTIONS <scope clause>
 <levels clause>::=CASCADED | LOCAL
 <domain definition>::=
                        CREATE DOMAIN <domain name> [ AS ] <data type>
                        [ <default clause> ] [ <domain constraint>... ] [ <collate clause>]
 <domain constraint>::=
                        [ <constraint name definition> ] <check constraint definition> [ <constraint
characteristics> ]
 <character set definition>::=
                        CREATE CHARACTER SET <character set name>
                        [ AS ] <character set source> [ <collate clause> ]
 <character set source>::=GET <character set specification>
 <collation definition>::=
                        CREATE COLLATION <collation name> FOR <character set specification>
                        FROM <existing collation name> [ <pad characteristic> ]
 <existing collation name>::= <collation name>
 <pad characteristic>::=NO PAD | PAD SPACE
 <translation definition>::=
                        CREATE TRANSLATION <translation name> FOR <source character set
specification>
                        TO <target character set specification> FROM <translation source>
 <source character set specification>::= <character set specification>
 <target character set specification>::= <character set specification>
 <translation source>::= <existing translation name> | <translation routine>
 <existing translation name>::= <translation name>
 <translation routine>::= <specific routine designator>
--p
--small
--i <specific routine designator> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <specific routine designator>::=
                        SPECIFIC <routine type>  <specific name>
            |           <routine type>  <member name> [ FOR <user-defined type name> ]
--p
--small
--i <specific routine designator> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <routine type>::=
                        ROUTINE | FUNCTION | PROCEDURE
```

```
            |           [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD
<specific name>::= <schema qualified name>
<member name>::= <schema qualified routine name> [ <data type list> ]
<schema qualified routine name>::= <schema qualified name>
<data type list>::=
                        <leftparen> [ <data type> [ { <comma>  <data type> }... ] ] <rightparen>
<assertion definition>::=
                        CREATE ASSERTION <constraint name>
                        CHECK  <leftparen>  <search  condition>  <rightparen>  [  <constraint
characteristics> ]
<trigger definition>::=
                        CREATE TRIGGER <trigger name>  <trigger action time>  <trigger event>
                        ON <table name> [ REFERENCING <old or new values alias list> ] <triggered
action>
<trigger name>::= <schema qualified name>
<trigger action time>::=BEFORE | AFTER
<trigger event>::=INSERT | DELETE | UPDATE [ OF <trigger column list> ]
<trigger column list>::= <column name list>
<old or new values alias list>::= <old or new values alias>...
<old or new values alias>::=
                        OLD [ ROW ] [ AS ] <old values correlation name>
            |           NEW [ ROW ] [ AS ] <new values correlation name>
            |           OLD TABLE [ AS ] <old values table alias>
            |           NEW TABLE [ AS ] <new values table alias>
<old values correlation name>::= <correlation name>
<new values correlation name>::= <correlation name>
<old values table alias>::= <identifier>
<new values table alias>::= <identifier>
<triggered action>::=
                        [ FOR EACH { ROW | STATEMENT } ]
                        [  WHEN  <leftparen>  <search  condition>  <rightparen>  ]  <triggered  SQL
statement>
<triggered SQL statement>::=
                         <SQL procedure statement>
            |           BEGIN ATOMIC { <SQL procedure statement>  <semicolon> }... END
<user-defined type definition>::=CREATE TYPE <user-defined type body>
--p
--small
--i <user-defined type body> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<user-defined type body>::=
                         <user-defined type name> [ <subtype clause> ] [ AS <representation> ]
                        [ <instantiable clause> ] <finality> [ <reference type specification> ]
                        [ <ref cast option> ] [ <cast option> ] [ <method specification list> ]
<subtype clause>::=UNDER <supertype name>
<supertype name>::= <user-defined type>
<representation>::= <predefined type> | <member list>
<member list>::= <leftparen>  <member> [ { <comma>  <member> }... ] <rightparen>
<member>::= <attribute definition>
<attribute definition>::=
                         <attribute name>  <data type> [ <reference scope check> ] [ <attribute default> ]
                        [ <collate clause> ]
<attribute name>::= <identifier>
<attribute default>::= <default clause>
<instantiable clause>::=INSTANTIABLE | NOT INSTANTIABLE
<finality>::=FINAL | NOT FINAL
```

<reference type specification>::= <user-defined representation> | <derived representation> | <system-generated representation>
<user-defined representation>::=REF USING <predefined type>
<ref cast option>::=[<cast to ref>] [<cast to type>]
<cast to ref>::=
CAST <leftparen> SOURCE AS REF <rightparen> WITH <cast to ref identifier>
<cast to ref identifier>::= <identifier>
<cast to type>::=
CAST <leftparen> REF AS SOURCE <rightparen> WITH <cast to type identifier>
<cast to type identifier>::= <identifier>
<derived representation>::=REF FROM <list of attributes>
<list of attributes>::=
<leftparen> <attribute name> [{ <comma> <attribute name> }...] <rightparen>
<system-generated representation>::=REF IS SYSTEM GENERATED
<cast option>::=[<cast to distinct>] [<cast to source>]
<cast to distinct>::=
CAST <leftparen> SOURCE AS DISTINCT <rightparen> WITH <cast to distinct identifier>
<cast to distinct identifier>::= <identifier>
<cast to source>::=
CAST <leftparen> DISTINCT AS SOURCE <rightparen> WITH <cast to source identifier>
<cast to source identifier>::= <identifier>
<method specification list>::= <method specification> [{ <comma> <method specification> }...]
<method specification>::= <original method specification> | <overriding method specification>
<original method specification>::=
<partial method specification> [SELF AS RESULT] [SELF AS LOCATOR]
[<method characteristics>]
--p
--small
--i <partial method specification> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<partial method specification>::=
[INSTANCE | STATIC | CONSTRUCTOR] METHOD <method name>
<SQL parameter declaration list> <returns clause> [SPECIFIC <specific method name>]
<SQL parameter declaration list>::=
<leftparen> [<SQL parameter declaration>
[{ <comma> <SQL parameter declaration> }...]] <rightparen>
<SQL parameter declaration>::=
[<parameter mode>] [<SQL parameter name>] <parameter type> [RESULT]
<parameter mode>::=IN | OUT | INOUT
<SQL parameter name>::= <identifier>
<parameter type>::= <data type> [<locator indication>]
<returns clause>::=RETURNS <returns data type> [<result cast>]
<returns data type>::= <data type> [<locator indication>]
<result cast>::=CAST FROM <result cast from type>
<result cast from type>::= <data type> [<locator indication>]
--p
--small
--i <specific method name> added per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<specific method name>::=[<schema name> <period>] <qualified identifier>
<method characteristics>::= <method characteristic>...

```
--p
--small
--i <method characteristic> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <method characteristic>::=
                        <language clause>
              |         <parameter style clause>
              |         <deterministic characteristic>
              |         <SQL-data access indication>
              |         <null-call clause>
 <parameter style clause>::=PARAMETER STYLE <parameter style>
 <parameter style>::=SQL | GENERAL
 <deterministic characteristic>::=DETERMINISTIC | NOT DETERMINISTIC
 <SQL-data access indication>::=
                        NO SQL
              |         CONTAINS SQL
              |         READS SQL DATA
              |         MODIFIES SQL DATA
 <null-call clause>::=
                        RETURNS NULL ON NULL INPUT
              |         CALLED ON NULL INPUT
 <overriding method specification>::=OVERRIDING <partial method specification>
 <schema routine>::= <schema procedure> | <schema function>
 <schema procedure>::=CREATE <SQL-invoked procedure>
 <SQL-invoked procedure>::=
                        PROCEDURE <schema qualified routine name>
                         <SQL parameter declaration list>  <routine characteristics>  <routine body>
 <routine characteristics>::=[ <routine characteristic>... ]
--p
--small
--i <routine characteristic> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <routine characteristic>::=
                         <language clause>
              |          <parameter style clause>
              |         SPECIFIC <specific name>
              |          <deterministic characteristic>
              |          <SQL-data access indication>
              |          <null-call clause>
              |          <dynamic result sets characteristic>
 <dynamic result sets characteristic>::=
                        DYNAMIC RESULT SETS <maximum dynamic result sets>
 <maximum dynamic result sets>::= <unsigned integer>
 <routine body>::= <SQL routine body> | <external body reference>
 <SQL routine body>::= <SQL procedure statement>
--p
--small
--i <external body reference> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <external body reference>::=
                        EXTERNAL [ NAME <external routine name> ] [ <parameter style clause> ]
                        [ <transform group specification> ] [ <external security clause> ]
 <external routine name>::= <identifier> | <character string literal>
```

```
<external security clause>::=
                    EXTERNAL SECURITY DEFINER
          |         EXTERNAL SECURITY INVOKER
          |         EXTERNAL SECURITY IMPLEMENTATION DEFINED
<schema function>::=CREATE <SQL-invoked function>
<SQL-invoked function>::=
                    { <function specification> | <method specification designator> } <routine body>
<function specification>::=
                    FUNCTION <schema qualified routine name>  <SQL parameter declaration list>
                     <returns clause>  <routine characteristics> [ <dispatch clause> ]
<dispatch clause>::=STATIC DISPATCH
--p
--small
--i <method specification designator> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <method specification designator>::=
                    [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD <method name>
                     <SQL parameter declaration list> [ <returns clause> ] FOR <user-defined type
name>
 <grant statement>::= <grant privilege statement> | <grant role statement>
 <grant privilege statement>::=
                    GRANT <privileges> TO <grantee> [ { <comma>  <grantee> }... ]
                    [ WITH HIERARCHY OPTION ] [ WITH GRANT OPTION ] [ GRANTED BY
<grantor> ]
 <privileges>::= <object privileges> ON <object name>
 <object privileges>::=ALL PRIVILEGES | <action> [ { <comma>  <action> }... ]
 <action>::=
                    SELECT
          |         SELECT <leftparen>  <privilege column list>  <rightparen>
          |         SELECT <leftparen>  <privilege method list>  <rightparen>
          |         DELETE
          |         INSERT [ <leftparen>  <privilege column list>  <rightparen> ]
          |         UPDATE [ <leftparen>  <privilege column list>  <rightparen> ]
          |         REFERENCES [ <leftparen>  <privilege column list>  <rightparen> ]
          |         USAGE
          |         TRIGGER
          |         UNDER
          |         EXECUTE
 <privilege column list>::= <column name list>
 <privilege method list>::=
                     <specific routine designator> [ { <comma>  <specific routine designator> }... ]
 <object name>::=
                    [ TABLE ] <table name>
          |         DOMAIN <domain name>
          |         COLLATION <collation name>
          |         CHARACTER SET <character set name>
          |         MODULE <module name>
          |         TRANSLATION <translation name>
          |         TYPE <user-defined type name>
          |          <specific routine designator>
 <grantee>::=PUBLIC | <authorization identifier>
 <grantor>::=CURRENT_USER | CURRENT_ROLE
 <grant role statement>::=
                    GRANT <role granted> [ { <comma>  <role granted> }... ] TO <grantee> [ {
<comma>  <grantee> }... ]
                    [ WITH ADMIN OPTION ] [ GRANTED BY <grantor> ]
 <role granted>::= <role name>
```

<role definition>::=CREATE ROLE <role name> [WITH ADMIN <grantor>]
<SQL-invoked routine>::= <schema routine> | <module routine>
<user-defined cast definition>::=
CREATE CAST <leftparen> <source data type> AS <target data type> <rightparen>
WITH <cast function> [AS ASSIGNMENT]
<source data type>::= <data type>
<cast function>::= <specific routine designator>
--p
--small
--i <user-defined ordering specification> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<user-defined ordering definition>::=
CREATE ORDERING FOR <user-defined type name> <ordering form>
<ordering form>::= <equals ordering form> | <full ordering form>
<equals ordering form>::=EQUALS ONLY BY <ordering category>
<ordering category>::= <relative category> | <map category> | <state category>
<relative category>::=RELATIVE WITH <relative function specification>
<relative function specification>::= <specific routine designator>
<map category>::=MAP WITH <map function specification>
<map function specification>::= <specific routine designator>
<state category>::=STATE [<specific name>]
<full ordering form>::=ORDER FULL BY <ordering category>
--p
--small
--i <transform definition> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<transform definition>::=
CREATE { TRANSFORM | TRANSFORMS } FOR <user-defined type name>
<transform group>...
<transform group>::= <group name> <leftparen> <transform element list> <rightparen>
<transform element list>::= <transform element> [<comma> <transform element>]
<transform element>::= <to sql> | <from sql>
<to sql>::=TO SQL WITH <to sql function>
<to sql function>::= <specific routine designator>
<from sql>::=FROM SQL WITH <from sql function>
<from sql function>::= <specific routine designator>
--hr
--h2 SQL Schema Manipulation Statements
--/h2
<SQL schema manipulation statement>::=
<drop schema statement>
| <alter table statement>
| <drop table statement>
| <drop view statement>
| <alter routine statement>
| <drop routine statement>
| <drop user-defined cast statement>
| <revoke statement>
| <drop role statement>
| <alter domain statement>
| <drop domain statement>
| <drop character set statement>
| <drop collation statement>
| <drop translation statement>

| <drop assertion statement>
| <drop trigger statement>
| <alter type statement>
| <drop data type statement>
| <drop user-defined ordering statement>
| <drop transform statement>
| <drop module statement>

<drop schema statement>::=DROP SCHEMA <schema name> <drop behavior>
<drop behavior>::=CASCADE | RESTRICT
<alter table statement>::=ALTER TABLE <table name> <alter table action>
<alter table action>::=
<add column definition>
| <alter column definition>
| <drop column definition>
| <add table constraint definition>
| <drop table constraint definition>
<add column definition>::=ADD [COLUMN] <column definition>
<alter column definition>::=ALTER [COLUMN] <column name> <alter column action>
<alter column action>::=
<set column default clause>
| <drop column default clause>
| <add column scope clause>
| <drop column scope clause>
<set column default clause>::=SET <default clause>
<drop column default clause>::=DROP DEFAULT
<add column scope clause>::=ADD <scope clause>
<drop column scope clause>::=DROP SCOPE <drop behavior>
<drop column definition>::=DROP [COLUMN] <column name> <drop behavior>
<add table constraint definition>::=ADD <table constraint definition>
<drop table constraint definition>::=DROP CONSTRAINT <constraint name> <drop behavior>
<drop table statement>::=DROP TABLE <table name> <drop behavior>
<drop view statement>::=DROP VIEW <table name> <drop behavior>
<alter routine statement>::=
ALTER <specific routine designator> <alter routine characteristics> <alter routine behaviour>
<alter routine characteristics>::= <alter routine characteristic>...
<alter routine characteristic>::=
<language clause>
| <parameter style clause>
| <SQL-data access indication>
| <null-call clause>
| <dynamic result sets characteristic>
| NAME <external routine name>
<alter routine behaviour>::=RESTRICT
<drop routine statement>::=DROP <specific routine designator> <drop behavior>
<drop user-defined cast statement>::=
DROP CAST <leftparen> <source data type> AS <target data type> <rightparen>
<drop behavior>
<revoke statement>::= <revoke privilege statement> | <revoke role statement>
<revoke privilege statement>::=
REVOKE [<revoke option extension>] <privileges> FROM <grantee>
[{ <comma> <grantee> }...] [GRANTED BY <grantor>] <drop behavior>
<revoke option extension>::=GRANT OPTION FOR | HIERARCHY OPTION FOR
<revoke role statement>::=
REVOKE [ADMIN OPTION FOR] <role revoked> [{ <comma> <role revoked>}...]
FROM <grantee> [{ <comma> <grantee> }...] [GRANTED BY <grantor>] <drop behavior>

<role revoked>::= <role name>
<drop role statement>::=DROP ROLE <role name>
<alter domain statement>::=ALTER DOMAIN <domain name> <alter domain action>
<alter domain action>::=
<set domain default clause>
| <drop domain default clause>
| <add domain constraint definition>
| <drop domain constraint definition>
<set domain default clause>::=SET <default clause>
<drop domain default clause>::=DROP DEFAULT
<add domain constraint definition>::=ADD <domain constraint>
<drop domain constraint definition>::=DROP CONSTRAINT <constraint name>
<drop domain statement>::=DROP DOMAIN <domain name> <drop behavior>
<drop character set statement>::=DROP CHARACTER SET <character set name>
<drop collation statement>::=DROP COLLATION <collation name> <drop behavior>
<drop translation statement>::=DROP TRANSLATION <translation name>
<drop assertion statement>::=DROP ASSERTION <constraint name>
<drop trigger statement>::=DROP TRIGGER <trigger name>
<alter type statement>::=ALTER TYPE <user-defined type name> <alter type action>
<alter type action>::=
<add attribute definition>
| <drop attribute definition>
| <add original method specification>
| <add overriding method specification>
| <drop method specification>
<add attribute definition>::=ADD ATTRIBUTE <attribute definition>
<drop attribute definition>::=DROP ATTRIBUTE <attribute name> RESTRICT
<add original method specification>::=ADD <original method specification>
<add overriding method specification>::=ADD <overriding method specification>
--p
--small
--i <drop method specification> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<drop method specification>::=DROP <specific method specification designator> RESTRICT
--p
--small
--i <specific method specification designator> added per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<specific method specification designator>::=
SPECIFIC METHOD <specific method name>
| [INSTANCE | STATIC | CONSTRUCTOR] METHOD <method name> [<data type list>]
<drop data type statement>::=DROP TYPE <user-defined type name> <drop behavior>
--p
--small
--i <drop user-defined ordering statement> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<drop user-defined ordering statement>::=
DROP ORDERING FOR <user-defined type name> <drop behavior>
--p
--small
--i <drop transform statement> modified per ISO 9075:1999/Cor.1:2000(E)
--/i

