فصل اول

تاریخچه و مفاهیم اولیه

نسلهاي ذخيره و بازيابي اطلاعات

۱ـ نسل اول نسل فايلهاي ساده ترتيبي

در این نسل رسانه خارجی معمولاً نوار بوده است. این نسل را می‌توان نسل بی‌نرم‌افزار واسط نیز نامید. مشخصات کلی این نسل عبارتند از :

۱ـ ساختار فایلها ترتیبی است.

۲ـ ساختار فیزیکی همان ساختار منطقی فایل است.

۳ـ تنها روش پردازش فایلها، پردازش یکجا یا دسته‌ای (Batch Processing) است.

۴ـ نرم‌افزار تنها عملیات ورودی/ خروجی را انجام می‌دهد. نرم‌افزار واسطی برای مدیریت پردازش فایلها وجود ندارد.

۵ـ طراحی ساختار فیزیکی فایلها هم، برعهده کاربر است.

۶ـ هرگونه تغییر در ساختار داده‌ها و یا رسانه‌های ذخیره‌سازی سبب بروز تغییر در برنامه و بازنویسی و کامپایل آن می‌شود.

۷ـ داده‌ها برای کاربرد خاصی طراحی و سازماندهی می‌شوند.

۸ـ اشتراک داده‌ها (Data Sharing) مطرح نیست.

۹ـ تکرار در ذخیره‌سازی داده‌ها در بالاترین حد است.

۱۰ـ برای انجام عملیات بهنگام‌سازی، الزاماً فایل دیگری ایجاد و تغییرات را در آن وارد کرده، نسخه قدیمی را به عنوان «فایل پدر» نگهداری می‌کنند و به این دلیل نسخه‌های متعددی از یک فایل نگهداری می‌شوند.

۲ـ نسل دوم نسل شیوه‌های دستیابی

این نسل را باید نسل شیوه‌های دستیابی (Access Methods) نامید. مهمترین ویژگی این نسل را باید پیدایش نرم‌افزارهای موسوم به «شیوه‌های دستیابی» و همچنین ایجاد رسانه‌های با دستیابی مستقیم (یعنی دیسک) دانست.

نرم‌افزار شیوه دستیابی، نرم‌افزاری است که به جنبه‌های فیزیکی محیط ذخیره‌سازی و عملیات در این محیط می‌پردازد. به نحوی که دیگر برنامه کاربر نیازی به پرداختن به این جنبه‌ها را ندارد.

مشخصات این نسل عبارتند از :

۱ـ نرم‌افزار واسط برای ایجاد فایلها با ساختارهای گوناگون بین برنامه‌های کاربردی و محیط ذخیره‌سازی وجود دارد.

۲ـ امکان دستیابی ترتیبی و مستقیم به رکوردها (نه فیلدها) وجود دارد.

۳ـ پردازش در محیط‌های بلادرنگ (Real Time) و برخط (On – Line) بسته به نوع سیستم عامل می‌تواند انجام شود.

۴ـ ساختار فیزیکی و ساختار منطقی فایلها از یکدیگر جدا هستند ولی نه تا حدی که برنامه‌های کاربردی از محیط فیزیکی ذخیره‌سازی مستقل شوند.

۵ـ تغییر در رسانه‌های ذخیره‌سازی بر روی برنامه‌های کاربردی تاثیر چندان ندارد.

۶ـ هنوز امکان بازیابی براساس چندین کلید وجود ندارد.

۷ـ ایمنی و حفاظت داده‌ها مطرح بوده ولی روشهای تامین امنیت و حفاظت ابتدایی هستند.

۸ـ داده‌ها همچنان برای کاربردهای خاص طراحی و ذخیره‌سازی می‌شوند.

۹ـ تکرار ذخیره‌سازی هنوز در حد نسبتاً بالایی وجود دارد.

۱۰ـ برای پیاده‌سازی فایل با ارتباط خاصی بین انواع رکوردها (مثلاً ارتباط سلسله مراتبی) خود برنامه‌ساز باید ارتباطات را در برنامه‌اش بسازد.

۳ـ نسل سوم سیستم مدیریت داده‌ها

در این نسل نرم‌افزاری کاملتر از نرم‌افزار دستیابی به عنوان واسط بین برنامه‌های کاربردی و فایلهای محیط فیزیکی طراحی و ایجاد شد. در این نسل دریافتند که می‌توان برنامه‌های کاربردی را در قبال رشد فایلها (File Growth) مثلاً افزودن یک فیلد به یک نوع رکورد از یک فایل مصون نگاه داشت. تا قبل از این نسل برنامه‌های کاربردی فقط در قبال تغییرات سخت‌افزاری و رشد کمی فایلها (یعنی افزایش حجم داده‌های فایل) مصون بودند. مشخصات کلی این نسل عبارتند از :

۱ـ نرم‌افزار نسبتاً پیچیده‌ای به نام سیستم مدیریت داده‌ها واسط بین برنامه کاربردی و محیط فیزیکی ذخیره‌سازی است.

۲ـ فایلهای منطقی متعددی می‌توانند از داده‌های فیزیکی مشترک بهره‌برداری کنند و این فایلها می‌توانند به هم مرتبط باشند.

۳ـ میزان تکرار ذخیره‌سازی کاهش یافته است.

۴ـ داده‌های مشترک در کاربردهای متنوع به کار می‌روند.

۵ـ صحت داده‌های ذخیره شده تا حدی تامین می‌شود.

۶ـ نشانی‌دهی به داده‌ها در سطح فیلد یا گروهی از فیلدها امکان‌پذیر است.

۷ـ تسهیلاتی برای پردازش فایلها پیش‌بینی شده است.

۸ـ بازیابی به کمک چند کلید (Multikey Retrieval) امکان‌پذیر است.

۹ـ ترکیبی از انواع ساختارهای فایل به کار گرفته می‌شود.

٤ـ نسل چهارم ـ نسل DBMS

این نسل از اواخر دهه ۶۰ آغاز شد و هم‌اکنون نیز ادامه دارد. مهمترین خصیصه این نسل مستقل شدن برنامه‌های کاربردی (Application Program) از جنبه‌ها و خصوصیات محیط فیزیکی ذخیره‌سازی است که اصطلاحاً به آن استقلال داده‌یی فیزیکی (Physical Data Independence) می‌گویند. در بقیه این کتاب در واقع ویژگی‌های DBMS را شرح می‌دهیم لذا در اینجا فقط بعضی از ویژگی‌های اصلی آن را به صورت خلاصه بیان می‌کنیم.

۱ـ کاربران در یک محیط انتزاعی (Abstractive) و مبتنی بر یک ساختار داده‌یی تجریدی کار می‌کنند. بدین ترتیب برنامه‌های کاربردی از داده‌های محیط فیزیکی کاملاً مستقل می‌شوند. اصطلاح داده‌افزار (Data Ware) نیز ناظر به همین جنبه انتزاعی می‌باشد.

۲ـ امکان کنترل متمرکز روی تمام داده‌های عملیاتی وجود دارد و یکی از مزایای این کنترل متمرکز کاهش میزان افزونگی (Redundency) در ذخیره‌سازی داده‌هاست.

۳ـ نرم‌افزار پیچیده و جامع موسوم به سیستم مدیریت بانک اطلاعاتی (DBMS یعنی Data Base Management System) واسط بین برنامه‌های کاربران و محیط داخلی و فیزیکی ذخیره‌سازی است. این نرم‌افزار امکان می‌دهد تا کاربران در یک محیط انتزاعی کار کنند و در عین حال به داده‌های ذخیره شده دستیابی داشته و عملیات موردنظر خود را انجام دهند.

۴ـ سرعت دستیابی به داده‌ها بالا می‌باشد. ایمنی داده‌ها زیاد است و امکان استفاده اشتراکی از داده‌ها وجود دارد.

۵ـ در این نسل مفهوم چند سطحی بودن ساختار داده‌یی و معماری چند سطحی ذخیره‌سازی مطرح و به تدریج بسط یافت این سطوح از پائین به بالا عبارتند از ساختار فیزیکی بانک ـ ساختار داخلی بانک ـ ساختار ادراکی بانک و ساختار خارجی بانک (این سطوح را بعداً شرح می‌دهیم).

۶ـ رشدپذیری یکی دیگر از ویژگی‌های این سیستم است. این رشدپذیری به خاطر وجود معماری چند سطحی و نیز استقلال برنامه‌های کاربردی از ساختار ذخیره‌سازی و استراتژی دستیابی تامین می‌شود.

در واقع باید گفت اصلی‌ترین تفاوت این نسل با نسل‌های قبلی وجود حصاری نفوذناپذیر به نام سیستم مدیریت بانک اطلاعاتی یا DBMS است که هرگونه دستیابی به داده‌ها می‌بایست از طریق آن انجام شود.

برای راهبری این نرم‌افزار DBMS، مثل اطلاعات امنیتی درباره اجازه استفاده کاربران و همچنین انجام تغییرات به دو نوع نیروی انسانی نیاز است :

الف) مدیر بانک اطلاعات (Data Base Administrator = DBA) که مسئولیت تصمیم‌گیری و طراحی موارد مذکور را بر عهده دارد.

ب) برنامه‌ساز بانک اطلاعات (Data Base Programmer = DBP) که تصمیمات مدیر را پیاده‌سازی می‌کند. تنها کسانی که می‌توانند دور از چشم DBMS به داده‌ها دسترسی داشته باشند مدیر و برنامه‌سازان مجاز بانک اطلاعات هستند. DBMS در واقع انقلابی در بانکهای اطلاعاتی به شمار می‌آید.

در اواخر این نسل دو تحول دیگر در تکنولوژی بانک اطلاعاتی پدید آمد : یکی طراحی و ایجاد بانکهای اطلاعاتی توزیع شده (Distributed Data Base) تحت شبکه‌ها (که بعداً شرح می‌دهیم) و دیگری طراحی و ایجاد سیستم‌های مدیریت بانک اطلاعاتی برای کامپیوترهای شخصی.

۵ـ نسل پنجم نسل بانك معرفت یا پایگاه شناخت

این نسل به نسل Knowledge Base یا پایگاه شناخت (پایگاه دانش) معروف است. در این تکنولوژی، ضمن استفاده از تکنولوژی بانکهای اطلاعاتی با بهره‌گیری از منطق صوری (Formal Logic)، سیستم‌های خبره (Expert System)، مفاهیم هوش مصنوعی (Artificial Intelligence = AI) و پردازش زبان طبیعی (Natural Language) سیستمی طراحی و ایجاد می‌شود که قادر به استنتاج منطقی از داده‌های ذخیره شده است.

تعریف Knowledge Base : مجموعه‌ای از واقعیت‌های ساده و قواعد عام که نشاندهنده بخشی از جهان واقعی (Real World) باشد.

منظور از جهان واقعی محیطی است عینی که در رابطه با آن می‌خواهیم اطلاعاتی داشته باشیم و گاه به آن محیط عملیاتی نیز گفته می‌شود.

تفاوت اساسی بین بانکهای اطلاعاتی و بانکهای معرفت در این است که بانک معرفت حاوی مجموعه‌ای است از واقعیت‌های ساده و قواعد عام که به طور صریح بیان شده باشند، همراه با تعداد نسبتاً کمی از قواعد عام که به طور ضمنی بیان می‌شوند.

از خصوصیات اساسی این نسل وجود سیستمی به نام «سیستم بانک معرفت» یا KBS مخفف Knowledge Base System است که خود مجموعه‌ای است از امکانات نرم‌افزاری و سخت‌افزاری که مسئولیت ذخیره‌سازی معرفت، تامین امنیت (Security) و جامعیت (Integrity) بانک و نیز تامین نیازهای کاربران را بر عهده دارد.

اضافه کردن بانک دانش (Knowledge – Base) و قدرت استنتاج منطقی به بانک، با نامهای مختلفی مثل بانک اطلاعات پویا، بانک اطلاعات خبره، بانک اطلاعات استنتاجی (Deductive Database)، بانک اطلاعات بازگشتی (Recursive Database) بانک اطلاعات استنباطی (Refrential Database) و غیره مطرح می‌شود.

مثلاً اگر در بانک اطلعات دانشگاه داشته باشیم «مدرک آقای امیری دکتری است» و «تخصص آقای امیری شبکه‌های کامپیوتری است» می‌توان خود به خود نتیجه گرفت که «آقای امیری می‌تواند درس شبکه‌های کامپیوتری را تدریس کند». بنابراین با استفاده از این روش، اطلاعات

جدیدی مرتباً کشف شده و بر داده‌های بانک افزوده می‌گردد. همچنین می‌توان بسیاری از اطلاعات را ذخیره نکرد و در موقع لزوم آنها را استنتاج نمود.

تذکر : نوع جدید بانکهای اطلاعاتی بانکهای اطلاعاتی شیءگرا یا OODB (مخفف Object Oriented Data Base) می‌باشد که در این کتاب آن را شرح نمی‌دهیم.

تعاریف اولیه

موجودیت : (پدیده، نهاد یا Entity) مفهوم کلی پدیده، شیء یا فردی که مورد آنها می‌خواهیم اطلاع داشته باشیم.

صفت خاصه : (Attribute) ویژگی جداساز یک نوع موجودیت از نوع دیگر است.

مثال : موجودیت دانشجو می‌تواند دارای صفات خاصه : نام ـ نام‌خانوادگی ـ سال تولد ـ معدل ترم پیش باشد و مقادیر این صفات خاصه برای یک دانشجوی خاص برابر است با :

علی ـ اکبر ـ ۱۳۵۲ ـ ۱۶

به صفات خاصه گاهی اوقات خواص (Properties) نیز گفته می‌شود.

نکته : هر صفت فاصله دارای دو موله می‌باشد الف) اسم صفت خاصه ب) مقدار صفت خاصه که در صورت وجود این جفت مولفه «اطلاع» حاصل می‌گردد.

Relation Ship : به ارتباط بین موجودیتها در یک محیط عملیاتی گفته می‌شود. مثل ارتباط بین دانشجویان و اساتید در محیط عملیاتی دانشگاه (مثلاً دانشجوی A با چه اساتیدی در این ترم درس گرفته است).

فیلد : کوچکترین واحد داده ذخیره شده می‌باشد.

رکورد : مجموعه‌ای از فیلدهای مرتبط با هم می‌باشد.

فایل : مجموعه‌ای از تمام نمونهٔا یا رویدادهای یک نوع رکورد.

سیستم فایل : به ساختار کلی نام‌گذاری، ذخیره‌سازی و سازماندهی فایلها در یک سیستم عامل، سیستم فایل گفته می‌شود.

داده (Data)

ANSI دو تعریف برای داده ارائه کرده است :

۱ـ نمایش واقعیات (Facts)، پدیده‌ها، مفاهیم یا معلومات به صورتی صوری و مناسب برای برقرای ارتباط، تفسیر یا پردازش توسط انسان یا امکانات خودکار.

۲ـ هر نمایشی، اعم از کاراکتری یا کمیت‌های آنالوگ، که به آن معنایی منتسب است و یا باید منتسب شود و به طور کلی ما عملیاتی را روی داده یا اقلام داده‌یی انجام می‌دهیم تا در مورد یک موجودیت اطلاعاتی تهیه کنیم.

از نظر ساختاری، داده عبارت است از مقادیر صفحات خاصه انواع موجودیتها و مادر این درس از این تعریف استفاده می‌کنیم.

داده ارزشهای واقعی هستند که از طریق مشاهده و تحقیق بدست می‌آیند و به عبارت دیگر داده نمودی از وقایع، معلومات، رخدادها، پدیده‌ها و مفاهیم می‌باشند.

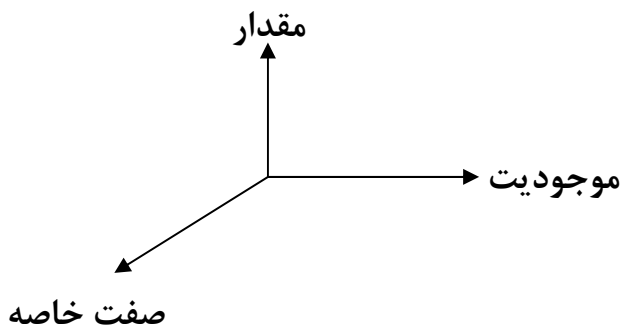
اطلاع (Information)

ANSI اطلاع را اینگونه تعریف کرده است :

«معنایی که انسان به داده منتسب می‌کند، از طریق قراردادهای شناخته شده‌ای که در نمایش داده به کار رفته‌اند» می‌توان گفت از پردازش داده‌ها، اطلاعات حاصل می‌شود و یا داده پس از آنکه مورد تفسیر قرار گرفت تبدیل به اطلاع می‌شود.

در این درس تفسیر داده به نحوی که حامل معنا و شناخت شود با انتساب یک مقدار به اسم صفت خاصه صورت می‌پذیرد و می‌گوئیم هنگامی که اسم صفت خاصه و مقدار منسوب به آن در دست باشد، اطلاعی در مورد موجودیت حاصل می‌شود.

به طور کلی یک فقره اطلاع (مقدار و اسم صفت خاصه از یک موجودیت) را می‌توان به صورت نقطه‌ای در یک فضای سه بعدی نمایش داد.



مثلاً برای دانشجو (علی = نام) یک فقره اطلاع است.

تذکر یک پدیده یا واقعیت می‌تواند از دید کاربران یک محیط عملیاتی، موجودیت تلقی شود و از دید کاربرانی دیگر، از همان محیط یا محیط دیگر، صفت خاصه باشد. مثلاً برای بانک اطلاعاتی اداره راهنمایی و رانندگی رنگ یکی از صفات خاصه موجودیت اتومبیل است ولی برای بانک اطلاعاتی کارخانه رنگ‌سازی خود رنگ موجودیتی است که صفات خاصه‌ای مثل مواد شیمیایی، درجه تبخیر و غیره دارد.

تعریف : داده‌های عملیاتی مورد نیاز کاربر (Operational Data) داده‌هایی است که کاربر بطور روزانه با آنها سر و کار دارد.

تذکر : در بعضی از کتابها کلمات «اطلاعات» و «داده‌ها» به جای هم به کار می‌روند. بعضی از نویسندگان، داده‌ها را همان چیزهایی می‌دانند که در بانک اطلاعاتی ذخیره می‌شوند و «اطلاعات» را به معنای آن داده‌ها از دید کاربر می‌دانند.

سرمایه‌های یک سازمان

امروزه کل سرمایه‌های یک سازمان تشکیل شده از : داده‌ها (داده‌افزار) ـ نرم‌افزار ـ سخت‌افزار ـ امکانات مالی ـ نیروهای تخصصی

داده‌های پایدار (مانا)

داده‌های بانک اطلاعاتی، داده‌های پایدار و با ثبات هستند. منظور از پایداری این است که نوع داده‌های بانک اطلاعاتی با داده‌های ناپایداری مثل داده‌های ورودی، داده‌های خروجی، دستورات کنترلی، صفها، بلوکهای کنترل نرم‌افزار و نتایج میانی که ماهیت آنها گذرا است، تفاوت دارد.

به این دلیل می‌گوئیم داده‌های بانک اطلاعاتی پایدار است که «وقتی داده‌ها توسط سیستم مدیریت بانک اطلاعاتی برای ورود به بانک پذیرفته شد، فقط در صورتی می‌تواند حذف شود که درخواستی به سیستم مدیریت بانک اطلاعاتی ارسال شود و با اثرات جانبی ناشی از اجرای برنامه حذف نخواهد شد. با توجه به این پایداری، تعریف دقیق‌تر بانک اطلاعاتی به صورت زیر خواهد بود :

بانک اطلاعاتی مجموعه‌ای از داده‌های پایدار است که توسط سیستم‌های کاربردی موجود در موسسه‌ای مورد استفاده قرار می‌گیرد. مثلاً در موسسه دانشگاه داده‌های مربوط به دانشجویان ذخیره می‌شود.

تذکر : در کتابهای قدیمی به جای واژه «اده‌های پایدار» از «داده‌های عملیاتی» استفاده می‌شد. سیستم‌های بانک اطلاعاتی عملیاتی، برنامه‌هایی هستند که بارها اجراء می‌شوند تا عملیات روزانه موسسات را انجام دهند و به چنین محیط‌هایی پردازش تراکنش پیوسته (Online Transaction Processing) نیز گفته می‌شود. اما امروزه بانکهای اطلاعاتی برای کارهای دیگری نظیر پشتیبانی تصمیم‌گیری است. داده‌های انبارداری معمولاً شامل اطلاعات خاصه‌ای نظیر مجموع، میانگین و غیره است. اطلاعات خلاصه در دوره‌های زمانی خاصی از بانک اطلاعاتی استخراج می‌شوند (مثل روزانه یا هفتگی)

فصل دوم

نمودار EER (طراحي كلي بانك اطلاعاتي)

مدل‌سازي داده‌اي (Data Modeling)

در طراحی یک بانک اطلاعاتی ابتدا می‌بایست مراحلی چون امکان‌سنجی، بررسی نیازها و محدودیتها، بررسی سیستم دستی موجود و غیره صورت گیرد که این مراحل در درس تجزیه و تحلیل سیستم‌ها مطرح می‌گردد.

پس از انجام مراحل فوق طراح بانک به کمک نمودارهایی، شمای کلی بانک را مستقل از مدل بانک (جدولی ـ شبکه‌ای ـ سلسله مراتبی) و نیز مستقل از جنبه‌های برنامه‌نویسی ترسیم می‌کند. برای این کار مدلهای مختلفی از جمله مدل دودویی (Binary Model)، Semantic Data Model، مدل ER، مدل ER گسترش یافته، مدل شی‌گراء (OR) و غیره وجود دارد. ما در این فصل مدل ER و سپس EER را که معروف‌تر از بقیه هستند شرح می‌دهیم.

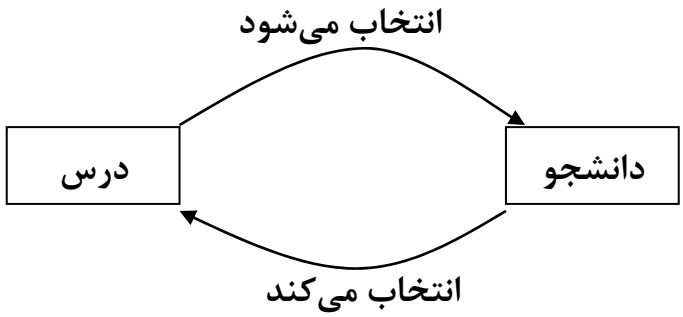
تذکر : این مدلها اصولاً ابزار طراحی هستند و نه پیاده‌سازی

تشخیص موجودیتها و بازشناسی ارتباط بین آنها و ترسیم نمودارهای ER و EER از وظایف طراح بانک اطلاعاتی است.

نمودار ER(Entity Relationship Diagram)

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

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

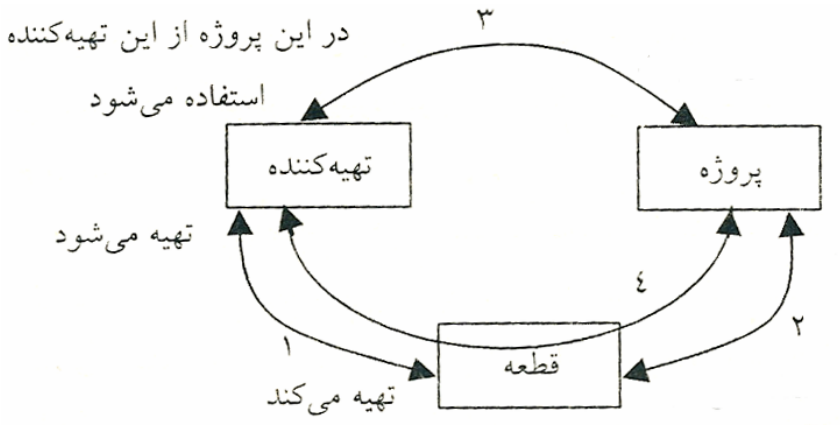


هر ارتباطی بین موجودیتها مفهوم یا سمانتیک (Smantic) خاصی است. به بیان دیگر حاوی بار اطلاعاتی مخصوصی است که هر ارتباط دیگری فاقد آن است. سمانتیک مستتر در هر ارتباط باید به نحوی در بانک ذخیره شود.

تذکر : همیشه لزومی ندارد که یک ارتباط حتماً بین دو موجودیت باشد ممکن است بین بیش از دو موجودیت یک ارتباط وجود داشته باشد.

مثال ۲ : بین موجودیت‌های تهیه کننده (S)، قطعه (P) و پروژه (J) ارتباطات زیر را می‌توان در نظر گرفت.

توجه کنید از سه ارتباط ۱ و ۲و ۳ همیشه نمی‌توان ارتباط ۴ را نتیجه گرفت به بیان دیگر سمانتیک موجود در سه ارتباط ۱ و ۲و ۳ همان سمانتیک ۴ را تشکیل نمی‌دهد.



مثال ۳ : سمانتیک ۱ : S2 قطعه ۳ را تهیه کرده است.

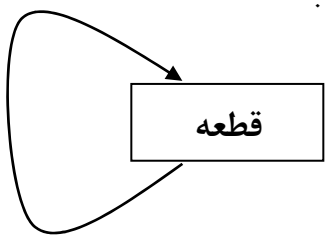
سمانتیک ۲ : قطعه P3 در پروژه J4 به کار رفته است.

سمانتیک ۳ : S2 برای پروژه J4 قطعه تهیه کرده است.

از سه عبارت بالا نمی‌توان همیشه نتیجه گرفت که S2 قطعه ۳ را برای پروژه J4 تهیه کرده است.» چنین حالاتی که گاه باعث بروز اشتباه در استنتاج اطلاع می‌شوند دام ارتباطی یا دام الحاقها (Connection Trap) می‌گویند.

نکته : ممکن است یک موجودیت با خودش ارتباط داشته باشد.

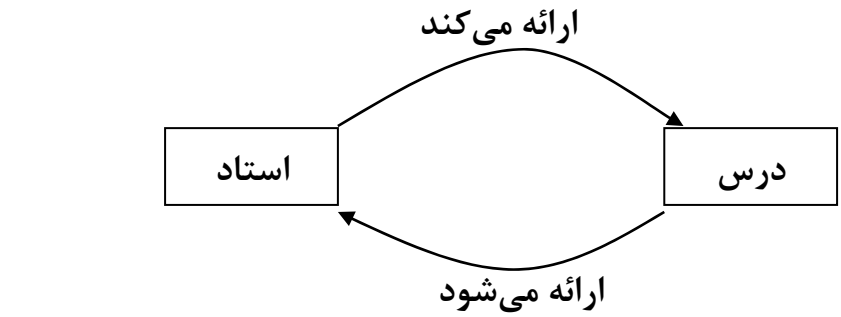
مثال ۴ :

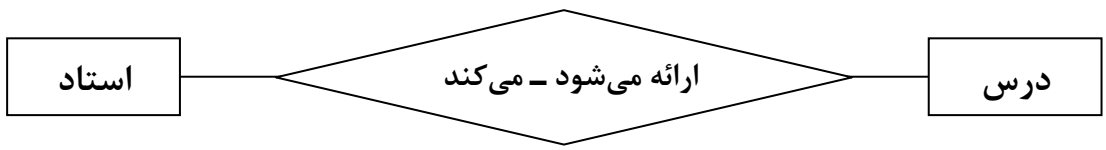


این بدین معناست که «یک قطعه از قطعه یا قطعات دیگر ساخته شده است».

تذکر : در کتابهای جدید ارتباط را داخل لوزی ترسیم می‌کنند.

مثال ۵ : نمودار روبرو معادل نمودار زیر است :





نکته : در ترسیم نمودار ER درجه ارتباط (Relationship Degree) می‌تواند یک به یک (1:1)، یک به چند (1: n) چند به یک (n : 1) یا چند به چند (n : n) باشد.

مثال ۶ : ارتباط یک به یک

در این شکل هر استاد یک درس و هر درس فقط توسط یک استاد ارائه می‌شود. البته ممکن است استادی اصلاً درس نداشته باشد یا درسی توسط هیچ استادی این ترم ارائه نگردد.

مثال ۷ : ارتباط چند به یک

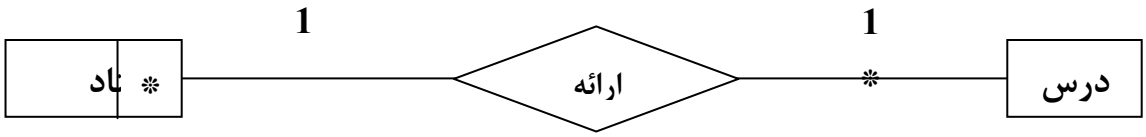
در این شکل چند استاد ممکن است یک درس را ارائه کنند ولی هر استاد فقط یک درس را ارائه می‌کند.

مثال ۸ : ارتباط چند به چند

در این شکل هر درس ممکن است توسط چند استاد ارائه شود و هر استاد ممکن است چند درس مختلف را ارائه کند.

در شکلهای فوق شرکت موجودیتها در ارتباط اختیاری بود یعنی ممکن بود استادی درسی ارائه نکند و یا درسی این ترم ارائه نشود. در نمودار ER برای اینکه شرکت در ارتباط اجباری شود علامت × به جای روی خط ارتباط، داخل مستطیل موجودیت ترسیم می‌شود.

مثال ۹ :

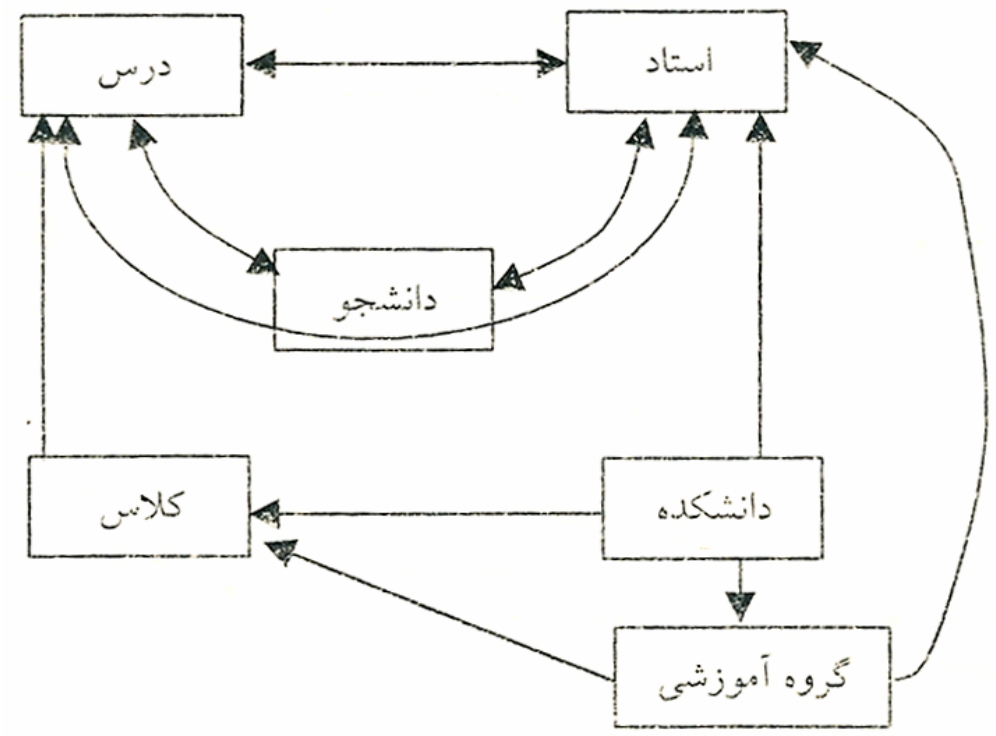


در این شکل هر استاد حتماً باید درسی ارائه کند (فقط یک درس) و هر درس فقط توسط یک استاد ارائه می‌شود (البته ممکن است درس خاصی ارائه نشود) ولی استادها نمی‌توانند درس ندهند.

نکته : ارتباط بین موجودیتها به تعبیری خود یک نوع موجودیت است زیرا با توجه به تعریف موجودیت (پدیده شیء یا چیزی که می‌خواهیم در موردش اطلاع داشته باشیم) وجود ارتباط نیز پدیده‌ای است که باید در مورد آن اطلاعات در بانک داشته باشیم.

پس بانک اطلاعاتی به تعبیری مجموعه‌ای از اطلاعات در مورد موجودیتها یک محیط عملیاتی و ارتباط بین آنها می‌باشد.

مثال ۱۰ : در شکل زیر نمودار ER یک دانشکده ترسیم شده است. سمانتیک هر یک از ارتباطات در شکل مشخص سازید.

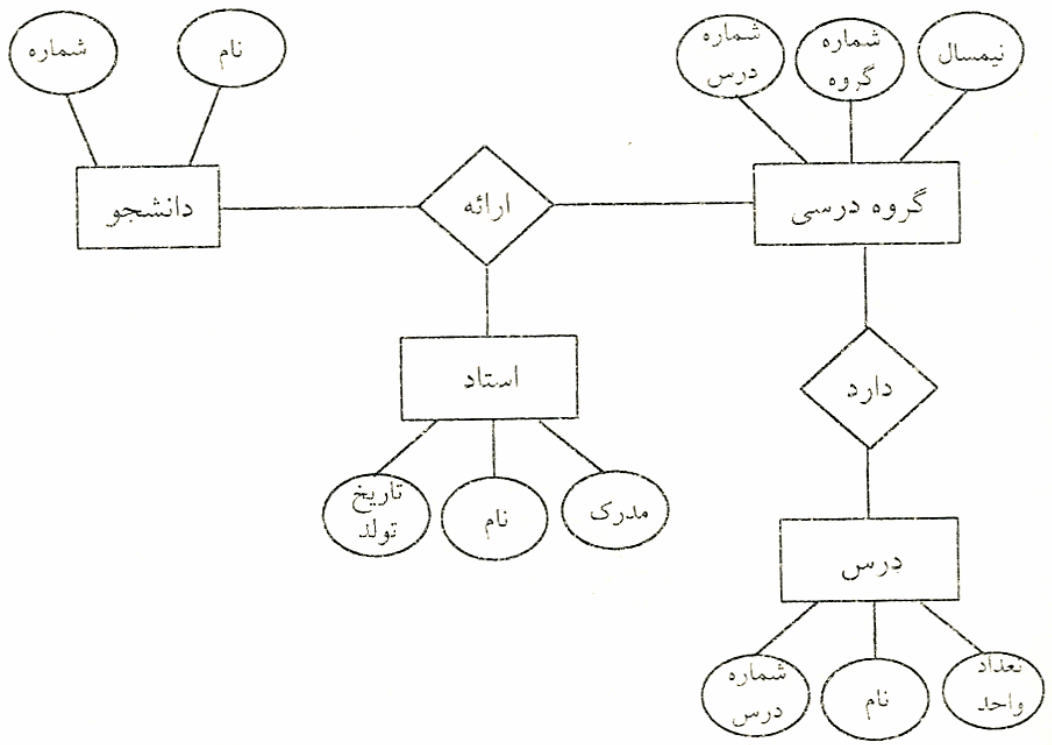


نمودار EER

در سال ۱۹۷۶ چن (Chen) از دانشگاه MIT مدل ER (Entity Relation) را جهت طراحی بانک پیشنهاد کرد. این مدل طول زمان پیشرفت کرد و بنام Extended ER = EER معروف گردید.

در این طراحی کلی موجودیت با مستطیل، صفتها به صورت بیضی و ارتباط (Relationship) بصورت لوزی ترسیم می‌شوند.

مثال ۱۱ :



انواع صفت در نمودار EER

الف) صفت کلیدي

کلید عبارت است از یک یا چند صفت که در یک موجودیت منحصر به فرد باشد. مثلاً در موجودیت دانشجو شماره دانشجویی کلید است. چون هر دانشجو یک شماره یکتا دارد.

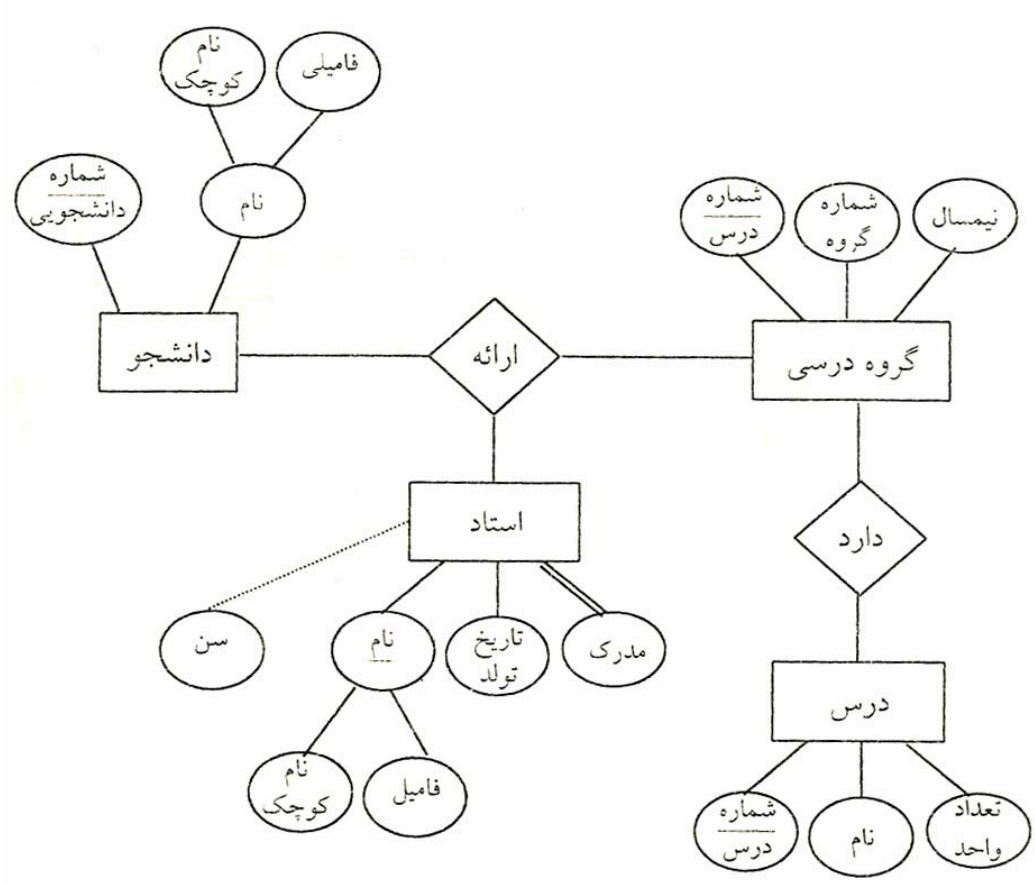
ولی نام نمی‌تواند کلید باشد. گاهی اوقات یک صفت تنها نمی‌تواند کلید باشد بلکه مجموعه‌ای از دو یا چند صفت با همدیگر کلید می‌شوند. مثلاً نام و شماره شناسنامه هر یک به تنهایی کلید نیست ولی هر دو با هم کلید می‌شوند. برای مشخص کردن کلید یک موجودیت زیر آن صفت خط می‌کشیم. (...، نام پدر، نام، شماره دانشجو)

ب) صفت ساده و مرکب

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

در نمودار EER صفات ترکیبی را در دو سطح می‌کشیم.

مثال ۱۲ :



می‌توان اجزاء صفات مرکب را داخل پرانتز نوشت.

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

ج) صفت تك مقداري يا چند مقداري

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

د) صفت مشتق

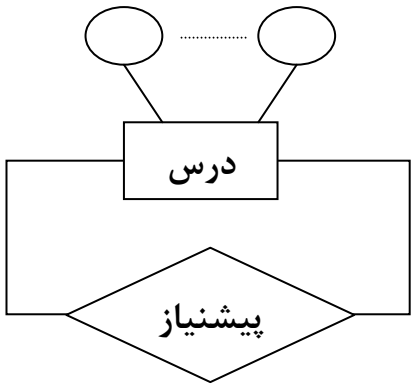
صفت مشتق صفتی است که به کمک صفتهای دیگر می‌توان آن را محاسبه کرد. مثلاً سن استاد یک صفت مشتق است که با توجه به تاریخ تولد قابل محاسبه می‌باشد. تصمیم‌گیری در مورد صفت مشتق به عهده طراح است مثلاً معدل کل برای دانشجو بهتر است مشتق باشد زیرا مرتباً با گذراندن دروس بیشتر عوض می‌شود ولی برای فارغ‌التحصیلان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

صفت مشتق در نمودار EER بصورت خط‌چین ترسیم می‌شود. در شکل مثلا ۱۲ سن استاد صفت مشتق است.

درجه ارتباط

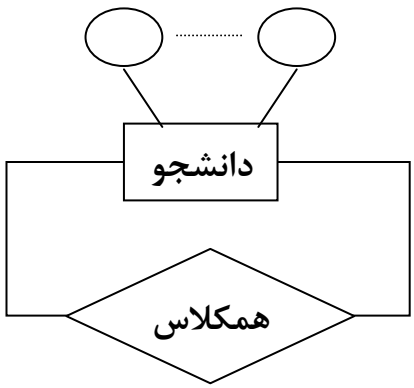
درجه ارتباط برابر تعداد موجودیتهایی است که در آن ارتباط مشارکت دارند معمولاً این درجه ارتباط ۱ یا ۲ و یا ۳ است و درجات بالاتر بندرت استفاده می‌شوند.

مثال ۱۳ : ارتباط درجه ۱ :



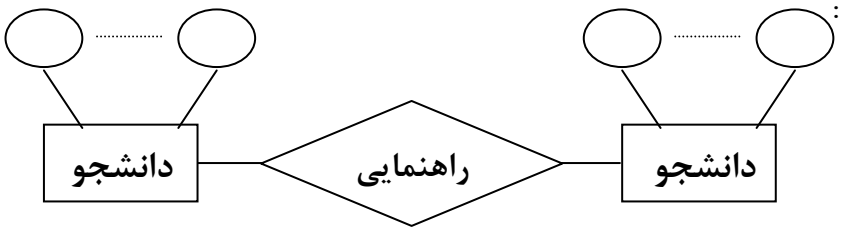
این ارتباط مشخص می‌سازد چه درسی پیشنیاز چه درسی است.

این ارتباط مشخص می‌سازد کدام دانشجو همکلاس کدام دانشجو است. این هم ارتباط درجه ۱ می‌باشد.



تذکر : در هر یک از شکلهای بالا فقط یک پدیده یا موجودیت وجود دارد.

مثال ۱۴ : ارتباط درجه ۲ :

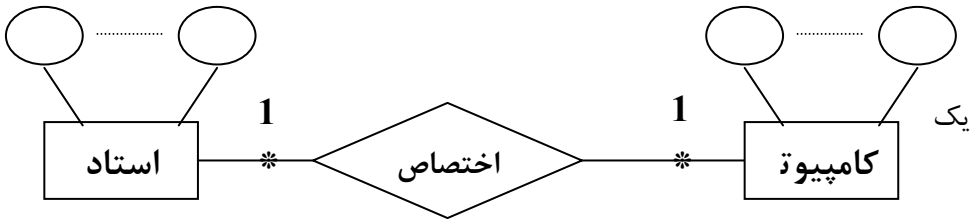


شکل ترسیم شده برای استاد ـ دانشجو ـ گروه درسی در مثال ۱۲ نمونه‌ای از ارتباط درجه ۳ است.

اتصال (Connectivity) و حد (Cardinality)

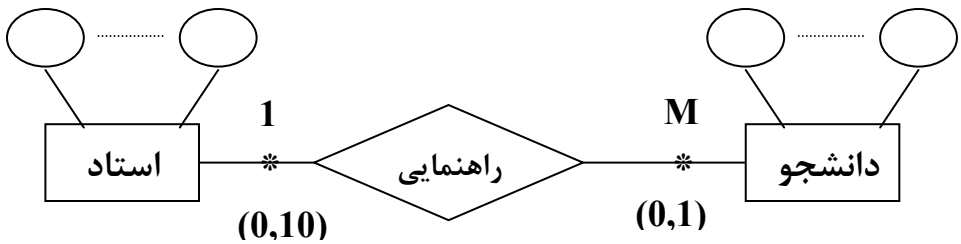
ارتباط از نظر نوع بر سه نوع است : N-M, 1-M, 1-1

مثال ۱۵ : در یک دانشگاه ممکن است هر استاد یک کامپیوتر اختصاصی داشته باشد.



ارتباط یک به یک

مثال ۱۶ :

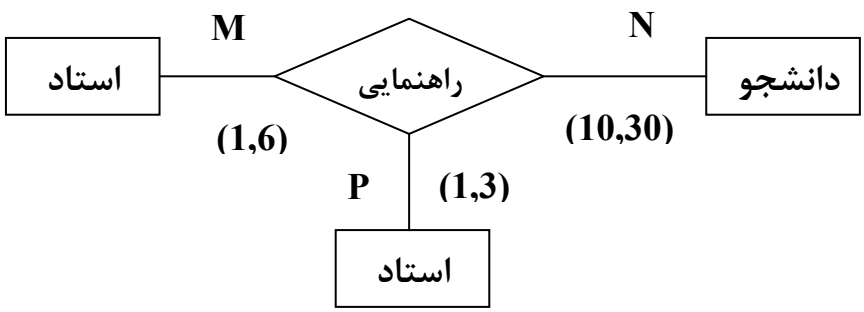


در این مثال ارتباط 1-M است یعنی یک استاد چندین دانشجو را راهنمایی می‌کند.

مشخصه دیگر ارتباط حد آن است که در پائین خط ارتباط مقدار حداقل و حداکثر آن در پرانتز نوشته می‌شود. در شکل فوق (0,10) یعنی یک استاد ممکن است راهنمای هیچ دانشجویی نباشد و حداکثر ۱۰ دانشجو را راهنمایی کند.

همچنین در شکل فوق (0,1) یعنی یک دانشجو ممکن است استاد راهنما نداشته باشد و حداکثر توسط یک استاد راهنمایی می‌شود. اتصال و حد برای ارتباطهای سه‌تایی گاهی مبهم است.

مثال ۱۷ :



ـ هر دانشجو حداقل ۱ و حداکثر ۶ درس می‌گیرد.

ـ هر درس ممکن است ۱۰ تا ۳۰ دانشجو داشته باشد.

ـ هر استاد می‌تواند ۱ تا ۳ درس را تدریس کند.

در شکل فوق این ابهام وجود دارد که حدود (10,30) مربوط به دانشجوست یا استاد؟ این ابهام می‌بایست با توضیح برطرف کرد. البته این موضوع یکی از نارسایی‌های مدل EER است.

شرکت اجباري یا اختیاري در ارتباط

اگر در دانشگاهی قانونی وجود داشته باشد که «هر استاد حداقل باید یک درس را تدریس کند» آنگاه ارتباط استاد با گروه درسی اجباری می‌شود ولی اگر در دانشگاهی تدریس استاد اختیاری باشد به کمک یک دایره کوچک تو خالی این موضوع نشان داده می‌شود.

مثال ۱۸ :

در این شکل تدریس استاد اختیاری است.

اگر دایره نداشته باشیم ارتباط اجباری است.

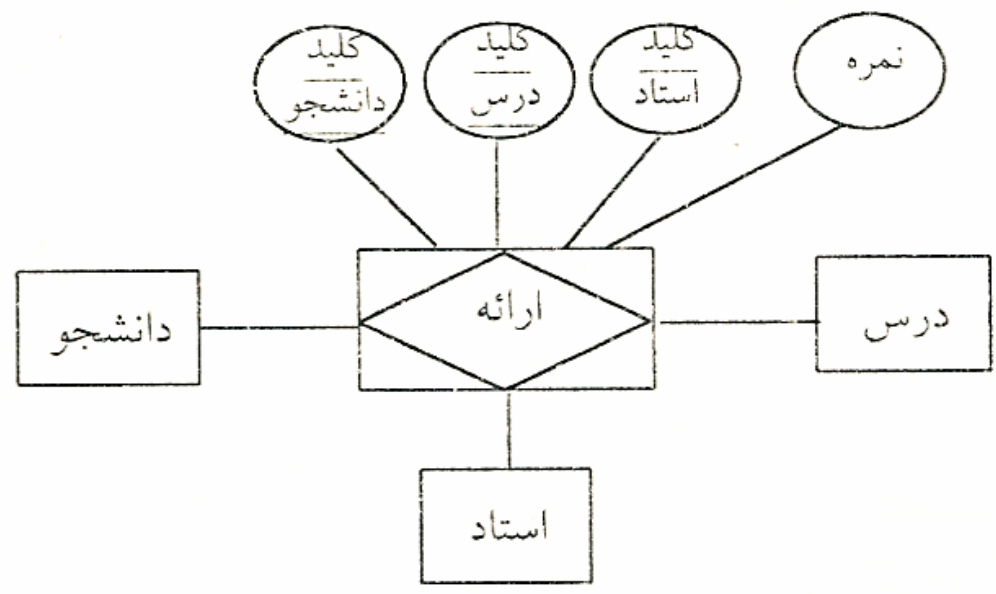
صفت در ارتباط

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

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

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

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



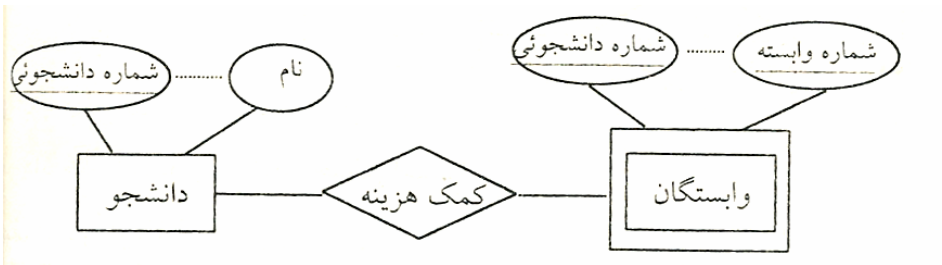
وابستگي وجودي (Existence Dependency)

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

این نوع وابستگی را، وابستگی وجودی و پدیده وابسته را موجودیت ضعیف (Weak Entity) می‌نامند. پدیده وابسته باید کلید پدیده اصلی را که به آن وابسته است به ارث ببرد تا به سادگی قابل شناسایی باشد. پدیده وابسته را با دو مستطیل تو در تو نمایش می‌دهیم.

مانند شکل زیر.

مثال ۲۰ :

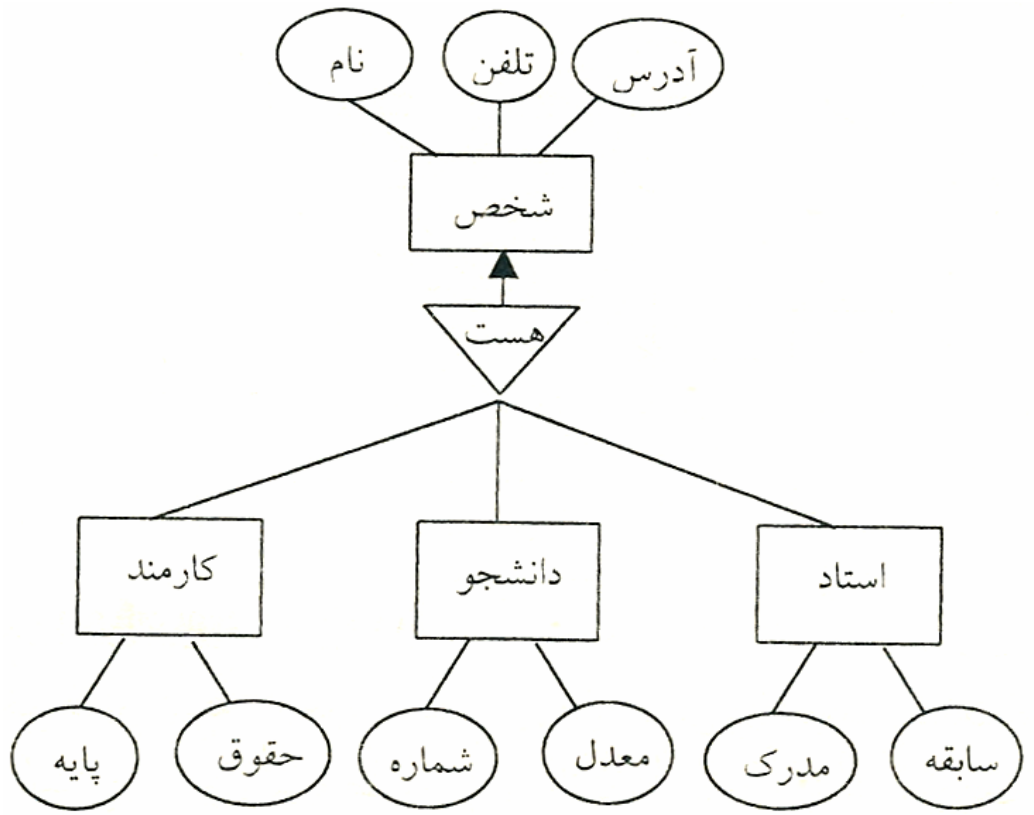


تذکر : رابطه وابستگی را نیز با دو لوزی تو در تو نمایش می‌دهیم.

