بسمه تعالی
فصل اول

DBMS

نرم افزار مدیریت بانک اطلاعاتی یا DBMS(Data Base Manegemant System)رابط بین برنامه های کاربردی و داده هاست و هر گونه دستیابی به داده ها از طریق DBMSصورت می گیرد. مهمترین خصیصه استفاده از DBMSمستقل شدن برنامه های کاربردی (Application program) از جنبه ها و خصوصیات محیط فیزیکی ذخیره سازی است که اصطلاحاً به آن استقلال داده یی فیزیکی (Physical Data Independence) می گویند.

ویژگی های DBMS

1- برنامه های کاربردی از داده های محیط فیزیکی کاملاً مستقل می شوند.
2- امکان کنترل متمرکز روی تمامی داده های عملیاتی وجود دارد.
3 - نرم افزار پیچیده و جامع موسوم به سیستم مدیریت بانک اطلاعاتی (DBMS) واسط بین برنامه های کاربران و محیط داخلی و فیزیکی ذخیره سازی است.
4- سرعت دستیابی به داده ها بالا می باشد.
5- در DBMS از مفهوم چند سطحی بودن ساختار داده یی و معماری چند سطحی ذخیره سازی استفاده می شود.
6- رشد پذیری یکی دیگر از ویژگیهای این سیستم است.

برای راهبری این نرم افزار 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 تصمیم گیری وطراحی موارد مذکور را بر عهده دارد.

ب)برنامه ساز بانک اطلاعت (Data Base Programmer=DBP) که تصمیمات مدیر را پیاده سازی می کند.

تعاریف اولیه

موجودیت:(پدیده،نهاد یاEntily) پدیده ،شی یا فردی که در مورد آن می خواهیم اطلاعات داشته باشیم.

صفت خاصه(Attribute) ویژگی جدا ساز یک نوع موجودیت از نوع دیگر است.

مثال:موجودیت دانشجو می تواند دارای صفات خاصه:نام- نام خانوادگی- سال تولد- معدل ترم پیش باشد و مقادیر این صفات خاصه برای یک دانشجوی خاص برابر است با:

علی – اکبر – 1352 – 16

به صفات خاصه گاهی اوقات خواص (properties) نیز گفته می شود.

اطلاع: دو مولفه الف)اسم صفت خاصه ب) مقدار صفت خاصه ،برای هر صفت خاصه را می گویند.

ارتباط(Relation ship) : به ارتباط بین موجودیت ها در یک محیط عملیاتی گفته می شود. مثل ارتباط بین دانشجویان و اساتید در محیط عملیاتی دانشگاه(مثلاً دانشجویی A با چه ساتیدی در این ترم درس گرفته است)

فیلد : کوچکترین واحد داده ذخیره شده می باشد.

رکورد : مجموعه ای از فیلد های مرتبط به هم می باشد.

فایل : مجموعه ای از تمام نمونه ها یا رویدادهای یک نوع رکورد.

سیستم فایل: به ساختار کلی نامگذاری،ذخیره سازی و سازماندهی فایلها در یک سیستم عامل، سیستم فایل گفته می شود.

داده(Data) :از نظر ساختاری ،داده عبارت است از مقادیر صفات خاصه انواع موجودیت ها و مادر این درس از این تعریف استفاده می کنیم.

اطلاع (Information) :از پردازش داده ها،اطلاعات حاصل می شود و یا داده پس از آنکه مورد تفسیر قرار گرفت تبدیل به اطلاع می شود.هنگامی که اسم صفت خاصه و مقدار منسوب به آن در دست باشد،اطلاعی در مورد موجودیت حاصل می شود.

داده های بانک اطلاعاتی ،داده های پایدارو با ثبات هستند.منظور از پایداری این است که نوع داده های بانک اطلاعاتی با داده های نا پایداری مثل داده های ورودی،داده های خروجی،دستورات کنترلی،صفها، بلوکهای کنترلی نرم افزار و نتایج میانی که ماهیت آنها گذرا است،تفاوت دارد.بانک اطلاعاتی مجموعه ای از داده های پایدار است که توسط سیستم های کاربردی موجود در مؤسسه ای مورد استفاده قرار می گیرد.

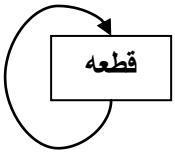
پس از انجام امکان سنجی ،بررسی نیازها و محدودیت و ... طراح بانک به کمک نمودارهائی ،شمای کلی بانک را مستقل از مدل بانک (جدولی ،شبکهای،سلسله مراتبی) و نیز مستقل از جنبه های برنامه نویسی ترسیم می کند.برای کار مدلهای مختلفی از جمله مدل دودویی (Binery Model)، Semantic data model ، مدل ER ،مدل ER گسترش یافته ،مدل شی گراء (OR) و غیرهوجود دارد.در این جا مدل ER و سپس EER را معروف تر از بقیه هستند شرح داده شده است.

نمودارER (Entity Relationship Diagram)

این نمودار نمایشگر ارتباط بین موجودیت های یک محیط عملیاتی است و به کمک آن داده های موجود مدل بندی می شوند.

ارتباط:فاکتورهای مهم در ارتباط:

1 - تعداد موجودیت های شرکت کننده در ارتباط
2 - مقدار نمونه های شرکت کننده در ارتباط
3 - اختیار شرکت در ارتباط



حالت 1 خود به سه دسته تقسیم می شود:

1 - یگانه : ارتباط بین یک موجودیت و خودش
2 - دو گانه : ارتباط بین دو موجودیت
3 - چند گانه : ارتباط بین بیش از دو موجودیت

حالت 2 نیز خود سه حالت دارد:

1 - یک به یک : یک نمونه از A با یک نمونه از B با یک نمونه از A ارتباط دارد.

2 - یک به چند :یک نمونه از A با چند نمونه B و یک نمونه A ارتباط دارد.

3 - چند به چند : یک نمونه از َ با چند نمونه B و یک نمونه B با چند نمونه A ارتباط دارد.

حالت 3 نیز دو دسته می شود:

1 - اختیاری : یک نمونه خاص می تواند در یک ارتباط شرکت نداشته باشد.
2 - اجباری :شرکت در ارتباط برای همه نمونه ها اجباری است.

Cordinality : تعداد دقیق در جه ارتباطات را در آن ذکر می کنیم که با استفاده از آن می توان اختیاری یا اجباری بودن را تشخیص داد.

نمودار EER

در سال 1976 چن (chen) از دانشگاه MIT مدل ER(Entity Relation) را جهت طراحی بانک پیشنهاد کرد. این مدل در طول زمان پیشرفت کرد و بنام Extended ER=EER معروف گردید.

انواع صفت

الف) صفت کلیدی :کلید عبارت است از یک یا چند صفت که در یک موجودیت منحصربه فرد باشد.مثلاً در موجودیت دانشجو شماره دانشجویی کلید اشت.

ب) صفت ساده و مرکب : بعضی از صفتها ساده هستند مثل شماره دانشجویی ، ولی صفت مرکب صفتی است که هم خودش معنی دار است و هم بخشهایی از آن .

ج)صفت تک مقداری یا چند مقداری: صفتی که دارای یک مقدار باشد تک مقداری و اگر برای یک صفت چندین مقدار داد شود چند مقداری گویند. مثل مدرک تحصیلی استاد.

د)صفت مشتق: صفت مشتق صفتی است که به کمک صفتهای دیگر می توان آن را محاسبه کرد. مثلاً سن استاد یک صفت مشتق است که با توجه به تاریخ تولد قابل محاسبه می باشد.

درجه ارتباط

درجه ارتباط برابر تعداد موجودیتهائی است که در آن ارتباط مشارکت دارند.

صفت در ارتباط

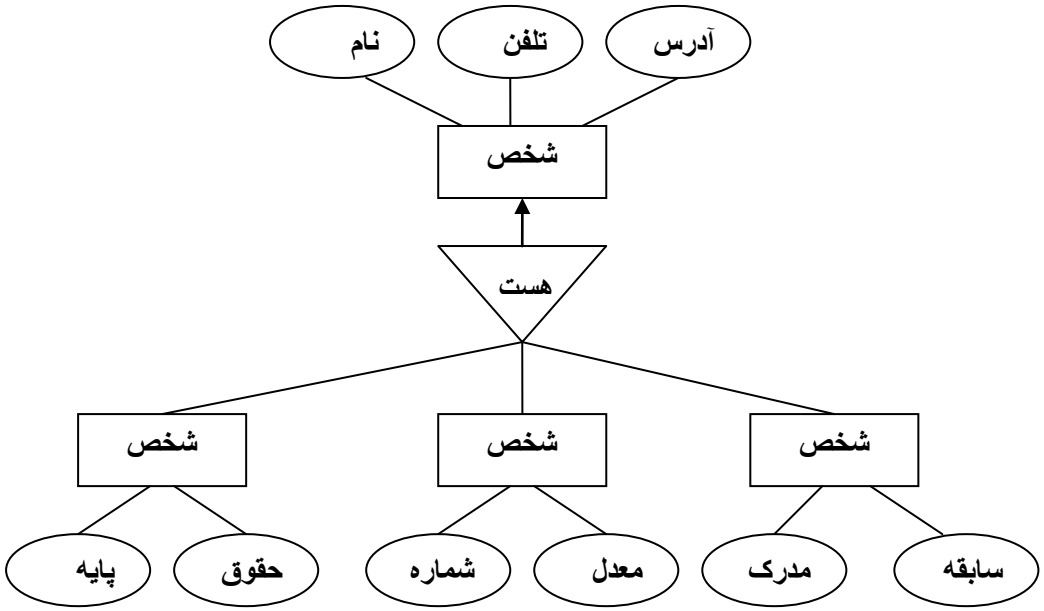
ارتباط ها نیز می توانند صفت داشته باشند. مثلاً صفت نمره در نمودار بانک اطلاعات دانشگاه می تواند صفت ارتباط باشند.

وابستگی وجودی(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 طورخودکار حذف شوند. مثلاً به محض حذف دانشجو از بانک دانشگاه (مثلاً بر اثر فارغ التحصیلی یا اخراج شدن) وابستگان او نیز (مثل همسر و فرزند) از سیستم کمک هزینه باید حذف شوند.

ارث بری (اشتراک صفت)

در مواردی که موجودیتها ،صفات مشترکی داشته باشند از تکرار آن برای موجودیتهای مختلف خودداری می کنیم و از ارث بری همانند شکل زیر استفاده می کنیم.



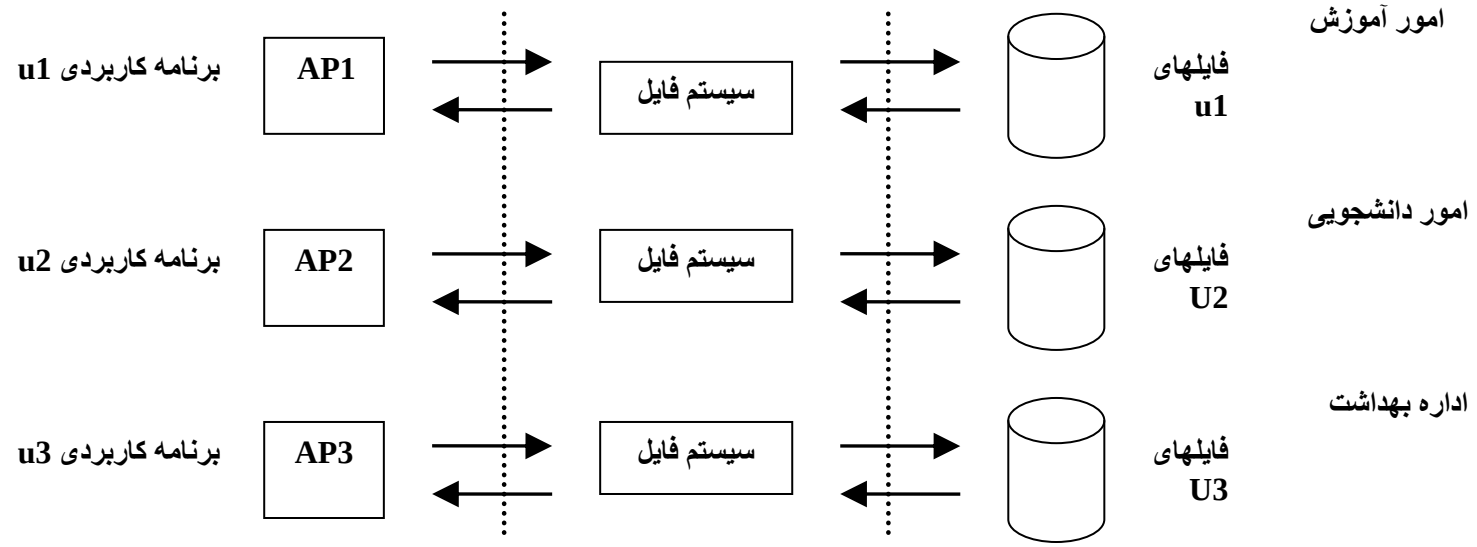
فصل دوم

پایگاه داده ها چیست؟

مجموعه ای است از داده ها ی 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.

برای ایجاد این سیستم دو روش کلی وجود دارد:

الف)روش کلاسیک:هر بخش از محیط عملیاتی ، توسط گروههای جداگانه بدون هماهنگی با یکدیگر ایجاد می کنند. در هر فایل مورد نظر با فیلدهای مورد نیاز تعریف می شود.



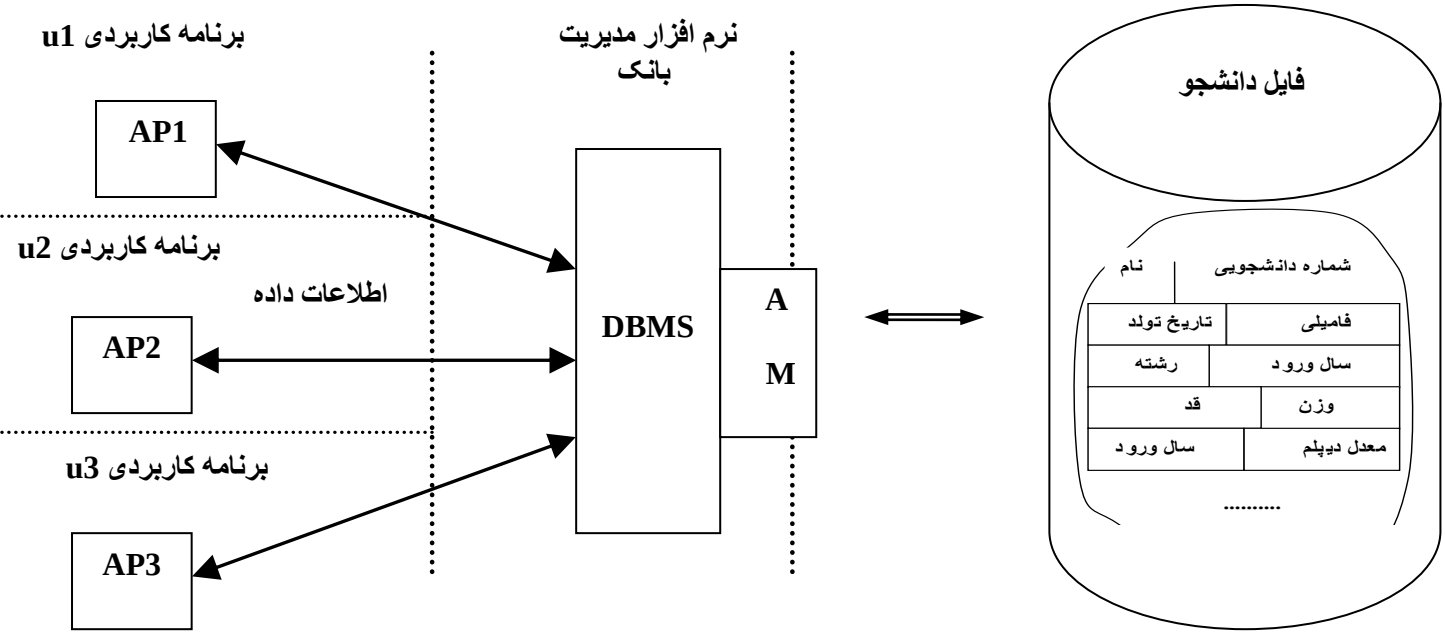
ویژگی ها:

1 - بروز پدیده افزونگی
2 - زبانهای برنامه سازی متعدد
3 - ناسازگاری
4 - عدم امکان اعمال استاندارد ها
5 - مصرف بیشتر حافظه
6 - وابستگی به محیط فیزیکی
7 - عدم وجود ارتباط بین برنامه های کاربردی
8 - توسعه بدون هماهنگی

ب)روش بانکی:کاربران مختلف هر یک طبق نیاز خود از آن به طور همزمان،مشترکاً استفاده می کنند.با آنکه در اینجا وحدت ذخیره سازی داریم ولی هر کاربری دید خاص خود را نسبت به داده ها دارد. و هر کاربر بدون ایجاد محدودیت برای کاربر دیگر در هر لحظه می تواند با بانک کار کند.

شکل زیر روش بانکی را نشان می دهد.