```
--/small
--/p
 <drop transform statement>::=
                        DROP { TRANSFORM | TRANSFORMS } <transforms to be dropped>
                        FOR <user-defined type name>  <drop behavior>
 <transforms to be dropped>::=ALL | <transform group element>
 <transform group element>::= <group name>
--hr
--h2 SQL Data Manipulation Statements
--/h2
 <SQL data statement>::=
                         <open statement>
              |          <fetch statement>
              |          <close statement>
              |          <select statement: single row>
              |          <free locator statement>
              |          <hold locator statement>
              |          <SQL data change statement>
 <open statement>::=OPEN <cursor name>
 <fetch statement>::=
                        FETCH [ [ <fetch orientation> ] FROM ] <cursor name> INTO <fetch target list>
 <fetch orientation>::=
                        NEXT | PRIOR | FIRST | LAST
              |         { ABSOLUTE | RELATIVE } <simple value specification>
 <simple value specification>::=
                         <literal>
              |          <host parameter name>
              |          <SQL parameter reference>
              |          <SQL variable reference>
              |          <embedded variable name>
 <fetch target list>::= <target specification> [ { <comma>  <target specification> }... ]
 <close statement>::=CLOSE <cursor name>
 <select statement: single row>::=
                        SELECT [ <set quantifier> ] <select list> INTO <select target list>  <table
expression>
 <select target list>::= <target specification> [ { <comma>  <target specification> }... ]
 <free locator statement>::=
                        FREE LOCATOR <locator reference> [ { <comma>  <locator reference> }... ]
 <locator reference>::= <host parameter name> | <embedded variable name>
 <hold locator statement>::=
                        HOLD LOCATOR <locator reference> [ { <comma>  <locator reference> }... ]
 <SQL data change statement>::=
                         <delete statement: positioned>
              |          <delete statement: searched>
              |          <insert statement>
              |          <update statement: positioned>
              |          <update statement: searched>
 <delete statement: positioned>::=
                        DELETE FROM <target table> WHERE CURRENT OF <cursor name>
--p
--small
--i <target table> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <target table>::=
                         <table name>
              |          [ ONLY ] <leftparen>  <table name>  <rightparen>
 <delete statement: searched>::=
```

DELETE FROM <target table> [WHERE <search condition>]
<insert statement>::=
INSERT INTO <insertion target> <insert columns and source>
<insertion target>::= <table name>
<insert columns and source>::= <from subquery> | <from constructor> | <from default>
<from subquery>::=
[<leftparen> <insert column list> <rightparen>] [<override clause>] <query expression>
<insert column list>::= <column name list>
<from constructor>::=
[<leftparen> <insert column list> <rightparen>] [<override clause>]
<contextually typed table value constructor>
<override clause>::=OVERRIDING USER VALUE | OVERRIDING SYSTEM VALUE
<contextually typed table value constructor>::=
VALUES <contextually typed row value expression list>
<contextually typed row value expression list>::=
<contextually typed row value expression>
[{ <comma> <contextually typed row value expression> }...]
<contextually typed row value expression>::=
<row value special case>
| <contextually typed row value constructor>
<contextually typed row value constructor>::=
<contextually typed row value constructor element>
| [ROW] <leftparen> <contextually typed row value constructor element list> <rightparen>
<contextually typed row value constructor element>::=
<value expression> | <contextually typed value specification>
<contextually typed value specification>::=
<implicitly typed value specification> | <default specification>
<default specification>::=DEFAULT
<contextually typed row value constructor element list>::=
<contextually typed row value constructor element>
[{ <comma> <contextually typed row value constructor element> }...]
<from default>::=DEFAULT VALUES
<update statement: positioned>::=
UPDATE <target table> SET <set clause list> WHERE CURRENT OF <cursor name>
<set clause list>::= <set clause> [{ <comma> <set clause> }...]
<set clause>::=
<update target> <equals operator> <update source>
| <mutated set clause> <equals operator> <update source>
--p
--small
--i <update target> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<update target>::=
<object column>
| <object column> <left bracket or trigraph> <simple value specification> <right bracket or trigraph>
<object column>::= <column name>
<update source>::= <value expression> | <contextually typed value specification>
<mutated set clause>::= <mutated target> <period> <method name>
<mutated target>::= <object column> | <mutated set clause>
<update statement: searched>::=
UPDATE <target table> SET <set clause list> [WHERE <search condition>]
--hr
--h2 SQL Control Statements
--/h2

```
<SQL control statement>::=
                    <call statement>
          |         <return statement>
          |         <assignment statement>
          |         <compound statement>
          |         <case statement>
          |         <if statement>
          |         <iterate statement>
          |         <leave statement>
          |         <loop statement>
          |         <while statement>
          |         <repeat statement>
          |         <for statement>
<call statement>::=CALL <routine invocation>
<return statement>::=RETURN <return value>
<return value>::= <value expression> | NULL
--hr
--h2 Transaction Management
--/h2
<SQL transaction statement>::=
                    <start transaction statement>
          |         <set transaction statement>
          |         <set constraints mode statement>
          |         <savepoint statement>
          |         <release savepoint statement>
          |         <commit statement>
          |         <rollback statement>
<start transaction statement>::=
               START TRANSACTION <transaction mode> [ { <comma>  <transaction mode> }...]
<transaction mode>::= <isolation level> | <transaction access mode> | <diagnostics size>
<isolation level>::=ISOLATION LEVEL <level of isolation>
<level of isolation>::=
                    READ UNCOMMITTED
          |         READ COMMITTED
          |         REPEATABLE READ
          |         SERIALIZABLE
<transaction access mode>::=READ ONLY | READ WRITE
<diagnostics size>::=DIAGNOSTICS SIZE <number of conditions>
<number of conditions>::= <simple value specification>
<set transaction statement>::=SET [ LOCAL ] <transaction characteristics>
<transaction characteristics>::=
                    TRANSACTION <transaction mode> [ { <comma>  <transaction mode> }... ]
<set constraints mode statement>::=
                    SET CONSTRAINTS <constraint name list> { DEFERRED | IMMEDIATE }
<constraint name list>::=ALL | <constraint name> [ { <comma>  <constraint name> }... ]
<savepoint statement>::=SAVEPOINT <savepoint specifier>
--p
--small
--i <savepoint specifier> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
<savepoint specifier>::= <savepoint name>
<savepoint name>::= <identifier>
<simple target specification>::=
                    <host parameter specification>
          |         <SQL parameter reference>
          |         <column reference>
          |         <SQL variable reference>
```

```
            |               <embedded variable name>
<release savepoint statement>::=RELEASE SAVEPOINT <savepoint specifier>
<commit statement>::=COMMIT [ WORK ] [ AND [ NO ] CHAIN ]
<rollback statement>::=ROLLBACK [ WORK ] [ AND [ NO ] CHAIN ] [ <savepoint clause> ]
<savepoint clause>::=TO SAVEPOINT <savepoint specifier>
--hr
--h2 Connection Management
--/h2
<SQL connection statement>::= <connect statement> | <set connection statement> | <disconnect
statement>
<connect statement>::=CONNECT TO <connection target>
<connection target>::=
                        <SQL-server name> [ AS <connection name> ] [ USER <connection user name>
]
            |           DEFAULT
<SQL-server name>::= <simple value specification>
<connection name>::= <simple value specification>
<connection user name>::= <simple value specification>
<set connection statement>::=SET CONNECTION <connection object>
<connection object>::=DEFAULT | <connection name>
<disconnect statement>::=DISCONNECT <disconnect object>
<disconnect object>::= <connection object> | ALL | CURRENT
--hr
--h2 Session Attributes
--/h2
<SQL session statement>::=
                        <set session user identifier statement>
            |           <set role statement>
            |           <set local time zone statement>
            |           <set session characteristics statement>
            |           <set catalog statement>
            |           <set schema statement>
            |           <set names statement>
            |           <set path statement>
            |           <set transform group statement>
<set session user identifier statement>::=
                        SET SESSION AUTHORIZATION <value specification>
<set role statement>::=SET ROLE <role specification>
<role specification>::= <value specification> | NONE
<set local time zone statement>::=SET TIME ZONE <set time zone value>
<set time zone value>::= <interval value expression> | LOCAL
<set session characteristics statement>::=
                        SET SESSION CHARACTERISTICS AS <session characteristic list>
<session characteristic list>::= <session characteristic> [ { <comma>  <session characteristic> }... ]
<session characteristic>::= <transaction characteristics>
<SQL diagnostics statement>::= <get diagnostics statement> | <signal statement> | <resignal statement>
<get diagnostics statement>::=GET DIAGNOSTICS <SQL diagnostics information>
<SQL diagnostics information>::= <statement information> | <condition information>
<statement information>::=
                        <statement information item> [ { <comma>  <statement information item> }... ]
<statement information item>::=
                       <simple target specification><equals operator><statement information item name>
<statement information item name>::=
                        NUMBER
            |           MORE
            |           COMMAND_FUNCTION
            |           COMMAND_FUNCTION_CODE
            |           DYNAMIC_FUNCTION
            |           DYNAMIC_FUNCTION_CODE
```

| ROW_COUNT
| TRANSACTIONS_COMMITTED
| TRANSACTIONS_ROLLED_BACK
| TRANSACTION_ACTIVE

<condition information>::=
EXCEPTION <condition number>
<condition information item> [{ <comma> <condition information item> }...]

<condition number>::= <simple value specification>

<condition information item>::=
<simple target specification> <equals operator> <condition information item name>

<condition information item name>::=
CATALOG_NAME
| CLASS_ORIGIN
| COLUMN_NAME
| CONDITION_IDENTIFIER
| CONDITION_NUMBER
| CONNECTION_NAME
| CONSTRAINT_CATALOG
| CONSTRAINT_NAME
| CONSTRAINT_SCHEMA
| CURSOR_NAME
| MESSAGE_LENGTH
| MESSAGE_OCTET_LENGTH
| MESSAGE_TEXT
| PARAMETER_MODE
| PARAMETER_NAME
| PARAMETER_ORDINAL_POSITION
| RETURNED_SQLSTATE
| ROUTINE_CATALOG
| ROUTINE_NAME
| ROUTINE_SCHEMA
| SCHEMA_NAME
| SERVER_NAME
| SPECIFIC_NAME
| SUBCLASS_ORIGIN
| TABLE_NAME
| TRIGGER_CATALOG
| TRIGGER_NAME
| TRIGGER_SCHEMA

<dereference operation>::=
<reference value expression> <dereference operator> <attribute name>

<method reference>::=
<value expression primary> <dereference operator> <method name> <SQL argument list>

<method selection>::= <routine invocation>

--p
--small
--i <new invocation> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p

<new invocation>::= <method invocation> | <routine invocation>

<static method selection>::= <routine invocation>

<token>::= <nondelimiter token> | <delimiter token>

<nondelimiter token>::=
<regular identifier>
| <key word>
| <unsigned numeric literal>
| <national character string literal>

```
            |           <bit string literal>
            |           <hex string literal>
            |           <large object length token>
            |           <multiplier>
 <key word>::= <reserved word> | <non-reserved word>
--p
--small
--i <reserved word> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <reserved word>::=
                    ABSOLUTE | ACTION | ADD | AFTER | ALL | ALLOCATE | ALTER | AND |
ANY | ARE
            |       ARRAY | AS | ASC | ASSERTION | AT | AUTHORIZATION
            |       BEFORE | BEGIN | BETWEEN | BINARY | BIT | BLOB | BOOLEAN | BOTH
            |       BREADTH | BY
            |       CALL | CASCADE | CASCADED | CASE | CAST | CATALOG | CHAR |
CHARACTER
            |       CHECK | CLOB | CLOSE | COLLATE | COLLATION | COLUMN | COMMIT
            |       CONDITION | CONNECT | CONNECTION | CONSTRAINT | CONSTRAINTS
            |       CONSTRUCTOR | CONTINUE | CORRESPONDING | CREATE | CROSS |
CUBE
            |       CURRENT|CURRENT_DATE|
CURRENT_DEFAULT_TRANSFORM_GROUP
            |       CURRENT_TRANSFORM_GROUP_FOR_TYPE | CURRENT_PATH |
CURRENT_ROLE
            |       CURRENT_TIME | CURRENT_TIMESTAMP | CURRENT_USER | CURSOR |
CYCLE
            |       DATA | DATE | DAY | DEALLOCATE | DEC | DECIMAL | DECLARE |
DEFAULT
            |       DEFERRABLE | DEFERRED | DELETE | DEPTH | DEREF | DESC
            |       DESCRIBE | DESCRIPTOR | DETERMINISTIC
            |       DIAGNOSTICS | DISCONNECT | DISTINCT | DO | DOMAIN | DOUBLE
            |       DROP | DYNAMIC
            |       EACH | ELSE | ELSEIF | END | END-EXEC | EQUALS | ESCAPE | EXCEPT
            |       EXCEPTION | EXEC | EXECUTE | EXISTS | EXIT | EXTERNAL
            |       FALSE | FETCH | FIRST | FLOAT | FOR | FOREIGN | FOUND | FROM | FREE
            |       FULL | FUNCTION
            |       GENERAL | GET | GLOBAL | GO | GOTO | GRANT | GROUP | GROUPING
            |       HANDLE | HAVING | HOLD | HOUR
            |       IDENTITY | IF | IMMEDIATE | IN | INDICATOR
            |       INITIALLY | INNER | INOUT | INPUT | INSERT | INT | INTEGER
            |       INTERSECT | INTERVAL | INTO | IS | ISOLATION
            |       JOIN
            |       KEY
            |       LANGUAGE | LARGE | LAST | LATERAL | LEADING | LEAVE | LEFT
            |       LEVEL | LIKE | LOCAL | LOCALTIME | LOCALTIMESTAMP | LOCATOR |
LOOP
            |       MAP | MATCH | METHOD | MINUTE | MODIFIES | MODULE | MONTH
            |       NAMES | NATIONAL | NATURAL | NCHAR | NCLOB | NESTING | NEW |
NEXT
            |       NO | NONE | NOT | NULL | NUMERIC
            |       OBJECT | OF | OLD | ON | ONLY | OPEN | OPTION
            |       OR | ORDER | ORDINALITY | OUT | OUTER | OUTPUT | OVERLAPS
            |       PAD | PARAMETER | PARTIAL | PATH | PRECISION
            |       PREPARE | PRESERVE | PRIMARY | PRIOR | PRIVILEGES | PROCEDURE |
PUBLIC
```

```
        |       READ | READS | REAL | RECURSIVE | REDO | REF | REFERENCES |
REFERENCING
        |       RELATIVE | RELEASE | REPEAT | RESIGNAL | RESTRICT | RESULT |
RETURN
        |       RETURNS | REVOKE | RIGHT | ROLE | ROLLBACK | ROLLUP | ROUTINE
        |       ROW | ROWS
        |       SAVEPOINT | SCHEMA | SCROLL | SEARCH | SECOND | SECTION |
SELECT
        |       SESSION | SESSION_USER | SET | SETS | SIGNAL | SIMILAR | SIZE
        |       SMALLINT | SOME | SPACE | SPECIFIC | SPECIFICTYPE | SQL |
SQLEXCEPTION
        |       SQLSTATE | SQLWARNING | START | STATE | STATIC | SYSTEM_USER
        |       TABLE | TEMPORARY | THEN | TIME | TIMESTAMP
        |       TIMEZONE_HOUR | TIMEZONE_MINUTE | TO | TRAILING |
TRANSACTION
        |       TRANSLATION | TREAT | TRIGGER | TRUE
        |       UNDER | UNDO | UNION | UNIQUE | UNKNOWN | UNNEST | UNTIL |
UPDATE
        |       USAGE | USER | USING
        |       VALUE | VALUES | VARCHAR | VARYING | VIEW
        |       WHEN | WHENEVER | WHERE | WHILE | WITH | WITHOUT | WORK |
WRITE
        |       YEAR
        |       ZONE
--p
--small
--i <non-reserved word> modified per ISO 9075:1999/Cor.1:2000(E)
--/i
--/small
--/p
 <non-reserved word>::=
        ABS | ADA | ADMIN | ASENSITIVE | ASSIGNMENT | ASYMMETRIC |
ATOMIC
        |       ATTRIBUTE | AVG
        |       BIT_LENGTH
        |       C | CALLED | CARDINALITY | CATALOG_NAME | CHAIN |
CHAR_LENGTH
        |       CHARACTERISTICS | CHARACTER_LENGTH |
CHARACTER_SET_CATALOG
        |       CHARACTER_SET_NAME | CHARACTER_SET_SCHEMA | CHECKED |
CLASS_ORIGIN
        |       COALESCE | COBOL | COLLATION_CATALOG | COLLATION_NAME |
COLLATION_SCHEMA
        |       COLUMN_NAME | COMMAND_FUNCTION |
COMMAND_FUNCTION_CODE | COMMITTED
        |       CONDITION_IDENTIFIER | CONDITION_NUMBER |
CONNECTION_NAME
        |       CONSTRAINT_CATALOG | CONSTRAINT_NAME |
CONSTRAINT_SCHEMA | CONTAINS
        |       CONVERT | COUNT | CURSOR_NAME
        |       DATETIME_INTERVAL_CODE | DATETIME_INTERVAL_PRECISION |
DEFINED
        |       DEFINER | DEGREE | DERIVED | DISPATCH
        |       EVERY | EXTRACT
        |       FINAL | FORTRAN
        |       G | GENERATED | GRANTED
        |       HIERARCHY
        |       IMPLEMENTATION | INSENSITIVE | INSTANCE | INSTANTIABLE |
INVOKER
```

```
        |       K | KEY_MEMBER | KEY_TYPE
        |       LENGTH | LOWER
        |       M | MAX | MIN | MESSAGE_LENGTH | MESSAGE_OCTET_LENGTH |
MESSAGE_TEXT
        |       MOD | MORE | MUMPS
        |       NAME | NULLABLE | NUMBER | NULLIF
        |       OCTET_LENGTH | ORDERING | OPTIONS | OVERLAY | OVERRIDING
        |       PASCAL | PARAMETER_MODE | PARAMETER_NAME
        |       PARAMETER_ORDINAL_POSITION | PARAMETER_SPECIFIC_CATALOG
        |       PARAMETER_SPECIFIC_NAME | PARAMETER_SPECIFIC_SCHEMA | PLI
| POSITION
        |       REPEATABLE | RETURNED_CARDINALITY | RETURNED_LENGTH
        |       RETURNED_OCTET_LENGTH | RETURNED_SQLSTATE |
ROUTINE_CATALOG
        |       ROUTINE_NAME | ROUTINE_SCHEMA | ROW_COUNT
        |       SCALE | SCHEMA_NAME | SCOPE | SECURITY | SELF | SENSITIVE |
SERIALIZABLE
        |       SERVER_NAME | SIMPLE | SOURCE | SPECIFIC_NAME | STATEMENT |
STRUCTURE
        |       STYLE | SUBCLASS_ORIGIN | SUBSTRING | SUM | SYMMETRIC |
SYSTEM
        |       TABLE_NAME | TOP_LEVEL_COUNT | TRANSACTIONS_COMMITTED
        |       TRANSACTIONS_ROLLED_BACK | TRANSACTION_ACTIVE |
TRANSFORM
        |       TRANSFORMS | TRANSLATE | TRIGGER_CATALOG |
TRIGGER_SCHEMA
        |       TRIGGER_NAME | TRIM | TYPE
        |       UNCOMMITTED | UNNAMED | UPPER
<delimiter token>::=
                <character string literal>
        |       <date string>
        |       <time string>
        |       <timestamp string>
        |       <interval string>
        |       <delimited identifier>
        |       <SQL special character>
        |       <not equals operator>
        |       <greater than or equals operator>
        |       <less than or equals operator>
        |       <concatenation operator>
        |       <right arrow>
        |       <left bracket trigraph>
        |       <right bracket trigraph>
        |       <double colon>
        |       <double period>
<CLI routine>::= <CLI routine name>  <CLI parameter list> [ <CLI returns clause> ]
<CLI routine name>::= <CLI name prefix>  <CLI generic name>
<CLI name prefix>::= <CLI by-reference prefix> | <CLI by-value prefix>
<CLI by-reference prefix>::=SQLR
<CLI by-value prefix>::=SQL
<CLI generic name>::=
                AllocConnect | AllocEnv | AllocHandle | AllocStmt
        |       BindCol | BindParameter
        |       Cancel | CloseCursor | ColAttribute | ColumnPrivileges | Columns | Connect |
CopyDesc
        |       DataSources | DescribeCol | Disconnect
        |       EndTran | Error | ExecDirect | Execute
        |       Fetch | FetchScroll | ForeignKeys | FreeConnect | FreeEnv | FreeHandle | FreeStmt
```