اشتراك صفت (ارث بري)

در بسیاری از موارد موجودیتها در یک بانک، صفات مشترکی دارند.

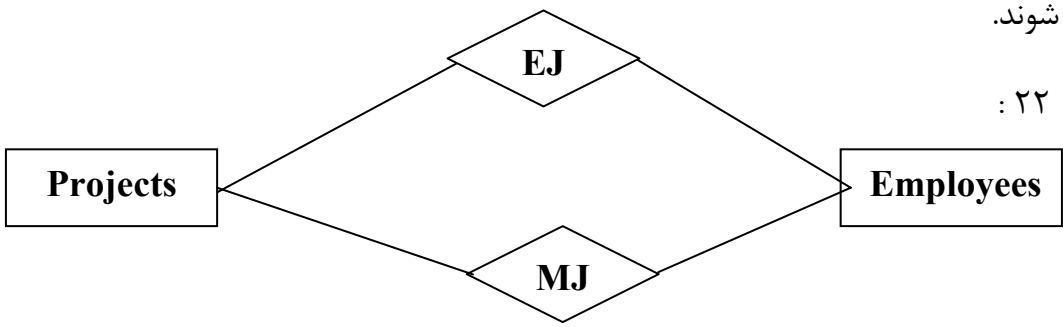
مثال ۲۱ : در دانشگاه تمامی افراد اعم از استاد، دانشجو و کارمند دارای صفاتی مثل نام، تلفن و آدرس هستند. برای جلوگیری از تکرار بی‌رویه ارتباطی از نوع ارث‌بری به صورت زیر تعریف کرد :



ز

تذکر ۱ : مجموعه‌آی از انواع موجودیتها ممکن است از طریق پیوندهای متعدد به یکدیگر پیوند داده شوند.

مثال ۲۲ :



در شکل فوق،دو رابطه مجزا شامل پروژه‌ها و کارکنان موجود است : یکی از آنها (EJ) تعیین می‌کند که کارکنانی به پروژه‌ها منصوب می‌شوند و دیگری (MJ) نشان می‌دهد که کارکنان، پروژه‌ها را مدیریت می‌کنند.

تذکر ۲ : در نمودار EER علامت لوزی (ارتباط و رابطه بین دو موجودیت) نیز مانند موجودیتهای اصلی بخشی از داده‌ها بوده و همانند موجودیتهای اصلی باید در بانک اطلاعاتی نمایش داده شوند.

رابطه (با نماد لوزی) را می‌توان به عنوان یک موجودیت در نظر گرفت.

تذکر ۳ : مدل دیگری برای طراحی کلی بانک اطلاعاتی، روش NIAM می‌باشد. البته EER مشهورتر از NIAM است.

فصل سوم

تعریف پایگاه داده‌ها

پایگاه داده‌ها چیست؟

اصطلاحات بانک اطلاعاتی، پایگاه داده‌ها، بانک داده‌ها، پایگاه اطلاعات معادل یکدیگر می‌باشند. بانک اطلاعاتی «مجموعه‌ای است از داده‌های ذخیره شده (در مورد انواع موجودیتهای یک محیط عملیاتی و ارتباطات بین آنها) به صورت مجتمع و مبتنی بر یک ساختار، تعریف شده به طور صوری با حداقل افزونگی، تحت کنترل متمرکز، مورد استفاده یک یا چند کاربر به طور اشتراکی و همزمان»

منظور از تعریف شده به طور صوری آن است که سیستم باید به کاربران امکان دهد تا داده‌های خود را آنگونه که خود می‌بینند، به صورت انتزاعی و بدور از جنبه‌های پیاده‌سازی و نشست فیزیکی آنها روی رسانه تعریف کنند. مجتمع و مبتنی بر یک ساختار به این معناست که کل داده‌های عملیاتی محیط موردنظر در یک ساختار مشخص به صورت یکجا ذخیره شده باشند. لازمه هر تجمعی وجود یک ساختار است.

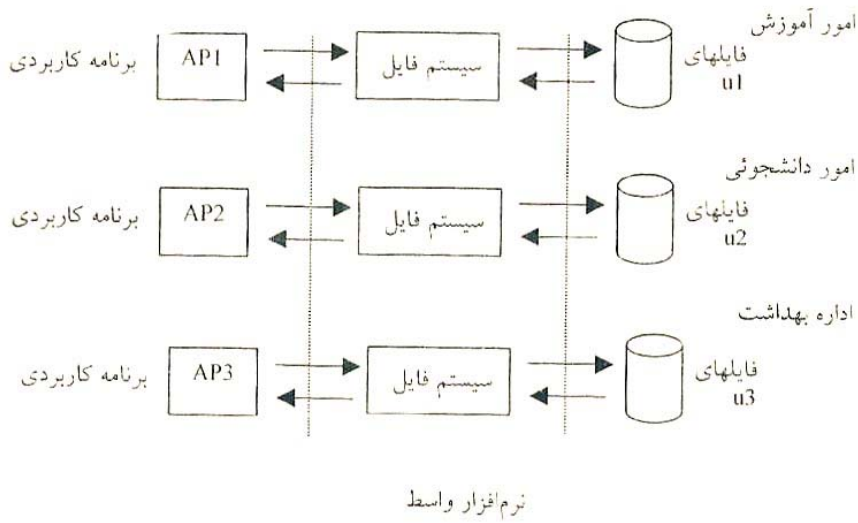
ساختار فیزیکی بانک همان ساختار فایلهایی است که آن را تشکیل می‌دهد. با مفهوم افزونگی در درس ذخیره و بازیابی آشنا شده‌اید. افزونگی عبارت است از تکرار مقادیر یک یا چند صفت خاصه در نمونه‌های مختلف یک نوع رکورد از یک فایل، به بیانی دیگر ذخیره‌سازی آن مقادیر در بیش از یک نقطه از فایل. تجمع داده‌ها و وحدت ذخیره‌سازی باعث از بین رفتن پدیده افزونگی خواهد شد یا آنرا به حداقل می‌رساند.

مثال ۱ : در محیط عملیاتی دانشگاه بخشهای امور آموزش ـ امور دانشجویی و اداره بهداشت را در نظر می‌گیریم. می‌خواهیم یک سیستم مکانیزه برای این بخشها پدید آوریم. در این مثالها تنها موجودیت دانشجو را در نظر می‌گیریم.

دانشجو دارای صفات خاصه متعددی است مثل نام ـ معدل دیپلم ـ شمارهٔ دانشجویی ـ تاریخ تولد ـ سال ورود ـ شماره دفترچه بیمه ـ وضعیت جسمانی ـ وضعیت مسکن و غیره که برای هر بخش می‌تواند قدری متفاوت باشد. برای ایجاد این سیستم دو روش کلی وجود دارد :

الف) روش کلاسیک

در این روش هر یک از بخشهای سه‌گانه به طور جداگانه سیستم خاص خود را ایجاد می‌کنند و با استفاده از امکانات سیستم فایل موجود در سیستم عامل و یک زبان سطح بالا فایلهایی را تعریف و ایجاد می‌کنند. در هر فایل رکورد موردنظر با فیلدهای موردنیاز تعریف می‌شود :



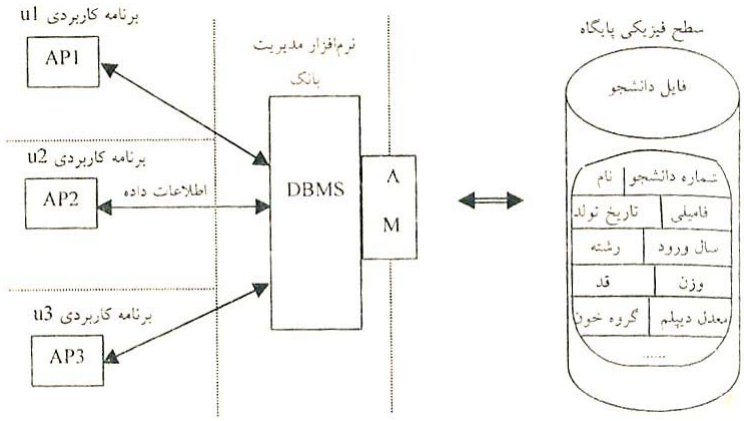
بسیاری از داده‌های مورد نیاز u1 همانهایی هستند که برای u2 یا u3 نیز مورد نیاز است. عدم تجمع داده‌های ذخیره شده و عدم وحدت ذخیره‌سازی به وضوح در این مثال مشهود است و در نتیجه پدیده افزونگی در ذخیره‌سازی داده‌ها وجود دارد.

ضمناً برنامه‌های کاربردی برای ایجاد و پردازش فایلهای خاصی نوشته شده‌اند و در صورت وجود تغییر در ساختار رکوردها و فایلها، برنامه‌ها نیز باید متناسباً تغییر کنند. در این روش امنیت و حفاظت داده‌ها مشکل است.

ب) روش بانکي

در این روش رکورد نوع دانشجو فقط یکبار در فایل ذخیره می‌شود و کاربران مختلف هر یک طبق نیاز خود از آن به طور همزمان، مشترکاً استفاده می‌کنند. در رکورد نوع دانشجو تمام صفات خاصه موردنیاز کاربران مختلف وجود دارد و طبعاً صفات خاصه مشترک، تنها یکبار در رکورد منظور می‌شوند. با آنکه در اینجا وحدت ذخیره‌سازی داریم ولی هر کاربری دید خاص خود را نسب به داده‌ها دارد. در اینجا کاربران مختلف می‌توانند به صورت همزمان با بانک کار کنند. یعنی هر کاربر بدون ایجاد محدودیت برای کاربر دیگر در هر لحظه می‌تواند بانک کار کند.

شکل زیر روش بانکی را نشان می‌دهد.



نرم‌افزار مدیریت بانک اطلاعاتی (DBMS = Data Base Management System) رابط بین برنامه‌های کاربردی و داده‌هاست و هرگونه دستیابی به داده‌ها از طریق DBMS صورت می‌گیرد و به این دلیل امنیت داده‌ها نیز در این روش زیاد است.

تذکر ۱ : همزمانی عملیات از جنبه دیگری نیز مطرح است و آن همزمانی از نظر سیستم است، یعنی فرآیندهای درون سیستمی بتوانند با همزمانی، در محیط سیستم جریان داشته باشند. اینکه تا چه حد همزمانی فرآیندها در درون سیستم امکان‌پذیر است بستگی دارد به نحوه طراحی سیستم مدیریت بانک، همچنین عملکرد سیستم عاملی که خود DBMS در محیط آن مشابه یک برنامه کاربردی اجراء می‌شود. این همزمانی به معماری سخت‌افزار کامپیوتر نیز بستگی پیدا می‌کند.

تذکر ۲ : سیستم بانک اطلاعاتی، سیستم کامپیوتری نیز بستگی پیدا می‌کند.

تذکر ۳ : به طور کلی داده‌ها در بانک اطلاعاتی، بخصوص در سیستم‌های بزرگ هم به صورت مجتمع و هم به صورت اشتراکی هستند. مفهوم مجتمع بودن این است که بانک اطلاعاتی مجموعه‌ای از فایلهاست که بخشی از اطلاعات اضافی بین آنها حذف شده است. مثلاً در بانک زیر که از دو فایل تشکیل شده است لازم نیست که در فایل وضعیت تحصیلی، محل تولد ذخیره شود چرا که در صورت نیاز با رجوع به فایل مشخصات می‌توان آن را بدست آورد.

فایل وضعیت تحصیلی	
معدل	شماره‌دانشجویی
۱۸	۱۷۳

فایل مشخصات		
متولد	نام	شماره دانشجویی
تهران	علی	۱۷۳

اگر بانک اطلاعاتی مشترک نباشد، بانک اطلاعاتی مشترک نباشد، بانک اطلاعاتی «شخصی» یا «کاربرد ویژه» نامیده می‌شود.

تذکر ۴ : بین بانک اطلاعاتی فیزیکی (مثلاً داده‌های ذخیره شده) و کاربران سیستم، لایه‌ای از نرم‌افزار وجود دارد که مدیر بانک اطلاعاتی یا کارگزار بانک اطلاعاتی (Database Server) یا سیستم مدیریت بانک اطلاعاتی (DBMS) نام دارد.

تذکر ۵ : سیستم مدیریت بانک اطلاعاتی، مهمترین جزء نرم‌افزاری در کل سیستم است. اما تنها جزء نرم‌افزاری نیست. بقیه اجزای نرم‌افزاری عبارتند از : برنامه‌های کمکی، ابزارهای تولید برنامه‌های کاربردی، ابزارهای طراحی، نویسندگان گزارش و از همه مهمتر مدیر تراکنش (Transaction Manager)

تذکر ۶ : سیستم مدیریت بانک اطلاعاتی به محصولات خاصی از فرشندگان خاص نیز اطلاق می‌شود. مثل بانک اطلاعاتی جهانی DB2 از شرکت IBM.

تذکر ۷ : داده‌ها حقایق موجود هستند که حقایق دیگری از آنها استنتاج خواهد شد. استنتاج حقایق دیگری از حقایق موجود، کاری است که سیستم مدیریت بانک اطلاعاتی هنگام پاسخ به کاربر انجام می‌دهد.

یک «واقعیت موجود» از نظر اهل منطق، گزاره درستی است، به عنوان مثال جمله «معدل دانشجو با شماره 753416 برابر ۱۷ می‌باشد» ممکن است گزاره درستی باشد. بدین ترتیب بانک اطلاعاتی «مجموعه‌ای از گزاره‌ای درست» است.

عناصر اصلي محيط بانكي

محیط بانک اطلاعاتی از عناصر اصلی زیر تشکیل شده است :

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

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

الف) سخت‌افزار ذخیره‌سازی داده‌ها

ب) سخت‌افزار پردازنده مرکزی

ج) سخت‌افزار ارتباطی

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

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

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

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

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

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

تذکر : داده‌های عملیاتی با داده‌های ورودی و خروجی تفاوت دارند. داده‌های ورودی اطلاعاتی هستند که نخستین بار وارد سیستم شده می‌توانند سبب ایجاد تغییر در داده‌های عملیاتی شوند یا خود جزیی از داده‌های عملیاتی محیط گردند.

داده‌های خروجی عبارتند از پیام‌ها، پاسخ‌ها و نتایجی که سیستم پیرو درخواست کاربر به او می‌دهد. این داده‌ها می‌توانند از داده‌های عملیاتی استخراج شوند ولی خود بخشی از داده‌های عملیاتی تلقی نمی‌شوند.

بانك اطلاعاتي توزيع شده

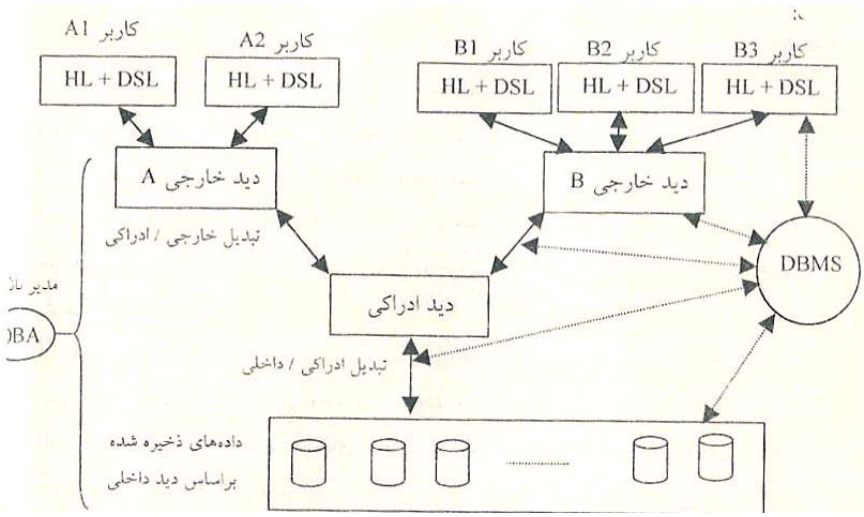
بانکهای اطلاعاتی را می‌توان در یک شبکه مورد استفاده قرار داد که خود دو نوع می‌باشند :

۱ـ بانکهای اطلاعاتی با داده‌های توزیع شده : در این شبکه یک کامپیوتر به عنوان کامپیوتر اصلی است که هدایت کل شبکه و مدیریت بانک اطلاعاتی را بر عهده دارد ولی داده‌ها در قسمتهای مختلف شبکه پخش شده‌اند.

۲ـ بانکهای اطلاعاتی با داده‌ها و سیستم توزیع شده : در این سیستم خود مدیریت بانک اطلاعاتی به صورت توزیع شده در چند کامپیوتر جای دارد یعنی پردازشها ممکن است در کامپیوترهای مختلف صورت گیرد و داده‌ها نیز در شبکه پخش می‌باشند. این نوع بانک در سیستم عاملهای توزیع شده مثل UNIX یا Windows NT قابل پیاده‌سازی است.

معماري سيستم بانك اطلاعاتي

مدل پیشنهادی ANSI (ارائه شده در سال ۱۹۷۵) برای معماری سیستم بانک اطلاعاتی به شکل زیر است :



معماری سیستم بانک اطلاعاتی از اجزاءزیر تشکیل شده است :

الف) دید ادراکی یا مفهومی (Conceptual View)

ب) دید خارجی (External View)

ج) دید داخلی یا فیزیکی (Internal View)

د) تبدیلات بین سطوح (Transformation یا Mappings)

هـ) زبان میزبان یا HL (Host Language)

و) زبان فرعی داده‌یی یا DSL (Data Sub Language)

به علاوه در چنین سیستمی سه عنصر مهم دیگر نیز وجود دارند :کاربر ـ DBMS – DBA

الف) دید ادراکي (مفهومي)

دید طراح بانک است از داده‌های ذخیره در بانک. یعنی داده‌های انواع موجودیتها و ارتباط بین آنها،آنگونه که طراح می‌بیند. دید طراح دیدی است جامع دیدهای همه کاربران و در عین حال متفاوت با هر یک از دیدهد. در این سطح ساختار داده‌ها مطرح می‌شود.

مثال ۲ : در محیط عملیاتی دانشگاه موجودیتهای درس و دانشجو را در نظر می‌گیریم. فرض می‌کنیم ساختار بانک به صورت جدولی باشد. طراح برای هر نوع موجودیت یک جدول طراحی می‌کند و برای نمایش ارتباط بین موجودیتها نیز یک جدول دیگر. مثلاً به صورت زیر :

به این ترتیب شمای ادراکی طراح بانک ادراکی به احکام کار با بانک مانند عملگرهای جدول‌یاب، جستجو، درج و حذف سطر نیاز داریم در نتیجه در شمای ادراکی کل داده‌های عملیاتی و ارتباطات بین آنها تعریف و تشریح می‌شود. شمای ادارکی به عنوان مجموعه‌ای از احکام یک برنامه است که باید کامپایل و تبدیل به برنامه مقصد شود.

جدول دانشجويي STTAB

سال ورود	نام خانوادگی	نام	شماره دانشجویی

جدول درس COTAB

ماهیت درس	تعداد واحد	نام درس	شماره درس

جدول دانشجو ـ درس STCOTAB

نمره	ترم	شماره درس	شماره دانشجو

تذکر : در سطح ادراکی ارتباط موجودیتها و صفات خاصه، امنیت و جامعیت داده‌ها و اطلاعات معنایی داده‌ها مطرح می‌گردد.

ب) ديد خارجي

دید خاص کاربر است از داده‌های ذخیره شده در بانک. هر کاربر دید خاص خود را دارد. همچنین چند کاربر می‌توانند دارای دید یکسانی باشند. مثل A1 , A2 که دید خارجی A را

دارند. همانند دید ادراکی، دید خارجی نیز برای معرفی شدن نیاز به یک ساختار یا مدل داده‌یی دارد. مدل داده‌یی مورد استفاده در سطح خارجی معمولاً همان سطح ادارکی است.

مثال ۳ : کاربر A1 می‌تواند دید روبر را روی جدول STTAB داشته باشد :

STTAB1

سال ورود	شماره دانشجویی

یا کاربر B2 دید روبرو را می‌تواند روی جدول STCOTA داشته باشد :

STCOTAB1

شماره درس	شماره دانشجویی

پس در دید خارجی می‌توان شرطهایی قرار داد تا سطرهایی (یا ستونهایی) از جدول مبنا در دید کاربر قرار گیرد. شمای خارجی هر کاربر نیز مثل شمای ادراکی، به هر حال یک برنامه است که باید کامپیایل شود. سیستم مدیریت بانک اطلاعاتی برای انجام درخواستهای یک کاربر باید به شمای خارجی مراجعه کند.

لایه خارجی تنها لایه‌ای است که کاربران با آن سر و کار دارند لایه‌ای دیگر به مدیر و برنامه‌سازان بانک مربوط می‌شود. طبق قانون «پنهان‌سازی اطلاعات» که می‌گوید «به هر کس

به همان اندازه اطلاعات بده که نیاز دارد و نه بیشتر»؛ در لایه خارجی دیدهای مختلف کاربران مطرح است تا هر کدام بخشی از بانک را که نیاز دارند ببینند.

تذکر : سطح خارجی نزدیک‌ترین سطح به کاربران است.

ج) دید داخلي

در این سطح در واقع فایلهای محیط فیزیکی تعریف می‌شود. از نظر محتوا، ساختار و استراتژی دستیابی. این تعریف در اساس همانست که در محیط‌های غیربانکی برای ایجاد فایلها لازم است.

یک سیستم بانک اطلاعاتی، کاربران اساساً به مسائل این سطح نمی‌پردازند. در سیستم‌های موجود طراح بانک دخالت چندانی در این سطح ندارد ولی در بعضی دیگر تا حد زیادی به جنبه‌های خاص این سطح می‌پردازد. در شمای داخلی، انواع رکوردها، فایلها، صفات خاصه شاخص (استراتژی دستیابی)، نحوه نمایش و تشریح رکوردهای ذخیره شده در فایل، توالی رکوردها، تخصیص فضای ذخیره‌سازی برای داده‌ها، محل رکورد، فشردگی داده‌ای و تکنیکهای رمزگذاری داده‌ها تشریح می‌شوند. مباحث این سطح مربوط به درس ذخیره و بازیابی اطلاعات است و ما در این درس به این مسئله نمی‌پردازیم.

تذکر : سطح داخلی نزدیکترین سطح به رسانه ذخیره‌سازی فیزیکی است.

د) تبدیلات بین سطوح

در شکل استاندارد ANSI دو تبدیل وجود دارد : تبدیل ادراکی/داخلی و تبدیل خارجی / ادراکی عمل تبدیل را با توجه به آنچه که تبدیل می‌یابد می‌توان به سه دسته زیر تقسیم کرد.

۱ـ تبدیل داده‌ها : یعنی تبدیل داده‌های تعریف شده در سطح خارجی به داده‌های تعریف شده سطح ادراکی و بالاخره به داده‌های تعریف شده در سطح داخلی و نیز مسیر برعکس.

۲ـ تبدیل احکام : یعنی تبدیل حکم عمل کننده در سطح خارجی به حکم عمل کننده در سطح ادراکی و بالاخره به حکم یا احکامی در سطح داخلی

۳ـ تبدیل ساختار : یعنی تبدیل ساختار سطح خارجی به ساختار سطح ادراکی. مثلاً اگر ساختار داده‌یی در سطح ادراکی سلسله مراتبی و در سطح خارجی جدولی باشد می‌بایست تبدیل ساختار سلسله مراتبی به جدولی و برعکس را داشته باشیم به این سیستم‌ها دو ساختاری می‌گویند. البته اغلب سیستم‌های موجود یک ساختاری هستند. حال تبدیلات بین سطوح را بیان می‌کنیم :

تبدیل ادراکی/ داخلی : مثلاً اگر طراح بانک تعدادی جدول را طراحی کرده باشد، در تبدیل ادراکی به داخل برای هر جدول می‌توان فایلی تعریف کرد بصورتی که هر سطر جدول رکوردی از این فایل باشد. تغییرات در سطح داخلی بانک همیشه ممکن است بروز کند. اینگونه تغییرات نباید در دید ادارکی تاثیر داشته باشد. در تبدیل ادراکی/داخلی از سیستم عامل نیز کمک گرفته می‌شود.

تبدیل خارجی/ادراکی : در واقع این تبدیل مکانیسمی برای برقراری تناظر بین دیدهای خارجی مختلف و دید واحد ادراکی است. یک دید مشخص از یک کاربر خاص، بخشی است از دید واحد ادراکی و از نظر انواع موجودیتها، صفات خاصه هر موجودیت، نوع صفت و غیره لزوماً

همان نیست که در دید ادراکی از نظر طراح وجود دارد. مثلاً ممکن است یک صفت خاصه از یک موجودیت، از دید یک کاربر ترکیبی از چند صفت خاصه از سطح ادراکی باشد همچنین یک کاربر می‌تواند چندین دید داشته باشد. تبدیل خارجی/ادراکی مسائل فوق را حل می‌کند. اسامی فیلدها و رکوردها که به صورت تغییرپذیر قرار دارند، در این سطح دیده می‌شوند.

هـ) زبان میزبان (HL) و زبان فرعي داده‌يي (DSL)

منظور از زبان میزبان یکی از زبانهای سطح بالای برنامه‌سازی مثل کوبول، C,PL/1، پاسال، بیسیک، Delphi، Visual C، Visual Basic، Java می‌باشد.

زبان DSL زبانی است از سطح بالاتر که میهمان یک زبان سطح بالا مثل Visual C می‌شود هر مدل داده‌یی خاص (مثل سلسله مراتبی، شبکه‌ای، رابطه‌ای) زبان فرعی خاص خود را دارد. تعداد احکام این زبانها معمولاً کم است. برای هر سطح از معماری دستوراتی وجود دارد موسوم به : زبان فرعی داده‌یی خارجی، زبان فرعی داده‌یی ادراکی و زبان فرعی داده‌یی داخلی.

احکام زبان DSL را می‌توان به سه دسته زیر تقسیم کرد :

۱ـ احکام تعریف داده‌ها (Data Definition Language = DDL)

۲ـ احکام کار با (پردازش) داده‌ها (Data Manipulation Language = DML)

۳ـ احکام کنترلی (Data Control Language = DCL)

به طور کلی دو دسته زبان داده‌یی وجود دارد یکی زبان داده‌یی نامستقل یا ادغام شده (Embedded) و دیگری زبان داده‌یی مستقل. در نوع نامستقل DSL حتماً باید میهمان یک زبان سطح بالا باشد مثل SQL که در دلفی یا ویژوال بیسیک استفاده می‌شود یا Btrieve که

زبان فرعی داده‌ای برای C یا پاسکال است. در نوع مستقل DSL نیازی به زبان میزبان ندارد مثلاً Access و Foxpro نیازی به زبان میزبان ندارند.

البته SQL هم به صورت مستقل و هم به صورت نامستقل وجود دارد.

چند نکته در رابطه با DSL

ـ هر چه جنبه انتزاعی بودن سطح خارجی و ادراکی بانک قوی‌تر باشد، DSL منتزع از مفاهیم فایل‌پردازی خواهد بود.

ـ هر DSL فقط برای یک مدل داده‌یی مشخص طراحی می‌شود.

ـ مجموعه احکام DSL باید حداقل باشد.

ـ اصل وحدت عملگرها باید در DSL رعایت شود یعنی برای انجام یک عمل مشخص در سطوح خارجی و ادراکی حکم واحدی وجود داشته باشد.

ـ اصل وحدت عملگرها در یک سطح مشخص نیز مطرح است مثلاً هم برای درج داده‌ها و هم برای درج ارتباط بین آنها می‌بایست یک حکم واحد وجود داشته باشد.

ـ DSL باید داده‌های مختلف کاربران (مثل بردارها، ماتریسها، رشته‌ها و ...) را نیز بپذیرد.

تذکر : DSL یک زبان بیانی (Declarative) است که در آن کاربر می‌گوید چه می‌خواهد ولی رویه انجام کار را بیان نمی‌کند، برعکس و C و پاسکال که رویه‌ای (Procedural) هستند و کاربر باید رویه انجام کار را بیان کند.

کاربران بانك اطلاعاتي

کاربران بانک اطلاعاتی را می‌توان به سه دسته تقسیم کرد : ۱ـ برنامه‌نویسان کاربردی (DBP). برنامه‌ها می‌توانند به صورت دسته‌ای (Batch) و یا به صورت پیوسته (Online) باشند که کاربران می‌توانند از یک ترمینال Online به آن دستیابی داشته باشند. ۲ـ کاربران نهایی که از طریق یکی از برنامه‌های کاربردی Online می‌توانند به بانک اطلاعاتی دستیابی داشته باشند. ۳ـ مدیر بانک (DBA)

تذکر : کاربران نهایی (End Users) همچنین می‌توانند از طریق «پردازنده زبان تقاضا» مثل SQL که بخشی از نرم‌افزار سیستم بانک اطلاعاتی است به بانک دسترسی داشته باشند.

تفاوت DA با DBA

مدیر داده‌ها یا (Data Administrator) شخصی است که کنترل مرکزی داده‌ها را در سازمان به عهده دارد. این فرد لازم است مفهوم داده‌ها را درک کند و نیاز موسسه به داده‌ها را در سطح مدیریت عالی قرار دهد.

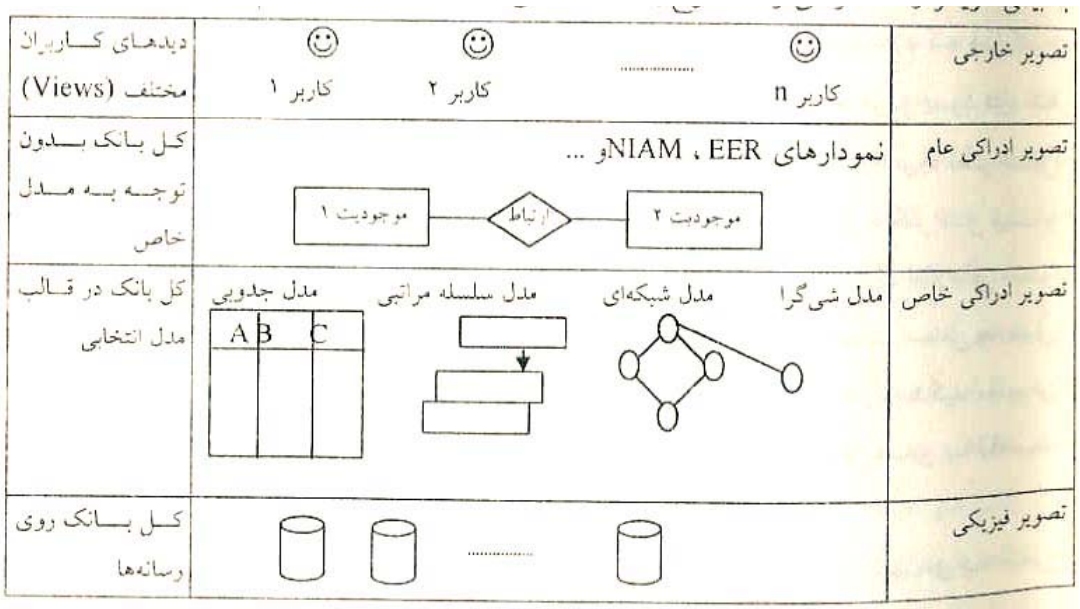
مدیر داده‌ها تصمیم می‌گیرد که چه داده‌هایی از همان اول در بانک اطلاعاتی قرار گیرد و پس از ذخیره آنها، سیاستهایی را برای دستیابی به آنها تنظیم کند. توجه کنید که مسئول داده‌ها یک مدیر است نه یک نفر فنی.

مدیر بانک اطلاعاتی یا (DataBase Administrator) یک شخص فنی است که مسئول پیاده‌سازی تصمیمات مدیر داده‌هاست. DBA برخلاف DA، یک فرد حرفه‌ای در تکنولوژی اطلاعات (IT) می‌باشد. وظیفه DBA ایجاد بانک اطلاعاتی و پیاده‌یازی کنترلهای فنی است که سیاستگذاری مدیر داده (DA) را اعمال کند. همچنین DBA می‌بایست تضمین کند که

سیستم با کارآیی قابل قبولی کار کند. DBA مجموعه‌ای از برنامه‌نویسان و سایر افراد فنی را در اختیار دارد.

بیانی دیگر برای معماری بانک اطلاعاتی

به بیانی گویاتر و ساده‌تر می‌توان سطوح بانک اطلاعاتی را به صورت زیر ترسیم کرد :



تصویر ادراکی عام و ادراکی خاص مربوط به به طراح بانک است. تصویر ادراکی عام فقط در مرحله طراحی مطرح می‌شود. پس از طراحی نهایی بانک و انتخاب یک مدل برای پیاده‌سازی آن این لایه به تصویر ادراکی خاص تبدیل می‌شود. در واقع در یک بانک اطلاعات که در مرحله بهره‌برداری است. تصویر ادراکی عام فقط در مستندات آن وجود دارد.

در معماری بانک اطلاعات، کلمه تصویر یا شما (Shema) مترادف لایه می‌باشد. مثلاً لایه خارجی همان مفهوم تصویر خارجی را دارد. مجموعه ساختارهای طراحی شده در یک بانک

بدون توجه به داده‌هایی که در آنها قرار می‌گیرند شمای بانک اطلاعاتی نام دارند. مثلاً در مدل رابطه‌ای،شمای یک بانک را جداول تشکیل می‌دهند یا مثلاً نوع داده هر ستون به شمای بانک مربوط می‌شود ولی تعداد سطرهای موجود در جدول ربطی به شمای بانک ندارد.

سه سطح معماري بانك اطلاعاتي

سه سطح معماری بانک اطلاعاتی عبارت است از :

۱ـ سطح داخلی یا سطح فیزیکی

۲ـ سطح خارجی که سطح منطقی کاربر نیز نامیده می‌شود.

۳ـ سطح مفهومی یا ادراکی، که سطح منطقی اجتماع یا سطح منطقی نیز نامیده می‌شود.

سطح خارجی با دیدگاه‌های انفرادی کاربران سر و کار دارد، در حالیکه سطح مفهومی با دیدگاه‌های اجتماعی کاربران سر و کار دارد. به عبارت دیگر، چندین دیدگاه خارجی وجود دارد که هر کدام حاوی نمایش انتزاعی کل بانک است. منظور از انتزاعی این است که نمایش موردنظر شامل ساختارهایی مثل رکوردها و فیلدهاست که کاربرگرا است، برخلاف ساختمانهایی دیگر مثل بیت و بایت که ماشین‌گرا هستند.

فقط یک دیدگاه داخلی وجود دارد که نمایش فیزیکی بانک اطلاعاتی است. عناصر داده‌ۀا در نقاط مختلف بانک (سطوح خارجی، ادرای و داخلی) ممکن است اسامی مختلفی داشته باشند. مثلاً شماره یک کارمند در دیدگاه خارجی می‌تواند EMPNO، در دیدگاه مفهومی EMPLOYEE-NO و در دیدگاه داخلی #EMP باشد. یکی از وظایف واحد تطابق یا نگاشت (Mapping) منطبق ساختن این اسامی مختلف است.

تذکر ۱ : در سیستم رابطه‌ای سطح مفهومی کاملاً رابطه‌ای است، دیدگاه خارجی نیز رابطه‌ای یا خیلی نزدیک به آن است ولی سطح داخلی رابطه‌ای نیست.

در دیدگاه مفهومی به داده‌ها به همان شکلی که هستند نگاه می‌شود، نه به شکلی که کاربران به دلیل محدودیت‌های موجود در یک زبان یا سخت‌افزار مجبور به دیدن آنها می‌شوند.

تذکر ۲ : دیدگاه داخلی بالاتر از سطح فیزیکی است زیرا با اصطلاح رکوردهای فیزیکی که بلوکها یا صفحات نیز نامیده می‌شوند، سر و کار ندارد. همچنین با ملاحظات دستگاهها مثل اندازه سیلندرها و شیارها نیز سر و کار ندارد. به عبارت دیگر، دیدگاه داخلی، فضای خطی نامحدودی را فرض می‌کند. جزئیات چگونگی نگاشت فضای آدرس به حافظه فیزیکی کاملاً وابسته به سیستم است و از معماری کلی حذف شده است.

دیدگاه داخلی به وسیله شمای داخلی توصیف می‌شود که نه تنها انواع رکورد ذخیره شده را تعریف می‌کند بلکه مشخص می‌کند چه اندیسهایی وجود دارد، فیلدهای ذخیره شده چگونه نمایش داده می‌شوند، ترتیب فیزیکی ذخیره رکوردها چگونه است و غیره. شمای داخلی با استفاده از یک زبان تعریف داده‌ها به نام DDL داخلی نوشته می‌شود.

تذکر ۳ : در اغلب سیستم‌ها می‌توان تعریف بعضی از دیدگاههای خارجی را برحسب دیدگاههای دیگر از طریق نگاشتهای خارجی/داخلی بیان کرد.

لغت‌نامه داده‌ها و کاتالوگ سیستم

لغتنامه داده‌ها (Data Dictionary) شبیه لغتنامه‌های معمولی، تمامی اسامی استفاده شده در سیستم و معنای آنها را در بر می‌گیرد. در مرحله طراحی بانک اطلاعات هرگاه طراح برای

مفهومی نامی انتخاب می‌کند، باید آن را در لغتنامه داده‌ها همراه با معنای آن و فرمت آن وارد کند.

این اسامی شامل تمامی نامهای جداول، شیء‌ها، صفتها و غیره است. در بانکهای جدید نرم‌افزار ویژه‌ای برای کار با لغتنامه‌ها وجود دارد که به کمک آن می‌توان اسامی را وارد یا جستجو کرد. این نرم‌افزارها از اشتباهاتی نظیر وارد کردن یک نام با دو معنای مختلف (Homonym) و یا دو نام برای یک مفهوم (Synonym) جلوگیری می‌کنند.

علاوه بر اسامی داده‌ها اطلاعات دیگری باید در مورد بانک نگهداری شود مثل اطلاعات مربوط به حق دستیابی افراد به داده‌های مختلف، تاریخ ایجاد و یا تغییر داده‌ها، تعداد نسخه‌های هر پرونده، اندازه هر جدول یا شیء و غیره. اینگونه اطلاعات در کاتالوگ سیستم (System Catalog) نگهداری می‌شوند. در واقع لغتنامه داده‌ها زیر مجموعه کاتالوگ سیستم است ولی بدلیل کاربرد ویژه آن مجزا شده و برای کار با آن، نرم‌افزار خاصی طراحی شده است. سیستم DBMS به طور خودکار و به کمک کاربران اطلاعات موجود در کاتالوگ سیستم را همواره به روز نگه می‌دارد.

اطلاعات موجود در دیکشنری داده‌ها اصطلاحاً فراداده یا دادگان (Meta Data) می‌گویند که به معنی داده در مورد داده است.

برای مدیریت دیکشنری داده‌ها نیاز به نرم‌افزار خاصی به نام سیستم مدیریت دیکشنری بانک (Data Dictionary Management System) است. این سیستم می‌تواند مستقل از سیستم مدیریت بانک باشد که مزیت استقلال از سیستم مدیریت بانک را دارد ولی وجود افزونگی یکی از ایرادات آن است. همچنین این سیستم می‌تواند در سیستم مدیریت بانک ادغام شده باشد. دیکشنری داده‌ها خود باید به صورت یک بانک طراحی شود و دیکشنری اساساً باید بانک باشد. در واقع دیکشنری داده‌ها امکانی است برای کنترل و نگاهداری بانک و در صورت لزوم توسعه دادن آن در طول حیات بانک.

مطالبی که در کاتالوگ سیستم ذخیره می‌شوند عبارتند از :

ـ نام ساختارهای داده‌یی مثلاً نام جدولها در مدل بانک رابطه‌ای

ـ نام موجودیتها و ارتباطات بین آنها

ـ نام صفات خاصه هر نوع موجودیت، نوع و محدوده مقادیر آنها

ـ شماهای خارجی و ادراکی و رویه‌های تبدیل بین سطوح مختلف و نیز شمای داخلی

ـ مشخصات کاربران و چگونگی حق دستیابی آنها به داده‌ها و محدوده مجاز عملیات آنها

ـ مشخصات سیستمی پایانه‌های متصل به بانک

ـ تراکنش‌هایی که باید روی بانک انجام شود. تراکنش را جلوتر شرح می‌دهیم.

ـ مشخصات گزارشاتی که باید از بانک گرفته شوند.

ـ واحدهای اندازه‌گیری

ـ تاریخ ایجاد داده‌ها، مکانیسم ورود داده‌ها، به بانک و چگونگی استفاده از آنها

ـ ارتباط بین برنامه‌های کاربردی و داده‌ها یعنی چه برنامه‌هایی از چه داده‌هایی استفاده می‌کنند.

تذکر : در کاتالوگ سیستم می‌توان عملیات جستجو، درج و حذف را انجام داد و استفاده از کاتالوگ سیستم باعث افزایش استقلال داده‌ای می‌شود.

امنیت و جامعیت

امنیت یا Security به معنای محافظت در برابر خطراتی از قبیل آتش‌سوزی و نیز جلوگیری از دستیابی غیرمجاز آنهاست. راه‌های مختلفی برای جلوگیری از دستیابی غیرمجاز به داده‌ها مثل استفاده از رمز عبور (Password) وجود دارد ولی همواره ممکن است افرادی پیدا شوند و این رمزها را بگشایند. جامعیت (Integrity) به معنای صحت داده‌ها و پردازش‌ها و پیروی از مقررات سیستم است. مثلاً موجودی واقعی حسابهای بانکی نباید منفی باشد و یا شخص نتواند بیش از موجودی خود از حساب برداشت کند.

تراکنش (Transaction)

هر برنامه‌ای که توسط کاربر در محیط بانک اطلاعاتی اجرا می‌شود تراکنش نام دارد. تراکنش یک واحد منطقی از کار است و معمولاً شامل چندین عمل بانک اطلاعاتی است. تفاوت اصلی یک تراکنش با یک برنامه معمولی در محیط غیربانکی این است که تراکنش همواره به DBMS تسلیم می‌شود و DBMS در اعمال هرگونه کنترل و حتی به تعویق انداختن و ساقط کردن آن آزادی عمل دارد.

هدف اصلی این کنترلها حفظ جامعیت و صحت بانک اطلاعاتی است. چرا که در بانک اطلاعاتی آنچه در درجه اول اهمیت دارد داده است نه برنامه. داده‌های بانک اطلاعاتی را مانا (Persistent) می‌نامند زیرا برنامه‌ها می‌آیند و می‌روند اما داده‌ها می‌مانند.

مثلاً برنامه‌ای که پولی را به حسابی می‌ریزد یا برداشت می‌کند آنقدرها مهم نیست. مهم این است که موجودی حسابها اشتباه نشود.

تضمین جامعیت بانك اطلاعاتي

آقای جیم‌گری (Jim Gray) در سال ۱۹۸۱ ثابت کرد که چهار کنترل زیر لازم است روی تمامی تراکنش‌ها در بانک اطلاعات اعمال گردد تا صحت و جامعیت آن تضمین شود این کنترلها به خواص ACID معروفند.

۱ـ یکپارچگی (Atomicity) این خاصیت به همه یا هیچ موسوم است. منظور این است که یا تمام دستورات یک تراکنش باید اجراء شود یا هیچکدام از آنها نباید اجراء شود. مثلاً تراکنشی می‌خواهد مبلغی را از حسابی به حساب دیگر منتقل کند. فرض کنید بخش اول کار (برداشت پول) در یک ماشین و بخش دوم کار (واریز پول) در ماشینی دیگر اجراء می‌شود. حال در نظر بگیرید پس از انجام بخش اول (برداشت پول) ارتباط با ماشین دوم ناگهان قطع شود بدیهی است که در این حالت باید پول برداشت شده دوباره به همان حساب اول بازگردانده شود.

۲ـ همخوانی (Comsistency) این خاصیت به این صورت بیان می‌گردد که :«هر تراکنش اگر به تنهایی اجراء شود بانک اطلاعات را از حالتی صحیح به حالت صحیح دیگری منتقل می‌کند»

یعنی این تراکنش ممکن است دو نوع پایان داشته باشد : الف) پایان ناموفق که آنرا سقوط (Abort) می‌نامند ب) پایان موفق که آنرا انجام (Commit) می‌نامند.

۳ـ انزوا (Isolation) در بانک اطلاعاتی ممکن است تراکنش‌های همروند وجود داشته باشد (مثل Multitasking در سیستم عامل Windows که چند برنامه همزمان اجراء می‌شوند).

بر طبق خاصیت انزوا همروندی تراکنش‌ها باید کنترل شود تا اثر مخرب بر روی هم نداشته باشند به عبارتی دیگر اثر تراکنش‌های همروند روی یکدیگر چنان است که گویا هر کدام در انزوا انجام می‌شود. این کنترل توسط بخشی از DBMS به نام واحد کنترل همروندی (Consurrency Control) انجام می‌شد. کتاب دیت مفهوم Isolate را به صورت زیر بیان می‌کند :

Isolate یعنی به هنگام‌سازی حاصل از تراکنش TI توسط تراکنش دیگری مثل T2 قابل مشاهده نیست. مگر اینکه عمل COMMIT را اجراءکند. COMMIT موجب می‌شود تا به هنگام‌سازیهایی که توسط یک تراکنش انجام شد، توسط تراکنش‌های دیگر قابل رویت باشد. اگر تراکنش ROLLBACK را اجراءکند تمام به هنگام‌سازیهایی که انجام شده از بین می‌روند.

۴ـ پایائی (Durability) براساس این خاصیت تراکنش‌هایی که به مرحله انجام (Commit) برسند اثرشان ماندنی است و هرگز به طور تصادفی از بین نمی‌رود. مثلاً اگر مبلغی به حسابی واریز شود تراکنش مربوطه انجام یافته اعلام شود حتی در صورت وقوع آتش‌سوزی در آن شعبه بانک، مشتری نباید متضرر شود، یعنی مثلاً عمل واریز قبل از اعلام انجام موفق باید در جای دیگری نیز ثبت شده باشد.

تذکر ۱ : دو عمل یکپارچگی و پایانی توسط واحدی از DBMS به نام واحد مدیریت باز گرد (Recovery Management) انجام می‌گیرد.

تذکر ۲ : در تراکنشها باید تضمین شود که اجرای ناپیوسته (Interleaved) مجموعه‌ای از تراکنش‌های همزمان (معمولاً) به صورت سریال انجام شود. یعنی اجرای سریال آنها همان نتایجی را دارد، که همان تراکنش‌ها به ترتیب نامشخصی اجراء می‌شوند.

تذکر ۳ : تراکنش با اجرای Begin Transaction شروع می‌شد و با اجرای Commit و Rollback خاتمه می‌یابد. تراکنش‌ها اتمی، پایدار و مجزا از یکدیگر هستند. اجرای ناپیوسته مجموعه‌ای از تراکنش‌های همزمان، تضمین می‌کند که به صورت سریال باشند.

استقلال داده‌ها

منظور از استقلال داده‌ها مستقل بودن ذخیره‌سازی داده‌ها از کاربرد آنهاست. مثلاً مدل رابطه‌ای تجریدی به نام جدول استفاده می‌کند و داده‌ها هر چه باشند در قالب چند جدول ریخته می‌شوند. نحوهٔ ذخیره‌سازی داده‌ها روی رسانه‌ها از دید کاربران مخفی است.

استقلال داده‌ها به دو صورت فیزیکی و منطقی تعبیر می‌شود.

در استقلال فیزیکی داده‌ها (Physical Data Independence) اگر تغییری در ذخیره‌سازی داده‌ها انجام گیرد (مثلاًنوع دیسک عوض شود) برنامه‌های کاربردی هیچ تغییری نمی‌کنند. در استقلال منطقی داده‌ها (Logical Data Independence) تغییر تصویر ادراکی بانک از دید کاربران و برنامه‌های آنها مخفی می‌ماند. مثلاً اگر جدولی چهار ستون داشته و برنامه‌هایی روی آن ستونها نوشته شده، در صورتی که ستون پنجمی به آن اضافه شود برنامه‌های سابق نیاز به دستکاری ندارند و با همان شکل قبلی قابل اجراء هستند.

همچنین ایجاد جدولهای جدید (برای نمایش موجودیتی جدید یا ارتباطی جدید بین موجودیتهای قدیم) نشان‌دهنده رشد بانک در سطح ادراکی است که نباید روی برنامه‌های کاربردی و دید خارجی کاربران تاثیر داشته باشد.

یا مثلاً طراح ممکن است تصمیم بگیرد جدولی را در سطح ادراکی به دو جدول تقسیم کند این عمل نیز نباید اثری بر دید کاربران داشته باشد. همچنین تغییر در نوع صفات خاصه، تغییر در اندازه فیلدها و تغییر در واحدهای اندازه‌گیری نباید اثری در دید کاربران داشته باشد.

تذکر : نگاشت مفهومی / داخلی (در معماری ANSI) کلید استقلال فیزیکی داده‌هاست و نگاشتهای خارجی / مفهومی کلید استقلال منطقی داده‌هاست. به عبارت دیگر در استقلال منطقی طراح در تغییرات سطوح ادراکی / خارجی آزاد است و این تغییرات شامل موجودیتها، صفات خاصه و ارتباط بین موجودیتهاست. در استقلال فیزیکی طراح در تغییرات سطح داخلی / ادراکی آزاد است و این تغییرات شامل ساختار فایلها، ساختارهای ذخیره‌سازی و دستگاه‌های ذخیره‌سازی متفاوت است.

CASE

بعضی از DBMSها دارای ابزار کمکی (Computer Aided Software Engineering) هستند که در طراحی و پیاده‌سازی بانک مورد استفاده قرار می‌گیرند. از جمله موارد استفاده این ابزارها می‌توان از رسم نمودار EER، کار با لغتنامه‌ها، تعریف شمای بانک، تهیه نمودارها و گزارش‌ها، طراحی شیءگرا و رسم نمودارهای آن و مهندسی وارون (Reverse Engineering) که گزارش طراحی را از سیستم پیاده شده استخراج می‌کند نام برد.

حتی ابزارهای ویژه‌ای عرضه شده‌اند که طراحی مفهومی عام مثل EER را گرفته و طراحی مفهومی خاص را در مدل رابطه‌ای تولید می‌کنند. مثلاً نرم‌افزار اورکل دارای چنین ابزاری است.