سطح فیزیکی پایگاه



ویژگی ها:

1 - نرم افزار پیچیده

2 - تولید و کار کردن با آن مشکل

3 - هزینه نگهداری آن زیاد است

DBMS :نرم افزار مدیریت بانک اطلاعاتی (DBMS=Data Base Manegemant System) رابط بین برنامه های کاربردی و داده هاست و هر گونه دستیابی به داده ها از طریق DBMS صورت می گیرد و به این دلیل امنیت داده ها نیز در این روش زیاد است.

عناصر اصلی محیط بانکی

محیط بانک اطلاعاتی از عناصر اصلی زیر تشکیل شده است:

1- سخت افزار 2- نرم افزار 3- کاربر 4- داده

1- سخت افزار محیط بانکی را می توان به صورت زیر تقسیم بندی کرد. الف) سخت افزار ذخیره سازی داده ها ب) سخت افزار پردازنده مرکزی ج) سخت افزار ارتباطی

2- نرم افزار محیط بانکی را می توان به دو دسته الف) نرم افزار کاربردی ب) نرم افزار سیستمی تقسیم کرد.

نرم افزار کاربردی : نرم افزاری است که کاربر باید برای تماس با سیستم بانک اطلاعاتی آماده کند. این نرم افزار به کمک یک زبان سطح بالا و یک زبان داده یی (Data Language) و برخی تسهیلات نرم افزاری برای تماس با بانک ساخته می شود.

نرم افزار سیستمی : که از نرم افزار سیستمی خاص بانک (یعنی DBMS) و نرم افزار سیستمی عمومی (یعنی سیستم عامل) تشکیل شده است. DBMS در یک تعریف مقدماتی ،سیستمی است که به کاربران امکان می دهد. عملیات مورد نظرشان را (مثل تعریف دادها –بازیابی داده ها و ذخیره سازی داده ها) انجام دهند .

DBMS که نرم افزار پیچیده است میهمان یک سیستم عامل است و از امکانات سیستم عامل در انجام وظایفش استفاده می کند.

3-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 ،پیاده سازی و نگهداری بانک را در محیط عملیاتی بر عهده دارد.کاربر دیگر بر اساس نوع کار با بانک اطلاعاتی الف)کاربران Batch(Ofline) کاربران لحظه ای (Online). براساس سطح دسترسی به بانک اطلاعاتی الف)کاربران

خارجی ب)کاربران داخلی. براساس مهارت کاربر : الف)کاربران بدون مهارت ب) کاربران برنامه نویس و ماهر

4- داده : منظور از داده در اینجا داده هایی است که در مورد موجودیت های مختلف محیط عملیاتی می خواهیم ذخیره کنیم و نیز ارتباط بین انواع موجودیتها و اصطلاحاً به آن «داده های عملیاتی» می گوئیم.

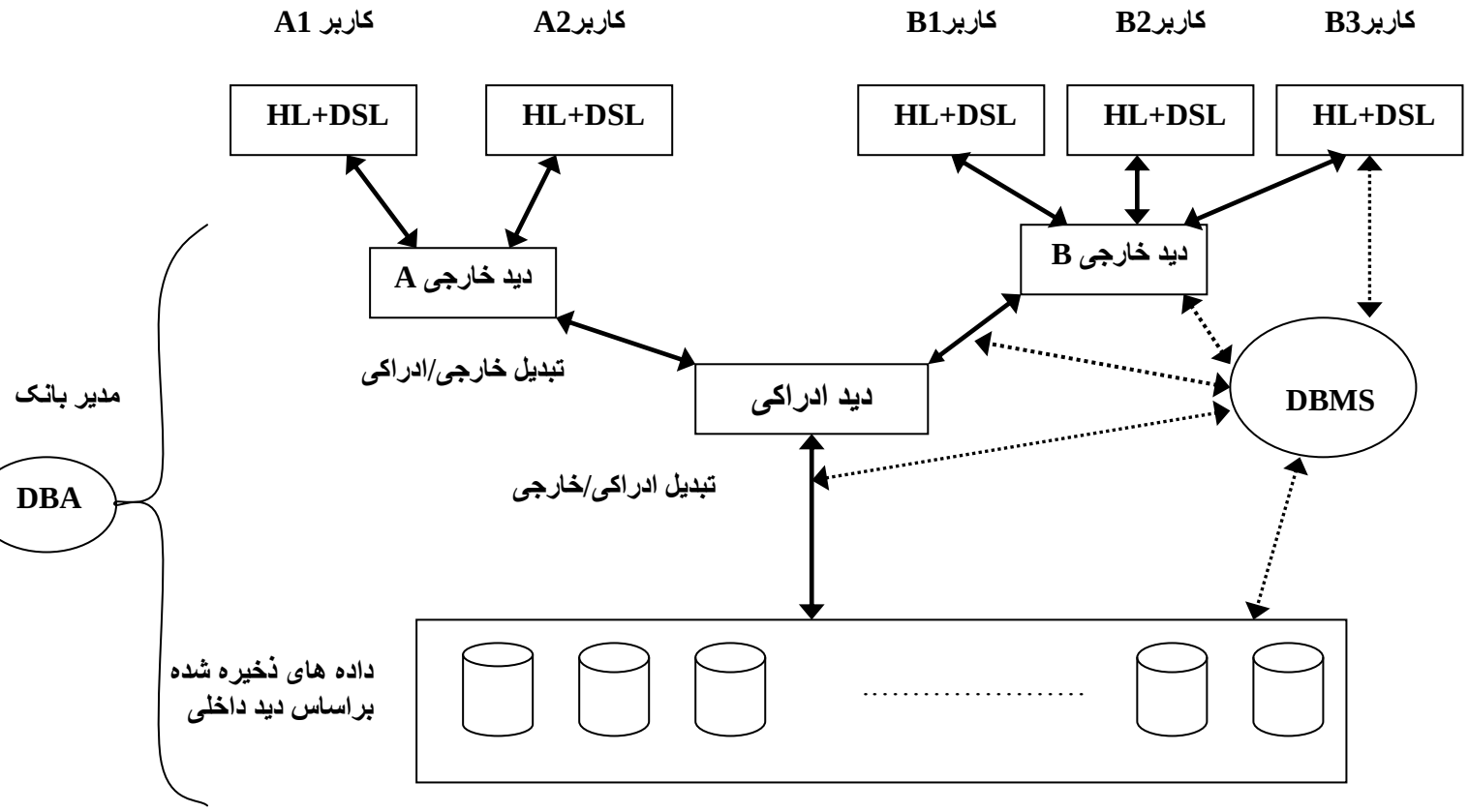
بانک اطلاعاتی توزیع شده

1-بانک های اطلاعاتی با داده های توزیع شده:در این حالت مدیر بانک اطلاعاتی متمرکز داده ها در کل شبکه توزیع شده اند.

2-بانک های اطلاعاتی با داده ها و سیستم های توزیع شده: همه مدیریت بانک اطلاعاتی و هم داده ها در کل شبکه توزیع شده اند.

معماری سیستم بانک اطلاعاتی

مدل پیشنهادی ANSI (ارائه شده در سال 1975) برای معماری سیستم بانک اطلاعاتی به شکل زیر است:



بانک فیزیکی

معماری سیستم بانک اطلاعاتی از اجزاء زیر تشکیل شده است:

الف) دید ادارکی یا مفهومی (Conceptual view)
ب) دید خارجی (External view)
ج) دید داخلی یا فیزیکی (Internel view)
د) تبدیلات بین سطوح(Transformation یا mapping)
ه) زبان تبدیلات یا HL(Host Language)
و) زبان فرعی داده یی یا DSL (Data Sub Language)

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

الف: دید ادراکی(مفهومی)

دید طراح بانک است از داده های ذخیره شده در بانک.

ب) دید خارجی

دید کاربر خاص است از داده های ذخیره شده در بانک. هر کاربر دید خاص خود را دارد. همچنین چند کاربر می توانند دارای دید یکسانی باشند.

STTAB1

سال ورود	شماره دانشجویی

ج) دید داخلی

در این سطح در واقع فایل های محیط فیزیکی تعریف می شود. از نظر محتوا ،ساختار و استراتژی دستیابی.

د) تبدیلات بین سطوح

STCOTAB1

شماره درس	شماره دانشجویی

1 - تبدیل داده ها :یعنی تبدیل داده های تعریف شده در سطح خارجی به داده های تعریف شده سطح ادراکی و بالاخره به داده های تعریف شده در سطح داخلی و نیز مسیر بر عکس.

2 - 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.

3 - تبدیل ساختار : یعنی تبدیل ساختار سطح خارجی به ساختار سطح ادراکی.

ه)زبان میزبان (HL) و زبان فرعی داده یی (DSL)

منظور از زبان میزبان یکی از زبان های سطح بالای برنامه سازی مثل کوبول ، PL/1 ،C ،پاسکال،بیسیک،Delphi،visual، visual Basic، Java می باشد.

زبان DSL زبانی است از سطح بالاتر که میهمان یک زبان سطح بالا مثل Visual C می شود هر مدل داده یی خاص (مثل سلسله مراتبی ،شبکه ای ،رابطه ای)زبان فرعی خاص خود را دارد.

احکام زبان DSL را می توان به سه دسته زیر تقسیم کرد:

1 - احکام تعریف داده ها (Data Definition Language= DDL)

2 - احکام کار با (پردازش) داده ها (Data Manipulation Language=DML)

3 - احکام کنترلی (Data Control Language=DCL)

لغت نامه داده ها و کاتا لوگ سیستم

لغتنامه داده ها (Data Dictionary) شبیه لغتنامه های معمولی ،تمامی اسامی استفاده شده در سیستم و معنای آنها را در بر می گیرد.

کاتالوگ : علاوه براسامی داده ها و مشخصات آنها ، اطلاعات دیگری نیز در مورد بانک اطلاعات باید نگهداری شود مثلاً حق دستیابی به داده ها و ... . کاتالوگ شامل لغتنامه نیز می شود.

امنیت و جامعیت

امنیت Security به معنای محافظت در برابر خطراتی از قبیل آتش سوزی و نیز جلوگیری از دستیابی غیر مجاز به آنها ست مثلاً استفاده از رمز عبور (Password)

جامعیت (integrity) به معنای صحت داده ها و پردازش ها و پیروی از مقررات سیستم است.مثلاً موجودی واقعی حسابهای بانکی نباید منفی باشد و یا شخص نتواند بیش از موجودی خود از حسابش برداشت کند.

تراکنش(Transaction)

هر برنامه ای که توسط کاربر در محیط بانک اطلاعاتی اجرا می شود تراکنش نام دارد.تفاوت اصلی یک تراکنش با یک برنامه معمولی در محیط غیر بانکی این است که تراکنش همواره به DBMS تسلیم می شود و DBMS در اعمال هر گونه کنترل و حتی به تعویق انداختن و ساقط کردن آن آزادی عمل دارد.

تضمین جامعیت بانک اطلاعاتی

آقای جیم گری (Jim Gray) در سال 1981 ثابت کرد که چهار کنترل زیر لازم است روی تمامی تراکنش ها در بانک اطلاعات اعمال گردد تا صحت و جامعیت آن تضمین شود این کنترلها به خواص ACID معروفند.

1- یکپارچگی (atomicty) : 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 می خواهد 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 می شود . حال در نظر بگیرید پس از انجام پخش اول (برداشت پول) ارتباط با ماشین دوم ناگهان قطع می شود بدیهی است که در اینحالت باید پول برداشت شده دوباره به همان حساب اول بازگردانده شود.

2- همخوانی (Consistency) : این خاصیت را به این صورت بیان می گردد که «هر تراکنش اگر به تنهایی اجراء شود بانک اطلاعات را از حالتی صحیح به حالت صحیح دیگری منتقل می کند» یعنی این خاصیت می گوید که هر تراکنش باید تمامی قوانین جامعیت بانک اطلاعاتی را راعایت کند.

تراکنش ممکن است دو نوع پایان داشته باشد : الف)پایان ناموفق که آنرا سقوط (abort) می نامند.ب) پایان ناموفق که آنرا انجام (commit) می نامند.

3- انزوا (isolation) : در بانک اطلاعاتی ممکن است تراکنش های همروند وجود داشته باشند(مثل multitasking در سیستم عامل windows که چند برنامه همزمان اجراء می شوند). بر طبق خاصیت انزوا همروندی تراکنش ها باید کنترل شود تا اثر مخرب بر روی هم نداشته باشند به عبارتی دیگر اثرتراکنش های همروند روی یکدیگر چنان است که گویا هر کدام در انزوا انجام ی شود این کنترل ها توسط بخشی از DBMS به نام واحد کنترل همروندی (Concurrency Control) انجام می شود. کتاب isolate را به صورت زیر بیان می کند:

Isolate یعنی به هنگام سازی حاصل از تراکنش T1 توسط تراکنش دیگری مثل T2 قابل مشاهده نیست مگر اینکه T1 عمل COMMIT را اجراء کند.COMMIT موجب می شود تا به هنگام سازی هایی که توسط یک تراکنش انجام شد ، توسط تراکنش های دیگر قابل رویت باشد. اگر تراکنش ROLLBACKرا اجراء کند تمام به هنگام سازیهایی که انجام شده از بین می روند.

4- پایانی (durability) : براساس این خاصیت تراکنش هائی که به مرحله انجام (commit) برسند.اثرشان ماندنی است و هرگز به طور تصادفی از بین نمی رود. مثلاً اگر مبلغی به حسابی واریز شود تراکنش مربوط انجام یافته اعلام شود حتی در صورت وقوع آتش سوزی 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.

اسقلال داده ها

منظور از استقلال داده ها مستقل بودن ذخیره سازی داد ها از کاربرد آنهاست. مثلاً مدل رابطه ای از تجریدی به نام جدول استفاده می کند و داده ها هر چند باشند در قالب چند جدول ریخته می شوند و نحوهٔ ذخیره سازی داده ها روی رسانه ها از دید کاربران مخفی است.

مزایای سیستم بانک اطلاعاتی

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

1 - وحدت ذخیره سازی کل داده های محیط عملیاتی
2 - اشتراکی شدن دادهها
3 - کاهش میزان افزونگی
4 - عدم وجود ناسازگاری در داده ها
5 - تضمین جامعیت
6 - تأ مین استقلال داده یی
7 - تعدد زبانهای میزبان

معایب سیستم بانک اطلاعاتی

ممکن است امنیت لازم (بدون کنترلهای مناسب) به مخاطره بیفتد چرا که داده ها متمرکز بوده و این تمرکز آنها را آسیب پذیر می سازد به همین ترتیب بدون کنترل های مناسب ممکن است جامعیت داه ها نیز به خطر بیفتد.

1-متمرکز بودن دادهها باعث امنیت پائین می شود

2- ممکن است سخت افزار اضافی نیاز باشد.
3-عملیات پیچیده تر

وظایف DBMS

1 - تعریف داده ها
2 - دستکاری داده ها .می تواند «برنامه ریزی شده » یا «برنامه ریزی نشده» باشد. در خواستهای برنامه ریزی شده در خواستی است که قبل از اجرای آن،پیش بینی شده باشد.احتمالاًDBMS طراحی بانک اطلاعاتی فیزیکی را طوری تغییر می دهد که کارایی در خواستهای برنامه ریزی شده بالا باشد.در خواستهای برنامه ریزی نشده یا موردی ،در خواستهای است که ازقبل پیش بینی نشده است و در صورت نیاز ارائه می شود. طراحی بانک اطلاعاتی فیزیکی ممکن است برای پاسخ به درخواست مورد نیاز ایده آل باشد یا نباشد . در خواستهای برنامه ریزی نشده معمولاً به صورت محاورهای صادر می شوند.
3 - بهینه سازی و اجراء
4 - جامعیت و امنیت داده ها. از نظر درستی می تواند در زمان کامپایل ،زمان اجراء یا هر دو زمان انجام شود.
5 - ترمیم و سازگاری داده ها
6 - فرهنگ داده ها.راجع به داده هاست که گاهی شبه داده ها یا توصیفات نامیده می شوند.یعنی سایر اشیای سیستم را تعریف می کنند.نام دیگر آن در بعضی کتاب ها کاتالوگ
7 - کارایی

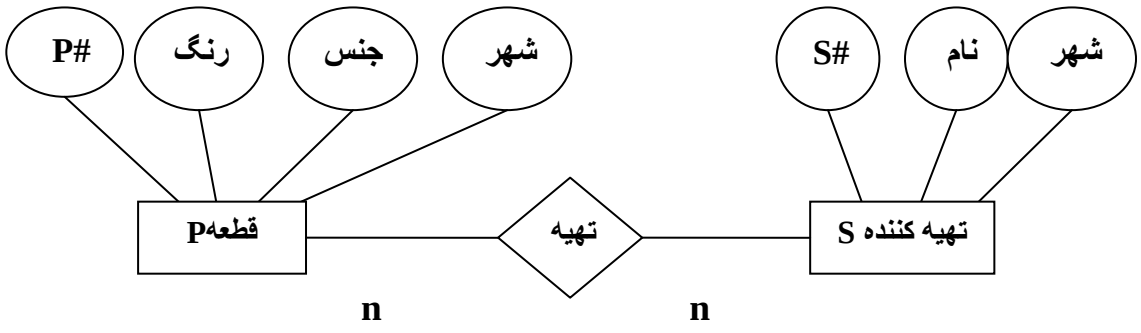
فصل سوم

به کمک ساختار دادهیی،مهمترین سطح بانک یعنی سطح ادراکی تعریف و تشریح می شود ، سطحی که حالت انتزاعی داردو مستقل از مفاهیم فایلینگ باید شکل گرفته و معرفی شود.