```
        |       GetConnectAttr | GetCursorName | GetData | GetDescField | GetDescRec |
GetDiagField
        |       GetDiagRec | GetEnvAttr | GetFeatureInfo | GetFunctions | GetInfo | GetLength
        |       GetParamData | GetPosition | GetSessionInfo | GetStmtAttr | GetSubString |
GetTypeInfo
        |       MoreResults
        |       NextResult | NumResultCols
        |       ParamData | Prepare | PrimaryKeys | PutData
        |       RowCount
        |       SetConnectAttr | SetCursorName | SetDescField | SetDescRec | SetEnvAttr |
SetStmtAttr
        |       SpecialColumns | StartTran
        |       TablePrivileges | Tables
        |       <implementation-defined CLI generic name>
<implementation-defined CLI generic name>::=!! (See the Syntax Rules)
<CLI parameter list>::=
                <leftparen> <CLI parameter declaration> [ { <comma> <CLI parameter
declaration> }... ] <rightparen>
<CLI parameter declaration>::=
                <CLI parameter name> <CLI parameter mode> <CLI parameter data type>
<CLI parameter name>::=!! (See the individual CLI routine definitions)
<CLI parameter mode>::=IN | OUT | DEFIN | DEFOUT | DEF
<CLI parameter data type>::=
                INTEGER
        |       SMALLINT
        |       ANY
        |       CHARACTER <leftparen> <length> <rightparen>
<CLI returns clause>::=RETURNS SMALLINT
<assignment statement>::=
                SET <assignment target> <equals operator> <assignment source>
<assignment target>::= <target specification> | <modified field reference> | <mutator reference>
<SQL variable reference>::= <basic identifier chain>
<modified field reference>::= <modified field target> <period> <field name>
<modified field target>::=
        <target specification> | <leftparen> <target specification> <rightparen> | <modified field
reference>
<mutator reference>::= <mutated target specification> <period> <method name>
<mutated target specification>::=
                <target specification> | <leftparen> <target specification> <rightparen> |
<mutator reference>
<assignment source>::= <value expression> | <contextually typed source>
<contextually typed source>::=
                <implicitly typed value specification> | <contextually typed row value
expression>
<compound statement>::=
                [ <beginning label> <colon> ] BEGIN [ [ NOT ] ATOMIC ]
                [ <local declaration list> ] [ <local cursor declaration list> ] [ <local handler
declaration list> ]
                [ <SQL statement list> ] END [ <ending label> ]
<beginning label>::= <statement label>
<statement label>::= <identifier>
<local declaration list>::= <terminated local declaration>...
<terminated local declaration>::= <local declaration> <semicolon>
<local declaration>::= <SQL variable declaration> | <condition declaration>
<SQL variable declaration>::=
                DECLARE <SQL variable name list> <data type> [ <default clause> ]
<SQL variable name list>::=
                <SQL variable name> [ { <comma> <SQL variable name> }... ]
<SQL variable name>::= <identifier>
```

<condition declaration>::=
DECLARE <condition name> CONDITION [FOR <sqlstate value>]
<condition name>::= <identifier>
<sqlstate value>::=SQLSTATE [VALUE] <character string literal>
<local cursor declaration list>::= <terminated local cursor declaration>...
<terminated local cursor declaration>::= <declare cursor> <semicolon>
<local handler declaration list>::= <terminated local handler declaration>...
<terminated local handler declaration>::= <handler declaration> <semicolon>
<handler declaration>::=
DECLARE <handler type> HANDLER FOR <condition value list> <handler action>
<handler type>::=CONTINUE | EXIT | UNDO
<condition value list>::= <condition value> [{ <comma> <condition value> }...]
<condition value>::=
<sqlstate value> | <condition name> | SQLEXCEPTION | SQLWARNING | NOT FOUND
<handler action>::= <SQL procedure statement>
<SQL statement list>::= <terminated SQL statement>...
<terminated SQL statement>::= <SQL procedure statement> <semicolon>
<ending label>::= <statement label>
<case statement>::= <simple case statement> | <searched case statement>
<simple case statement>::=
CASE <simple case operand 1> <simple case statement when clause>... [<case statement else clause>] END CASE
<simple case operand 1>::= <value expression>
<simple case statement when clause>::=
WHEN <simple case operand 2> THEN <SQL statement list>
<simple case operand 2>::= <value expression>
<case statement else clause>::=ELSE <SQL statement list>
<searched case statement>::=
CASE <searched case statement when clause>... [<case statement else clause>] END CASE
<searched case statement when clause>::=
WHEN <search condition> THEN <SQL statement list>
<if statement>::=
IF <search condition> <if statement then clause>
[<if statement elseif clause>...] [<if statement else clause>]
END IF
<if statement then clause>::=THEN <SQL statement list>
<if statement elseif clause>::=ELSEIF <search condition> THEN <SQL statement list>
<if statement else clause>::=ELSE <SQL statement list>
<iterate statement>::=ITERATE <statement label>
<leave statement>::=LEAVE <statement label>
<loop statement>::=
[<beginning label> <colon>] LOOP <SQL statement list> END LOOP [<ending label>]
<while statement>::=
[<beginning label> <colon>] WHILE <search condition> DO <SQL statement list> END WHILE [<ending label>]
<repeat statement>::=
[<beginning label> <colon>] REPEAT <SQL statement list> UNTIL <search condition> END REPEAT [<ending label>]
<for statement>::=
[<beginning label> <colon>] FOR <for loop variable name> AS
[<cursor name> [<cursor sensitivity>] CURSOR FOR] <cursor specification>
DO <SQL statement list> END FOR [<ending label>]
<for loop variable name>::= <identifier>
<signal statement>::=SIGNAL <signal value> [<set signal information>]
<signal value>::= <condition name> | <sqlstate value>
<set signal information>::=SET <signal information item list>
<signal information item list>::=
<signal information item> [{ <comma> <signal information item> }...]

<signal information item>::=
<condition information item name> <equals operator> <simple value specification>
<resignal statement>::=RESIGNAL [<signal value>] [<set signal information>]
<SQL-server module definition>::=
CREATE MODULE <SQL-server module name> [<SQL-server module character set specification>]
[<SQL-server module schema clause>][<SQL-server module path specification>]
[<temporary table declaration>]
<SQL-server module contents>...
END MODULE
<SQL-server module name>::= <schema qualified name>
<SQL-server module character set specification>::=NAMES ARE <character set specification>
<SQL-server module schema clause>::=SCHEMA <default schema name>
<default schema name>::= <schema name>
<SQL-server module path specification>::= <path specification>
<SQL-server module contents>::= <SQL-invoked routine> <semicolon>
<module routine>::= <module procedure> | <module function>
<module procedure>::=[DECLARE] <SQL-invoked procedure>
<module function>::=[DECLARE] <SQL-invoked function>
<drop module statement>::=DROP MODULE <SQL-server module name> <drop behavior>
<triggered SQL statement>::= <SQL procedure statement>
--hr
--h2 Dynamic SQL
--/h2
--p
Much, if not all, of the following material comes from ISO/IEC 9075-5:1999, SQL/Bindings.
--/p
<SQL dynamic statement>::=
<system descriptor statement>
| <prepare statement>
| <deallocate prepared statement>
| <describe statement>
| <execute statement>
| <execute immediate statement>
| <SQL dynamic data statement>
<system descriptor statement>::=
<allocate descriptor statement>
| <deallocate descriptor statement>
| <set descriptor statement>
| <get descriptor statement>
<allocate descriptor statement>::=
ALLOCATE [SQL] DESCRIPTOR <descriptor name> [WITH MAX <occurrences>]
<descriptor name>::=[<scope option>] <simple value specification>
<scope option>::=GLOBAL | LOCAL
<embedded variable name>::= <colon> <host identifier>
<host identifier>::=
<Ada host identifier>
| <C host identifier>
| <COBOL host identifier>
| <Fortran host identifier>
| <MUMPS host identifier>
| <Pascal host identifier>
| <PL/I host identifier>
<Ada host identifier>::=!! (See the Syntax Rules.)
<C host identifier>::=!! (See the Syntax Rules.)
<COBOL host identifier>::=!! (See the Syntax Rules.)
<Fortran host identifier>::=!! (See the Syntax Rules.)
<MUMPS host identifier>::=!! (See the Syntax Rules.)

<Pascal host identifier>::=!! (See the Syntax Rules.)
<PL/I host identifier>::=!! (See the Syntax Rules.)
<occurrences>::= <simple value specification>
<deallocate descriptor statement>::=DEALLOCATE [SQL] DESCRIPTOR <descriptor name>
<set descriptor statement>::=SET [SQL] DESCRIPTOR <descriptor name> <set descriptor information>
<set descriptor information>::=
<set header information> [{ <comma> <set header information> }...]
| VALUE <item number><set item information>[{<comma><set item information>...]
<set header information>::=
<header item name> <equals operator> <simple value specification 1>
<header item name>::=
COUNT | KEY_TYPE | DYNAMIC_FUNCTION | DYNAMIC_FUNCTION_CODE | TOP_LEVEL_COUNT
<simple value specification 1>::= <simple value specification>
<item number>::= <simple value specification>
<set item information>::=
<descriptor item name> <equals operator> <simple value specification 2>
<descriptor item name>::=
CARDINALITY
| CHARACTER_SET_CATALOG
| CHARACTER_SET_NAME
| CHARACTER_SET_SCHEMA
| COLLATION_CATALOG
| COLLATION_NAME
| COLLATION_SCHEMA
| DATA
| DATETIME_INTERVAL_CODE
| DATETIME_INTERVAL_PRECISION
| DEGREE
| INDICATOR
| KEY_MEMBER
| LENGTH
| LEVEL
| NAME
| NULLABLE
| OCTET_LENGTH
| PARAMETER_MODE
| PARAMETER_ORDINAL_POSITION
| PARAMETER_SPECIFIC_CATALOG
| PARAMETER_SPECIFIC_NAME
| PARAMETER_SPECIFIC_SCHEMA
| PRECISION
| RETURNED_CARDINALITY
| RETURNED_LENGTH
| RETURNED_OCTET_LENGTH
| SCALE
| SCOPE_CATALOG
| SCOPE_NAME
| SCOPE_SCHEMA
| TYPE
| UNNAMED
| USER_DEFINED_TYPE_CATALOG
| USER_DEFINED_TYPE_NAME
| USER_DEFINED_TYPE_SCHEMA
<simple value specification 2>::= <simple value specification>
<item number>::= <simple value specification>
<get descriptor statement>::=
GET [SQL] DESCRIPTOR <descriptor name> <get descriptor information>

```
<get descriptor information>::=
                    <get header information> [ { <comma> <get header information> }... ]
          |         VALUE <item number> <get item information> [ { <comma> <get item
information> }... ]
<get header information>::=
                    <simple target specification 1> <equals operator> <header item name>
<simple target specification 1>::= <simple target specification>
<get item information>::=
                    <simple target specification 2> <equals operator> <descriptor item name>
<simple target specification 2>::= <simple target specification>
<prepare statement>::=
                    PREPARE <SQL statement name> FROM <SQL statement variable>
<SQL statement name>::=
                    <statement name>
          |         <extended statement name>
<statement name>::= <identifier>
<extended statement name>::=[ <scope option> ] <simple value specification>
<SQL statement variable>::= <simple value specification>
<deallocate prepared statement>::=DEALLOCATE PREPARE <SQL statement name>
<describe statement>::= <describe input statement> | <describe output statement>
<describe input statement>::=
          DESCRIBE INPUT <SQL statement name> <using descriptor> [ <nesting option> ]
<using descriptor>::=USING [ SQL ] DESCRIPTOR <descriptor name>
<nesting option>::=WITH NESTING | WITHOUT NESTING
<describe output statement>::=
          DESCRIBE [ OUTPUT ] <described object> <using descriptor> [ <nesting option> ]
<described object>::=
                    <SQL statement name> | CURSOR <extended cursor name> STRUCTURE
<extended cursor name>::=[ <scope option> ] <simple value specification>
<execute statement>::=
          EXECUTE <SQL statement name> [ <result using clause> ] [ <parameter using clause> ]
<result using clause>::= <output using clause>
<output using clause>::= <into arguments> | <into descriptor>
<into arguments>::=INTO <into argument> [ { <comma> <into argument> }... ]
<into argument>::= <target specification>
<dynamic parameter specification>::= <question mark>
<embedded variable specification>::= <embedded variable name> [ <indicator variable> ]
<indicator variable>::=[ INDICATOR ] <embedded variable name>
<into descriptor>::=INTO [ SQL ] DESCRIPTOR <descriptor name>
<parameter using clause>::= <input using clause>
<input using clause>::= <using arguments> | <using input descriptor>
<using arguments>::=USING <using argument> [ { <comma> <using argument> }... ]
<using argument>::= <general value specification>
<using input descriptor>::= <using descriptor>
<execute immediate statement>::=EXECUTE IMMEDIATE <SQL statement variable>
<SQL dynamic data statement>::=
                    <allocate cursor statement>
          |         <dynamic open statement>
          |         <dynamic fetch statement>
          |         <dynamic close statement>
          |         <dynamic delete statement: positioned>
          |         <dynamic update statement: positioned>
<allocate cursor statement>::=ALLOCATE <extended cursor name> <cursor intent>
<cursor intent>::= <statement cursor> | <result set cursor>
<statement cursor>::=
                    [<cursor sensitivity>] [ SCROLL ] CURSOR [ WITH HOLD ] [ WITH RETURN
]
                    FOR <extended statement name>
<result set cursor>::=
```

FOR PROCEDURE <specific routine designator>
<dynamic open statement>::=
OPEN <dynamic cursor name> [<input using clause>]
<dynamic cursor name>::= <cursor name> | <extended cursor name>
<dynamic fetch statement>::=
FETCH [[<fetch orientation>] FROM] <dynamic cursor name> <output using clause>
<dynamic close statement>::=CLOSE <dynamic cursor name>
<dynamic delete statement: positioned>::=
DELETE FROM <target table> WHERE CURRENT OF <dynamic cursor name>
<dynamic update statement: positioned>::=
UPDATE <target table> SET <set clause list> WHERE CURRENT OF <dynamic cursor name>
--p
--small
--i
Note that <double period> must be a pair of period characters with no intervening space, not a pair of period symbols separated by arbitrary white space. Normally, the lexical analyzer would return <double period> as a symbol.
--/i
--/small
--/p
<double period>::= <period> <period>
<direct SQL statement>::= <directly executable statement> <semicolon>
<directly executable statement>::=
<direct SQL data statement>
| <SQL schema statement>
| <SQL transaction statement>
| <SQL connection statement>
| <SQL session statement>
| <direct implementation-defined statement>
<direct SQL data statement>::=
<delete statement: searched>
| <direct select statement: multiple rows>
| <insert statement>
| <update statement: searched>
| <temporary table declaration>
<direct select statement: multiple rows>::= <query expression> [<order by clause>]
<set catalog statement>::=SET <catalog name characteristic>
<catalog name characteristic>::=CATALOG <value specification>
<set schema statement>::=SET <schema name characteristic>
<schema name characteristic>::=SCHEMA <value specification>
<set names statement>::=SET <character set name characteristic>
<character set name characteristic>::=NAMES <value specification>
<set path statement>::=SET <SQL-path characteristic>
<SQL-path characteristic>::=PATH <value specification>
<set transform group statement>::=SET <transform group characteristic>
<transform group characteristic>::=
DEFAULT TRANSFORM GROUP <value specification>
| TRANSFORM GROUP FOR TYPE <user-defined type> <value specification>
<direct implementation-defined statement>::=!! (See the Syntax Rules)
<embedded SQL declare section>::=
<embedded SQL begin declare>
[<embedded character set declaration>]
[<host variable definition>...]
<embedded SQL end declare>
| <embedded SQL MUMPS declare>
<embedded SQL begin declare>::=
<SQL prefix> BEGIN DECLARE SECTION [<SQL terminator>]

```
<SQL prefix>::=EXEC SQL | <ampersand> SQL <leftparen>
<SQL terminator>::=END-EXEC | <semicolon> | <rightparen>
<embedded character set declaration>::=
                    SQL NAMES ARE <character set specification>
<host variable definition>::=
                    <Ada variable definition>
          |         <C variable definition>
          |         <COBOL variable definition>
          |         <Fortran variable definition>
          |         <MUMPS variable definition>
          |         <Pascal variable definition>
          |         <PL/I variable definition>
<Ada variable definition>::=
                    <Ada host identifier> [ { <comma> <Ada host identifier> }... ] <colon>
                    <Ada type specification> [ <Ada initial value> ]
<Ada type specification>::=
                    <Ada qualified type specification>
          |         <Ada unqualified type specification>
          |         <Ada derived type specification>
<Ada qualified type specification>::=
                    Interfaces.SQL <period> CHAR [ CHARACTER SET [ IS ] <character set
specification> ]
                    <leftparen> 1 <double period> <length> <rightparen>
          |         Interfaces.SQL <period> BIT <leftparen> 1 <double period> <length>
<rightparen>
          |         Interfaces.SQL <period> SMALLINT
          |         Interfaces.SQL <period> INT
          |         Interfaces.SQL <period> REAL
          |         Interfaces.SQL <period> DOUBLE_PRECISION
          |         Interfaces.SQL <period> BOOLEAN
          |         Interfaces.SQL <period> SQLSTATE_TYPE
          |         Interfaces.SQL <period> INDICATOR_TYPE
<Ada unqualified type specification>::=
                    CHAR <leftparen> 1 <double period> <length> <rightparen>
          |         BIT <leftparen> 1 <double period> <length> <rightparen>
          |         SMALLINT
          |         INT
          |         REAL
          |         DOUBLE_PRECISION
          |         BOOLEAN
          |         SQLSTATE_TYPE
          |         INDICATOR_TYPE
<Ada derived type specification>::=
                    <Ada CLOB variable>
          |         <Ada BLOB variable>
          |         <Ada user-defined type variable>
          |         <Ada CLOB locator variable>
          |         <Ada BLOB locator variable>
          |         <Ada user-defined type locator variable>
          |         <Ada array locator variable>
          |         <Ada REF variable>
<Ada CLOB variable>::=
                    SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
                    [ CHARACTER SET [ IS ] <character set specification> ]
<Ada BLOB variable>::=
                    SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<Ada user-defined type variable>::=
                    SQL TYPE IS <user-defined type> AS <predefined type>
<Ada CLOB locator variable>::=
```

```
                    SQL TYPE IS CLOB AS LOCATOR
<Ada BLOB locator variable>::=
                    SQL TYPE IS BLOB AS LOCATOR
<Ada user-defined type locator variable>::=
                    SQL TYPE IS <user-defined type name> AS LOCATOR
<Ada array locator variable>::=
                    SQL TYPE IS <collection type> AS LOCATOR
<Ada REF variable>::=
                    SQL TYPE IS <reference type>
<Ada initial value>::=
                     <Ada assignment operator>  <character representation>...
<Ada assignment operator>::= <colon>  <equals operator>
<C variable definition>::=
                    [<C storage class>][<C class modifier>]<C variable specification> <semicolon>
<C storage class>::=auto | extern | static
<C class modifier>::=const | volatile
<C variable specification>::= <C numeric variable> | <C character variable> | <C derived variable>
<C numeric variable>::=
                    { long | short | float | double } <C host identifier> [ <C initial value> ]
                    [ { <comma>  <C host identifier> [ <C initial value> ] }... ]
<C initial value>::=
                     <equals operator>  <character representation>...
<C character variable>::=
                     <C character type> [ CHARACTER SET [ IS ] <character set specification> ]
                     <C host identifier>  <C array specification> [ <C initial value> ]
                    [ { <comma>  <C host identifier>  <C array specification> [ <C initial value> ]
}... ]
<C character type>::=char | unsigned char | unsigned short
<C array specification>::= <left bracket>  <length>  <right bracket>
<C derived variable>::=
                     <C VARCHAR variable>
          |          <C NCHAR variable>
          |          <C NCHAR VARYING variable>
          |          <C CLOB variable>
          |          <C NCLOB variable>
          |          <C BLOB variable>
          |          <C bit variable>
          |          <C user-defined type variable>
          |          <C CLOB locator variable>
          |          <C BLOB locator variable>
          |          <C array locator variable>
          |          <C user-defined type locator variable>
          |          <C REF variable>
<C VARCHAR variable>::=
                    VARCHAR [ CHARACTER SET [ IS ] <character set specification> ]
                     <C host identifier>  <C array specification> [ <C initial value> ]
                    [ { <comma>  <C host identifier>  <C array specification> [ <C initial value> ]
}... ]
<C NCHAR variable>::=
                    NCHAR [ CHARACTER SET [ IS ] <character set specification> ]
                     <C host identifier>  <C array specification> [ <C initial value> ]
                    [ { <comma>  <C host identifier>  <C array specification> [ <C initial value> ]
}...]
<C NCHAR VARYING variable>::=
                    NCHAR VARYING [ CHARACTER SET [ IS ] <character set specification> ]
                     <C host identifier>  <C array specification> [ <C initial value> ]
                    [ { <comma>  <C host identifier>  <C array specification> [ <C initial value> ]
}...]
<C CLOB variable>::=
```