عدم وجود ناسازگاري در داده‌ها

ناسازگاری داده‌ها هنگامی بروز می‌کند که بنابر دلایلی یک فقره اطلاع در بیش از یک نقطه از بانک ذخیره شود و لازم باشد که بهنگام درآید. اگر عمل بهنگام سازی در تمام نقاطی که آن فقره اطلاع وجود دارد، توسط سیستم مدیریت بانک انجام نشود ناهمگونی در اطلاعات و به عبارت دیگر پدیده ناسازگای داده‌یی بروز می‌کند. در واقع نوعی افزونگی در بانک وجود داشته، سیستم مدیریت بانک از وجود آن آگاه نبوده و لذا عمل بهنگام سازی بطور کامل انجام نپذیرفته است. سیستم مدیریت بانک باید چنان عمل کند که در عین کاهش حتی‌الامکان میزان افزونگی‌ها، در صورت وجود افزونگی بهنگام‌سازی را بصورت منتشر شونده انجام دهد. افزونگی به خودی خود چندان پدیده نامطلوب نیست ولی افزونگی کنترل نشده پدیده‌ایست بسیار نامطلوب.

می‌توان گفت بروز هر یک از پدیده‌های : ناسازگاری داده‌ها، نادقیق بودن داده‌ها، نا ایمن بودن داده‌ها. عدم وجود ارتباطات بین موجودیتها (گسست پیوندهای سمانتیک)، کاهش کیفیت اطلاعات، بروز اشتباهات و اشکالات در بانک سبب خدشه‌دار شدن جامعیت و تمامیت بانک می‌شود.

مزاياي سيستم بانك اطلاعاتي

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

۱ـ مولینگ داده‌های عملیاتی براساس سمانتیک آنها

۲ـ وحدت ذخیره‌سازی کل داده‌های محیط عملیاتی

۳ـ اشتراکی شدن داده‌ها

۴ـ کاهش میزان افزونگی

۵ـ تعدد شیوه‌های دستیابی و تسهیل دستیابی به داده‌ها

۶ـ عدم وجود ناسازگاری در داده‌ها

۷ـ تامین سیستم کاراتر برای ذخیره و بازیابی

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

۹ـ امکان اعمال ضوابط دقیق ایمنی

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

۱۱ـ تامین استقلال داده‌یی

۱۲ـ حفظ محرمانگی داده‌ها

۱۳ـ امکان اعمال استانداردها

۱۴ـ ایجاد تعادل بین نیازهای حتی گاه متضاد کاربران

۱۵ـ تسهیل گسترش موارد کاربردی و رشدپذیری محیط ذخیره‌سازی

۱۶ـ تسریع در دریافت پاسخ پرس‌و جوها

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

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

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

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

۲۱ـ تعداد زبانهای میزبان

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

۲۳ـ کاهش هزینه‌های سازمان

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

۲۵ـ استانداردسازی امکان‌پذیر می‌شود.

۲۶ـ پشتیبانی از تراکنش امکان‌پذیر است. تراکنش (Transaction) یک واحد منطقی از کار است و معمولاً حاوی چند عمل بانک اطلاعاتی است. مثلاً انتقال مبلغی از حساب A به حساب B یک تراکنش است که از دو بهنگام‌سازی در حساب A , B تشکیل یافته است.

امتیازات سیستم بانک اطلاعاتی نسبت به سیستم سنتی که رکوردها بر روی کاغذ نگهداری می‌شوند عبارتند از :

ـ فشردگی : نیازی به فایلهای متنی حجیم نیست.

ـ سرعت : ماشین می‌تواند سریعتر از انسان داده‌ها را بازیابی و بهنگام‌سازی کند.

ـ بودجه کمتر : خیلی از یکنواختی‌ها در نگهداری فایلها به روش دستی حذف می‌شود. همواره کار مکانیکی توسط ماشین سریعتر از انسان انجام می‌گیرد.

ـ دسترسی : در هر زمان می‌توان اطلاعات دقیق و به هنگامی را دریافت کرد. سیستم بانک اطلاعاتی موجب می‌شود تا موسسه بر روی داده‌هایش کنترل مرکزی داشته باشد.

معایب سیستم بانك اطلاعاتي

ـ ممکن است امنیت لازم (بدون کنترل‌های مناسب) به مخاطره بیفتد چرا که داده‌ها متمرکز بوده و این تمرکز آنها را آسیب‌پذیر می‌سازد. به همین ترتیب بدون کنترل‌های مناسب ممکن است جامعیت داده‌ها نیز به خطر بیفتد.

ـ ممکن است سخت‌افزار اضافی نیاز باشد.

ـ بالاسری اجرای DBMS ممکن است دارای اهمیت باشد.

ـ سیستم و عملیات موفقیت‌آمیز تا حدودی پیچیده خواهد بود (البته این پیچیدگی از دید کاربر مخفی است)

تذکر ۱ : دیتابیس بیان کننده موجودیتها، صفات خاصه و ارتباط منطقی بین موجودیتها می‌باشد.

تذکر ۲ : اجزای یک DBMS به طور کلی عبارتند از :

DML Processor – DDL Compiler – File Manager – Database Manager – Query Processor – System Buffer – Catolog Manager

کارها و وظایف DBA

۱ـ تعریف شمای مفهومی

۲ـ تعریف شمای داخلی (طراحی فیزیکی همواره بعد از طراحی منطقی انجام می‌شود)

۳ـ مرتبط بودن با کاربر، DBA باید با کاربران ارتباط برقرار کند تا اطمینان حاصل کند که داده‌های مورد نیاز آنها وجود دارد و شمای خارجی مورد نیاز را به کمک DDL خارجی بنویسد. سایر جنبه‌های ارتباط با کاربر عبارتند از : مشورت در طراحی برنامه‌های کاربردی، تهیه آموزشهای تکنیکهای شناسایی مساله و ارائه راه‌حلهای آن

۴ـ تعریف محدودیت‌های جامعیت و امنیتی

۵ـ تعریف سیاستهای ترمیم و پشتیبانی

۶ـ نظارت بر کارآیی و پاسخ به تغییر نیازها

تذکر : سیستم‌های VLDB (Very Large DataBase) سیستم‌های ترابایتی هستند که میلیاردها بایت از اطلاعات را ذخیره می‌کنند.

DBMS نرم‌افزاری است که تمام دستیابی‌ها به بانک اطلاعاتی را انجام می‌دهد.

وظایف DBMS

۱ـ تعریف داده‌ها

۲ـ دستکاری داده‌ها. به طور کلی درخواستهای DBMS می‌تواند «برنامه‌ریزی شده» یا «برنامه‌ریز نشده» باشد. درخواستهای برنامه‌ریزی شده درخواستی است که قبل از اجرای آن، پیش‌بینی شده باشد. احتمالاً DBMS طراحی بانک اطلاعاتی فیزیکی را طوری تغییر می‌دهد که کارآیی درخواستهای برنامه‌ریزی شده بالا باشد. درخواستهای برنامه‌ریزی نشده یا موردی، درخواستی است که از قبل پیش‌بینی نشده است و در صورت نیاز ارائه می‌شود.

طراحی بانک اطلاعاتی فیزیکی ممکن است برای پاسخ به درخواست موردنیاز ایده‌آل باشد یا نباشد. درخواستهای برنامه‌ریزی نشده معمولاً به صورت محاوره‌ای صادر می‌شوند.

۳ـ بهینه‌سازی و اجراء

۴ـ جامعیت و امنیت داده‌ها. بررسی درخواستهای کاربر از نظر درستی می‌تواند در زمان کامپایل، زمان اجراء یا هر دو زمان انجام شود.

۵ـ ترمیم و سازگاری داده‌ها

۶ـ فرهنگ داده‌ها : فرهنگ داده حاوی داده‌هایی راجع به داده‌هاست که گاهی شبه‌داده‌ها یا توصیفات نامیده می‌شوند. یعنی سایر اشیای سیستم را تعریف می‌کنند. نام دیگر آن در بعضی کتابها کاتالوگ سیستم، فهرست راهنما، مخزن داده‌ها و دایره‌المعارف داده‌ها می‌باشد.

۷ـ کارآیی

تذکر : هدف DBMS آماده کردن رابط کاربران با سیستم بانک اطلاعاتی است. رابط کاربر را می‌توان به عنوان مرزی در نظر گرفت که هر آنچه در زیر آن قرار دارد، در دید کاربر نیست.

مدیر فایل

مدیر فایل بخشی از سیستم عامل است که فایلهای ذخیره شده را مدیریت می‌کند، بنابراین نسبت به DBMS به دیسک نزدیکتر است. در واقع DBMS بر روی بعضی از انواع مدیر فایل ساخته می‌شود.

ـ مدیر فایل از ساختار داخلی رکوردهای ذخیره شده اطلاع ندارد و بنابراین نمی‌تواند درخواستهایی را که براساس آگاهی از آن ساختار تنظیم می‌شوند، پاسخ دهد.

ـ محدودیتهای جامعیت و امنیت را چندان پشتیبانی نمی‌کند.

ـ کنترلهای ترمیم و سازگاری را انجام نمی‌دهد یا بسیار کم انجام می‌دهد.

ـ در سطح مدیر فایل فرهنگ داده واقعی وجود ندارد.

ـ استقلال داده‌ها را نسبت به DBMS به ندرت فراهم می‌کند.

ـ فایلها «مجتمع» یا «مشترک» نیستند. یعنی به کاربر و یا برنامه کاربردی خاصی تعلق دارند.

مدیر ارتباطات داده‌ها یا مدیر DC

درخواستهای کاربر نهایی از بانک اطلاعاتی از ایستگاه کاربر (که ممکن است از نظر فیزیکی دو از سیستم بانک اطلاعاتی باشد) به برنامه‌های کاربردی پیوسته (درونی یا غیره) فرستاده می‌شود و نتیجه به شکل پیام‌های ارتباطی به DBMS می‌رسند.

پاسخهای DBMS و برنامه‌های کاربردی به ایستگاه‌های کاری کاربران نیز به همین صورت ارسال می‌شوند. ارسال این پیام‌ها تحت کنترل بخشی از نرم‌افزار به نام مدیر ارتباطات داده‌ها (مدیر DC) انجام می‌شود.

مدیر ارتباطات داده‌ها، بخشی از DBMS نیست بلکه یک سیستم مستقل است اما از آنجا که DC باید با DBMS کار کند به عنوان بخشی از سیستم بالاتر به نام «سیستم ارتباطات داده‌ای / بانک اطلاعاتی» (DB/DC) محسوب می‌شوند که در آن DBMS از بانک اطلاعاتی مراقبت می‌کند و مدیر DC تمام پیام‌های ورودی ـ خروجی DBMS را پردازش می‌کند.

معماري مشتري / کارگزار و پردازش توزیع شده

سیستم بانک اطلاعاتی دارای ساختار دو بخشی است که شامل کارگزار (Server) و مشتری (Client) است. سطوح گوناگونی از پردازش توزیع شده امکان‌پذیر است. یک حالت ساده اجرای کارگزار DBMS در یک ماشین و اجرای مشتری در ماشین دیگر است.

چند ماشین مختلف مشتری می‌تواند به یک ماشین کارگزار دسترسی داشته باشند. همچنین یک ماشین مشتری ممکن است بتواند به چند ماشین کارگزار دستیابی داشته باشد، این دستیابی ممکن است به چند روش انجام شود :

الف) یک مشتری می‌تواند هر چند تا کارگزار دستیابی داشته باشد اما در هر زمان به یکی از آنها دسترسی دارد. در چنین سیستمی، در یک درخواست نمی‌توان داده‌هایی از دو یا چند کارگزار مختلف را ترکیب کرد. علاوه بر این در چنین سیستمی باید بداند چه ماشینی چه اطلاعاتی را دارد.

ب) مشتری ممکن است قادر باشد، همزمان به چند کارگزار دستیابی داشته باشد یعنی یک درخواست بانک اطلاعاتی می‌تواند داده‌هایی از چند کارگزار را ترکیب کند. در این حال کاربر لازم نیست بداند که کدام ماشین کدام داده‌ها را در اختیار دارد. این حالت سیستمی را به نام «سیستم بانک اطلاعاتی توزیع شده» ایجاد می‌کند.

یک برنامه کاربردی باید بتواند بر روی داده‌هایی که در چندین بانک اطلاعاتی وجود دارند به طور شفاف عمل کند. معنای شفاف این است که از دیدگاه منطقی، برنامه کاربردی طوری عمل می‌کند که گویی داده‌هایی که توسط یک DBMS مدیریت می‌شود، بر روی یک ماشین وجود دارد.

تذکر : اصطلاح پردازش موازی نیز گاهی به معنای پردازش توزیع شده به کار می‌رود. با این تفاوت که ماشین‌های توزیع شده در سیستم موازی باید از نظر فیزیکی در کنار هم باشند.

برنامه‌هاي سودمند

برخی از برنامه‌های سودمند عبارتند از :

ـ روالهای بار کردن برای ایجاد نسخه اولیه بانک اطلاعاتی از یک یا چند فایل

ـ روالهای بار کردن مجدد و ذخیره‌سازی برای بار کردن دوباره بانک اطلاعاتی و کپی کردن از پشتیبان

ـ روالهای سازماندهی مجدد مثلاً برای استفاده مجدد از فضاهایی که داده‌های بلااستفاده در آن ذخیره شده است.

ـ روالهای آماری برای محاسبه آمارهای کارایی مثل اندازه فایلها یا شمارش عملیات I/O و غیره

سه مقوله اصلي DBMS

عبارتند از :

۱ـ پایگاه داده‌ها (Data Base) شامل فایلها و نحوه دستیابی و عملیات بر روی فایلها است.

۲ـ سیستم ارتباطات که عهده‌دار تبادل بین کاربران و سیستم می‌باشد و کنترل‌های لازم را روی پیام‌ها و نمایش‌های خروجی انجام می‌دهد.

۳ـ سیستم مدیریت تراکنش‌ها که کنترل دستیابی به فایلها، برنامه‌ریزی کارهای کاربران، زمانبندی کارها و پیاده‌سازی روشهای نگهداری سیستم را برعهده دارد.

مدیریت منابع اطلاعاتي در DBMS

امروزه تکنولوژی اطلاعات در بانک اطلاعاتی نقش گسترده‌ای دارد و این وظیفه به عهده مدیریت تابع اطلاعاتی یا (Information Resource Management) می‌باشد. IRM شامل فعالیتهای مرتبط با پردازش داده‌ها، جریان تبادل داده‌ها و خودکار کردن کارهای اداری می‌باشد. فعالیتهای IRM سه دسته تقسیم می‌شود. الف) پردازش داده‌ها ب) ارتباط مکانیزه ج) ترکیب (یعنی بین سیستم‌های اطلاعاتی و پاسخگویی به کاربر ارتباط نزدیک و تنگاتنگی وجود دارد).

فصل چهارم

ساختارهاي داده‌يي

به کمک ساختار داده‌یی، مهمترین سطح بانک یعنی سطح ادراکی تعریف و تشریح می‌شود، سطحی که حالت انتزاعی دارد و مستقل از مفاهیم فایلینگ باید شکل گرفته و معرفی شود. بعضی از مولفین دو اصطلاح ساختار داده‌یی و مدل داده‌یی را یکسان می‌گیرند ولی بعضی دیگر ساختار داده‌یی را بخشی از مدل داده‌یی را می‌دانند و به عبارتی دیگر عناصر مدل داده‌یی عبارتند از : ۱ـ ساختار داده‌یی ۲ـ عملگرهای عمل کننده روی ساختار ۳ـ قواعد عام برای تامین جامعیت

با آنکه ساختار داده‌یی متعددی برای طراحی سطوح ادراکی و خارجی بانک وجود دارد ما در این فصل سه ساختار رابطه‌ای، سلسله مراتبی و شبکه‌ای را شرح می‌دهیم. البته امروزه ساختار شی‌گرا نیز معروفیت زیادی یافته است که در فرصتی دیگر آن را بیان خواهیم کرد.

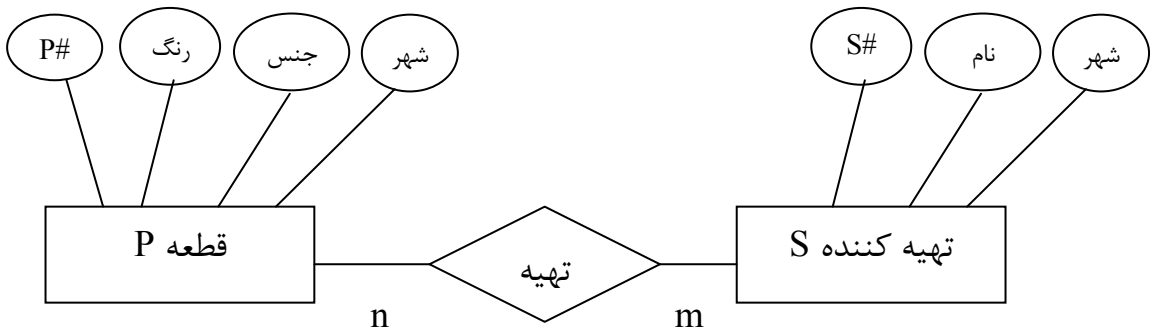
همانطور که قبلاً گفتیم بسته به نوع ساختار داده‌یی، زبان تعریف داده‌ها و زبان کار با داده‌ها متفاوت است.

۱ـ ساختار رابطه‌اي

از دید کاربر بانک اطلاعاتی تشکیل شده است از تعدادی جدول. جدول ساختاری است نامدار که از تعدادی سطر و ستون تشکیل یافته است. هر ستون نمایشگر یک صفت خاصه از یک نوع موجودیت است و هر سطر نمایشگر یک نمونه از یک نوع موجودیت می‌باشد.

می‌توان تصور کرد که یک جدول شبیه یک فایل ترتیبی مسطح است و بانک رابطه‌ای مجموعه‌ای از همین فایلهاست. از آنجا که رابطه با جدول معادل است فعلاً بانک رابطه‌ای را می‌توانیم بانک جدولی نیز بنامیم.

مثال ۱ : موجودیتهای قطعه و تولید کننده را در نظر گرفته و نمودار EER آن را ترسیم می‌کنیم.



برای تشریح نمودار فوق در بانک رابطه‌ای یک جدول برای هر یک از دو موجودیت و جدولی نیز برای بیان ارتباط آنها، استفاده می‌شود :

S#	P#	تعداد Qty
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

جدول SP (محموله)

P#	رنگ	جنس	City
P1	قرمز	آهن	تهران
P2	سبز	مس	تبریز
P3	آبی	برنج	شیراز

جدول P (قطعه)

S#	نام تهیه کننده	City
S1	فن‌آوران	تهران
S2	ایران قطعه	تبریز

جدول S (تهیه کننده)

می‌بینیم که در مدل رابطه‌ای از ساختار جدول هم برای نمایش موجودیتها استفاده می‌شود هم برای نمایش ارتباط بین آنها و این یکی از مزایای مدل رابطه‌آی است که در مدلهای دیگر (سلسله مراتبی شبکه‌ای) وجود ندارد.

در این مدل جدولها از یکدیگر مجزا هستند. مثلاً برای پیمایش جدول SP نیازی به پیمایش جدول S و سپس رفتن از این جدول به جدول SP نیست. این مزیت نیز در مدلهای دیگر وجود ندارد.

یکی از مزایای مهم مدل رابطه‌ای سادگی زیاد و درک راحت این ساختار است. همچنین این مدل پشتوانه تئوری ریاضی و قوی برخوردار می‌باشد.

بررسي رفتار مدل رابطه‌اي

برای بازیابی در این مدل به عملگر سطریاب نیاز داریم.

برای بازیابی یک سطر کافی است نام جدول و شرط یا شرایط موردنظر پرس‌وجو را بدهیم.

مثال ۲ (QI) : «شماره تهیه کنندگان قطعه P2 را بیابید»

ابتدا باید مشخص کنیم در کدام جدول جستجو صورت گیرد که در مثال فوق جدول SP می‌باشد. رویهٔ پاسخگویی به سئوال فوق به صورت زیر است :

سطرهای جدول SP را از ابتدا تا انتها نگاه می‌کنیم. در سطری که #P آن برابر ''P2'' می‌باشد، #S را چاپ می کنیم.

جدول SP جدول محموله (Shipment) است. خروجی دستور فوق S3, S2, S1 خواهد بود.

البته این تصور نباید پیش بیاید که برای جستجوی در بانک فیزیکی (در فایل) لزوماً باید پیمایش ترتیبی رکوردهای فایل ذخیره شده انجام شود. نحوه جستجو و بازیابی سطر در محیط فیزیکی به ساختار ذخیره‌سازی فایلها در محیط فیزیکی بستگی دارد.

حال قرینه پرس و جوی قبلی را درنظر می‌گیریم :

مثال ۳ (Q2) : «شماره قطعات تهیه شده توسط S2 را بیابید»

رویه پاسخگو به این پرس و جو (Query) به صورت زیر است :

سطرهای جدول SP را از ابتدا تا انتها نگاه می‌کنیم. در سطری که S# آن برابر "S2" می‌باشد، P# را چاپ می‌کنیم. می‌بینیم که این رویه، دقیقاً قرینه رویه پاسخگویی قبلی است. خروجی این دستور P1 , P2 می‌باشد.

نتیجه اینکه در مدل رابطه‌ای برای پاسخگویی به پرس و جوهای قرینه، رویه واحد وجود دارد و این نیز یکی دیگر از مزایای مدل رابطه‌آی است.

در عملیات ذخیره‌سازی (درج، حذف و بهنگام‌سازی) در واقع باید منطقاً سطری را در یک جدول درج کرد، از یک جدول حذف کرد و یا در یک جدول بهنگام کرد. عملیات ذخیره‌سازی در مدل رابطه‌ای در سطح انتزاعی به سهولت انجام می‌شوند و هیچگونه دشواری برای انجام این عملیات وجود ندارد و همچنین پس از انجام این عملیات وضعیت نامطلوبی رخ نمی‌دهد (مدلهای دیگر ممکن است اینگونه نباشند).

مثال ۴ (مثال از درج) : در جدولS این اطلاع را درج کنید «تهیه کننده جدید S4 با مشخصات نام آلومین و شهر اصفهان»

Insert Into Values ('S4'، 'آلومین'، 'اصفهان')

دستور فوق با فرمت زبان SQL نوشته شده است. این زبان را بعداً بطور کامل شرح می‌دهیم.

مثال ۵ (مثال از حذف) : در جدول SP این اطلاع را حذف کنید «S3 از قطعه P2 تعداد 200 عدد تهیه کرده است».

Delete From SP Where S# = ‘S3’ and P# = ‘P2’

مثال ۶ (مثال از بهنگام کردن) : در جدول S «شهر تهیه کننده S1 را از تهران به مشهد تغییر دهید»

Update S Set Sity = ’مشهد‘ Where s# = ‘S1’

دستورات Update , Delete به فرم SQL نوشته شده‌اند.

نتیجه اینکه عملیات درج و حذف و بهنگام سازی در مدل رابطه‌ای به سادگی و بدون دشواری و بروز وضعیت نامطلوب انجام می‌پذیرد.

امروزه نرم‌افزارهای زیادی در دنیای PCها وجود دارد که ادعا می‌کنند بانک اطلاعاتی رابطه‌ای هستند. اما این سیستم‌ها که بانکهای جدولی ایجاد می‌کنند در واقع سیستم‌های شبه رابطه‌ای می‌باشند. توجه کنید فقط وجود ساختار جدولی دلیل بر ساختار رابطه‌ای نمی‌باشد. بانک اطلاعات رابطه‌ای تعاریف و شرایط دقیقی دادر که در فصل آینده شرح می‌دهیم. کاد (واضح و تئوریسین مدل رابطه‌ای) در سال ۱۹۸۱ اقلاً دو شرط را برای اتلاق «شبه رابطه‌ای» به اینگونه نرم‌افزارها بیان کرد :

۱ـ دید جدولی را برای کاربر تامین کند.

۲ـ دارای عملگرهای جبر رابطه‌ای از قبیل گزینش و پرتو باشند. جبر رابطه‌ای را بطور مفصل در فصل ۶ شرح خواهیم داد. در اینجا بطور خلاصه می‌گوئیم که :

عملگر گزینش (Select) سطرهایی از یک جدول را که حائز شرط یا شرایطی باشند، بازیابی می‌کند. عملگر پرتو (Project) ستون یا ستون‌هایی از یک جدول را می‌دهد.

خصوصیات مدل رابطه‌اي

۱ـ از دید کاربر، دارای وضوح است و محیط انتزاعی آن محیطی مسطح می‌باشد.

۲ـ داده‌ها و ارتباطات بین آنها با مکانیسم واحدی نشان داده می‌شود (سطرها)

۳ـ عملگر بازیابی نسبتاً ساده است.

۴ـ برای پرس‌وجوهای قرینه دارای رویه پاسخگویی واحدی است.

۵ـ در عملیات ذخیره‌سازی دشواری ندارد و سبب بروز وضعیت نامطلوب نمی‌شود.

۶ـ از مبنای تئوریک ریاضی برخوردار است.

۷ـ غواصی در هر یک از رابطه‌ها (جدولها) می‌تواند مستقل از رابطه دیگر انجام شود.

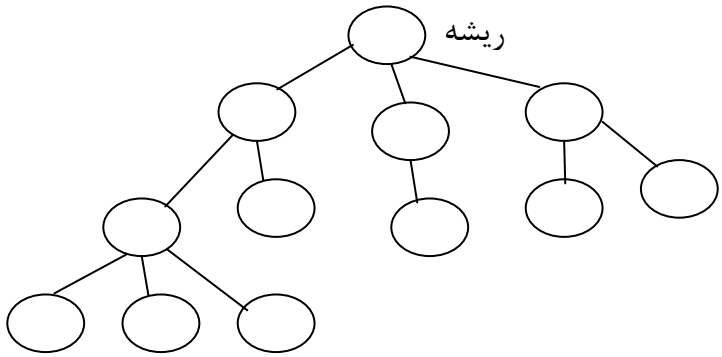
۸ـ برای پاسخگویی به بعضی از پرس‌و جوها باید رابطه‌های مستقل از یکدیگر به نحوی با هم مرتبط شوند و این امر می‌تواند باعث افزایش زمان پاسخ‌دهی شود و لذا لزوم بهینه‌سازی رویه‌های پرس‌وجو مطرح می‌گردد. در فصلهای بعدی در این باره مفصلاً بحث خواهیم کرد.

۹ـ برای طراحی رابطه‌ها به صورت مطلوب دارای ابزار تئوریک است. این ابزار موسوم به تئوری نرمال‌سازی رابطه‌ها می‌باشند که در فصول آخر آنها را شرح می‌دهیم.

ساختار سلسله مراتبي

این ساختار قدیمی‌ترین ساختار داده‌یی برای طرحی بانک اطلاعاتی در سطح انتزاعی است. در ساختار داده‌ها و ارتباط بین آنها به کمک یک درختواره نمایش داده می‌شوند. درختواره گرافی است دارای یک ریشه، به هم بسته و غیر چرخشی. منظور از به هم بسه این است که بین هر دو گره پیوندی وجود دارد.

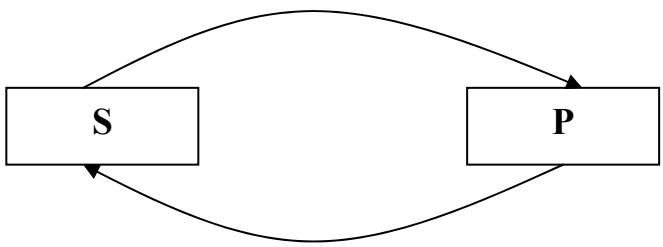
غیرچرخشی یعنی مسیری از گره سطح پائین‌تر به گرهی از سطح بالاتر وجود ندارد. رابطه همواره از سطح بالاتر به سطح پائین‌تر است. هر گره پدر می‌تواند چندین فرزند داشته باشد ولی هر فرزند فقط یک پدر دارد.



با حذف گره پدر، گره‌های فرزند نیز حذف می‌شوند.

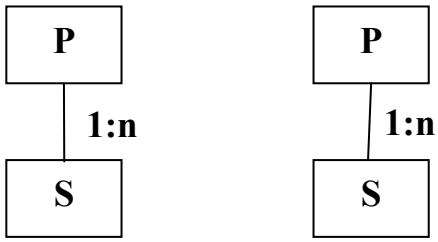
ساختار سلسله مراتبی برای مدلینگ ارتباطات یک به چند یکسویه بین انواع موجودیتها مناسب است. هر گره از درختواره می‌تواند رکوردی باشد که یک نوع موجودیت را نشان دهد. در درختواره انواع مختلف رکوردها ممکن است وجود داشته باشد و از این رو درختواره معمولاً از نظر انواع رکوردها ناهمگن (Heterogeneous) است. حالت خاصی از درختواره، درختواره همگن (Homogemeous) است که در آن یک نوع رکورد وجود دارد. در بعضی سیستم‌های سلسله مراتبی اصطلاح سگمنت (Segment) برای رکورد به کار برده می‌شود.

مثال ۷ : موجودیتهای تهیه کننده (S) و قطعه (P) را در نظر بگیرید.

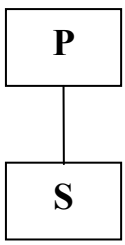


ارتباطات یک به چند دو سویه به ناچار به کمک دو سلسله مراتب مجزا نشان داده می‌شود.

ارتباط بین دو موجودیت S , P را می‌توان به کمک دو سلسله مراتب زیر نشان داد :



مثلاً اطلاعات موجود در جداول SP, P, S در مدل رابطه‌ای را در مدل سلسله مراتبی می‌توان به شکل زیر نشان داد :



در شکل فوق P4 هنوز توسط تهیه‌کننده‌ای تولید نشده است لذا فرزندی ندارد. با این همه P4 یک سلسله مراتب است که فقط ریشه دارد و اصطلاحاً به آن سلسله مراتب «فقط ریشه» یا Root Only گفته می‌شود. نمونه‌ای از سیستم سلسله مراتبی نرم‌افزار IMS می‌باشد.

بررسي رفتار مدل سلسله مراتبي

در این مدل برای بازیابی نیاز به عملگر ریشه‌یاب و عملگر وابسته‌یاب داریم. یعنی در عملیات غواصی، سیستم باید روی یک نمونه پدر توقف کرده و وابسته یا وابستگان حائز شرط موردنظر را بیابد.

در زیر همان دو سئوال قرینه را که در مدل رابطه‌ای مطرح کردیم بیان می‌کنیم.

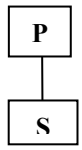
مثال ۸ (Q1) : «شماره تهیه‌کنندگان P2 را بیابید». در این حال باید ابتدا ریشه P2 را یافته و سپس به سادگی فرزندان ۲ را پویش کرده و شماره‌های S3, S2, S1 را بازیابی کنیم.

حال پرس و جوی قرینه فوق را در نظر می‌گیریم.

مثال ۹ (Q2) : «تهیه کننده S2 چه قطعاتی را تهیه کرده است»

در اینجا آرگومان جستجو مربوط به ریشه نیست یعنی آرگومان S2 ناظر به فرزند است و نه پدر. با توجه به اینکه مسیر بررسی همواره از بالا به پائین است به ناچار باید تمام نمونه ریشه‌ها را پیمایش کرد و زیر هر رشته را بررسی کرد که آیا تهیه کننده S2 وجود دارد و یا خیر و در صورت وجود چنین فرزندی پدرش باید بازیابی شود.

برای این عمل بازیابی باید تمام بانک جستجو شود. همانطور که مشاهده می‌شود رویه پاسخگویی به Q2 قرینه رویه مربوط به Q1 نیست و این عدم تقارن از نقاط ضعف مدل سلسله مراتبی می‌باشد.



البته اگر سلسله مراتب را هم در بانک پدید می‌آوردیم آنگاه برای جواب دادن به Q2 از این ساختار دومی استفاده کرده و رویه پاسخگویی قرینه می‌شد ولی در این روش افزونگی زیاد اطلاعات را خواهیم داشت.

این مدل در عملیات ذخیره‌سازی (یعنی در سه عمل حذف، درج و بهنگام سازی) دارای آنومالی است. آنومالی (Anomaly) یعنی وجود دشواری در انجام یک عمل خاص و یا عدم امکان انجام عمل و یا بروز عوارض نامطلوب در پی انجام یک عمل خاص.

مثال ۱۰ (مثال از درج) : اطلاع از زیر را درج کنید : «تهیه کننده جدید S4 را در بانک درج کنید» انجام این درج امکان‌پذیر نیست تا زمانی ندانیم S4 چه قطعه‌ای تهیه کرده است. در واقع پدر این فرزند نامشخص است. به ناچار برای درج این رکورد از یک نمونه قطعه خالی یا مجازی (Dummy) استفاده می‌شود تا بتوان S4 را به عنوان فرزند در ذیل آن درج کرد. بنابراین می‌گوئیم مدل سلسله مراتبی در عمل درج آنومالی دارد.

مثال ۱۱ (مثال از حذف) : اطلاع زیر را حذف کنید : «تهیه کننده S3 از قطعه P2 به تعداد 200 عدد تهیه کرده است».

انجام این عمل حذف منطقاً امکان‌پذیر است کافی است نمونه S3 در زیر P2 حذف شود ولی با این حذف اطلاعات ناخواسته دیگری (مثل نام و شهر S3) نیز حذف می‌شود. بنابراین این عمل حذف عوارض نامطلوب در پی دارد و مدل در حذف دارای آنومالی است.

البته این آنومالی عمومیت ندارد. مثلاً در حذف «S1 از P1 تعداد 300 عدد تهیه کرده است» آنومالی وجود ندارد چرا که اطلاعات S1 در جاهای دیگری مثل زیر P3 و زیر P2 نیز ذخیره شده است.

آنومالی دیگر در حذف ریشه است. مثلاً اگر بخواهیم قطعه P2 را حذف کنیم تمام فرزندان آن یعنی نمونه‌های S3, S2, S1 از بین می‌روند. بنابراین از آنجا که S3 تنها در ذیل P2 وجود دارد، اطلاعات مربوط به S3 از بین می‌رود.

مثال ۱۲ (مثال از بهنگام‌سازی) : «شهر تهیه کننده S1 را از تهران به مشهد تغییر دهید.» این عمل که منطقاً بهنگام سازی یک فقره اطلاع ساده در یک رکورد است باید در تمام نمونه‌های S1 موجود در بانک انجام شود. اصطلاحاً می‌گوئیم عمل بهنگام سازی منتشر شونده (Propagating Update) باید صورت گیرد که طبعاً حجم عملیات را افزایش می‌دهد. اگر این بهنگام‌سازی در تمام نقاطی که S1 وجود دارد انجام نشود در این صورت بانک دچار پدیده ناسازگاری داده‌ها شده و جامعیت بانک لطمه می‌خورد. عمل بهنگام‌سازی منتشر شونده مطلوب نیست زیرا عملیات رکوردی را تبدیل به عملیات روی مجموعه‌ای از رکوردها می‌کند و خود نوعی آنومالی است.

خصوصیات مدل سلسله مراتبي

۱ـ از دید کاربر وضوح دارد ولی نه به حد مدل رابطه‌ای و محیط انتزاعی آن مسطح نیست.

۲ـ ارتباط بین داده‌ها به کمک یک درختواره که مسیر منطقی آن از بالا به پائین است نشان داده می‌شود، لذا خاص محیط‌هایی است که در آنها ارتباط‌های یک به چند یک سویه وجود دارد.

۳ـ عملگرهای بازیابی به سادگی عملگرهای مدل رابطه‌ای نیستند. برای انجام عملیات از پیمایش اشاره‌گرها استفاده می‌گردد.

۴ـ غواصی در سطحی پائین‌تر الزاماً باید از نقطه ورود، در سطح بالاتر شروع شود.

۵ـ برای پاسخگویی به پرس و جوهای قرینه رویه‌های پاسخگویی قرینه ندارد.

۶ـ در عملیات ذخیره‌سازی آنومالی دارد.

۷ـ از مبانی تئوریک ریاضی (آنگونه که مدل رابطه‌ای دارد) برخوردار نیست.

۸ـ قدیمی‌ترین ساختار داده‌یی برای طراحی بانک در سطح انتزاعی است.

۹ـ طراحی ساختار برای یک محیط عملیاتی ممکن است در بیش از یک صورت انجام شود و برخلاف مدل رابطه‌ای تئوری منسجمی برای طراحی ندارد.

۱۰ـ وجود ارتباطات از پیش پیاده‌سازی شده بین انواع رکوردها، تا حدی پاسخگویی به پرس و جوها را در مرحله اجراء تسریع می‌کند.

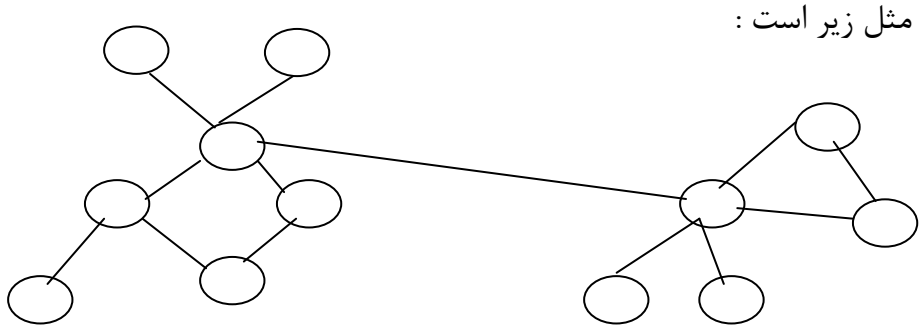
تذکر :نمونه‌ای از سیستم سلسله مراتبی، سیستم IMS شرکت IBM می‌باشد که به جای واژه رکورد از Segment استفاده می‌کند.

ساختار شبکه‌ای

ساختار شبکه‌ای که به آن ساختار پلکس (PLEX) نیز می‌گویند، نخستین بار در سال ۱۹۶۶ توسط DBTG(Data Base Task Group) پیشنهاد شد و در سال ۱۹۷۱ معروفترین سیستم بانک اطلاعاتی شبکه‌ای و قابل قبول ANSI به نام کوداسیل (CODASYL) طراحی و معرفی شد.

در این ساختار هر گرهٔ فرزند می‌تواند بیش از یک گره پدر داشته باشد. این ساختار که جامع‌تر از ساختار سلسله مراتبی است برای نمایش ارتباطات یک به چند دوسویه مناسب است.

در واقع ساختار سلسله مراتبی حالت خاصی از ساختار شبکه‌ای است و ساختار شبکه‌ای را می‌توان با پذیرش مقداری افزونگی به ساختار سلسله مراتبی تبدیل کرد. در واقع ساختار شبکه‌ای گرافی مثل زیر است :



ساختار شبکه‌ای هم در سطوح انتزاعی و هم در سطوح داخلی پیچیده‌تر از ساختار سلسله مراتبی است. مثلاً کسی که ساختارهای فیزیکی مناسب برای ارتباطات شبکه‌ای ساختار چند حلقه‌ای است (که در درس ذخیره و بازیابی مطرح می‌شود).

طراحی سطح ادراکی در مدل شبکه‌ای مبتنی بر مفهوم مجموعه کوداسیلی (CODASYL SET) می‌باشد. مجموعه کوداسیلی به معنای دقیق ریاضی، مجموعه نیست. مجموعه را می‌توان یک درختواره دو سطحی تلقی کرد که در هر سطح آن یک نوع رکورد وجود دارد.

سطح بالاتر را اصطلاحاً مالک (Owner) و رکورد سطح پائین‌تر، عضو (Member) نامیده می‌شود. یک نوع مجموعه (Set type) دارای یک رکور نوع مالک و یک رکورد نوع عضو است. هر مجموعه می‌تواند نمونه‌های مختلفی داشته باشد در هر نمونه از یک نوع مجموعه، یک نمونه رکوردنوع مالک و از صفر تا چندین نمونه از رکورد نوع عضو وجود دارد. مالک با اولین عضو پیوند دارد و سپس اعضاء دیگر با همدیگر در پیوندند و بالاخره آخرین عضو با مالک در پیوند است.

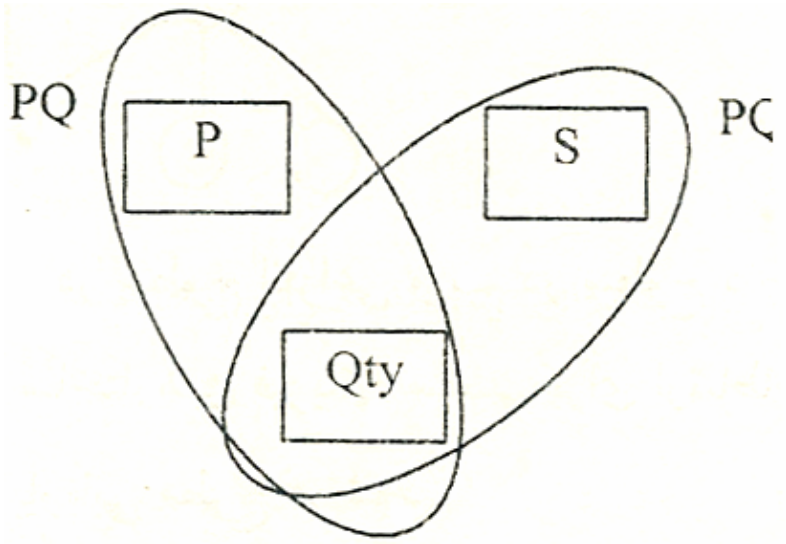
مثال ۱۳ : ارتباط دو موجودیت کلاس و دانشجو را در نظر گرفته و فرض می‌کنیم رکورد نوع کلاس C مالک و رکورد نوع دانشجو (S) عضو باشد. مجموعه نوع کلاس ـ دانشجو (CS) ارتباط بین این دو نوع رکورد را نشان می‌دهد و دو نمونه از این مجموعه CS مثلاً به صورت زیر است :

نکته مهم در این ساختار این است که یک نمونه رکورد عضو می‌تواند عضو دو مجموعه متمایز باشد. یعنی دو پدر داشته باشد و به بیان دیگر ذیل دو نوع مالک باشد. به کمک همین امکان است که ارتباطات یک به چند دوسویه نمایش داده می‌شود.

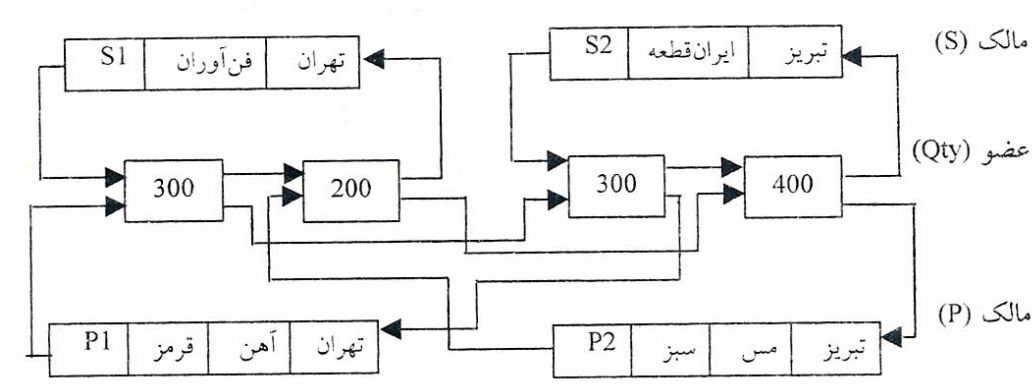
نکته مهم دیگر اینکه یک عضو خود می‌تواند مالک در مجموعه‌ای دیگر باشد و بدین ترتیب امکان نمایش ارتباطات چند سطحی نیز وجود دارد.

مثال ۱۴ : حال همان مثال قطعه و تهیه کننده را در نظر می‌گیریم. برای طراحی در سطح ادراکی دو مجموعه را به صورت زیر در نظر می‌گیریم :

در مجموعه SQ، رکورد نوع S را مالک و رکورد نوع Qty را عضو می‌گیریم. در مجمو.عه PQ، را مالک و Qty را عضو می گیریم.



بنابراین Qty عضو هر دو مجموعه است و همین عضویت ارتباط دو سویه بین S , P را پیاده‌سازی می‌کند. رکورد Qty که دارای یک فیلد است رکورد پیونددهنده نام دارد. نمونهٔایی از این دو مجموعه و شکل ارتباطی آنها به صورت زیر است (برای سادگی فقط S1 , S2 و نیز P1 , P2 را در نظر گرفته‌ایم) :



با توجه به شکل فوق می‌فهمیم که S1، به تعداد 300 عدد از P1 و 200 عدد از P2 تولید کرده است. چرا که مثلاً بلوک 300 هم عضو مالک S1 و هم عضو مالک P2 می‌باشد. همچنین در می‌یابیم که S2 از P2 به تعداد 400 عدد تولید کرده است. چرا که عضو 400 هم متعلق به مالک S2 و هم متعلق به مالک P2 می‌باشد. همچنین از شکل فوق در می‌یابیم که قطعه P2 توسط S1 به تعداد 200 عدد و توسط S2 به تعداد 400 عدد تهیه شده است.

بررسي رفتار مدل شبکه‌اي

عمل بازیابی در مدل شبکه‌ای پیچیده‌تر از مدل سلسله مراتبی است.

مثال ۱۵ (Q1) : «شماره تهیه کنندگان P2 را بیابید».

برای این دستور ابتدا باید به سراغ مالک P2 برویم. سپس فلش‌های خارج شده از آن را دنبال کنیم تا به رکوردهای پیوند دهندهٔ 200 و سپس 400 برسیم. هنگامی که به بلوک 200 رسیدیم فلش بالای آن را باید دنبال کنیم تا S1 استخراج شود. همچنین هنگامی که به بلوک 400 رسیدیم فلش بالای آن را باید دنبال کنیم تا S2 استخراج شود.

مثال ۱۶ : حال پرس و جوی قرینه فوق را در نظر می‌گیریم «تهیه کنندهٔ S2 چه قطعاتی را تهیه کرده است؟» در این حال با توجه به فلش خروجی از S2 به سراغ بلوکهای 300 , 400 رفته و سپس به کمک فلش‌های خروجی از آنها به اسامی P1 , P2 می‌رسیم.

با توجه به مثال فوق در می‌یابیم که رویه پاسخگویی به سئوالات قرینه حالت تقارن دارد و از این نظر مدل شبکه‌ای بر مدل سلسله مراتبی مزیت دارد ولی رویه پاسخگویی پیچیده‌تر از

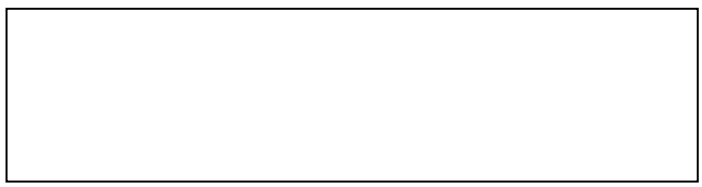
مدلهای رابطه‌ای یا سلسله مراتبی است. از این رو نباید به قرینه بودن رویه پاسخگویی چندان بها داد.

در دو مثال فوق بازیابی مالک بود حال ببینیم اگر بازیابی براساس رکورد پیوند دهنده باشد چه اتفاقی می‌افتد.

مثال ۱۷ : «تهیه کننده S1 از قطعه P2 به چه تعداد تهیه کرده است؟»

در اینجا DBMS با انتخاب مسیر دستیابی مواجه است آیا ابتدا باید S1 را یافته و سپس عضوی از آنرا که مالکش P2 نیز هست را چاپ کند یا برعکس. این انتخاب در عمل ممکن است مشکل‌ساز باشد. مدل شبکه‌ای در عملیات ذخیره‌سازی (درج، حذف و بهنگام‌سازی) آنومالیهای مدل سلسله مراتبی را ندارد.

مثال ۱۸ (درج) : می‌توان تهیه کننده‌ای جدید را در بانک درج کرد بی‌آنکه بدانیم چه قطعه‌ای را تهیه کرده است. تهیه کننده جدید به عنوان یک نمونه از مالک مجموعه SQ درج می‌شود و با خودش پیوند برقرار می‌گردد تا زمانی که قطعه‌ای را تهیه کند :



در کوداسیل برای درج رکوردها و پیوندها دو عملگر جداگانه وجود دارد. عملگر STORE برای درج نمونه رکورد و عملگر CONNECT برای برقراری پیوند بین نمونه رکوردها. بنابراین اصل وحدت عملگر برای عمل واحد در کوداسیل رعایت نشده است.

مطلوب آن است که برای عمل درج (چه درج رکورد و چه درج ارتباط) مانند مدل رابطه‌ای فقط یک عملگر واحد وجود داشته باشد.

مثال ۱۹ (حذف) : می‌توان اطلاع مورد نظری را حذف کرد بی‌آنکه اطلاع ناخواسته‌ای از دست برود. حذف این اطلاع که «S2 از P2 به تعداد 400 عدد تهیه کرده است» منجر به حذف

اطلاع در مورد خود S2 نمی‌شود. کافی است فلش خروجی از بلوک 300 را که به 400 رفته پاک کرده و به S3 ببریم.

مثال ۲۰ (بهنگام‌سازی) : اگر بخواهیم شهر تهیه کننده S1 را عوض کنیم با مشکل بهنگام سازی منتشر شونده مواجه نیستیم زیرا S1 فقط یکبار در بانک ذخیره می‌شود.

خصوصیات مدل شبکه‌ای

۱ـ از دید کاربر وضوح کاربری ندارد و محیط انتزاعی آن محیط مسطح نیست.

۲ـ ارتباط یک به چند دوسویه با طراحی دو مجموعه که رکورد پیوند دهنده در هر دو عضویت دارد، پیاده‌سازی می‌شود.

۳ـ برای محیط‌های دارای ارتباط یک به چند دو سویه مدل مناسبی است. ارتباط یک به چند یک سویه حالتی خاص از آن است.

۴ـ عملگر بازیابی پیچیده‌تر از مدل سلسله مراتبی است ولی خاصیت تقارن دارد.

۵ـ در عملیات ذخیره‌سازی آنومالی ندارد.

۶ـ اصل وحدت عملگر در یک عمل واحد مثل درج رعایت نمی‌شود.

۷ـ علیرغم وجود مفهوم مجموعه، از مبانی تئوریک ریاضی برخوردار نیست (آنگونه که مدل رابطه‌ای برخوردار است)

۸ـ در بعضی پرس و جوها، سیستم با مساله انتخاب مسیر (نقطه ورود غواصی) مواجه است.

۹ـ وجود ارتباطات از پیش پیاده‌سازی شده، تا حدی پاسخگویی پرس و جوها را در مرحله اجراء تسریع می‌کند.

تذکر : در ساختار بانک اطلاعاتی از مفاهیم ریاضی : مجموعه‌ها، رابطه، درختواره (Tree) گراف (Graph) و هایپرگراف (Hypergraph) و غیره استفاده شده است.

تذکر : سیستم‌های قدیمی به سه دسته تقسیم می‌شوند لیست معکوس (Inverted List)، سلسله مراتبی و شبکه‌ای. البته لیست معکوس را در اینجا بررسی نکردیم.

اخیراً محصولات شیی‌ءگرا و شیء گرا/رابطه‌ای نیز به بازار عرضه شده‌اند. علاوه بر اینها تحقیقات گسترده‌ای بر روی روشهای دیگری مثل روش چند بعدی و روش مبتنی بر منطق (یا روش استقرائی یا خبره) انجام شده است.

مدلهاي داده‌اي

مجموعه‌ای است از اشیاء مربوط به هم، عملگرها و قوانین جامعیت داده‌ای که موجودیت انتزاعی (غیرعینی) را تشکیل می‌دهند که DBMS از آنها پشتیبانی می‌کند. هدف از مدل داده‌ای درک داده و ارائه آن در یک محط بانک اطلاعاتی است. عموماً هر DBMS از یک مدل داده‌ای حمایت می‌کند.

مدلهای داده‌ای به سه گروه اصلی تقسیم می‌شوند.