1 - ساختار رابطه ای

ازدید کاربر ،بانک اطلاعاتی از تعدادی جدول تشکیل یافته است. 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 می باشد.می توان تصور کرد که یک جدول شبیه یک فایل ترتیبی مسطح است و بانک رابطه ای مجموعه ای از همین فایلهاست. از آنجاد که رابطه با جدول معادل است فعلاًبانک رابطه ای را می توانیم بانکجدولی نیز بنامیم.

مثال 1 : موجودیتهای قطعه و تولید کننده را در نظر گرفته و نمودار EER آن ا ترسیم می کنیم.



برای تشریح نمودار فوق دربانک رابطه ای یک جدول برای هر یک از دو موجودیت و جدولی نیز برای بیان ارتباط آنها ،استفاده می شود:

P#	رنگ	جنس	city

تهران	آهن	قرمز	P1

⇓

تهران	آهن	قرمز	P1
تبریز	مس	سبز	P2
شیراز	برنج	آبی	P3
تهران	آهن	قرمز	P4

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

جدولSP (محموله)

جدول P(قطعه)

S#	نام تهیه کننده	city
S1	فن آوران	تهران
S2	ایران قطعه	تبریز
S3	پولادین	تبریز

جدول S (تهیه کننده)

از ساختار جدو ودیتها استفاده می شود و
زایای مدل رابطه ای است که در مدلهای دیگر (سلسله مراتبی و [illegible]

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

Insert Into S (نام جدولی که این مشخصات را دارد) ('S4'،' آلومین' ، 'اصفهان')

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

Delet form SP where S#='S3' and P#='P2'

(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 Detet و Update به فرم SQL نوشنه شده اند.

خصوصیات مدل رابطه ای

1 - از دید کاربر ،دارای وضوح بوده و محیط انتزاعی آن محیطی مسطح است.
2 - داده ها وارتباطات بین آنها با مکانیسم واحدی نشان داده می شود(سطرها).
3 - عملگر بازیابی نسبتاً ساده است.
4 - برای پرس و جوهای قرینه دارای رویه پاسخگویی واحدی است.
5 - در عملیات ذخیره سازی دشواری ندارد و سبب بروز وضعیت نا مطلوب نمی شود.
6 - از مبنای تئوریک ریاضی برخوردار است.
7 - پیمایش (غواصی) در هر یک از رابطه ها (جدولها) می تواند مستقل از رابطه دیگرانجام شود.
8 - برای پاسخگوئی به بعضی از پرس و جوها باید رابطه های مستقل از یکدیگر به نحوی با هم مرتبط شوند و این امر می تواند باعث افزایش زمان پاشخ دهی شود و لذا لزوم بهینه سازی رویه های پرس و جو مطرح می گردد.
9 - برای طراحی رابطه ها به صورت مطلوب دارای ابزار تئوریک است.

ساختار سلسله مراتبی (ساختار درختی)

این ساختار قدیمی ترین ساختار دادهیی برای طراحی بانک اطلاعاتی در سطح انتزاعی است. در این ساختار داده ها و ارتباط بین آنها به کمک یک درختواره نمایش داده می شوند.درختواره گرافی دارای یک ریشه ،به هم بسته و غیر چرخشی است.

مثال

تبریز	مس	سبز	P2

⇓

200	تهران	فن آوران	S1
400	تبریز	ایران قطعه	S2
200	تبریز	پولادین	S3
تهران	آهن	قرمز	P4

300	تهران	فن آوران	S1
300	تبریز	ایران قطعه	S2
شیراز	برن ج	آبی	P3
400	تهران	فن آوران	S1

در شکل فوق P4 هنوز توسط تهیه کننده ای تولید نشده است لذا فرزندی ندارد. با این همه P4یک سلسله مراتب است که فقط ریشه دارد و اصطلاحاً به آن سلسله مراتب «فقط ریشه» یا Root Only گفته می شود.

خصوصیات مدل سلسله مراتبی

1-از دید کاربر وضوح دارد ولی نه به حد مدل رابطه ای و محیط انتزاعی آن مسطح نیست.

2-ارتباط بین داده ها به کمک یک درختواره که مسیر منطقی آن از بالا به پایین است نشان داده می شود،لذا خاص محیط هایی است که در آنها ارتباط های یک یا چند یک سویه وجود دارد.

3 - عملگرهای بازیابی به سادگی عملگرهای مدل رابطه ای نیستند. برای انجام عملیات از پیمایش اشاره گرها استفاده می گردد.

4 - غواصی در سطحی پایین تر الزاماً باید از نقطه ورود،در سطح بالاتر شروع شود.

5 - برای پسخگوی به پرس وجوهای قرینه رویه های پاسخگوئی قرینه ندارد.

6 - در عملیات ذخیره سازی آنومالی دارد.

7 - از مبانی تئوریک ریاضی (آنگونه که مدل رابطه ای دارد) برخوردار نیست.

8 - قدیمی ترین ساختار داده یی برای طراحی بانک در سطح انتزاعی است.

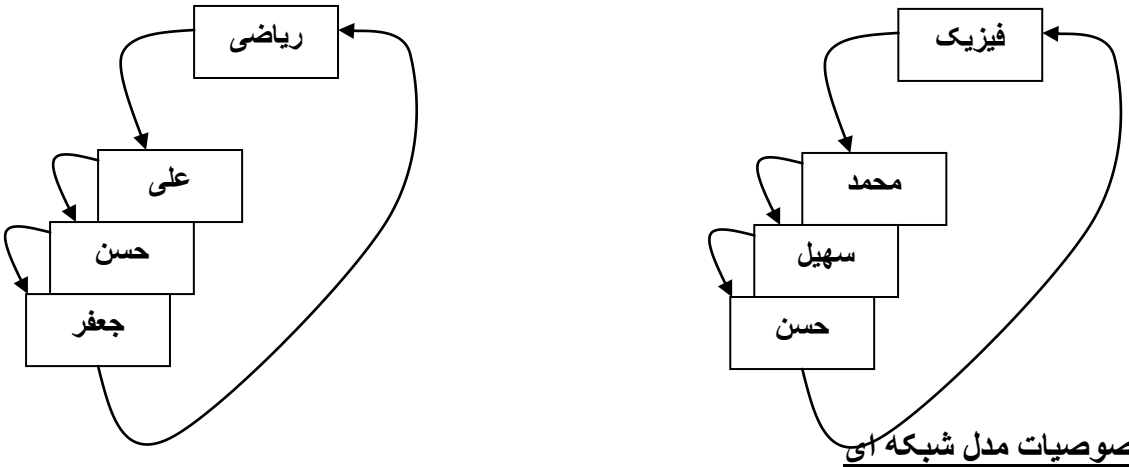
9 - طراحی ساختار براییک محیط عملیاتی ممکن است در بیش از یک صورت انجام شود و بر خلاف مدل رابطه ای تئوری منسجمی برای طراحی ندارد.

10 -وجود ارتباطات از پیش پیاده سازی شده بین انواع رکوردها ،تا حدودی پاسخگوئی به پرس وجوها را در مرحله اجراء تسریع می کند.

ساختار شبکه ای

در این ساختار هر گره فرزند می تواند بیش از یک گره پدر داشته باشد.این ساختار که جامع تر از ساختار سلسلع مراتبی است برای نمایش ارتباطات یک به چند دوسویه مناسب است.

مثال:



خصوصیات مدل شبکه ای

1 - از دید کاربر وضوح کاربری ندارد و محیط انتزاعی آن محیطی مسطح نیست.

2 - ارتباط یک به چند دوسویه با طراحی دو مجموعه که رکورد پیوند دهنده در هر عضویت دارد،پیاده سازی می شود.

3 - برای محیط های دارای ارتباط یک به چند دو سویه مدل مناسبی است. ارتباط یک به چند یک سویه حالتی خاص از ان است.
4 - عملگر بازیابی پیچیده تر از مدل سلسله مراتبی است ولی خاصیت تقارن دارد.
5 - در عملیات ذخیره سازی آنومالی ندارد.
6 - اصل وحدت عملگر در یک عمل واحد مثل درج رعایت نمی شود.
7 - علیرغم وجود مفهوم مجموعه ،از مبانی تئوریک ریاضی برخوردار نیست(آنگونه که مدل رابطه ای برخوردار است).
8 - در بعضی پرس و جو ها ،سیستم با مساله انتخاب مسیر (نقطه ورود غواصی مواجه است).
9 - وجود ارتباطات از پیش پیاده سازی شده ،تا حدودی پاسخگوئی پرس و جوها را در مرحله اجراء تسریع می کند.

فصل چهارم

رابطه در ریاضیات

دامنه یا میدان(Domain) : مجموعه مقادیر مجازیک صفت .
رابطه:رابطه زیر مجموعه ای است از ضرب دکارتی چند دامنه.
تاپل(Tuple) :به عبارتی دیگر تاپل مجموعه ای است از مقادیر صفات خاصه.
کلید :یکسری صفت های خاصه است که منحصربه فرد است.
رابطه از دو مجموعه عنوان (Heading) و پیکر (body) تشکیل یافته است. مجموعه عنوان مجموعه اسامی صفات خاصه است و مجموعه پیکر، مجموعه ای است متغیر در زمان از تاپل ها.

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

خصوصیات رابطه

رابطه طبق تعریف از دو مجموعه (عنوان- پیکر) تشکیل شده است و لذا خصوصیات زیر را داراست:
1 - 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 رکوردها ی تکراری نداریم.
2 - تاپلها در رابطه نظم ندارند. این خوصیت نیز از مجموعه بودن پیکر رابطه نتیجه میشود.به همین ترتیب در بانکهای اطلاعاتی جدولی نیز ترتیب رکوردها در جدول مهم نیست. هر چند که معمولاً سطرها با ترتیب خاص نشان داده می شود.
3 - صفات خاصه نظم ندارند. این خاصیت نیز از مجموعه بودن عنوان رابطه نتیجه می شود. به همین ترتیب فیلد های یک جدول نیز نظم ندارد و می توان آنها را جا به جا کرد.هنگام تعریف یک جدول مهم نیست ترتیب فیلدها چگونه باشد مثلاً: S(sname,city,s#) ⟷ S(s#,sname,city)
4 - همه مقادیر صفات خاصه تجزیه ناپذیرند. به عبارتی دیگر در رابطه، یک تاپل نمی تواند حاوی تاپل دیگری باشد. مثلاً مجموعه R زیر ،رابطه نیست، چون عضو((1, (4,5)) قابل قبول نمی باشد:
R{(1,(2,3)),(1,(4,5)), …}

مقدار اتومیک(atomic) مقداری است ساده که قابل تجزیه به مقادیر دیگر نباشد،به بیان دیگر از یک میدان ساده برگرفته شود.

انواع کلید در مدل رابطه ای

1 - 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 مجموعه ای از ان هم ممکن است کلید باشد. مثلاً «شماره دانشجوئی» و «نام دانشجو – شماره دانشجوئی» هر دو ابر کلید هستند.

2 - کلید کاندید(candidate key یا C.K.) : هر ترکیبی از صفتها ست که کلید کمینه باشد . یک رابطه ممکن است چند کلید کاندید داشته باشد.

3 - کلید اصلی (primary key یا P.K.) : کلید کاندیدی است که توسط مدیر بانک اطلاعاتی انتخاب و معرفی می شود.

دو شرط کلید اصلی

الف) نقش و اهیت کلید اصلی نسبت به سایر کلید های کاندید در پاسخگوئی به نیازهای اطلاعاتی کاربران

ب) کوتاه تر بودن طول کلید کاندید از نظر طول رشته بایتی.

4- کلید فرعی یا بدیل (Alternative key یا A.k.) : هر کلید کاندید غیر از کلید اصلی را کلید فرعی می نامند

5-کلید خارجی (Foreign key یا F.K.) : صفتی است در یک رابطه که در رابطه دیگری کلید اصلی (یا فرعی) است و برای برقراری ار تباط بین دو رابطه استفاده می شود.

قواعد جامعیت در مدل رابطه ای

1 - قاعده جامعیت درون رابطه ای (intra-relation integrity rule) یعنی هر رابطه ای به تنهایی صحیح باشد. مثلاًعضو تکراری نداشته باشد وکلیدهایش درست باشند.

2 - قاعده جامعیت موجودیتی (Entity integrity rule) یعنی هیچ جزء تشکیل دهنده کلید اصلی نباید دارای مقدار هیچ (NULL value) باشد.

3 - 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 رابطه R2 :

- می تواند مقدار NULL داشته باشد.
- در غیر این صورت حتماً باید مقداری داشته باشد که در تاپلی از رابطه R1 موجود باشد.

رابطه های DEE وDUM

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

DUM(یا TABLE-DUM) که هیچ تاپلی ندارد.

فصل پنجم

جبر رابطه ای در واقع مبنای تئوریک مدل رابطه ای است به مجموعه ای از قوانین و عملگرها که امکان پردازش جداول را فراهم می سازند، جبر رابطه ای می گویند . نوع داده در جبر رابطه ای فقط رابطه است یعنی ورودی و خروجی تمامی عملگرها رابطه می باشد.

عملگرها در جبر رابطه ای را می توان به چهار دسته تقسیم کرد:

1 - عملگرهای ساده شامل گزینش (restrict , select یا σ) و پرتو (project یا л)

2 - عملگرهای مجموعه ای شامل اجتماع (ں)، اشتراک(∩) ،و تفاضل (-)

3 - عملگر های پیوند شامل ضرب دکارتی (×)، پیوند طبیعی (∞) ،نیم پیوند (œ) ،پیوند شرطی (X_θ) و فرا پیوند

4 - عملگر های دیگر مثل نامگذاری(p) ، تقسیم (÷) ، جایگزینی (←) و غیره.

خاصیت بسته بودن

این واقعیت که خروجی حاصل از هر عمل رابطه ای یک رابطه دیگر است،خاصیت بسته بودن یا بستار (closure) نام دارد. مثلاً اجتماع دو رابطه (جدول) یک رابطه (جدول) می شود.

تذکر: اکثر مثال ها براساس جداول S و P و SP زیر بیان شده است.

S P SP

S#	P#	Qtyتعداد

S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

P#	رنگ	جنس	city
P1	قرمز	آهن	تهران
P2	سبز	مس	تبریز
P3	آبی	برنج	شیراز
P4	قرمز	آهن	تهران

S#	نام تهیه کننده	city
S1	فن آوران	تهران
S2	ایران قطعه	تبریز
S3	پولادین	تبریز

عملگر گزینش

عملگر گزینش سطرهایی از جدول را می دهد.

مثال 1 : تاپل هایی را از رابطه S مشخص سازید که صفت شهر آنها تهران باشد.

خروجی (تهران ، فن آوران،S1) خواهد بود یکی از دستورهای زیر را می توان نوشت:

الف)$\sigma_{city = 'تهران'}$ (S)

ب)Select S where city= 'تهران '

ج)Restrict :

S where city = 'تهران'

عملگر پرتو

عملگر پرتو یا تصویر ستونهایی از جدول را انتخاب می کند و برای آن هیچگونه شرطی قائل نمی گردد. در خروجی پرتو سطرهای تکراری حذف می شوند.

مثال 3: ستون شهر را از جدول S چاپ کنید.

الف)$\prod_{city}$ (s)

ب)Project S[city]

ج) project:

S over city

خروجی

تهرا ن

تبریز

عملگر های ∪ ،∩،–

اجتماع دو رابطه رابطه ای است که تاپل هایش در یک یا هر دو رابطه وجود دارند. یعنی رکوردهای دو جدول با هم ترکیب شده و رکورد های تکراری یکبار نوشته می شوند. اجتماع را به یکی از دو صورت زیر نمایش می دند:R=R1 Union R2 یا R1=R ∪ R2

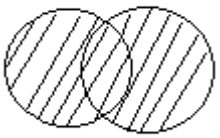
اشتراک دو رابطه رابطه ای است که تاپل هایش در هر دو رابطه وجود داشته باشند:

R=R1Interect R2 یا R=R1 ∩ R2

تفاضل دو رابطه ، رابطه ای است که تاپل هایش در رابطه اول موجود است ولی در رابطه دوم وجود ندارد.

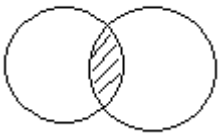
R=R1 Minus R2 با R = R1 - R2

R1 R2



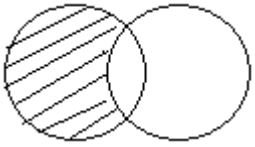
R1∪ R2

R1 R2



R1∩ R2

R1 R2



R1 - R2

لیست شهر های تهیه کنندگان و قطعات را بدهید.