```
                    SQL TYPE IS CLOB <leftparen>  <large object length>  <rightparen>
                    [ CHARACTER SET [ IS ] <character set specification> ]
                     <C host identifier> [ <C initial value> ] [ { <comma>  <C host identifier> [ <C
initial value> ] }... ]
 <C NCLOB variable>::=
                    SQL TYPE IS NCLOB <leftparen>  <large object length>  <rightparen>
                    [ CHARACTER SET [ IS ] <character set specification> ]
                     <C host identifier> [ <C initial value> ] [ { <comma>  <C host identifier> [ <C
initial value> ] }... ]
 <C BLOB variable>::=
                    SQL TYPE IS BLOB <leftparen>  <large object length>  <rightparen>
                     <C host identifier> [ <C initial value> ] [ { <comma>  <C host identifier> [ <C
initial value> ] }... ]
 <C bit variable>::=
                    BIT <C host identifier>  <C array specification> [ <C initial value> ]
                    [ { <comma>  <C host identifier>  <C array specification> [ <C initial value> ]
}... ]
 <C user-defined type variable>::=
                    SQL TYPE IS <user-defined type> AS <predefined type>
                     <C host identifier> [ <C initial value> ] [ { <comma>  <C host identifier> [ <C
initial value> ] }... ]
 <C CLOB locator variable>::=
                    SQL TYPE IS CLOB AS LOCATOR
                     <C host identifier> [ <C initial value> ] [ { <comma>  <C host identifier> [ <C
initial value> ] }... ]
 <C BLOB locator variable>::=
                    SQL TYPE IS BLOB AS LOCATOR
                     <C host identifier> [ <C initial value> ] [ { <comma>  <C host identifier> [ <C
initial value> ] }... ]
 <C array locator variable>::=
                    SQL TYPE IS <collection type> AS LOCATOR
                     <C host identifier> [ <C initial value> ] [ { <comma>  <C host identifier> [ <C
initial value> ] }... ]
 <C user-defined type locator variable>::=
                    SQL TYPE IS <user-defined type> AS LOCATOR
                     <C host identifier> [ <C initial value> ] [ { <comma>  <C host identifier> [ <C
initial value> ] }... ]
 <C REF variable>::=
                    SQL TYPE IS <reference type>

 <COBOL variable definition>::=
                    {01|77} <COBOL host identifier>  <COBOL type specification>
                    [ <character representation>... ] <period>
 <COBOL type specification>::=
                     <COBOL character type>
             |       <COBOL national character type>
             |       <COBOL bit type>
             |       <COBOL numeric type>
             |       <COBOL integer type>
             |       <COBOL derived type specification>
 <COBOL character type>::=
                    [ CHARACTER SET [ IS ] <character set specification> ]
                    { PIC | PICTURE } [ IS ] { X [ <leftparen>  <length>  <rightparen> ] }...
 <COBOL national character type>::=
                    [ CHARACTER SET [ IS ] <character set specification> ]
                    { PIC | PICTURE } [ IS ] { N [ <leftparen>  <length>  <rightparen> ] }...
 <COBOL bit type>::=
                    { PIC | PICTURE } [ IS ] { X [ <leftparen>  <length>  <rightparen> ] }...
                    USAGE [ IS ] BIT
```

<COBOL numeric type>::=
{ PIC | PICTURE } [IS] S <COBOL nines specification>
[USAGE [IS]] DISPLAY SIGN LEADING SEPARATE
<COBOL nines specification>::= <COBOL nines> [V [<COBOL nines>]] | V <COBOL nines>
<COBOL nines>::={ 9 [<leftparen> <length> <rightparen>] }...
<COBOL integer type>::= <COBOL binary integer>
<COBOL binary integer>::=
{ PIC | PICTURE } [IS] S <COBOL nines> [USAGE [IS]] BINARY
<COBOL derived type specification>::=
<COBOL CLOB variable>
| <COBOL NCLOB variable>
| <COBOL BLOB variable>
| <COBOL user-defined type variable>
| <COBOL CLOB locator variable>
| <COBOL BLOB locator variable>
| <COBOL array locator variable>
| <COBOL user-defined type locator variable>
| <COBOL REF variable>
<COBOL CLOB variable>::=
[USAGE [IS]] SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
[CHARACTER SET [IS] <character set specification>]
<COBOL NCLOB variable>::=
[USAGE [IS]] SQL TYPE IS NCLOB <leftparen> <large object length> <rightparen>
[CHARACTER SET [IS] <character set specification>]
<COBOL BLOB variable>::=
[USAGE [IS]] SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<COBOL user-defined type variable>::=
[USAGE [IS]] SQL TYPE IS <user-defined type> AS <predefined type>
<COBOL CLOB locator variable>::=
[USAGE [IS]] SQL TYPE IS CLOB AS LOCATOR
<COBOL BLOB locator variable>::=
[USAGE [IS]] SQL TYPE IS BLOB AS LOCATOR
<COBOL array locator variable>::=
[USAGE [IS]] SQL TYPE IS <collection type> AS LOCATOR
<COBOL user-defined type locator variable>::=
[USAGE [IS]] SQL TYPE IS <user-defined type name> AS LOCATOR
<COBOL REF variable>::=
[USAGE [IS]] SQL TYPE IS <reference type>
<Fortran variable definition>::=
<Fortran type specification> <Fortran host identifier> [{ <comma> <Fortran host identifier> }...]
--p
--small
--i
The standard documents 'CHARACTER KIND = n' but there is no explanation of the italic 'n' that is used. Presumably, it is an integer literal, hence <unsigned integer>.
--/i
--/small
--/p
<Fortran type specification>::=
CHARACTER [<asterisk> <length>] [CHARACTER SET [IS] <character set specification>]
| CHARACTER KIND <equals operator> <unsigned integer> [<asterisk> <length>] [CHARACTER SET [IS] <character set specification>]
| BIT [<asterisk> <length>]
| INTEGER

```
            |       REAL
            |       DOUBLE PRECISION
            |       LOGICAL
            |       <Fortran derived type specification>
<Fortran derived type specification>::=
                    <Fortran CLOB variable>
            |       <Fortran BLOB variable>
            |       <Fortran user-defined type variable>
            |       <Fortran CLOB locator variable>
            |       <Fortran BLOB locator variable>
            |       <Fortran user-defined type locator variable>
            |       <Fortran array locator variable>
            |       <Fortran REF variable>
<Fortran CLOB variable>::=
                    SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
                    [ CHARACTER SET [ IS ] <character set specification> ]
<Fortran BLOB variable>::=
                    SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<Fortran user-defined type variable>::=
                    SQL TYPE IS <user-defined type> AS <predefined type>
<Fortran CLOB locator variable>::=
                    SQL TYPE IS CLOB AS LOCATOR
<Fortran BLOB locator variable>::=
                    SQL TYPE IS BLOB AS LOCATOR
<Fortran user-defined type locator variable>::=
                    SQL TYPE IS <user-defined type name> AS LOCATOR
<Fortran array locator variable>::=
                    SQL TYPE IS <collection type> AS LOCATOR
<Fortran REF variable>::=
                    SQL TYPE IS <reference type>
<MUMPS variable definition>::=
                     <MUMPS numeric variable> <semicolon>
            |        <MUMPS character variable> <semicolon>
            |        <MUMPS derived type specification> <semicolon>
<MUMPS numeric variable>::=
                     <MUMPS type specification> <MUMPS host identifier>
                    [ { <comma> <MUMPS host identifier> }... ]
<MUMPS type specification>::=
                    INT
            |       DEC [ <leftparen> <precision> [ <comma> <scale> ] <rightparen> ]
            |       REAL
<MUMPS character variable>::=
                    VARCHAR <MUMPS host identifier> <MUMPS length specification>
                    [ { <comma> <MUMPS host identifier> <MUMPS length specification> }... ]
<MUMPS length specification>::=
                     <leftparen> <length> <rightparen>
<MUMPS derived type specification>::=
                     <MUMPS CLOB variable>
            |        <MUMPS BLOB variable>
            |        <MUMPS user-defined type variable>
            |        <MUMPS CLOB locator variable>
            |        <MUMPS BLOB locator variable>
            |        <MUMPS user-defined type locator variable>
            |        <MUMPS array locator variable>
            |        <MUMPS REF variable>
<MUMPS CLOB variable>::=
                    SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
                    [ CHARACTER SET [ IS ] <character set specification> ]
<MUMPS BLOB variable>::=
```

```
                    SQL TYPE IS BLOB <leftparen>  <large object length>  <rightparen>
<MUMPS user-defined type variable>::=
                    SQL TYPE IS <user-defined type> AS <predefined type>
<MUMPS CLOB locator variable>::=SQL TYPE IS CLOB AS LOCATOR
<MUMPS BLOB locator variable>::=SQL TYPE IS BLOB AS LOCATOR
<MUMPS user-defined type locator variable>::=
                    SQL TYPE IS <user-defined type name> AS LOCATOR
<MUMPS array locator variable>::=
                    SQL TYPE IS <collection type> AS LOCATOR
<MUMPS REF variable>::=SQL TYPE IS <reference type>
<Pascal variable definition>::=
                     <Pascal host identifier> [ { <comma>  <Pascal host identifier> }... ] <colon>
                     <Pascal type specification>  <semicolon>
<Pascal type specification>::=
                    PACKED ARRAY <left bracket> 1 <double period>  <length>  <right bracket>
                    OF CHAR [ CHARACTER SET [ IS ] <character set specification> ]
          |         PACKED ARRAY <left bracket> 1 <double period>  <length>  <right bracket>
OF BIT
          |         INTEGER
          |         REAL
          |         CHAR [CHARACTER SET [ IS ] <character set specification> ]
          |         BIT
          |         BOOLEAN
          |          <Pascal derived type specification>
<Pascal derived type specification>::=
                     <Pascal CLOB variable>
          |          <Pascal BLOB variable>
          |          <Pascal user-defined type variable>
          |          <Pascal CLOB locator variable>
          |          <Pascal BLOB locator variable>
          |          <Pascal user-defined type locator variable>
          |          <Pascal array locator variable>
          |          <Pascal REF variable>
<Pascal CLOB variable>::=
                    SQL TYPE IS CLOB <leftparen>  <large object length>  <rightparen>
                    [ CHARACTER SET [ IS ] <character set specification> ]
<Pascal BLOB variable>::=
                    SQL TYPE IS BLOB <leftparen>  <large object length>  <rightparen>
<Pascal user-defined type variable>::=
                    SQL TYPE IS <user-defined type> AS <predefined type>
<Pascal CLOB locator variable>::=SQL TYPE IS CLOB AS LOCATOR
<Pascal BLOB locator variable>::=SQL TYPE IS BLOB AS LOCATOR
<Pascal user-defined type locator variable>::=
                    SQL TYPE IS <user-defined type name> AS LOCATOR
<Pascal array locator variable>::=
                    SQL TYPE IS <collection type> AS LOCATOR
<Pascal REF variable>::=SQL TYPE IS <reference type>
<PL/I variable definition>::=
                     {DCL | DECLARE }
                     { <PL/I host identifier> | <leftparen>  <PL/I host identifier> [ { <comma>  <PL/I
host identifier> }... ] <rightparen> }
                      <PL/I type specification> [ <character representation>... ] <semicolon>
<PL/I type specification>::=
                     { CHAR | CHARACTER } [ VARYING ] <leftparen>  <length>  <rightparen>
                     [ CHARACTER SET [ IS ] <character set specification> ]
          |          BIT [ VARYING ] <leftparen>  <length>  <rightparen>
          |          <PL/I type fixed decimal>  <leftparen>  <precision> [ <comma>  <scale> ]
<rightparen>
          |          <PL/I type fixed binary> [ <leftparen>  <precision>  <rightparen> ]
```

```
        |       <PL/I type float binary> <leftparen> <precision> <rightparen>
        |       <PL/I derived type specification>
<PL/I type fixed decimal>::=
                { DEC | DECIMAL } FIXED
        |       FIXED { DEC | DECIMAL }
<PL/I type fixed binary>::=
                { BIN | BINARY } FIXED
        |       FIXED { BIN | BINARY }
<PL/I type float binary>::=
                { BIN | BINARY } FLOAT
        |       FLOAT { BIN | BINARY }
<PL/I derived type specification>::=
                <PL/I CLOB variable>
        |       <PL/I BLOB variable>
        |       <PL/I user-defined type variable>
        |       <PL/I CLOB locator variable>
        |       <PL/I BLOB locator variable>
        |       <PL/I user-defined type locator variable>
        |       <PL/I array locator variable>
        |       <PL/I REF variable>
<PL/I CLOB variable>::=
                SQL TYPE IS CLOB <leftparen> <large object length> <rightparen>
                [ CHARACTER SET [ IS ] <character set specification> ]
<PL/I BLOB variable>::=
                SQL TYPE IS BLOB <leftparen> <large object length> <rightparen>
<PL/I user-defined type variable>::=
                SQL TYPE IS <user-defined type> AS <predefined type>
<PL/I CLOB locator variable>::=SQL TYPE IS CLOB AS LOCATOR
<PL/I BLOB locator variable>::=SQL TYPE IS BLOB AS LOCATOR
<PL/I user-defined type locator variable>::=
                SQL TYPE IS <user-defined type name> AS LOCATOR
<PL/I array locator variable>::=SQL TYPE IS <collection type> AS LOCATOR
<PL/I REF variable>::=SQL TYPE IS <reference type>
<embedded SQL end declare>::=
                <SQL prefix> END DECLARE SECTION [ <SQL terminator>]
<embedded SQL MUMPS declare>::=
                <SQL prefix> BEGIN DECLARE SECTION
                [ <embedded character set declaration> ] [ <host variable definition>... ]
                END DECLARE SECTION <SQL terminator>
<embedded SQL host program>::=
                <embedded SQL Ada program>
        |       <embedded SQL C program>
        |       <embedded SQL COBOL program>
        |       <embedded SQL Fortran program>
        |       <embedded SQL MUMPS program>
        |       <embedded SQL Pascal program>
        |       <embedded SQL PL/I program>
<embedded SQL Ada program>::=!! (See the Syntax Rules.)
<embedded SQL C program>::=!! (See the Syntax Rules.)
<embedded SQL COBOL program>::=!! (See the Syntax Rules.)
<embedded SQL Fortran program>::=!! (See the Syntax Rules.)
<embedded SQL MUMPS program>::=!! (See the Syntax Rules.)
<embedded SQL Pascal program>::=!! (See the Syntax Rules.)
<embedded SQL PL/I program>::=!! (See the Syntax Rules.)
<embedded SQL statement>::= <SQL prefix> <statement or declaration> [ <SQL terminator> ]
<statement or declaration>::=
                <declare cursor>
        |       <dynamic declare cursor>
        |       <temporary table declaration>
```