1-Object Based Data Model

2- Record Based Data Model

3- Physical Data Model

۱ـ مدلها بر پایه شيء

این مدل برای اجزایی چون موجودیتها، صفات خاصه و ارتباطات آنها به کار برده می‌شود. یک موجودیت چیزی است که می‌خواهیم در مورد آن اطلاعات داشته باشیم و صفت خاصه مشخص کننده شیء موردنظر (صفات مربوط به آن شیء) می‌باشد. بعضی از زیر مدلهای مربوط به مدل داده‌ای بر پایه شیء عبارتند از :

الف) Entity Relationship که قبلاً به طور کامل تشریح شده است.

ب) Semantic (معناشناسی) در برنامه‌سازی به رابطه بین کلمات، نماها و معانی موردنظر آنها گفته می‌شود.

ج) Functional (عملیاتی) مشخصات رابطه‌های بین بخشهای یک سیستم از جمله جزئیات، اجزاء یا روشهای کاربرد آنها با یکدیگر می‌باشد.

شیی‌گرایی : روندی است برای ایجاد محصولا نرم‌افزاری که در این روش می‌توان اشیاء را جهت برآورده نمودن نیازهای جدید به کار برد.

۲ـ مدلها بر پایه رکورد

در این مدل بانک اطلاعاتی تعدادی رکورد به شکلهای متفاوت دارد. هر رکورد از چند فیلد تعریف شده است و هر فیلد برای خود یک طول و یک مقدار دارد این مدل به سه دسته کلی الف) سلسله مراتبی ب) شبکه‌ای ج) رابطه‌ای تقسیم می‌شود که قبلاً شرح داده شد.

۳ـ مدل داده‌اي فيزيكي

این مدل بیانگر حفظ داده در کامپیوتر و اطلاعاتی درباره ساختارهای رکوردها، ترتیب رکوردها و دستیابی رکوردها می‌باشد.

مدل و پیاده‌سازی

مدل رابطه‌ای تنها مدل داده‌ای نیست.

مدل داده‌ای یک تعریف انتزاعی، خود شمول (Self – Contained) و منطقی از اشیاء، عملگرها و غیره است که ماشین انتزاعی را به وجود می‌آورند که کاربران با آنها در تعاملند. با اشیاء می‌توان ساختمان داده‌ها را مدل‌سازی کرد و با عملگرها می‌توان رفتار را مدلسازی نمود.

پیاده‌سازی یک مدل داده‌ای عبارت از، درک ماشین واقعی از قطعات ماشین انتزاعی است که با یکدیگر تشکیل مدل را می‌دهند.

خلاصه اینکه : مدل چیزی است که کاربران راجع به آن می‌دانند و آن را می‌شناسند ولی پیاده‌سازی چیزی است که کاربران از آن خبر ندارند. تمایز بین مدل و پیاده‌سازی حالت خاصی از تمایز بین حالت منطقی و فیزیکی است.

اما بسیاری از بانکهای اطلاعاتی امروزی آن طور که باید بین آنها تمایز قائل نمی‌شوند. در واقع می‌توان گفت که درک خوبی از تمایز این دو وجود ندارد. در نتیجه «اصول بانک اطلاعاتی» (آنچه که باید باشد) و «عمل» آن (آنچه که هست) فاصله وجود دارد. در این کتاب با اصول سر و کار داریم.

هدف از این درس حفظ ارزش داده‌ها و گرفتن اطلاعات مفید از داده‌هاست.

فصل پنجم

مفاهیم مدل رابطه‌ای

ساختار رابطه‌ای نخستین بار توسط کاد (ریاضی‌دان) به عنوان ساختاری برای طراحی بانک مطرح شد.

رابطه در ریاضیات

دامنه یا میدان (Domain) : مجموعه‌ای است که مقادیر یک صفت خاصه از آن برگرفته می‌شوند. مثلاً میدان نام تهیه‌کننده‌ها و شهرها به صورت زیر است :

D_{sname} = {آلومین، پولادین، ایران قطعه، فن‌آوران}

D_{city} = {شیراز، تهران، تبریز}

رابطه : رابطه زیر مجموعه‌ای است از ضرب دکارتی چند دامنه.

مثال ۱ : $\{1,2,3\} \times \{4,5\} = \{(1,4), (1,5), (2,4), (2,5), (3,4), (3,5)\}$

R یک رابطه است : $R = \{(1,5), (2,4), (3,4)\}$

تاپل (Tuple) : به عضو (3,4) از رابطه R یک تاپل گویند. پس تاپل به اعضاء رابطه گفته می‌شود. به عبارتی دیگر تاپل مجموعه‌ای است از مقادیر صفات خاصه.

یک راه دیگر نمایش رابطه استفاده از جدول است. مثلاً رابطه R را می‌توان به صورت جدول زیر نمایش داد :

$\Rightarrow R = \{(1,5), (2,4), (3,4)\}$

1	5
2	4
3	4

مثال ۲ :

$\{1,2,3\} \times \{k,m\} \times \{A,B\} =$

$\{(1,k,A)\}, \{1,k,B), (2,k,A), (2,k,B), (3,k,A), (3,k,B), (1,m,A), \ldots\}$

ضرب دکارتی ۳ مجموعه فوق مجموعه‌ای از ۳ تائی‌های مرتب را می‌دهد. هر زیر مجموعه‌ای از حاصلضرب فوق یک رابطه است. از آنجا که رابطه مذکور از ۳تائی‌های مرتب تشکیل شده است، درجه این رابطه ۳ می‌باشد. پس درجه رابطه تعداد صفات خاصه رابطه است.

مثال ۳ :

{S1, S2} × {پولادین، فن‌آوران} × {شیراز، تبریز، تهران}

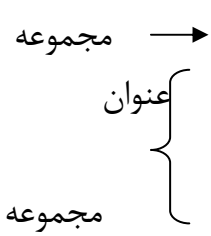
= {(شیراز، فن‌آوران، S1) و (تبریز، فن‌آوران، S1) و (تهران، فن‌آوران، S1) ...}

تعداد ستونهای جدول همان درجه رابطه است. کاردینالیتی رابطه : تعداد تاپلهای رابطه در یک لحظه از حیات آن، کاردینالیتی رابطه نام دارد و در طول حیات رابطه متغیر است.

S#	Sname	City
S1	فن‌آوران	تهران
S1	فن‌آوران	تبریز
S1	فن‌آوران	شیراز
S2	فن‌آوران	تهران
S2	فن‌آوران	تبریز
. . .	. . .	

تذکر : رابطه از دو مجموعه عنوان (Heading) و پیکر (Body) تشکیل یافته است. مجموعه عنوان مجموعه اسامی صفات خاصه است و مجموعه پیکر، مجموعه‌ای است متغیر در زمان از تاپلها.

مثال ۴ : در شکل روبرو :



S#	Sname	City
S1	فن‌آوران	تهران
S2	ایران قطعه	تبریز
S3	پولادین	تبریز

مجموعه عنوان رابطه S عبارت است از :

H_s = {S#, Sname, City}

و مجموعه پیکر در یک لحظه از حیات رابطه عبارت است از :

B_s = {{S1، {تهران، فن‌آوران}}، {S2، {تبریز، ایران‌قطعه}}، {S3، {تبریز، پولادین}}}

تذکر : کاردینالیتی مجموعه عنوان، همان درجه رابطه است.

با توجه به توضیحات و مثالهای فوق در می‌یابیم که جدول در واقع نمایشی از مفهوم ریاضی رابطه است و مفاهیم زیر با هم معادلند :

در ریاضیات	رابطه	تاپل	صفت خاصه	میدان یا دامنه
در کامپیوتر	جدول	سطر (رکورد)	ستون (فیلد)	مقادیر مجاز هر فیلد

سطرهای نمایشی جدول رابطه (یعنی عناصر مجموعه پیکر رابطه) در یک لحظه از حیات آن را اصطلاحاً «گسترده یا بسط رابطه» می‌نامند.

تذکر : میدانهای یک رابطه لزوماً از یکدیگر مجزا نیستند، یعنی دو ستون می‌توانند دارای یک میدان یکسان باشند، مثلنام کوچک فرد و نام پدر. اگر میدانهای رابطه از یکدیگر مجزا باشند می‌توان گفت درجه رابطه همان تعداد میدان‌های رابطه است.

خصوصیات رابطه

رابطه طبق تعریف از دو مجموعه (عنوان ـ پیکر) تشکیل شده است و لذا خصوصیات زیر را داراست :

۱ـ در رابطه تاپل تکراری وجود ندارد. زیرا پیکر رابطه یک مجموعه است و مجموعه در ریاضیات طبق تعریف عناصر تکراری ندارد. به همین ترتیب در بانک جدولی نیز رکوردهای تکراری نداریم.

۲ـ تاپلها در رابطه نظم ندارند. این خصوصیت نیز از مجموعه بودن پیکر رابطه نتیجه می‌شود. به همین ترتیب در بانک اطلاعاتی جدولی نیز ترتیب رکوردها در جدول مهم نیست. هر چند که معمولاً سطرها با یک نظم خاص نشان داده می‌شوند.

۳ـ صفات خاصه نظم ندارند. این خاصیت نیز از مجموعه بودن عنوان رابطه نتیجه می‌شود. به همین ترتیب فیلدهای یک جدول نیز نظم ندارند و می‌توان آنها را جابجا کرد. هنگام تعریف یک جدول مهـــــم نیست ترتیب فیلدها چگونه باشد.

مثلاً $S(s\#, sname, city) \Leftrightarrow S(sname, city, s\#)$

۴ـ همه مقادیر صفات خاصه تجزیه ناپذیرند. به عبارتی دیگر در رابطه، یک تاپل نمی‌تواند حاوی تاپل دیگری باشد. مثلاً مجموعه R زیر، رابطه نیست، چون عضو $(1,(4,5))$ قابل قبول نمی‌باشد :

$R = \{(1,(2,3)), (1,(4,5)), \ldots\}$

به همین ترتیب در بانک اطلاعاتی رابطه‌ای نمی‌توان مثلاً جدولی به صورت زیر تعریف کرد که فیلدی از آن مرکب بوده و از فیلدهای ساده‌تری تشکیل شده باشد. به عبارت دیگر از تقاطع هر سطر و ستون باید یک مقدار بدست آید.

معدل	نام		تاریخ		
	نام کوچک	فامیلی	ماه	روز	سال

مقدار اتومیك (atomic) : مقداری است ساده که قابل تجزیه به مقادیر دیگر نباشد، به بیان دیگر از یک میدان ساده برگرفته شود. مثلاً در شکل فوق نام و تاریخ اتومیک نیست ولی معدل اتومیک است.

تعریف : رابطه‌ای که همه مقادیر صفات خاصه آن اتومیک باشند، به رابطه نرمال شده (Normalized) موسوم است. در واقع اصطلاح رابطه در بانک رابطه‌ای، همیشه به رابطه نرمال اطلاق می‌شود.

نقش میدان در عملیات روي بانك

میدان در عملیات روی بانک سه نقش دارد :

۱ـ کنترل مقداري پرس‌و جوها

مقادیر یک صفت خاصه در طول حیات رابطه از مقادیر میدان برگرفته می‌شوند.

مثلاً اگر میدان مقادیر City به صورت روبرو باشد : D_{city} = {اصفهان، تبریز،تهران} و بخواهیم تاپل (یزد، پردازش، S7) را در جدول S درج کنیم این کار امکان‌پذیر نیست و سیستم مدیریت بانک (DBMS) این درخواست را رد می‌کند چرا که مقدار شهر آن خارج از مقادیر میدان است.

۲ـ کنترل سمانتیك پرس و جوها

مثال زیر را در نظر بگیرید : «شماره جفت قطعاتی را بدهید که وزن قطعه اول برابر با تعداد تهیه شده از قطعه دوم باشد». این پرس و جو با آنکه امکان‌پذیر است ولی اساساً فاقد سمانتیک منطقی است. با آن که وزن و تعدادهر دو فیلدی عددی هستند ولی مقایسه آنها بی‌معناست. سیستمی که در آن مفهوم میدان پیاده‌سازی شده باشد می‌تواند اینگونه پرس و جوهای غیرمنطقی را شناسایی کرده و آنها را رد کند.

۳ـ تسهیل در پاسخگویي به بعضي از پرس و جوها

پرس و جوی زیر را در نظر بگیرید «در چه رابطه‌هایی، اطلاعاتی در مورد تهیه‌کنندگان وجود دارد» به عبارتی دیگر «در چه رابطه‌هایی اقلا یک صفت خاصه وجود دارد که روی میدان شماره تهیه کننده تعریف شده باشد؟» در سیستمی که در آن مفهوم میدان پیاده‌سازی شده باشد، پرس و جوی فوق به پرس و جویی از کاتالوگ بانک تبدیل می‌شود. کاتالوگ بانک همانطور که قبلاً گفتیم نوعی راهنمای بانک است که در آن اطلاعاتی در مورد بانک وجود دارد از جمله مشخصات کلیه رابطه‌ها

کلید در مدل رابطه‌ای

۱ـ ابر کلید (Super Key یا S.K.) : یعنی هر ترکیبی از صفتها که خاصیت کلید داشته باشد. این تنها نوع کلید است که الزاماً کمینه نیست یعنی زیر مجموعه‌ای از آن هم ممکن است کلید باشد. مثلاً «شماره دانشجویی» و «نام دانشجو ـ شماره دانشجویی» هر دو ابر کلید هستند.

۲ـ کلید کاندید (Candidate Key یا C.K) : مجموعه‌ای از صفات خاصه $(A_I, A_j, \ldots, A_k)$ که دارای دو خاصیت زیر باشد، کلید کاندید رابطه R نامیده می‌شود.

الف) یکتائی مقدار (Uniqueness) به این معنا که هر لحظه از حیات رابطه، مقدار $(A_I, A_j, \ldots, A_k)$ را از آن حذف کنیم یکتایی مقدار از بین برود.

مثال ۵ : در رابطه S #, S کلید کاندید است و در رابطه صفت خاصه #P کلید کاندید می‌باشد.

تذکر : بدیهی است اگر کلید فقط شامل یک صفت خاصه باشد شرط minimality را خودبه خود دارد.

مثال ۶ : در رابطه S (S#, Sname) با آنکه کلید است (یکتائی مقدار دارد) ولی کلید کاندید نیست چرا که اگر Sname را از آن حذف کنیم هنوز #S به تنهایی یکتایی مقدار در تمام نمونه‌ها تاپلها خواهد داشت.

مثال ۷ : با آنکه نام دانشجو و شماره دانشجویی با همدیگر به صورت یکتا تمام دانشجویان را از یکدیگر متمایز می‌سازند ولی نام دانشجو در این بین زائد است و شماره دانشجویی برای این منظور کفایت می‌کند. لذا (نام و شماره دانشجویی) برای مجموعه دانشجویان کلید کاندید نیست. در این حالت به (نام و شماره دانشجویی) ابر کلید یا Super Key گفته می‌شود.

مثال ۸ : در رابطه SP صفت خاصه S# به تنهایی یا P# به تنهایی کلید کاندید نمی‌توانند باشند زیرا هیچیک از این دو به تنهایی یکتایی مقدار ندارند. در عوض جفت صفت خاصه (S#, P#) کلید کاندید رابطه SP است.

نکته : در هر رابطه حتماً حداقل یک کلید کاندید وجود دارد. زیرا در بدترین حالت خود مجموعه عنوان کلید کاندید رابطه است.

مثال ۹ : در جدول زیر (S#, P#, J#) کلید کاندید رابطه است. زیرا هیچیک از صفات خاصه به تنهایی یا دو به دو یکتایی مقدار ندارند.

S#	P#	J#
S1	P1	J1
S1	P2	J1
S2	P1	J1
S1	P1	J2

تعریف : رابطه‌ای که مجموعه عنوانش کلید کاندید آن باشد اصطلاحاً به رابطه «تمام کلید» (All – Key) موسوم است.

۳ـ کلید اصلي (Primary Key یا P.K) :کلید کاندیدی است که توسط طراح بانک انتخاب و معرفی می‌شود. دو ضابطه را در تعیین کلید اصلی، از بین کلیدهای کاندید باید در نظر گرفت :

الف) نقش و اهمیت کلید اصلی نسبت به سایر کلیدهای کاندید در پاسخگویی به نیازهای اطلاعاتی کاربران. مثلاً اگر کاربران اکثر سئوالاتشان را بر مبنای شماره دانشجویی مطرح می‌کنند (مثلاً مدل دانشجو با شماره 781392 چیست؟) شماره دانشجویی باید کلید اصلی باشد.

ب) کوتاه‌تر بودن طول کلید کاندید از نظر طول رشته بایتی. این ضابطه در سطوح ادراکی حائز اهمیت نیست بلکه در سطوح داخلی و فیزیکی بانک مطرح است.

٤ـ کلید فرعي یا بدیل (Alternative Key یا A.K.) : هر کلید کاندید غیر از کلید اصلی را کلید فرعی می‌نامند. طراح می‌تواند در شمای ادراکی هم کلید اصلی را معرفی کند و هم یک یا چند کلید فرعی را.

در اینجا این سئوال مطرح می‌شود که چرا هر رابطه‌ای باید کلید اصلی داشته باشد؟ جواب این است که : تنها راه مشخص کردن یک تاپل در یک رابطه، معرفی نام رابطه و مقدار کلید اصلی آن تاپل در رابطه است. یعنی جفت اطلاع K , R که در آن R نام رابطه و K کلید اصلی رابطه است. البته به کمک هر صفت خاصه‌ای، اعم از کلید یا غیرکلید می‌توان به تاپلهای رابطه دستیابی داشت ولی نتیجه دستیابی به تاپلها از طریق کلید اصلی، در پاسخگویی به پرس و جوها، حداکثر یک تاپل خواهد بود و نه بیشتر. از این رو وجود کلید اصلی در رابطه یک الزام اساسی است تا حدی که سیستم‌هایی مثل DB-2 که در آنها مفهوم کلید اصلی مطرح نیست سیستم‌های واقعاً رابطه‌ای نیستند.

٥ـ کلید خارجي (Foreign Key یا F.K) : صفت خاصه A_i از رابطه R2 کلید خارجی این رابطه نامیده می‌شود، اگر A_i در رابطه ۱ کلید اصلی (یا فرعی) باشد. (لزوماً R1 , R2 متمایز نیستند).

مثال ۱۰ :صفت خاصه #S کلید خارجی رابطه SP است زیرا #S کلید اصلی رابطه S می‌باشد و به همین‌ترتیب صفت خاصه #P کلید خارجی رابطه SP است زیرا #P کلید اصلی رابطه P می‌باشد.

کلید خارجی امکان است برای ارجاع از یک رابطه به رابطه دیگر و در واقع وسیله‌ای است برای پیوند دادن رابطه‌های بانک اطلاعاتی با یکدیگر. مثلاً وجود کلیدهای خارجی S# , P# در جدول SP نمایشگر ارتباطی است که بین تاپلهای رابطه S و تاپلهای رابطه P وجود دارد.

با این همه کلید خارجی تنها امکان ایجاد ارتباط نیست بلکه وجود هر صفت خاصه مشترک بین دو رابطه عاملی است برای نمایش ارتباط بین رابطه‌ها.

مثلاً صفت خاصه City هم در رابطه S وجود دارد و هم در رابطه P و در هیچکدام کلید اصلی نیست پس نمی‌تواند در هیچیک از این دو رابطه کلید خارجی باشد ولی City می‌تواند ارتباطی با سمانتیک مشخص را بین این دو رابطه (موجودیتهای قطعه و تهیه کننده) نشان دهد.

قواعد جامعیت در مدل رابطه‌ای

در مدل رابطه‌ای باید قواعدی وجود داشته باشد تا براساس آنها جامعیت و بی‌نقصی بانک، کنترل و تضمین شود. به این‌ها قواعد جامعیت یا Intefrity Rules گفته می‌شود. در مدل رابطه‌ای سه نوع جامعیت زیر مورد تاکید قرار می‌گیرد :

۱ـ قاعده جامعیت درون رابطه‌ای (Intra – Relation Integrity Rule) یعنی هر رابطه‌ای به تنهایی صحیح باشد. مثلاً عضو تکراری نداشته باشد و کلیدهایش درست باشند.

۲ـ قاعده جامعیت موجودیتی (Entity Integrity Rule) یعنی هیچ جزء تشکیل دهنده کلید اصلی نباید برای مقدار هیچ (NULL Value) باشد. مقدار هیچ مقدار خاصی است که برای نمایش مقدار شناخته یا مقدار غیرقابل اعمال به کار می‌رود و با جای خالی و یا صفر فرق دارد. مثلاً در جدول SP در تاپلی مقدار Qty می‌تواند NULL باشد. مثلاً تاپل (S1, P4, NULL) برای جدول SP(S#, P#, Qty) به این معناست که S1 قطعهٔP4 را تهیه کرده است ولی نمی‌دانیم به چه تعداد.

تذکر : در بعضی کتابها این قاعده به نام جامعیت دامنه‌ای (Domain integrity) بیان شده و بدین معناست که تمام صفات در تمامی رابطه‌ها باید از نوع دامنه خود باشند. مثلاً 54.7 به عنوان شماره دانشجویی که عدد صحیح مثبتی است پذیرفته نمی‌شود و نیز کلیدها نباید دارای مقادیر NULL یا تکراری باشند.

۳ـ قاعده جامعیت ارجاعی (Referential Integrity Rule) یعنی کلید خارجی درست تعریف شده باشد. اگر صفت خاصه A_i از رابطه R2 کلید خارجی در این رابطه باشد (که طبعاً باید در رابطه R1 کلید اصلی باشد)، این صفت خاصه در تاپلی از رابطهٔ R2 :

ـ می‌تواند مقدار NULL داشته باشد.

ـ در غیر اینصورت حتماً باید مقداری داشته باشد که در تاپلی از رابطه R1 موجود باشد.

مثال ۱۱ : S# در جدول S کلید اصلی و در جدول SP کلید خارجی است. همچنین P# در جدول P کلید اصلی و در جدول SP کلید خارجی است. همچنین P# در جدول P کلید اصلی و در جدول SP کلید خارجی است. حال اگر مثلاً بخواهیم تاپل (S5, P4, 300) را در جدول SP(S#, P#, Qty) درج کنیم، قاعده جامعیت ارجاعی به شرطی اجازه این کار را می‌دهد که قبلاً S5 در جدول S و P4 در جدول P وجود داشته باشد. یا مثلاً اگر بخواهیم مشخصات S6 را از جدول S حذف کنیم در حالیکه در جدول SP سطری شامل S6 وجود دارد، DBMS جلوی اینکار را خواهد گرفت و یا اینکه تمام سطرهای شامل S6 در جدول SP را نیز حذف می‌کند.

نکته : مدل رابطه‌ای اقلاً باید دارای سه جنبه اساسی زیر باشد :

۱ـ ساختار داده‌یی رابطه‌ای

۲ـ قواعد جامعیت

۳ـ امکان کار با داده‌ها یعنی وجود مجموعه‌ای از عملگرهای جبر رابطه‌ای (مثل اجتماع و گزینش و پرتو و پیوند و ... که در فصل بعدی بیان خواهیم کرد).

تعریف : سیستمی را رابطه‌ای گوئیم اگر و فقط اگر دارای دو جنبه باشد : ۱ـ بانک اطلاعاتی مبتنی بر رابطه‌ها به نحوی که کاربر بانک را به صورت مجموعه‌ای از جداول بیفتد. ۲ـ اقلاً دارای عملگرهای گزینش و پرتو و پیوند باشد بی‌آنکه عملکرد این عملگرها نیازی به وجود مسیرهای دستیابی فیزیکی از پیش تعریف شده داشته باشند.

قاعده اطلاعات

در بانک اطلاعتی رابطه‌ای قاعده اطلاعات به صورت زیر وجود دارد :

«تمام اطلاعات موجود در بانک اطلاعاتی فقط به یک روش نمایش داده می‌شوند، یعنی به صورت مقادیری در موقعیتهای ستونی از سطرهای جدول».

این روش نمایش تنها روش (البته در سطح منطقی) در سیستم رابطه‌ای است. یعنی در این سطح هیچ اشاره‌گری وجود ندارد. وقتی می‌گوئیم در سیستم رابطه‌ای اشاره‌گری وجود ندارد، منظور این نیست که اشاره‌گرها در سطح فیزیکی نمی‌توانند وجود داشته باشند. در سطح فیزیکی ممکن است اشاره‌گرها استفاده گردد.

رابطه‌هاي DUM , DEE

می‌دانیم که رابطه مجموعه‌ای از صفات و مجموعه‌ای از چندتایی‌هاست. در ریاضیات مجموعه تهی، مجموعه جالبی است که کاربردهای زیادی دارد. می‌خواهیم ببینیم معادل مجموعه تهی در مدل رابطه‌ای چه می‌شود.

رابطه می‌تواند مجموعه‌ای خالی از تاپلها باشد (مثل فایل بدون رکورد). البته هر رابطه مبنا وقتی که در ابتدا ایجاد می‌شود، مجموعه‌ای تهی از تاپلهاست. این رابطه را می‌توان رابطه خالی نامید ولی چندان مناسب نیست و ما در این قسمت با آن کاری نداریم.

رابطه می‌تواند مجموعه خالی از صفات باشد که مشابه مجموعه تهی در تئوری مجموعه‌ها و صفر در محاسبات معمولی دارای اهمیت است. رابطه‌ای که مجموعه خالی از صفات دارد (درجه صفر) خود به دو دسته DUM , DEE تقسیم می‌شود. رابطه صفر می‌تواند صفر یا حداکثر یک تاپل داشته باشد و آن تاپل صفر است. رابطه درجه صفر نمی‌تواند بیش از یک تاپل داشته باشد، چرا که تمام تاپلهای صفر تکرارهایی از یکدیگرند. لذا دقیقاً دو رابطه از درجه صفر وجود دارند :

Dee (یا TABLE – DEE) که فقط حاوی یک تاپل است.

DUM (یا RABLE – DUM) که هیچ تاپلی ندارد.

در فصل بعدی کاربرد این رابطه‌ها را در جبر رابطه‌ای بیان می‌کنیم. در اینجا فقط می‌گوئیم که DEE متناظر TRUE , DUM متناظر FALSE است.

نکته ۱ : کلید کاندید روابط DUM , DEE مجموعه‌ای تهیه از صفات { } (یا ϕ) می‌باشد. البته DUM , DEE تنها روابطی نیستند که می‌توانند دارای کلید کاندید تهی باشند.

نکته ۲ : اگر ϕ یک کلید کاندید برای رابطه R باشد آنگاه :

۱ـ این کلید تنها کلید کاندید رابطه R است : زیرا تهی زیر مجموعهٔ هر کلید کاندید دیگری است که معرفی می‌گردد.

۲ـ R حداکثر یک تاپل دارد : زیرا هر تاپل برای مجموعه تهی از صفات دارای مقدار یکسانی است.

فصل ششم

جبر رابطه‌اي

در جبر معمولی داده‌ها اعداد حقیقی و عملگرها + ، - ، ÷ ، × می‌باشد. عملگرهای دیگر نظیر توان برای ساده کردن عملگرهای ابتدایی مثل ضرب استفاده می‌شوند. در جبر منطقی نوع داده مجموعه {TRUE, FALSE} بوده و عملگرهایش NOT, OR, AND می‌باشد.

جبر رابطه‌ای در واقع مبنای تئوریک مدل رابطه‌ای است. به مجموعه‌ای از قوانین و عملگرها که امکان پردازش جداول را فراهم می‌سازند، جبر رابطه‌ای می‌گویند. نوع داده در جبر رابطه‌ای فقط رابطه است. یعنی ورودی و خروجی تمامی عملگرها رابطه می‌باشد.

عملگرها در جبر رابطه‌ای را می‌توان به چهار دسته تقسیم کرد :

۱ـ عملگرهای ساده شامل گزینش (Restrict , Select یا σ) و پرتو (Project یا π)

۲ـ عملگرهای مجموعه‌ای شامل اجتماع ($\cup$)، اشتراک ($\cap$) و تفاضل (-)

۳ـ عملگرهای پیوند شامل ضرب دکارتی (×) پیوند طبیعی (∞)، نیم پیوند (α)، پیوند شرطی (X_θ) و فرا پیوند.

۴ـ عملگرهای دیگر مثل نامگذاری (ρ)، تقسیم (÷)، جایگزینی ($\leftarrow$) و غیره.

خاصیت بسته بودن

این واقعیت که خروجی حاصل از هر عمل رابطه‌ای یک رابطه دیگر است، خاصیت بسته بودن یا بستار (Closure) نام دارد. مثلاً اجتماع دو رابطه (جدول) یک رابطه (جدول) می‌شود.

معنای بستار این است که می‌توانیم عبارات رابطه‌ای تو در تو بنویسیم.

خروجی عملگرهای مدل رابطه‌ای یک جدول است که اصطلاحاً گفته می‌شود عملیات به صورت مجموعه در یک زمان (set – at – a – time) می‌باشد و نه به صورت سطر در یک زمان (row – at – a – time) . سیستم‌های غیر رابطه‌ای به صورت row – at – a – time عمل می‌کنند.

لذا قابلیت پردازش مجموعه، خاصیت اصلی سیستم رابطه‌ای است.

تذکر : اکثر مثالهای این فصل را براساس جداول SP, P, S زیر بیان خاهیم کرد :

S#	P#	Qty تعداد
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

P#	Color	Type	City
P1	قرمز	آهن	تهران
P2	سبز	مس	تبریز
P3	آبی	برنج	شیراز
P4	قرمز	آهن	تهران

S#	Sname	City
S1	فن‌آوران	تهران
S2	ایران قطعه	تبریز

تذکر : در بسیاری از کتابها به جای متغیر رابطه‌ای از اصطلاح رابطه استفاده می‌شود. مثلاً در شکل فوق در واقع SP یک متغیر رابطه‌ای است.

عملگر گزینش

عملگر گزینش سطرهایی از جدول را می‌دهد. برای این عملگر در کتب مختلف از نمادهای σ، نام جدول جلوی علامت σ در پرانتز و شرط انتخاب سطرها زیر آنها نوشته می‌شود. در این عملگر تمام ستونهای آن جدول در خروجی می‌آید.

مثال ۱ : تاپلهایی را از رابطه S مشخص سازید که صفت شهر آنها تهران باشد.

خروجی (تهران، فن‌آوران، S1) خواهد بود. برای این خروجی یکی از دستورهای زیر را می‌توان نوشت :

الف) $\sigma_{city = \text{'تهران'}} (S)$

ب) Select S Where city = 'تهران'

ج) Restrict :

د) S Where city = 'تهران'

مثال ۲ : تاپلهایی را از رابطه SP مشخص سازید که S# برابر S1 و P# برابر P1 باشد.

خروجی S1, P1, 300 می‌باشد.

$$\sigma_{S\# = S1 \,\wedge\, PH = P1} (SP)$$

علامت ^ به معنای AND می‌باشد. دستور فوق به صورت زیر نوشته می‌شود :

Select SP where P# = P1 AND S# = S1

عملگر پرتو

عملگر پرتو یا تصویر ستونهایی از جدول را انتخاب می‌کند و برای آن هیچگونه شرطی اعمال نمی‌گردد. در خروجی پرتو سطرهای تکراری حذف می‌شوند. برای این عملگر از نمادهای II یا

Project استفاده می‌شود. هنگام استفاده از نماد II، ستونهای انتخاب شده زیر علامت II نوشته شده و نام جدول جلوی علامت II در پرانتز نوشته می‌شود.

مثال ۳ : ستون شهر را از جدول S چاپ کنید.

الف) Iicity (S)

ب) Project S[city]

ج) project :

S over city

خروجی

تهران

تبریز

مثال ۴ : ستونهای Qty و #P را از جدول SP نمایش دهید.

خروجی

$\Pi_{P\#,city}$ (SP)

P1	300
P2	200
P3	400
P2	400

تذکر : با ترکیب کردن عملگرهای σ و II می‌توان اطلاعات بیشتری را از جداول بدست آورد.

مثال ۵ : شماره تهیه‌کنندگان و قطعات تولیدی آنها را چاپ کنید به شرط آنکه قطعه تولیدی آنها بیشتر از 200 عدد باشد.

خروجی :

$\Pi_{S\#,P\#}$ ($\sigma_{Qty > 200}$ (SP))

S#	P#
S1	P1
S1	P3
S2	P1
S2	P2

عملگرهاي ∪، ∩ و -

اجتماع دو رابطه رابطه‌ای است که تاپلهایش در یک یا هر دو رابطه وجود دارند. یعنی رکوردهای ۲ جدول با هم ترکیب شده و رکوردهای تکراری یکبار نوشته می‌شوند. اجتماع را به یکی از دو صورت زیر نمایش می‌دهند :

$R = R1 \cup R2$ یا $R = R1$ Union R2

اشتراک دو رابطه رابطه‌ای است که تاپلهایش در هر دو رابطه وجود داشته باشند :

$R = R1 \cap R2$ یا $R = R1$ Intersect R2

تفاضل دو رابطه، رابطه‌ای است که تاپلهایش در رابطه اول موجود است ولی در رابطه دوم وجود ندارد.

R = R1 – R2 یا R = R1 Minus R2

مثال ۶ : لیست شهرهای تهیه کنندگان و قطعات را بدهید.

خروجی

تهران

تبریز $Ii_{city}(S) \cup II_{city}(P)$

شیراز

مثال ۷ : لیست شهرهای مشترک بین تهیه‌کنندگان و قطعات را بدهید.

خروجی

تهران $Ii_{city}(S) \cap II_{city}(P)$

تبریز

مثال ۸ : اسامی شهرهای قطعاتی را بدهید که در رابطه S وجود ندارند.

IIcity (P) – IIcity (S)

خروجی : شیراز می‌شود.

نکته ۱ : روابط ورودی برای عملگرهای ∪، ∩ و - باید همتا (Same ariy) باشند یعنی : تعداد صفتهای دو رابطه (تعداد ستونهای دو جدول) مساوی باشد و همچنین صفتها به ترتیب دارای دامنه‌های یکسان باشند.

نکته ۲ : عملگرهای اجتماع و اشتراک خاصیت جابجایی دارند ولی تفاضل این خاصیت را ندارد یعنی $B - A \neq A - B$

نکته ۳ : عملگرهای اجتماع و اشتراک خاصیت شرکت‌پذیری دارند ولی تفاضل این خاصیت را ندارد.

عملگرهاي پيوند يا الحاق (Join)

این عملگرها پر کاربرد و قدرتمند می‌باشند ولی گاهی اوقات باعث مصرف فضا و زمان زیادی می‌گردند لذا هنگام استفاده از آنها باید دقت کافی کرد. این عملگرها عبارتند از ضرب دکارتی (×)، پیوند طبیعی (∞)، نیم پیوند (α)، پیوند شرطی (X_θ) و فرا پیوند.

ضرب دكارتي (×)

ضرب دکارتی از گرانترین عملگرهای بانک رابطه‌ای است که زمان و فضای زیادی می‌خواهد و تا حد امکان باید از آن اجتناب کرد. حاصلضرب دو رابطه، رابطه‌ای است که تاپلهایش از الصاق هر یک از دو تاپل دو رابطه بدست می‌آیند. به عبارت دیگر در $R1 \times R2$ هر سطر R1 را پشت تمام سطرهای R2 قرار می‌دهیم. ضرب دکارتی را به صورتهای زیر نشان می‌دهند :

$R = R1 \times R2$ $\qquad$ $R = R1$ TIMES $R2$

مثال ۹ : حاصلضرب دکارتی دو رابطه S , P چه می‌شود ؟

S#	Sname	S.City	P#	Color	Type	P.city
S1	فن‌آوران	تهران	P1	قرمز	آهن	تهران
S1	فن‌آوران	تهران	P2	سبز	مس	تبریز
S1	فن‌آوران	تهران	P3	آبی	برنج	شیراز
S1	فن‌آوران	تهران	P4	قرمز	آهن	تهران
S2	ایران قطعه	تبریز	P1	قرمز	آهن	تهران
S2	ایران قطعه	تبریز	P2	سبز	مس	تبریز
S2	ایران قطعه	تبریز	P3	آبی	برنج	شیراز
S2	ایران قطعه	تبریز	P4	قرمز	آهن	تهران
S3	پولادین	تبریز	P1	قرمز	آهن	تهران
S3	پولادین	تبریز	P2	سبز	مس	تبریز
S3	پولادین	تبریز	P3	آبی	برنج	شیراز
S3	پولادین	تبریز	P4	قرمز	آهن	تهران

نتیجه شامل همه فیلدهای دو رابطه است. از آنجا که هر دو رابطه حاوی فیلدی به نام City می‌باشند لذا برای جلوگیری از ابهام و اشتباه آنها از S.City و P.City نامگذاری می‌کنند.

توجه کنید که جدول فوق 12 (12 = 4 × 3) سطر و 7 (7 = 4 + 3) ستون دارد.

نکته ۱ : اگر جدول A دارای m سطر و n ستون و جدول B دارای p سطر و q ستون باشد آنگاه $A \times B$ تعداد $m \times p$ سطر و تعداد $n + q$ ستون خواهد داشت.

نکته ۲ : ضرب دکارتی در ریاضیات مجموعه‌ها خاصیت جابجایی ندارد یعنی ($A \times B \neq B \times A$) ولی در جبر رابطه‌ای چون ترتیب ستونها مهم نیست خاصیت جابجایی دارد یعنی ($A \times B = B \times A$)

نکته ۳ : ضرب دکارتی خاصیت شرکت‌پذیری نیز دارد یعنی :

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

مثال ۱۰ : خروجی دستور $\Pi_{S\#.type}\ (S \times P)$ چه می‌شود؟

جواب :

S#	Type
S1	آهن
S1	مس
S1	برنج
S2	آهن
S2	مس
S2	برنج
S3	آهن
S3	مس
S3	برنج

پیوند طبیعي (∞)

این عملگر دو رابطه را بر مبنای یک یا چند فیلد مشترک به هم پیوند می‌دهد و رکوردهایی از دو جدول را که مقدار آن فیلد مشترک برای آنها یکسان است به هم می‌چسباند. به عبارتی دیگر فقط سطرهایی از دو جدول را کنار هم می‌دهد که همه ستونهای همنام آن دو جدول مقادیر مساوی داشته باشند. ستونهای هم نام فقط یکبار در خروجی ظاهر می‌شوند.

پیوند دو جدول A , B را به صورتهای زیر نمایش می‌دهند :

الف) A ∞ B

ب) A JOIN B

ج) Join :

A and B Over k (k نام ستون مشترک است)

مثال ۱۱ : خروجی SP ∞ S را بدست آورید.

ستون مشترک بین S , SP فیلد S# می‌باشد پس هر سطر جدول S را برداشته پشت سطرهایی از جدول SP می‌گذاریم که S# آنها با هم برابر باشند :

S#	Sname	City	P#	Qty
S1	فن‌آوران	تهران	P1	300
S1	فن‌آوران	تهران	P2	200
S1	فن‌آوران	تهران	P3	400
S2	ایران قطعه	تبریز	P1	300
S2	ایران قطعه	تبریز	P2	400

200	P1	تبریز	پولادین	S3

شما به عنوان تمرین خروجی $S \infty P$ را بدست آورید.

نکته ۱ : پیوند طبیعی خاصیت شرکت‌پذیری و جابجایی دارد یعنی :

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

نکته ۲ : اگر دو جـــــدول فیلد همنام نـــــداشته باشند در اینصورت $R1 \infty R2$ تبدیل به $R1 \times R2$ می‌شود.

نکته ۳ : اگر تمام فیلدهای دو جدول یکسان باشند آنگاه $R1 \infty R2$ معادل $R1 \cap R2$ خواهد بود.

نکته ۴ : پیوند طبیعی عملگر گرانی نیست و زمان و فضای کمتری نسبت به ضرب دکارتی می‌گیرد.

مثال ۱۲ : حاصل $S \infty$ S# / S1 / S2 چه می‌شود؟

جواب :

S#	Sname	City
S1	فن‌آوران	تهران
S2	ایران	تبریز

مثال ۱۳ : اسامی تهیه کنندگان قطعه P2 را بدهید.

خروجی

فن‌آوران

ایران قطعه $SP)) \infty_{p\# = P2}(S \sigma$ Iisname (

پولادین

تذکر : در بعضی از کتابها $SP \infty S$ را به صورت زیر نمایش می‌دهند :

Join :

S and SP over S#

تعریف : اگر Cjoin کاردینالیتی رابطه حاصل از پیوند دو رابطه و Ccart کاردینالیتی رابطه حاصل از ضریب کارتزین دو رابطه باشد، در اینصورت Cjoin/Ccart را ضریب گزینش عملگر پیوند (Join Selectivity Factor) می‌نامیم.

عملگر جایگزینی

با علامت ← جدول حاصل از دستورات ذخیره می‌شود تا در ادامه کار مورد استفاده قرار گیرد. اگر دستوری طولانی باشد می‌توان با استفاده از جایگزینی، آن را در چند مرحله نوشت. بعضی از کتابها از نماد = : و بعضی دیگر از عبارت GIVING برای این منظور استفاده کرده‌اند.

مثال ۱۴ : اسامی تهیه کنندگان قطعه P2 را بدهید :

S Join SP Giving Templ

Select Temp1 Where P# “P2” Giving Temp2

Project Temp2 [Sname]

و با نمادی دیگر :

$SP \infty S \leftarrow Temp1$

$\sigma_{P\#='P2'}(Temp1) \leftarrow Temp2$

$\Pi_{Sname}(Temp2)$

البته دستور فوق را می‌توان در یک خط به صورت زیر نوشت :

$\Pi_{Sname}(\sigma_{P\#='P2'}(S \infty SP))$

خروجی دستورات فوق فن‌آوران، ایران قطعه و پولادین می‌شود.

عملگر پیوند شرطي (JOIN - θ)

این عملگر، زیر مجموعه‌ای از ضرب دکارتی است که شرط θ روی سطرهای آن اعمال شده باشد. ستونهای خروجی معادل ستونهای ضرب دکارتی است. در بعضی کتابها آن را به صورت $X\theta$ نمایش داده‌اند که θ شرط موردنظر می‌باشد.

مثال ۱۵ : خروجی این دستور را بدست آورید :

$P.City\ P \geq S\ X_{s.City}$

S#	Sname	S.City	P#	Color	Type	P.City
S1	فن‌آوران	تهران	P1	قرمز	آهن	تهران
S1	فن‌آوران	تهران	P2	سبز	مس	تبریز
S1	فن‌آوران	تهران	P4	قرمز	آهن	تهران
S2	ایران قطعه	تبریز	P2	سبز	مس	تبریز
S3	ایران قطعه	تبریز	P2	سبز	مس	تبریز

اگر شرط θ به صورت = بیان شود آنگاه پیوند شرطی به پیوند طبیعی (∞) تبدیل می‌گردد. در واقع پیون دطبیعی که قبلاً بیان شد مشابه پیوند شرطی است با این تفاوت که :

۱ـ خود به خود شرط تساوی روی همه ستونهای همنام دو جدول اعمال می‌شود یعنی فقط سطرهایی از دو جدول انتخاب می‌شوند که همه ستونهای همنام آن دو جدول مقادیر مساوی داشته باشند.

۲ـ ستونهای تکراری فقط یکبار در خروجی ظاهر می‌شوند. ۳ـ نرم‌افزارهایی شبیه SQL هیچگاه جدول ترکیبی را برای ∞ تشکیل نمی‌دهند تا سطرهای همتراز را جدا کنند، بلکه با استفاده از روشهایی مانند مرتب کردن و شاخص‌گذاری کار را بسیار ساده می‌کنند.

به این دلیل است که پیوند طبیعی عملگر گرانی نیست و زمانی و فضای آن بسیار کمتر از ضرب دکارتی است.

تذکر : در بعضی کتابها این عملگر به نام الحاق یا پیوند (θ یا theta) و به صورت زیر معرفی شده است :

$Y\theta$ (A TIMES B) WHERE X

مثال ۱۶ : معادل عبارت $P \geq S X_{s.city}$ $_{P.city}$ به صورت زیر است :

P.City≥(S TIMES P) WHERE S.City

پس عملگر θ وقتی به کار می‌رود که می‌خواهیم دو رابطه را بوسیله عملگرهای غیر از تساوی با هم الحاق کنیم.

تذکر ۱ : در قرارداد Tutorial D (که برای نوشتن عبارت جبری در کتاب دیت استفاده شده) مستقیماً عملگر θ (الحاق) پشتیبانی نمی‌شود چرا که : ۱ـ در عمل چندان به کار نمی‌آید. ۲ـ یک عملگر اولیه نیست.

تذکر ۲ : شرط θ زیر $X\theta$ می‌تواند شامل AND نیز باشد.

تذکر ۳ : در بعضی کتابها نمادهای زیر نیز معرفی شده‌اند :

Theta	نوع پیوند	عملگر
=	پیوند به شرط تساوی	E-JOIN

$>$	پیوند به شرط بزرگتر	G-Join
$<$	پیوند به شرط کوچکتر	L – JOIN
$\geq$	پیوند به شرط بزرگتر یا مساوی	GE – JOIN
$\leq$	پیوند به شرط کوچکتر یا مساوی	LE – JOIN

مثال ۱۷ : دو دستور زیر معادل یکدیگرند :

P.City≥(S TIME P) WHERE S.City

S GE-JOIN P

تذکر ۴ : عبارت S JOIN P هم ارز عبارت پیچیده زیر است :

((S TIMES (P RENAME CITY AS PCITY))

WHERE CITY = PCITY {ALL BUT PCITY}

Rename برای تغییر نام است که جلوتر شرح می‌دهیم.

عملگر نیم پیوند (Semi – join یا شبه‌الحافي)

این عملگر مشابه پیوند طبیعی است با این تفاوت که فقط ستونهای جدول اول را می‌دهد. بعضی کتابها این عملگر را با نماد α نشان داده‌اند و بعضی دیگر از نماد SEMIJOIN استفاده کرده‌اند.

مثال ۱۸ : مشخصات قطعاتی را بدست آورید که در تهران تهیه شده و تعداد آنها از ۲۰۰ بیشتر باشد.

خروجی :

P#	Color	Type	City

$$_{city}\sigma\,(\text{'تهران'} \text{ ، } \text{Qty} > 200\ (\text{SP}))\,\sigma\ (\alpha(\text{P}))$$

تهران	آهن	قرمز	P1

نکته ۱ : ممکن است تعداد سطرهای خروجی به مراتب کمتر از پیوند طبیعی باشد، زیرا با کنار رفتن چند ستون، سطرهای تکراری پدید می‌آیند و حذف می‌شوند.

نکته ۲ : برخلاف سایر عملگرهای این بخش، ترتیب جداول ورودی در نیم پیوند مهم است یعنی ($A \alpha B \neq B \alpha A$) چرا که همواره ستونهای جدول اول را می‌دهد.

نکته ۳ : کاربرد اصلی عملگر نیم پیوند در بانکهای نامتمرکز است. اگر جدول A در سایت اول و جدول B در سایت دوم خیره شده باشد و دستور $B \alpha A$ را در سایت اول صادر کنیم، ضمن اینکه ارزش پیوند طبیعی را دارد از انتقال داده‌های جدول B از سایت دوم به سایت اول پرهیز می‌کند.

مثال ۱۹ : دستور زیر S# و Sname، Status و City مربوط به عرضه‌کنندگانی را مشخص می‌کند که قطعه P2 را عرضه کرده‌اند :

S SEMIJOIN (SP WHERE P# = ‘P2’)

عملگرهای فراپیوند (Outer Join)

نوع دیگری از عملگر پیوند که معمولاً به شرط تساوی است عملگرهای فراپیوند (یا پیوند خارجی) می‌باشد. عملگرهای فراپیوند به سه دسته زیر تقسیم می‌شوند :

۱ـ فراپیوند چپ (Left Outer Join) با نماد L-O-JOIN

۲ـ فراپیوند راست (Right Outer Join) با نماد R-O-JOIN

۳ـ فراپیوند کامل (Full Outer Join) با نماد F-O-JOIN

فراپیوند چپ گونه‌ای از عملگر پیوند طبیعی (∞) است با این تفاوت که علاوه بر تاپلهای پیوند شدنی از دو رابطه، تاپلهای پیوندنشدنی از رابطه چپ هم، پیوند شده با مقادیر NULL، در جواب وارد می‌شوند. در حالت فراپیوند راست، تاپلهای پیوندنشدنی از رابطه سمت راستی، پیوند شده با مقادیر NULL، در جواب وارد می‌شوند. فرا پیوند کامل هر دو حالت راست و چپ را شامل می‌شود.

مثال ۲۰ : جدول SP , S زیر را با هم پیوند چپ دهید.

جدول SP

S#	P#	Qty
S1	P1	200
S2	P3	400
S2	P4	100
S3	P1	300

جدول S

S#	Sname	Status
S1	Sn1	10
S2	Sn2	15
S3	Sn3	10
S4	Sn4	20

$\Rightarrow$S L-O-JOIN SP

S#	Sname	Status	P#	Qty
S1	Sn1	10	P1	200
S2	Sn2	15	P3	400
S2	Sn2	15	P4	100
S3	Sn3	10	P1	300
S4	Sn2	20	NULL	NULL

کاربرد این عملگر را با یک مثال نشان می‌دهیم.

مثال ۲۱ : دو جدول استاد (PROF) و گروه آموزشی (DEPT) را در نظر بگیرید :

DEPT

شماره گروه آموزشی	عنوان گروه	دفتر گروه
D11	نرم‌افزار	اتاق 115
D22	سخت‌افزار	اتاق 213
D33	هوش مصنوعی	اتاق 117

PROF

شماره‌گروه آموزشی	نام استاد	شماره استاد
D11	امیری	11
D22	اکبری	25
NULL	سلطانی	39
D33	جوادی	46
D11	سعیدی	53
NULL	غلامی	42

حال فرض کنید نام تمام استادان و عنوان گروه آموزشی آنها را بخواهیم. اگر برای پاسخگویی به این سئوال DEPT ∞PROF را بدست آوریم، نام استادان سلطانی و غلامی در رابطه حاصل وجود ندارد و در نتیجه بخشی از اطلاعات از بین می‌رود. برای حل این مشکل باید به جای عملگر پیوند طبیعی از عملگر فراپیوند چپ استفاده کنیم :

(PRF L-O-JOIN DEPT) عنوان گروه و نام استاد II

عملگر شبه تفاضل یا نیم تفاضل

این عملگر به صورت زیر تعریف می‌شود :

A SEMIMINUS B = A MINUS (A SEMIJOIN B)

یعنی سطرهایی را از جدول A می‌دهد که نباید در B همتایی داشته باشند.

مثال ۲۲ : مشخصات عرضه‌کنندگانی را بدهید که قطعه P2 را تولید نمی‌کنند :

S SEMIMINUS (SP WHERE P# = 'P2’)

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

عملگر تغییر نام یا نام‌گذاري

این عملگر به صورت $p_b a$ نشان داده شده که نام b روی جدول a نیز گذاشته می‌شود. در این حال بدون ذخیره‌سازی مجدد یک جدول می‌توان از آن دوبار استفاده کرد، در واقع اشاره‌گر جدیدی تعریف می‌شود. محدوده عملکرد p تنها در همان دستور مربوطه است، یعنی پس از اتمام آن دستور، نام جدید دیگر وجود ندارد. گاهی یک پرس و جو، دو یا چند بار به یک جدول نیاز دارد.

مثال ۲۳ : جدول اساتید را با مشخصات زیر در نظر بگیرید :

prof (pn, O#, esp, degree, clg#)

(شماره دانشکده، مدرک، تخصص، شماره دفتر، نام استاد) جدول استاد

حال شماره دفتر و نام اساتیدی را بدهید که دفتر کارشان مشترک است :

$$\pi_{pn,o\#}(prof)\,(\rho_{k.pn \neq prof} X_{prof.O\# = k.o\#\hat{}prof.pn})$$

ابتدا ستونهای نام استاد و دفتر او از جدول prof جدا شده و با نام نامگذاری می‌شود. سپس سطرهایی از prof که با k دفتر کار یسکان ولی نام متفاوتی دارند، انتخاب می‌شوند. توجه کنید که همه اساتید با خودشان هم اطاق هستند لذا برای جلوگیری نمایش این اطلاعات بدیهی شرط $k.pn \neq prof\ .\ pn$ را با شرط زیر X ترکیب AND کرده‌ایم.

