به نام خدا

پایگاه داده ها

تاریخچه ای از بانک اطلاعاتی :

نسل اول فایل های ساده ی ترتیبی:

ویژگی ها :

۱- نوار مغناطیسی وجود داشت .

۲- فایل منطقی و فیزیکی یکی بوده است.

۳- به هنگام سازی با ایجاد فایل پدر انجام می شد که باعث افزونگی می شود .

۴- نرم افزاری برای مدیریت فایل ها وجود نداشته است .

فایل فیزیکی:

فایل فیزیکی ان چیزی است که طراحی می شود و در پایین ترین سطح ممکن قرار دارد.

فایل منطقی:

فایل منطقی دیدی از همان فایل فیزیکی است که در بالاترین سطح یعنی همان کاربر عادی می باشد.

افزونگی:

یعنی تکرار ذخیره سازی و افزونگی برای بانک یک مطلب بسیار بد می باشد.

نسل دوم access method شیوه های دست یابی:

در این نسل حافظه های جانبی از نوع دیسک اختراع شد به همین دلیل شیوه ی دست یابی به صورت مستقیم شد و نه تر تیبی

ویژگیها:

۱- مستقیم شدن دست یابی

۲- نرم افزار a.m

۳- جدا شدن فایل منطقی و فیزیکی

۴- نرم افزاری برای مدیریت وجود نداشت

۵- افزونگی بالا می باشد

Index :به علامتهایی که گذارده می شود تا جستجو شود index می گویند.

نسل سوم سیستم مدیریت data:

نرم افزار کامل تری نسبت به نرم افزار a.m تهیه شد به عنوان واسط برنامه های کاربردی و فایل های محیط فیزیکی

ویژگیها :

۱- نرم افزار مدیریت

۲- فایل های منطقی متعددی از یک فایل فیزیکی به دست می اید .

۳- اشتراک داده ها یعنی افزونگی دیگر وجود ندارد.

نسل چهارم data base manenger system

در نسل چهارم استقلال داده به طور کامل رعایت می شود در این نسل برنامه ی جامعی به نام dbms به وجود امد که وظیفه ی ان مدیریت بانک اطلاعاتی است وواسطی است بین برنامه های کاربردی و محیط فیزیکی

مهمترین ویژگی این نسل چند سطحی شدن معماری بانک است

. oracle ,access, sql server وغیره که همگی dbms هستند .

نرم افزار access چون کمپایلر ندارد بنابراین فایل exeنمی سازد.

نسل پنجمknowlege base :هوش مصنوعی :که در این زمینه کامپیوتر نیز قابلیت اموزش را دارد.

Data:

یک سری داده های خام می باشد.

Information:

یک مجموعه از اطلاعات گفته می شود.

Entity:

به هر چیزی که ما باید اطلاعات در مورد ان را ذخیره کنیم را Entity گویند.

صفت خاصه :

به هر کدام ازاجزء اطلاعات یک فرد یا یک شئ صفت خاصه گویند وصفت خاصه در هر مکانی ممکن است متفاوت باشد .

Relationship(رابطه):

فیلد :صفت خاصه

رکورد: تعدادی فیلد

فایل: تعدادی از رکوردها

انواع رابطه Relationship:

۱- یک به یک

۲- یک به چند

۳- چند به یک

Er: Entity Relation

چند به چند در بانک های اطلاعاتی قابلیت پیاده سازی ندارد.

فضای اطلاعاتی :

صفت خاصه ی مشترک:

در بسیاری از موجودیت های یک بانک صفات خاصه ی مشترکی پیدا می شود که این صفات را به ارث می برند.

تعریف جامع و کامل از بانک اطلاعاتی :

بانک اطلاعاتی مجموعه ای است از داده های ذخیره شده در مورد انواع موجودیت ها یا انواع entity ها یک محیط عملیاتی و ارتباط نین انها به صورت مستمر و مبتنی بر یک ساختار تعریف شده به صورت صوری با حداقل افزونگی تحت کنترل متمرکز و مورد استفاده ی یک یا چند کاربر به طور اشتراکی وهمزمان.

تعریف شده به صورت صوری : یعنی اینکه کاربر داده های خود را ان طور که می بیند ذخیره کند ویک دید انتزاعی باید داشته باشد.
به صورت مستمر و مبتنی بر یک ساختار:یعنی داده های عملیاتی در یک ساختار به خصوص ذخیره شود مجتمع بودن بانک باعث می شود افزونگی کاهش یابد.
تحت کنترل متمرکز: یعنی اینکه داده ها توسط یک سیستم مرکزی کنترل شود که در این صورت امنیت سیستم کنترل شود.