```
            |           <embedded authorization declaration>
            |           <embedded path specification>
            |           <embedded transform group specification>
            |           <embedded exception declaration>
            |           <handler declaration>
            |           <SQL-invoked routine>
            |           <SQL procedure statement>
<dynamic declare cursor>::=
                        DECLARE <cursor name> [ <cursor sensitivity> ] [ <cursor scrollability> ]
CURSOR
                        [ <cursor holdability> ] [ <cursor returnability> ] FOR <statement name>
<embedded authorization declaration>::=DECLARE <embedded authorization clause>
<embedded authorization clause>::=
                        SCHEMA <schema name>
            |           AUTHORIZATION <embedded authorization identifier>
                        [ FOR STATIC { ONLY | AND DYNAMIC } ]
            |           SCHEMA <schema name> AUTHORIZATION <embedded authorization
identifier>
                        [ FOR STATIC { ONLY | AND DYNAMIC } ]
<embedded authorization identifier>::= <module authorization identifier>
<embedded path specification>::= <path specification>
<embedded transform group specification>::= <transform group specification>
<embedded exception declaration>::=WHENEVER <condition>  <condition action>
<condition>::= <SQL condition>
--p
--small
--i
The standard documents 'SQLSTATE ( <SQLSTATE class value> [ , <SQLSTATE subclass value> ] )',
but it is not clear why the <leftparen> , <comma> and <rightparen> are not designated more accurately.
--/i
--/small
--/p
<SQL condition>::=
                         <major category>
            |           SQLSTATE <leftparen>  <SQLSTATE class value> [ <comma>  <SQLSTATE
subclass value> ] <rightparen>
            |           CONSTRAINT <constraint name>
<major category>::=SQLEXCEPTION | SQLWARNING | NOT FOUND
<SQLSTATE class value>::= <SQLSTATE char>  <SQLSTATE char> !! (See the Syntax Rules.)
<SQLSTATE char>::= <simple Latin upper case letter> | <digit>
<SQLSTATE subclass value>::= <SQLSTATE char>  <SQLSTATE char>  <SQLSTATE char> !! (See
the Syntax Rules.)
<condition action>::=CONTINUE | <go to>
<go to>::={ GOTO | GO TO } <goto target>
<goto target>::= <host label identifier> | <unsigned integer> | <host PL/I label variable>
<host label identifier>::=!! (See the Syntax Rules.)
<host PL/I label variable>::=!! (See the Syntax Rules.)
<interval primary>::=
                         <value expression primary> [ <interval qualifier> ]
            |            <interval value function>
<module authorization clause>::=
                        SCHEMA <schema name>
            |           AUTHORIZATION <module authorization identifier>
                        [ FOR STATIC { ONLY | AND DYNAMIC } ]
            |           SCHEMA <schema name> AUTHORIZATION <module authorization
identifier>
                        [ FOR STATIC { ONLY | AND DYNAMIC } ]
<preparable statement>::=
                         <preparable SQL data statement>
```

| <preparable SQL schema statement>
| <preparable SQL transaction statement>
| <preparable SQL control statement>
| <preparable SQL session statement>
| <preparable implementation-defined statement>
<preparable SQL data statement>::=
<delete statement: searched>
| <dynamic single row select statement>
| <insert statement>
| <dynamic select statement>
| <update statement: searched>
| <preparable dynamic delete statement: positioned>
| <preparable dynamic update statement: positioned>
<dynamic single row select statement>::= <query specification>
<dynamic select statement>::= <cursor specification>
<preparable dynamic delete statement: positioned>::=
DELETE [FROM <target table>] WHERE CURRENT OF [<scope option>] <cursor name>
<preparable dynamic update statement: positioned>::=
UPDATE [<target table>] SET <set clause list> WHERE CURRENT OF [<scope option>] <cursor name>
<preparable SQL schema statement>::= <SQL schema statement>
<preparable SQL transaction statement>::= <SQL transaction statement>
<preparable SQL control statement>::= <SQL control statement>
<preparable SQL session statement>::= <SQL session statement>
<preparable implementation-defined statement>::=!! (See the Syntax Rules.)
--hr
--h2 END OF SQL-99 GRAMMAR
--/h2
--hr
--h2 Notes on Automatically Converting the SQL-99 Grammar to a YACC Grammar
--/h2
--p
Automatic translation of this grammar is non-trivial for a number of reasons. One is that the grammar has a number of actions '!!
(See the Syntax Rules.)' which cannot be translated automatically. Another is that the grammar contains rules that are usually better handled by the lexical analyzer than the grammar proper. Then there are incomplete rules such as those which reference parts 6
to 10 (they are not defined; indeed, part 7, which was going to be SQL/Temporal, is in complete abeyance), and the packages (almost completely undefined in the grammar).It is not clear whether these can be ignored, or annotated out of the way.
--/p
--p
Another complication is automatically generating rules to deal with optional components and repetitive components in the grammar. Square brackets do not contain alternative non-terminals; all those expressions are contained within curly brackets within the square brackets. However, some square brackets do contain alternative terminals. Curly brackets contain and group mandatory elements. However, they are usually used in conjunction with the 'one or more times' repeater ellipsis '...' mark.
--/p
--hr

واژه نامه

انگلیسی به فارسی

A

Abort	طرد
Abstract machine	ماشين انتزاعي
Abstract Data Type (ADT)	نوع داده اي انتزاعي
Abstract Data Type (ADT)	نوع داده مجرد
Abstractive	انتزاعي
Abstractive construct	ساخت انتزاعي
Access rules	قواعد دستيابي
Access Method (AM)	شيوه دستيابي
Access right	حقوق دستيابي
Access strategy	استراتژي دستيابي
Accessability	دستيابي بهتر
Accuracy	دقت
Action	عملي
Active rule	قاعده فعال
Active Data Objects (ADO)	شيئ هاي داده اي فعال
Active Database Management System (ADBMS)	سيستم فعال مديريت پايگاهداده
Active dictionary	ديكشنري فعال
Active DBMS (ADBMS)	سيستم فعال
Activity diagram	نمودار فعاليت
Actor	كنشگر
Ad hoc (Unplanned)	موردي
Aggregate function	تابع جمعي يا گروهي
Aggregation	تجمع
ALL-Key	تمام كليد
Alternate Key(A K)	كليد ديگر
Analogue	قياسي
Anomaly	آنومالي
Antisemijoin	ضد نيم پيوند
Application perspective	ديدگاه كاربردي
Application Programming Interface (API)	واسط برنامه سازي كاربرد
Application developer	كاربرد ساز (بهره بردار)
Application Program (AP)	برنامه كاربردي (كاربردار)
Application Program conversion	تبديل برنامه هاي كاربردي
Application system	سيستم كاربردي
Arity	آريتي
Assertion	اظهار
Asset	سرمايه
Association	بستگي
Asymmetry	عدم تقارن
Atomic	تجزيه نشدني
Atomicity	تجزيه ناپذيري
Attribute	صفت
Attribute features	جنبه هاي صفت
Attribute inheritance	وراثت صفت
Attribute preserving	حافظه صفات محدوديت
Attribute constraint	صفتي
Attributes set	مجموعه صفات
Augmentation	افزايش

Authorization	مجاز شماري
Authorization	شناسه مجاز
Identifier (AuthID)	شماري
Automatic Design	ابزار (امكان)
Tool (ADT)	طراحي خودكار
Automatic navigation	ناوش خودكار
Axiom	ارز آغازه (اصل موضوعه

B

Bachman diagram	نمودار با چمن
Back office server	افزارگان پشت صحنه
Back-end	پسا
Backend	ماشين خدمتگزار
Dbserver	پايگاهي پسا
Backup	پشتيبان
Bags	توده
Base relation	رابطه مبنا
Base attribute	مبنا
Base Table (BT)	جدول مبنا(پايه)
Basic data type	نوع داده اي مبنايي
Batch	يكجا
Batch mode	اسلوب يكجا
Behavioural	رفتاري
Benchmarking	محك زني
Bi- structure	دو ساختاره
Bi-directional association	بستگي دو سويه
Body	پيكر (بدنه)
Bound	بسته
Box data structure	ساختار داده اي جعبه اي
BOYCE	بويس
Boyce-Codd Normal Form	صورت نرمال بايس-كاد
Bridge	پل
Browsing	گشت زني
Business rule	قواعد تجاري
Business rules	قواعد فعاليتها
Buttleneck	تنگنا

C

C.J.DATE	ديت
Cache memory	حافظه نهان
CAD/CAM	طراحي به كمك كامپوتر-توليد به كمك كامپيوتر
Calculated attribute	صفت محاسبه شده
Call Level Interface (CLI)	واسط در سطح فراخوان (CLI
Candidate domain	ميدان كانديد
Candidate Key (CK)	كليد كانديد
Canonical cover	پوش كانونيك
Cardinality	كارديناليتي نسبت
Cardinality ratio	كارديناليتي
Cascade	تسلسلي
Catalog manager	مدير كاتالوگ
Catalogue	كاتالوگ
Categorization	دسته بندي
Category	دسته
Centralized	متمركز
Centralized control	كنترل متمركز
Centralized database	پايگاه متمركز
Certification	تكنيك تاييد (تصديق)
Chasm trap	دام شكاف
Check constraints	محدوديت ناظر به ستون
Checkpointing	نقطه وارسي
CHEMBERLIN	چمبرلين
CHEN	چن
CHEN notation	نمودار چن
Class	رده
Class diagram	نمودار رده
Classification	رده بندي
Clause	فراكرد (كلاز)
Clean design	طراحي بهتر و واضح
Client / Multi-Server (C / MS)	يك مشتري-چند خدمتگزار كامپيوتر
Client machine	مشتري
Client / Server (C/S)	مشتري/ خدمتگزار پايگاه‌داده
Client / Server Database (C/S DB)	مشتري- خدمتگزار
Closed World	جهان بسته
Closed World	تفسير جهان

interpretation	بسته
Closure	بستار
Clustering	خوشه واري
CODASYL data model	مدل داده اي كوداسيل
CODD	كاد
Collection	گردايه
Command driven	هدايت شده با فرمان
Command Language Interface (CLI)	واسط زبان فرمان
Commit	تثبيت
Communative	انتقال دهندگي
Communication	همرسانش (ارتباط)
Comparand	قياسوند
Complete relational system	سيستم رابطه‌اي كامل
Complex	پيچيده (مختلط)
Complex Data Type (CDT)	نوع داده اي پيچيده
Complex query	پرسش پيچيده
Component	سازند
Composite	مركب
Composition	تركيب
Computational completeness	كمال محاسباتي
Conceptual schema	شماي ادراكي
Conceptual design	طراحي ادراكي
Conceptual level	سطح ادراكي (مفهومي)
Conceptual redundancy	افزونگي ادراكي
Conceptual view	ديد ادراكي
Conceptual / Internal mapping	نگاشت ادراكي / داخلي
Concurrent	همزمان
Concurrent schedule	طرح اجراي همروند
Conference On Data System Langeage (CODASYL)	سيستم كوداسيل
Configuration	پيكربندي
Connection traps	دامهاي پيوندي
Connector	پيوند دهنده
Consistency	سازگاري
Constraint	محدوديت
Constraint specification language	زبان توصيف محدوديت
Constructor	ساختگر
CONTAINS	شامل است
Containtement	شمول
Continuous operation	تداوم عمليات
Conventional	سنتي
Conventional system	سيستم متعارف
Correcthness	صحت
Cost-benefit analysis	تحليل برآورد هزينه / مزايا
Cover	پوش
CREATE view	ايجاد ديد
Cross join	پيوند ضرب
Current AuthID	شناسه جاري
Cursor	مكان نما

D

Dangling	معلق
Data Sub Language (DSL)	زبان داده اي فرعي
Data compression	فشرده سازي داده ها
Data Structure(DS)	ساختار داده اي
Data accessibility	دسترسي(بهتر) به داده ها
Data Administrator (DA)	مدير داده
Data bank	بانك داده ها
Data communication manager	مدير انتقال داده
Data Control (DC)	كنترل داده
Data Control Language (DCL)	دستورات كنترل داده ها
Data conversion and data transfer	تبديل و انتقال داده ها
Data Definition	تعريف داده
Data Definition Language (DDL)	دستورات تعريف داده ها
Data encryption	نهان نگاري داده
Data entry	وارد كردن داده
Data export	صدور داده ها

Data extraction	استخراج داده ها
Data Flow Diagram (DFD)	نمودار روند داده
Data import	ورودي داده ها
Data inconsistency	ناسازگاري داده
Data independence	استقلال داده اي
Data Management System (DMS)	سیستم مدیریت داده ها
Data Manager	مدیر داده ها
Data Manipulation	دستور عملیات در داده ها دستورات
Data Manipulation Language (DML)	عملیات روي داده ها
Data marts	بازار داده ها
Data materialization	ساختن داده
Data migration	کوچاندن داده
Data mining and knowledge discovery system	سیستم داده کاوي و کشف شناخت
Data Model	مدل داده اي
Data naming	نامگذاري داده ها
Data placement	جايدهي داده
Data sharing	اشتراك داده ها
Data source	منبع داده اي
Data type	نوع داده اي
Data type extensibility	گسترش پذيري نوع داده اي الگوي استفاده
Data usage pattern	از داده ها
Data Warehousing	سیستم انبارش داده ها
Data Communication	داده رساني
Data keeping- data processing	داده داري- داده پردازي
Database	پایگاه‌داده
Database growth	رشد پایگاه‌داده
Database Language (DBL)	زبان پایگاهي
Database space	فضاي پایگاه‌داده وضعیت
Database state	پایگاه‌داده تنظیم پایگاه‌داده
Database tuning	ها
Database administration	امکان مدیریتي براي مدیر

support	پایگاه
Database Administrator(DBA	مدیر پایگاه‌داده ها
Database approach	مشي پایگاهي محدودیت
Database constraint	پایگاهي
Database environment	مدیریت محیط
management	پایگاه‌داده ها
Database Machine (DBM)	ماشین پایگاه‌داده ها
Database Management System (DBMS)	سیستم مدیریت پایگاه‌داده ها
Database Protection	ایمني و حفاظت پایگاه‌داده ها
Database quality	کیفیت پایگاه‌داده ها
Database server	خدمتگزار پایگاه‌داده ها
Datalogical	داده شناسانه
Datalogical	دیتالوگ
Data-object	شیئ-داده
DBMS vendors	فروشندگان سیستمها
DBMS-web integration	ادغام در وب
DC manager	مدیر انتقال داده ها
Deadlock	بن بست
Decentralized architecture	معماري نامتمرکز
Decision making	تصمیم گیري
Decision support	پشتیباني تصمیم
Decision Support System (DSS)	سیستم پشتیبان تصمیم
Declarative	اعلاني
Decomposition	تجزیه
Deduction	استنتاج پایگاه‌داده
Deductive database	استنتاجي
Default length	طول پیش نهاده
Default value	مقدار پیش نهاده
Deferred checking	وارسي با تاخیر بهنگام سازي با
Deffered update	تاخیر
Defining attribute	صفت معرف
Degree	درجه

Denormalization	کاهش درجه (سطح) نرماليتي
Dependency theory	تئوري وابستگي
Derived	مشتق
Derived relation	رابطه مشتق
Descriptive	توصيفي
Descriptive attribute	صفت توصيفي
Design by analysis	طراحي با تحليل
Design tool	ابزار طراحي
Destructor	ويرانگر
Detabase perspctive	ديدگاه پايگاهي
Determinant	دترمينان
Developement	ايجاد
Device configuration	پيكربندي رسانه
Diadic	دو عملوندي
Direct materialization	ساختن مستقيم
Discriminator	مميزه
Disjoint	مجزا
Diskless machine	ماشين بدون ديسك
Distributed	توزيع شده
Distributed Database (DDB)	پايگاهداده توزيع شده (پراكنده)
Divide	تقسيم
Document server	خدمتگزار مدارك و اسناد
Documentation	مستند سازي
Documented procedure	رويه مستند
Domain	ميدان
Domain - Key Normal Form	صورت نرمال ميدان-كليدي
Domain oriented	حساب ميداني محدوديت
Domain constraint	ميداني
Domain variable	متغير ميداني
Downsizing	كاهش‌اندازه
Driver manager	مدير درايور
DROP view	حذف ديد
Dummy record	ركورد ساختگي
Dumping	نسخه برداري
Durability	مانايي (دوام)
Dynamic data	داده پويا
Dynamic SQL	پوياSQL

E

Elementary type	نوع ابتدايي
E-mail server	خدمتگزار پست الكترونيكي
Embedded	ادغام شده
Embedded Multi Valued Dependency (EMVD)	وابستگي چند مقداري ادغام شده
End user	كاربر پاياني
Engine	موتور
Enhanced (Extended) ER	روش EER
Entity Identifier (EID)	شناسه موجوديت
Entity integrity rule	قاعده جامعيت موجوديتي
Entity relationship	موجوديت-ارتباط (ER)
Entity type	نوع موجوديت
Entity types set	مجموعه انواع موجوديتها
Equational Dependency (ED)	وابستگي برابري
Equi-Join	پيوند به شرط تساوي
ER diagram	نمودار ER
Essential	اساسي
Event journaling	رويداد نگاري
Event- condition-action rules	قواعد رويداد-شرط-اقدام
Event Flow Diagram (EFD)	نمودار روند رويداد
Evolutionary Prototyping	نمونه سازي گسترشي
Execution plan	طرح اجرا
Executive Information System	سيستم اطلاعات اجرايي
Existance dapendency	وابستگي وجود
Existential	سور وجودي
Exploitation	بهره برداري
Expression relation	رابطه عبارتي
Extended cartesian product	ضرب كاتزين گسترش يافته
Extended data type	نوع داده اي گسترش يافته

Extension	گسترده
External level	سطح خارجي
External/Conceptual mapping (E / C)	نگاشت خارجي / ادراكي / خارجي
External / External	خارجي
External schema	شماي خارجي
External view	ديد خارجي

F

Facts	واقعيات
FAGIN	فاگين
Fail (Failure)	عيب (نقص)
Fan trap	دام يك-چندي
Fault tolerance	تحمل خرابي
FD ' s diagram	نمودار وابستگيهاي تابعي
FD' s preserving decomposition	تجزيه حافظ وابستگيها
Federated	فدرال
Fifth Normal Form	صورت پنجم نرمال
File server	خدمتگزار فايل
Filing System (FS)	سيستم فايلينگ
First Normal Form	صورت نخست نرمال
Five- minute rule	قاعده پنج دقيقه
Flat	مسطح
Flat Linear record	ركورد خطي مسطح
Foreign Key (FK)	كليد خارجي
Form generator	مولد فرم هدايت شده با
Form driven	فرم
Formal	صوري
Four(4)GL	زبان نسل چهارم
Fourth Normal Form	صورت چهارم نرمال
Free	آزاد
Front-end	پيشا
Full outer join	فرا پيوند كامل
Fully Functional Dependency (F FD)	وابستگي تابعي تام (كامل)
Fully relational system	سيستم رابطه‌اي تام
Functional Dependency (FD)	وابستگي تابعي
Functional analysis	تحليل عملكردي
Functionalality	عملياتي