در کتاب دیت عملگر RENAME معرفی شده که هدف آن تغییر نام صفات در یک رابطه است. به بیانی دیگر عملگر RENAME رابطه‌ای را گرفته، رابطه‌ای یکسان با آن را بر می‌گرداند با این تفاوت که حداقل یکی از صفاتش نام دیگری داشته باشد.

مثال ۲۴ : عبارت S RENAME CITY AS SCITY رابطه‌ای را ایجاد می‌کند که عنوان و بدنه آن مانند رابطه S است با این تفاوت که نام صفت شهر به جای CITY برابر با SCITY است. توجه داشته باشید که عبارت فوق، متغیر رابطه‌ای S را در بانک اطلاعاتی تغییر نداده است.

مثال ۲۵ : عبارت زیر را تغییر نام چندگانه را نشان می‌دهد :

P Rename Pname AS PN , Weight AS WT

عملگر تقسیم (÷ یا DIVIDEBY)

تقسیم کاربرد بسیار ارزشمندی دارد ولی متاسفانه در زبانهای متداول بانک اطلاعاتی مستقیماً پیاده‌سازی نشده است و به سختی می‌توان آن را معادل‌سازی کرد. کاربرد عملگر تقسیم زمانی است که بخواهیم همه حالتهای یک اتفاق را بررسی کنیم مثل حالتهای زیر :

ـ دانشجویانی که همه درسهای استاد اکبری را گرفته‌اند.

ـ درسهایی که توسط همه دانشکده‌ها ارائه می‌شوند.

ـ اسامی تهیه کنندگانی که تمام قطعات را تهیه می‌کنند.

پاسخگویی به پرس و جوهای فوق با استفاده از دستور تقسیم بسیار ساده و بدون آن بسیار مشکل است. ابتدا بخشی را که شامل شرط همه می‌شود پیدا می‌کنیم (مقسوم) سپس بخش دیگر را بر آن تقسیم می‌کنیم.

در آن بخش حتماً باید صفت‌های مندرج در مقسوم علیه وجود داشته باشند و خروجی شامل صفتهای باقی مانده خواهد بود.

مثال ۲۶ : اگر R1 و R2 به صورتهای زیر باشند :

خروجی R1 ÷ R2 چه می‌شود؟

R1

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

R2

P#
P1
P2
P4

جواب :

S#
S1
S2

مثال ۲۷ : اگر بخواهیم R1 , R2 = همان مقدار مثال قبلی را داشته باشد خـــــروجی R2 ÷ R1 چه می‌شود؟ (یعنی کدام تهیه‌کنندگان همه P2 و P4 را تهیه کرده‌اند؟)

جواب :

S#
S1
S4

مثال ۲۸ : R2 = باشد و R1 همان رابطه مثال قبلی باشد، R1 ÷ R2 چه می‌شود؟

P#
P1
P2
P3
P4
P5
P6

(یعنی کدام تهیه‌کنندگان تمام قطعات را تهیه کرده‌اند؟)

جواب :

S#
S1

تذکر : بعضی کتابها از نمادهای DIVIDEBY , DIV به جای ÷ استفاده کرده‌اند.

تذکر : دو رابطه $R_1(\overbrace{A_1, A_2,\ldots, A_n}^{Z}, \overbrace{B_1, B_2, \ldots, B_m}^{X})$ و $R_2(B_1,B_2,\ldots,B_m)$ را در نظر می‌گیریم. شرط تقسیم این است که $Z \subseteq X$ باشد. اگر $Y = Z - X$ باشد، حاصل عبارت R1(Z) DIV R2(X) رابطه R3(Y) می‌شود.

تقسیم سه عملوندي

اکثر کتابها تقسیم دو عملوندی را مشابه فوق تعریف کرده‌اند (همان تعریف اولیه که Codd ارائه کرده) ولی کتاب دیت ۲۰۰۰، فرم اصلاح شده‌ای از تقسیم را با ۳ عملوند معرفی کرده است که مشکلات مربوط به عملگر تقسیم قدیمی را در مورد رابطه‌های خالی حل کرده است و به صورت زیر است :

فرض کنید عنوان رابطه‌های $A(\overbrace{X_1,X_2,\ldots X_m}^{X})$ و $B(\overbrace{Y_1,Y_2,\ldots,Y_n}^{Y})$ به صورت X , Y باشند.

و فرض کنید عنوان رابطه C اجتماع عنوانهای A , B است :

$$C\ \{\overbrace{X_1,X_2,\ldots,X_n, Y_1,Y_2,\ldots,Y_n}^{Z}\}$$

آنگاه تقسیم A بر B در C، که A مقسوم،B علیه و C میانجی (Mediator) است، به صورت زیر نمایش داده می‌شود :

A DIVIDEBY B PER C

حاصل این تقسیم رابطه‌ای است که عنوان آن {X} و بدنه آن متشکل از آن مقادیر X از A است که مقادیر Y متناظر با آنها در C حاوی تمام مقادیر Y از B هستند.

مثال ۲۹ : در شکل زیر DEND مقسوم، MED میانجی و DOR مقسوم‌علیه است. برای سه حالت DOR سه جواب را نشان داده‌ایم.

DEND

S#
S1
S2
S3
S4
S5

MED

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
…	

DOR

P#
P1

DOR

P#
P2
P4

DOP

P#
P1
P2
P3
P4
P5
P6

DEND DIVIDEBY DOR PER MED

S#
S1
S2

S#
S1
S4

S#
S1

این مثال را با ۳ مثال اول بحث تقسیم مقایسه کنید.

چند مثال پیچیده

مثال ۳۰ : اسامی تهیه کنندگانی را بیابید که اقلا یک قطعه قرمز رنگ تهیه می‌کنند.

جواب نهایی فن‌آوران و ایران‌قطعه می‌باشد. ابتدا باید در جدول P ببینیم قطعات قرمز رنگ کدامند که جواب P1 , P4 می‌شود. سپس باید به سراغ جدول SP برویم و ببینیم قطعات P4 P1 , را چه شرکتهایی تهیه کرده‌اند که جواب S1 , S2 می‌شود. در آخر باید به سراغ جدول S برویم و ببینیم نام S1 , S2 چیست، که جواب فن‌آوران و ایران قطعه می‌شود.

Select P Where Color = 'Red' Giving Temp1
Project Temp1 [P#] Giving Temp2
Temp2 Join SP Giving Temp3
Project Temp3 [S#] Giving Temp4
Temp4 Join S Giving Temp5
Project Temp5 [Sname] Giving Temp6

نماد دیگر :

$\sigma_{color = 'Red'}(P) \leftarrow Temp1$

$\Pi_{P\#}(Temp1) \leftarrow Temp2$

$(temp2 \leftarrow Temp3) \infty SP)$

$\Pi_{S\#}(Temp3) \leftarrow Temp4$

$(Temp4 \leftarrow Temp5) \infty S)$

$\Pi_{Sname}(Temp5) \leftarrow Temp6$

Temp2

P1
P2

Temp4

S1
S2

Temp6

فن‌آوران
ایران قطعه

دستورات فوق معادل است با :

$$S] \infty SP)) \infty_{color = 'Red'} (P)) \sigma \Pi_{sname} [(\Pi_{S\#} (\Pi_{P\#}($$

مثال ۳۱ : اسامی تهیه کنندگانی را بدهید که تمام قطعات را تهیه می‌کنند.

Project SP [S# , P#] Giving Temp1

Project P [P#] Giving Temp2

Temp1 DIVIDEBY Temp2 Gicing Temp3

S JOIN Temp3 Giving Temp4

Project Temp4 [Sname] Giving Temp5

نماد دیگر :

$$(\Pi_{S\#,P\#} (SP) \infty \Pi_{sname} (S \div \Pi_{P\#}(P))$$

مثال ۳۲ : اسامی تهیه کنندگانی را بیابید که قطعه P2 را تهیه نمی‌کنند.

Project S[S#] Giving Temp1

Select SP Where P# = ‘P2’ Giving Temp2

Project Temp2 [S#] Giving Temp3

Temp1 Minus Temp3 Gicing Temp4

Temp4 Join S Giving Temp5

Project Temp5 [Sname] Giving Temp6

نماد دیگر :

$$\Pi_{S\#} (S) \leftarrow Temp1$$

$$_{p\# ,'P2'} (SP) \sigma \leftarrow Temp2$$

$$\Pi_{S\#} (Temp2) \leftarrow Temp3$$

$$Temp1 - Temp3) \leftarrow Temp4$$

$$S \infty Temp4 \leftarrow Temp5$$

$$\Pi_{Sname} (Temp5) \leftarrow Temp6$$

$\Pi_{sname}([\Pi_{S\#}(S) - (\Pi_{S\#}(\sigma_{p\#,'P2'}(SP)))] \infty S]$

عملگر بسط (EXTEND) یا گسترش

عملگر EXTEND یک رابطه را گرفته رابطه دیگری را بر می‌گرداند که همانند رابطه اولیه است، با این تفاوت که حاوی صفت دیگری است که مقادیر آن با ارزیابی یک عبارت محاسباتی به دست می‌آید.

مثال ۳۳ : EXTEND P ADD (Weight * 454) AS GMWT

اگر P به صورت زیر باشد حاصل دستور فوق رابطهٔ سمت راستی زیر می‌شود :

P#	Pname	Weight
P1	Nut	12.0
P2	Bolt	17.0
P3	Screw	17.0

→

P#	Pname	Weight	GMWT
P1	Nut	12.0	5448.0
P2	Bolt	17.0	7718.0
P3	Screw	17.0	7718.0

نکته مهم : توجه داشته باشید که این عبارت EXTEND، رابطه قطعات (P) را در بانک اطلاعاتی تغییر نداده است. حاصل عبارت EXTEND را می‌توان در عبارت دیگری به صورت تو در تو استفاده کرد.

مثال ۳۴ :

```
((EXTEND P ADD (WEIGHT * 454) AS GMWT)
Where GMWT > 10000)  {ALL BUT GMWT}
```

مثال ۳۵ : EXTEND (P JOIN SP) ADD (Weight * Qty) AS SHIPWT

اگر جدول P مشابه مثال ۳۱ و جدول SP به صورت زیر باشد خروجی مثال فوق را بدست می‌آوریم :

SP

S#	P#	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300

خروجی

P#	Pname	Weight	S#	QTY	SHIPWR
P1	Nut	12	S1	300	3600
P1	Nut	12	S2	300	3600
P2	Bolt	17	S1	200	3400
P3	Screw	17	S1	400	6800

نکته ۱ : RENAME یک عملگر اولیه نیست و می‌تواند بر حسب EXTEND و عملگر تصویر ساخته شود.

مثال ۳۶ : دستور (EXTEND S ADD CITY AS SCITY) {ALL BUT CITY}

معادل عبارت زیر است : S RENAME CITY AS SCITY

مثال ۳۷ : EXTEND P AND Weight *454 AS GMWT,Weight*16AS OZWT

نکته ۳ : در این عملگر می‌توان از توابع SUM (جمع)، AVG (میانگین)، MAX(حداکثر)، MIN (حداقل) و نیز COUNT (تابع شمارشگر تاپلها) نیز استفاده کرد.

مثال ۳۸ :

```
EXTEND S
ADD COUNT ((SP RENAME S# AS X) WHERE X = S#
AS NP
```

خروجی :

S#	Sname	City	NP
S1	فن‌آوران	تهران	3
S2	ایران قطعه	تبریز	2

1	تبریز	پولادین	S3

عملگر خلاصه یا گروه‌بندي (SUMMARIZE)

این عملگر تک عملوندی است و تاپلهای رابطه را بر حسب مقادیر یک (یا بیش از یک) صفت گروه‌بندی می‌کند و آنگاه محاسبه‌ای روی مقادیر صفت دیگری از هر گروه انجام می‌دهد.

مثال ۳۹ : تعداد تهیه شده از هر قطعه را بدهید.

یعنی به جدول SP نگاه کرده و خروجی زیر را پدید آورید.

P#	TOYQTY
P1	600
P2	800
P3	400

SUMMARIZE SP PER SP{P#} ADD SUM (Qty) AS TOTQTY

نماد فوق، نمادی است که در کتاب دیت استفاده شده است.

در بعضی کتابها به جای کلمه کلیدی PER از کلمه کلیدی BY استفاده شده است و عبارت فوق به صورت زیر نوشته می‌شود.

SUMMARIZE SP BY(P#) ADD SUM(Qty) AS TOTQTY

مثال ۴۰ : خروجی این دستور چیست؟

SUMMARIZE (P JOIN SP) PER P{CITY} ADD COUNT AS NSP

در واقع این دستور مشخص می‌سازد چند محموله در هر شهر نگهداری می‌شود.

City	NSP
تهران	2
تبریز	3
شیراز	1
تهران	0

نکته ۱ : خلاصه‌سازی چندگانه نیز امکان‌پذیر است.

مثال ۴۱ : عبارت زیر درست است و جدولی ۳ ستونه از جمع مقادیر تهیه شده و متوسط مقادیر تهیه شده هر قطعه را می‌دهد :

SUMMARIZE SP PER P{P#} ADD SUM(Qty) AS TOTQTY, AVG(QTY) AS AVGQTY

خروجی :

P#	TOTQTY	AVGOTY
P1	600	300
P2	800	266.6
P3	400	400

نکته ۲ : SUMMARIZE یک عملگر اولیه نیست یعنی با استفاده از EXTEND ساخته می‌شود.

مثال ۴۲ : عبارت SUMMARIZE SP PER S{S#} ADD COUNT AS NP شکل ساده‌ای از عبارت زیر است :

(EXTEND S{S#}
ADD ((SP RENAME S# AS X) WHERE X = S#) AS Y.
COUNT(Y) AS NP) {S#NP}

مقایسه‌های رابطه‌ای

جبر رابطه‌ای که در ابتدا تعریف شد، روش مستقیمی برای مقایسه دو رابطه ارائه نکرده بود. ولی بعداً شرط جدیدی به نام «مقایسه رابطه‌ای» به صورت $R2\,\theta\,R1$ تعریف شد که θ می‌تواند یکی از موارد زیر باشد :

(نامساوی ≠) (مساوی =)

(زیرمجموعه مناسب <) (زیر مجموعه ≤)

(فوق مجموعه مناسب >) (فوق مجموعه ≥)

نکته ۱ : انتخاب نمادهای عملگر ممکن است معقول نباشد، زیرا نقیض « A زیر مجموعه مناسبی است» یقیناً این جمله نیست که « A فوق مجموعه B است». یعنی > و ≥ معکوس یکدیگر نیستند.

مثال ۴۳ : معنای عبارت S{City} = P{City} این است که «آیا تصویر رابطه عرضه کنندگان بر روی صفت City همانند تصویر رابطه قطعات بر روی صفت City است؟»

مثال ۴۴ : مفهوم عبارت S{S#} > SP{S#} این است که آیا عرضه کننده‌ای وجود دارد که هیچ قطعه‌ای را عرضه نکند؟

نکته ۲ : یکی از مقایسه‌های رابطه‌ای که اغلب مورد نیاز است، تست این موضوع است که آیا مقدار رابطه‌ای خالی است یا خیر. برای این کار میانبر IS-EMPTY معرفی شده است.

مثال ۴۵ : عبارت IS-EMPTY(SP) اگر جدول SP خالی باشد مقدار TRUE و در غیر اینصورت مقدار FALSE بر می‌گرداند.

مجموعه کامل عملگرها

بعضی از عملگرهای جبر رابطه‌ای، مبنایی (اصلی) هستند یعنی آنها یک مجموعه کامل است و هر عملگر دیگر را می‌توان بر حسب عملگرهای این مجموعه بیان کرد. این مجموعه کامل عبارتست از :

$$\leftarrow , - , x , \cup , II , \sigma$$

عملگرهای دیگر را می‌توان بر حسب عملگرهای فوق بدست آورد. پس مثلاً عملگرهای زیر اضافی هستند :

$$\text{و فراپیوند} \quad , \quad X_{\theta} , \alpha , \infty , \rho \div \cap ,$$

در زیر برخی از عملگرهای غیرمبنایی را بر حسب عملگرهای مبنایی بیان می‌کنیم.

$$B = A - (A - B) = B - (B - A) \cap 1) \; A$$

$$2) \; A(Y , X) \div B(X) = A[Y] - ((A[Y] \times B) - A) \; [Y]$$

عبارت ÷ در فرمول فوق تقسیم دو عملوندی می‌باشد.

$$3) \; A \; JOIN \; B = (A \; TIMES \; (B \; RENAME \times AS \; px) \; Where \; x = px \; \{ALLBUT \; x\}$$

در فرمول فوق x ستونی مشترک در دو رابطه A , B می‌باشد.

همانطور که از فرمول فوق مشاهده می‌شود، عمل پیوند طبیعی به کمک پرتو، گزینش و ضرب کارتزین قابل انجام است.

مثال ۴۶ : $SP \infty S$ را که بر مبنای صفت خاصه مشترک S# با هم پیوند می‌خورند را می‌تان با دستورات زیر نیز انجام داد :

$$S.S\# = SP.S\# \; (S \sigma \; SP = II_{ss = sname.P\#Qty} \; (\infty S \times SP))$$

کامل بودن جبر رابطه‌اي

دیدیم که با ترکیب عملگرها می‌توان عبارت جبری نوشت و حاصل ارزیابی هرعبارت معتبر، باز هم یک رابطه است. بنابراین می‌گوئیم جبر رابطه‌ای، از نظر رابطه‌ای کامل است و به عبارتی

دیگر، اکمال رابطه‌ای (Relationally Comptlteness) دارد. به خاطر این خاصیت، معمولاً از جبر رابطه‌ای به عنوان محک تشخیص اکمال رابطه‌ای استفاده می‌شود.

به این معنا که زبانی دارای اکمال رابطه‌ای است که حداقل همتوان با جبر رابطه‌ای باشد، یعنی هر رابطه‌ای که با عبارت جبر رابطه‌ای قابل تعریف باشد، توسط آن زبان هم تعریف شدنی باشد.

خواص و فرمولهاي عملگرها

$$A \times B = B \times A$$

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

$$A \infty B = B \infty A$$

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

$$A \cup B = B \cup A$$

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

$$A \cap B = B \cap A$$

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

$$_p(B) \sigma \cup_p(A) \sigma B) = \cup_p(A \sigma$$

$$_p(B) \sigma \cap_p(A) \sigma B) = \cap_p(A \sigma$$

$$_p(B) \sigma_p(A) - \sigma_p(A) - B = \sigma_p(A - B) = \sigma$$

منظور از P زیر σ یک شرط است.

عملگر تفریق (-) خاصیت جابجایی و شرکت‌پذیری ندارد.

عمل ضرب دکارتی در تئوری مجموعه‌ها در ریاضیات، نه شرکت‌پذیر است و نه جابجا پذیر اما نسخه‌ای از آن که در جبر رابطه‌ای تعریف شده است هر دو خاصیت را دارد.

دو دستور زیر معادلند (R نام رابطه و C1 , C2 شرط می‌باشند)

Select R Where C1 and C2

(Select R Where C1) Intersect (Select R Where C2)

دو دستور زیر معادلند:

Select R Where C1 OR C2

(Select R Where C1) Union (Select R Where C2)

دو دستور زیر معادلند :

Select R Where not C

Select R Minus (Select R Where C)

روابط زیر نیز در جبر رابطه‌ای برقرارند :

$B) \cup \Pi_{x1,\dots,xp}(B) = \Pi_{x1,\dots,xp}(A \cup \Pi_{x1,\dots,xp}(A)$

$B) \infty \Pi_{x1,\dots,xp}(B) = \Pi_{x1,\dots,xp}(A \infty \Pi_{x1,\dots,xp}(A)$

$B) \infty_p (A \sigma B = \infty_p (A) \sigma$

$_p(A)) \sigma_p(\Pi_q(A)) = \Pi_q (\sigma$

در فرمول آخری، p یک شرط و q نام ستونهای موردنظر در خروجی می‌باشد. البته عبارت سمت چپی فرمول فوق به شرطی قابل محاسبه است که در شرط p فقط نام ستونهای داخل لیست q ذکر شده باشد.

بهینه‌سازي پرس و جو

در جبر رابطه‌ای برای یک سئوال ممکن است چند پاسخ وجود داشته باشد ولی همه آنها از نظر فضای مصرفی و زمان معادل نیستند.

مثال ۴۷ : خروجی دو دستور زیر را بدست آورید.

الف : $SP) \infty_{,p,} (S \neq_p \sigma$

ب : $_{,p1,}(SP)) \neq_p \sigma (\infty S$

حل :

S1	فن‌آوران	تهرا ن	P1	300

خروجی →

$SP \infty S$

S#	Sname	City	P#	Qty
S1	فن‌آوران	تهرا ن	P1	300
S1	فن‌آوران	تهرا ن	P2	200
S1	فن‌آوران	تهرا ن	P3	400

خروجی دستور الف = خروجی دستور ب

→

S

S1	فن‌آوران	تهران
S2	ایران قطعه	تبریز
S2	پولادین	تبریز

∞

S1	P1	300
S2	P1	300

پس خروجی هر دو دستور یکسان است ولی دستور ب از نظر فضای مصرفی و زمان بهینه‌تر از دستور الف است.

در دستور الف ابتدا دو جدول S , SP پیوند خورده سپس سطرهایی از آن پیوند، انتخاب می‌شود. ولی در دستور ب ابتدا سطرهایی از جدول SP انتخاب شده، سپس آن سطرها با جدول S پیوند می‌خوردو بدیهی است پیوند دو جدول کوچکتر زمان و فضای کمتری می‌خواهد.

در بعضی از DBMSها بخشی به عنوان بهینه‌ساز وجود دار که دستورات ورودی کاربران را بهینه‌سازی می‌کند. البته همه نرم‌افزارهای DBMS پرس و جوها را بهینه نمی‌کنند و بنابراین بهتر است حتی‌الامکان پرس و جوی بهینه را وارد کنیم.

جهت بهینه‌سازی پرس و جو می‌توان از قواعد زیر استفاده کرد :

۱ـ گزینش را هر چه ممکن است زودتر انجام دهید.

۲ـ شـــــــرط‌های ترکیبی را به شرط‌های متوالی تبدیل کنید. مثلاً می‌توان $\sigma_{p1 \wedge p2}(e)$ با $\sigma_{p1}(\sigma_{p2}(e))$ جایگزین کرد.

۳ـ عملگر پرتو را زود انجام دهید (ولی دیرتر از گزینش)

۴ـ همانطور که می‌دانید از نظر ریاضی عملگر پیوند طبیعی خاصیت جابجایی و شرکت‌پذیری دارد. ولی از نظر کامپیوتری زمان و مصرف حافظه عملیات $A \infty B$ ممکن است خیلی بیشتر از $B \infty A$ باشد و یا اینکه ممکن است $(A \infty B) \infty C$ بهینه‌تر از $A \infty (B \infty C)$ باشد.

مثال ۴۸ : دستور $S \infty SP$ سریعتر اجراء می‌شود یا $SP \infty S$ ؟

الگوریتم $S \infty SP$:

```
} برای هر سطر جدول S
{ برای هر سطر جدول SP
مقایسه کن
انتخاب کن }
}
```

الگوریتم $SP \infty S$:

```
} برای هر سطر جدول SP
{ برای هر سطر جدول S
مقایسه کن
انتخاب کن }
}
```

فرض کنید جدول S کوچک است و جدول SP بسیار بزرگتر از آن. جدول S به اندازه‌ای کوچک است که در حافظه سریع Cache جای می‌گیرد. در الگوریتم $SP \infty S$ باید سطرهای زیاد جدول SP به دفعات وارد حافظه اصلی شده و مقایسه و انتخاب انجام گیرد.

به عبارت دیگر تعداد دستیابی به دیسک به اندازه حاصلضرب سایز دو جدول است. در الگوریتم $S \infty SP$، هر سطر SP فقط یکبار به حافظه اصلی می‌آید، زیرا جدول S، به طور کال در حافظه Cache است و همه مقایسه‌ها به یکباره انجام می‌شود. در این حال تعداد دستیابی به دیسک به اندازه سایز جدول SP می‌باشد.

به روز در آوردن داده‌ها

تغییر جداول مانند اضافه کردن و حذف کردن یک یا چند ستون و تغییر دامنه ستون‌ها به جبر رابطه‌ای مربوط نمی‌شود، زیرا این قبیل تغییرات در واقع تغییر تصویر ادراکی بانک اطلاعات است. جبر رابطه‌ای با رابطه‌ها کار دارد و نه با تصویر ادراکی آنها. منظور از به روز درآوردن داده‌ها سه مورد : اضافه کردن داده به جدول، حذف داده از جدول و تغییر داده‌های جدول است.

الف : اضافه کردن داده به جدول

افزودن یک یا چند سطر به جدول با استفاده از عملگرهای $\cup$, $\leftarrow$ قابل انجام است.

مثال ۴۹ : قطعه P5 با رنگ قرمز و از جنس آهن که در تهران تولید می‌شود را به جدول P اضافه کنید.

$$\cup, P \leftarrow P \ \{(\text{‘P5’ ؤ ‘قرمز’ ، ‘آهن’ ، ‘تهران’})\}$$

ب : حذف داده از جدول

برای این کار در جبر رابطه‌ای نیاز به عملگر جدیدی نیست و با دستورات - و $\leftarrow$ انجام‌پذیر است.

مثال ۵۰ : کلیه قطعاتی را که تعداد آنها کمتر از ۳۰۰ عدد می‌باشد را از جدول SP حذف کنید.

$$SP \leftarrow SP - \sigma_{Qty > 300}(SP)$$

ج : تغییر داده‌هاي جدول

تغییر به کمک عملگرهای σ و $\leftarrow$ قابل انجام است.

مثال ۵۱ : نام شهر ‘تبریز’ را در جدول p به ‘لاهیجان’ تغییر دهید.

$$\leftarrow city\ \sigma_{\text{‘لاهیجان’}}(city = \sigma_{\text{‘تبریز’}}(P))$$

مثال ۵۲ : به تعداد قطعات در جدول SP، ۵۰ عدد اضافه کنید.

$$Qty \leftarrow \sigma_{Qty+50}(SP)$$

کاربرد رابطه DUM , DEE در جبر رابطه‌ای

در حساب معمـــولی عدد ویژه ”1“ وجود دارد که دارای خاصیت n*1=1*n=n می‌باشد یعنی 1 در عمل ضرب عضو خنثی است. همچنین عدد ”0“ در حساب معمولی دارای خاصیت n*0=0*n=0 می‌باشد. می‌خواهیم ببینیم معادل این دو رقم 0 , 1 در جبر رابطه‌ای چیست. یادآوری می‌کنیم که DUM , DEE هر دو رابطه‌ای از درجه صفر هستند که مجموعه صفات خاصه آنها تهی است. DEE فقط حاوی یک تاپل صفر است وDUM هیچ تاپلی ندارد.

DEE مانند عدد ”1“ در ضرب حساب معمولی عمل می‌کند. یعنی برای تمام رابطه‌های R داریم :

R TIMES DEE = DEE TIMS R = R

به عبارت دیگر، DEE در عملگر TIMES , JOIN به عنوان رابطه همانی عمل می‌کند. ولی هیچ رابطه‌ای وجود ندارد که در عمل TIMES مثل صفر در ضرب حساب معمولی عمل کند. اما رفتار DUM تا حدودی شبیه صفر است به طوری که برای تمام رابطه‌های R داریم :

R TIMES DUM = DUM TIMES R = an empty relation with the same headimg as R

حال اکثر عملگرهای جبر رابطه‌ای را بر روی DUM , DEE بررسی می‌کنیم.

۱ـ ابتدا توجه کنید که فقط رابطه‌هایی که با DUM , DEE همنوع هستند، خود رابطه‌های DUM , DEE می‌باشند پس ∪، ∩ و – فقط برای ترکیب این دو تعریف شده است.

عملوند اول	عملوند دوم	∪	∩	-
DUM	DUM	DUM	DUM	DUM
DUM	DEE	DEE	DUM	DUM
DEE	DUM	DEE	DUM	DEE
DEE	DEE	DEE	DEE	DUM

ستون ∪ شبیه OR و ستون ∩ شبیه AND می‌باشد.

۲ـ هر محدودیت از DEE در صورتی منجر به DEE می‌شود که شرط محدودیت درست باشد در صورتی منجر به DUM می‌شود که شرط محدودیت نادرست باشد. هر محدودیت از DUM منجر به DUM می‌شود.

۳ـ تصویر هر رابطه‌ای روی هیچ صفتی، وقتی منجر به DUM می‌شود که رابطه اصلی تهی باشد و گرنه برابر DEE می‌گردد. به طور خاص تصویر DEE یا DUM، بر روی هیچ صفتی، برابر با ورودی آن است.

۴ـ در مورد عملگر تقسیم (تقسیم دو عملوندی) قواعد زیر را داریم :

رابطه خالی $R \div DUM = DUM \div R = R$

$R \div DEE = DEE \div R = R$

منظور از رابطه خالی R، رابطه‌ای است تهی که عنوان آن همانند رابطه R است.

اگر R غیر تهی باشد DEE

$R \div R =$

اگر R تهی باشد	DUM

با توجه به نکات تقسیم متوجه می‌شویم که در جبر رابطه‌ای مانند حساب معمولی مشکل تقسیم عدد بر صفر را نداریم و عمل تقسیم در جبر رابطه‌ای بسته است.

کلید کاندید در روابط حاصله

فرض کنید A , B دو رابطه دلخواه باشند آنگاه :

۱ـ (σ) هر محدودیت از A تمام کلیدهای کاندید A را به ارث می‌برد.

۲ـ (II) اگر تصویر دلخواه بر روی A شامل هر کلید کاندید K باشد، در اینصورت K یک کلید کاندید برای تصویر خواهد بود. در غیر اینصورت، تنها لید کاندید ترکیب تمام صفات تصویر می‌باشد.

۳ـ (×) هر ترکیب مانند K از یک کلید کاندید K_A از رابطه A و یک کلید کانیدید K_B از رابطه B یک کلید کاندید برای A × B می‌باشد.

۴ـ ($\cup$) تنها کلید کاندید برای $A \cup B$ ترکیب تمامی صفات است.

۵ـ ($\cap$) هر کلید کاندید A یا B یک کلید کاندید برای $A \cap B$ می‌باشد.

۶ـ (-) هر کلید کاندید از A یک کلید کاندید برای A – B می‌باشد.

۷ـ (∞) در حالت خاص هرگاه که صفت الحاقی موجود در A یک کلید کاندید A باشد، هر کلید کاندید B یک کلید کاندید برای الحاق خواهد بود.

۸ـ (EXTEND) کلیدهای کاندید مربوط به یک توسعه دلخواه از A همان کلیدهای کاندید A می‌باشند.

۹ـ (SUMMARIZE) تنها کلید کاندید برای یک خلاصه‌سازی دلخواه از A مجموعه صفات مشخص شده در عبارت BY می‌باشد.

کاربرد کاتالوگ در پاسخگویی به سئوالات مربوط به جدول

کاتالوگ نوعاً شامل دو متغیر رابطه‌ای سیستمی به نامهای TABLE , COLUMN است که اهداف آن توصیف جدولهای بانک اطلاعاتی و ستونهای آن جدولهاست.

مثال ۵۳ : اگر بانک شامل دو جدول زیر باشد :

EMP

EMP#	ENAME	DEPT#	SALARY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K
E4	Saito	D2	35K

DEPT

DEPT#	DNAME	BUDGET
D1	Marketing	10M
D2	Development	12M
D3	Research	5M

متغیرهای TABLE و COLUMN به صورت زیر خواهند بود :

TABNAME	COLNAME	…
DEPT	DEP#	…
DEPT	DNAME	
DEPT	BUDGET	
EMP	EMP#	
EMP	ENAME	
EMP	DEPT#	
EMP	SALARY	
⋮	⋮	

TABNAME	COLCOUNT	ROWCOUNT	…
DEPT	3	3	…
EMP	4	4	
	⋮	⋮	

کاتالوگ باید خود توصیف (Self Descriptor) باشد یعنی باید حاوی اطلاعاتی باشد که متغیرهای رابطه‌ای کاتالوگ را توصیف نماید.

حال می‌خواهیم بدانیم که محتویات ستونهای متغیر رابطه‌ای DEPT چیست. عبارت زیر (در زبان Tutorial D) این کار را می کند.

(COLUMN Where TABNAME = DEPT) {COLNAME}

یا مثلاً می‌خواهیم بدانیم کدام متغیر رابطه‌ای حاوی ستونی به نام #EMP است؟

(CLOUMN Where COLNANE = EMP#) {TABNAME}

فصل هفتم

حساب رابطه اي

معرفي حساب رابطه‌اي

حساب رابطه ای با جبر رابطه ای منطقاً معادل است یعنی برای هر عبارت جبر رابطه‌ای، یک عبارت معادل در حساب رابطه‌ای وجود دارد و برعکس. تفاوت آنها این است که جبر رابطه‌ای، دستوری است ولی حساب رابطه‌ای توصیفی می‌باشد. به عبارتی دیگر، جبر رابطه‌ای رویه‌ای و حساب رابطه‌ای، نارویه ای اسن و به زبان نزدیکتر است. جبر رابطه‌ای مجموعه ای از عملکردها مثل الحاق، اجتماع، تصویر و غیره را تدارک می‌بیند که به سیستم می‌گویند«چگونه رابطه مطلوبی را از رابطه‌های دیگر بسازد». در حالیکه حساب رابطه‌ای فقط نشانه گذاریی را تدارک می‌بیند که برای «تعریف آن رابطه مطلوب بر حسب رابطه های دیگر» به کار می‌روند. در مورد فرموله کردن حسابی ، کاربر فقط ویژگیهای تعریف نتایجه مطلوب را بیان کرده و تصمیم گیری در مورد اعمال محدودیت ، پرتو، الحاق و غیره را جهت بدست آوردن نتیجه به عهده سیستم می‌گذارد. کاد نشان داد که قدرت جبر رابطه ای با حساب رابطه ای یکسان است.

آقای کاد دو سال پس از انتشار مقاله اصلی خود و ارائه جبر رابطه‌ای در مقاله دیگری در سال ۱۹۷۲ تئوری خود را به صورت دیگری ارائه کرده و آن را حساب رابطه‌ای تاپلی (tupe relational calculus) نامید. پنج سال پس از آن دانشمند دیگری روش سومی به نام حساب رابطه‌ای دامنه ای یا میدانی (domain relational calculus) را ارائه کرد. این دو روش اخیر از قدرت محاسباتی مساوی برخوردارند. پس می‌توان گفت حساب رابطه‌ای خود دو شاخه دارد: ۱-حساب تاپلی ۲-حساب میدانی .

حساب رابطه‌ای مبتنی بر شاخه‌ای از منطق ریاضی به نام محمولات یا مسندات (Predicate calculus) می‌باشد. کاد، در آغاز زبانی به نام ALPHA بر اساس همین حساب پیشنهاد کرد. هر چند این زبان هرگز پیاده سازی نشد ولی زبانهای رابطه ای بعدی مانند SQL QUEL, از این زبان تأثیر پذیرفته اند.

حساب تاپلي (چندتايي)

در حساب تاپلی یک مفهوم مهم به نام متغیر تاپلی(متغیر محدوده‌ای-متغیر طیفی= Range Variable وجود دارد . متغر تاپلی، متغیری است که تنها مقادیر مجازش تاپلهای رابطه هستند.

مثال ۱: نمونه های زیر طرز تعریف متغیرهای تاپلی را نشان می‌دهند.

```
RANGEVAR   SX    RANGES  OVER  S;
RANGEVAR   SY    RANGES  OVER  S;
RANGEVAR   SPX   RANGES  OVER  SP;
RANGEVAR   SPY   RANGES  OVER  SP;
RANGEVAR   PX    RANGES  OVER   P;
```

SX, SY, SPX, SPY, PX نام اختیاری برای متغیرهای تاپلی و کلمات RANGEVAR , OVER,RANGES کلمات کلیدی می باشند. پس از OVER نام جدول آورده می‌شود. در این حساب دو سور(کمیت سنج-چندی نما=Quantifier) وجود دارد:

سور وجودی با نماد ∃ یا EXISTS و سور همگانی یا جهانی با نماد ∀ یا FORALL.

سور وجودی به صورت $\exists T(f)$ نوشته می‌شود و به این معناست که «حداقل یک مقدار متغیر T وجود دارد به نحوی که f به TRUE ارزیابی شود». F را یک فرمول خوش ساخت Well formed (formula =WFF) بخوانید و ف) می گوئیم.

سور همگانی به صورت $\forall T(f)$ نوشته می‌شود و به این معناست که «به ازای تمام مقادیر متغیر T، عبارت f به TRUE ارزیابی می‌شود.»

نکته: اگر $\neg$ نماد NOT باشد فرمولهای زیر برای سورها برقرار است:

$$\forall T(f) \equiv \neg(\exists T(\neg f))$$

$$\exists T(f) \equiv \neg(\forall T(\neg f))$$

$$\forall T(f\ AND\ g) \equiv \neg\exists T(\neg f\ OR \neg g)$$

$$\forall T(f\ OR\ g) \equiv \neg\exists T(\neg f\ AND \neg g)$$

$$\exists T(f\ OR\ g) \equiv \neg\forall T(\neg f\ AND \neg g)$$

$$\exists T(f\ AND\ g) \equiv \neg\forall T(\neg f\ OR \neg g)$$

مثال ۲: اگر x از مجموعه اعداد صحیح مقدار بگیرد آنگاه:

$$\exists X(X>100) \Rightarrow TRUE \qquad (EXISTS\ X(X>100))$$

$$\forall X(X>100) \Rightarrow FALSE \qquad (FORALL\ X(X>100))$$

شکل کلي عبارت حساب تاپلي

اگر T ، یک متغیر تاپلی روی رابطه$R(A_1,A_2,....,A_n)$ باشد، در اینصورت شکل کلی عبارت حساب تاپلی به صورت زیر است:

(target –item(s)) [WHERE f]

که در آن target –item(s) فهرستی از صفحات متغیر تاپلی T است:

$T.A_1 ,T.A_2 ,...,T.A_n$

مثال ۳: نمونه ای از ((target –item(s))به صورت زیر است:

الف) SX .SNAME

ب) (SX .S# ,SX .STATUS)

ج) (SX .S #AS SA, SY.S # AS SB)

د) SX

مثالهاي حساب رابطه اي تاپلي

مثال ۴: شماره و وضعیت عرضه کنندگان در پاریس را مشخص کنید که status >20 باشد.

(SX .S# ,SX .status)
Where SX.city = ‘paris’ AND Sx.status>20

مثال ۵: جفتهایی از شماره عرضه کنندگان را مشخص کنید که هر دو عرضه کننده در یک شهر باشند.

(SX .S#AS SA,SY .S# AS SB)
Where SX.city=SY.city AND SX.S #<SY.S#

توجه کنید که بخشهای AS به صفات نتیجهٔ حاصل ، نامهایی را اختصاص می‌دهد. بنابراین آن اسامی برای استفاده در بخش Where مهیا نیستند. به همین علت مقایسه دوم در بخش where برابر با ”SX .S #<SY.S#“ نوشته شده است و نه به صورت ”SA<SB“

مثال ۶: اطلاعات کامل عرضه کنندگانی را مشخص کنید که قطعه P2 را تولید می‌کنند.

SX
Where Exists SPX (SPX.S#=SX.S# AND SPX.P#= ‘p2’

دستور فوق معادل دستور زیر است :

```
(SX.S#,SX.SNAME .SX.STATUS.SX.CITY)
Where Exists SPX (SPX.S#=SX.S#AND SPX.P#= ‘P2’
```

مثال ۷: اسامی عرضه کنندگانی را مشخص کنید که حداقل یک قطعه قرمز را عرضه می‌کنند.

```
SX.Sname
Where Exisys SPX(SX.S#= SPX.S# AND
                    Exists PX(PX.P#=SPX.P#AND
                                PX.Color= ‘Red’))
```

فرمول فوق را می‌توان به شکل نرمال زیر که بـه prenex معـروف اسـت نیـز نوشـت. در فـرم prenex تمام سورها جلوی وف ظاهر می‌شوند:

```
SX.Sname
Where Exists SPX (Exists PX (SX.S#=SPX.S#  AND
                                SPX.P#=PX.P#  AND
                                PX.Color= ‘Red’))
```

مثال ۸: اسامی عرضه کنندگانی را مشخص کنید که تمام قطعات را عرضه می‌کنند.

```
SX.SNAME where FORALL PX(Exists SPX (SPX.S#=SX.S#
                                AND SPX.P#=PX.P#))
```

تذکر : تقاضای فوق بدون استفاده از FORALL به صورت زیر است:

```
SXSNAME Where NOT Exist PX(NOT Exists SPX
                                (SPX.S#=SX.S#AND
                                SPX.P#=PX.P#))
```

مثال ۹: اسامی عرضه کنندگانی را مشخص کنید که قطعه P2 را عرضه نمی‌کنند.

```
SX.Sname  where not Exists SPX
```

(SPX.S #= SX.S# AND SPX.P#= ‘p2’

تذکر: در حساب رابطه ای قابلیتهای محاسباتی نیز داریم. مثالهـای زیـر ایـن موضـوع را نـشان می‌دهند.

مثال۱۰: شماره و وزن قطعاتی را مشخص کنید که وزن آنها بیش از ۱۰۰۰۰ گرم باشد.

(PX.P#,PX.Weight*454 AS GMWT)

where PX.weight *454>10000

عبارت AS GMWT نامی را به صفت نتیجه حاصل اختصاص می‌دهد، بنـابراین ایـن نـام در بخش where نمی‌تواند مورد استفاده قرار گیرد.

مثال ۱۱: میزان کل ارسال را مشخص کنید.

SUM(SPX,Qty)AS GRANDTOTAL

مثال ۱۲: برای هر قطعه ، شماره و میزان کل عرضه را مشخص کنید.

PX.P#,SUM(SPX where SPX.P#=PX.P#,Qty)AS TOTQty

مثال ۱۳: برای هر عرضه کننده، شماره عرضه کننده و تعداد کل قطعات عرضه شده را مشخص کنید.

(SX.S#,Count(SPX where SPX.S#=SX.S#=SX.S#)AS TOT)

تذکر : هر عبارت حسابی می‌تواند به عبارت جبری معادل آن تبدیل شود و هر عبـارت جبـری نیز می‌تواند به عبارت حسابی معادل آن تبدیل گردد. پس جبـر رابطـه‌ای و حـساب رابطـه ای معادل یکدیگرند.

حساب رابطه اي ميداني (يا دامنه‌اي)

در این نوع حساب به جای متغیر تاپلی، متغیر میدانی یا محدوده(Domain Variable) داریم. متغیر میدانی متغیری است که از یک میدان مقدار می‌گیرد. تفاوت اصلی حساب میدانی با حساب تاپلی در این است که در حساب میدانی یک شرط اضافی به نام «شرط عضویت» وجود دارد.

شرط عضویت چنین نوشته می شود:

$R(A_1: V_1,A_2:V_2,...)$

که در آن R نام رابطه، A_i نام صفت و v_i یک مقدار (یا متغیر) از میدان است. این شرط به «درست» ارزیابی می‌شود اگر و تنها اگر تاپلی در R وجود داشته باشد که مقادیر داده شده برای صفات را داشته باشد.

مثال ۱۴: عبارت SP(S#: ‘S1’ ,P#: ‘p1’ به «درست» ارزیابی می‌شود، اگر و فقط اگر تاپلی در SP وجود داشته باشد که S# آن S1 و P# آن P1 باشد.

مثال ۱۵: شرط عضویت: SP(S#:SX ,P#:PX) ارزش درستی دارد، اگر و فقط اگر یک چند تایی ارسال وجود داشته باشد که مقدار S# آن برابر با مقدار فعلی متغیر محدوده SX و مقدار P# آن برابر با مقدار فعلی متغیر محدوده PX باشد.

در مثالهای زیر متغیرهای میدانی زیر را در نظر می‌گیریم:

متغیرهای میدانی	دامنه
SX,SY	S#
PX ,PY	P#
NameX,NameY	Name
CdorX ,colory	Color

Qty	QtyX,cityY
City	CityX,cityY
Status	StatuX ,status Y

مثال ۱۶: عبارت روبرو مجموعه شماره‌های تمام عرضه کنندگان را نشان می‌دهد. SX

مثال ۱۷:

SX where S(S#:SX)

مجموعه‌ای از شماره های تمام عرضه کنندگان را در متغیر رابطه‌ای (جدول) S نشان می دهد.

مثال ۱۸:

SX where S(S#: SX,city: ‘London’)

شماره عرضه کنندگانی را نشان می دهد که شهر آنها لندن است.

مثال ۱۹:

(SX ,city X) Where S(S#:SX ,city :cityX)

AND SP (S#:SX ,P#: ‘p2’)

دستور فوق شماره عرضه کنندگان و شهرهای آنها را مشخص می‌کند که قطعه p2 را عرضه می کنند.

معادل این دستور در جلسات تاپلی به صورت زیر بود:

(SX ,S# ,SX.city)

where Exists SPX(SPX .S#=SX.S# AND SPX.P#= ‘p2’)

مثال ۲۰: شماره عرضه کنندگانی را مشخص کنید که در پاریس هستند و status آنها بیشتر از 20 است.

SX where Exists status X

(statusx >20 AND

S(S#:SX ,status :status X,city : ‘paris’

معادل این دستور در حساب تاپلی به صورت زیر است:

SX.S#

Where SX.city = ‘paris’ AND SX.status>20

مثال ۲۱: اسامی عرضه کنندگان را بدهید که قطعه p2 را عرضه نمی‌کنند.

NAMEX where Exists SX (S(S#:SX,SNAME :NAMEX)

AND NOT SP (S#:SX,P#: ‘p2’))

مثال ۲۲: اسامی عرضه کنندگانی را مشخص کنید که حداقل یک قطعه قرمز را تولید می‌کنند.

NAMEX where Exists SX Exists PX

(S(S#:SX,Sname : NAMEX)

AND SP (S#:SX,P#:PX)

AND P(P#:PX ,Color : ‘Red’))

در حساب رابطه‌ای تاپلی دستور فوق به صورت زیر نوشته می شود:

SX .Sname

Where Exists SPX (Exists PX (SX.S#=SPX.S# AND)

SPX.P#=PX.P# AND

PX.Color= ‘Red’))

مثال ۲۳: تمام جفتهایی از شماره عرضه کنندگان را مشخص کنید که دو عرضه کننده در یک شهر باشند.

(SX AS SA ,Sy AS SB) where Exists city Z

(S(S#:SX ,city:cityZ) AND

S(S# :Sy ,city :cityZ) AND

Sx<Sy)

معادل دستور فوق در حساب رابطه‌ای تاپلی به صورت زیر است:

(SX.S#AS SA ,SY.S# AS SB)

where SX,city=Sy.city AND SX.S#<Sy.S#

تذکر: وضعیت امروزه SQL این است که بعضی از جنبه های SQL شبیه جبر و بعضی دیگر از جنبه‌های آن شبیه حساب است و بعضی دیگر از جنبه های آن هیچ کدام از این دو نیست.

حساب رابطه اي دامنه اي با نمادي ديگر

شکل کلی عبارت در حساب رابطه ای دامنه‌ای به صورت زیر است:

$\{<C_1,C_2,...C_n> \mid P(C_1,C_2,...,C_n,C_{n+1},...)\}$

و معنی آن این است که : $C_1,C_2,...C_n$ را بده به طوری که شرط $P(C_1,C_2,...,C_n,C_{n+1},...)$ برقرار باشد. قواعد زیر باید رعایت شود:

- برای ارتباط متغیرها به جداول از تعلق($\in$) استفاده می‌شود و شرط همتایی باید رعایت شود.
- ترکیب شرط‌ها با $\Rightarrow,\neg,\vee,\wedge$ (معادل و ،یا،نه، نتیجه می‌دهد) انجام می‌شود.
- عدم تعلق با منفی کردن شرط (استفاده از $\neg,\in$) انجام می شود و علامت عدم تعلق $\notin$ تعریف نشده است(زیرا با طبیعت بانک اطلاعاتی همخوانی ندارد)
- خروجی دستور شامل همهٔ ستونهای $<C_1,C_2,...C_n>$ خواهد بود.

- در شرط p می‌توان از متغیرهای دیگر نیز استفاده کرد ولی باید این متغیرها قبلا با استفاده از سورهای $\forall,\exists$ تعریف شده باشند. استفاده از ثابتها در صورتی که متعلق به دامنه متغیرهای مربوطه باشند ایرادی ندارد.
- برای پیوند جداول از متغیرهای همنام در آنها استفاده می‌شود.
- گاهی می‌توان با تغییر جملات پرس و جو، پاسخگوئی به آن را آسانتر کرد.
- از فرمولهای تئوری مجموعه‌ها برای پیدا کردن معادل‌های ساده تر استفاده می‌شود

مثل دو فرمول زیر :

$$A - B \equiv A \cap \neg B \qquad A \to B \equiv \neg A \cap B$$

در این معادل سازی ، عملکردهایی مانند تفریق که مربوط به جبر رابطه ای هستند و در حساب رابطه ای جائی ندارند حذف می‌شوند و به جای آنها عملگرهایی می آیند که به سادگی قابل معادل سازی هستند.

مثال ۲۴: این دستور شماره عرضه کنندگانی را نشان می دهد که شهر آنها لندن است.

$$\{< SX > | \exists NameY, stausX, cityX$$

$$(< SX, NameX, statusX, cityX > \in S \wedge city = 'London'\}$$

مثال ۲۵: دستور زیر مشخصات کامل عرضه کنندگانی را نشان می دهد که شهر آنها لندن است.

$$\{< SX, NameX, statusX, cityX > | < SX, NameX, statusX, cityX > \in S \wedge city = 'London'\}$$

جدول زیر را در نظر بگیرید(prof جدول اساتید است)

prof:

Pname	Office	Esp	Degree	Clg#

10	فوق لیسانس	کامپیوتر	4	میر شمسی
6	دکتری	مواد	3	ابوطالبی

مثال ۲۶: اساتید متخصص بانک اطلاعاتی که مدرک دکتری دارند:

$$\{< p,o,e,d,c >|< p,o,e,d,c >\in prof \wedge e="\text{اطلاعات}" \quad \text{بانك} \quad "\wedge d="\text{دكتري}" \}$$

جدول دانشجو (Stud) زیر را در نظر بگیرید:

stud (S#,sname ,city ,avg,clg#)

(شماره دانشکده، معدل، شهر، نام دانشجو ،شماره دانشجو) جدول دانشجو

مثال ۲۷: دستور زیر دانشجویانی که معدل آنها از ۱۸ بیشتر است را می‌دهد:

$$\{< s,sn,c,a,c1 >|< s,sn,c,a,c1 >\in stud \wedge a > 18\}$$

مثال ۲۸: دستور زیر نام دانشجویانی که معدل آنها از ۱۸ بیشتر است را می‌دهد:

$$\{< sn >|\exists s,c,a,c1(< s,sn,c,a,c1 >\in stud \wedge a > 18\}$$

خطر حلقه بي پايان در عبارتهاي حساب رابطه ي دامنه اي

عبارتهای حساب رابطه ای دامنه‌ای که در آنها از $\neg$ استفاده شود می‌توانند خطرناک باشند یعنی تولید حلقه بی انتها کنند و خروجی آنها بی پایان باشد.