حسن و مزایای اشتراکی شدن :

۱- عدم افزونگی
۲- امنیت (کنترل متمرکز)
۳- استفاده ی بهینه و سرعت بالا در اطلاعات

دام ارتباطی:
استنتاجهای غلطی که از نمودار er به وجود می اید (رابطه ی بین موجودیت ها) دام ارتباطی گفته می شود.
به دو روش می توان بانک اطلاعاتی ایجاد کرد.

۱- روش فایلی(کلاسیک)
۲- روش بانکی

روش فایلی(کلاسیک):

در این روش برای هر برنامه ی کاربردی یک فایل وجود دارد و برای هر فایل نیز باید یک سیستم فایل نوشته شود. در این سیستم تجمع داده وجود ندارد در این سیستم ارث بری وجود ندارد در این روش اگر ساختار فایل تغییر کند برنامه ها نیز باید تغییر کند.

روش بانکی:

در این روش وحدت ذخیره سازی وجود دارد ولی هر کاربر دید خاصی از داده ها دارد در این روش تغییر کردن فایل تاثیری بر روی برنامه های

کاربردی ندارد به دلیل اینکه فقط یک فایل به وجود می آید داده ها مجتمع هستند دراین روش مهمترین قسمت سیستم DBMS است.

اجزا و عناصر محیط بانک :

۱.سخت افزار : سخت افزار مورد نیاز بستگی به نوع بانک دارد که معمولا موارد زیر مورد نیاز است:

۱- یک ماشین محاسبه گر که معمولا کامپیوتر است

۲- یک سخت افزار ذخیره سازی

۳- در صورت امکان سخت افزار ارتباطی (برای بانکهای شبکه ای)

۴- نرم افزار

۵- کاربر : افرادی که با بانک کار می کنند به چند دسته تقسیم می شوند

a. DBA (Data base Administrator): این فرد اداره کننده بانک می باشد.

b. DBP (Data base Programmer): این فرد طراح و برنامه نویس سیستم است.

c. End User : فردی که با سیستم عملیات ذخیره و بازیابی را انجام می دهد.

۶- داده ها : اطلاعات و داده هایی که برای موجودیت ها ذخیره می شوند.

بانک های اطلاعاتی توزیع شده: بانک های اطلاعاتی که در شبکه قرار می گیرند بانک های اطلاعاتی توزیع شده می گویند.

انواع بانک های اطلاعاتی توزیع شده:

۱- بانک توزیع شده با داده

۲- بانک توزیع شده با داده و سیستم

معماری بانک اطلاعاتی

سطح خارجی:

سطح کاربر خاص است و کاربر با آن کار می کند، هرکاربر دارای دو زبان HL و DSL است که به این وسیله دو زبان را می تواند محاسبات را انجام دهد و سئوالات خود را از بانک بپرسد.در سطح خارجی کاربر در مورد داده ها یک تفکر انتزاعی دارد، هر کاربر می تواند دید خارجی مخصوص به خود داشته باشد، هر کاربر می تواند از داده هایی که ذخیره شده است دید خاصی داشته باشد که البته این دید را در سطح مفهومی (ادراکی) طراح بانک ایجاد می کند.

زبان برنامه نویسی HL : Hy Language (زبان میزبان)

این زبان یک زبان سطح بالای برنامه نویسی است که محاسبات و برنامه نویسی غیر بانکی با آن انجام می شود مانند زبان ویژوال بیسیک که در Access استفاده می شود.

زبان برنامه نویسی DSL :

یک زبان سطح بالاست که معمولا به صورت میهمان در کنار HL قرار می گیرد و به وسیله این میزبان می توان:

الف) داده ها را تعریف کرد DDL

ب) با داده ها کار کرد DML

ج) داده ها را کنترل کرد DCL

مثلا زبان SQL یک DSL است که در Access وجود دارد.

اصل وحدت عملگر:

این اصل یعنی اینکه برای انجام یک عمل مشخص در سطوح خارجی و ادراکی فقط یک دستور وجود داشته باشد این اصل باعث می شود بین سطح خارجی و سطح ادراکی مترجمی وجود نداشته باشد.

سطح مفهومی یا ادراکی:

دید طراح بانک است از داده های ذخیره شده در بانک در سطح مفهومی (ادراکی) طراح بانک انواع موجودیت ها ، ارتباط بین آن ها ، انواع فیلدها و نوع آن ها ، و دیدهای خارجی را تعریف می کند.