خروجی
تهران
تبریز
شیراز

$\prod_{city}(S) \cup \prod_{city}(P)$

عملگر های پیوند یا الحاق (Join)

این عملگر ها پر کاربرد و قدرتمند می باشند ولی گاهی اوقات باعث مصرف فضا و زمان زیادی می گردند لذا هنگام استفاده از آن ها باید دقت کافی را کرد. این عملگر ها عبارتند از ضرب دکارتی (×)، پیوند طبیعی (∞) ،نیم پیوند (œ) ،پیوند شرطی (X_θ) و فرا پیوند.

ضرب دکارتی (×)

وظیفه اش ترکیب کردن جداول است. ضرب دکارتی را به صورت های زیر نشان می دهند:

R=R1 ×R2 R=R1 TIMES R2

مثال: حاصاضرب دکارتی دو رابطه S و P چه می شود؟ S×P

S#	Sname	S.city	P#	color	Type	P.city
S1	فن آوران	تهران	P1	قرمز	آهن	تهران
S1	فن آوران	تهران	P2	سبز	مس	تبریز
S1	فن آوران	تهران	P3	آبی	برنج	شیراز
S1	فن آوران	تهران	P4	قرمز	آهن	تهران
S2	ایران قطعه	تبریز	P1	قرمز	آهن	تهران
S2	ایران قطعه	تبریز	P2	سبز	مس	تیریز
S2	ایران قطعه	تبریز	P4	آبی	برنج	شیراز
S2	ایران قطعه	تبریز	P3	قرمز	آهن	تهران
S3	پولادین	تبریز	P1	قرمز	آهن	تهران
S3	پولادین	تبریز	P2	سبز	مس	تبریز
S3	پولادین	تبریز	P4	آبی	برنج	شیراز
S3	پولادین	تبریز	P3	قرمز	آهن	تهران

نکته 1 :اگر جدول Aدارای m سطرو n ستون و جدول B دارای p سطر وq ستون باشد آنگاه A×B تعداد m×p سطرو تعداد n×q ستون خواهد داشت.

پیوند طبیعی (∞)

این عملگر دو رابطه را بر مبنای یک یا چند فیلد مشترک به هم پیوند می دهد و رکوردهایی از دو جدول را که مقدار آن فیلد مشترک برای آنها یکسان است به هم می چسباند.

الف)A∞B
ب)A JOIN B
ج) Join :
A and B over k

خروجی S∞ SP را بدست آورید.
ستون مشترک بین S وsp فیلد s# می باشد پس هر سطر جدول S را برداشته پشت سطر هایی از جدول SPمی گذاریم که S# آنها با هم برابر باشند:

S#	Sname	City	P#	Qty

S1	فن آوران	تهران	P1	300
S1	فن آوران	تهران	P2	200
S1	فن آوران	تهران	P3	400
S2	ایران قطعه	تبریز	P1	300
S2	ایران قطعه	تبریز	P2	400
S3	پولادین	تبریز	P2	200

نکته 1: پیوند طبیعی خاصیت شرکت پذیری و جابه جائی دارد یعنی:

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

نکته2: اگر دو جدول فیلد همنام نداشته باشند در این صورت $R1 \infty r2$ تبدیل به $R1 \times R2$ می شود.

نکته 3: اگر تمام فیلدهای دو جدول یکسان باشند آنگاه $R1 \infty R2$ معادل $R1 \cap R2$ خواهد بود.

نکته 4: پیوند طبیعی عملگر گرانی نیست و زمان و فضای کمتری نسبت به ضرب دکارتی می گیرد.

عملگر جایگزینی

با علامت ← جدول حاصل از دستورات ذخیره می شود تا در ادامه مورد استفاده قرار گیرد. اگر دستوری طولانی باشد می توان با استفاده از جایگزینی ، ن را در چند مرحله نوشت. بعضی از کتاب ها از نما =: و بعضی دیگر از عبارت GIVING برای این منظور استفاده کرده اند .

مثال : اسامی تهیه کنندگان قطعه P2 را بدهید:

```
S Join SP Giving Temp1
Select Temp1 where P#='p2' g iving TEMP2
Project TEMP2 [Sname]
```

با نمادی دیگر:

$$\begin{cases} Temp1 \leftarrow S \infty SP \\ Temp2 \leftarrow \sigma_{P\#='P2'} (Temp1) \\ \prod_{Sname} (Temp2) \end{cases}$$

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

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

خروجی دستورات فوق فن آوران ،ایران قطعه و پولادین می شود.

عملگر پیوند شرطی (θ - JOIN)

این عملگر ،زیر مجموعه ای از ضرب دکارتی است که شرط θ روی سطر های آن اعمال شده باشد.

مثال : خروجی این دستور را بدست اورید: $S \; X_{S.city \geq P.city} \; P$

S#	Sname	S.city	P#	color	Type	P.city
S1	فن آوران	تهران	P1	قرمز	آهن	تهران
S1	فن آوران	تهران	P2	سبز	مس	تبریز
S1	فن آوران	تهران	P4	قرمز	آهن	تهران
S2	ایران قطعه	تبریز	P2	سبز	مس	تیریز
S3	ایران قطعه	تبریز	P2	سبز	مس	تبریز

عملگر نیم پیوند (semi – join یا شبه الحاقی)

این عملگر مشابه پیوند طبیعی است با این تفاوت که فقط ستونهای جداول اول را می دهد.

عملگر های فرا پیوند (Outer Join)

نوع دیگری از عملگر پیوند که معمولاً به شرط تساوی است عملگر های فرا پیوند (یا پیوند خارجی) می باشد. عملگر های فراپیوند به سه دسته زیر تقسیم می شوند:

1 - فراپیوند چپ (Left Outer Join) با نماد L-O-JOIN
2 - فرا پیوند راست (Right Outer Join) با نماد R-O-JOIN
3 - فراپیوند کامل (Full Outer Join) با نماد F-O-JOIN

جدول SP و 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

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	Sn4	20	NULL	NULL

عملگر شبه تفاضل یا نیم تفاضل

این عملگر به صورت زیر تعریف می شود:

S SEMIMINUS B = A MINUS (S SEMIJOIN B)

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

عملگر تغییر نام یا نامگذاری

این عملگر به صورت $\rho_b\ \alpha$ نشان داده شده که نام b روی جدول a نیز گذاشته می شود. در این حال بدون ذخیره سازی مجدد یک جدول می توان از آن دوبار استفاده کرد،در واقع اشاره گر جدیدی تعریف می شود. محدوده عملکرد ρ تنها در همان دستور مربوطه است ،یعنی پس از اتمام آن دستور نام جدید دیگر وجود ندارد. گاهی یک پرس وجو ،دو یا چند بار به یک جدول نیاز دارد.

عملگر تقسیم (÷ یا DIVIDEBY)

کاربرد عملگر تقسیم زمانی است که بخواهیم همه حالتهای یک اتفاق را بررسی کنیم مثل حالتهای زیر:

- دانشجویانی که همه درس های استاد اکبری را گرفته اند.
- درس هایی که توسط همه دانشکده ها ارائه می شوند.
- اسامی تهیه کنندگانی که تمام قطعات را تهیه می کنند.

پاسخگوئی به پرس وجو های فوق با استفاده از دستور تقسیم بسیار ساده و بدون آن بسیار مشکل است. ابتدا بخشی را که شامل شرط همه می شود پیدا می کنیم (مقسوم) سپس بخش دیگر را بر آن تقسیم می کنیم . در آن بخش حتماا باید صفت های مندرج در مقسوم علیه وجود داشته باشند و خروجی شامل صفت های باقی مانده خواهد بود.

مثال :

اسامی تهیه کنندگانی را بیابید که اقلاً یک قطعه قرمزرنگ تهیه می کنند.

$\prod_{Sname}$ [($\prod_{S\#}$ ($\prod_{P\#}$ ($\sigma_{Color='Red'}$ (P)) ∞ SP)) ∞

اسامی تهیه کنندگانی را بدهید که تمام قطعات را تهیه می کنند.

$\prod_{Sname}$ (S ∞ ($\prod_{S\#,P\#}$ (SP) ÷ $\prod_{P\#}$ (P))

اسامی تهیه کنندگانی را بیابید که قطعه P2 را تهیه نمی کنند.

$\prod_{Sname}$ [[$\prod_{S\#}$ (S) – ($\prod_{S\#}$ ($\sigma_{P\#='P2'}$ (SP)))] ∞ S]

خواص و فرمو ل های عملگر ها

A×B = B × A

(A ×B) ×C = A × (B ×C)

A∞ B = B∞ A

(A∞ B)∞ C = A∞ (B ∞C)

A∪ B = B ∪A

(A ∪B) ∪C = A ∪ (B ∪C)

A ∩B =B ∩A

(A ∩B)∩C =A ∩ (B ∩C)

σ_P(A ∪B) =σ_P (A) ∪ σ_P (B)

σ_P (A ∩B) =σ_P(A) ∩ σ_P(B)

σ_P (A-B)= σ_P (A) –B=σ_P(A)- σ_P (B)

بهینه سازی پرس و جو

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

جهت بهینه سازی پرس وجو می توان از قواعد زیر استفاده کرد :

1 - گزینش را هر چه ممکن است زودتر انجام دهید.

2 - شرط های ترکیبی را به شرط های متوالی تبدیل کنید. مثلاً می توان $\sigma_{P1 \wedge P2}$ (e) را با $\sigma_{P1}(\sigma_{P2}(e))$ جایگزین کرد.

3 - عملگر پرتو را زود انجام دهید (ولی دیرتر از گزینش)

4 - همانطور که می دانید از نظرریاضی عملگر پیوند طبیعی خاصیت جا به جایی و شرکت پذیری دارد ولی از نظر کامپیوتری زمان و مصرف حافظه عملیات A∞B ممکن است خیلی بیشتر از B∞A باشد و یا اینکه ممکن است A∞(B∞C) بهینه تر از (A∞B)∞C باشد.

به روز در آوردن داده ها

الف: اضافه کردن داده به جدول

افزودن یک یا چند سطر به جدول با استفاده از عملگر های ∪ و ← قابل انجام است.

ب: حذف داده از جدول

برای این کار در جبر رابطه ای نیاز به عملگر جدیدی نیست و با دستورات - و ← انجام پذیر است.

ج : تغییر داده ها جدول

تغییر به کمک عملگر های σ و ← قابل انجام است.

کلید کاندید در روابط حاصله

فرض کنید A وB دو رابطه دلخواه باشند آنگاه :

1 - (σ) هر محدودیت از تمام کلید های کاندید A را به ارث می برد.
2 - ([]) اگر تصویر دلخواه بر روی A شامل هر کلید کاندیدK باشد ، در اینصورت K یک کلید کاندید برای تصویر خواهد بود. در غیر این صورت ،تنها کلید کاندید ترکیب تمام صفات تصویر می باشد.
3 - (×) هر ترکیب مانند K از یک کلید کاندید K_A از رابطه A و یک کلید کاندید K_B از رابطه B یک کلید کاندید برای A×B می باشد.
4 - (∪) تنها کلید کاندید برای A∪B ترکیب تمامی صفات است.
5 - (∩) هر کلید کاندید A یا B یک کلید کاندید برای A∩B می باشد.
6 - (-) هر کلید کاندید از A یک کلید کاندید برای A – B می باشد.
7 - (∞) در حالت خاص هر گاه که صفت الحاقی موجود در A یک کلید کاندید A باشد ، هر کلید کاندید B یک کلید کاندید برای الحاق خواهد بود.
8 - (EXTEND) کلید های کاندید مربوط به یک توسعه دلخواه از Aهمان کلید های کاندیدA می باشند.
9 - (SUMMARIZA) تنها کلید کاندید برای یک خلاصه سازی دلخواه از A مجموعه صفات مشخص شده در عبارت BY می باشد.

فصل ششم

معرفیSQL

زبان SQL (structured Query Language) پیاده سازی آزادی از جبر رابطه ای است که البته بعضی از عملگر های آن مثل تقسیم را نمی پو شاند ولی در عوض عملگر های کاربردی زیاد دیگری تعریف می کند که کارکردن با جداول را آسان می کنند.

این زبان اولین بار در سال 1976 پدید آمده و ده سال بهد توسطANSI استاندارد شد. SQL یک زبان بیانی (declarative) است.بدین معنا که کاربر تنها می گوید «چه می خواهد» ولی چگونگی بدست آوردن آن را مشخص نمی کنند. در واقع تبدیل دستوراتSQL به عملگر های جبر رابطه ای توسط خود سیستم SQL انجام می پذیرد.

SQL به دو صورت مستقل و ادغام شدنی به کار می رود. در دنیای PC ها عموماً SQL را تحت یک نرم افزار دیگر مثل دلفی ، ویژوال بیسیک ، ویژوال C یا Access به کار می بریم. در این حال برای تعریف انواع متغیر ها 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 می باشد. SQL/DS زبانی مستقل می باشد.

معمولترین نسخه این نرم افزار در حال حاضر SQL2 می باشد. البته در حال حاضر زبانی به نام OSQL پدید آمده و ANSI نسخه جدیدی به نام SQL3 را اعلام کرده که شیی گرایی را پشتیبانی می کند. ما در این فصل SQL2 را شرح می دهیم.

تذکر: تعدادی از زبانهای رابطه ای موجود عبارتند از :DATALOG – QBE –QUEL –SQL

- SQL در ابتدا یک زبان داده ها (DSL) بود. با قرار دادن ویژگی «روال ذخیره شده پایدار» (Persistent Stored Modules = PSM) در استاندارد سال 1996 ، SQL از نظر محاسبات نیز کامل شد و اکنون حاوی دستوراتی مثل CALL،Return،set،case، if، loop، while و Repeat و ویژگی هائی مثل متغیر ها و پردازش استثنا ها می باشد. 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 رابطه و متغیر رابطه ای از جدول استفاده می کند. SQL از اصطلاحات عنوان و بدنه استفاده نمی کند.
- SQL با زبان رابطه ای فاصله دارد. با این حال استاندارد است و اغلب محصولات موجود در بازار آن را پشتبانی می کنند.
- کاراکتر # در اسامی ستون ها در SQL مجاز نیست هر چند که ما در مثال هایمان از آن استفاده می کنیم.

انواع داده ای در SQL2

تعدادی از انواع متغیر ها در SQL2 عبارتند از :

INTEGER عدد صحیح

SMSLLINT عدد صحیح
DECIMAL(p,q) عدد دهدیی دارای ح رقم و ض 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≤n≤254)
VERCHAR(n) رشته ای کاراکتری به طول متغیر حداکثر n(1 ≤ n≤32767)
DATE تاریخ با نمایش هشت رقم دهدهی بدون علامت(yyyymmdd)
TIME زمان با نمایش شش رقم دهدهی بدون علامت (hh mmss)
TIMESTAMP ترکیبی از تاریخ و زمان با دقت میکروثانیه با نمایش 20 رقم دهدهی بون علامت (yyyymmddhhmmssnnnnnn)
BIT(n) اعداد مبنای 2
LOGICAL برای داده های منطقی

تعریف دامنه های جدید

با دستور Creat domain می توان دامنه های جدیدی را تعریف کرد.
مثال 1 :

```
Creat domain seasion cher (8)
Default 'bahar'
Check (Value in 'bahar' , 'tabestan', 'paeez' 'zemestan');
```

با دستور فوق دامنه ای جدید به نام Seasion تعریف می شود که مقادیر اسامی فصل ها را به خود می گیرد و مقدار پیش فرض آن بهار بوده و از نوع کاراکتری 8 خانه ای می باشد.
تذکر: با دستور Alter domain می توان تعریف میدان را تغییر داد و با دستور Drop domaion می توان میدان تعریف شده ای را حذف کرد.

اپراتور ها در SQL

عملگر های ریاضی عبارتند از :+ ،- ، * ، /
عملگر ‖ برای الصاق دو رشته کاراکتری استفاده می شود. مثلاً با دستور NAME ‖ FAMILY دو متغیر رشته ای با هم ترکیب می شوند. در بعضی از پیاده سازی ها علامت + برای چسباندن دو رشته استفاده می شود.
عملگرهای 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 , SND
تذکر : وقتی که فیلدی از یک رکورد NULL (پوچ – هیچ – هیچ مقدار) باشد معنایش این است که مقدار آن فیلد نا شناخته است . هر فیلدی کی تواند 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 آن را NULL می دهد. اگر یکی از عملوندهای عبارات محاسباتی مقدار NULL داشته باشد تمام عبارات برابر NULL می شود. مثلاً اگر فیلد WEIGHT برابر NULL باشد عبارتWEIGHT * 454 نیز NULL می شود.
مقدار NULL در عمل مقایسه نیز نقش خاصی دارد اگر یکی از عملوندهای مقاسه ای NULL باشد نتیجه نیز NULL خواهد شد. نقش مقدار ناشناخته یا NULL (که در جدول زیر با ? نشان داده شده است) در رابطه منطقی به صورت زیر است:

A	B	AND B
T	T	T
T	F	F
F	T	F
F	F	F
T	?	?

?	T	?
F	?	F
?	F	F
?	?	?

A	B	A OR B
T	T	T
T	F	T
F	T	T
F	F	F
T	?	T
?	T	T
F	?	?
?	F	?
?	?	?