مثال ۲۹ : دستور روبرو :

$$\{< c,cn,u,c1 >|\neg(< c,cn,u,c1 >\in crs)\}$$

که جدول درس (crs) به صورت زیر است:

crs(c# ,cname ,unit ,clg#)

(شماره دانشکده، تعداد واحد درس، نام درس، شماره درس) جدول درس

به این معناست: «تمام جداول چهارستونی ممکن به غیر از جدول crs را بده که ستونهای آنها با جدول crs هم دامنه باشند.» خروجی این دستور بی‌پایان است.

در عبارتی که در آنها $\forall,\exists$ استفاده می‌شود دو خطر وجود دارد یکی خروجی بی پایان و دیگری تعداد بی‌پایان مقایسه .

مثال ۳۰: دستور $\{< c,u,c1 > | \exists cn(\neg(< c,cn,u,c1 > \in crs))\}$ علاوه بر اینکه خطر خروجی بی‌پااین را مشابه مثال قبلی دارد، باید برای تعداد بی‌پایان رشته که می‌توانند جای نام درس بنشینند، مقایسه را انجام می‌دهد، هنگام نوشتن عبارتهای حساب رابطه ای دامنه‌ای باید به این نکته مهم توجه داشت.

تذکر: برای پیاده سازی حساب رابطه ای دامنه ای کوشش هایی صورت گرفته که قدیمی‌ترین آنها QBE(Query By Example) و موفق‌ترین آنها زبان پرس و جوی Datalog (Database Logic) است.

فصل هشتم

SQL

معرفي SQL

زبان SQL (Structured Query Language) پیاده سازی ازادی از جبر رابطه ای است که البته بعضی از عملکردهای آن مثل تقسیم را نمی پوشاند ولی در عوض عملکردهای کاربردی زیاد دیگری تعریف می‌کند که کار کردن با جداول را آسان می‌کنند.

این زبان اولن بار در سال ۱۹۷۶ پدید آمده و ده سال بعد توسط ANSI استاندارد شد. SQL یک زبان بیانی (declarative) است. بدین معنا که کاربر تنها می‌گوید«چه می خواهد» ولی چگونگی بدست آوردن آن را مشخص نمی‌کند. در واقع تبدیل دستورات SQL به عملکردهای جبر رابطه‌ای توسط خود سیستم SQL انجام می‌پذیرد.

SQL به دو صورت مستقل و ادغام شدنی به کار می‌رود. در دنیای PC ها عموماً SQL را تحت یک نرم افزار دیگر مثل دلفی، ویژوال بیسیک C یا Access به کار می‌بریم. در این حال برای تعریف انواع متغیرها و همچنین برای دستورات کنترلی و توابع ریاضی رشته ای از زبان میزبان استفاده می شود.

یکی از نسخه های معروف SQL برای کامپیوترهای بزرگ SQL/DS که محصول شرکت IBM است می باشد. SQL/DS یک سیستم کامپایلری است البته در حال حاضر بسیاری از سیستم های بانک اطلاعاتی مفسری می‌باشند. SQL/DS زبانی مستقل می‌باشد.

معمولترین نسخه این نرم افزار در حال حاضر SQL2 می باشد. البته در حال حاضر زبانی به نام OSQL پدیدآمده و ANSL نسخه جدیدی به نام SQL3 را اعلام کرده که شئی گرایی را پشتیبانی می‌کند. ما در این فصل SQL2 را شرح می دهیم.

تذکر: زبانهای رابطه ای موجود عبارتند از : DATALOG-QBE-QUEL-SQL

- SQL در ابتدا یک زبان داده‌ها (DSL) بود. با قراردادن ویژگی«روال ذخیره شدهٔ پایدار (persistent Stored Modules=PSM) در استاندارد سال ۱۹۹۶، SQL از نظر محاسبات نیز کامل شد و اکنون حاوی دستوراتی مثل Repeat,while loop,if,Case,Set,Return CALL, و ویژگی هایی مثل متغیرها و پردازش استثناها می‌باشد. در نتیجه در حال حاضر لازم نیست SQL را با زبان دیگری به نام میزبان ترکیب کنیم تا برنامه کاربردی کاملی ایجاد شود.
- SQL به جای دو اصطلاح دو اصطلاح رابطه و متغیر رابطه‌ای از جدول استفاده می‌کند. SQL از اصطلاحات عنوان و بدنه استفاده نمی‌کند.
- SQL با زبان رابطه ای فاصله دارد. با ان حال استاندارد است و اغلب محصولات موجود در بازار آن را پشتیبانی می کنند.
- کاراکتر # در اسامی ستونها در SQL مجاز نیست هر چند که ما در مثالهایمان از آن استفاده می کنیم.

انواع داده‌اي در SQL2

تعدادی از انواه متغیرها در SQL2 عبارتند از :

INTEGER عدد صحیح

SMALLINT عدد صحیح

DECIMAL(p,g) عدد هرمی دارای p رقم و q رقم اعشاری در سمت راست

FLOAT برای اعداد اعشاری با نقطه شناور

NUMERIC(p,q) عدد حقیقی

CHARACTER(n) یا CHAR(n) رشته ای کاراکتری به طول n ($1 \le n \le 254$)

VARCHAR(n) رشته ای کاراکتری به طول متغیر حداکثر $n(1 \le n \le 32767)$

DATE تاریخ با نمایش هشت رقم دهدهی بدون علامت (yyyymmdd)

TIME زمان با نمایش شش رقم دهدهی بدون علامت (hh mm ss)

TIMESTAMP ترکیبی از تاریخ و زمان با دقت میکروثانیه با نمایش ۲۰ رقم دهدهی بدون علامت (yyyymmddhhmmssnnnnn)

BIT(n) اعداد مبنای ۲

تعریف دامنه هاي جدید

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

مثال ۱:

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

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

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

اپراتورها در SQL

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

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

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

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

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

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

A	NOT A
T	F
F	T
؟	؟

A	B	A OR B
T	T	T
T	F	T
F	T	T
F	F	F
T	?	T
?	T	T
F	?	?
?	F	?
?	?	?

A	B	A AND B
T	T	T
T	F	F
F	T	F
F	F	F
T	?	?
?	T	?
F	?	F
?	F	F
?	?	?

مفهوم NULL می‌تواند منشأ اشتباهات و ابهاماتی در سیستم گردد.

تذکر: اگر SQL میزبان زبان دیگری باشد آنگاه از عملگرها، متغیرها و توابع داخلی آن زبان استفاده می‌گردد.

تعریف بانك در SQL

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

الف) دستور Create table

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

مثال ۲:

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

```
status smallint ,
city char (15) NOT NULL,
primary key (S#))
```

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

مثال ۳: دستور زیر جدول sp را تعریف می کند:

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

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

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

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

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

ب) Alter table

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

مثال ۴:

```
ALTER TABLES
ADD  DISCOUNT  SMALLINT
```

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

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

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

با عبارت < نام ستون > ALTER یـا < نـام سـتون > Modify مـی تـوان جلـوی Alter table تعریف یک ستون را تغییر داد.

مثال ۵:

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

این مثال در واقع نوع داده را تغییر نمی‌دهد بلکع طول S# را از ۵ به ۱۰ افزایش می‌دهد. اگـر بخواهیم نوع داده را به طور کلی عـوض کنـیم، اکثـر نـسخه هـای SQL از اینکـار جلـوگیری می‌کنند مگر آنکه ستونهای مربوطه فاقد داده باشند.

مثال ۶:

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

تذکر : حذف یک ستون در بعضی از SQL ها پیاده سازی نشده است زیرا حذف ستون ممکن است تـأثیر نـامطلوبی روی ارتبـاط بـا یکـدیگر بگـذارد. ولـی در بعـضی نـسخه‌هـا بـا عبـارت < نام ستون > Drop بعد از Alter table می توان ستونی را حذف کرد.

ج) DROP TABLE

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

د) Create Index

شاخص(Index) جدولی است که بر اساس فیلدی از یک جدول پایه، به صورت مرتب شده ساخته می‌شود.

مفهوم جدول ایندکس را در درس ذخیره و بازیابی خوانده‌اید.

مثال۷: با دستور زیر :

Create Index SN on S(sname ,city)

شاخصی به نام اختیاری SN روی جدول S ایجاد می‌شود و نظم اصلی روی مقادیر صعودی Sname و سپس روی مقادیر صعودی city می‌باشد. اگر بخواهیم بر اساس نزولی باشد بعد از نام فیلد عبارت DESC را می آوریم.

مثال ۸:

Create Index SN on S(Sname DESC)

مثال ۹: اگر بعد از Create عبارت UNIQUE را بیاوریم:

Create UNIQUE INDEX SN ON S(Sname)

در اینصورت SQL از درج مقدار تکراری برای Sname (مثلاً توسط دستور Insert در جدول S جلوگیری می‌کند. در واقع یکتایی مقدار Sname را تضمین می‌کنیم. ضمناً نمی‌توان روی ستونی که مقادیر جاری‌اش یکتایی ندارد، درخواست ایجاد شاخص یکتا را کرد . سیستم SQL به صورت خودکار روی کلید اصلی شاخص یکتا ایجاد می‌کند.

مثال ۱۰:

Create Index SC On S(city)

چون فیلد شهر یکتا نیست، نمی‌توانیم درخواست ایجاد شاخص یکتا را بکنیم. در مورد استفاده یا عدم استفاده از شاخص در پاسخگوئی به یک سؤال کاربر سیستم SQL تصمیم می‌گیرد و نه کاربر.

تذکر: تعریف شاخص اگر تعداد داده ها زیاد باشد، سرعت عملیات را خیلی بالا می‌برد. ولی دو ایراد کوچک دارد. اول آنکه مقداری از حافظه جهت ذخیره شاخص مصرف می‌شود(که البته در حال حاضر که حافظه ها بزرگ می‌باشند مهم نیست). ایراد دوم مقدار زمانی است که برای بررسی فایل اطلاعاتی و شاخص‌گذاری آن صرف می شود. هر چند که این زمان ممکن است زیاد باشد ولی فایل یکبار شاخص‌گذاری شده و برای مدت طولانی استفاده می‌شود. به علاوه شاخص‌گذاری را می‌توان در مواقع بی‌کاری سیستم(مثل شبها) انجام داد. با افزودن رکوردهای جدید به فایل، شاخص‌های آن به صورت خودکار توسط سیستم به هنگام می‌شوند.

تذکر : از آنجا که foxpro برای ساخت جدول پایه و ایندکس دستورات خاص خود را دارد دستورات Create Index , Create table مربوط به SQL را پشتیبانی نمی‌کند.

هـ) Drop Index

با این دستور شاخص ایجاد شده حذف می‌گردد مثل: Drop index Sn

تذکر : در SQL می‌توان کار طراحی و ایجاد بانک را به طور تدریجی انجام داد. یعنی ابتدا تعداد محدودی جدول ایجاد و بلافاصله داده های بانک را وارد کرد. سپس به تدریج بانک را گسترش داد.

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

دستور SELECT

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

ساده ترین شکل این دستور به صورت زیر است:

Select نام فیلدها

from نام جدول

where شرط جستجو

مثال ۱۱:

```
select S# ,status
froms   S
where   city= 'C2'
```

خروجی ⟵

S#	status
S1	20
S3	30
S4	20

مثال ۱۲:می‌توان نام جدول را همراه نام فیلدها به کار برد.

دستورات مقابل دقیقاً مثال قبل عمل می‌کنند.

```
select   S.S# ,status
```

```
froms   S
where   city= ‘C2’
```

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

فرمت کلی دستور select به صورت زیر است:

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

مثال ۱۳: دستور روبرو :

```
select  P# from  SP
```

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

(البته زیر هم نوشته می‌شوند)

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

مثال ۱۴: دستور روبرو:

```
Select Distinct  P#   from    SP
```

به علت استفاده از Distinct مقادیر ستون P# از جدول SP را با حذف تکراری‌ها می دهد یعنی خروجی به صورت زیر می‌شود:

(البته زیر هم نوشته می‌شوند)

P1 ,P2 ,P3 ,P4 ,P5 ,P6

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

مثال ۱۵: شماره تمام قطعات و وزن هر یک را بر حسب گرم بدهید. فرض کنید وزنها بر حسب پوند در جدول P باشند.

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

خروجی⇒

(هر پوند ۴۵۴ گرم می‌باشد)

P#			
P1	weight	in	gram=5447
P2	weight	in	gram=7718
P3	weight	in	gram=7718
P4	weight	in	gram=6356
P5	weight	in	gram=5448
P6	weight	in	gram=8626

مثال ۱۶: دستور زیر کل جدول S را چاپ می‌کند:

```
select      *
from        S
```

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

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

مثال ۱۷: شماره تهیه کنندگانی را بیابید که ساکن C2 بوده و وضعیت آنها از 20 بیشتر باشد:

خروجی :

```
Select        S#
```

S#

From S S3

Where city = ‘C2’ AND status >20

مثال ۱۸: شماره وضعیت تهیه‌کنندگان ساکن C2 را بر اساس نظم نزولی مقادیر وضعیت بدهید.

خروجی :

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

S#	Status
S3	30
S1	20
S4	20

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

select S# from S order by city⇒ غلط

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

مثال ۱۹:

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

عدد ۲ یعنی ستون دوم جدول جواب

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

P6 8626

توجه کنید که خروجی ابتدا بر اساس ستون دوم یعنی وزن مرتب می‌شود و اگر دو سطر وزن یکسانی داشته باشند آنگاه بر اساس P# مرتب می شود. به سطر ۱ و ۲ و همچنین ۴و۵ توجه کنید.

عملگر between و in در select

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

مثال ۲۰: خروجی این دستور چیست؟

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

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

می توان از not between نیز استفاده کرد.

مثال ۲۱: خروجی این دستور چیست؟

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

P#	Weight
P1	12
P2	17
P3	17
P5	12

می توان از not in نیز استفاده کرد.

عملگر like در select

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

مثال ۲۲: مشخصات قطعاتی را بیابید که اسم آنها با حرف C شروع شده باشد.

Select *
From P
Where pname like 'C%'

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

مثال ۲۳: دستور زیر نام قطعاتی را می‌دهد که ۳ حرفی بوده و با حرف C شروع می‌شوند:

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

مثال ۲۴: دستور زیر نام قطعاتی را می‌دهد که حاوی کاراکتر 'E' نمی‌باشد:

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

مثال۲۵:"_A%B" like اسامی را می‌دهد که حرف دوم آنها A بوده و مختوم به B می‌باشند.

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

مثال ۲۶:

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

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

پرس و جوهاي مبتني بر پيوند جدولها

پیوند نوعی پرس وجو است که طی آن عمل بازیابی از بیش از یک جدول انجام می‌پذیرد.

مثال ۲۷: خروجی دستور زیر چیست؟

```
select  S.S# , S.city ,P.P# ,P.city
From   S,P  where   S.city=P.city
```

خروجی در زیر ترسیم شده است.

برای اجتناب از ابهام ، دو ستون city ، به صورت S.city و P.city باید نوشته شوند. ولی S.S# را می‌توانستیم به صورت S# هم بنویسیم.

S.S#	S.city	P.P#	P.city
S1	C2	P1	C2
S1	C2	P4	C2
S1	C2	P6	C2
S2	C3	P2	C3
S2	C3	P5	C3
S3	C2	P1	C2
S3	C2	P4	C2
S3	C2	P6	C2
S4	C2	P1	C2
S4	C2	P4	C2
S4	C2	P6	C2

مثال ۲۸:

```
select  S* ,P.*
```

```
from  S,P
where  S.city=P.city
```

دستور فوق تمام فیلدهای دو جدول را در صورت برقراری شرط می‌نویسد. خروجی آن را ترسیم نکرده ایم.

عمل پیوند به شرط تساوی طبق تعریف باید جدولی را بدهد حاوی دو ستون یکسان. اگر یکی از دو ستوان را از جدول جواب حذف کنیم، جدول حاصله را پیوند طبیعی می‌نامیم.

تذکر: در بعضی از نسخه های SQL می‌توان تا ۱۶ جدول را با همدیگر پیوند داد.

نکته: دستور select S* ,P.* عمل ضرب کارتزین دو رابطه P,S را در SQL انجام می‌دهد.

```
from  S,P
```

مثال ۲۹: نام جفت‌شهرهایی را بیابید که تهیه کننده‌گان ساکن شهر اول قطعه ای انبار شده در شهر دوم را تهیه کرده باشد. مثلاً S1 قطعه P2 را تهیه می‌کند. S1 ساکن شهر C2 و P2 ساکن شهر C3 است. بنابراین جفت شهر (C2 ,C3) یک سطر از جواب است:

```
sekect Distinct  S.city ,P.city
From S,SP,P
Where S.S#=Sp.S#  AND  SP.P#=P.P#
```

S.city	P.city
C2	C2
C2	C3
C2	C4
C3	C2
C3	C3

مثال فوق ، مثالی از پیوند سه جدول می‌باشد.

مثال ۳۰: (پیوند یک جدول با خودش)تمام جفت شماره تهیه کنندگانی را بیابید که از یک شهر باشند.

```
Select  First .S# ,Second .S#
From  S  First , S Second
Where  First.city =Second.city
```

First.S#	Second.S#
S1	S1
S1	S3
S1	S4
S2	S2
S3	S1
S3	S3
S3	S4
S4	S1
S4	S3
S4	S4
S5	S5

فرض کنید در یک لحظه دو نسخ مجزا از S وجود دارد یا نامهای اختیاری First , Second . البته در سطح فیزیکی دو نسخه از جدول S ایجاد نمی‌شود. در بعضی از نسخه‌های SQL خط From به صورت زیر (با AS) نوشته می‌شود:

```
From  S  AS  First ,S  AS  Second
```

بنابراین AS هم برای تغییر نام ستونهای جدول جواب و هم برای تغییر نام خود جدولها استفاده می‌شود. البته حوزه عملکرد این تغییر نام فقط در همان دستور است.

مثال ۳۱: برای حذف زوائد در مثال قبلی باید دستور زیر را بنویسیم:

```
select First.S#,Second .s#
From  S  First ,S Secound
Where First .City=second.city
AND  first.S# <second.S#
```

First .S#	Second.S#
S1	S3
S1	S4
S3	S4

دستورات Select متداخل

مثال ۳۲: نام تهیه‌کنندگانی را بیابید که قطعه P1 را تهیه می‌کنند.

این پرس و جو را می‌توانیم به کمک عملیات پیوند تنظیم کنیم:

```
select  S.sname
Frome  S,SP
Where   S.S#=SP.S#   AND  SP.P#= ‘P2’
```

Sname
Sn1
Sn2
Sn3
Sn4

ولی یک راه دیگر استفاده از select متداخل به فرم زیر است:

```
select  sname from s
where  S# in(select S# from SP
where  P#= ‘P2’)
```

سیستم ابتدا پرس و جوی درونی را انجام می‌دهد. پاسخ این پرس و جو مجموعه {S1,S2,S3,S4} می‌شود بنابراین دستور فوق معادل دستور تک سطحی زیر است:

```
select sname from  S
where  S#  in (‘s1’ , ‘s2’ , ‘s3’ , ‘s4’)
```

تذکر: از نظر کارائی روش پیوند بهتر از پرس و جوی متداخل است ولی از نظر ظاهری و درک برنامه روش متداخل خواناتر است. در SQL ممکن است برای حل یک مسأله چندین راه حل مختلف وجود داشته باشد و به نظر دیت، این ایرادی است که بر زبان SQL وارد است.

مثال ۳۳: شماره تهیه کنندگانی را بیابید، که در همان شهری ساکن باشند که تهیه‌کنندهٔ S1 ساکن است.

```
Select   S#  from S
Where city =(select city from s
                Shere s#= ‘s1’)
```

S#
S1
S3
S4

به پرسشی که تعدادی پرسش فرعی در درون خود داشته باشد، پرش تودرتو(Nested Query) یا چند سطحی هم می گویند.

توابع ستونی در select

این توابع عبارتند از :

Count: تعداد مقادیر در یک ستون Sum : مجموع مقادیر یک ستون

AVG: میانگین مقادیر یک ستون MAX: بزرگترین مقدار در یک ستون

MIN: کوچکترین مقدار در یک ستون

مثال۳۴: کل تعداد تهیه شده از قطعه P2 را بدهید.

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

جواب
1000

مثال ۳۵: شماره تهیه‌کنندگانی را بدهید که مقدار وضعیت آنها کوچکتر از مقدار ماکزیمم وضعیت باشد.

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

S#
S1
S2
S4

مثال۳۶:

```
select Min (Qty) AS  MIN_QTY
```

```
from  S
```

با AS می‌توان نام جدید به ستون خروجی داد.

تذکر: در صورت لزوم فوق را می‌توان با کلمه Distinct نیز به کار برد. در اینحالت داده های تکراری در نظر گرفته نمی‌شوند(برای MAX, MIN داده های تکراری اهمیتی ندارند). مقادیر NULL قبل از اجرای این توابع حذف شده و روی این توابع تأثیری ندارند. مثلا در محاسبه مقدار تابع AVG مقادیر NULL در محاسبه تعداد داده ها(مخرج کسر) اثری ندارد.

مثال ۳۷: تعداد تهیه کنندگان شهر C2 چند تاست؟

```
select  count(S#) from  S
where  City= 'C2'
```

خروجی : ۳

مثال ۳۸: تعداد شهرهای موجود درجدول P چند تاست؟

در این مثال باید از شمارش تکراری جلوگیری کرد.(با AS نام ستون خروجی را تغییر داده‌ایم).

```
Select  Count (Distinct city) AS  CT#
From P
```

خروجی :

CT#
3

نکته: تابع ویژه Count(*) برای شمارش سطرهای جدول است. در این تابع نمی‌توان از Distinct استفاده کرد و این تابع سطرهای NULL را نیز می‌شمارد. اگر جدول تهی باشد ، این تابع صفر بر می‌گرداند.(برخلاف سایر توابع که NULL برمی‌گردانند)

مثال۳۹: مشخص سازیر چند تهیه کننده P2 را تهیه کرده‌اند؟

```
select  count(*) from  SP
```

where P#= ‘P2’

جواب : ۴

Having و Group By در select

مثال ۴۰: کل مقدار تهیه شده از هر قطعه را در جدول جواب بدهید(همراه با شماره هر قطعه)

جواب :

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

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

مثال ۴۱ : برای هر قطعه تهیه شده، شماره قطعه، کل تعداد و ماکزیمم تعداد تهیه شـده از آن را بدون در نظر گرفتن S1 بدهید.

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

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

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

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

مثال ۴۲: شماره قطعه تمام قطعاتی که توسط بیش از یک تهیه کننده تهیه شده‌اند را بیابید:

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

تذکر : Group by , having افزونه هستند یعنی هر پرسشی که با این دو تنظیم شود، با امکانات دیگر موجود درSQL نیز قابل تنظیم است.

عملگرهاي مجموعه‌اي در select

عملگرهای مجموعه‌ای تعریف شده در جبر رابطه ای در SQL پیاده سازی شده‌اند. عمل اجتماع با دستور UNION ، عمل اشتراک با INTERSECT و عمل تفاضل با EXCEPT پیاده سازی شده است.

در اینجا هم باید همتایی داده‌ها رعایت شود. یعنی باید تعداد ستونها و همچنین دامنه های ستونها در جدول ، با هم برابر باشند.

در SQL عملگر تعلق به صورتIN که قبلاً بیان شد، پیاده سازی شده است. به علاوه عملگر دیگری به نام CONTAINS وجود دارد که مشابه عملگر زیر مجموعه(⊂) عمل می‌کند. مجموعه سمت راست زیر مجموعه سمت چپ می‌باشد یعنی:

$$A \subset B \Rightarrow B \quad CONTAINS \quad A$$

دستور Contains غالباً نیاز به گروه بندی داده ها(Group by) دارد که جلوتر کاربرد آن را شرح می‌دهیم.

مثال ۴۳: شماره قطعاتی را بیابید که یا وزن آنها بیش از ۱۶ باشد یا توسط S2 تهیه شده باشند با هر دو شرط را دارا باشند.

```
Select P# from P
Where weight>16
UNION
Select P# From SP
Where  S#=S2
```

P#
P2
P3
P6
P1

P2, P3, P6 مربوط به قسمت اول

P1 مربوط به قسمت دوم

عناصر تکراری فقط یکبار در خروجی نوشته می‌شوند مگر اینکه به جای UNION عبارت UNTON ALL را بنویسیم که در اینصورت در مثال فوق P2 دوباره در خروجی ظاهر می‌شود.

Exists در Select

فرم کلی آن به صورت exists (select*From...) است. چنین عبارتی به مقدار «درست» ارزیابی می‌شود، اگر و فقط اگر مجموعه حاصل از ارزیابی پرسوجوی داخلی select *from ... تهی نباشد. در واقع هر پرس و جوئی که با استفاده تز IN قابل تنظیم باشد با استفاده از exists نیز قابل تنظیم است ولی عکس این معنا، درست نیست. Exists وجود سطر و not exists عدم وجود سطر را بررسی می‌کند.

مثال ۴۴: اسامی تهیه کنندگان قطعه P2 را بیابید.

```
Select sname from  S
Where exists (select * from  SP)
                Where   S#=S.S#  AND  P#= 'P2'
```

هر یک از Sname ها را به نوبت در نظر گرفته و بررسی می‌کنیم آیا سبب می‌شود که حاصل ارزیابی exists مقدار «درست» گردد.

دستور Insert

این دستور دو فرم کلی زیر را دارد:

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

در فرم اول یک سطر با مقادیر مشخص برای فیلدها، در یک جدول درج می‌شود. ثابت i ام در لیست ثابتها متناظر با فیلد iام در لیست فیلدها می‌باشد.

در فرم دوم، پرس و جوئی فرعی ارزیابی می شود و جواب آن که معمولاً چندین سطر است در جدول درج می شود. در هر دو شکل ننوشتن لیست فیلدها به معنای این است که تمام فیلدهای جدول مورد نظر است.

مثال ۴۵: قطعه P7(شهر C1، وزن 24 ، اسم و رنگ در حال حاضر ناشناخته) را در جدول P درج کنید.

```
Insert  Into  P(P# ,City,weight)
Values (‘P7’ .’ ‘C1’ ,24)
```

مقدار فیلدهای اسم و رنگ برابر NULL می‌شود.

تذکر : نظم از چپ به راست فیلدها در جلوی دستور Insert لزما مشابه Create نیست.

مثال ۴۶: قطعه P8 را با مشخصات (P8 ,Pn8, ‘Pink’ , 14 , ‘C8’) را در جدول P درج کنید.

```
Insert Into  P
Values (P8,Pn8, ‘Pink’ , 14, ‘C8’)
```

چون لیست فیلدها نوشته نشده است، منظور تمام فیلدهاست ، با همان نظم دستور Create.

تذکر: بهتر است همیشه نام فیلدها ذکر گردد.

مثال۴۷: برای هر قطعه تهیه شده، شماره قطعه و کل تعداد تهیه شـده از آن را بدسـت آورده و نتیجه را در بانک ذخیره کنید.

```
Create Table Temp
(P#  CHAR(6)  NOT  NULL,
TOTQTY  Int,
PRIMARY   KEY(P#))
Insert Into Temp (P# ,TOTQTY)
Select  P# ,SUM(Qty)
From  Sp
Group  By P#
```

Temp

P#	TOTQTY
P1	600
P2	1000
P3	400
P4	500
P5	500
P6	100

نتیجه insert در جدول temp ذخیره می شود. کاربر می‌تواند با این جدول کار کند و هر گاه که خواست آن را با دستور drop table temp حذف کند.

دستور Update

شکل کلی این دستور به صورت زیر است:

Update< نام جدول>

Set <فیلد 1>=<مقدار 1>[,<فیلد۲>=<مقدار2>]...

[where <شرط>]

مثال ۴۸: رنگ قطعه P2 را به زرد تغییر داده به وزن آن ۵ بیفزایید و شهر محل انبار کردن آن را ناشناخته اعلام کنید.

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

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

مثال ۴۹: تعداد را در محموله‌های تهیه‌کنندگان ساکن C3 صفر کنید .

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

تذکر :ممکن است لازم باشد بهنگام سازی در چند جدول همزمان صورت گیرد.

مثال ۵۰: شماره S1 را در جدول S به S11 تبدیل کنید:

```
Update  S   Set   S#= 'S11'  where  S#= 'S1'
Update  SP  Set   S#= 'S11'  where  S#= 'S1'
```

از آنجا که S# در جدول SP کلید خارجی است پس تغییرات S# در جدول S باید به ستون S# در جدول SP نیز اعمال گردد.

دستور Delete

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

Delete From < نام جدول> [where <شرط>]

مثال ۵۱: تهیه کننده S5 را حذف کنید .

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

مثال ۵۲: تمام محموله‌هایی که تعداد آنها از ۳۰۰ بیشتر است را حذف کنید.

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

مثال ۵۳: تمام محموله‌ها را حذف کنید(یعنی تمام سطرهای جدول SP را)

```
Delete From SP
```

جدول SP هنوز وجود دارد ولی خالی است.

مثال ۵۴: تمام محموله های تهیه‌کنندگان ساکن C3 را حذف کنید.

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

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

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

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

دستور فوق غلط است.

دیدها (View) یا دیدگاه

دید جدولی مجازی است، یعنی جدولی که در واقع وجود ندارد ولی به نظر کاربر چنین می آید که وجود دارد برای ساخت دید از دستور Create View با فرم کلی زیر استفاده می‌شود:

CREATE VIEW < نام دید>[(فیلد ۱ [,فیلد 2]...)]

AS<جستجو>

مثال ۵۶:

```
Create View Redpart (P# ,WT ,City)
As Select P# ,weigh ,City
From P where Color = 'Red'
```

Redpart

P#	WT	City
P1	12	C2
P4	14	C2
P6	19	C2

دستور فوق یک دید به نام Redpart پدید می‌آورد.

مثال ۵۷:

```
Create View PQ(P#,TOTQTY)
AS Select P# ,SUM(QTY)
    From SP
    Group By P#
```

دیدPQ

P#	TOTQTY
P1	600
P2	1000
P3	400
P4	500
P5	500
P6	100

برای از بین بردن یک دید از دستور<نام دید > Drop View استفاده می‌کنیم مثل:

```
Drop View Redpart
```

تذکر: اگر جدول مبنائی حذف شود، تمام دیدهای تعریف شده روی آن جدول مبنا نیز حذف می گردد . از دید کاربر View مشابه یک جدول عمل می‌کند که تمامی دستورات گفته شده را می‌توان برای آن به کار برد.

مثال ۵۸:

```
Create View GS
```

دیدGS

S#	Status	City
S1	20	C2
S3	30	C2
S4	20	C2
S5	30	C1

As Select S# ,Status ,City

From S where Status >15

پس از ساختن دید فوق اگر دستور زیر را صادر کنیم خروج ذیل ظاهر می گردد:

Select *

From GS where City <> ‘C2’ ⇒

S#	Status	City
S5	30	C1

در واقع SQL دستور فوق را به دستور زیر تبدیل می کند:

Select S# ,Status ,City

From S

Where City <> ‘C2’ AND Status >15

در مثال فوق می توان گفت S متغیر رابطه ای پایه و GS متغیر رابطه ای مشتق شده است.

تذکر: به جای هر ارجاعی به نام دیدگاه، عبارت تعریف کننده دیدگاه (که در کاتالوگ ذخیره شده) قرار می‌گیرد.

تذکر: می‌توان گفت که در SQL سه نوع جدول وجود دارد :

۱-جداول اصلی (base tables) که با دستورCreate table ساخته می‌شوند که می‌توان به آنها دسترسی داشت و هر گونه تغییری در آنها اعمال کرد.

۲-جداول میانی (intermediate tables) که خود سیستم آنها را پدید آورده و استفاده می کند و از دسترسی کاربران خارج هستند. مثلاً در دستوراتی که دارای زیر دستور هستند(مثل select متداخل) ابتدا بخش زیر دستور محاسبه و به طور موقت ذخیره می‌گردد و سپس به

کمک آن کل پرس و جو انجام می‌گیرد. جداول میانی قابل دسترسی نبوده و بدین دلیل قابل تغییر نمی‌باشند.

۳-جداول مجازی(Views) که وجود خارجی نداشته و می‌توان به آنها دسترسی داشت و در موارد معدودی آنها را تغییر داد.

تذکر: هدف اصلی از جدول مجازی ایجاد جداول خلاصه از اطلاعات موجود و محدود کردن دید کاربران است. مثلاً ممکن است رئیس سازمان فقط شماره قطعه و شهر قطعات را بخواهد و با وزن و رنگ آنها کاری نداشته باشد. دسترسی به View از دید کاربر مستقیم ولی از دید سیستم غیر مستقیم است یعنی سیستم هر گونه استخراج اطلاعات را از جداول اصلی انجام می‌دهد.

نکته: مشکل اصلی در جداول مجازی به روزدرآوردن آنها می‌باشد. در صورتی می توان جدول مجازی را به هنگام کرد که تغییرات مورد نظر روی جداول اصلی بدون اشکال و ابهام باشد.

مثال زیر این موضوع را نشان می‌دهد.

مثال۵۹: رکورد جدید P8 , 200 را به جدول مجازی PQ اضافه کنید.

حل: این درخواست امکان‌پذیر نیست. چرا که جدول مجازی PQ بخشی از Sp می‌باشد و در صورت درج در زیر S# باید عبارت NULL قرار بگیرد. حال آنکه S# بخشی از کلید اصلی SP است و نمی‌تواند NULL باشد.

بررسی تمام شرایطی که بهنگام سازی جدول مجازی را مجاز می‌دارد مشکل است. بعضی از نسخه‌های SQL نیز به طور کلی این عمل را انجام نمی‌دهند. در بعضی دیگر نیز اختیار انجام بهنگام سازی به عهده خود سیستم گذاشته شده تا در صورت امکان آن را انجام دهد.

یکی از قواعد ساده برای بهنگام سازی جداول مجازی به صورت زیر است:

۱-جدول مجازی فقط از یک جدول اصلی ساخته شده باشد.

۲-در ساختن جدول مجازی از Select Distinct استفاده نشده باشد.

۳-اگر جدول مجازی روی جدول اصلی A ساخته شده آنگاه A در زیر دستور نیامده باشد. به چینن جداولی ، جداول قابل تغییر (Updateable) می‌گویند.

عملگرهای ANY ,ALL

عملگرALL برای مقایسه «همه مقادیر» و عملگر ANY(که در بعضی از نسخه های SQL نام SOME دارد) برای «هر یک از مقادیر» استفاده می‌شوند. نتیجه هر دو عملگر TRUE یا FALSE است . این دو عملگر را می‌توان با توابع و عملگرهای دیگر معادل سازی کرد.

مثال ۶۰: نام قطعاتی را بدهید که وزن آنها عددی فرد مابین ۱۰ تا ۲۰ باشد.

```
Select Pname  from P
Where weight =AND(11,13,15,17,19)
```

خروجی : Pname

Bolt

Screw

Cog

تذکر : AND= معادل in می باشد.

مثال ۶۱: نام تهیه کنندگانی را بدهید که وضعیت انها از همه ساکنان شهر C2 بیشتر باشد.

```
Select Sname from  S
Where status >ALL (Select status from  S
                   Where  city = 'C2'
```

این دستور برای جداول داده شده در آخر این فصل خروجی ندارد.

پرس و جو با قید «همه»

معادل عملگر÷ جبر رابطه ای در SQL وجود ندارد. به چند روش می توان تقسیم را در SQL معادل سازی کرد. ساده‌ترین این روشها استفاده از تابع Count است. در واقع حالتها را می‌شماریم تا ببینیم «همه» آنها رخ داده اند یا خیر .

مثال ۶۲: شماره تهیه کنندگانی که همه قطعات را تولید کرده اند بدهید.(جواب S1 است)

اطلاعات مربوط به همه قطعات در جدول P وجود دارد. پس برای هر تهیه کننده (در جدول SP) تعداد قطعات تولیدی آن را می‌شماریم و با تعداد قطعات موجود در جدول P مقایسه می‌کنیم.

```
Select S#  from  SP
Group  by  S#
Having  Count (P#) =(Select  Count (P#) from P);
```

پیوند در SQL

در SQL1 عملیات پیوند مستقیما پشتیبانی نمی‌شد و با استفاده از شرط در جلوی where یا توسط select متداخل شبیه سازی می گردید. ولی در SQL2 دستوراتی اضافه شد تا انواع پیوند را مستقیماً پشتیبانی کنند.

۱- ضرب دکارتی (×) جبر رابطه ای با دستور CROSS JOIN پیاده سازی شده است.

مثال ۶۳:

```
S  CORSS  JOIN  Sp
```

دستور فوق ضرب دکارتی دو جدول S , SP را می‌دهد.

۲-پیوند شرط (X_θ) جبر رابطه ای با دستور ...ON ... JOIN پیاده سازی شده است.

مثال ۶۴:

```
S  JION  SP  ON
S.S# =SP.S#  AND  Qty >200,
```

۳-پیوند طبیعی (∞) جبر رابطه ای با دستور NATURAL JOIN پیاده سازی شده است.

مثال ۶۵:

```
S  Natural  Join  Sp
```

تذکر : از آنجا که خروجی این دستورات «رابطه» است می توان آنها را در بخش FROM از دستور select استفاده کرد.

مثال ۶۶:

```
select sname ,Qty
From  S  Natural Join  Sp
```

۴-پیوند OUTER . می دانیم که اگر سطری از جدول A با هیچ سطری از جدول B فیلد همنام یکسان نداشته باشد، این سطر A در $A\infty B$ ظاهر نمی‌شود. این سطر (که به سطر سرگردان یا dangling tuple معروف است) بعضی اوقات ایجاد اشکال می کند.

دستور Natural Full Outer Join سطرهای هر دو جدول را که با جدول دیگر مقدار همنام مشترک ندارند نیز در خروجی می آورد.

امنیت در SQL

در SQL هر کاربری یک شناسنامه ویژه دارد. شناسنامه PUBLIC همه کاربران را شامل می‌شود. چهار نوع امتیاز برای دستیابی به جداول عبارتند از : Update ,Delete,Insert Select, . مثلاً اگر کاربری بخواهد در جدولی اطلاعاتی درج کند باید شناسه او امتیاز Insert را روی آن جدول داشته باشد.

امتیاز دیگر References است که برای کنترل و اعمال محدودیتهای جامعیتی می‌باشد. مثلا رئیس یک سازمان باید بتواند روی دستمزد کارمندان خود قواعد محدودیت را اعمال کند(پس باید امتیاز References را داشته باشد) ولی دیگران چینن حقی را ندارند.

همچنین امتیاز USAGE برای استفاده از قابلیتهایی مثل VIEW یا اجازه استفاده از یک میدان می باشد. به کمک دستور GRANT می توان به کاربری امتیاز داد و با دستور REVOKE می توان این امتیازها را پس گرفت . فرم کلی دستور GRANT به صورت زیر است:

GRANT <لیست امتیازها> ON <بخشی از بانک اطلاعاتی>

To <لیستی از کاربران>[with Grant Option]

نوشتن عبارت With Great Option باعث می‌شود کاربر مورد نظر بتواند این امتیازها را با دستور Great دیگری به سایر کاربران واگذار کند.

مثال ۶۷:

Great Update ,insert ON S,SP To ALL ,JAVAD

فرم کلی دستور Revoke به صورت زیر است.

Revoke<لیست امتیازها> ON<بخشی از بانک> From<لیستی از کاربران>

Revoke insert ON S from ALL

تذکر: امتیاز Insert(X), Update(X) ,References(X) فقط برای ستون X می‌باشد.

تذکر: با دستور Revoke Grant Option ON می‌توان فقط حق واگذاری امتیاز به غیر را باز پس گرفت ولی خود امتیاز همچنان به قوت خود باقی باشد.

تذکر: می‌توان دستورات SQL را به ۳ دسته زیر تقسیم کرد:

۱-دستورات DDL(مثل alter domain,Create domain,drop table,Alter table Create table ,drop index ,Create index,drop domain)

۲- دستورات DML(مثل delete ,Update ,insert ,Select)

۳- دستورات DCL(مثل revoke ,grant)

مکان‌نما در SQL

اصل مهمی که در مورد SQL تعبیه شده حاکم است و اصل «دوحالتی» (dualmode principle) نام دارد، این است که، هر دستور SQL که می‌تواند به طور محاوره ای مورد

استفاده قرار گیرد در یک برنامه‌کاربردی نیز قال استفاده است. توجه داشته باشید که بسیاری از دستورات SQL تعبیه شده نمی‌توانند به طور محاوره ای مورد استفاده قرار گیرند.

دستورات SQL معمولاً چندین سطر(و نه فقط یک سطر) را بازیابی می کنند و زبانهای میزبان معمولاً این امکان را ندارند که بازیابی چند سطر همزمان را پشتیبانی کنند. لذا لازم است پلی بین قابلیتهای دستیابی سطح مجموعه در SQL و قابلیتهای بازیابی سطح سطر در زبان میزبان برقرار شود. برای این منظور از مکان‌نما(Cursor) استفاده می‌شود. مکان‌نما نوع خاصی از شئی SQL است که فقط در SQL تعبیه شده و قابل استفاده است زیرا SQL محاوره ای به آن نیاز ندارد. مکان نما متشکل از یک نوع اشاره گر منطقی است که می تواند در مجموعه‌ای از سطرها حرکت کند و هر سطر را قابل آدرس‌دهی کند.

نمونه ای از تعریف مکان نما در SQL تعبیه شده به صورت زیر می باشد:

```
Dellare  X  cursor for
Select  S.S# ,S.Sname ,S.Status
From  S
Where  S.city = 'London'
```

دستور Delcare X Cursor یک مکان نما به نام X را با عبارت جدولی مربوط تعریف می کند. این دستور اعلانی است و نه اجرائی. با دستور Close X می‌توان این اشاره‌گر را از بین برد.

SQL پویا

SQL پویا این امکان را می دهد که دستورات SQL مورد نیاز را به طور پویا در هنگام اجرای برنامه ایجاد کرد و سپس آن دستورات ساخته شده را به طور پویا کامپایل و اجراء کرد.

دو دستور پویای اصلی عبارتند از EXECUTE ,PREPARE که کاربرد آن را در مثال زیر (SQL تعبیه شده در PL/1) نشان می دهیم:

```
DCL SQLSOURCE  CHAR VARYING (65000);
SQLSOURCE= ‘DELECT FROM SP WHERE QTY <300;
EXEC SQL PREPARE  SQLPREPPED  FROM : SQLSOURCE;
EXEC SQL EXECUTE SQLPREPPED
```

-SQLSORCE متغیر رشته ای با طول متغیر در PL/1 است که دستورات SQL را به صورت رشته درآن می‌ریزیم.

-SQLPREPPED یک متغیر SQL است که شکل ترجمه شده دستوری از SQL را در خود نگه می‌دارد.

-اسامی SQLPREPPED, SQLSOURCE اختیاری هستند.

-خط دوم، دستور SQL را به صورت رشته در متغیر SQLSOURCE ذخیره می‌سازد. البته در عمل این فرآیند پیچیده‌تر می‌باشد یعنی این رشته باید مثلا توسط کاربر وارد شود.

-دستورPREPARE دستور منبع را گرفته و آن را برای تولید نسخه اجرایی آن آماده کرده و در متغیر SQLPREPPED ذخیره می سازد.

-دستورEXECUTE نسخه SQLPREPPED را اجراء می کند و موجب می‌شود DELETE اتفاق بیفتد.

-چون SQLPREPPED یک متغیر SQL است نه متغیر PL/1 هنگام استفاده از آن پیشوند کولن(:) استفاده نمی‌شود.

SQL/SLI

امکان جدیدی به نام «ربط سطح فراخوانی SQL» (CLI =Call-Level Interface) به استاندارد SQL اضافه شده است. CLI تا حد زیادی به ODBC شرکت میکروسافت مربوط می‌شود (ODBC=Open Data Base Connectivity).CLI به برنامه کاربردی که در زبانهای میزبان معمولی نوشته شده‌اند اجازه می‌دهد با فراخوانی روالهایی که در بازار موجود است، تقاضایی را به بانک اطلاعاتی بفرستند. سپس این روالها، که باید به برنامه کاربردی مورد نظر متصل شوند، با استفاده از SQL پویا اعمال بانک اطلاعاتی درخواست شده را بر روی برنامه کاربردی اجراء می‌کنند. از دیدگاه سیستم مدیریت بانک اطلاعاتی روالهای CLI را می‌توان به عنوان برنامه های کاربردی دیگری در نظر گرفت.

همانطور که مشاهده می کنید SQL/CLI (و همچنین ODBC) با همان مساله ای روبرو هستند که SQL پویا با آن مواجه است: یعنی هر دو تکنیک اجازه می‌دهند در برنامه‌های کاربردی که برای آنها نوشته می شوند، دستورات SQL واقعی که باید اجراء شوند تا زمان اجزاء ناشناخته باشند.

به دو دلیل روش SQL/CLI بهتر از SQL پویا می‌باشد:

۱-SQL پویا یک استاندارد Source Code است. لذا هر برنامه کاربردی که از SQL پویا استفاده می‌کند به خدماتی از کامپایلر SQL نیاز دارد تا اعمالی مثل EXECUTE PREPARE, را با آن استاندارد انجام دهد. در حالیکه CLI جزئیات فراخوانیهای روالها را به

ندرت استاندارد می‌کند، در نتیجه برنامه های کاربردی می‌توانند به شکل Object code بسته بندی شده و توزیع گردند(احتمالا توسط فروشندگان دیگر).

۲- این برنامه های کاربردی می‌توانند مستقل از سیستم مدیریت بانک اطلاعاتی باشند، یعنی با CLI می‌توان برنامه‌های کاربردیی کلی را تولید کرد که توسط سیستم های مدیر بانک اطلاعاتی مختلفی مورد استفاده قرار گیرند(به جای اینکه ویژه یک سیستم مدیریت بانک اطلاعاتی باشند).

تذکر: SQL کامل نیست. SQL با زبان رابطه ای کامل فاصله زیادی دارد. در نتیجه روشن نیست که محصولات SQL امروزی استحقاق این را دارند که بر چسب «رابطه ای» را داشته باشند یا خیر. امروزه محصولی در بازار نیست که تمام جزئیات مدل رابطه ای را پشتیبانی کند.

جداول SP,P,S

S#	P#	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

جدول SP

S#	Sname	Status	City
S1	Sn1	20	C2
S2	Sn2	10	C3
S3	SN3	30	C2
S4	Sn4	20	C2
S5	Sn5	30	C1

جدول S

P#	Pname	Color	Weight	City
P1	Nut	Red	12	C2
P2	Bolt	Green	17	C3
P3	Screw	Blue	17	C4
P4	Screw	Red	14	C2
P5	Cam	Blue	12	C3
P6	Cog	Red	19	C2

جدول P

فصل نهم

وابستگی تابعی

همانطور که جبر رابطه‌ای مبنای ریاضی زبان SQL بود، مفهوم ریاضی وابستگی ها نیز مبنای ریاضی بحث نرمال سازی(که در فصل بعدی شرح می‌دهیم) می‌باشد. وابستگی ها سه نوع می باشند.

۱-وابستگی تابع(FD) که آنها را به طور کامل در این فصل شرح می‌دهیم.

۲- وابستگی چند مقداری (MVD)

۳-وابستگی پیوندی (JD)

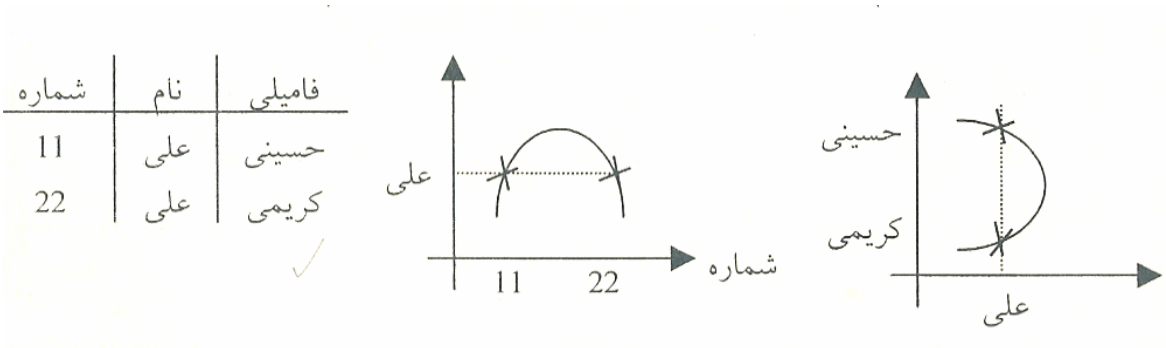
ثابستگی های JD,MVD را در فصل بعدی به صورت خلاصه بیان می کنیم چرا که کمتر مورد استفاده قرار می‌گیرند. MVD ها حالت کلی FD ها و JD ها حالت کلی MVD ها می‌باشند.

وابستگی تابعی (FD=Functional Dependency)

صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد و می‌نویسیم $R.X \rightarrow R.Y$ اگر و فقط اگر در طول حیات رابطه، به هر مقدار X در رابطه R، دقیقاً یک مقدار Y از رابطه R متناظر باشد. X,Y می‌توانند صفات مرکب باشند. اصطلاحاً می‌گوئیم صفت خاصه X خاصه Y را تعیین می‌کند.

مثال ۱: در جدول زیر نام، تابعی از شماره است، ولی فامیلی تابعی از نام نیست.

فامیلی	نام	شماره
حسینی	علی	۱۱
کریمی	علی	۲۲



مثال ۲:در جدول S، هر یک از صفات خاصه Sname, city,status با صفت خاصه S# از همین جدول وابستگی تابعی دارند زیرا به هر مقدار S# در این رابطه فقط یک مقدار از Sname، یک مقدار از Status و یک مقدار از city متناظر است. یعنی :

$$S.S\# \rightarrow S.city, S.S\# \rightarrow S.Status, S.S\# \rightarrow S.sname$$

یا به طور خلاصه:

$$S.S\# \rightarrow S.(sname, status, city)$$

مثال۳: در جدول SP داریم: $SP.(S\#,P\#) \rightarrow SP.Qty$ یعنی Qty با (S#,P#) وابستگی تایعی دارد.

نکته: اگر X کلید کاندید و به ویژه کلید اصلی رابطه R باشد، آنگاه هر صفت خاصه دیگر این رابطه الزاماًبا X وابستگی تابعی دارد چرا که طبق تعریف کلید کاندید یکتائی مقدار دارد. البته در تعریف وابستگی تابعی الزامی ندارد که صفت خاصه X کلید رابطه R باشد، به بیان دیگر لزومی ندارد که مقدار X فقط در یک تاپل از رابطه R وجود داشته باشد.

مثال ۴: در جدول SP' زیر داریم : $SP'.S\# \rightarrow SP'.status$

SP'

S#	P#	Qty	Status
S1	P1	300	20
S1	P2	200	20
S1	P3	400	20
S1	P4	200	20
S1	P5	100	20
S1	P6	100	20
S2	P1	300	10
S2	P2	400	10

توجه کنید که در جدول SP' مقدارS# تکرار شده است ولی برای هر P# فقط یک مقدار Status وجود دارد.

با توجه به مثال فوق می توان تعریف زیر را نیز برای FD ارائه کرد:

صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد اگر و فقط اگر هر وقت در دو تاپل از R، یک مقدار X وجود داشته باشد، مقدار Y نیز در آن دو تاپل یکسان باشد.

تعریف فوق مشابه تعریف تابع در ریاضیات معمولی است که می‌گوید: رابطه‌ای تابع است که به ازاء هر زوج مرتب که عضو اول یکسان دارند، عضو دوم آنها نیز یکسان باشد. مثلاً :

رابطه روبرو تابع است اگر b=c باشد. $R=\{(a,b)(a,c)\}$

تعریف : به سمت چپ یک FD ، **دترمینان** و به سمت راست آن **وابسته** می‌گویند.