Fuzzy data mining and knowledge discovery system	سيستم ژولاي داده كاوي و كشف دانش

G

Gantt chart	چارت گانت
Gateway	گدار
General purpose	همه منظوره
General unification	يكسان عمومي
Generalization	تعميم
Global application	كاربردي سرتاسري
Global conceptual schema	شماي ادراكي جامع
Global System Catalog (G S C)	كاتالوگ جامع
GRANT	اعطاء
Graphic driven	كاربر گرافيكي
Greater then Join	پيوند به شرط بزرگتر
Greater then or Equal	پيوند به شرط مساوي يا بزرگتر
Grow and shrink	رشد و كاهش

H

Harmonic mean	متوسط هارمونيك
Heading	سر آيند
HEATH theorem	قضيه هيث
Help	كاربريار
Heterogenous	ناهمگن
Heterogenous parrallel systems control database	پايگاه‌داده هاي كنترل سيستم هاي ناهمگن موازي
Hidden data	داده نهان
Hierarchycal Data Structure (HDS)	ساختار داده اي سلسله مراتبي
Hierarchycal Database (HDB)	پايگاه‌داده سلسله مراتبي
Homogenous	همگن
Horizontal decomposition	تجزيه افقي
Horizontal fragmentation	پارسازي (تقطيع) افقي
Host Language	زبان ميزبان

	پایگاه‌داده
Host database	میزبان خدمتگزار
Host server	میزبان
Hybrid	پارسازي
fragmentation	ترکیبي
Hyper graph	ساختار داده اي
Database Structure	هایپرگرافي

I

Identifying	شناساي
Identifying entity	موجودیت شناسا
Identifying relationship	ارتباط شناسا
Identity projection	پرتو همانی
Immediate checking	وارسي بلافاصله
Immediate update	بهنگام سازي بلافاصله
Implementation	پیاده سازي
Implementation diagram	نمودار پیاده سازي
Import / Export interface	واسط ورود / صدور
Inapplicable	غیر قابل اعمال
Inclusion Dependency	صورت نرمال وابستگي شمول
Inclusion Dependency (ID)	وابستگي شمول
Independent (Stand alone)	مستقل (خودکفا) بازسازي
Index rebuilding	شاخص
Indirect materialization	ساختن غیر مستقیم
Induction	استقراء
Inference rules	قواعد استنتاج
Informal	غیر صوري
Information space	فضاي اطلاعاتي
Information Storage and Retrieval (ISR)	ذخیره و بازیابي اطلاعات
Information System	سیستم اطلاعاتي
Information Technology (IT)	تکنولوژي اطلاعات
Informative	ارتباط دهندگي
Inner query	پرسش دروني
Inplace updating	بهنگام سازي در جا
Instance	نمونه
Instances set	مجموعه نمونه
Integrated	مجتمع-یکپارچه
Integrated system	سیستم جامعه (یکپارچه)
Integrity	جامعیت
Integrity rule (constraint)	قاعده- محدودیت جامعیت محدودیت
Integrity constraint	جامعیتي
Intelligent system	سیستم هوشمند
Intension	رابطه
Inter related	بهم مرتبط اندرکنش
Interaction	(تعامل)
Interactive	اندرکنشي
Interactive mode	اسلوب تعاملي
Intermediate relation	رابطه بینابیني
Internal level	سطح داخلي
Internal view	دید داخلي
Internal schema	شماي داخلي
Interoperability	تعامل
Interpretation	تفسیر
Intrinsic	ذاتي
Intuitive	شهودي
Inverted list data structure	ساختار داده اي لیستهاي وارون
Irreducible	کاهش ناپذیر
Irreduciblity	کاهش ناپذیري
IS-A	هست یك...
IS-A PART-OF	جزیي است از...
Islands of information	جزایر اطلاعاتي
Isolation	جدایي (انفراد)

J

Java-script	جاوا اسکریپت
Job	کار
Join	پیوند
Join attribute	صفت پیوند
Join Dependency	وابستگي پیوندي
Join selectivity factor	ضریب گزینش عملکرد پیوند

K

Key-based inclusion	وابستگي شمول مبتني بر کلید

dependency

Key preserving	داراي كليد
Key-Complete	صورت نرمال
Normal Form	كاملا كليد
Knowledge Base Management System (KBMS)	سيستم مديريت پايگاه شناخت
Knowledge-based system	سيستم شناختي

L

Large Database	پايگاه داده بزرگ
Layer	لايه
Left augmentation	افزايش چپ
Left outer join	فرا پيوند چپ
Less than Join	پيوند به شرط كوچكتر
Less than or Equal Join	پيوند به شرط مساوي يا كوچكتر
Library function	تابع كتابخانه اي
Link	پيوند
Load	بارگذاري
Local	محلي
Local application	كاربردي محلي
Local conceptual schema	شماي ادراكي محلي
Local database	پايگاه‌داده محلي
Location anatomy	خود مختاري محلي
Lock granularity	اندازه واحد قفل پذير
Lock granular	واحد قفل شدني (قفل پذير)
Locking	قفل گذاري
Locking technique	تكنيك قفل گذاري
Locking / Logging	قفل گذاري-توليد فايلهاي ثبت
Logic programming	زبان برنامه
Language	سازي منطق
logical Data Independence	استقلال داده اي منطقي
Logical construct	ساخت منطقي
Logical design	طراحي منطقي
Logical design refinement	بهبود طراحي منطقي
Logical sequence	توالي منطقي
Logical unit of work	واحد منطقي كار
Lookup table	جدول جستجو
Loop trap	دام حلقه اي
Loosely coupled	پيوند سست
Lost decomposition	تجزيه با حذف

M

Main computer	كامپيوتر بزرگ معماري حول
Main frame centric	كامپيوتر بزرگ
Main Memory Database (MMDB)	پايگاه‌داده هاي حافظه اصلي
Manipulative	عملياتي
Many to many	چند به چند
Map driven	هدايت شده با نقشه
Mapping	نگاشت
Materialized view	ديد به عينيت در آمده
Meaning	معنا
Mediator	ميانجي
Member	عضو
Membership condition	شرط عضويت
Menu driven	هدايت شده با منو
Menu generator	مولد منو
Message server	خدمتگزار پيام
Meta byte	متا بايت
Meta-constraint	متا محدوديت
Meta data	متا داده
Meta rules	متا قواعد
Micro real world	خرد جهان واقع
Middleware	ميان افزار
Mini computer	كامپيوتر متوسط
Minimal	كهينه
Minimal relational system	سيستم رابطه‌اي حداقل
Missing data	داده نهست
Missing information	اطلاع نهست
Mixed design method	روش تركيبي
Mobile office	دفتر كار همراه
Mobile database	خدمات

server	پایگاه‌داده هاي موبایل
Mobile database system	سیستم پایگاه‌داده اي با معماري
Mobile database system	سیستم پایگاههاي همراه
Mobile DB	پایگاه‌داده موبایل
Mobility and personal database	پایگاه‌داده هاي شخصي و همراه
Module	واحد شناسه براي
Module AuthID	تعریف ماجول
Monadic	تك عملوندي
Monitoring	نظارت
Mono- server	تك خدمتگزار
Multi database management system	سیستم مدیریت چند پایگاهي
Multi Database System (MDBS)	معماري چند پایگاهي
Multi ring	چند حلقه اي
Multi- Client / Server (MC /S)	چند مشتري- یك خدمتگزار
Multi- Client / Multi- Server(MC / MS)	چند مشتري- چند خدمتگزار
Multi- media	چند رسانه اي
Multi database	چند پایگاهي
Multi-Dimensional OLAR (MDOLAP)	سیستم پردازش تحلیلي برخط
Multimedia-data mining	سیستم داده کاوي چند رسانه اي
Multimember set	مجموعه چند عضوي
Multiple inheritance	وراثت (توارث) چند گانه
Multiple-user access	دستیابي چند کاربردي
Multiplicity	چندي
Multi query	چند پرسشي
Multi-server	چند خدمتگزار
Multi set	چند مجموعه
Multi-user	چند کاربري
Multi valued	چند مقداري
Multi valued Dependency (MVD)	وابستگي چند مقداري
Multi version concurrency	همروندي چند نسخه اي
Multi versioning	تکنیك چند نسخه سازي

N

Named data items	فقره داده نامدار
Named relation	رابطه نامدار
Natural Join (NJ)	پیوند طبیعي
Natural language driven	کاربر با زبان طبیعي
Natural language like	شبه زبان طبیعي
Nature	ماهیت
Navigation	ناوش (غواصي)
Nested Relational Query Language (NRQL)	زبان پرسش با رابطه غیر نرمال
Nested statement	احکام مرکب (تو در تو)
Nested query	پرسش تو در تو
Nested relation	رابطه تودر تو
Network	شبکه
Network Data Structure (NDS)	ساختار داده اي شبکه اي
Network Database Language (NDL)	زبان سیستم شبکه اي (NDL)
Network centric	معماري حول شبکه
Network topology	توپولوژي شبکه
News server	خدمتگزار اخبار
No action	عدم اقدام
Non database approach	مشي ناپایگاهي
Non orthogonal design	طراحي نا متعامد
Non procedural	نا رویه اي (ناروشمند)
No additive	بي حشو
Non-loss (Lossless)	بي گمشدگي
Nontrivial	مهم (مطرح)
Normal relation	رابطه نرمال
Not Normal relation (NNR:	رابطه غیر نرمال
Not-Equi-Join	پیوند به شرط عدم تساوي
Nullifying	هیچمقدار گذاري
Null value	هیچمقدار

O

Object	شیئ

environment	سفارش
Ordering	نظم-آراستار
Oriented graph	گراف جهت دار
Orthogonal design	طراحي متعامد
Outer query	پرسش بيروني
Outer union	فرا اجتماع
Outplace updating	بهنگام سازي برون از جا
Overhead	فزونكاري (بيشكاري)
Overlapping	همپوشا
Owner	مالك

P

Page	صفحه
Panel interface	واسط پانل
Parallel processing	پردازش موازي
Parameter driven	هدايت شده با پارامتر
Parent-Child Link type (PCL)	نوع پيوند پدر-فرزندي
Partial	جزيي
Partial Key	كليد جزيي
Partial participation	مشاركت غيرالزامي (ناكامل)
Participant	شركت كننده
Passive dictionary	ديكشنري غير فعال
Peak performance	بيشترين كارايي
peak time	زمان اوج
Peer-to-peer	همطراز
Performance	كارايي
Permanent (planned)	هميشگي
Persistent	پايا
Persistent (Permanent) table	جدول ماندگار
Persistent Stored Modules (PSM)	ماجولهاي ذخيره شده ماندگار
Physical redundancy	افزونگي فيزيكي
Physical Data Independence (PDI)	استقلال داده اي فيزيكي
Physical design	طراحي فيزيكي
Piece	تكه
Plex	پلكس
Pointer	نشان نما -
Object Oriented Database Language	زبان پايگاهي شيئ گرا
Object Query Language (OQL)	زبان پرسش شيئ گرا
Object Oriented Language(OOL)	زبان شيئ گرا
Object behavior	رفتار شيئ
Object Data Model	مدل داده اي شيئ گرا
Object modeling	مدلسازي شيئي
Object Modeling Technique (OMT)	تكنيك مدلسازي شيئ
Object Oriented Database Management System (OODBMS)	سيستم مديريت پايگاه‌داده هاي شيئ گرا
Object Type	نوع شيئ
Object -Relational Database Management System (ORDBMS)	سيستم مديريت پايگاه‌داده هاي شيئ -رابطه‌اي
Object-Relational Data Model (ORDM)	مدل رابطه‌اي شيئ گرا
Occurrence	نمود
On-line	برخط (پيوسته)
On Line Analytic Processing (OLAP)	پردازشهاي تحليلي بر خط
One to many	يك به چند
One to one	يك به يك
One fact - one relation	يك بوده (واقعيت): يك رابطه
One level query	پرسش تك سطحي
One- minute rule	قاعده يك دقيقه
Online Analytical Processing (OLAP)	امكان پردازش تحليلي بر خط
Online controller	كنترولر بر خط
Online mode	اسلوب پيوسته (بر خط)
Operand	عملوند
Operational environment	محيط عملياتي
Operational mode	اسلوب عملياتي
Option	گزيدار (گزينه)
Order-entry	محيط ورود

	اشارهگر
Portability	جابجايي پذيري
Positional ordering	نظم مكاني
Post relational	پسا رابطه‌اي
Precompiled	پيش كامپايلر
Predicate	مسند
Predicate calculus	مسندات
Prenex form	صورت پيشوندي
Pre-relational	پيش رابطه‌اي
Prescriptive	دستوري
Presentation	نمايش
Price per TPS	هزينه به ازاء TPS
Price / performance ratio	نسبت هزينه / كارايي
Primary Key (PK)	كليد اصلي
Primary domain	ميدان اصلي
Prime attribute	صفت عمده
Printer server	خدمتگزار چاپ
Privacy	محرمانگي
Privilege	امتياز
Problem context	متن مسئله
Procedural	رويه اي
Procedure	رويه
Production rules	قواعد توليد
Program-data independence	استقلال برنامه از داده
Programming Language (PL)	زبان برنامه سازي
Programming completeness	كمال برنامه سازي
Project	پرتو
Propagating update	بهنگام سازي منتشر شونده
Proposition	گزاره
Protection	حفاظت
Prototype	نمونه نخست- نمونه سازي
Pseudo transitivity	شبه تعدي

Q

Qualified association	بستكي مقيد
Qualifier	قيد
Quantifier	چندي نما
Queries	بهنگام سازي
optimization	پرسشها
Queries set	مجموعه پرسش
Query Language (QL)	زبان پرسش
Query By Example (QBE)	پرسش به كمك مثال
Query modification	تغيير پرسش
Query optimizer	بهينه ساز پرسش پردازشگر
Query processor	پرسش خدمتگزار
Query server	پرسش
Query By Form (QBF)	پرسش به كمك فرم
Query optimization	بهينه سازي پرسش

R

Range variable	متغير طيفي
Range query	پرسش طيفي
Real	واقعي
Real world	جهان واقع
Real time application	كاربرد بلادرنگ
Real-time DBMS	پايگاه‌داده بي درنگ
Real-world object	اشياء جهان واقع
Real-time database management system	سيستم مديريت پايگاه‌داده هاي بي درنگ
Record type	نوع ركورد
Recovery	ترميم
Recursive relationship	ارتباط بازگشتي
Redundancy	افزونگي
Redundant Array of Inexpensive Disks (RAID)	آرايه ديسكهاي ارزان افزونه
Redundency Free Normal Form	صورت نرمال بي افزونگي
Reengineering	مهندسي دوباره
Reference	ارجاع
Reference diagram	نمودار ارجاع
Reference cycle	حلقه ارجاع
Reference graph	گراف ارجاع
Reference path	مسيرگراف
Referenced	مرجع

	شونده
Request manager	مدير درخواستها
Request modification	تغيير درخواست
Requirements Engineering (RE)	مهندسي نيازها
Requirements Prototyping	نمونه سازي نيازي
Research-Development (RD)	پژوهش-توسعه
Response time	زمان پاسخدهي
Responsing procedure	رويه پاسخگو
Restrict	تحديد-گزينش تعويقي
Restricted	(مشروط)
Restriction - Union Normal Form	صورت نرمال تحديد- اجتماع
Restriction predicate	مسند گزينش
Result relation	رابطه نتيجه
REVOKE	لغو
Right outer join	فرا پيوند راست مناسب
Rightsizing	سازي‌اندام
Rissanen	ريسانن
Role	نقش
Root only	فقط ريشه
R-system	سيستم R
Rule based optimizer	بهينه ساز مبتني بر قاعده
Rules	قواعد
Run time manager	مدير زمان اجرا
Run time supervisor	ناظر زمان اجرا
S	
Safe expression	عبارت مطمئن
Scalability	گسترش پذيري
Scaleup	افزايش مقياس كار (گسترش پذيري)
Schema	شما
Schema AuthID	شناسه تعريف شما
Schema definition	تعريف شما
Schema evolution	تغيير شما
Schema generator	واحد توليد شما
Referencing	رجوع كننده
Referential integrity rule	قاعده جامعيتي ارجاعي
Reflexive association	بستگي انعكاسي
Reflexivity	انعكاس
Relationally completeness	كمال رابطه‌اي
Relation assignment	انتساب رابطه محدوديت
Relation constraint	رابطه‌اي
Relation predicate	سند ارتباط
Relation schema	شماي رابطه
Relation state	حالت رابطه
Relation type generator	مولد نوع رابطه
Relation value (Relval)	مقدار رابطه‌اي
Relation variable (Relvar)	متغير رابطه‌اي
Relational expression	عبارت جبر رابطه‌اي
Relational synthesis	روش سنتز رابطه‌اي
Relational algebra	جبر رابطه‌اي
Relational assignment	انتساب رابطه‌اي
Relational calculus	حساب رابطه‌اي
Relational Data Structure (RDS)	ساختار داده اي ارتباطي
Relational Database Management System (RDBMS)	سيستم مديريت پايگاه‌داده هاي رابطه‌اي
Relationship type	نوع ارتباط
Reload	باز بارگذاري
Remote call	فراخوان از دور
Remote data access	دستيابي از دور به داده ها
Remote data access	دستيابي به داده هاي دوردست
Remote Data Access System (RDAS)	سيستم با امكان دستيابي به داده ها از دور
Renaming	دگرنامي
Replica	نسخه
Replication	نسخه سازي
Report generator	مولد گزارش
Repreting group	گروه تكرار

database system	جغرافيايي
Special purpose	تك منظوره
Specialization	تخصيص
Speedup	افزايش سرعت (كارايي)
Spread sheet	گستربرگ
Spurious	اطلاع جعلي
SQL-Session	شناسه كار با
AuthID SQL	محدوديت
State constraint	وضعيتي
Storage Definition Language (SDL)	زبان تعريف ذخيره سازي
Stored procedure	رويه ذخيره شده
Stored data	داده ذخيره شده
Stored data	ذخيره شده
Stored relation	رابطه ذخيره شده
Strong	قوي
Structural	ساختاري
Structural completeness	كمال ساختاري
Structural constraint	محدوديت ساختاري
Structural domain	ميدان ساختمند
Structured	ساختمند
Structured record	ركورد ساختمند
Structured Query Language (SQL)	زبان SQL
Sub entity type	زير نوع موجوديت
Sub query	پرسش فرعي
Sub table	زير جدول
Subtype	زير رده
Subtype	زير نوع
Summarize	گروه بندي
Super computer	كامپيوتر خيلي
Super Key (S. K)	ابركليد
Superentity type	زبر نوع
Supertable	زبر جدول
Supertype	زبر رده
Surrogate	كليد جايگزين نهان
System defined	تعريف شده در سيستم
System developer	سيستم ساز - پياده ساز

Search argument	نشانوند جستجو
Second Normal Form	صورت دوم نرمال
Security	ايمني
Selective setup	برپايي گزينشي
Self dependency	خود وابستگي
Self-relationship	ارتباط با خود
Self referencing	به خود رجوع كننده
Semantic	معنايي
Semantic unit of data	واحد معنايي داده
Semantic concept	مفهوم معنايي محدوديت
Semantic constraint	معنايي
Semantic Data Modeling (SDM)	مدلسازي معنايي داده
Semantic integrity constraint	محدوديت جامعيت معنايي
Semantic optimizer	بهينه ساز معنايي
Semantic rule	قواعد سمانتيك
Semi-join	نيم پيوند
Semi structured	نيم ساختمند
Serial schedule	طرح اجراي متوالي
Serializable	توالي پذير
Session manager	مدير تماسهاي اجرايي كاربران
Set type	نوع مجموعه
Set-oriented query	پرسش مجموعه
Setup	برپايي
Shared	اشتراكي
Shared entity	زير نوع مشترك
Side effect	عارضه جانبي
Single	ساده
Single user (Personal)	تك كاربري
Single-valued	تك مقداري
Site	مانه (سايت)
Smart sizing	اندازه تنظيم شونده
Snapshot	لحظه اي
Software facilities (tools)	تسهيلات نرم‌افزاري
Spatial and geographic	حيطه هاي فضايي و

integrity	
Transaction Processing Performance Council (TPC)	سازمان TPC خدمتگزار
Transaction server	تراكنشها
Transactions concurrency management	مديريت همروندي تراكنش
Transition constraint	محدوديت گذاري
Transitive	تراگذاري
Transitivity	تعدي
Transparency rules	قواعد نامرئي بودن
Trigger	رهانا (راه‌انداز)
Trivial	نا مهم (بديهي)
True proposition	گزاره درست
Tuple oriented	حساب تاپلي
Tuple variable	متغير تاپلي
Two Phase Locking (2PL)	تكنيك قفل گذاري دو مرحله اي
Two-tier	دو رديفي
Type compatible	سازگار نوع
Type constraint	محدوديت نوع

U

Undefined	تعريف نشده
Unfederated	نا فدرال
Unidirectional association	بستگي يك سويه
Union type	نوع اجتماع
Uniqueness	يكتايي
Universal	سور همگاني (عمومي)
Universal concept	مفهوم همگاني
Universal Database Management System (UDBMS)	سيستم مديريت پايگاه‌داده هاي همگاني
Universal Modeling Language (UML)	زبان عمومي مدل سازي
Universal relation	رابطه عمومي
Universe of Discourse	جهان مورد نظر
Universe of relations	مجموعه رابطه هاي ممكن
Unknown	ناشناخته
System specification	مشخصات سيستم