A	NOT A
T	F
F	T
?	?

مفهوم NULL می تواند منشأ اشتباهات و ابهاماتی در سیستم گردد.

تذکر : اگر SQL میزبان زبان دیگری باشد آنگاه از عملگرها ، متغیرها و توابع داخلی آن زبان استفاده می گردد.

تعریف بانک در SQL

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

الف) دستور Creat table

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

مثال2 :

```
Creat table s
        (S# cher (5) NOTNULL,
         Sname cher (20)   NOTNULL,
        Status smallint ,
        City cher(15)   NOTNULL,
        Primary Key (S#)
```

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

مثال3 : دستور زیر جدول SP را تعریف می کند:

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

نام پس از References معین می کند که این کلید خارجی به کدام جدول ارجاع می شود. عبارات On delet cascade و On update cascade می گویند هنگامی که این کلید در جدول اصلی خودش حذف شد یا تغییر کرد ، در این جدول هم حذف یا تغییرات اعمال گردد. وجود این قید های On delete و On update اختیاری می باشند.

قسمت Check که اختیاری می باشد برای بیان قوانین جامعیتی به کار می رود. در جدول فوق فیلد Qty باید مقداری ،بین 1 تا 1000 داشته باشد.

تذکر : با کلمه کلیدی UNIQUE می توان کلیدهای فرعی را مشخص ساخت که نمی توانند تکراری باشند.

ب) Alter table

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

```
ALTER TABEL S
ADD DISCOUNT SMALLINT
```

دستور فوق (به همراه کلمه کلیدی ADD) ستونی به نام اختیاری DISCOUNT را به جدول S اضافه می کند. تمام رکوردهای موجود S با اجرای این حکم به جای چهار فیلد ، 5 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 می توان تعریف کلید اصلی یا کلید خارجی را به تعریف جدول اضافه یا از آن حذف کرد. همچنین می توان یک قاعده جامعیت جدید را برای یک جدول وضع کرد یا آن را حذف کرد.

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

مثال 5 :

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

مثال در واقع نوع داده را تغییر نمی دهد بلکه طول #S را از 5 به افزایش می دهد. اگر بخواهیم نوع داده را به طور کلی عوض کنیم ، اکثر نسخه های SQL از این کار جلو گیری می کنند مگر آنکه ستون های مربوطه فاقد داده باشند :

مثال 6:

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

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

ج) DROP TABEL

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

د)Creat Index

شاخص (Index) جدولی است که براساس فیلدی از یک جدول پایه، یه صورت مرتب شده ساخته می شود. مفهوم جدول ایندکس را در درس ذخیره و بازیابی خوانده اید.

مثال 7 : با دستور زیر :

```
Creat Index SN on s (sname, city)
```

شاخصی به نام اختیاری SN روی جدول S ایجاد می شود و نظم اصلی روی مقادیر صعودی Sname و سپس روی مقادیر صعودی city می باشد. اگر بخواهیم براساس نزولی باشد بعد از نام فیلد عبارت DESC را می آوریم.

مثال8:

```
Creat Index SN on S (Sname DESC)
```

مثال 9 :اگربعد از Creat عبارت UNIQQUE را بیاوریم:

```
Creat UNIQYE INDEX SN ON S (Sname)
```

در اینصورت SQL از درج مقدار تکراری برای Sname(مثالً توسط دستور insert) در جدول S جلوگیری می کند. در واقع یکتایی مقدار Sname را تضمین می کنیم. ضمناً نمی توان روی ستونی که مقادیر جاری اش یکتائی ندارند،درخواست ایجاد شاخص یکتا را کرد. سیستم SQL به صورت خودکار روی کلید اصلی شاخص یکتا ایجاد می کند.

مثال 10:

```
Creat Index SC On S (city)
```

چون فیلد شهر یکتا نیست ،نمی توانیم درخواست ایجاد شاخص یکتا را بکنیم.در مورد استفاده یا عدم استفاده از شاخص در پاسخگوئی به یک سؤال کاربر سیستم SQL تصمیم می گیرد و نه کاربر.

تذکر :تعریف شاخص اگر تعداد داده ها زیاد باشد،سرعت عملیات را خیلی بالا می برد. ولی دو ایراد کوچک دارد.اول آنکه مقداری از حافظه جهت ذخیره شاخص مصرف می شود(که البته در حال حاضر که حافظه ها بزرگ می باشند مهم نیست). ایراد دوم مقدار زمانی است که برای برسی فایل اطلاغاتی و شاخص گذاری آن صرف می شود هر چند که این زمان ممکن است زیاد باشد ولی فایل یکبارشاخص گذاری شده و برای مدت طولانی استفاده می شود. به

علاوه شاخص گذاری را می توان در مواقع بی کاری سیستم (مثل شبها) انجام داد. با افزودن رکوردهای جدید به فایل ،شاخص های آن به صورت خودکار توسط سیستم به هنگام می شوند.

تذکر : از آنجا که foxpro برای ساخت جدول پایه و ایندکس دستورات خاص خود را دارد دستورات Creat table وCreat Index مربوط به SQL را پشتیبانی نمی کند.

Drop Index (ه

با این دستور شاخص ایجاد شده حذف می گردد مثل :Drop index Sn

تذکر : در SQL می توان کار طراحی و ایجاد بانک را به طور تدریجی انجام داد. یعنی ابتدا تعداد محدودی جدول ایجاد و بلافصله داده های بانک را وارد کرد. سپس به تدریج بانک را گسترش داد. در SQL برای کار با داده ها چهار دستور وجود دارد:DELETE,UPDATE.INSERT,SELECT

در ادامه این دستورات را شرح می دهیم و برای این کار از جداول بانک اطلاعاتی تهیه کنندگان و قطعات (جداول S وPوSP) استفاده می کنیم.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 و∏ و ∞ جبر رابطه ای را می توان با این دستور به راحتی انجام داد.

ساده ترین شکل این دستور به صورت زیراست :

Select نام فیلد ها
form نام جدول
where شرط جستجو

Select S#, status
form S خروجی →

wher city='C2'

S#	status
S1	20
S3	30
S4	20

مثال 12: می توان نام جدول را همراه نام فیلدها به کاربرد . دستورات روبرو دقیقاً مانند مثال قبل عمل می کنند.

```
select   S.S#,S.status
Form     S
Where city='C2'
```

هر چند که در مثال فوق استفاده از نام جدول اختیاری است ولی دربعضی 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 زیراست:

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

مثال 13: دستور روبرو: select P# form SP

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

(البته زیر هم نوشته می شوند) P1,P2,P3,P4,P5,P6,P1,P2,P2,P2,P4,P5

مثال 14: دستور روبرو: SELECT Distinct P# form SP

به علت استفاده از Distinct مقادیر ستون #P از جدولSP را با حذف تکراری ها می دهد. یعنی خروجی به صورت زیر می شود:

(البته زیر هم نوشته می شوند) P1,P2,P3,P4,P5,P6

اگر به جای Disinctاز عبارت ALL استفاده شود آنگاه تکراری ها نوشته می شوند (البته ALL حالت پیش فرض است)

مثال 15: شماره تمام قطعات و وزن هریک را بر حسب گرم بدهید. فرض کنید وزن ها بر حسب پوند در جدول P باشند.

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

```
Form   P
```

خروجی ⟹

(هر پوند 454 گرم می باشد)

P#				
P1	weight	in	gram	= 5448
P2	weight	in	gram	= 7718
P3	weight	in	gram	= 7718
P4	weight	in	gram	= 6356
P5	weight	in	gram	= 5448
P5	weight	in	gram	= 8626

مثال 16: دستور زیر کل جدول S را چاپ می کند:

```
Slect     *
Form     S
```

وجود * به معنای تمام ستون ها می باشد . دستور فوق معادل دستور زیر است:

```
Select S# ,Sname,status,city          form S
```

مثال17 : شماره تهیه کنندگانی را بیابید که ساکن C2 بوده و وضعیت آنها از 20 بیشتر باشد:

```
    Select S#
Form S
Where   city='S2' and   status > 20
```

خروجی:

S#	
S3	30

مثال 18 : شماره و وضعیت تهیه کنندگان ساکن C2 را براساس نظم نزولی مقادیر وضعیت بدهید.

```
    select S# , status
Form  S
Where   city='C2'
Order  by status  DESC
```

خروجی:

S#	status
S3	30
S1	20
S4	20

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

```
 Select   S#   form   S  order     by     city
```

غلط ⟹

تذکر : می توان به جای نام ستون ، شماره ستون را نوشت. ستون ها از چپ به راست از یک شماره گذاری می شوند. این کار امکان می دهد تا نتیجه پرس وجو براساس مقادیر یک ستون محاسبه شده که فاقد اسم است،منظم گردد.

مثال 19:

```
Select   P#, weight  *454
Form  P
 order   by    2, P#
```

عدد 2 یعنی ستون دوم جدول جواب

P#	.
P1	5448
P2	5448
P3	6353
P4	7718
P5	7718
P6	8626

توجه کنید که خروجی ابتدا براساس ستون دوم یعنی وزن مرتب می شود و اگر دو سطر وزن یکسانی داشته باشند آنگاه براساس P# مرتب می شود. به سطر 1و2و همچنین 4 و 5 توجه کنید.

عملگر between و in در select

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

مثال 20: خروجی این دستور چیست؟

```
Select  P# , color  , weight
form     P
where     weight    between   16  and   19
```

P#	color	weight
P2	Green	17
P3	Blue	17
P6	Red	19

می توان از دستور not between نیز استفاده کرد.

مثال21: خروجی این دستور چیست؟

```
Select P#   , weight
```

P#	weight

```
Form P
Where weight  in  (12 , 16  ,17)
```

P1	12
P2	17
P3	17
P4	12

می توان از not in نیز استفاده کرد.

عملگر like در sehect

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

مثال 22: مشخصات قطعاتی را بیابید که اسم آنها با حرف C شروع شده باشد.

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

P#	Pname	color	weight	city
P5	Cam	Bule	12	C3
P6	Cog	Red	19	C2

مثال 23: دستور زیرنام قطعاتی را می دهد که 3 حرفی بوده و با حرف C شروع می شوند:

```
Select  pname   form P  where pname   like   'C_ _ '
```

مثال 24: دستور زیر نام قطعاتی را می دهد که حاوی کارکتر 'E' نمی باشد :

```
Select pname   form P where not like '%E%'
```

مثال 25: :"_ A%B" like اسامی را می دهد که حرف دوم آنها A بوده ومختوم به B می باشند.

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 26: select S# form S where statuse is NULL

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

پرس و جو های مبتنی بر پیوند جدول ها

پیوند نوعی از پرس وجو است که طی آن عمل بازیابی از بیپ از یک جدول انجام می پذیرد.

مثال 27:خروجی این دستور چیست؟

```
Select   S.S# , S.city , P.P#, P.city
Form    S,P  where  S.city = P.city
```

خروجی در زیر ترسیم شده است.

برای اجتناب از ابهام ، دو ستون city، به صورت S.city وP.city باید نوشنه شوند. ولی S.S# را می توانستیم 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 28:

```
Select   S.*, P.*
Form S,P
```

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

دستور فوق تمام فیلدهای دو جدول را در صورت برقراری شرط می نویسید.خروجی آن را ترسیم کنید.

عمل پیوند به شرط تساوی طبق تعریف باید جدولی را بدهید حاوی دو ستون یکسان. اگر یکی از دو ستون را از جدول جواب حذف کنیم ، جدول حاصله را پیوند طبیعی می نامیم.

تذکر : در بعضی از نسخه های SQL می توان تا 16 جدول را با یکدیگر پیوند داد.

نکته: دستور Select S.*, P.* Form S,P عمل ضرب کارتزین دو رابطه S و P را در SQL انجام می دهد.

مثال 29: تمام جفت شهر های را بیابید که تهیه کننده ساکن شهر اول قطعه ای انبار شده در شهر دوم را تهیه کرده باشد.مثلاًS1 قطعه P2 را تهیه می کند.S1 ساکن شهر C3 است.بنابراین جفت شهر (C2,C3) یک سطر از جواب است:

```
Select Distinct S.city  , P.city
Form 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 می باشد.

مثال 30: (پیوند یک جدول با خودش) تمام جفت شماره تهیه کنندگانی را بیابید که ازیک شهر باشند.

```
Select   First.S#  , Second.S#
Form   S Firest,S Second
Shere Firest.city = Second.city
```

Firest.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 نسخه مجزا از S وجود دارد با نام های اختیاری Firest وSecond .البته در سطح فیزیکی دو نسخه از جدول S ایجاد نمی شود. در بعضی از نسخه های SQL خط Form به صورت زیر (با AS) نوشته می شود:

```
Form  S  AS   Firest,  S AS  Second
```

بنابراین AS هم برای تغییر نام ستون های جدول جواب و هم برای تغییر نام خود جدول ها استفاده می شود. البته حوزه عملکرد این تغییر نام فقط در همان دستور است.

مثال 31: برای حذف زوائد در مثال قبلی باید دستور زیر را بنویسیم:

```
Select  Firest.S#, Second.s#
Form S  Firest,  S second
Where   Firest,   S   Second
AND  Firest.S#  < Second.S#
```

Firest.S#	Second.S#
S1	S3
S1	S4
S3	S4

دستورات select متداخل

مثال 32: نام تهیه کنندگانی را بیابید که قطعه P2 را تهیه می کنند.

این پرس وجو را می توانیم به کمک عملیات پیوند تنظیم کنیم:

```
Select  S.sname
Form  S.SP
Shere    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  form s
Shere  s# in (select S#  form  SP
                Where  P# ='P2')
```

سیستم ابتدا پرس وجو ی درونی را انجام می دهد . و پاسخ این پرس وجو مجموعه { S1,S2,S3,S4} می شود بنااین دستور فوق معادل دستور تک سطحی زیر است:

```
Select sname  form  S
Where s#  in('s1', 's2' , 's3' , 's4' )
```

تذکر : از نظر کارائی روش پیوند بهتر از پرسو جو 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 33: شماره تهیه کنندگانی را بیابید، که در همان شهری ساکن باشند که تهیه کننده S1 ساکن است.

```
Select S# form  S
Where  city = (select  city  form s
               Where  s#='s1')
```

S#
S1
S3
S4

به پرسشی که تعدادی پرسش فرعی در درون خود داشته باشد،پرسش تودر تو (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

مثال 34: کل تعداد تهیه شده از قطعه P2 را بدهید.

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

جواب
1000

مثال 35: شماره تهیه کنندگانی را بدهید که مقدار وضعیت آنها کوچکتر از مقدار ماکزیمم وضعیت باشد.

```
Select  S#  form  S
Shere status <  (select MAX(status)  form S)
```

S#
S1
S2
S4

مثال 36:

```
Select  Min(Qty) AS MIN_QTY
Form    SP
```

با AS می توان نام جدید به ستون خروجی داد.

تذکر: در صورت لزوم توابع فوق را می توان با کلمه Distinct نیز به کار برد. در این حالت داده های تکراری در نظر گرفته نمی شوند (برای MAX وMIN داده های تکراری اهمیتی ندارند). مقادیر NULL قبل از اجرای این توابع حذف شده و روی این توابع تأثیری ندارند. مثلاً در محاسبه مقدار تابع AVG مقادیر NULL محاسبه تعداد داده ها (مخرج کسر) اثری ندارد.

مثال 37: تعداد تهیه کنندگان شهر C2 چند تاست؟

```
Select count  (S#)  form S
Where City='C2'
```

خروجی :3

مثال 38: تعداد شهر های موجود در جدول P چند تاست؟

در این مثال باید از شمارش تکراری جلوگیری کرد.(با AS نام ستون خروجی را تغییر داده ایم)

```
Select Count (Distinct city)  AS CT#
Form  P
```

خروجی:

CT#
3

نکته : تابع ویژه (*)Count برای شمارش سطر های جدول است. در این تابع نمی توان از Distinct استفاده کرد و این تابع سطر های NULL را نیز می شمارد. اگر جدول تهی باشد،این تابع صفر برمی گرداند.(برخلاف سایر توابع که NULL بر می گردانند)

مثال 39: مشخص سازید چند تهیه کننده P2 را تهیه کرده اند؟

```
Select count(*) form SP
Where  P#+'P2'
```

جواب: 4

Group By وHaving در select

مثال 40: کل مقدار تهیه شده از هر قطعه را در جدول جواب بدهید(همراه با شماره هر قطعه)

جواب:

```
Select  P#, SUM(Qty)
Form SP
Group By P#
```

P#	
P1	600
P2	1000
P3	400
P4	500
P5	500
P6	100

مثال 41: برای هر قطعه تهیه شده ،شماره قطعه ،کل تعداد و ماکزیمم تعداد تهیه شده از آن را بدون در نظر گرفتنs1 بدهید.

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

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

تذکر : صفتی که گروه بندی روی آن انجام می شود حتماً باید در خروجی بیاید. بخش Group by جزو آخرین بخش های یک دستور است و فقط having وorder by می تواند بعد از آن بیاید.

Havingهمواره با group by استفاده می شود. نقش having در گروه مانند نقش where در سطر است. به عبارت دیگر از having برای در نظر گرفتن گروه ها استفاده می شود،همانطور که از where برای در نظر نگرفتن سطر هایی در جدول جواب استفاده می گردد.

مثال 42: شماره قطعه تمام قطعاتی که توسط بیش از یک تهیه کننه تهیه شده اند را بیابید:

```
Select   P#   form
Group By P#
Having   count (*) > 1
```

P#
P1
P2
P4
P5

تذکر : Group Byو having افزوننه هستند یعنی هر پرسشی که با این دو تنظیم شوند، با امکانات دیگر موجود در SQL نیز قابل تنظیم است.

عملگر های مجموعه ای در SQL

عملگر های مجموعه ای تعریف شده در جبر رابطه ای در SQL پیاده سازی شده اند. عمل اجتماع با دستور UNION ، عمل اشتراک با INTERSECT و عمل تفاضل با EXCEPT پیاده سازی شده است. در اینجا هم باید همتایی داده ها دعایت شود. یعنی باید تعداد ستون ها و همچنین دامنه های ستون های دو جدول، با هم برابر باشند.

در SQL عملگر تعلق به صورت IN که قبلاً بیان شد ،پیاده سازی شده است. به علاوه عملگر دیگری به نما CONTALNS وجود دارد که مشابه عملگر زیر مجموعه (⊆) عمل می کند. مجموعه سمت راست زیر مجموعه سمت چپ می باشد یعنی:

A ⊆B ⟹ B CONTAINS A

دستور contains غالباً نیاز به گروه بندی داده ها (Group By) دارد که جلوتر کاربرد آن را شرح می دهیم.

مثال 43: شماره قطعاتی را بیابید که یا وزن آنها بیش از 16 باشد یا توسط S2 تهیه شده باشند یا هر دو شرط را دارا باشند.