مثلاً در $A \to B$ ، به A دترمینان و به B وابسته گفته می‌شود.

نکته ۱: FDها در واقع محدودیت جامعیت را نشان می‌دهند و بنابراین DBMS باید آنها را اعمال کند. به عنوان مثال واقعیت $S\# \to city$ بدین معناست که هر عرضه کننده منحصراً در یک شهر قرار دارد. FDها یک مفهوم ادراکی هستند.

نکته ۲: اگر در رابطه R داشته باشیم: $A \to B$ لزوماً $B \to A$ برقرار نیست.

نکته ۳: وابستگی تابعی بین صفات یک رابطه، یک مفهوم مستقل از زمان است یعنی فقط در مقدار خاصی از متغیر رابطه‌ای R و در لحظه خاصی وجود ندارد، بلکه این وابستگیها، در صورت وجود ، در جمیع مقادیر R و همیشه برقرارند.

نکته ۴: اگر K سوپر کلید رابطه R باشد در این صورت $K \to R(H)$ که در آن H مجموعه عنوان R است.

نکته ۵: در رابطه تمام کلید ، بین اجزاء کلید ، وابستگی تابعی وجود ندارد.

وابستگی تابعی کامل (FFD=Full Functional Dependency)

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. در این تعریف صفت X را مرکب فرض کرده‌ایم اگر صفت خاصه X مرکب نباشد وابستگی حتماً کامل خواهد بود.

مثال ۵: در رابطه S صفت خاصه city با صفت خاصه مرکب (S# ,Sname) وابستگی دارد یعنی $(S\#,sname) \to city$ ولی این وابستگی کامل نیست زیرا $S\# \to city$ یعنی city با یکی از زیر مجموعه های (S# ,sname) وابستگی تابعی دارد.

مثال۶: در رابطه SP صفت خاصه Qty با صفت خاصه مرکب (S#,P#) وابستگی تابعی کامل دارد زیرا $(S\#,P\#) \to Qty$و Qty با هیچیک از دو جزء S# یا P# به تنهایی وابستگی ندارد.

تعریف: اگر برای تمام صفتهای B در A داشته باشیم $A \to B$ آنگاه A را **ابر کلید R** می‌نامند و اگر این وابستگی از نوع FFD باشد آنگاه A را **کلید کاندید R** است.

تعریف: اگر B زیر مجموعه‌ای از A باشد آنگاه همواره $A \to B$. این وابستگی تابعی را بدیهی (trivial FD) می‌نامند.

مجموعه پوششي وابستگي (بستار وابستگي)

هنگام طراحی یک بانک در قدم اول می‌بایست از همه وابستگی های موجود بین صفات خاصه مطلع شد. تعدادی از وابستگی ها ممکن است بدیهی بوده و شناسائی و فهم آنها ساده باشد. همچنین ممکن است کلیدها از همان ابتدا مشخص و معین باشند. ولی گاهی اوقات نیز بعضی از وابستگی‌ها مستتر است کلیدها از همان ابتدا مشخص و معین باشند. ولی گاهی اوقات نیز بعضی از وابستگی‌ها مستتر بوده و ممکن است در نگاه اول طراح بانک آنها را نبیند.

لذا می‌بایست روش سیستماتیکی بنا نهاد که به کمک آن بتوانیم وابستگی‌های دیگری را بدست آورده و یا بعضی از آنها که حرف تازه‌ای نمی زنند را حذف کنیم. و همچنین بتوان کلید کاندید را بدست آورد. در واقع وابستگی ها قواعد بانک را مشخص می‌سازند.

تعریف: اگر F یک مجموعه از وابستگی های تابعی باشد آنگاه مجموعه تمام وابستگی‌های تابعی که از آن منتج می‌شود را مجموعه پوششی یا بستار (closure) مـی نـامیم و بـا F^+ نمـایش می‌دهیم.

آقای آرمسترانگ در سال ۱۹۷۴ ثابت کرد که با اعمال مکرر سه قاعده زیـر مـی‌تـوان بـه تمـام وابستگی‌های منتج دست یافت و هیچ وابستگی اضافی نیز تولید نمی‌شود.

۱- بازتاب (reflexivity) : اگر B زیر مجموعه A باشد آنگاه $A \to B$.

۲- افـزایش یـا بـسط پـذیری (augmentation) اگـر $A \to B$ و C صـفت باشـد آنگـاه $AC \to BC$.

۳- انتقال یا تعدی (transitivity): اگر $A \to B$ و $B \to C$ آنگاه $A \to C$.

هر چند قاعده‌های فوق برای استخراج F^+ کفایت می‌کرد ولی اعمال آنهـا مـشکل بـود. بعـدها دیگران قواعد دیگری را بیان کردند که کار را سهولت بخشید و مهمترین آنها عبارتند از :

۴- اجتماع (union): اگر $A \to B$ و $A \to C$ آنگاه $A \to BC$

۵- تجزیه(decomposition) : اگر $A \to BC$ آنگاه $A \to B$ و $A \to C$

۶- ترکیب(Composition): اگر $A \to B$ و $C \to D$ آنگاه $AC \to DB$

۷- خود تعیینی (self-determination) $A \to A$

۸- شبه تعدی(pseudotransitivity): اگر $A \to B$ و $BC \to D$ آنگاه $AC \to D$

۹- اگر $A \to B$ و $AB \to C$ آنگاه $A \to C$

۱۰- اتحـــاد کلـــی(General unification) : اگـــر $A \to B$ و $C \to D$ آنگـــاه $A \cup (C - B) \to BD$

مثال۷:فرمول روبرو را اثبات کنید. $A \to B, AB \to C \Rightarrow A \to C$

حل:

$A \to B \Rightarrow A \to AB$
$A \to AB, AB \to C \Rightarrow A \to C$

مثال ۸: فرض کنید متغیر رابطه R با صفات FD,F,E,D,C,B,A های زیر موجودند:

$F = \{A \to BC, B \to E, CD \to EF\}$

آیا وابستگی $AD \to F$ برای R برقرار است یا خیر؟

حل:

$A \to BC \Rightarrow A \to C$
$A \to C \Rightarrow AD \to CD$
$AD \to CD, CD \to EF \Rightarrow AD \to EF$
$AD \to EF \to AD \to F$

نکته۱: از $A \to BC$ دو وابستگی $A \to C, A \to B$ نتیجه می‌شود ولی در حالت کلی از $AB \to C$ نمی‌توان نتیجه گرفت $A \to C$ و یا $B \to C$

نکته۲: سه قانون اولیه آرمسترانگ یعنی بازتاب ، افزایش و تعدی ، قوانین کامل هستند یعنی با توجه به مجموعه F از وابستگی ها و فقط با اعمال این سه قانون می‌توان F^+ را بدست آورد. از طرف دیگر این قواعد معتبر هم هستند یعنی هیچ FD اضافه بر آنچه قابل استنتاج است، تولید نمی‌شود.

نکته ۳: دو مجموعه وابستگی‌های تابعی F و D معادل و یا هم ارزند اگر F^+ مساوی با G^+ باشد.

مثـــال ۹: فـــرض کنیـــد داریـــم R=(S,T,U,V,W) و $F = \{S \to T, V \to SW, T \to U\}$ وابستگی‌های تابعی دیگر را بدست آورید.

$S \to T, T \to U \Rightarrow S \to U$
$V \to SW \Rightarrow V \to S, V \to W$
$V \to S, S \to T \Rightarrow V \to T$
$V \to S, S \to U \Rightarrow V \to U$
$V \to S, V \to T, V \to U, V \to W$

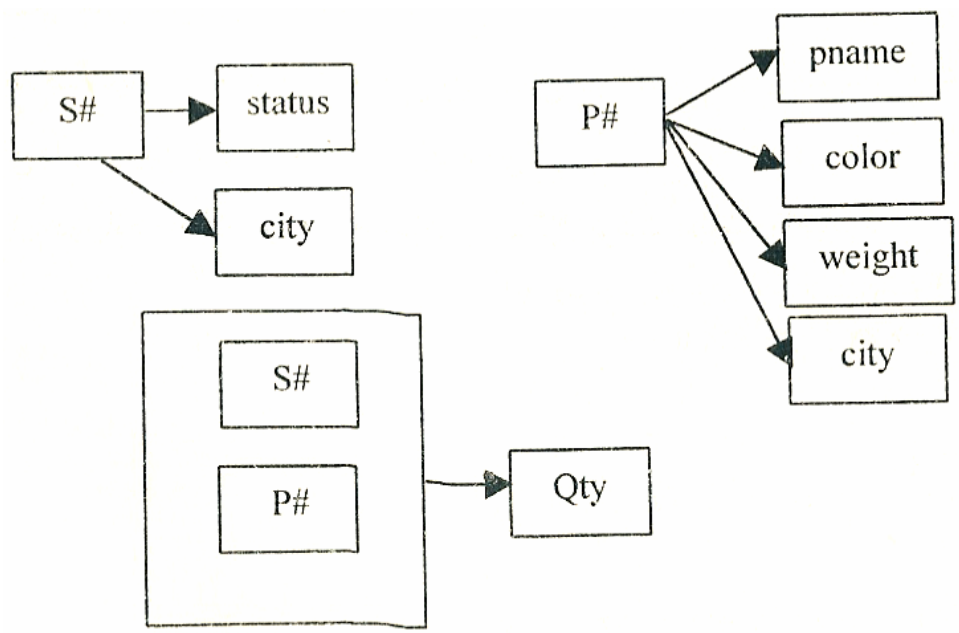
پس داریم :

یعنی ۷ کلید کاندید است.

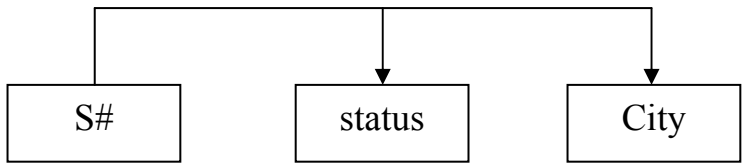
نمودار وابستگي تابعي (FD Diagram)

به کمک این نمودار وابستگی‌های تابعی یک بانک ترسیم مـی‌شـود. در ایـن نمـودار صـفتها در مستطیل قرار می‌گیرند و پیکانی از آنها به هر یک از صفتهای وابسته به آن رسم می‌شود.

مثال ۱۰: نمودار وابستگی SP,P,S به شکل زیر است(برای سادگی در جدول S فیلد Sname را حذف کرده‌ایم)



تذکر: اغلب پیکانهائی که وابستگی به کلید اصلی را نشان می‌دهند در بالای صفتها و سایر پیکانها زیر آنها کشیده می‌شوند. همچنین معمولاً ابتدا مجموعه پوششی بهینه وابستگی‌ها را بدست آورده و سپس نمودار وابستگی ترسیم می‌شود. مثلاً نمودار FD جدول S معمولاً به صورت زیر ترسیم می‌شود:



مجموعه وابستگي بهينه و مجموعه وابستگي كهينه(كاهش ناپذير)

با اعمال قواعد آرمسترانگ وابستگی‌های زیادی به دست می آید که تعدادی از آنها اضافی و تکراری هستند. در زیر روشی را برای حذف اینگونه وابستگی‌ها زائد و رسیدن به مجموعه وابسته بهینه ارائه می کنیم.

تعریف: دو مجموعه وابستگی تابعی F_1, F_2 معادل یا هم‌ارز هستند اگر مجموعه پوششی آنها برابر هم باشد یعنی $F_1^+ = F_2^+$.

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

۱- سمت راست هر وابستگی فقط یک صفت باشد.

۲- هر صفتی که F^+ را تغییر نمی‌دهد از سمت چپ حذف شود.

۳- وابستگی های تکراری و اضافی حذف شود.

بطور خلاصه باید گفت که برای یافتن وابستگی‌های تابعی در یک بانک ابتدا مجموعه پوشـشی وابستگی ها را تعیین کرده و سپس انرا بهینه می‌کنیم.

مثال ۱۱: در بانک اطلاعاتی زیر مجموعه وابستگی‌پوششی بهینه را بیابید.

$R=\{u,v,w,x,y,z\} \qquad F=\{u \to xy, x \to y, xy \to zv\}$

$u \to xy, xy \to zv \Rightarrow u \to zv \Rightarrow u \to z, u \to v$

حل :

$u \to xy \Rightarrow u \to x, u \to v$

$x \to y, xy \to zv \Rightarrow x \to zv \Rightarrow x \to z, x \to v$

پس:

$F_{opt}=\{u \to x, u \to y, x \to y, x \to z, x \to v, u \to z, u \to v\}$

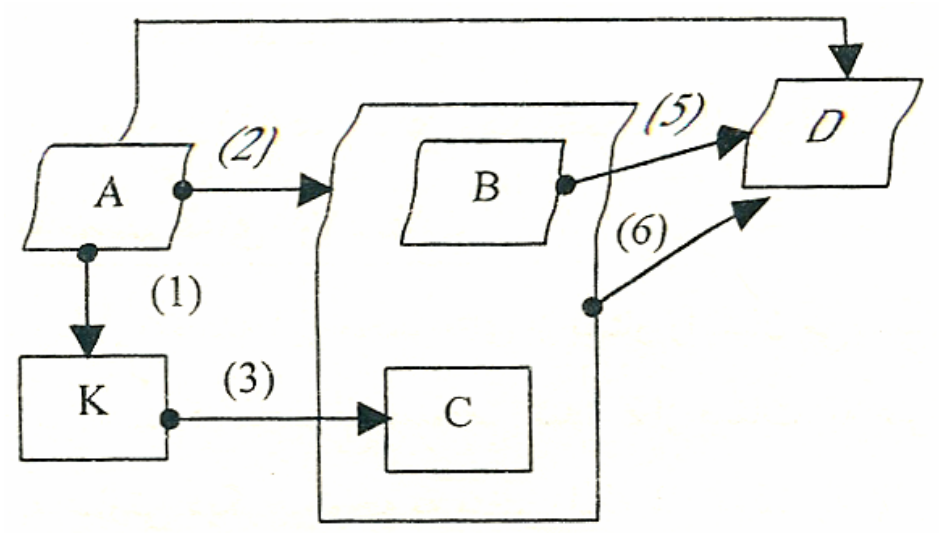
توجه کنید با توجه به خاصیت انتقال $u \to v, u \to z, u \to y$ بدست مـی‌آینـد پـس اگـر از مـا وابستگی کمینه(کهینه) را خواستند جوتب به صورت زیر است:

$F_{\min}=\{u \to x, x \to y, x \to z, x \to v\}$

مثال ۱۲: در یک رابطه وابستگی‌ها به صورت زیر است:

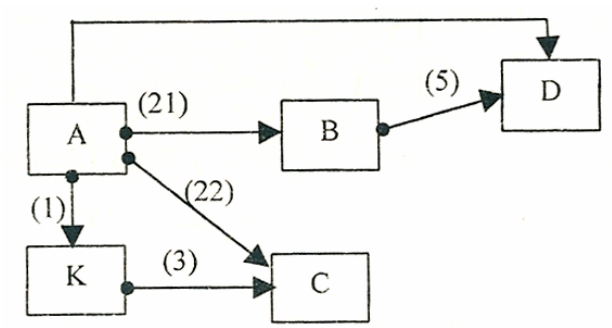
$A \to (B,C) \qquad A \to D \qquad A \to K$
$K \to C \qquad B \to D \qquad (B,C) \to D$

نمودار وابستگی را ترسیم کنید. سپس مجموعه کهینه این وابستگی ها را بدست آورده و نمودار آن را ترسیم کنید.



الف) $(B,C) \to D$ زائد است چون $B \to D \Rightarrow (B,C) \to D$ پس فلش (6) را حذف می کنیم.

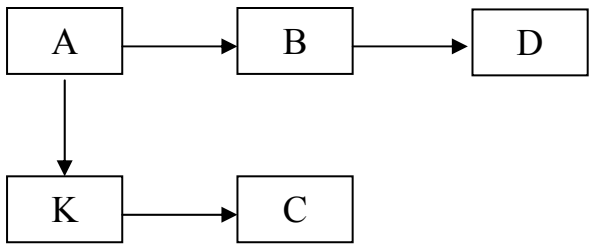
ب) از $A \to (B,C)$ می توان نتیجه گرفت $A \to C, A \to B$ پس می توان فلش(2) را حذف کرد و به جای آن دو فلش یکی از A به B و دیگری از A به C ترسیم کرد.



ج)فلش شماره۴ زائد است چرا که $A \to B, B \to D \Rightarrow A \to D$

د) فلش شماره ۲۲ زائد است چرا که $A \to K, K \to C \Rightarrow A \to C$

پس خلاصه ، شکل کمینه وابستگی به صورت زیر است.



همانطور که مشاهده می‌کنید در شکل کمینه وابستگی‌ها می بایست: ۱-سمت راست هر FD فقط یک صفت باشد. ۲- سمت چپ هر FD کاهش ناپذیر باشد. ۳-FD زائدی وجود نداشته باشد یعنی هیچ FD به کمک FD های دیگر بدست نیاید.

نکته: برای هر مجموعه از FD ها، حداقل یک مجموعه هم ارز وجود دارد که کاهش ناپذیر است.

بستار (Closure) مجموعه‌اي از صفات

در این قسمت چگونگی محاسبه زیر مجموعه ای از بستار را نشان می‌دهیم. یعنی زیر مجموعه ای که حاوی تمام FD هایی است که مجموعه مشخص شدهٔ Z از صفات ، به عنوان بخش سمت چپ آنها تعیین شده است. به عبارتی دیگر با توجه به متغیر رابطه R ، مجموعه Z از صفات R و مجموعه F از FD ها که برای R برقرار است، می‌توانیم مجموعه‌ای از تمام صفات R را پیدا کنیم که از نظر تابعی به Z وابسته است و «بستار Z^+ از Z تحت F» نام دارد. الگوریتم ساده زیر این کار را انجام می‌دهد:

$Z^+ := Z;$

do

$for\ each \quad X \to Y\ in\ F \quad do$

$if\ \ each\ X \subseteq Y\ in\ F\ do$

$whit(Z^{+}\ did\ not\ change)$

مثال ۱۳:اگر R=(S,T,U,V,W) و $F = \{S \to T, V \to SW, T \to U\}$ باشد، آنگاه:

الف) $\{S,V\}^{+}$ و ب) $\{V\}^{+}$ را بدست آورید.

حل الف)

$$Z^{+} = \{S,V\} \quad , \quad S \to T \Rightarrow Z^{+} = \{S,V,T\}$$
$$Z^{+} = \{S,V,T\}, \ V \to SW \Rightarrow Z^{+} = \{S,V,T,W\}$$
$$Z^{+} = \{S,V,T,W\}, T \to U \Rightarrow Z^{+} = \{S,V,T,W,U\}$$
$$\{S,V\}^{+} = \{S,V,T,W,U\}$$

با تکرار الگوریتم دیگر Z^{+} تغییری نمی‌کند پس از آنجا که {S,V} تمام صفات R را می‌دهند پس {S,V} یک سوپر کلید رابطه R می‌باشد.

ب)

$$Z^{+} = \{V\}, S \to T \Rightarrow Z^{+} = \{V\}$$
$$Z^{+} = \{V\}, V \to SW \Rightarrow Z^{+} = \{S,V,W\}$$
$$Z^{+} = \{V,S,W\}, T \to U \Rightarrow Z^{+} = \{V,S,W\}$$

حلقه do-while را دوباره اجرا می‌کنیم:

$$Z^{+} = \{V,S,W\}, S \to T \Rightarrow Z^{+} = \{V,S,W,T\}$$
$$Z^{+} = \{V,S,W,T\}, T \to U \Rightarrow Z^{+} = \{V,S,W,T,U\}$$

در نتیجه داریم:

$$\{V\}^{+} = \{V,S,W,T,U\}$$

یعنی V نیز یک سوپر کلید است. با توجه به تعریف کلید کاندید در می‌یابیم که {S,V} کلید کاندید نمی‌باشد و V کلید کاندید است.

بدست آوردن بستار مجموعه ای از صفات دو کاربرد اصلی دارد:

۱- با توجه به مجموعه‌ای از FD ها به نام F می‌توانیم بگوئیم که آیا وابستگی خاصی مثل $X \to Y$ فقط و فقط وقتی برقرار است که Y زیر مجموعه ای از بستار X^+ (تحت F) باشد.

۲- به کمک بدست آوردن X^+ می‌توان تعیین کرد که آیا X سوپر کلید (فوق کلید)هست یا خیر؟

به عبارت دیگر X یک کلید فوق کلید است، اگر و فقط اگر بستار X^+ دقیقاً مجموعه‌ای از تمام صفات R باشد و X یک کلید کاندید است اگر و فقط اگر یک سوپر کلید کاهش ناپذیر باشد.

تذکر: برای محاسبه بستار Z^+ تحت F ، بهتر است ابتدا F را بهینه کنیم.

بدست آوردن کلیدهای کاندید

برای یافتن کلیدهای کاندید بهتر است ابتدا F را بهینه کنیم سپس با توجه به نکات زیر کلیدهای کاندید را بدست می آوریم.

۱-هر کلید کاندید شامل مجموعه ای از صفتهایی است که در سمت چپ پیکانها می آیند.

۲-کلید کاندید باید کمینه باشد یعنی زیر مجموعه‌ای از آن خاصیت کلیدی نداشته باشد.

۳-ممکن است چند کلید کاندید وجود داشته باشد.

۴-کلیدهای کاندید ممکن است در یک یا چند صفت مشترک باشند.

مثال ۱۴: اگر $F = \{S \to T, V \to SW, T \to U\}$ و R=(S,T,U,V,W) باشد آنگاه کلید کاندید این رابطه چیست؟

حل:

$$S \to T, T \to U \Rightarrow S \to U$$
$$V \to SW \Rightarrow V \to S, V \to W$$
$$F = \{S \to T, S \to U, T \to U, V \to S, V \to W\}$$

پس:

$$V \to S, S \to T \Rightarrow V \to T$$
$$V \to T, T \to U \Rightarrow V \to U$$
$$F = \{S \to T, S \to U, T \to U, V \to S, V \to T, V \to U, V \to W\}$$

از آنجا که V همه صفتهای دیگر را می‌دهد پس کلید کاندید است.

مثال ۱۵:اگر R=(A,B,C,D,E,F,G) و

$$F = \{AF \to BE, FC \to DE, F \to CD, D \to E, C \to A\}$$

باشد کلید کاندید این رابطه چیست؟

حل: ابتدا سمت راست وابستگی‌ها را به یک صفت تبدیل می کنیم:

$$F = \{AF \to B, AF \to E, FC \to D, FC \to E, F \to C, F \to D, D \to E, C \to A\}$$

حال باید صفتهای اضافی را از سمت چپ حذف کنیم:

$$F \to C, FC \to D \Rightarrow F \to D$$
$$F \to C, FC \to E \Rightarrow F \to E$$
$$F \to C, C \to A \Rightarrow F \to A$$
$$F \to A, AF \to B \Rightarrow F \to B$$
$$F \to A, AF \to E \Rightarrow F \to E$$

پس :

$$F_{opt} = \{F \to A, F \to B, F \to C, F \to D, F \to E, D \to E, C \to A\}$$

در نتیجه F همه صفتهای دیگر به جز G را می دهد. پس {F,G} کلید کاندید است.

این کلید کاندید منحصر به فرد است. زیرا هیچ صفتی G,F را نمی دهد یعنی در هر کلید کاندید این دو صفت لازم هستند.

مثال ۱۶: در رابطه زیر همه کلیدهای کاندید را بدست آورید.

R=(U,V,W,X,Y,Z,O,P,Q)

$$F = \{U \to VXQ, UVP \to O, \to YZ, UP \to XY\}$$

حل: ابتدا مجموعه بهینه F را پیدا می‌کنیم:

$$V \to VXQ \Rightarrow U \to V, U \to X, U \to Q$$
$$OQ \to YZ \Rightarrow OQ \to Y, OQ \to Z$$
$$U \to V, UVP \to O \Rightarrow UP \to O$$
$$UP \to XY \Rightarrow UP \to X, UP \to Y$$

$U \to X$ را قبلاً داشته‌ایم پس $UP \to X$ زائد است.

پس $F_{opt} = \{U \to V, U \to X, U \to Q, UP \to O, OQ \to Y, OQ \to Z, UP \to Y\}$

حال مجموعه صفتهای وابسته به تمام مجموعه صفتهای چپ پیکانها را می‌یابیم:

$$\{U\}^+, U \to V, U \to X, U \to Q \Rightarrow \{U, V, X, Q\}$$
$$\{U, P\}^+, U \to U, U \to X, U \to Q, UP \to O, UP \to Y, OQ \to Z \Rightarrow \{U, P, V, X, Q, O, Y, Z\}$$
$$\{O, Q\}^+, OQ \to Y, OQ \to Z \Rightarrow \{O, Q, Y, Z\}$$

نتیجه می‌گیریم که بیشترین صفتها را $\{U,P\}^+$ می‌دهد. تنها صفتی که وابسته $\{U,P\}^+$ نیست w می باشد پس {u,p,w} کلید کاندید است.

کلید کاندید دیگری بدست نمی آید زیرا از Q,O نمی‌توان به P,U,W رسید پس ابر کلید در اینحالت باید {O,Q,P,U,W} باشد که کلید کاندید نیست چرا که زیر مجموعه آن یعنی {P,U,W} کلید کاندید است.

فصل دهم

نرمال سازي

نرمال سازی چیست؟

هنگام طراحی یک بانک اطلاعاتی رابطه‌ای، این سؤال مهم مطرح می‌شود که:«با توجه بـه داده های عملیاتی و ارتباط بین موجودیتها، چند جدول می بایست طراحی کرد؟ در هر جدول چـه فیلدهایی باید قرارگیرد؟ رابطهٔ جدولها باید چگونه باشد؟» در این فصل به ایـن سـؤالات پاسـخ می‌گوئیم.

مثال۱: به سه جدول معروف SP,P,S (فصل هشتم) نگاه کنید. فرض کنیـد جـدول SP را بـه صورت (S#,P#,Qty,Status) SP' تعریف و جـدول S را بـه صـورت (S#,Sname,City) S' تعریف می کردیم، یعنی فیلد Status را از جدول S به جدول SP می‌بردیم. در اینـصورت جدول SP' به صورت روبرو می‌شد:

همانطور که مشاهده می‌گردد فیلد Status برای S1 همواره ثابت و مشخص است(عدد 20) و بی جهت در جدول SP' تکرار شده و بدین جهت افزونگی اطلاعات داریم.

SP′

S#	P#	Qty	Status
S1	P1	300	20
S1	P2	200	20
S1	P3	400	20
S1	P4	200	20
S1	P5	100	20
S1	P6	100	20
S2	P1	300	10
S2	P2	400	10

حال سؤال دیگری را مطرح می‌کنیم:«آیا می‌توان تمام اطلاعات جداول SP,P,S را در یک جدول ریخت؟»جواب این سؤال «بله» است.در این صورت بانک اطلاعاتی فقط یک جدول بوده و از دید کاربران و برنامه نویسان مسأله بسیار ساده می شود، چرا که دیگر نیازی به پیوند طبیعی با ضرب دکارتی جداول نداریم.

S#	Sname	Status	City	P#	Pname	Color	Weight	City	Qty
S1	Sn1	20	C2	P1	Nut	Red	12	C2	300
S1	Sn1	20	C2	P2	Bolt	Green	17	C3	200
S1	Sn1	20	C2	P3	Screw	Blue	17	C4	400

ولی این عمل ۳ اشکال اساسی دارد:

۱-افزونگی داده‌ها (Data redundancy) قبلاً بیان شد که افزونگی یعنی تکرار بی‌رویهٔ داده‌ها. در بانک اطلاعات رابطه‌ای، تکرار داده ها تنها راه برقراری ارتباط بین جداول است و از

آن به عنوان کلید خارجی یاد می‌شود. تکرار بیش از این، بی‌رویه است و افزونگی نام دارد. بدیهی است که جدول بالا تکرار بی‌رویه است و افزونگی نام دارد. بدیهی است که جدول بالا تکرار بی رویه دارد. وقتی جداول را ادغام می کنیم همواره افزونگی به میزان بزرگترین جدول رخ می‌دهد. این رخداد دو زیان بزرگ دارد یکی به هدردادن فضای حافظه و دیگری پائین آوردن سرعت .

۲-بی‌نظمی(anomaly) وجود افزونگی در جدول مثال فوق باعث آنومالی در تغییر داده‌ها می شود. مثلاً اگر شهر S1 تغییر کند باید در تمام رکوردها به دنبال S1 گشته و شهر آن را تغییر دهیم که عملی مشکل‌زا می‌باشد.

۳-مقادیر تهی(NULL Value) با ادغام جداول گاهی اوقات مجبور خواهیم بود برای نشان دادن بعضی از اقلام اطلاعاتی از NULL استفاده کنیم. مثلاً فرض کنید تهیه کننده S قطعه‌ای را تهیه نکرده است، آنگاه سطر مربوط به آن به صورت زیر ذخیره می‌شود:

S#	Sname	Status	City	P#	Pname	Color	Weight	City	Qty
S5	Sn5	30	C1	NULL	. . NULL	NULL	NULL	NULL	NULL

مقادیر تهی علاوه بر اینکه جای زیادی را اشغال می کنند مشکلاتی را نیزباعث می‌گردند که قبلاً بیان شده است. در هنگام طراحی بانک با روش‌های کلاسیک نمال سازی، جداول را طوری طراحی می‌کنیم که این مشکلات حداقل گردد. مجموعه روشهائی که در طراحی بانک

اطلاعاتی موجب کاهش افزونگی اطلاعات و آنومالی گردد و بر اساس آن روابط بـین اقـلام داده ها اداره شوند، نرمال سازی گفته می‌شود.

سطوح نرمال

کاد(واضع مدل رابطه‌ای) در ابتدا سه سطح نرمال 1NF, 2NF, 3NF را تعریف کرد ولی بعداً دانشمندان دیگر، صورتهای دیگری را نیز معرفی کردند.

سطوح نرمال بودن عبارتند از :

1NF	(First Normal From)
2NF	(Second Normal From)
3NF	Third Normal From)
BCNF	(Boyce/Codd Normal From)
4NF	(Fourth Normal From)
5NF	(Fifth Normal From)
DKNF	Domain-key Normal From)

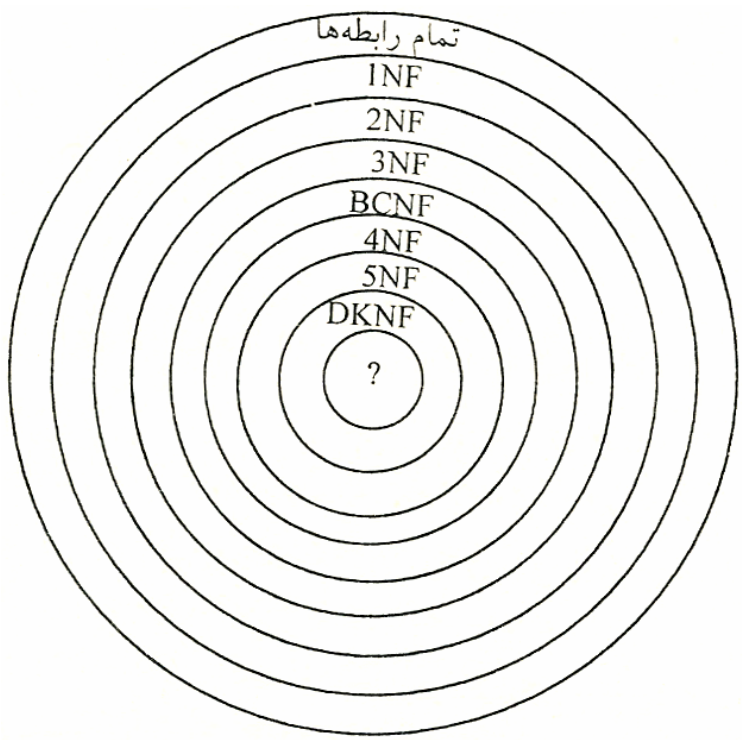
به طور کلی در مجموعه رابطه ها بعضی غیر نرمال(UNF=Unormalized From) هستند.

رابطه های نرمال همان 1NF است.

در واقع صور نرمال فوق، هر یک از قبلی خود نرمالتر هستند. یعنـی مـثلاً رابطـه‌ای کـه 3NF است حتماً 2NF و۱NF می‌باشد. به عبارتی دیگر در رابطه های 1NF برخی 2NF بوده و در رابطه‌های 2NF برخی 3NF می‌باشند و...

در عمل حالتی که یک رابطه BCNF باشد اما 4NF نباشد به ندرت رخ می‌دهد. لذا در عمل اگر رابطه را تا سطح BCNF (و حتی گاه تا 3NF) نرمال کنیم کفایت می کند.

شکل زیر ارتباط صورتهای نرمال را نشان می‌دهد.



تذکر: در حال حاضر به درستی نمی‌دانیم که آیا ممکن است صورتهای دیگر مثل 7NF,6NF نیز وجود دارند یا خیر. از اینرو به نظر می‌رسد که باید همان 5NF را آخرین صورت نرمال دانست هر چند که DKNF نیز مطرح شده است. ما در این کتاب تا 5NF را شرح می‌دهیم.

همواره این سوال برای طراحان بانک اطلاعاتی مطرح است که «آیا آنچه که ما ارائه کرده‌ایم بهترین است؟» یا به عبارتی دیگر بانک را به چه جداولی تقسیم کنیم، در هر جدولی چه

فیلدهایی وجود داشته باشد و رابطه بین جدولها چگونه باشد تا بانک بهترین حالت را داشته باشد.

در مدل رابطه‌ای روشی کاملاً کلاسیک و ریاضی گونه برای پاسخگوئی به سؤالات فوق وجود دارد که ره روش«نرمال سازی»(normalization) موسوم است. نرمال سازی در عمل یعنی پیروی از یک سری فرم‌های نرمال که منجر به تجزیه جداول می‌شوند.

فرم 1NF

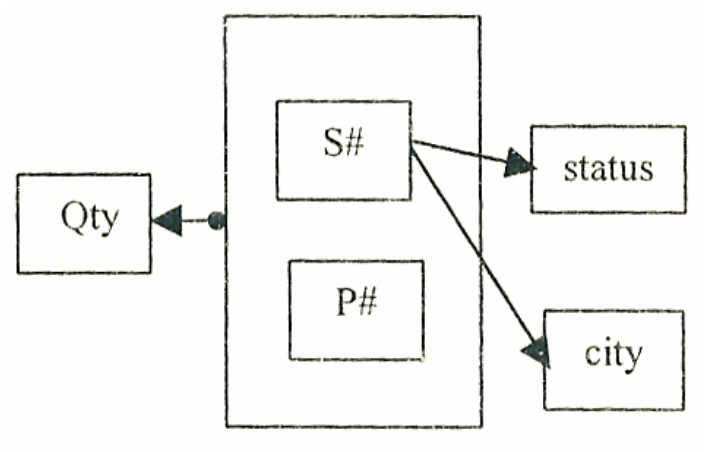
رابطه R به فرم 1NF (First Normal From) است اگر و فقط اگر تمام صفات خاصه آن روی میدانهای اتومیک تعریف شده باشند به عبارتی دیگر صفتهای آن از دامنه تودرتو (nested domain) نباشد. یعنی صفت ترکیبی نداشته باشیم. مثلاً اگر جدولی شامل فیلد تاریخ باشد که خود فیلد تاریخ از سه فیلد کوچکتر (سال-ماه-روز) تشکیل شده باشد آنگاه جدول NF۱ نیست. در واقع هر رابطه نرمالی 1NF است.

تذکر: از آنجا که در foxpro, Access فیلد تجزیه پذیر وجود ندارد پس تمامی جداول در آنها 1NF می باشد.

مثال ۲: فرض کنید به جای دو جدول مجزای SP,S یک جدول به نام First به صورت زیر تعریف می‌کردیم:

First(S# ,status city ,P#,Qty)

همچنین فرض کنید در این بانک این قاعده وجود دارد که وضعیت یک تهیه کننده از روی شهر او تعیین می شود(یعنی $city \rightarrow status$) بدین ترتیب نمودار وابستگی جدول First و نیز محتویات آن به صورت زیر خواهد بود:



S#	Status	City	P#	Qty
S1	20	C2	P1	300
S1	20	C2	P2	200
S1	20	C2	P3	400
S2	10	C3	P1	100
S2	10	C3	P2	200
S3	10	C3	P2	200
S4	20	C2	P2	200
S4	20	C2	P4	300

کلید اصلی این رابطه (S#,P#) است.

همانطور که مشاهده می شود اولین مشکل ایـن فـرم افزونگـی اطلاعـات اسـت. وجـود پدیـده افزونگی در این رابطه آنومالهایی ار ایجاد می‌کند که در زیر بیان می‌کنیم:

آنومالی درج: در جدول فوق نمی‌توان واقعیت «S7 در شهر C4 ساکن است» را درج کرد تا زمانیکه ندانیم چه قطعه‌ای را تهیه کرده است. دلیلش آن است که #P جزو کلید اصلی است و طبق قواعد جامعیت نمی‌تواند NULL باشد.

آنومالی حذف: اگر این اطلاع را که «S3 از P2 به تعداد 200 عدد تهیه کرده است» را حذف کنیم اطلاع«S3 ساکن شهر C3 است» نیز ناخواسته حذف می‌شود.

آنومالی بهنگام سازی: در این عمل با مشکل بهنگام سازی منتشر شونده مواجه هستیم زیرا مثلاً شهر یک تهیه کننده به دفعات تکرار شده است. اگر این بهنگام سازی منتشر شونده انجام نشود داده های بانک ناسازگار می شوند.

در واقع مشکلات فوق از آنجایی ناشی شدند که در جدول First فیلدهای status, city با کلید اصلی (#P, #S) وابستگی تابعی کامل ندارند(چرا که با #S وابستگی دارند و #S بخشی از کلید اصلی است. پس ما باید سعی کنیم این مورد را برطرف سازیم.

فرم 2NF

رابطه R در صورت دوم نرمال است اگر و فقط اگر 1NF باشد و هر صفت خاصه غیر کلید با کلید اصلی وابسته تابعی کامل داشته باشد. به عبارت دیگر هر صفت غیر کلید با کلید اصلی به طور کاهش ناپذیر وابسته باشد.

مثلاً رابطه First در مثال ۲ یک رابطه 2NF نیست چرا که صفات خاصه غیر کلید status , city با کلید اصلی (#S,#P) وابستگی تابعی کامل ندارند.

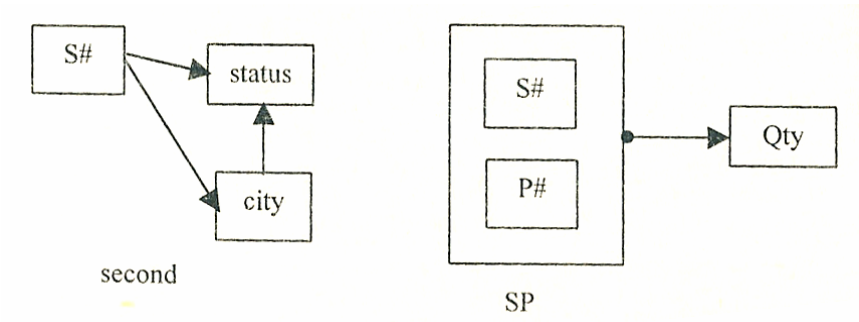
تذکر : صفت غیر کلید یعنی صفتی که خود کلید نباشد و جزء تشکیل دهنده کلید هم نباشد، صفتی است که جزء تشکیل دهنده کلید باشد را صفت عمده (prime Attribute) می نامیم.

در مثال ۲ صفت city غیر کلید و صفت S# از نوع عمده می‌باشند.

مثال ۳: رابطه First در مثال ۲ را به دو رابطهٔ زیر تجزیه می‌کنیم:

$$First(S\#, status, city, P\#, Qty) \rightarrow \begin{cases} second(S\#, status, city \\ SP(S\#, P\#, Qty \end{cases}$$

نمودار وابستگی این دو رابطه به شکل زیر خواهد بود:



بسط این دو رابطه جدید به صورت زیر خواهد بود:

S#	P#	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	200

S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

جدول SP

S#	Status	City
S1	20	C2
S2	10	C3
S3	10	C3
S4	20	C2
S5	30	C1

جدول second

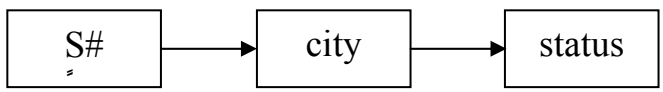
روابط فوق مشکلات رابطه First را ندارند. مثلاً می توان این اطلاع را که «S5 در شهر C1 است» در بانک درج کرد بدون آنکه بدانیم چه قطعه ای را تولید کرده است. همچنین می‌توان این اطلاع را که «S3 از P2 به تعداد 200 عدد تهیه کرده است» را حذف کرد بی آنکه اطلاع ناخواسته‌ای از بین برود . به همین ترتیب از آنجا که در رابطه Second شهر هر تهیه کننده فقط یکبار در رابطه وجود دارد لذا با مشکل بهنگام سازی منتشر شونده مواجه نیستیم.

در واقع رابطه Frist را با عملگر پرتو چنان تجزیه کرده‌ایم که وابستگی تابعی غیر کامل موجود در First از بین رفته است. حذف همین وابستگی تابعی غیر کامل سبب از بین رفتن مشکلات First شده است.

اما رابطه Second هنوز هم آنومالیهائی دارد(هر چند که آنومالیهای First را ندارد) از جمله در درج نمی‌توان این اطلاع را که«یک شهر خاص دارای یک مقدار مشخص وضعیت» است،

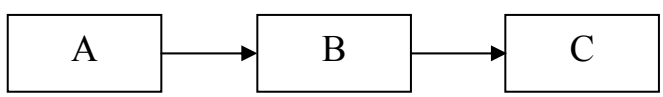
درج کرد. در واقع تا ندانیم چه تهیه کننده‌ای در شهر ساکن است درج اطلاعات فوق ناممکن است(مقدار کلید اصلی#S در این حالت باد NULL شود که ناممکن است). همچنین اگر تاپلی از رابطه Second را حذف کنیم نه تنها تهیه‌کننده ای را حذف کرده ایم بلکه این اطلاع که «شهر خاصی دارای وضعیت مشخص است»نیز حذف می شود. البته این مشکل در حالتی که فقط یک تهیه کننده در شهر ساکن باشد بروز می کند. همچنین در بهنگام سازی وضعیت یک شهر چندین بار تکرار شده لذا سیستم باید بهنگام سازی منتشر شونده را انجام دهد.

علت بروز مشکلات فوق این است که status با صفت #S (کلید اصلی) وابستگی کامل دارد و از سوی دیگر Status از طریق city نیز باید #S وابستگی دارد. اصطلاحاً می‌گوئیم یک وابستگی با واسطه (تراگذاری- تعدی transitive) بین status و #S (با واسطه city) وجود دارد.



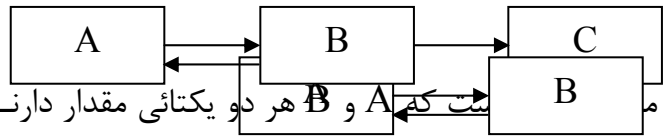
پس باید این نوع وابستگی‌ها را از بین ببریم تا مشکلات فوق برطرف گردد(در فرم 3NF اینکار را انجام خواهیم داد) به عبارت دیگر مشکلات فوق از آنجا ناشی می‌شود که دو فیلد غیر کلید با یکدگر وابستگی داشته باشند.

تعریف وابستگی یا انتقالی (Transitive) : اگر صفت B از رابطه R با صفت خاصه A از همین رابطه وابستگی داشته باشد و صفت خاصه C از همین رابطه با صفت خاصه B بستگی داشته باشد یعنی :



و A با B وابستگی نداشته باشد می گوئیم C با A وابستگی با واسطه دارد.

باید توجه داشت وابستگی شکل زیر «وابستگی با واسطه» نیست و باعث بروز آنومالی‌های فوق نخواهد شد.



چرا که وجود ارتباط م[illegible]ست که A و B هر دو یکتائی مقدار دارند و در نتیجه هر صفت خاصه دیگر ، مثل C هم با A وابستگی خواهد داشت و هم با B.

نکته: هر رابطه‌ای که در صورت اول نرمال باشد را همیشه می‌توان به تعدادی رابطه 2NF تبدیل کرد. در تبدیل رابطه 1NF به 2NF رد واقع باید چنان عمل کرد که وابستگی‌های تابعی غیر کامل موجود در رابط 1NF از میان بروند برای این منظور با انجام پرتوهای مناسب روی 1NF آنرا به 2NF تبدیل می‌کنیم.

نکته: در تجزیه 1NF به منظور رسیدن به رابطه ای 2NF هیچ اطلاعی از دست نمی‌رود زیرا با پیوند مجدد و مناسب رابطه های 2NF می توان به رابطه اولی رسید.

فرم 3NF

رابطه R در سطح 3NF است اگر و فقط اگر صفات خاصه غیر کلید (در صورت وجود):

الف) متقابلاً به یکدیگر ناوابسته باشند.

ب)با کلید اصلی رابطهR، وابستگی تابعی کامل داشته باشند.

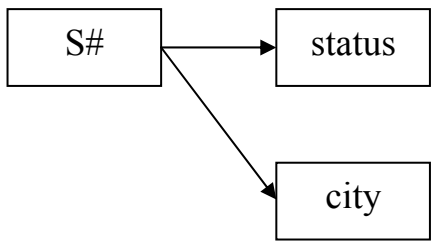
منظور از «صفت خاصه غیر کلید» یعنی صفت خاصه‌ای که جزئی از کلید اصلی رابطه نباشد.

مثال ۴: رابطه P(P# ,pname ,color,weight,city) رابطه ای در سطح 3NF است زیرا همه صفات با کلید اصلی رابطه (یعنی P#) وابستگی تابعی کامل دارند و همه صفات غیر کلید weight, city, color,pname ناوابسته به یکدیگرند. ولی جدول second در مثال ۱۷ در سطح 3NF نیست(فقط در سطح 2NF است) چرا که فیلد غیر کلید status وابسته به فیلد غیر کلید city می‌باشد. این موضوع ایجاد آنومالیهائی می‌کرد که می‌بایست با تجزیه جدول second و تبدیل آن به جداول 3NF برطرف گردد.

رابطه 3NF را می‌توان به صورت زیر نیز تعریف کرد: رابطه R در صورت سوم نرمال است اگر و فقط اگر 3NF بوده و هر صفت خاصه غیر کلید بطور مستقیم (بی واسطه) با کلید اصل وابستگی داشته باشد.

مثال۵: جدول Second را به دو جدول 3NF تجزیه کنید.

یک تجزیه رابطه second(S# ,city ,status) با نمودار وابستگی روبه‌رو به صورت زیر است:

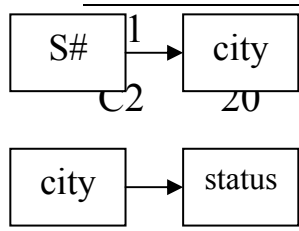


second $\Rightarrow$ SC(S# ,city),CS (city ,status)

بسط دو رابطه اخیر به صورت زیر خواهد شد:

SC	S#	City
	S1	C2
	S2	C3

CS	City	Statu s
	C1	
	C2	20



S3	C3	C3	10
S4	C2		
S5	C1		

در اینحالت جداول حاصله آنومالیهای second را ندارند. در درج می‌توان این اطلاع را که «شهر خاص دارای وضعیت مشخص است» را به راحتی درج کرد بی‌انکه لازم باشد بدانیم چه تهیه کننده ای در آن شهر ساکن است. در حذف اگر تاپلی را از رابطه SC حذف کنیم اطلاع در مورد وضعیت شهر آن از دست نمی رود. در بهنگام سازی وضعیت یک شهر فقط یک بار در رابطه CS وجود دارد و لذا با مشکل بهنگام سازی منتشر شونده مواجه نیستیم. روابط SC CS, هر دو 3NF هستند.

نکته : همیشه می توان رابطه‌ای را که 2NF است به رابطه های 3NF تبدیل کرد و در این تبدیل هیچگونه گمشدگی اطلاعات نخواهیم داشت.

تجزیه مطلوب(خوب)

تا اینجا برای تبدیل روابط 1NF به 2NF و همچنین 2NF به 3NF از عمل تجزیه جداول استفاده کردیم. ولی تجزیه یک جدول به صورتهای مختلفی امکان‌پذیر است. سئوال اصلی آن است که یک جدول چگونه باید تجزیه شود، یعنی در هر زیر جدول چه صفاتی باید قرار داده شود؟ در این قسمت به این سوال پاسخ می‌گوئیم.

تعریف: تجزیه ای که در آن با پیوند رابطه های حاصل از تجزیه ، همان محتوای اطلاعاتی رابطه اولیه بدست آید و اطلاعات حشو یا افزونه(ناموجود در رابطه اولیه) پدید نیاید و به علاوه تمام

وابستگی‌های تابعی رابطه اولیه محفوظ بماند تجزیه‌ای مطلوب است. به عبارت دیگر تجزیه‌ای خوب است که بی حشو و حافظ وابستگیها باشد.

تذکر: همیشه پس از تجزیه یک رابطه به دو رابطه و پیوند مجدد رابطه‌های حاصله لزوما به رابطه نخستین نمی رسیم و ممکن است در اینحالت تاپلهای اضافی پدید آید.

مثال۶: رابطه SPJ(S#,P#,J#) را با بسط زیر در نظر بگیرید که آن را یکبار روی صفات خاصه P# و S# و بار دیگر روی صفات خاصه P#,J# پرتو کرده‌ایم:

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

⟶

S#	P#
S1	P1
S1	P2
S2	P1

P#	J#
P1	J2
P2	J1
P1	J1

حال به صورت بر عکس دو جدول حاصله را روی صفت P# با هم پیوند می دهیم، حاصل برابر با جدولَ SPJ روبرو می‌شود:

S#	P#	J
S1	P1	J2
S1	P1	J1
S1	P2	J1
S2	P1	تاپل اضافه شده
S2	P1	J1

همانطور که مشاهده می‌کنید بر اثر پیوند تاپل اضافی(S2,P1,J2) که در رابطه اولیه وجود نداشته، پدیدآمده است.

یکی از پژوهشگران به نام ریسانن (Rissanen) ضوابطی را به صورت زیر، برای تجزیه مطلوب ارائه کرده است.

ضوابط ریسانن: تجزیه رابطهR به دو رابطه R2,R1 مطلوب است اگر R2,R1 مستقل از یکدیگر باشند. R2,R1 مستقل از یکدیگرند اگر و فقط اگر:

۱-صفت مشترک در دو رابطه، حداقل در یکی از آنها کلید کاندید باشد.

۲-تمام FD های موجود در R یا در مجموعه FDهای R2,R1 موجود باشند و یا از این مجموعه FD ها قابل استنتاج باشند.

ضابطه اولی تضمین می کند که با پیوندR2,R1 تاپل حشو ظاهر نشود و ضابطه دوم تضمین می‌کند که تجزیه حافظ وابستگیها باشد.

مثال۷: رابطه second(S#,sname ,city ,status) را به سه صورت زیر می‌توان تجزیه کرد که ما قبلاً از تجزیه الف استفاده کردیم:

الف)SC(S# ,city) , CS(city ,status)

ب) SC(S#,city) , SS(S#,status)

ج) SS(S# ,status) , CS(city,status)

کدام تجزیه مطلوب است؟

تجزیه(ب) دارای مشکلاتی است مثلاً در (ب) نمی توان این اطلاع را که«شهر خاصی دارای مقدار وضعیت مشخصی است» را درج کرد تا زمانی که ندانیم چه تهیه کننده ای در آن شهر ساکن است. در تجزیه (ج) اگر وضعیت شهری تغییر کند این تغییرهم در جدول CS و هم در جدول SS باید اعمال شود. در واقع این مشکلات از آنجا ناشی می شود که پرتوهای روابط ب

و ج به یکدیگر وابسته اند پس تجزیه خوب تجزیه‌ای از پرتوهای آن، پرتوهای مستقل از یکدیگر باشند.