System variable	متغير سيستمي

T

Tabular Data Structure (TDS)	ساختار داده اي جدولي
Tabular system	سيستم جدولي پايگاه‌داده
Tabular Database	جدولي
Task	وظيفه
Temporal	زمانمند
Temporal Dependency	وابستگي زماني نمودار موجوديت-
Temporal ERD	ارتباط زمانمند
Temporal data	داده زمانمند پايگاه‌داده
Temporal database	زمانمند
Temporary table	جدول موقت
Temporary view	ديد موقت
Test data	داده تستي
Test strategy	استراتژي تست
Third Normal Form	صورت سوم نرمال
Three level architecture	معماري سه سطحي
Three-schema architecture	معماري سه شمايي
Three-tier	سه رديفي
Three-Valued Logic (3VL)	منطق سه ارزشي
Throughput	توان عملياتي
Tightly coupled	پيوند قوي
Time stamping	تكنيك زمانمهر
Tool developer	ابزار ساز
Top-down design method	روش بالا به پايين
Total participation	مشاركت الزامي (كامل)
Tow-Valued Logic (2VL)	منطق دو ارزشي
Trace facility	تسهيل براي ردگيري
Transactions Log Files (T. L. F)	فايلهاي ثبت تراكنشها
Transaction	تراكنش
Transaction	جامعيت تراكنش

Unload	خالي كردن
Unsafe expression	عبارت نامطمئن
Unstructured	نا ساختمند
Upgradeability	ارتقاء
Upsizing	افزايش‌اندازه
Use case diagram	نمودار مورد كاربرد
User Friendly Interface (U F I)	واسط كاربر پسند
User groups	واسط كاربري
User defined rule	قواعد كاربري
User usage monitoring	امكان نظارت بر عمليات كاربران
User defined	تعريف شده توسط كاربر
User Interface (UI)	واسط كاربر
Users' Ers integration	يكپارچه سازي نمودارهاي ER كاربران

V

Value -based security constraint	اعمال قواعد ايمني مبتني بر مقادير
Vector	بردار
Version	نگارش
Vertical decomposition	تجزيه عمودي
Vertical fragmentation	پارسازي عمودي
Very Large Database (VLDB)	پايگاه‌داده خيلي بزرگ
Very large information integrated system	سيستم اطلاعاتي بسيار بزرگ
Very large distributed information system	سيستم اطلاعات توزيع شده بسيار
View updatability	قابليت بهنگام سازش ديد
View Definition Language (VDL)	زبان تعريف ديد
View definition condition (s)	شرط تعريف داده
View derivation	اشتقاق ديد
View on view	ديد روي ديد
Virtual Child	فرزند مجازي
Virtual record type	نوع ركورد مجازي
Virtual relation	زابطه مجازي
Virtual table	جدول مجازي
Virtual attribute	صفت مجازي
Virtual entity	موجوديت مجازي
Virtual Parent Child Link (VPCL)	تكنيك VPCL
Virtual Parent (V P)	پدر مجازي
Visual-basic -script	وي-بي-اسكريپ
Voice driven	كاربر صوتي

W

Weak	ضعيف
Web DB	پايگاه‌داده وب
Well designed relation	رابطه خوش طرح
Well Formed Formula (WFF)	فرمول خوش ساخت
Work station	ايستگاه كار
Work station	ايستگاه كاري
Working table	جدول كاري
Workload	باركاري
World Wide Web (W W W)	شبكه جهاني اطلاع رساني
World Wide Web Database system (WWW-DB)	سيستم پايگاه‌داده در شبكه جهاني اطلاع رساني

واژه نامه

فارسی به انگلیسی

آ	
آرایه دیسکهای ارزان افزونه	Redundant Array of Inexpensive Disks (RAID)
آریتی	Arity
آزاد	Free
آنومالی	Anomaly
ابرکلید	Super Key (SK)
ابزار طراحی خودکار	Automatic Design Tool
ابزار ساز	Tool developer
ابزار طراحی	Design tool
احکام مرکب (تو در تو)	Nested statement
ادغام در وب	DBMS-web integration
ادغام شده	Embedded
ارتباط با خود	Self-relationship
ارتباط بازگشتی	Recursive relationship
ارتباط دهندگی	Informative
ارتباط شناسا	Identifing relationship
ارتقاء	Upgradibility
ارجاع	Reference
اصل موضوعه	Axiom

اساسی	Essential
استخراج داده ها	Data extraction
استراتژی تست	Test strategy
استراتژی دستیابی	Access strategy
استقراء	Induction
استقلال برنامه از داده	Program-data independence
استقلال داده ای	Data independence
استقلال داده ای فیزیکی	Physical Data Independence
استقلال داده ای منطقی	logical Data Independence
استنتاج	Deduction
اسلوب پیوسته (بر خط)	Online mode
اسلوب تعاملی	Interactive mode
اسلوب عملیاتی	Operational mode
اسلوب یکجا	Batch mode
اشتراک داده ها	Data sharing
اشتراکی	Shared
اشتقاق دید	View derivation
اشیاء جهان واقع	Real-world object
اطلاع جعلی (اضافی)	Spurious
اطلاع نهست	Missing

information	
Assertion	اظهار
GRANT	اعطاء
Declarative	اعلانی
Value -based security constraint	اعمال قواعد ایمنی مبتنی بر مقادیر
Back office server	افزارگان پشت صحنه
Augmentation	افزایش
Upsizing	افزایش‌اندازه
Left augmentation	افزایش چپ
Speedup	افزایش سرعت (کارایی)
Scale up	افزایش مقیاس کار - گسترش پذیری
Redundancy	افزونگی
Physical redundancy	افزونگی فیزیکی
Conceptual redundancy	افزونگی ادراکی
Data usage pattern	الگوی استفاده از داده ها
Privilege	امتیاز
Online Analytical Processing	امکان پردازش تحلیلی بر خط
Database administration support	امکان مدیریتی برای مدیر پایگاه
User usage monitoring	امکان نظارت بر عملیات کاربران
Abstractive	انتزاعی
Relation assignment	انتساب رابطه
Relational assignment	انتساب رابطه‌ای
Commutative	انتقال دهندگی
Smart sizing	اندازه تنظیم شونده
Lock granularity	اندازه واحد قفل پذیر
Interaction	اندرکنش (تعامل)
Interactive	اندرکنشی
Reflexivity	انعکاس

Development	ایجاد
CREATE view	ایجاد دید
Work station	ایستگاه کار
Work station	ایستگاه کاری
Security	ایمنی
Database Protection	حفاظت پایگاه‌داده ها

ب

Workload	بارکاری
Load	بارگذاری
Reload	باز بارگذاری
Data marts	بازار داده ها
Index rebuilding	بازسازی شاخص
Data bank	بانک داده ها
Setup	برپایی
Selective setup	برپایی گزینشی
On-line	برخط (پیوسته)
Vector	بردار
Application Program (AP)	برنامه کاربردی_ کاربردار
Closure	بستار
Qualified association	بستکی مقید
Association	بستگی
Reflexive association	بستگی انعکاسی
Bi-directional association	بستگی دو سویه
Unidirectional association	بستگی یک سویه
Bound	بسته
Deadlock	بن بست
BOYCE	بویس
Self referencing	به خود رجوع کننده
Logical design refinement	بهبود طراحی منطقی
Exploitation	بهره برداری
Inter related	بهم مرتبط
In place updating	بهنگام سازی در جا
Differed update	بهنگام سازی با تاخیر

بهنگام سازی برون از جا	Outplace updating
بهنگام سازی بلافاصله	Immediate update
بهنگام سازی پرسشها	Queries optimization
بهنگام سازی منتشر شونده	Propagating update
بهینه ساز پرسش	Query optimizer
بهینه ساز مبتنی بر قاعده	Rule based optimizer
بهینه ساز معنایی	Semantic optimizer
بهینه سازی پرسش	Query optimization
بی حشو	No additive
بی گمشدگی	Non-loss (Lossless)
بیشترین کارایی	Peak performance

پ

پارسازی (تقطیع) افقی	Horizontal fragmentation
پارسازی ترکیبی	Hybrid fragmentation
پارسازی عمودی	Vertical fragmentation
پایا	Persistent
پایگاه‌داده	Database
پایگاه‌داده بزرگ	Large Database
پایگاه‌داده محلی	Local database
پایگاه‌داده استنتاجی	Deductive database
پایگاه‌داده بی درنگ	Real-time DBMS
پایگاه‌داده توزیع شده (پراکنده)	Distributed Database (DDB)
پایگاه‌داده جدولی	Tabular Database
پایگاه‌داده خیلی بزرگ	Very Large Database
پایگاه‌داده زمانمند	Temporal database
پایگاه‌داده سلسله مراتبی	Hierarchical Database (HDB)
پایگاه‌داده مشتری-	Client / Server Database (C/S DB)
خدمتگزار	
پایگاه‌داده موبایل	Mobile DB
پایگاه‌داده میزبان	Host database
پایگاه‌داده وب	Web DB
پایگاه‌داده های حافظه اصلی	Main Memory Database
پایگاه‌داده های شخصی و همراه	Mobility and personal database
پایگاه‌داده های کنترل سیستم های ناهمگن موازی	Heterogeneous parallel systems control database
پایگاه متمرکز	Centralized database
پدر مجازی	Virtual Parent
پرتو	Project
پرتو همانی	Identity projection
پردازش موازی	Parallel processing
پردازشگر پرسش	Query processor
پردازشهای تحلیلی بر خط	On Line Analytic Processing
پرسش به کمک فرم	Query By Form
پرسش به کمک مثال	Query By Example (QBE)
پرسش بیرونی	Outer query
پرسش پیچیده	Complex query
پرسش تک سطحی	One level query
پرسش تو در تو	Nested query
پرسش درونی	Inner query
پرسش طیفی	Range query
پرسش فرعی	Subquery
پرسش مجموعه ای	Set-oriented query
پژوهش-توسعه	Research-Development (RD)
پسا	Back-end
پسا رابطه‌ای	Post relational
پشتیبان	Backup

Decision support	پشتیبانی تصمیم
Bridge	پل
Cover	پوش
Canonical cover	پوش کانونیک
Dynamic SQL	پویاSQL
Implementation	پیاده سازی
Complex	پیچیده (مختلط)
Pre-relational	پیش رابطه‌ای
Precompiled	پیش کامپایلر
Front-end	پیشا
Body	پیکر (بدنه)
Configuration	پیکربندی
Device configuration	پیکربندی رسانه
Join	پیوند
Link	پیوند
Greater then Join	پیوند به شرط بزرگتر
Equi-Join	پیوند به شرط تساوی
Not-Equi-Join	پیوند به شرط عدم تساوی
Less than Join	پیوند به شرط کوچکتر
Greater then or Equal	پیوند به شرط مساوی یا بزرگتر
Less than or Equal Join	پیوند به شرط مساوی یا کوچکتر
Connector	پیوند دهنده
Loosly coupled	پیوند سست
Cross join	پیوند ضرب
Natural Join (NJ)	پیوند طبیعی
Tightly coupled	پیوند قوی

ت

Dependency theory	تئوری وابستگی
Aggregate function	تابع جمعی یا گروهی
Libraey function	تابع کتابخانه ای
Application Program conversion	تبدیل برنامه های کاربردی
Data conversion and data transfer	تبدیل و انتقال داده ها
Commit	تثبیت
Decomposition	تجزیه
Horizontal decomposition	تجزیه افقی
Lost decomposition	تجزیه با حذف
FD' s preserving decomposition	تجزیه حافظ وابستگیها
Vertical decomposition	تجزیه عمودی
Atomicity	تجزیه ناپذیری
Atomic	تجزیه نشدنی
Aggregation	تجمع
Restrict	تحدید-گزینش
Cost-benefit analysis	تحلیل برآورد هزینه / مزایا
Functional analysis	تحلیل عملکردی
Fault tolerance	تحمل خرابی
Specialization	تخصیص
Continuous operation	تداوم عملیات
Transaction	تراکنش
Transitive	تراگذاری
Composition	ترکیب
Recovery	ترمیم
Cascade	تسلسلی
Trace facility	تسهیل برای ردگیری
Software facilities (tools)	تسهیلات نرم‌افزاری(ابزارها)
Decision making	تصمیم گیری
Interoperability	تعامل
Transitivity	تعدی
Data Definition	تعریف داده
User defined	تعریف شده توسط

کاربر	
تعریف شده در سیستم	System defined
تعریف شما	Schema definition
تعریف نشده	Undefined
تعمیم	Generalization
تعویقی (مشروط)	Restricted
تغییر پرسش	Query modification
تغییر درخواست	Request modification
تغییر شما	Schema evolution
تفسیر	Interpretation
تفسیر جهان بسته	Closed World interpretation
تقسیم	Divide
تک خدمتگزار	Mono- server
تک عملوندی	Monadic
تک کاربری	Single user (Personal)
تک مقداری	Single-valued
تک منظوره	Special purpose
تکنولوژی اطلاعات	Information Technology (IT)
تکنیک VPCL	Virtual Parent Child Link
تکنیک تایید (تصدیق)	Certification
تکنیک چند نسخه سازی	Multiversioning
تکنیک زمانمهر	Timestamping
تکنیک قفل گذاری	Locking technique
تکنیک قفل گذاری دو مرحله ای	Two Phase Locking (2PL)
تکنیک مدلسازی شیئ	Object Modeling Technique
تکه	Piece
تمام کلید	ALL-Key
تنظیم پایگاه‌داده ها	Database tuning
تنگنا	Buttleneck
توالی پذیر	Serializable
توالی منطقی	Logical sequence
توان عملیاتی	Throughput
توپولوژی شبکه	Network topology
توده	Bags
توزیع شده	Distributed
توصیفی	Descriptive

ج

جابجایی پذیری	Portability
جامعیت	Integrity
جامعیت تراکنش	Transaction integrity
جاوا اسکریپت	Java-script
جایدهی داده	Data placement
جبر رابطه‌ای	Relational algebra
جدایی (انفراد)	Isolation
جدول جستجو	Lookup table
جدول کاری	Working table
جدول ماندگار	Persistent (Permanent) table
جدول مبنا(پایه)	Base Table
جدول مجازی	Virtual table
جدول موقت	Temporary table
جزایر اطلاعاتی	Islands of information
جزیی	Partial
جزیی است از...	IS-A PART-OF
جنبه های صفت	Attribute features
جهان بسته	Closed World
جهان تازه (شبکه جهانی اطلاع رسانی)	World Wide Web (W W W)
جهان مورد نظر	Universe of Discourse
جهان واقع	Real world

چ

چارت گانت	Gantt chart
چمبرلین	CHEMBERLIN

چن	CHEN
چند به چند	Many to many
چند پایگاهی	Multi database
چند پرسشی	Multi query
چند حلقه ای	Multi ring
چند خدمتگزار	Multi-server
چند رسانه ای	Multi- media
چند کاربری	Multi user
چند مجموعه	Multi set
چند مشتری- چند خدمتگزار	Multi- Client / Multi-Server(MC / MS)
چند مشتری- یک خدمتگزار	Multi- Client / Server(MC /S)
چند مقداری	Multi valued
چندی	Multiplicity
چندی نما	Quantifier
ح	
حافظه صفات	Attribute preserving
حافظه نهان	Cache memory
حالت رابطه	Relation state
حذف دید	DROP view
حساب تاپلی	Tuple oriented
حساب رابطه‌ای	Relational calculus
حساب میدانی	Domain oriented
حفاظت	Protection
حقوق دستیابی	Access right
حلقه ارجاع	Reference cycle
حیطه های فضایی و جغرافیایی	Spatial and geographic database system
خ	
خارجی / خارجی	External / External
خالی کردن	Unload
خدمات پایگاه‌داده های موبایل	Mobile database server
خدمتگزار میزبان	Host server
خدمتگزار اخبار	News server
خدمتگزار پایگاه‌داده ها	Database server
خدمتگزار پرسش	Query server
خدمتگزار پیام	Message server
خدمتگزار تراکنشها	Transaction server
خدمتگزار چاپ	Printer server
خدمتگزار فایل	File server
خدمتگزار مدارک و اسناد	Document server
خرد جهان واقع	Micro real world
خود مختاری محلی	Location anatomy
خود وابستگی	Self dependency
خوشه واری	Clustering
د	
داده پویا	Dynamic data
داده تستی	Test data
داده داری- داده پردازی	Data keeping-data processing
داده ذخیره شده	Stored data
داده رسانی	Data Communication
داده زمانمند	Temporal data
داده شناسانه	Data logical
داده نهان	Hidden data
داده نهست	Missing data
دارای کلید	Key preserving
دام حلقه ای	Loop trap
دام شکاف	Chasm trap
دام یک-چندی	Fan trap
دامهای پیوندی	Connection traps
دترمینان	Determinant
درجه	Degree
دسترسی(بهتر) به داده	Data accessibility
دستور عملیات در داده	Data

دستورات تعریف داده ها	Data Definition Language
دستورات عملیات روی داده ها	Data Manipulation Language
دستورات کنترل داده ها	Data Control Language
دستوری	Prescriptive
دسته	Category
دسته بندی	Categorization
دستیابی از دور به داده ها	Remote data access
دستیابی به داده های دوردست	Remote data access
دستیابی بهتر	Accessibility
دستیابی چند کاربردی	Multiple-user access
دفتر کار همراه	Mobile office
دقت	Accuracy
دگرنامی	Renaming
دو رابطه	2-Decomposable
دو ردیفی	Two-tier
دو ساختاره	Bi- structure
دو عملوندی	Dyadic
دیت	C.J.DATE
دیتالوگ	Data logical
دید ادراکی	Conceptual view
دید به عینیت در آمده (ساخته شده)	Materialized view
دید خارجی	External view
دید داخلی	Internal view
دید روی دید	View on view
دید موقت	Temporary view
دیدگاه پایگاهی	Database perspective
دیدگاه کاربردی	Application perspective
دیکشنری غیر فعال	Passive dictionary
دیکشنری فعال	Active dictionary

ذ

ذاتی	Intrinsic
ذخیره شده	Stored data
ذخیره و بازیابی اطلاعات	Information Storage and Retrieval (ISR)

ر

رابطه	Intension
رابطه بینابینی	Intermediate relation
رابطه تودر تو	Nested relation
رابطه خوش طرح	Well designed relation
رابطه ذخیره شده	Stored relation
رابطه عبارتی	Expression relation
رابطه عمومی	Universal relation
رابطه غیر نرمال	Not Normal relation (NNR: N2R)
رابطه مبنا	Base relation
رابطه مشتق	Derived relation
رابطه نامدار	Named relation
رابطه نتیجه	Result relation
رابطه نرمال	Normal relation
رجوع کننده	Referencing
رده	Class
رده بندی	Classification
رشد پایگاه‌داده	Database growth
رشد و کاهش	Grow and shrink
رفتار شیئ	Object behavior
رفتاری	Behavioral
رکورد خطی مسطح	Flat Linear record
رکورد ساختگی	Dummy record
رکورد ساختمند	Structured record

Enhanced (Extended) ER	روش EER
Top-down design method	روش بالا به پایین
Mixed design method	روش ترکیبی
Relational synthesis	روش سنتز رابطه‌ای
Event journaling	رویداد نگاری
Procedure	رویه
Procedural	رویه ای
Responding procedure	رویه پاسخگو
Stored procedure	رویه ذخیره شده
Documented procedure	رویه مستند
Trigger	رهانا (راه‌انداز)
ز	
Virtual relation	زابطه مجازی
ZANIOLO	زانیولو
Programming Language (PL)	زبان برنامه سازی
Structured Query Language (SQL)	زبان SQL
Logic programming Language	زبان برنامه سازی منطق
Database Language	زبان پایگاهی
Object Oriented Database Language	زبان پایگاهی شیئ گرا
Query Language	زبان پرسش
Nested relational Query language	زبان پرسش با رابطه غیر نرمال
Object Query Language(OQL)	زبان پرسش شیئ گرا
View Definition Language	زبان تعریف دید)
Storage Definition Language	زبان تعریف ذخیره سازی
Constraint specification language	زبان توصیف محدودیت
Data Sub Language	زبان داده ای فرعی
Network Database Language	زبان سیستم شبکه ای
Object Oriented Language(OOL)	زبان شیئ گرا
Universal Modeling Language (UML)	زبان عمومی مدل سازی
Host Language	زبان میزبان
4GL	زبان نسل چهارم
Super table	زبر جدول
Super type	زبر رده
Super entity type	زبر نوع
peak time	زمان اوج
Response time	زمان پاسخدهی
Temporal	زمانمند
Sub table	زیر جدول
Subtype	زیر رده
Subtype	زیر نوع
Shared entity	زیر نوع مشترک
Sub entity type	زیر نوع موجودیت
س	
Abstractive construct	ساخت انتزاعی
Logical construct	ساخت منطقی
Data Structure	ساختار داده ای
Tabular Data Structure (TDS)	ساختار داده ای جدولی
Relational Data Structure	ساختار داده ای ارتباطی
Box data structure	ساختار داده ای جعبه ای
Hierarchical Data Structure	ساختار داده ای سلسله مراتبی
Network Data Structure (NDS)	ساختار داده ای شبکه ای

ساختار داده ای لیستهای وارون	Inverted list data structure
ساختار داده ای هایپرگرافی	Hyper graph Database Structure
ساختاری	Structural
ساختگر	Constructor
ساختمند	Structured
ساختن داده	Data materialization
ساختن غیر مستقیم	Indirect materialization
ساختن مستقیم	Direct materialization