```
Select  P# form P
Where weight > 16
UONION
      SELECT  p# Form  SP
Where S# = S2
```

P#	
P2	مربوط به قسمت اول
P3	
P6	
P1	مربوط به قسمت دوم

عناصر تکراری فقط یکبار در خروجی نوته می شوند مگر اینکه به جای UONION عبارت UONION ALL را بنوسیم که در اینصورت در مثال فوق P2 دوبار در خروجی ظاهر می شود.

Select در Exists

فرم کلی آن به صورت (exists(select * Form ... است. چنین عبارتی به مقدار « درست » ارزیلبی می شود، اگر مجموعه حاصل از ارزیابی پرس وجوی داخلی .. select * formتهی نباشد. در واقع هر پرس وجوئی که با استفاده از IN قابل تنظم باشد با استفاده از existsنیز قابل تنظیم است ولی عکس این معنا درست نیست.
Exists وجود سطر و not exists عدم وجود سطر را بررسی می کند.

عملگرهای ALL وANY

عملگر ALL برای مقایسه «همه مقادیر » و عملگر ANY (که در بعضی از نسخه های SQL نام SOME دارد) برای « هر یک از مقادیر » استفاده می شوند.نتیجه هر دو عملگر TRUE یا FALSE است.
این دو عملگر را می توان با توابع و عملگر های دیگر معادل سازی کرد.
مثال 45: نام قطعاتی را بدهید که وزن آنها عددی فرد ما بین 10 تا 20 باشد.

```
Select   Pname  form P
Where weight = ANY (11 ,13, 15,17,19 )
```

خروجی:

Pname
Bolt
Screw
Cog

تذکر ANY= معادل in می باشد.

مثال 46: نام تهیه کنندگانی را بدهید که وضعیت آنها از همه ساکنان شهر C2 بیشتر باشد.

```
Select   Sname   form   S
Where   status   >  SLL  (Select  status   form  S
                          Where  city = 'C2')
```

این دستور برای جداول داده شده در آخر این فصل خروجی ندارد.

پرس وجو با قید «همه»

معادل عملگر ÷ جبر رابطه ای در SQL وجود ندارد .به چند روش می توان تقسیم را در SQL معادل سازی کرد. ساده ترین این روش ها استفاده از تابع COUNT است. در واقع حالتها را می شماریم تا ببینیم «همه» آنها رخ داده اند یا خیر.

مثال 47: 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 می شماریم و با تعداد قطعات موجود در جدول Pمقایسه می کنیم.

```
Select S# form SP
Group Count (P# ) = (select Count (P#)  form P ) ;
```

دستور INSERT

این دستور دو فرم کلی زیر را دارد:

فرم اول	فرم دوم
Insert Into <نام جدول>[(,1 فیلد [, 2 فیلد...)] Valus (1ثابت [,2 ثابت ...)	Insert Into <نام جدول>[(,1 فیلد [, 2 فیلد...)] <جستجو>

در فرم اول یک سطر با مقادیر مشخص برای فیلد ها ، در یک جدول درج می شود. ثابت iام در لیست ثابتها متناظر با فیلد iام در لیست فیلدها می باشد.

در فرم دوم، پرس وجویی فرعی ارزیابی می شود و جواب آن که معمولاً چندین سطر است در جدول درج می شود.در هر دو شکل نوشتن لیست فیلدها به معنای این است که تمام فیلد های جدول مورد نظر است.

مثال48: قطعه P7 (شهر C1 ، وزن 24 ، اسم و رنگ در حال حاضر ناشناخته) را در جدول Pدرج کنید.

```
Insert Into P (P# ,City, weight)
      Values ('P7.'C1' ,24)
```

مقدار فیلدهای اسم و رنگ برابر null می شود.

تذکر : نظم از چپ به راست فیلد ها در جلوی دستور Insert لزوماً مشابه Crest نیست.

مثال 49: 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 Creat .

تذکر: بهتر است همیشه نام فیلدها ذکر گردد.

مثال 50: برای هر قطعه تهیه شده ،شماره قطعه و کل تعداد تهیه شده ازآن بدست آورده و نتیجه را در بانک ذخیره کنید.

```
Creat Table Temp
(P# CHAR(6)  NOT NULL,
TOTQTY Int,

PRRIMARY KEY(P#))

Insert Into Temp (P#, TOTQTY)
Select P#,SUM(Qty)
Form SP
Group By P#
```

Temp

P#	TOTQTY
P1	600
P2	1000
P3	400
P4	500
P5	500
P6	100

نتیجه INSERT در جدولtemp ذخیره می شود. کاربر می تواند با این جدول کار کند وهر گاه که خواست آن را با دستور droup table temp حذف کند.

دستورUpdate

شکل کلی این دستور به صورت زیر است:

```
Update <نام جدول>
set <فیلد 1> =< مقدار 1 > [ , <فیلد 2 >=<مقدار 2>]    ...
```

[where < شرط >]

مثال51: رنگ قطعه P2 را به زرد تغییر داده به وزن آن 5 واحد 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 در چند رکورد صورت گیرد.

مثال 52: تعداد را در محموله های تهیه کنندگان ساکن C3 صفر کنید.

```
Update SP Aet Qty= 0
Where  S# in(select S#  Form S where city ='C3')
```

تذکر: ممکن است لازم باشد بهنگام شازی در چند جدول همزمان صورت گیرد.

مثال53: شماره S1را در جدول Sبه 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 Delet

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

Delet Form < نام جدول > [where < شرط >]

مثال54: تهیه کننده S5 را حذف کنید.

```
Delet Form S where S#='S5'
```

مثال 55: تمام محموله هایی که تعداد آنها از 300 بیشتر است حذف کنید.

```
Delet Form SP where Qty> 300
```

مثال56: تمام محموله ها را حذف کنید (یعنی تمام سطرهای جدول SP را)

```
Delet Form SP
```

جدول SP هنوز وجود دارد ولی خالی است.

مثال 57: تمام محموله های تهیه کنندگان ساکن C3 را حذف کنید.

```
Delet Form SP
Where S# in (select S# form S where city =' C3')
```

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

مثال 58: حذف تهیه کنندگانی که وضعیت آنها از میانگین وضعیتها کمتراست.

```
Delet Form S
 Where status < (select SVG (Status)  Form 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 creat View با فرم کلی زیر استفاده می شود:

CREATE VIEW <نام دید>[(فیلد 1, [, فیلد 2]...)]

AS < جستجو >

مثال 59:

```
Creat View Redpart  (P#, WT , City)
   As Select  P# , weight  , City
Fort P where Color ='Red'
```

Redpart

P#	WT	City
P1	12	C2
P4	14	C2
P6	19	C2

دستور فوق یک دید به نام Redpartپدید می آورد.
مثال 60:

```
Creat View PQ(P# , TOTQTY)
AS Select P# , SUM(QTY)
Form  SP
Goroup By P#
```

دید PQ

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 می کنیم 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 می کند که تمامی دستورات گفته شده را می توان برای آن به کار برد.
مثال61:

```
Reat View  GS
AS Select  S# , Stautus  , City
  Form   S  where Status  >  15
```

دید GS

S#	STATUS	City
S1	20	C2
S3	30	C2
S4	20	C2
S5	30	C1

پس از ساختن دید فوق اگر دستور زیر را صادر کنیم خروجی ذیل ظاهر می گردد:

```
Select *
Form  GS where  Ciry <> 'C2'
```

S#	statis	City
S5	30	C1

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 کننه دیدگاه (که در کاتالوگ ذخیره شده) قرار می گیرد.
تذکر: می توان گفت که در SQL سه نوع جدول وجود دارد

1 - جداول اصلی (base tables) که با دستور Creat table ساخته می شوند که می توان به آنها دسترسی داشت و هر گونه تغییری در آنها اعمال کرد.
2 - جداول میانی(intermediate table) 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 را محاسبه و به طور موقت ذخیره می گردد و سپس به کمک آن کل پرس وجو انجام می گیرد. جداول میانی قابل دسترسی نبوده و بدین دلیل قابل تغییر نمی باشند.
3 - جداول مجازی (Views) که وجود خارجی نداشته و می توان به 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 Views از دید کاربر مستقیم ولی از دید سیستم غیر مستقیم است یعنی سیستم هر گونه استخراج اطلاعات را از جداول اصلی انجام می دهد.

نکته : مشکل اصلی در جداول اصلی به روز در آوردن آنها می باشد. در صورتی می توان جدول مجازی را به هنگام کرد که تغییرات مورد نظر روی جداول اصلی بدون اشکال وابهام باشد.مثال زیراین موضوع را نشان می دهد.

مثال 62: رکورد جدید 200 وP8 را به جدول مجازی PQ اضافه کنید.

حل : این دستورات امکان پذیر نیست .چرا که جدول مجازی PQ بخشی از SP می باشد و در صورت درج در زیر #S باید عبارت NULL قرار بگیرد.حال آنکه #S بخشی از کلید اصلی SP است و نمی تواند NULL باشد.

بررسی تمام شرایطی که به هنگام سازی جدول مجازی را مجاز می دارد مشکل است. بعضی از نسخه های SQL نیز به طور کلی این عمل را انجام نمی دهند. 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 اصلی ساخته شده باشد.

1 - جدول مجازی فقط از یک جدول اصلی ساخته شده باشد.

2 - در ساختن جدول مجازی از Select Distinct استفاده نشده باشد.

3 - اگر جدول مجازی روی جدول اصلی A ساخته شده آنگاه A در زیر دستور نیامده باشد. به چنین جداولی ، جداول قابل تغییر (Updateable) می گویند.

پیوند در SQL

در SQL1 عملیات پیوند مستقیماً پشتیبانی نمی شد و با استفاده از شرط در جلوی where یا توسط select متداخل شبیه سازی می گردید. ولی در SQL 2 دستوراتی اضافه شد تا انواع پیوند را مستقیماً پشتیبانی کنند.

1 - ضرب دکارتی (×) جبر رابطه ای با دستور CROSS JOIN پیاده سازی شده است.

مثال 63: S CROSS JOIN SP

دستور فوق ضرب دکارتی دو جدول S وSP را می دهد.

2 - پیوند شرط(X_θ) جبر رابطه ای با دتور ... ON ... JOIN پیاده سازی شده است.

مثال 64:

```
S JOIN SP ON
  S.S# = SP.S# AND Qty > 200 ,
```

3- پیوند طبیعی (∞) جبر رابطه ای با دستور NATURAL JOIN پیاده سازی شده است.

مثال 65: S Natural Join SP

تذکر: از آنجا که خروجی این دستورات « رابطه » است می توان آنها را در بخش FORM از دستور select استفاده کرد.

مثال 66:

```
Select aname, Qty
From S Natural Join SP
```

4-پیوندOUTER.می دانیم که اگر سطری از جدول A با هیچ سطری از جدول B فیلد همنام یکسان نداشته باشد ، این سطر A در A ∞B ظاهر نمی شود.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 شناسه ویژه دارد. شناسه PUBLIC همه کاربران را شامل می شود. چهار نوع امتیاز برای دستیابی به جداول عبارتند از : Update, Delet , 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 می باشد.مثلاً رئیس یک سازمان باید بتواند روی دستمزد کارمندان خود قواعد محدودیت را اعمال کند (پس باید امتیاز refereneces را داشته باشد) ولی دیگران چنین 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 Grant Option باعث می شود کاربر مورد نظر بتواند این امتیازها را با دستور Grant دیگری به سایر کاربران نیز واگذار کند.

مثال67:

Grant 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 >

مثال 68:

Revoke insert ON S from All

تذکر : امتیاز Insert (X)، Update(X) و References(X) فقط برای ستون X می باشند.

تذکر: با دستور Revoke Grant Option ON می توان فقط حق واگذاری امتیاز به غیر را باز پس گرفت ولی خود امتیاز همچنان به قوت خود باقی باشد.

تذکر: می توان دستوراتSQL را به 3 دسته زیر تقسیم کرد:

1 - دستورات DDL (مثل Creat domain, drop table , Alter table , Creat tabel، Alter domain , drap domain ، Creat index ، drop index)

2 - دستورات DML (مثل delat , Update, insert,Select)

3 - دستورات DSL(مثل revoke , grant)

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 برچسب «رابطه ای » را داشته باشند یا خیر. امروزه محصولی در بازار نیست که تمام جزئیات مدل رابطه ای را پشتیبانی کند.

جداول S وPوSP

جدول SP	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 S	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 P	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	C2
	P6	Cog	Red	19	C2

فصل هفتم

وابستگی تابعی (FD=Functional Dependency)

صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد و می نویسیم $R.X \rightarrow R.Y$ 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 و Y می توانند صفات مرکب باشند. اصطلاحاً می گوئیم صفت خاصه X صفت خاصه Y را تعیین می کند.

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 مجموعه های X وابستگی تابعی نداشته باشد. در این تعریف صفت X را مرکب فرض کرده ایم اگر صفت خاصه X مرکب نباشد وابستگی حتماً کامل خواهد بود.

آقای آرمسترانگ در سال 1974 ثابت کرد که با اعمال مکرر سه قاعده زیر می توان به تمام وابستگی های منتج دست یافت و هیچ وابستگی اضافی نیز تولید نمی شود.

1 - بازتاب (reflexivity) :اگر B زیر مجموعه A باشد آنگاه $A \rightarrow B$

2 - افزایش یا بسط پذیری (augmentation) :اگر $A \rightarrow B$ و C صفت باشد $AC \rightarrow BC$

3 - انتقال یا تعدی (transitivity) : اگر $A \rightarrow B$ و $B \rightarrow C$ آنگاه $A \rightarrow C$

هر چند قاعده های فوق برای استخراج F^+ کفایت می کرد ولی اعمال آنها مشکل بود. بعد ها دیگران قداعد دیگری را بیان کردند که کار را سهولت بخشید و مهمترین آنها عبارتند از :

4 - اجتماع (UNION): اگر $A \rightarrow B$ و $A \rightarrow C$ آنگاه $A \rightarrow BC$

5 - تجزیه (decomposition): اگر $A \rightarrow BC$ و $C \rightarrow D$ آنگاه $AC \rightarrow BD$

6 - ترکیب (Composition) : اگر $A \rightarrow B$ و $C \rightarrow D$ آنگاه $AC \rightarrow BD$

7 - خود تعیینی(self-determination) : $A \rightarrow A$

8 - شبه تعدی (paeudotransitivity): اگر $A \rightarrow B$ و $BC \rightarrow D$ آنگاه $AC \rightarrow D$

9 - اگر $A \rightarrow B$ و $BC \rightarrow D$ آنگاه $AC \rightarrow D$

10 -اتحاد کلی (General unification): اگر $A \rightarrow B$ و $C \rightarrow D$ آنگاه $A \cup (C-B) \rightarrow$

مثال :فرض کنید متغیر رابطه R با صفات A وB وCو DوEو FوFD های زیر موجودند:

$F=\{A \rightarrow BC \ , B \rightarrow E \ , CD \rightarrow EF\}$

آیا وابستگی $AD \rightarrow F$ برای R برقرار است یا خیر ؟

حل : بله چرا که :

$A \rightarrow B \Longrightarrow A \rightarrow C$

$A \rightarrow C \Longrightarrow AD \rightarrow CD$

$AD \rightarrow CD \ , CD \rightarrow EF \Longrightarrow AD \rightarrow EF$

$AD \rightarrow EF \rightarrow AD \rightarrow F$

مثال: فرض کنید داریم $R=(S,T,U,V,W)$ و $F=\{S \rightarrow T, V \rightarrow SW, T \rightarrow V\}$ وابستگی های تابعی دیگر را بدست آورید.