حال با توجه به ضوابط ریسانن تجزیه مطلوب را تشخیص می‌دهیم.

تجزیه الف یک تجزیه خوب است . چرا که شط اول قضیه ریسانن را دارد:

$$S\# \to city, city \to status \Rightarrow S\# \to status$$

همچنین شرط دوم را نیز داراست. صفت مشترک بین دو رابطه تجزیه شده city می‌باشد که city در رابطه CS کلید کاندید است.

ولی تجزیه (ب) در مثال قبلی تجزیه خوبی نیت چرا که رابطه $city \to status$ قابل استنتاج نمی‌باشد یعنی شرط دوم قضیه ریسانن برقرار نیست.

همچنین تجزیه (ج) در مثال قبلی نیز، تجزیه خوبی نیست چرا که فیلد مشترک دو جدول status است و status کلد کاندید هیچکدام از دو جدول نیست پس شرط اول قضیه ریسانن را ندارد. البته تجزیه (ج) شرط دوم را هم ندارد چرا که $S\# \to city$ را نمی توان استنتاج کرد.

تعریف: رابطه‌ای که به پرتوهای مستقل تجزیه نمی شود، به رابطه تجزیه ناپذیر (اتومیک) موسوم است.

قضیه هیث(Heath)

رابطه(R(A,B,C که در آن C,B,A سه مجموعه از صفات هستند مفروض است. اگر $A \to B$ آنگاه می توان R را به دو رابطه R1(A,B) و R2(A,C) تجزیه کرد و این تجزیه، مطلوب (خوب) است.

تذکر: بر اساس ضوابط ریسانن اکر در رابطه R(A,B,C) وابستگیهای $A \to B$ و $\mathbf{B} \to C$برقرار باشد، در این صورت تجزیه مطلوب به صورت زیر است:

$$R(A,B,C) \Rightarrow R1(A,B), R2(B,C)$$

الگوریتم تبدیل 1NF به 2NF و 2NF به 3NF

الگوریتم تبدیل جدول 1NF به چند جدول 2NF به صورت زیر است:

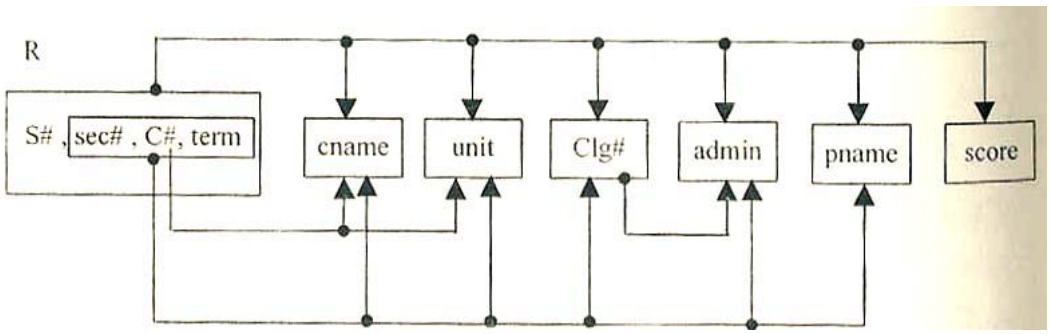
۱-هر بخش از کلید اصلی را که صفت وابسته دارد، با آن صفت ها کنار هم قرار می دهیم.

۲-کل کلید اصلی را با صفتهای باقی مانده کنار هم قرار می‌دهیم

۳-سایر وابستگی را ترسیم می کنیم.

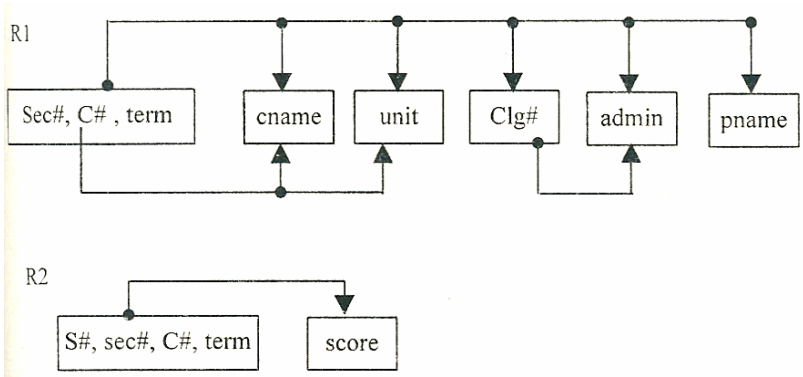
تذکر: اگر وابستگی‌به بخشی از کلید به صورت تودرتو بود آنگاه الگوریتم فوق را باید تکرار کنیم.

مثال ۸: فرض کنید نمودار وابستگی‌تابعی بانک اطلاعات ثبت نام دانشگاه به شکل زیر باشد.

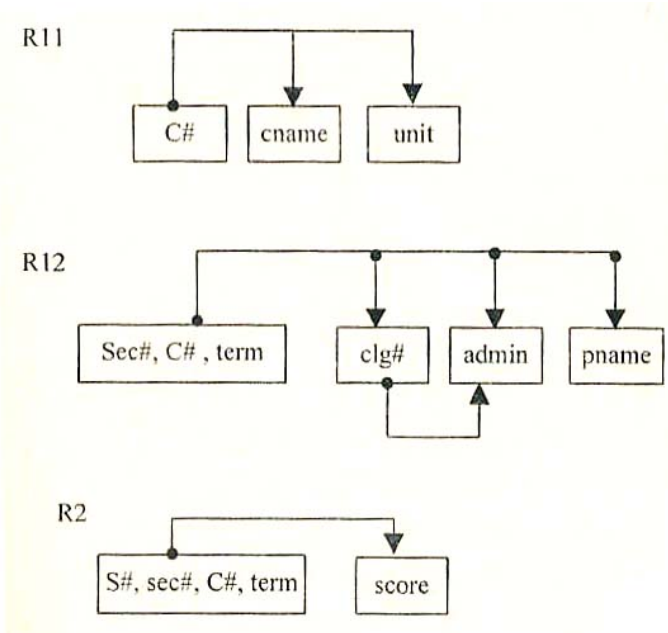


S# شماره دانشجو،Sec# شماره گروه. C# شماره درس، term شماره ترم(۱ یا ۲) ، cname نام درس، unit تعداد واحد ، Clg# شماره دانشکده، admin رئیس دانشکده، pname نام استاد و score نمره می‌باشد. کلید کاندید رابطه (S#,Sec#,C#,term) می‌باشد.

رابطه فوق 1NF بوده و 2NF نیست برای تبدیل آن به 2NF الگوریتم فوق را اعمال می کنیم:



ولی خود R1 هنوز در سطح 2NF نمی‌باشد و باید آن را با تجزیه به دو رابطه 2NF تجزیه کنیم . پس الگوریتم را دوباره برای R1 اعمال می کنیم و آن را به R11 و R12 تجزیه می کنیم.



حال هر سه جدول فوق در سطح 2NF می‌باشند. توجه کنید که معمولاً وابستگی به کلید اصلی را به صورت فلشهای بالای مستطیل ها ترسیم می‌کنیم.

حال الگوریتم تبدیل جدول 2NF به 3NF را بیان می‌کنیم:

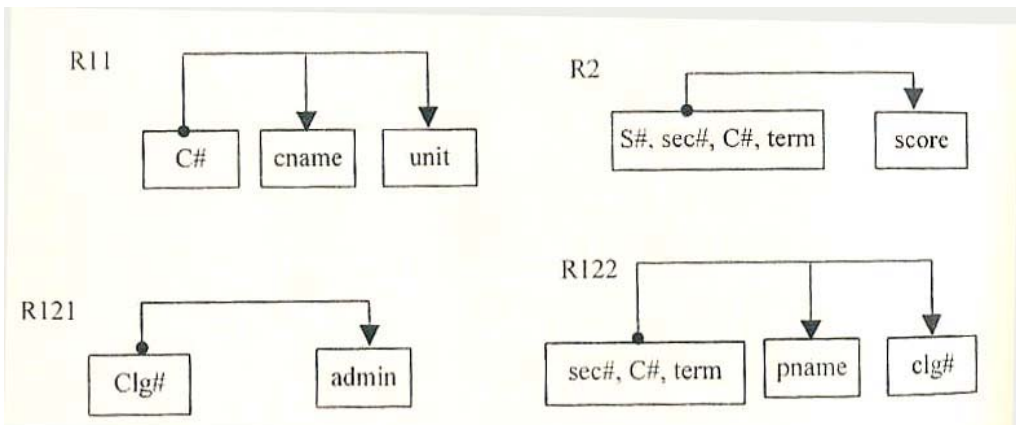
۱-صفتهایی را که وابستگی انتقالی ایجاد کرده اند ، با وابسته های آنها کنار هم قرار می دهیم.

۲-کلید اصلی را با صفتهای باقی مانده کنار هم قرار می‌دهیم.

۳-صفتهای کلیدی را به عنوان کلید خارجی در ۲ تکرار می‌کنیم.

مثال ۹: جداول مثال ۸ را به سطح 3NF ببرید.

حل: جداول R2,R11 در سطح 3NF هستند. جدول R12 را طبق الگوریتم فوق به جداول R121, R122 تجزیه می‌کنیم:



تمام جداول فوق در سطح 3NF می باشند.

فرم BCNF

صورت سوم نرمال در واقع تفسیر وابستگی تابعی کامل و بی واسطه تمام صفات خاصه با کلید اصلی می‌باشد. هر چند با 3NF کردن رابطه ها، افزونگی به مقدار قابل توجه‌ای کاهش می‌یابد ولی از سوی دیگر تبدیل یک رابطه به چند رابطه 3NF باعث افت کارایی سیستم در عمل بازیابی می شود، چرا که برای بازیابی اطلاعات موجود در رابطه اولیه، باید روابط 3NF بطور مناسب با یکدیگر پیوند داده شوند و این عمل زمانگیر است. گاهی اوقات برای اجتناب از این عمل پوند در عمل تا حدی باید افزونگی را پذیرفت.

به نظر دیت«رابطه‌ها را اقلا باید به صورت 3NF طراحی کرد و در صورت لزوم یک افزونگی کنترل شده در محیط فیزیکی ذخیره سازی ایجاد کرد، به بیانی دیگر افزونگی انتزاعی را باید حداقل کرد تا آنومالیهای عملیاتی در محیط انتزاعی کاهش یابند».

نکته :3NFدر مواردی که هر سه شرط زیر برقرار باشند ممکن است داشته باشند:

۱-وقتی که رابطه دارای چند کلید کاندید باشد.

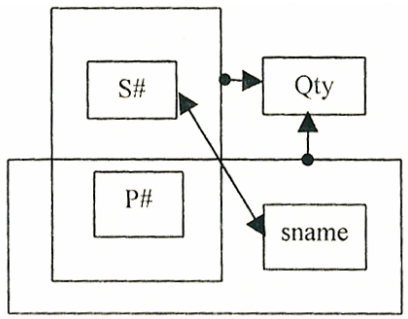
۲-وقتی که کلیدهای کاندید رابطه مرکب باشند.

۳-وقتی که کلیدهای کاندید با یکدیگر اشتراک صفت خاصه داشته باشند یعنی اقلا یک صفت خاصه در آنها مشترک باشد.

در این صورت این نوع جداول را تا سطح بیشتری باید نرمال سازی کرد.

مثالهای زیر این موضوع را نشان می دهند.

مثال۱۰: رابطه SSP(S#,Sname ,P#,Qty) را در نظر بگیرید . کلیدهای کاندید این رابطه عبارتند از (S#,P#), (Sname ,P#) نمودار وابستگی و یک بسط فرضی از این رابطه به صورت زیر است:



S#	Sname	P#	Qty
S1	Sn1	P1	300
S1	Sn1	P2	200
S1	Sn1	P3	400
S1	Sn1	P4	200

در این مثال Qty صفت غیر کلید و S#,Sname صفات عمده می‌باشند.

در رابطه فوق P# بین کلیدهای کاندید مشترک است. همانطور که مشاهده می‌شود رابطه فوق دارای افزونگی است لذا مشکل بهنگام سازی منتشر شونده را دارد.

توجه کنید رابطه فوق در 3NF است، اگر صفت خاصه‌ای خود جزئی از کلید کاندید باشد(یعنی صفت عمده باشد) طبق تعریف 2NF لزومی ندارد که با کلید اصلی وابستگی تابعی کامل داشته باشد، بدین ترتیب اگر فرض کنیم که (S#,P#) کلید اصلی است، این واقعیت که صفت خاصه Sname با آن وابستگی تابعی کامل ندارد نادیده گرفته می‌شود. توجه کنید که در تعریف 3NF,2NF بحث بر سر صفت غیر کلید است و نه صفت عمده.

به هر حال مثال فوق یک رابطه 3NF است که مشکل افزونگی داده‌ها را دارد. برای رفع مشکل فوق جدول را می‌بایست به دو جدول زیر تجزیه کرد(SP(S#,P#,Qty, SS(S#,Sname).

البته SP را می‌توان به صورت SP(Sname ,P# ,Qty) نیز در نظر گرفت. حال دو جدول SS, SP مشکل فوق را دیگر ندارد و این روابط حاصله در سطح BCNF قرار دارند.

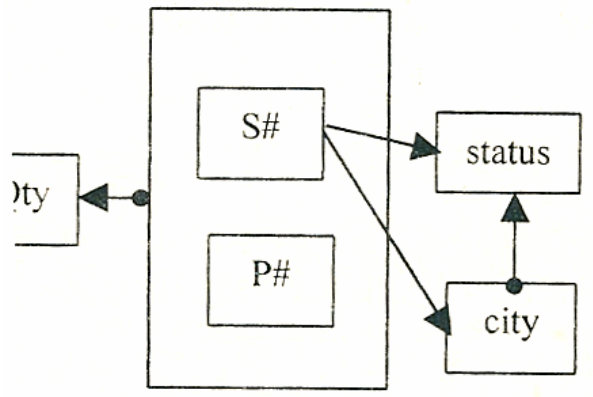
تعریف دترمینان(Determinant):هر صفت خاصه‌ای که صفت خاصه‌ای دیگر با آن وابستگی تابعی کامل داشته باشند، دترمینان نامیده می‌شود یعنی اگر داشته باشیم $R.A \rightarrow R.B$ و این وابستگی کامل باشد صفت خاصه A را دترمینان می‌نامیم.

تعریفBCNF: رابطه R در سطح BCNF است اگر و فقط اگر هر دترمینان، کلید کاندید باشد به عبارتی دیگر جدولی در BCNF است که ستونهای آن فقط به کلیدهای کاندیدش وابستگی تابعی داشته باشند. توجه کنید که در تعریف فوق تاکید بر روی کلید کاندید است و نه فقط کلید اصلی، همچنین تعریف BCNF ساده تر از تعریف 3NF است زیرا مبتنی بر مفاهیم 1NF و 2NF و وابستگی با واسطه نیست.

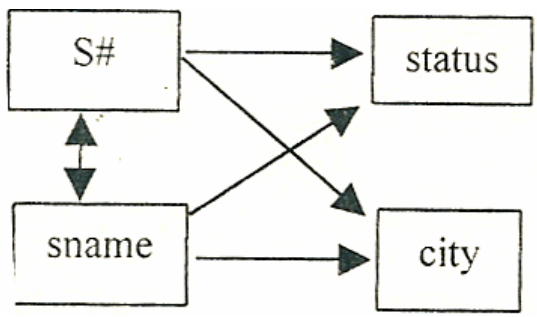
نکته: رابطه ی که 3NF نباشد، BCNF نیز نخواهد بود. هر رابطه BCNF حتما 3NF هم هست ولی هر رابطه 3NF الزاماً BCNF نیست.

مثال۱۱: رابطه SP(S#,P#,Qty) که کلید (S#,P#) را دارد در سطح BCNF است چرا که صفت Qty فقط به کلید (S#,P#) وابستگی دارد.

مثال۱۲: رابطه First(S#,status,city,P#,Qty) در سطح BCNF نیست چرا که سه دترمینان S# و city(S#,P#) دارد که دوتای آنها کلید نیست (یعنی city,S#)



مثال ۱۳: رابطه S(S#,Sname,status ,city) با نمودار وابستگی زیر در سطح BCNF است.



نکته: هر رابطه تمام کلید و هر رابطه باینری (جدولی که فقط دو ستون دارد)حتما BCNF هستند.

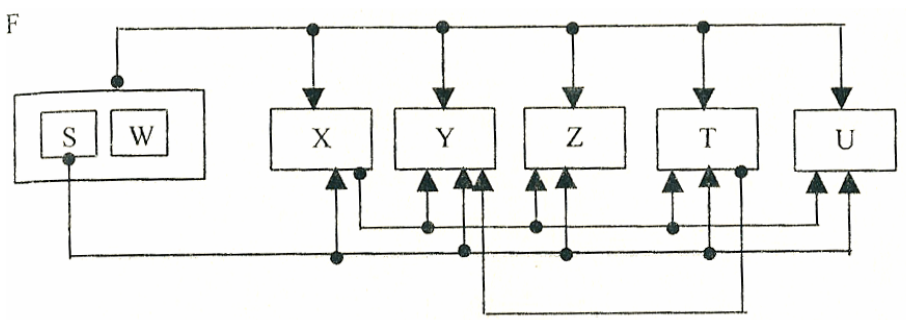
مثال۱۴:رابطه R={X,Y,Z,S,T,U,W} را با FDهای زیر در نظر بگیرید:

F=$\{S \to X, T \to Y, X \to Y, XY \to TUZ\}$

الف)کلیدهای کاندید را بیابید، ب) رابطه را به طور کامل نرمال سازی کنید.

$$S \to X, X \to Y \Rightarrow S \to Y \Rightarrow S \to XY$$
$$S \to XY, XY \to TUZ \Rightarrow S \to TUZ \Rightarrow S \to T, S \to U, S \to Z$$
$$X \to Y \Rightarrow X \to XY, XY \to TUZ \Rightarrow X \to TUZ$$
$$X \to TUZ \Rightarrow X \to T, X \to U, X \to Z$$
$$F_{OPT} = \{S \to X, S \to Y, S \to Z, S \to T, S \to U, T \to Y, X \to Y, X \to T, X \to U, X \to Z\}$$

s تمام صفات دیگر را به جز w می دهد پس(S,W) کلید کاندید است. رابطه F با نمودار وابستگی زیر در سطح 1NF است.



$$2NF\begin{cases}(S,W)\\(S,X,Y,Z,T,U)\end{cases}$$

$$3NF\begin{cases}(S,W)\\(S,X)\\(X,Z,T,U)\\(T,Y)\end{cases}$$

توجه کنید که نیازی به BCNF نیست زیرا در هر رابطه بیش از یک کلید کاندید نداریم. روابط 3NF فوق در سطح BCNF نیز می‌باشند.

تذکر: برای نرمال سازی حتما باید مجموعه وابستگی پوششی بهینه را بدست آوریم چرا که اگر بعضی از وابستگی های آن را در نظر نگیریم ممکن است به نتیجه اشتباهی برسیم.

نکته۱: رابطه 3NFی که تنها یک کلید کاندید دارد، BCNF است.

نکته۲: در حالت نبود صفت مشترک بین کلیدهای کاندید، اگر رابطه 3NF باشد، BCNF هم هست.

نکته۳: اگر رابطه ای به دو رابطه مستقل(طبق ضوابط ریسانن) تجزیه شود، معنایش این نیست که حتما باید تجزیه شود. تنها در صورتی تجزیه می‌کنیم که رابطه های حاصل از تجزیه نرمال‌تر شوند.

مواردي كه رابطه را نرمالتر نمي‌كنيم

در عمل رابطه ها را باید BCNF کرد، مگر اینکه با این کار وابستگیها حذف شوند. در این صورت همان 3NF کفایت می‌کند. یعنی ممکن است رابطه ای که BCNF نیست را جهت BCNF شدن تجزیه کنیم ولی با این عمل تجزیه بعضی از قواعد FD ها از بین بروند. پس چنین کاری را نباید انجام داد.

مثال زیر این مورد را نشان می‌دهد.

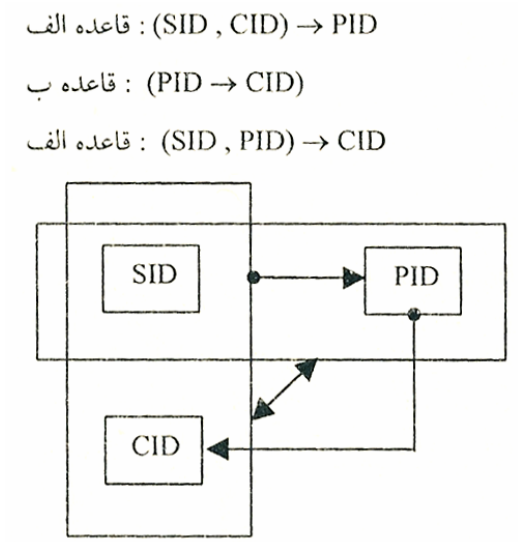
مثال ۱۵: رابطه R(SID,CID,PID) یعنی(شماره استاد، شماره درس، شماره دانشجو) R را در نظر بگیرید . قواعد سمانتیک زیر در این رابطه وجود دارد: الف) یک دانشجو، یک درس را با یک استاد انتخاب می کند. ب) یک استاد یک درس را تدریس می کند . ج)یک دروس توسط چند استاد تدریس می شود.

قاعده الف: $(SID, CID) \rightarrow PID$

قاعده ب: $(PID \rightarrow CID)$

قاعده الف: $(SID, PID) \rightarrow CID$

پس نمودار وابستگی به شکل روبرو است:



با توجه به شکل مشخص است که رابطه BCNF نیست زیرا کـه PID دترمینـان اسـت ولـی کلید کاندید نیست.

این رابطه را می توان به دو رابطه BCNF زیر تجزیه کرد: (منظور از C.K کلید کاندید است)

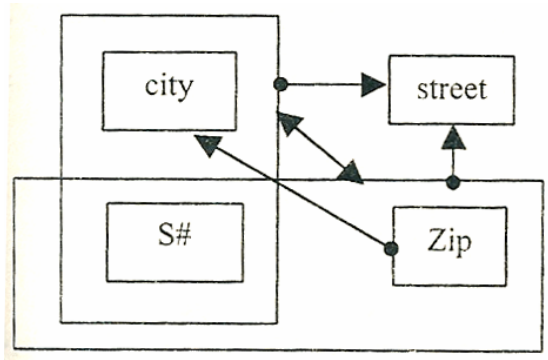
R1 (SID ,PID) R2 (PID,CID)

C.K C.K

اما R2,R1 مستقل از هم نیستند چرا که وابستگی تابعی $(SID,CID) \rightarrow PID$ از بین می‌رود و این تجزیه حافظ وابستگیها نیست. از این مثال این نتیجه مهم بدست می‌آید که گاه تجزیه یک رابطه 3NF به دو رابطه BCNF ضوابط ریسانن را ندارد.به بیان دیگر در BCNFکردن یـک رابطه خطر از دست رفتن وابستگیها وجود دارد . لذا در عمل رابطه ها را باید BCNFکرد مگر آنکه با این کار وابستگیهایی حذف شوند که در این صورت همان 3NFکفایت می کند .

تذکر : پرس و جو نیز می تواند در طراحی جداول نقش داشته باشد . اگر طراح تشخیص دهد که تجزیه یک جدول ، هر چند افزونگی هم داشته باشد باعث پایین امدن سرعت اکثر پرس و جو ها می شود ، مجاز است از نرمالتر سازی آن صرف نظر کند .

مثال ۱۶ : جدول آدرس دانشجو که در آن S# (شماره دانشجو) ، city (شهر) ، street (خیابان)، zip(کد پستی) می باشد را در نظر بگیرید . فرض کنید (S#، city) و همچنین (S#، zip) کلیدهای کاندید این رابطه هستند و در این رابطه وابستگی $Zip \rightarrow City$ برقرار است. این جدول در فرم 3NF است ولی در فرم BCNF نیست. چرا که $Zip \rightarrow City$ را داریم در حالیکه Zip کلید کاندید نیست.



با این وجود نمی‌توان با اطمینان این جدول را تجزیه کرد زیرا کد پستی ، بخش جدایی ناپذیر آدرس است و جدا کردن آن باعث پیچیده شدن پرس و جوهای مربوط به آدرس می‌گردد.

تذکر: همین مطلب هم برای روابط 2NF نیز صادق است.

مثال ۱۷: آدرس یک دانشجو به صورت زیر است:

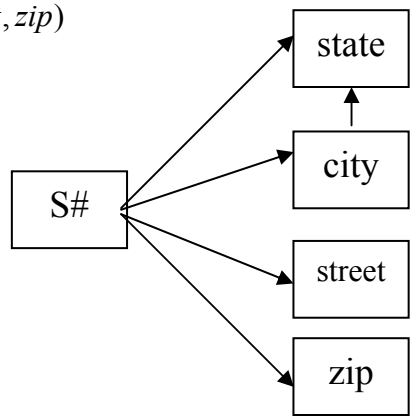
ADDR(S# ,state ,city,street ,zip)

(کد پستی، خیابان، شهر، استان، شماره دانشجو)آدرس

از آنجا که $City \rightarrow state$ پس نمودار وابستگی به صورت روبرو است:

همانطور که می‌دانید این رابطه 2NF بوده ولی 3NF نیست. اگر رابه را به دو رابطه زیر تجزیه کنیم:

$$\begin{cases} ADDr1(S\#, city, street, zip) \\ ADDR2(city, state) \end{cases}$$



جداولی به سطح 3NF خواهند رسید ولی از نظر معنایی و کاربردی جدول ADRR2 جالب نمی‌باشد. چرا که این موضوع که«یک شهر جزو کدام استان است» در چنین محیطی به عنوان یک اطلاع مطرح نمی‌باشد.بنابراین در این مثال بهتر است جدول ADDR را در همان سطح 2NF نگه داریم.

نکته: از مثالهای فوق نتیجه می‌گیریم که ممکن است در یک طراحی بهتر باشد که جداول تا حد امکان نرمال سازی نشوند.

رابطه 4NF

رابطه‌ای است در سطح BCNF باشئ ولی باز هم آنومالی داشته باشد. ابتدا این موضوع را با یک مثال نشان می‌دهیم.

مثال ۱۸: در یک دانشگاه درس C توسط استاد T از روی کتاب X تدریس می شود. یک درس توسط هر یک از استادان یک مجموعه استاد، از روی هر یک از کتابهای یک مجموعه کتاب، تدریس می‌شود.

جدول CTX

کتاب درسی(X)	نام استاد(T)	نام درس(C)
فیزیک دانشگاهی	اکبری	فیزیک حرارت
فیزیک هالیدی	جوادی	
فیزیک دانشگاهی		
مبانی فیزیک	اکبری	فیزیک مکانیک
اصول فیزیک		

بسط

کتاب درسی	نام استاد	نام درس
فیزیک	اکبری	فیزیک
دانشگاهی	اکبری	حرارت
فیزیک	جوادی	فیزیک
هالیدی	جوادی	حرارت
فیزیک	اکبری	فیزیک
دانشگاهی	اکبری	حرارت

رابطه CTX فوق یک رابطه تمام کلید است و بنابراین در سطح BCNF قرار دارد. (رابطه تمام کلید حتما BCNF است چرا که دترمینان، تنها کلید کاندید می‌باشد). ولی بدیهی است که CTX افزونگی دارد و منجر به آنومالی می گردد. به عنوان مثال برای اضافه کردن این اطلاعات که «درس فیزیک حرارت توسط سلطانی (استاد جدید) تدریس می‌شود»لازم است دو تاپل زیر در جدول درج گردد:

فیزیک دانشگاهی	سلطانی	فیزیک حرارت
فیزیک هالیدی	سلطانی	فیزیک حرارت

این مشکل از آنجا ناشی شد که اساتید و کتب درسی کاملاً مستقل از یکدیگرند. اگر جدول CTX را به دو جدول CX,CT به صورت زیر تجزیه کنیم افزونگی کاهش می‌یابد:

جدول CX

نام کتاب	نام درس
فیزیک	فیزیک حرارت
دانشگاهی	فیزیک حرارت
فیزیک هالیدی	فیزیک مکانیک
فیزیک	فیزیک مکانیک
دانشگاهی	فیزیک مکانیک

جدول CT

نام استاد	نام درس
اکبری	فیزیک حرارت
جوادی	فیزیک حرارت
اکبری	فیزیک مکانیک

حال برای اضافه کردن اطلاعات «درس فیزیک حرارت توسط سلطانی ارائه می‌گردد» فقط یک تاپل باید به رابطه CT اضافه شود. همچنین توجه داشته باشید که رابطه CTX را می‌توان با الحاق CT و CX بدوننقصان ، بدست آورد. بدیهی است که طراحی CTX اشکال دارد و تجزیه آن به CX,CT مناسب است.

CTX در سطح نرمال BCNF است و هیچ وابستگی تابعی بین اجزاء کلید وجود ندارد. پس ایده های مطرح شده تا نرمال‌سازیBCNF که بر اساس مفهوم «وابستگی تابعی» بود برای این مثال کاربرد ندارد.

پس از آنکه فاگین مفهوم وابستگی چند مقداری (Multi valued Dependency=MVD) را معرفی کرد مشکل فوق برطرف شد. MVD در واقع عمومیت بخشیدن به وابستگی‌های تابعی (FDها) است یعنی هر FD یک MVD است ولی عکس آن درست نیست یعنی MVD هایی وجود دارند که FD نیستند. ابتدا وابستگی MVD را به صورت ساده تعریف می‌کنیم.

تعریف اول : در رابطه R(A,B,C) با صفات ساده یا مرکب A و B و C ، صفت B با A وابستگی تابعی چند مقداری دارد و به صورت $A \rightarrow\rightarrow B$ نمایش می‌دهیم، اگر به یک مقدار A، مجموعه‌ای از مقادیر B متناظر باشد.

با توجه به تعریف فوق درمی یابیم که وابستگی تابعی B به A حالت خاصی از وابستگی تابعی چند مقداری B به A است که در آن، مجموعه مقادیر B متناظر با یک مقدار A ، یک عنصر دارد.

مثال ۱۹: در مثال قبلی در رابطه CTX داریم:

$$C \rightarrow\rightarrow T \quad , \quad C \rightarrow\rightarrow X$$

وابستگی چند مقداری $A \rightarrow\rightarrow B$ به این صورت خوانده می‌شود:«A می‌تواند به طور چندگانه B را تعیین کند»یا«B به A وابستگی چندگانه دارد».

از نظر شهودی معنای $C \rightarrow\rightarrow T$ در مثال قبلی آن است که، اگر چه یک درس فقط یک استاد ندارد(یعنی وابستگی تابعی $C \rightarrow T$ وجود ندارد) هر درس دارای یک مجموعه خوش تعریف از اساتید متناظر است. منظور از خوش تعریف این است که برای درس خاصی مانند c و کتاب درسی خاصی مانند x مجموعه‌ای از اساتید به نام t که با جفت (c,x) در CTX مطابقت می‌کند، فقط به مقدارc بستگی دارد.

تعریف دوم: در رابطه R(A,B,C) صفت B با A وابستگی تابعی چند مقداری دارد، اگر و فقط اگر مجموعه مقادیر B متناظر با یک مقدار از جفت (A,Z) در R فقط به مقدار A بستگی داشته باشد و وابسته به مقدار Z نباشد. به عبارت دیگر مجموعه مقادیر B فقط با

تغییرات A تغییر کند. یعنی تا زمانی که مقدار A در یک مقدار مشخص از جفت (A,Z) تغییر نکند، مجموعه B متناظر با آن نیز تغییر نکند.

نکته ۱: در رابطه R{A,B,C} وابستگی چندگانه $A \to\to B$ برقرار است اگر و فقط اگر وابستگی چند مقداری $A \to\to B$ نیز برقرار باشد. MVDها همیشه به صورت جفتهایی به این شکل با هم هستند، لذا می توان آنها را در یک جمله به صورت زیر نوشت:

$C \mid A \to\to B$

مثال $X \mid C \to T$

نکته۲: اگر در رابطه R، صفت B زیر مجموعه صفت A باشد در اینصورت $A \to\to B$ بدیهی (نامهم) است. اگر B مجموعه تهی باشد نیز این عبارت برقرار است یعنی $A \to\to \phi$

حال می‌توان گفت علت آنومالی رابطه CTX در مثال ۱۸، وجود وابستگی‌های MVD می باشد. بنابراین رابطه را باید چنان تجزیه کنیم که در روابط حاصله دیگر MVD وجود نداشته باشد. به بیانی کاربردی :رابطه‌ای 4NF است اگر BCNF بوده و در آن وابستگی‌های MVD وجود نداشته باشد. مثلاً جداول CX,CT در فرم 4NF می‌باشند. بعضی کتابها به بیانی دیگر 4NF را تعریف کرده‌اند:

تعریف 4NF : رابطه R در 4NF است اگر و فقط اگر ، هر گاه بین دو زیر مجموعه از صفات R ، مثلا A و B وابستگی چند مقداری (غیر بدیهی) $A \to\to B$ برقرار باشد، آنگاه تمام صفات R با A وابستگی تابعی داشته باشند.

تعریف دیگر 4NF : رابطه R در 4NF است در صورتی که اگر وابستگی چند مقداری غیر بدیهی $A \to\to B$ در R وجود داشته باشد آنگاه A ابر کلید R باشد.

جداولی که دارای وابستگی چند مقداری هستند را می‌توان مشابه BCNF به صورت زیر تجزیه کرد:

جدول اول: شامل ستونهای A و B

جدول دوم : شامل ستونهای A و سایر ستونهای R

قضیه فاگین(FAGIN): رابطه R(A,B,C) به صورت R1(A,B) و R2(A,C) تجزیه می‌شود اگر و فقط اگر : $A \to\to B$ (که طبعا $A \to\to C$ نیز برقرار است)

نکته۱: قضیه هیث حالت خاصی از قضیه فاگین است برای یادآوری:

قضیه هیث(HEATH):اگر در رابطه R(A,B,C) وابستگی $A \to B$ برقرار باشد، آنگاه می‌توان R را به دو رابطه R1(A,B) و R2(A,C) تجزیه کرد و این تجزیه ، خوب است.

نکته ۲: بنابر نظر ریسانن اگر در رابطه R(A,B,C) داشته باشیم $A \to\to B$ و $B \to\to C$ آنگاه R به صورت زیر تجزیه می‌شود و این تجزیه خوب است:

R1(A,B) , R2(B,C)

قبلاً بیان شده بود که بر اساس ضوابط ریسانن، اگر در رابطه R(A,B,C) وابستگی‌های $A \to B$ و $B \to C$ برقرار باشد، در این صورت بهتر است تجزیه به صورت R1(A,B) , R2(B,C) انجام گیرد.

نکته ۳: 4NF فرم BCNF را ارائه می‌کند.

نکته ۴: فاگین نشان داد 4NF همواره دستیافتنی است. یعنی هر رابطه را با تجزیه بدون نقصان، می توان به روابط 4NF تبدیل کرد.

تعدادی از خواص وابستگی تابعی مقداری به صورت زیر است: R(H) مجموعه عناوین صفات خاصه می‌باشد)

۱-اگر $A \subset B$ آنگاه $A \to\to B$

۲-اگر $A \to\to B$ آنگاه $AC \to\to BC$

۳-اگر $A \to\to B$. $\mathbf{B} \to\to C$ آنگاه $A \to\to C$

۴-اگر $A \to\to B$ و $B \to\to C$ آنگاه $A \to\to C - B$

۵-اگر $A \to\to B$ آنگاه $A \to\to R(H) - A - B$

۶-اگر $A \to B$ آنگاه $A \to\to B$

۷-اگر $A \to\to B$ و $A \to\to C$ آنگاه $A \to\to BC$

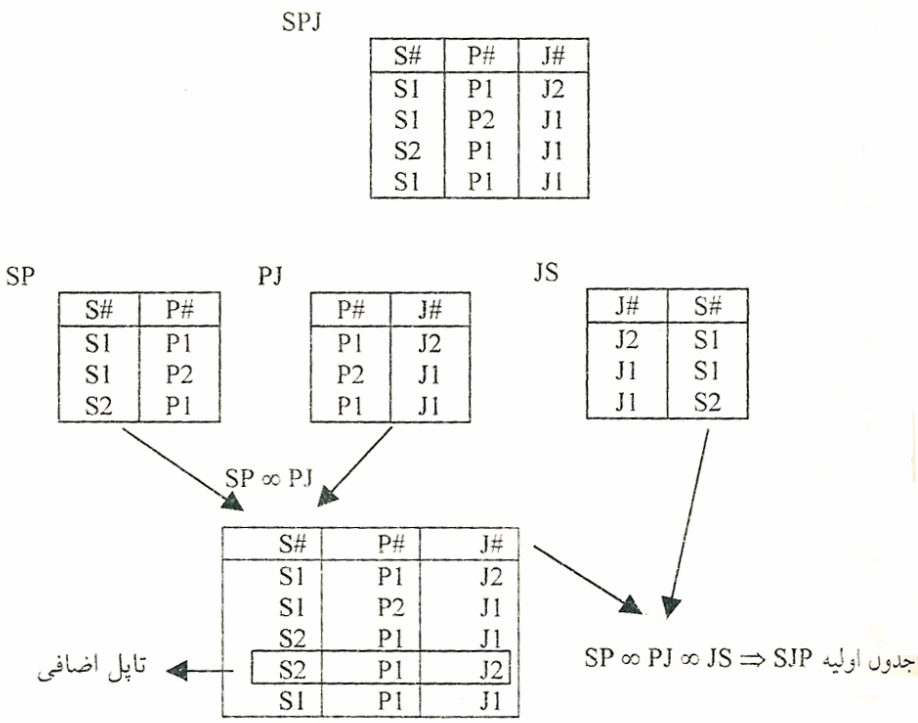
۸-اگر $A \to\to B$ و $A \to\to C$ آنگاه $A \to\to B \cap C$

۹-اگر $A \to\to B$ و $A \to\to C$ آنگاه $A \to\to (B - C)$ و $A \to\to (C - B)$

رابطه 5NF (PJNF)

در عملیات نرمال سازی تا 4NF ، روش عمومی عبارت بود از تجزیه رابطه به دو رابطه نرمالتر به صورتی که با پیوند دو رابطه همان رابطه اصلی و اولیه بدست آید. اما رابطه‌هایی وجود دارند که نمی‌توانند بدون نقصان به دو تصویر تجزیه شوند اما می‌توانند بدون نقصان به سه یا بیشتر از سه تصویر تجزیه گردند. چنین رابطه‌هایی را «تجزیه n گانه» می‌نامند(n>2) .

مثال ۲۰: رابطه SPJ«تمام کلید» است و شامل FD, MVD نیست و بدین دلیل به شکل 4NF می‌باشد . سه تصویر دودویی JS,PJ,SP در شکل نشان داده شده است:



با پیوند دو رابطه SP و PJ اطلاع حشو یا ساختگی پدید آمد. تا اینجا نتیجه می‌گیریم که رابطه SPJ تجزیه شدنی به دو رابطه نیست. به عبارتی دیگر با پیوند دوپرتوش، خود رابطه بدست نمی آید، ولی اگر سه پرتوش را با هم پیوند دهیم همان رابطه اولیه بدست می آید:

$$SPJ = SP \infty PJ \infty JS$$

در اینحال می‌گوئیم رابطه SPJ ، وابستگی پیوندی(Join Dependency=JD) به سه پرتوش دارد. به عبارتی دیگر رابطه اصلی SJP «تجزیه سه گانه» است.

توجه کنید این جمله که «SPJ برابر الحاق JS,PJ,SP است» معادل عبارت زیر می‌باشد:

«اگر جفت (S1,P1) در SP باشد و جفت (P1 ,J1) در PJ باشد و جفت (J1,S1) در JS باشد، آنگاه سه تایی (S1,P1,J1) در SPJ است» در واقع این نوعی محدودیت است که آنرا محدودیت 3D می‌نامیم . این محدودیت می‌گوید که اگر مثلا:

۱- فن آوران پیستون تولید می‌کند.

۲- پیستون در پروژه تولید پژو استفاده می‌شود.

۳- فن‌آوران در پروژه تولید پژو شرکت دارد.

آنگاه: فن آوران پیستون را برای پروژه تولید می‌کند.

همانطور که می‌دانید این نتیجه‌گیری در حالت کلی نادرست است و در فصل دوم به نام «تله الحاق» معرفی شد ولی در مثال فوق این نتیجه‌گیری درست است.

به ماهیت چرخه‌ای رابطه بالا توجه کنید«اگر S1 به P1 وصل باشد و P1 به J1 وصل باشد و J1 به S1 پیوند داشته باشد، آنگاه P1,S1 و J1 باید در یک تاپل وجود داشته باشند». یک رابطه به ازای $n>2$ تجزیه پذیر n گانه است اگر و فقط اگر محدودیت n طرفه را برآورده کند.

تعریف وابستگی پیوندی یا الحاقی (JD): رابطه R وابستگی پیوندی به n پرتوش را دارد، اگر و فقط اگر R حاصل پیوند n پرتوش (R1 ,R2,...Rn) باشد و نه کمتر . وابستگی پیوندی را به صورت $R=*\{R1,R2,....Rn\}$ یا $R=JD*(R1,R2,...Rn)$ نمایش می‌دهیم.

مثلا رابطه SPJ وابستگی پیوندی خود یعنی $*\{SP,PJ,JS\}$ را برآورده می کند.

تا اینجا دیدیم که رابطه SPJ با وابستگی پیوندی خود یعنی $*\{SP,PJ,JS\}$ می‌تواند تجزیه سه‌گانه شود. پرسش مهم آن است که آیا باید چنین تجزیه‌ای انجام گیرد. پاسخ در اینجا «احتمالاً بله » است.

رابطه SPJ (با وابستگی پیوندی‌ای که دارد) دارای آنومالی در اعمال به هنگام سازی است که این مشکلات با تجزیه سه‌گانه از بین می‌روند.

مثال۲۱: برای درج (S2,P1,J1) در جدول زیر باید تاپل (S1 ,P1,J1) نیز درج شود.

SPJ

S#	P#	J#
S1	P1	J1
S2	P2	J2

یا مثلاً برای حذف (S1,P1,J1) در جدول زیر باید تاپل (S2,P1,J1) نیز حذف گردد.

SPJ

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

پس می‌بینیم که جدول SPJ با آنکه در سطح 4NF است ولی آنومالی دارد. حال سئوال مهمی مطرح می‌شود: آیا وابستگی پیوندی یک رابطه به سه پرتوش(یعنی صرف وجود 3D) دلیل وجود آنومالی در رابطه 4NF و دلیل 5NF نبودن آن است؟ جواب این سؤال خیر است. برای روشن شدن مثالی را مطرح می‌کنیم.

مثال۲۲: رابطه S(S# ,sname ,status ,city) را در نظر گرفته و فرض می‌کنیم Sname S#, کلیدهای کاندید رابطه S هستند. رابطه S در سطح 4NF است. رابطه S چندین وابستگی الحاقی را برآورده می سازد مانند :

*{ {S# ,Sname ,status } }

یعنی S بدون نقصان می‌تواند به دو جدول S2(S# ,city) و S1(S#,sname ,status) تجزیه شود. به طور مشابه رابطه S وابستگی الحاقی زیر را برآورده می‌کند:

* { {S#,sname} ,{S#,status} , {Sname ,city} }

با وجود این وابستگی‌های پیوندی ، این رابطه آنومالی ندارد و رابطه های حاصل از تجزیه ، نرمالتر از S نمی‌باشند. با توجه به این مثال این ؤال مطرح می‌شود که چرا وجود 3D رد رابطه S دلیل وجود آنومالی در این رابطه نیست ولی وجود 3D در رابطه SPJ دلیل آنومالی است؟ با اندکی توجه درمی‌یابیم که در رابطه S تمام وابستگیهای پیوندی ناشی از کلید کاندید رابطه است در حالیکه در SPJ اینچنین نیست. به عبارت دیگر در همه پرتوهای رابطه S کلید کاندید حضور دارد ولی در SPJ اینگونه نیست پس به تعریف 5NF می‌رسیم.

تعریف۵NF: رابطه ای 5NF است اگر تمام وابستگیهای پیوندی آن ناشی از کلیدهای کاندید آن باشد. به عبارت دیگر اگر بتوانیم یک وابستگی پیوندی در R پیدا کنیم که در همه پرتوهایش کلید کاندید R وجود نداشته باشند، آنگاه رابطه 5NF نیست.

نکته ۱: اگر چه می توان FDها و MVD ها را به راحتی شناسایی کرد در مورد JD ها به سادگی امکان‌پذیر نیست زیرا معنای شهودی JD ها ممکن است واضح نباشد . بنابراین الگوریتم تعیین اینکه یک رابطه که به شکل 4NF است ولی به شکل 5NF نیست هنوز به وضوح بیان نشده است.

نکته ۲: 5NF آخرین شکل نرمال است یعنی وقتی یک رابطه به شکل 5NF باشد، تضمین می‌شود که عاری از بی‌نظمی است که می تواند با عمل پرتو، حذف شود. به عنوان مثال رابطه S به شکل 5VF است. و آن را می‌توان با روشهای مختلف بدون نقصان تجزیه کرد، اما هر

تصویر در چنین تجزیه‌ای حاوی یکی از کلیدهای کاندید خواهد بود و لـذا بـه نظـر نمـی‌رسـد تجزیه بیشتر امتیاز خاصی داشته باشد.

نکته۳ : همانطوز که MVD ها حالت کلی FD ها بودند، JD ها نیز شـکل کلـی MVD هـا هستند. به عبارتی اگر در رابطه R(H) وابستگی چند مقداری $A \to\to B$ وجود داشـته باشـد این وابستگی را می توان به صورت وابستگی پیوندی JD*(R1 ,R2) نیز بیان کرد کـه در آن R1(A,B) و R2(A,H-B) دو پرتو از R هستند.

نکته ۴: اگر رابطه‌ای 3NF باشد و تمام کلیدهای کاندید آن صفات ساده باشـند، در اینـصورت رابطه 5NF است.

نکته۵:اگر رابطه BCNF باشد و حداقل یکی از کلیدهای کاندید آن صفت ساده باشـد، چنـین رابطه‌ای 4NF است، اما لزوماً۵NF نمی‌باشد.

خلاصه مراحل نرمال سازي

۱-پرتوهایی را از رابطه اصلی 1NF ایجاد کنید تا FD هایی که کاهش ناپـذیر هـستند حـذف شوند. در این مرحله روابط 2NF تولید می گردد.

۲-پرتوهایی را از رابطه 2NF ایجاد کنید تا FD های تعدی حذف شوند. در این مرحله روابط 3NF تولید می‌گردد.

۳-پرتوهایی را از رابطه 3NF ایجاد کنید تا FD های باقی‌مانده ای که در آنها، دترمینان کلید کاندید نیست، حذف شوند. در این مرحله روابط BCNF تولید می‌گردد.

۴- پرتوهایی را روی رابطه BCNF ایجاد کنید تا MVD هایی که FD نیستند حذف شـوند. در این مرحله روابط 4NF تولید می‌گردد.

۵-پرتوهایی را روی رابطه 4NF ایجاد کنید تا JD هایی که توسط کلیدهای کاندید مطرح نمی‌شوند حذف گردند. در این مرحله روابط 5NF تولید می گردد.

نکته: تشابه جالبی بین تعاریف 5NF,4NF ,BCNF به صورت زیر وجود دارد:

رابطه R به شکل BCNF است اگر و فقط اگر هر FD در R توسط کلیدهای کاندید R مطرح شود.

رابطه R به شکل 4NF است اگر و فقط اگر هر MVD در R توسط کلیدهای کاندید R مطرح شود.

رابطه R به شکل 5NF است اگر و فقط اگر هر JD در R توسط کلیدهای کاندید R مطرح شود.

فرآیند ایجاد پرتو در هر مرحله باید به صورت بدون نقصان باشد و وابستگی‌ها را نیز حفظ کند. به عبارت دیگر در تجزیه جداول نباید اطلاعات و وابستگی‌های اصلی از بین برود. به عبارت دیگر هنگام نرمال سازی باید دو مورد زیر رعایت شده باشد.

۱-تجزیه یک جدول باید«پیوند پذیر »باشد یعنی به ازای تمام جدول مربوطه ، از پیوند طبیعی آنها دقیقا جدول اصلی بدست بیاید. تجزیهR به دو جدول R1,R2 در صورتی پیوند پذیر است که کلید حداقل یکی از دو جدول ، ستونهای مشترک آن دو باشد.

۲-تجزیه یک رابطه به روابط کوچکتر باید«حافظ وابستگی »باشد یعنی تمام وابستگی های اصلی حفظ شود.

اگر رابطه ای را به صورت {R,F} نمایش دهیم که R رابطه و F مجموعه وابستگی‌ها باشد، آنگاه تجزیه {R,F} به { {R1,F1} ,{R2,F2} , ...{Rn,Fn} } در صورتی حافظ وابستگی است که: $F^+ = \{F1 \cup F2 ... \cup Fn\}^+$

مزایا ، معایب و اهداف نرمال سازي

مزایای نرمال سازی عبارتند از :

۱-کاهش بعضی از آنومالیها

۲-ساده کردن اعمال بعضی از قواعد جامعیت

۳- کاهش بعضی از افزونگی‌ها

۴- ارائه یک طراحی بهتر و واضح تر با کمترین اختلاط اطلاعات

نکته : در بانک اطلاعاتی رابطه‌ای سه نوع افزونگی می‌تواند وجود داشته باشد :

۱- افزونگی طبیعی

۲- افزونگی تکنیکی (ناشی از وجود مثلاً کلید خارجی)

۳- افزونگی ناشی از طراحی بد و اختلاط و اطلاعات . نرمال‌سازی این افزونگی سوم را تا حد زیادی کاهش می‌دهد .

معایب نرمال سازی عبارتند از:

۱- سربارگذاری بر روی سیستم چرا که در صورت نیاز باید پرتوها را پیوند داد .

۲- ایجاد نوعی افزونگی . اگر قرار باشد تجزیهیک رابه به دو رابطه نرمالتر؛تجزیه ای خوب و بدون اضافات زائد باشد (بدون حشو) باید صفت مشترک در دو رابطه حداقل کلید کاندید یکی از آن دو باشد .

در اینحال در رابطه دیگر کلید خارجی خود سبب افزونگی است .

۳-فرایند نرمال‌سازی در محیط‌های بزرگ که تعداد رابطه‌ها و صفات خاصه زیاد است ،کاری زمانگیر بوده و یافتن همه وابستگی‌ها زمان زیادی می‌خواهد.

۴-چون مجموعه کاهش پذیر وابستگیهای تابعی یک رابطه ، یکتا نیست، لذا روشهای مختلفی در طراحی وجود خواهد داشت و بدین ترتیب مشکل تصمیم‌گیری برای طراح پیش می آید.

اهداف کلی نرمال سازی عبارتند از :

۱-حذف بعضی از انواع افزونگی‌ها

۲-پرهیز از آنومالی‌های به هنگام سازی

۳-ایجاد یک طراحی که نمایش خوبی از دنیای واقعی باشد، یعنی درک شهودی آن ساده بوده و مبنای خوبی برای رشد آینده باشد.

۴-سهولت در اعمال بعضی از محدودیتهای جامعیتی

نکته : نرمال‌سازی یعنی تجزیه یک رابطه به تعدادی رابطه دیگر که از نظر منطقی جدا از هم هستند و نه الزاما به صورت فایلهای ذخیره شده جدا از هم.