در این سطح DSL مفهومی وجود دارد یعنی طراح به وسیله DDL بانک را طراحی می کند و برای کنترل کردن داده ها از DCL استفاده می کند در این سطح طراح بانک از چگونگی ذخیره سازی و محیط گرافیکی مستقل است. نکته بعدی اینکه در این سطح طراح باید مدل بانک یا همان ساختار داده را مشخص کند.

مثلا در مدل رابطه ای ساختار آن جدول است ، یعنی هر موجودیت را با یک جدول معرفی می کنیم.

سطح داخلی:

در این سطح در واقع فایل های محیط فیزیکی تعریف می شود. مانند محتوا، ساختار و نحوه دست یابی

نکته: طراح بانک دخالت چندانی در سطح داخلی ندارد البته سطح داخلی با سطح فیزیکی فرق میکند.

در سطح داخلی محل رکورد ، توالی رکوردها، تخصیص فضای ذخیره سازی و تکنیکهایی برای فشرده سازی و رمزگذاری داده ها تعریف می شود.

سطح ادراکی از سطح داخلی مستقل است.

تبدیلات بین سطوح:

۱- تبدیل داده ای

۲- تبدیل احکام

۳- تبدیل ساختار

۱. تبدیل داده ای: یعنی اینکه داده های تعریف شده در سطح خارجی به داده های تعریف شده در سطح ادراکی تبدیل شود و برعکس

همچنین باید داده ها در سطح ادراکی به سطح داخلی تبدیل شوند و بلعکس

۲. تبدیل احکام : یعنی اینکه دستورات از سطح خارجی به دستورات سطح ادراکی تبدیل شوند و از سطح ادراکی به سطح داخلی و بر عکس در اینجا وجود **ندارد.**

۳. **تبدیل ساختار:**
   a. **تبدیل خارجی / ادراکی**
   b. **تبدیل ادراکی / داخلی**

a . تبدیل خارجی / ادراکی : مکانیزمی برای برقراری تناظر بین دیده های خارجی مختلف و دیدادراکی است ، این وظیفه DBMS است.

تغییراتی که در سطح ادراکی به وجود می آیند نباید در سطح خارجی اثر بگذارد.

این تبدیل یعنی اینکه اگر در سطح ادراکی تغییراتی اعمال شد ، سطح خارجی تغییر نکند که این باعت استقلال داده منطقی می شود. مثلا ممکن است در سطح ادراکی اطلاعات ساختار جدولی داشته باشند اما در سطح خارجی اطلاعات را بصورت گراف به کاربر نشان می دهیم حال اگر ساختار جدول ما تغییر کند گراف خارجی نباید تغییر کند.

b . **تبدیل ادراکی / داخلی : اگر در سطح داخلی تغییراتی ایجاد شود ، سطح مفهومی (ادراکی) نباید تغییر کند که این باعث استقلال داده فیزیکی خواهد شد.**

در اصطلاح به تبدیلات – چه داخلی و چه خارجی- mapping (نگاشت) می گویند و همیشه تغییرات در سطح پایینی به وجود می آید و نباید در سطح بالایی اثر بگذارد.

مزیت های به وجود آمدن سه سطح معماری:

۱. به وجود آمدن کاربران خارجی مختلف با دیدهای متفاوت
۲. ایجاد استقلال داده (چه منطقی و چه فیزیکی)
۳. برقراری قوانین جامعیت

فرق بین DA با DBA :

مدیر داده DA : شخصی است که کنترل داده های سازمان را به عهده دارد این فرد مشخص می کند که سازمان باید چه داده هایی داشته باشد. رابطه بین داده ها را درک میکند جدای از اینکه داده ها دستی باشند یا کامپیوتری این فرد یک مدیر است و نه یک فرد فنی
مدیر بانک DBA : مدیر بانک اطلاعاتی یک شخص فنی است که مسئول پیاده سازی تصمیمات مدیر داده است.