$S \rightarrow T \ , T \rightarrow U \Longrightarrow S \rightarrow U$

$V \rightarrow SW \Longrightarrow V \rightarrow S \ , V \rightarrow W$

$V \rightarrow S \ , S \rightarrow T \Longrightarrow V \rightarrow T$

$V \rightarrow S \ , S \rightarrow U \Longrightarrow V \rightarrow U$

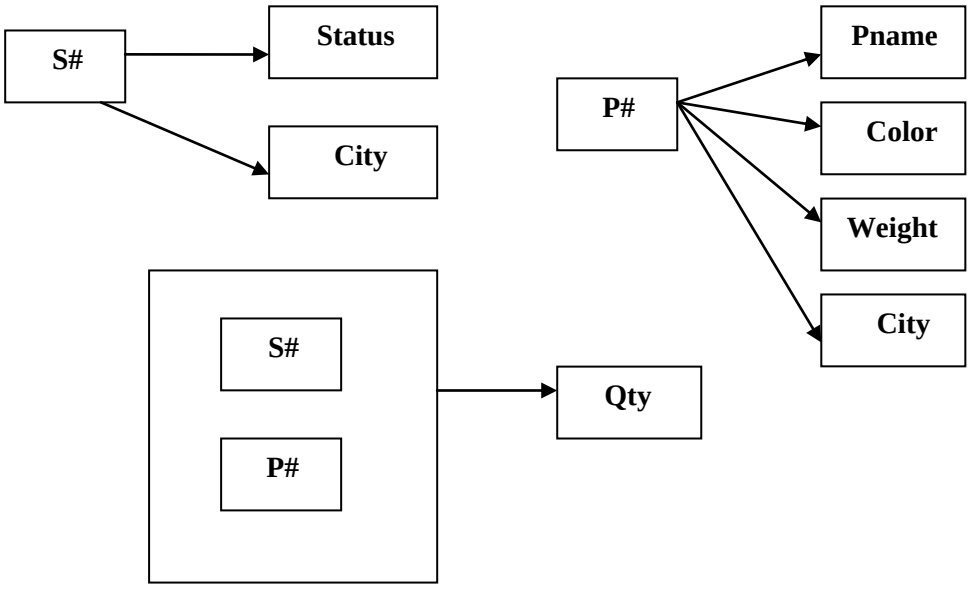
پس داریم: $V \rightarrow S, V \rightarrow T, V \rightarrow U \ , V \rightarrow W$

یعنی V کلید کاندید است.

نمودار وابستگی تابعی (FD Diagram)

به کمک این نمودار وابستگی های تابعی یک بانک ترسیم می شود.

مثال: نمودار وابستگی روابط S و P و SP به شکل زیر است (برای سادگی در جدول S فیلد Sname را حذف کرده ایم)



مجموعه وابستگی بهینه و مجموعه وابستگی کهینه (کاهش پذیر)

با اعمال قواعد آرمسترانگ وابستگی های زیادی به دست می اید که تعدادی از آنها اضافی و تکراری هستند. در زیر روشی را برای حذف اینگونه وابستگی های زائد و رسیدن به مجموعه وابسته بهینه ارائه می کنیم.

تعریف: دو مجموعه وابستگی تابعی F_1و F_2 معادل یا هم ارزهستند اگر مجموعه پوششی آنها برابر هم باشد یعنی $F_1^+=F_2^+$

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

1 - سمت راست هر وابستگی فقط یک صفت باشد.
2 - هر صفتی که F^+ را تغییر نمی دهد از سمت چپ حذف شود.
3 - وابستگی های تکراری و اضافی حذف شود.

به طور خلاصه باید گفت که برای یافتن وابستگی های تابعی در یک بانک ابتدا مجموعه پوششی وابستگی ها را تعیین کرده و سپس آنرا بهینه می کنیم.

مثال: در بانک اطلاعاتی زیر مجموعه وابستگی پوششی بهینه را بیابید.

$R=(u,v,w,x,y,z)$ $F=\{u\rightarrow xy,\ x\rightarrow y,\ xy\rightarrow zv\}$

حل :

$U\rightarrow xy, xy\rightarrow zv \Longrightarrow u\rightarrow zv \Longrightarrow u\rightarrow z, u\rightarrow v$
$U\rightarrow xy \Longrightarrow u\rightarrow x, u\rightarrow y$
$x\rightarrow y, xy\rightarrow zv \Longrightarrow x\rightarrow zv \Longrightarrow x\rightarrow z, x\rightarrow v$

پس :

$F_{opt}=\{u\rightarrow x, u\rightarrow y, x\rightarrow y, x\rightarrow z, x\rightarrow v, u\rightarrow z, u\rightarrow v\}$

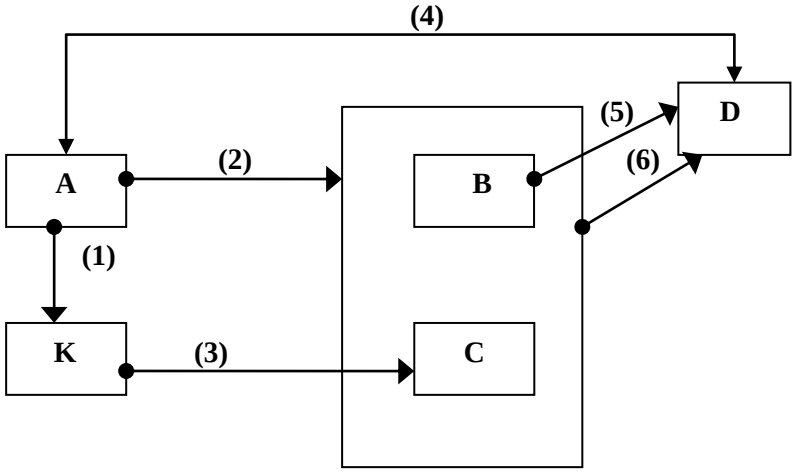
توجه کنید با توجه به خاصیت انتقال $u\rightarrow v, u\rightarrow z, u\rightarrow y$ بدست می آیند پس اگر از ما وابستگی کمینه (کهینه) را خواستند جواب به صورت زیر است:

$F_{opt}=\{u\rightarrow x,, x\rightarrow y, x\rightarrow z, x\rightarrow v\}$

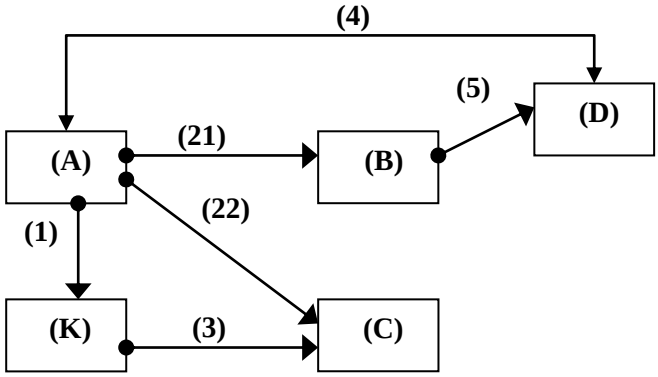
مثال: در یک رابطه وابستگی ها به صورت زیر است:

$A \rightarrow (B,C)$ $\quad$ $A \rightarrow D$ $\quad$ $A \rightarrow K$
$K \rightarrow C$ $\quad$ $B \rightarrow D$ $\quad$ $(B,C) \rightarrow D$

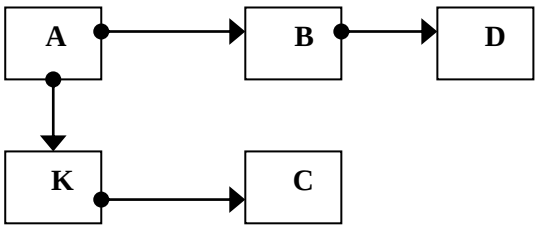
نمودار وابستگی را ترسیم کنید. سپس مجموعه کهینه (کمینه) این وابستگی ها را بدست آورده و نمودار آن را ترسیم کنید.



الف) $(B,C) \rightarrow D$ زائد است چون $(B,C) \rightarrow D$ $\quad$ $B \rightarrow D$ پس فلش (6) را حذف می کنیم.
ب) از $A \rightarrow (B,C)$ می توان نتیجه گرفت $A \rightarrow B$, $A \rightarrow C$ 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 شماره 4 زائد است چرا که $A \rightarrow D$ $\quad$ $A \rightarrow B, B \rightarrow D$
د) فلش شماره 22 زائد است چرا که $A \rightarrow C$ $\quad$ $A \rightarrow K, K \rightarrow C$
پس خلاصه ، شکل کمینه وابستگی به صورت زیر است:



همانطور که مشاهده می کنید در شکل کمینه وابستگی ها می بایست:

1 - سمت راست هر FD یک صفت باشد

2 - سمت چپ هر FD کاهش ناپذیر باشد.

3 - FD زائدی وجود نداشته باشد یعنی هیچ FD به کمک FDهای دیگر بدست نیاید.

نکته: برای هر مجموعه از FD ها، حداقل یک مجموعه هم ارز وجود دارد که کاهش ناپذیر است.

بستار (Closure) مجموعه ای از صفات

یعنی زیر مجموعه ای که حاوی تمام FD هایی است که مجمعه مشخص شده Z از صفات ،به عنوان بخش سمت آنها تعیین شده است.

مثال: اگر $R=(S,T,U,V,W)$ و $F=\{S\rightarrow T, V\rightarrow SW, T\rightarrow V\}$ باشد آنگاه:

الف) $\{S,V\}^+$ و ب) $\{V\}^+$ را بدست آورید.

حل الف)

$Z^+ = \{S,V\}$, $S \rightarrow T \Longrightarrow Z^+= \{S,V,T\}$

$Z^+ = \{S,V,T\}$, $V\rightarrow SW \Longrightarrow Z^+= \{S,V,T,W\}$

$Z^+ = \{S,V,T,W\}$, $T\rightarrow U \Longrightarrow Z^+= \{S,V,T,W,U\}$

$\{S,V\}^+=\{S,V,T,W,U\}$

با تکرار الگوریتم دیگر Z^+ تغییری نمی کند پس از آنجا که $\{S,V\}$ تمام صفاتR را می دهند پس $\{S,V\}$ یک سوپر کلید رابطه R می باشد.

ب)

$Z^+ = \{V\}$, $S\rightarrow T \Longrightarrow Z^+= \{V\}$

$Z^+ = \{V\}$, $V\rightarrow SW \Longrightarrow Z^+= \{V,S,W\}$

$Z^+ = \{V,S,W,T\}$,$T\rightarrow U \Longrightarrow Z^+= \{V,S,W\}$

حلقه do-while را دوباره اجرا می کنیم:

$Z^+ = \{V,S,W\}$, $S\rightarrow T \Longrightarrow Z^+= \{V,S,W,T\}$

$Z^+ = \{V,S,W,T\}$,$T\rightarrow U \Longrightarrow Z^+= \{V,S,W,T,U\}$

در نتیجه داریم : $\{V\}^+= \{V,S,W,T,U\}$

یعنی V نیز یک سوپر کلید است.با توجه به تعریف کلید کاندید در می یابیم که $\{S,V\}$ کلید کاندید نمی باشد وV کلید کاندید است.

بدست آوردن کلید های کاندید

برای یافتن کلید های کاندید بهتر است ابتدا F را بهینه کنیم سپس با توجه به نکات زیر کلید های کاندید را بدست می آوریم.

1 - هر کلید کاندید شامل مجموعه ای از صفتهایی است که در سمت چپ پیکانها می آیند.

2 - کلید کاندید باید کمینه باشد یعنی زیر مجموعه ای از آن خاصیت کلیدی نداشته باشد.

3 - ممکن است چند کلید کاندید وجود داشته باشد.

4 - کلیدهای کاندید ممکن است در یک یا چند صفت مشترک باشند.

مثال: اگر $F=\{S\rightarrow T, V\rightarrow SW, T\rightarrow V\}$ و $R=(S,T,U,V,W)$ باشد آنگاه کلید کاندید این رابطه چیست؟

حل :

$S\rightarrow T$, $T \rightarrow U \Longrightarrow S \rightarrow U$

$V\rightarrow SW \Longrightarrow V \rightarrow S$, $V \rightarrow W$

$F=\{S \rightarrow T, S \rightarrow U, T \rightarrow U, V\rightarrow S, V \rightarrow W\}$

پس

$V\rightarrow S$, $S\rightarrow T \Longrightarrow V\rightarrow T$

$V \rightarrow T$, $T \rightarrow U \Longrightarrow V \rightarrow U$

$F=\{S \rightarrow T, S \rightarrow U, T \rightarrow U, V \rightarrow S, V\rightarrow T, V \rightarrow W\}$

ازآنجا که V همه صفتهای دیگر را می دهد پس کلید کاندید است.

مثال: اگر $R=\{A,B,C,D,E,F,G\}$ و $F=\{AF \rightarrow BF, FC \rightarrow DF, F \rightarrow CD, D \rightarrow E, D \rightarrow E, C \rightarrow A\}$ باشد کلید کاندید این رابطه چیست؟

حل: ابتداسمت راست وابستگی ها را به یک صفت تبدیل می کنیم:

$F=\{AF\rightarrow B, AF \rightarrow E, FC \rightarrow D, FC\rightarrow E, F \rightarrow C, F \rightarrow D, D \rightarrow E, C \rightarrow A\}$

حال باید صفتهای اضافی را از سمت چپ حذف کنیم.

$F \rightarrow C, FC\rightarrow D \Longrightarrow F\rightarrow D$

$F \rightarrow C, FC\rightarrow E \Longrightarrow F\rightarrow E$

$F \rightarrow C, C\rightarrow A \Longrightarrow F\rightarrow A$

$F \rightarrow A, AF\rightarrow B \Longrightarrow F\rightarrow B$

$F \rightarrow A, AF \rightarrow E \Longrightarrow F\rightarrow E$

پس:

$F_{OPT}=\{F\rightarrow A, F\rightarrow B, F\rightarrow C, F \rightarrow D, F \rightarrow E, D \rightarrow E, C\rightarrow A\}$

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 F وG را نمی دهد یعنی در هر کلید کاندید این دو صفت لازم هستند.

مثال : در رابطه زیر همه کلیدهای کاندید را بدست آورید.

$R=\{U,V,W,X,Y,Z,O,P,Q\}$

$F= \{ U \rightarrow VXQ , UVP\rightarrow O , OQ \rightarrow YZ, UP \rightarrow XY\}$

حل:

$V \rightarrow VXQ \Longrightarrow U\rightarrow V, U\rightarrow X, U\rightarrow Q$

$OQ\rightarrow YZ \Longrightarrow OQ \rightarrow Y, OQ \rightarrow Z$

$U\rightarrow V, UVP\rightarrow O \Longrightarrow UP \rightarrow O$

$UP \rightarrow XY \Longrightarrow UP\rightarrow X, UP\rightarrow Y$

X U را قبلاً داشته ایم پس X UP زائد است.

پس $F_{opt}=\{U \rightarrow V, U \rightarrow X, U \rightarrow Q, UP \rightarrow O, OQ \rightarrow Y, OQ \rightarrow Z, UP \rightarrow Y\}$

حال مجموعه صفتهای وابسته به تمام مجموعه صفتهای چپ پیکان ها را می یابیم:

$\{U\}^+, U \rightarrow V, U \rightarrow X, U \rightarrow Q, \Longrightarrow \{U,V,X,Q\}$

$\{U,P\}^+, U \rightarrow V, U \rightarrow X, U \rightarrow Q, UP \rightarrow Y, OQ \rightarrow Z \Longrightarrow \{U,P,V,X,\rightarrow,O,Y,Z\}$

$\{O,Q\}^+, OQ \rightarrow Y, OQ \rightarrow Z \Longrightarrow \{O,Q, Y,Z\}$

نتیجه می گیریم که بیشترین صفتها را $\{U,P\}^+$ می دهد. 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 O وQ نمی توان به WوU وP رسید پس ابرکلید 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 هشتم

نرما ل سازی چیست؟

هنگام طراحی یک بانک اطلاعاتی رابطه ای ، این سؤال مهم مطرح می شودکه « با توجه به داده های عملیاتی و ارتباط بین موجودیتها ، چند جدول می بایست طراحی کرد؟ در هر جدول چه فیلد هایی باید قرار گیرد؟ رابطه جدول ها باید چگونه باشد؟

مشکلات

1 - افزونگی داده ها (Data redundancy) قبلاً بیان شد که افزونگی یعنی تکرار بی رویه داه ها

2 - بی نظمی (anomaly) وجود افزونگی در جدول باعث آنومالی در تغییر داده ها می شود.

3 - مقادیر تهی(NULL Value) با ادغام جداول گاهی اوقات مجبور خواهیم بود برای نشان دادن بعضی از اقلام اطلاعاتی از NULL استفاده کنیم.مقادیر تهی علاوه بر اینکه جای زیادی را اشغال می کنند مشکلاتی را نیز باعث می گردند .