ساده	Single
سازگار نوع	Type compatible
سازگاری	Consistency
سازمان TPC	Transaction Processing Performance Council (TPC)
سازند	Component
سر آیند	Heading
سرمایه	Asset
سطح ادراکی (مفهومی)	Conceptual level
سطح خارجی	External level
سطح داخلی	Internal level
سنتی	Conventional
سند ارتباط	Relation predicate
سور وجودی	Existential
سور همگانی (عمومی)	Universal
سه ردیفی	Three-tier
سیستم داده کاوی و کشف شناخت	Data mining and knowledge discovery system
سیستم کوداسیل	Conference On Data System Language (CODASYL)
سیستم R	R-system

سیستم اطلاعات اجرایی	Executive Information System (EIS)
سیستم اطلاعات توزیع شده بسیار	Very large distributed information system
سیستم اطلاعاتی	Information System (IS)
سیستم اطلاعاتی بسیار بزرگ	Very large information integrated system
سیستم انبارش داده ها	Data Warehousing
سیستم با امکان دستیابی به داده ها از دور	Remote Data Access System
سیستم پایگاه‌داده ای با معماری	Mobile database system
سیستم پایگاه‌داده در شبکه جهانی اطلاع رسانی	World Wide Web Database system (WWW-DB)
سیستم پایگاههای همراه	Mobile database system
سیستم پردازش تحلیلی برخط	Multi-Dimensional OLAR
سیستم پشتیبان تصمیم	Decision Support System
سیستم جامعه (یکپارچه)	Integrated system
سیستم جدولی	Tabular system
سیستم داده کاوی چند رسانه ای	Multimedia-data mining
سیستم رابطه‌ای تام	Fully relational system
سیستم رابطه‌ای حداقل	Minimal relational system
سیستم رابطه‌ای کامل	Complete relational system
سیستم ژولای داده کاوی و کشف دانش	Fuzzy data mining and knowledge discovery system

سیستم ساز (پیاده ساز)	System developer
سیستم شناختی	Knowledge-based system
سیستم فایلینگ	Filing System
سیستم فعال	Active DBMS
سیستم فعال مدیریت پایگاه‌داده	Active Database Management System
سیستم کاربردی	Application system
سیستم متعارف	Conventional system
سیستم مدیریت پایگاه‌داده ها	Database Management System (DBMS)
سیستم مدیریت پایگاه‌داده های بی درنگ	Real-time database management system
سیستم مدیریت پایگاه‌داده های رابطه‌ای	Relational Database Management System
سیستم مدیریت پایگاه‌داده های شیئ – رابطه‌ای	Object - Relational Database Management System
سیستم مدیریت پایگاه‌داده های شیئ گرا	Object Oriented Database Management System
سیستم مدیریت پایگاه‌داده های همگانی (عمومی)	Universal Database Management System
سیستم مدیریت پایگاه شناخت	Knowledge Base Management System (KBMS)
سیستم مدیریت چند پایگاهی	Multi database management system
سیستم مدیریت داده ها	Data Management System (DMS)
سیستم هوشمند	Intelligent system

ش

شامل است	CONTAINS
شبکه	Network
شبه تعدی	Pseudo transitivity
شبه زبان طبیعی	Natural language like
شرط تعریف داده	View definition condition (s)
شرط عضویت	Membership condition
شرکت کننده	Participant
شما	Schema
شمای ادراکی	Conceptual schema
شمای ادراکی جامع	Global conceptual schema
شمای ادراکی محلی	Local conceptual schema
شمای خارجی	External schema
شمای داخلی	Internal schema
شمای رابطه	Relation schema
شمول	Containment
شناسای	Identifying
شناسه برای تعریف ماجول	Module AuthID
شناسه تعریف شما	Schema AuthID
شناسه جاری	Current AuthID
شناسه کار با SQL	SQL-Session AuthID
شناسه مجاز شماری	Authorization Identifier
شناسه موجودیت	Entity Identifier (EID)
شهودی	Intuitive
شیئ	Object
شیئ–داده	Data-object
شیئ های داده ای فعال	Active Data Objects (ADO)
شیوه دستیابی	Access Method

ص

صحت	Correcthness
صدور داده ها	Data export
صفت	Attribute
صفت توصیفی	Descriptive attribute
صفت عمده	Prime attribute
صفت مجازی	Virtual attribute
صفت محاسبه شده	Calculated attribute
صفت پیوند	Join attribute
صفت معرف	Defining attribute
صفحه	Page
صورت پیشوندی	Prenex form
صورت نرمال بایس-کاد	Boyce-Codd Normal Form
صورت نرمال بی افزونگی	Redundency Free Normal Form
صورت نرمال تحدید-اجتماع	Restriction - Union Normal Form (RUNF)
صورت نرمال کاملا کلید	Key-Complete Normal Form
صورت نرمال میدان-کلیدی	Domain - Key Normal Form
صورت نرمال وابستگی شمول	Inclusion Dependency Normal Form
صورت پنجم نرمال	Fifth Normal Form
صورت چهارم نرمال	Fourth Normal Form
صورت دوم نرمال	Second Normal Form
صورت سوم نرمال	Third Normal Form
صورت نخست نرمال	First Normal Form
صوری	Formal
ض	
ضد نیم پیوند	Anti semi join
ضرب کاتزین گسترش	Extended Cartesian product
یافته	
ضریب گزینش	Join selectivity factor
عملکرد پیوند	
ضعیف	Weak
ط	
طراحی ادراکی	Conceptual design
طراحی با تحلیل	Design by analysis
طراحی/تولید به کمک کامپیوتر	CAD/CAM
طراحی بهتر و واضح	Clean design
طراحی فیزیکی	Physical design
طراحی متعامد	Orthogonal design
طراحی منطقی	Logical design
طراحی نا متعامد	Non orthogonal design
طرح اجرا	Execution plan
طرح اجرای متوالی	Serial schedule
طرح اجرای همروند	Concurrent schedule
طرد	Abort
طول پیش نهاده	Default length
ع	
عارضه جانبی	Side effect
عبارت جبر رابطه‌ای	Relational expression
عبارت مطمئن	Safe expression
عبارت نامطمئن	Unsafe expression
عدم اقدام	No action
عدم تقارن	Asymmetry
عضو	Member
عملوند	Operand
عملی	Action
عملیاتی	Functionally
عملیاتی	Manipulative
عیب (نقص)	Fail (Failure)
غ	

غیر صوری	Informal
غیر قابل اعمال	Inapplicable
ف	
فاگین	FAGIN
فایلهای ثبت تراکنشها	Transactions Log Files
فدرال	Federated
فرا اجتماع	Outer union
فرا پیوند چپ	Left outer join
فرا پیوند راست	Right outer join
فرا پیوند کامل	Full outer join
فراخوان از دور	Remote call
فراکرد (کلاز)	Clause
فرزند مجازی	Virtual Child
فرمول خوش ساخت	Well Formed Formula (WFF)
فروشندگان سیستمها	DBMS vendors
فزونکاری (بیشکاری)	Overhead
فشرده سازی داده ها	Data compression
فضای اطلاعاتی	Information space
فضای پایگاه‌داده	Database space
فقره داده نامدار	Named data items
فقط ریشه	Root only
ق	
قابلیت بهنگام سازش دید	View updatability
قاعده جامعیت موجودیتی	Entity integrity rule
قاعده جامعیتی ارجاعی	Referential integrity rule
قاعده (محدودیت) جامعیت	Integrity rule (constraint)
قاعده پنج دقیقه	Five- minute rule
قاعده فعال	Active rule
قاعده یک دقیقه	One- minute rule
قضیه هیث	HEATH theorem
قفل گذاری	Locking
قفل گذاری و تولید	Locking/logging
فایلهای ثبت	Rules
قواعد	Inference rules
قواعد استنتاج	Business rule
قواعد تجاری	Production rules
قواعد تولید	Access rules
قواعد دستیابی	Event- condition -action rules
قواعد رویداد–شرط–اقدام	Semantic rule
قواعد سمانتیک	Business rules
قواعد فعالیتها	User defined rule
قواعد کاربری	Transparency rules
قواعد نامرئی بودن	Strong
قوی	Analogue
قیاسی	Qualifier
قید	
ک	
کاتالوگ	Catalogue
کاتالوگ جامع	Global System Catalog (GSC)
کاد	CODD
کار	Job
کارایی	Performance
کاربر با زبان طبیعی	Natural language driven
کاربر پایانی	End user
کاربر صوتی	Voice driven
کاربر گرافیکی	Graphic driven
کاربرد بلادرنگ	Real time application
کاربرد ساز (بهره بردار)	Application developer
کاربردی سرتاسری	Global application
کاربردی محلی	Local application

کاربریار	Help
کاردینالیتی	Cardinality
کامپیوتر مشتری	Client machine
کامپیوتر بزرگ	Main computer
کامپیوتر خیلی بزرگ	Super computer
کامپیوتر متوسط	Mini computer
کاهش‌اندازه	Downsizing
کاهش درجه (سطح) نرمالیتی	De normalization
کاهش ناپذیر	Irreducible
کاهش ناپذیری	Irreduciblity
کلید اصلی	Primary Key
کلید جایگزین نهان	Surrogate
کلید جزیی	Partial Key
کلید خارجی	Foreign Key
کلید دیگر	Alternate Key
کلید کاندید	Candidate Key
کمال برنامه سازی	Programming completeness
کمال رابطه‌ای	Relationally completeness
کمال ساختاری	Structural completeness
کمال محاسباتی	Computational completeness
کنترل داده	Data Control
کنترل متمرکز	Centralized control
کنترولر بر خط	Online controller
کنشگر	Actor
کوچاندن داده	Data migration
کهینه	Minimal
کیفیت پایگاه‌داده ها	Database quality
گ	
گدار	Gateway
گراف ارجاع	Reference graph
گراف جهت دار	Oriented graph
گردایه	Collection

گروه بندی	Summarize
گروه تکرار شونده	Repeating group
گزاره	Proposition
گزاره درست	True proposition
گزیدار (گزینه)	Option
گستربرگ	Spread sheet
گسترده	Extension
گسترش پذیری	Scalability
گسترش پذیری نوع داده ای	Data type extensibility
گشت زنی (گذار گری)	Browsing
ل	
لایه	Layer
لحظه ای	Snapshot
لغو	REVOKE
م	
ماجولهای ذخیره شده ماندگار	Persistent Stored Modules (PSM)
ماشین انتزاعی	Abstract machine
ماشین بدون دیسک	Diskless machine
ماشین پایگاه‌داده ها	Database Machine
ماشین خدمتگزار پایگاهی پسا	Backend DB server
مالک	Owner
مانایی (دوام)	Durability
مانه (سایت)	Site
ماهیت	Nature
مبنا	Base attribute
متا بایت	Meta byte
متا داده	Meta data
متا قواعد	Meta rules
متا محدودیت	Meta-constraint
متغیر تاپلی	Tuple variable
متغیر رابطه‌ای	Relation variable(Relvar)

System variable	متغیر سیستمی
Range variable	متغیر طیفی
Domain variable	متغیر میدانی
Centralized	متمرکز
Problem context	متن مسئله
Harmonic mean	متوسط هارمونیک
Authorization	مجاز شماری
Integrated	مجتمع (یکپارچه)
Disjoint	مجزا
Entity types set	مجموعه انواع موجودیتها
Queries set	مجموعه پرسش
Multimember set	مجموعه چند عضوی
Universe of relations	مجموعه رابطه های ممکن
Attributes set	مجموعه صفات
Instances set	مجموعه نمونه
Constraint	محدودیت
Semantic integrity constraint	محدودیت جامعیت معنایی
Integrity constraint	محدودیت جامعیتی
Relation constraint	محدودیت رابطه‌ای
Structural constraint	محدودیت ساختاری
Attribute constraint	محدودیت صفتی
Transition constraint	محدودیت گذاری
Semantic constraint	محدودیت معنایی
Domain constraint	محدودیت میدانی
Check constraints	محدودیت ناظر به ستون
Type constraint	محدودیت نوع
State constraint	محدودیت وضعیتی
3D- constraint	محدودیت D۳
Database constraint	محدودیت پایگاهی
Privacy	محرمانگی
Benchmarking	محک زنی
Local	محلی
Operational envirnoment	محیط عملیاتی
Order-entry environment	محیط ورود سفارش
Data Model (DM)	مدل داده ای
Object Data Model (ODM)	مدل داده ای شیئ گرا
CODASYL data model	مدل داده ای کوداسیل
Object-Relational Data Model	مدل رابطه‌ای شیئ گرا
Object modeling	مدلسازی شیئی
Semantic Data Modeling	مدلسازی معنایی داده
Data communication manager	مدیر انتقال داده ها
DC manager	مدیر انتقال داده ها
Database Administrator	مدیر پایگاه‌داده ها
Session manager	مدیر تماسهای اجرایی کاربران
Data Administrator	مدیر داده
Data Manager	مدیر داده ها
Driver manager	مدیر درایور
Request manager	مدیر درخواستها
Run time manager	مدیر زمان اجرا
Catalog manager	مدیر کاتالوگ
Database environment management	مدیریت محیط پایگاه‌داده ها
Transactions concurrency management	مدیریت همروندی تراکنش
Referenced	مرجع
Composite	مرکب
Independent (Stand alone)	مستقل (خودکفا)
Documentation	مستند سازی

مسطح	Flat
مسند	Predicate
مسند گزینش	Restriction predicate
مسندات	Predicate calculus
مسیرگراف	Reference path
مشارکت الزامی (کامل)	Total participation
مشارکت غیرالزامی (ناکامل)	Partial participation
مشتری/ خدمتگزار	Client/Server
مشتق	Derived
مشخصات سیستم	System specification
مشی پایگاهی	Database approach
مشی ناپایگاهی	Non database approach
معلق	Dangling
معماری چند پایگاهی	Multi Database System (MDBS)
معماری حول شبکه	Network centric
معماری حول کامپیوتر بزرگ	Main frame centric
معماری سه سطحی	Three level architecture
معماری سه شمایی	Three-schema architecture
معماری نامتمرکز	Decentralized architecture
معنا	Meaning
معنایی	Semantic
مفهوم معنایی	Semantic concept
مفهوم همگانی	Universal concept
مقدار پیش نهاده	Default value
مقدار رابطه‌ای	Relation value (Relval)
مکان نما	Cursor
ممیزه	Discriminator
مناسب سازی‌اندام	Rightsizing
منبع داده ای	Data source

منطق دو ارزشی	Tow-Valued Logic (2VL)
منطق سه ارزشی	Three-Valued Logic (3VL)
موتور	Engine
موجودیت–ارتباط (ER)	Entity relationship
موجودیت شناسا	Identifing entity
موجودیت مجازی	Virtual entity
موردی	Ad hoc (Unplanned)
مولد فرم	Form generator
مولد گزارش	Report generator
مولد منو	Menu generator
مولد نوع رابطه	Relation type generator
مهم (مطرح)	Nontrivial
مهندسی دوباره	Reengineering
مهندسی نیازها	Requirements Engineering
میان افزار	Middleware
میانجی	Mediator
میدان	Domain
میدان اصلی	Primary domain
میدان ساختمند	Structural domain
میدان کاندید	Candidate domain
ن	
نا رویه ای (ناروشمند)	Non procedural
نا ساختمند	Unstructured
نا فدرال	Unfederated
نا مهم (بدیهی)	Trivial
ناسازگاری داده	Data inconsistency
ناشناخته	Unknown
ناظر زمان اجرا	Run time supervisor
نامگذاری داده ها	Data naming
ناوش (غواصی)	Navigation
ناوش خودکار	Automatic navigation

ناهمگن	Heterogenous
نسبت کاردینالیتی	Cardinality ratio
نسبت هزینه / کارایی	Price / performance ratio
نسخه	Replica
نسخه برداری	Dumping
نسخه سازی	Replication
نشان نما (اشاره گر)	Pointer
نشانوند جستجو	Search argument
نظارت	Monitoring
نظم-آراستار	Ordering
نظم مکانی	Positional ordering
نقش	Role
نقطه وارسی	Checkpointing
نگارش	Version
نگاشت	Mapping
نگاشت ادراکی / داخلی	Conceptual / Internal mapping
نگاشت خارجی / ادراکی	External / Conceptual mapping
نمایش	Presentation
نمود	Occurrence
نمودار ER	ER diagram
نمودار ارجاع	Reference diagram
نمودار با چمن	Bachman diagram
نمودار پیاده سازی	Implementation diagram
نمودار چن	CHEN notation
نمودار رده	Class diagram
نمودار روند داده	Data Flow Diagram (DFD)
نمودار روند رویداد	Event Flow Diagram (EFD)
نمودار فعالیت	Activity diagram
نمودار موجودیت-ارتباط	Temporal ERD
زمانمند	
نمودار مورد کاربرد	Use case diagram
نمودار وابستگیهای تابعی	FD ' s diagram
نمونه	Instance
نمونه سازی گسترشی	Evolutionary Prototyping
نمونه سازی نیازی	Requirements Prototyping
نمونه‌نخست-نمونه‌سازی	Prototype
نوع ابتدایی	Elementary type
نوع اجتماع	Union type
نوع ارتباط	Relationship type
نوع پیوند پدر-فرزندی	Parent-Child Link type (PCL)
نوع داده ای	Data type
نوع داده ای انتزاعی	Abstract Data Type (ADT)
نوع داده ای پیچیده	Complex Data Type (CDT)
نوع داده ای گسترش یافته	Extended data type
نوع داده ای مبنایی	Basic data type
نوع داده مجرد	Abstract Data Type (ADT)
نوع رکورد	Record type
نوع رکورد مجازی	Virtual record type
نوع شیئ	Object Type
نوع مجموعه	Set type
نوع موجودیت	Entity type
نهان نگاری داده	Data encryption
نیم پیوند	Semi-join
نیم ساختمند	Semi structured
و	
وابستگی برابری	Equational Dependency
وابستگی پیوندی	Join Dependency
وابستگی تابعی	Functional Dependency

وابستگی تابعی تام (کامل)	Fully Functional Dependency
وابستگی چند مقداری	Multivalued Dependency
وابستگی چند مقداری ادغام شده	Embedded Multi Valued Dependency
وابستگی زمانی	Temporal Dependency
وابستگی شمول	Inclusion Dependency
وابستگی شمول مبتنی بر کلید	Key-based inclusion dependency
وابستگی وجود	Existance dapendency
واحد	Module
واحد تولید شما	Schema generator
واحد قفل شدنی	Lock granular
واحد معنایی داده	Semantic unit of data
واحد منطقی کار	Logical unit of work
وارد کردن داده	Data entry
وارسی با تاخیر	Deferred checking
وارسی بلافاصله	Immediate checking
واسط برنامه سازی کاربرد	Application Programming Interface (API)
واسط پانل	Panel interface
واسط در سطح فراخوان	Call Level Interface
واسط زبان فرمان	Command Language Interface
واسط کاربر	User Interface
واسط کاربر پسند	User Friendly Interface
واسط کاربری	User groups
واسط ورود / صدور	Import / Export interface
واقعی	Real
واقعیات	Facts
وراثت (توارث) چند	Multiple

گانه	inheritance
وراثت صفت	Attribute inheritance
ورودی داده ها	Data import
وضعیت پایگاه‌داده	Database state
وظیفه	Task
وی–بی–اسکریپ	Visual-basic-script
ویرانگر	Destructor
هدایت شده با پارامتر	Parameter driven
هدایت شده با فرم	Form driven
هدایت شده با فرمان	Command driven
هدایت شده با منو	Menu driven
هدایت شده با نقشه	Map driven
هزینه به ازاء TPS	Price per TPS
هست یک...	IS-A
همپوشا	Overlapping
همرسانش (ارتباط)	Communication
همروندی چند نسخه ای	Multiversion concurrency
همزمان	Concurrent
همطراز	Peer-to-peer
همگن	Homogenous
همه منظوره	General purpose
همیشگی	Permanent (planned)
هیچمقدار	Nullvalue
هیچمقدار گذاری	Nullifing
یک واقعیت–یک رابطه	One fact-one relation
یک به چند	One to many
یک به یک	One to one
یک مشتری–چندخدمتگزار	Client / Multi-Server (C/MS)
یکپارچه سازی نمودارهای ER کاربران	Users' Ers integration

Uniqueness	یکتایی
Batch	یکجا
General unification	یکسان عمومی

منابع و ماخذ

1	The Relational Model for Database Management , 2nd Edition	Addison-Wesley
	CODD.E.F	1990
2	Mastering SQL , 1st Edition	Sybex
	Gruber, Martin	2000
3	A Guide to the SQL Standard , 4th Edition	Addison-Wesley
	DATE, C.J	1997
4	ORACLE PL/SQL , 1st Edition	Osborne
	JOSEPH , C.Trezzo	1999
5	Guide to Client/Server Databases , 2nd Edition	Ziff-David Press
	SALEMI , joe	1996
6	Inside SQL server 2000, 1st Edition	Microsoft Press
	Delaney , Kalen	2001
7	Database System Concepts , 4th Edition	MC Graw-Hill
	SILBERSCHATZ , Henry	2002
8	مفاهیم بنیادی پایگاه‌داده ها , ویراست جدید	جلوه
	روحانی رانکوهی ، سید محمد تقی	۱۳۸۳
9	پایگاه‌داده ها , چاپ ششم	گسترش علوم پایه
	مقسمی ، حمیدرضا	۱۳۸۳
10	ذخیره و بازیابی اطلاعات: مقدمه ای بر سیستم و ساختار فایلها , چاپ نهم	جلوه
	روحانی رانکوهی ، سید محمد تقی	۱۳۷۹