کاتالوگ سیستم :

هر DBMS باید شامل یک کاتالوگ باشد . کاتالوگ بانک (شبه داده) شامل:

الف) تمام دیدهای خارجی و ادراکی
ب) تمام نگاشت ها mapping
ج) موجودیت ها و صفات خاصه آن ها
د) ارتباط بین موجودیت ها و index آن ها
و) تاریخ ایجاد بانک اصلاح بانک و ویرایش بانک تغیر بانک
د) واحدهای اندازه گیری
ه) همه مشخصاتی که در سیستم مفهومی تعریف شده است

به داده های موجود در کاتالوگ (دیکشنری) داده meta data می گویند.
درواقع کاتالوگ بانکی است از بانک طراحی شده ؛ کاتالوگ امکانی است برای کنترل و نگهداری بانک و در صورت نیاز توسعه دادن آن در طول حیات آن.

مزایای سیستم بانک اطلاعاتی:

۱. مدلینگ داده های عملیاتی بر اساس ساختار آن ها:

وجه تمایز سیستمهای بانکی و غیر بانکی همین مسئله است و باعث می شود که کاربران دید انتزاعی از داده های ذخیره شده داشته باشند، مدل های ساختمانی متفاوتی برای بانکهای اطلاعاتی وجود دارد مانند رابطه ای مدل شبکه ای مدل سلسله مراتبی و مدل شئ گرایی.

۲ـ وحدت ذخيره سازي كل داده هاي عملياتي :

وجود سطح ادراكي اين امكان را فراهم مي كند منظور از وحدت ذخيره سازي داده ها اين است كه همه داده هاي مربوط به يك Entity در يك ساختار ذخيره مي شود.

۳ـ اشتراكي شدن داده ها :

سيستم DBMS امكان مي دهد كه كاربران از داده هاي ذخيره شده به طور اشتراكي استفاده كنند و هر كاربر ديد خارجي مخصوص به خود داشته باشد در صورتي كه يك فايل فيزيكي بيشتر وجود ندارد .

٤ـ كاهش ميزان افزونگي :

وحدت ذخيره سازي باعث كاهش افرونگي مي شود .

٥ـ تعدد شيوه هاي دست يابي و دست يابي آسان به داده ها :

وجود سطح ادراكي و نگاشت هاي آن بين سطوح خارجي و داخلي باعث مي شود براي دست يابي به داده از شيوه هاي متعدد دست يابي استفاده شود شيه هاي دست يابي را مدلها تعيين مي كنند.

٦ـ عدم وجود ناسازگاري داده ها : Inconsistency

منظور از ناسازگاري اين است كه افزونگي در بانك وجود داشته باشد و هنگام به هنگام سازي بعضي از داده ها را ويرايش كنيم و مابقي بدون ويرايش بماند مثلا معدل دانشجو در ۲ جاي مختلف ذخيره شود و هنگام تغيير كردن فقط يكي از آنها را تغيير دهيم با حذف افزونگي ناسازگاري نيز از بين مي رود.

۷ـ تأمين جامعيت :

بي نقص بودن داده ها را در بانك جامعيت مي گويند
مفهوم ميدان جامعيت را تأمين مي كند مثلا نمره منفي براي يك دانشجو نشان دهنده نقص بانك است يا دانشجويي كه ثبت نام هويتي نداشته اما انتخاب واحد داشته است .

۸ـ تأمين استقلال داده اي :

دو نوع استقلال داده وجود دارد استقلال داده فيزيكي و منطقي استقلال داده ها يعني مصونيت برنامه هاي كاربردي در قبال تغييراتي كه در سطوح اداراكي داخلي و فيزيكي ايجاد مي شود كليد استقلال داده فيزيكي نگاشت داخلي ادراكي است كليد استقلال داده منطقي نگاشت خارجي ادراكي است .

۹ـ حفظ محرمانگي اطلاعات:

براي حفظ محرمانگي اطلاعات مي توان از روشهاي مختلفي مانند :
كنترل دست يابي ها (ديدهاي خارجي) يا ذخيره سازي داده ها به طور رمزي و يا رمزگذاري روي بانك استفاده كرد .

۱۰ـ تسريع در دريافت پاسخ پرس و جوها :

اين شيوه به صورت آشكار نيست چون شركت microsoft لو نداده .