مجموعه روشهائی که در طراحی بانک اطلاعاتی موجب کاهش افزونگی اطلاعات و آنومالی ها گردد و براساس آن روابط بین اقلام دادها اداره شوند ، نرمال سازی گفته می شوند.

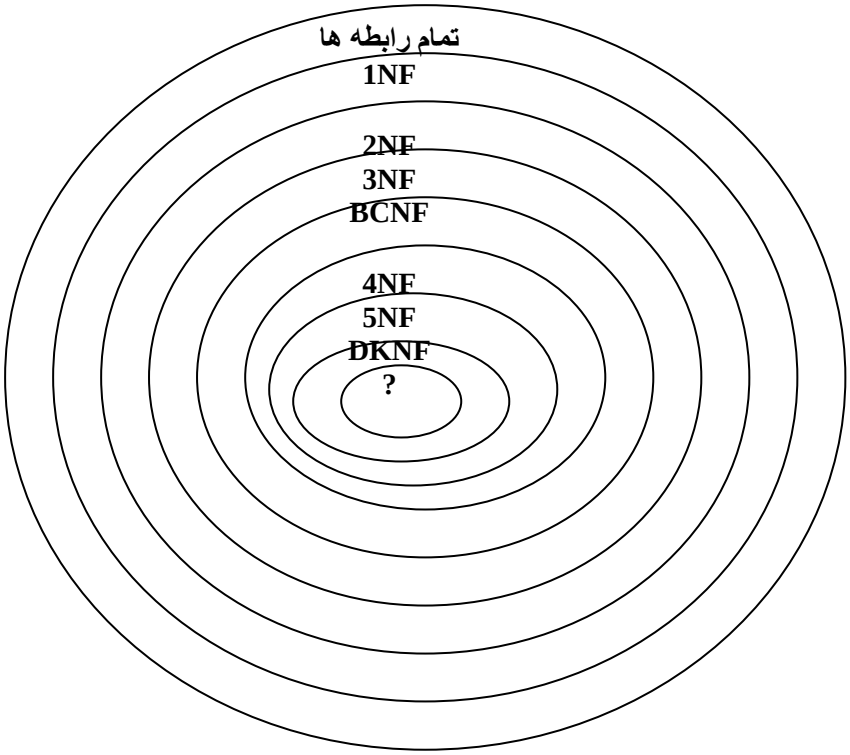
سطوح نرمال

کاد (واضع مدل رابطه ای) در ابتدا سه سطح نرمال 1NF، 2NF وسه3NF را تعریف کرد ولی بعداً دانشمندان دیگر، صورتهای دیگری را نیز معرفی کردند.

سطوح نرمال بودن عبارتند از:

1NF (Firest Normal From)

2NF	(Second Normal Form)
3NF	(Third Normal From)
BCNF	(Boyce/Codd Normal Form)
4NF	(Fourth Normal Form)
5NF	(Fifth Normal Form)
DKNF	(Domain-Key Normal Form)



فرم 1NF

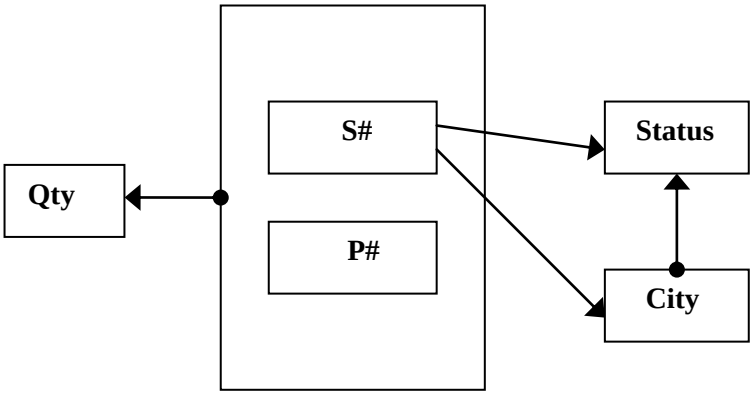
رابطه r به فرم 1NF (Firest Normal From) است اگر و فقط اگر تمام صفات خاصه آن روی میدان های اتومیک تعریف شده باشند به عبارتی دیگر صفتهای آن از دامنه تودرتو (nested domain) نباشند. یعنی صفت ترکیبی نداشته باشیم. مثلاً اگر جدولی شامل فیلد تازیخ باشد که خود فیلد تاریخ از سه فیلد کوچکتر (سال – ما ه – روز) تشکیل شده باشد آنگاه جدول 1NF نیست. در واقع هر نرمالی 1NFاست.

فرض کنید به جای دو جدول مجزای S وSP یک جدول به نامFirest به صورت زیر تعریف می کردیم:

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 → 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.
همینطورکه مشاهده می شود اولین مشکل این فرم افزونگی اطلاعات است.

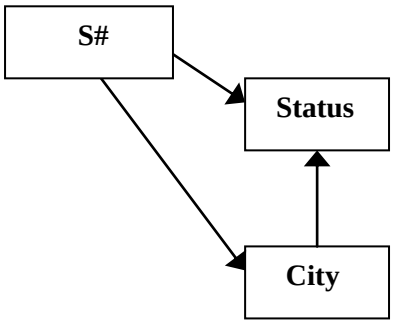
فرم 2NF

رابطه R در صورت دوم نرمال است اگر 1NF باشد و هر صفت خاصه غیر کلید با کلید اصلی وابستگی تابعی کامل داشته باشد به عبارت دیگر هر صفت خاصه غیر کلید با کلید اصلی به طور کاهش ناپذیر وابسته باشد.

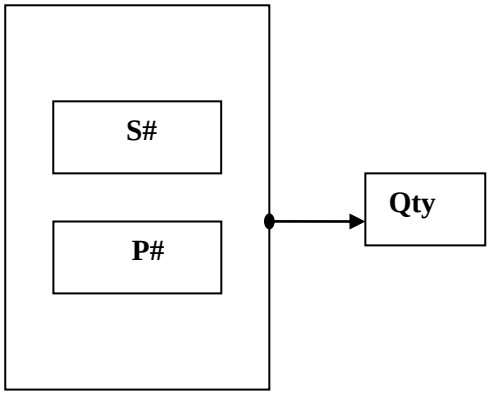
رابطه Firest در قبل را به دو رابطه زیر تجزیه می کنیم :

$$\text{First(S\#, status, city,P\#, Qty)} \rightarrow \begin{cases} \text{Second (S\#,status, city)} \\ \text{SP(S\#,P\#,Qty)} \end{cases}$$

نمودار وابستگی این دو رابطه به شکل زیر خواهد بود:



Second



SP

فرم 3NF

رابطه R در سطح 3NF است اگر و فقط صفات خاصه غیر کلید (در صورت وجود) :
الف) متقابلاً به یکدیگر ناوابسته باشند.
ب) با کلید اصلی رابطه R ، وابستگی تابعی کامل داشته باشند.

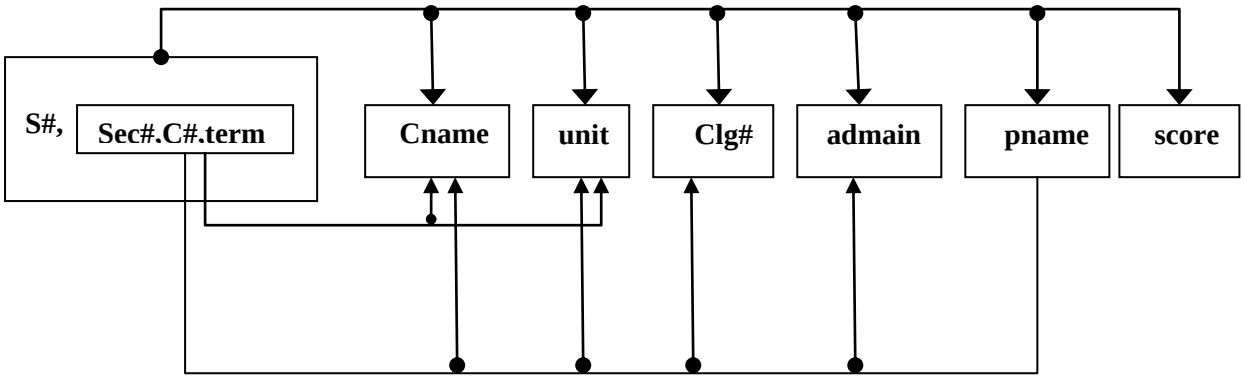
قضیه هیث (Heath)

رابطه R(A,B,C) که در آن A و B وc سه مجموعه از صفات هستند مفروض است. اگر $A \rightarrow B$ آنگاه می توان R را به دو رابطه R1(A,B) و R2(A,C) تجزیه کرد و این تجزیه ، مطلوب (خوب) است.
تذکر: براساس ضوابط رسانن اگر در رابطه R(A,B,C) وابستگیهای $A \rightarrow B$ و $B \rightarrow C$ برقرار باشد.
در این صورت تجزیه مطلوب به صورت زیر است:

$$R(A,B,C) \Longrightarrow R1(A,B)\ ,\ R2\ (B,C)$$

الگوریتم تبدیل 1NF به 2NF و2NF به 3NF

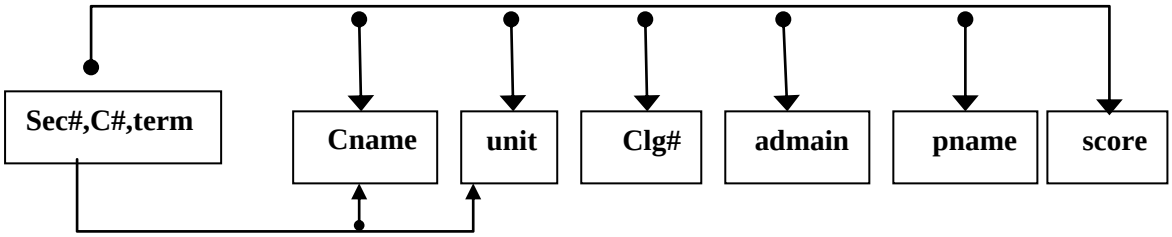
الگوریتم تبدیل جدول 1NF به چند جدول2NF به صورت زیر است:
1 - هر بخش از کلید اصلی را که صفت وابسته دارد، با آن صفت ها کنار هم قرار می دهیم.
2 - کل کلید اصلی را با صفتهای باقی مانده کنار هم قرار می دهیم.
3 - سایر وابستگی ها را ترسیم می کنیم.
تذکر : اگر وابستگی به بخشی از کلید به صورت تودرتو بود آنگاه الگوریتم فوق را باید تکرار کنیم.
مثال : فرض کنید نمودار وابستگی تابعی بانک اطلاعات ثبت نام دانشگاه به شکل زیر باشد.



S# شمار دانشجو ،Sec# شماره گروه ، C# شماره درس ، term شماره ترم (1 یا 2) ، cname نام درس ، unit تعداد واحد ، Clg# شماره دانشکده ، admin رئیس دانشکده ، pname نام استاد و score نمره می باشد. کلید کاندید رابطه (S#,Sec#,C#,term) می باشد.

رابطه فوق 1NF بوده و 2NF نیست برای تبدیل آن به 2NF الگوریتم فوق را اعمال میکنیم:

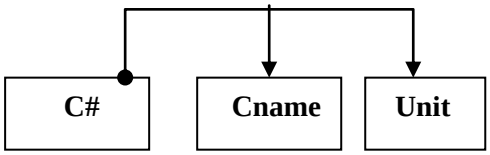
R1



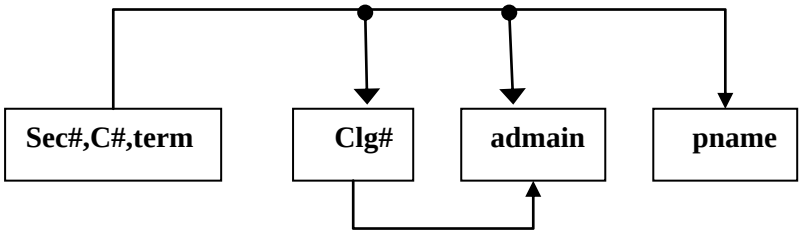
R2

ولی خود R1 هنوز در سطح 2NF نمی باشد و باید آن را با تجزیه به دو رابطه 2NF تجزیه کنیم پس الگوریتم را دوباره برای R1 اعمال می کنیم آن را به R11 وr12 تجزیه می کنیم.

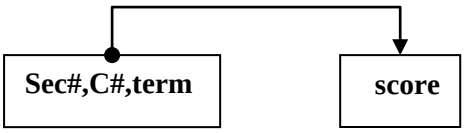
R11



R12



R2

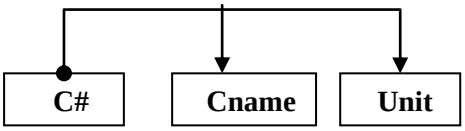


حال هر سه جدول فوق در سطح 2NF می باشند . توجه کنید که معمولاً وابستگی به کلید اصلی را به صورت فلشهای بالای مستطیل ترسیم می کنیم.
حال الگوریتم تبدیل جدول 2NF به 2NF را بیان می کنیم:

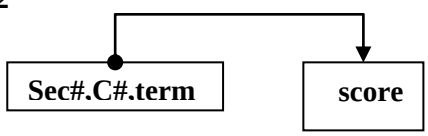
1 - صفتهایی را که وابستگی انتقالی ایجاد کردهاند، با وابسته های آنها کنار هم قرار می دهیم.
2 - کلید اصلی را با صفتهای کلیدی را به عنوان کلید خارجی در 2 تکرار می کنیم.
3 - صفتهای کلیدی را به عنوان کلید خارجی در 2 تکرار می کنیم.

مثال: جداول مثال قبل را به سطح 3NF ببرید.
حل: جداول R11و R2 در سطح 3NF هستند. جولR12 را طبق الگوریتم فوق به جداول R121 و R122 تجزیه می کنیم:

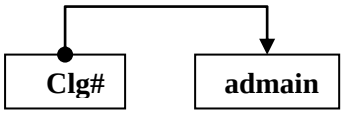
R11



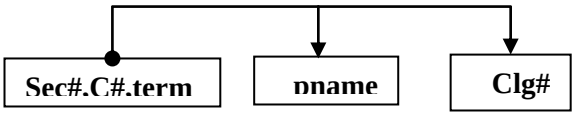
R2



R121



R122



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 می باشد. هر چند با 3NF کردن رابطه ها ، افزونگی به مقدار قابل توجه ای کاهش می یابد 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 پیوند در عمل تا حدودی باید افزونگی را پذیرفت.

به نظر دیت «رابطه ها را اقلاً 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: 3NF در مواردی که هر سه شرط زیر برقرار باشند ممکن است مشکل داشته باشد:

1 - وقتی که رابطه دارای چند کلید کاندید باشد.
2 - قتی که کلید های کاندید رابطه مرکب باشند.
3 - وقتی که کلید های کاندید با یکدیگر اشتراک صفت خاصه داشته باشند یعنی اقلاً یک صفت خاصه در آنها مشترک باشد.

مزایا ،معایب و اهداف نرما ل سازی

مزایای نرمال سازی عبارتند از:

1 - کاهش بعضی از آنومالیها

2 - ساده کردن اعمال بعضی از قواعد جامعیت
3 - کاهش بعضی از افزونگی ها
4 - ارئه یک طراحی بهتر و واضح تر با کمترین اختلاط اطلاعات
نکته : در بانک اطلاعاتی رابطه ای سه نوع افزونگی می تواند داشته باشد:
1-افزونگی طبیعی
2- افزونگی تکنیکی (ناشی از وجود مثلاً کلید خارجی)
3- افزونگی ناشی از طراحی بد و اختلاط و اطلاعات. نرما سازی این افزونگی سوم را تا حد زیادی کاهش می دهد.
معایب نرمال سازی عبارتند از:
1 - سربارگذاری بر روی سیستم چرا که در صورت نیاز باید پرتوها را پیوند داد.
2 - ایجاد نوعی افزونگی. اگر قرار باشد تجزیه یک رابطه به دو رابطه نر مالتر ، تجزیه ای خوب و بدن اضافات زائد باشد (بودن حشو) باید صفت مشترک در دو رابطه حداقل کلید کاندید یک از آن دو باشد.
در اینحال در رابطه کلید خارجی می شود و وجود کلید خارجی خود سبب افزونگی است.
3 - فرآیند نرمال سازی در محیط های بزرگ که تعداد رابطه ها و صفات خاصه زیاد است، کاری زمانگیر بوده و یافتن همه وابستگی ها زمان زیادی می خواهد.
4 - چ.ن مجموعه کاهش پذیر وابستگیهای تابعی یک رابطه، یکتا نیست، لذا روشهای مختلفی در طراحی وجود خواهد داشت و بدین ترتیب مشکل تصمیم گیری برای طراح پیش می آید.
اهداف کلی نرمال سازی عبارتند از:
1 - حذف بعضی از انواع افزونگی ها
2 - پرهیز از آنومالی های به هنگام سازی
3 - ایجاد یک طراحی که نمایش خوبی از دنیای واقعی باشد،یعنی درک شهودی آن ساده بوده و مبنای خوبی برای رشد آینده باشد.
4 - سهولت در اعمال بعضی از محدودیتهای جامعیتی

END