معايب بانك اطلاعاتي :

۱ـ به دلیل اینکه داده ها متمرکز هستند ممکن است داده ها به خطر بیفتند و راه حل آن هم پشتیبان گرایي است .
۲ـ به دلیل اینکه داده ها متمرکز هستند ممکن است جامعیت داده ها به خطر بیفتد .
۳ـ سخت افزار اضافه ممکن است نیاز باشد .
٤ـ برنامه نویس و پیاده سازي تمام مفاهیم بانك اطلاعاتي پیچیده است .

computer Aided software Engineering : Case

نرم افزارهاي مهندسي که ابزار طراحي هستند بعضي از DBMS هاي داراي ابزار کمکي (Case) هستند که در طراحي و پیاده سازي بانك مورد استفاده قرار مي گیرند از جمله :۱ـ ابزاري براي رسم نمودار ER ، ۲ـ کار با لغت نامه ها ، ۳ـ تعریف شمایي (شکلي) بانك یا همان تعریف گرافیکي بانك ، ٤ـ ایجاد رابطه بین موجودیت ها به صورت گرافیکي ،٥ـ تهیه نمودارها ، ٦ـ مهندسي وارون (گزارشي از طراحي سیستم از آخر به اول)

ساختار داده (مدلهاي باتك اطلاعاتي) :

ساختار داده اي که توسط طراح بانك در سطح ادراکي تعیین مي شود مدل بانك گفته مي شود به طور کلي ٤ مدل وجود دارد .

۱ـ ساختار سلسله مراتبي :

این ساختار قدیمي داده ها و ارتباط بین آنها را به کمك یك درخت نمایش مي دهد .
روش سلسله مراتبي براي ارتباطات یك به چند بین موجودیتها مناسب است .
براي بازیابي نیاز به عملکرد ریشه یاب (یا سطح یاب) و عملکرد وابسته یاب (فرزندیاب)

ویژگیهاي شیوه سلسله مراتبي :

آنومالي یعني اینکه از نظر اجرایي ناممکن است .

۱ـ عمل درج و حذف عناصر داراي آنومالي است (آنومالي يعني دشواري اجسام يك عمل در يك ساختار)
۲ـ در اين روش افزونگي بسيار بالاست .
۳ـ پيمايش (غواصي) در سطوح پايين تر الزاما بايد از سطوح بالا شروع شود .
٤ـ در اين روش كاربر وضوح دارد (يعني مطلب را مي فهمد) اما محيط انتزاعي آن مسطح نيست .
٥ـ به هنگام سازي اين روش بسيار مشكل است .
٦ـ تئوري رياضي در اين روش وجود ندارد .
نرم افزار IMS ساختار سلسله مراتبي را مي كشد .

۲ـ مدل رابطه اي

در اين مدل طراحي موجوديتها را به صورت جدول نمايش مي دهد بنابراين بانك تشكيل مي شود از مجموعه اي از جداول .
براي برقراري ارتباط بين جداول ، جداول ديگري را طراحي مي كنيم رابطه ها همان جداول هستند در اين مدل براي بازيابي اطلاعات فقط به عملكرد سطرياب نياز است .
عمليات درج و حذف در مدل رابطه اي بسيار آسان است .

ويژگيهاي مدل رابطه اي

۱ـ ديد كاربر بسيار واضح است و جدول محيطي مسطح دارد .
۲ـ جداول داراي تئوري رياضي مي باشند .
۳ـ پيمايش جداول يا رابطه ها ستقل از جداول يا رابطه ديگر است .
٤ـ براي پاسخگويي به پرسش ها جستجو به صورت خطي انجام مي شود.

٥ـ در اين مدل داده ها و ارتباط بين آنها با مكانيزم واحدي نشان داده مي شوند (جدول)
٦ـ براي طراحي بهينه رابطه ها قوانين تئوري نرمال سازي وجود دارد .
٧ـ افزونگي در مدل رابطه اي با توجه به قوانين نرمالسازي قابل حذف است .

٣ـ مدل شبكه اي

در اين ساختار كه به آن ساختار پلكس Plex گفته مي شود براي نمايش داده ها و ارتباط بين آنها از گراف استفاده مي كنند اين مدل براي نمايش ارتباطات چند به چند بسيار مناسب است .
چيزي به عنوان جهت در گرافها وجود ندارد .
عمل بازيابي در مدل شبكه اي بسيار پيچيد تر از مدل سلسله مراتبي است زيرا عمل بازيابي بسيار پيچيده تر است و اين باعث مي شود تا ما اندكي سردرگم نشويم زيرا پيمايش گرف به دو روش سطحي و عمقي راههاي متعددي دارد .
١ـ عمل ذخيره سازي (درج حذف به هنگام سازي) آتومالي دارد
٢ـ اصل وحدت عملكرد رعايت نمي شود يعني عملكرد با هم تفاوت دارد .
٣ـ از ديد كاربر وضوح ندارد و محيط آن مسطح نيست .
٤ـ مدل شيء گرايي :
در اين روش براي هر موجوديت يك كلاس با طبقه ايجاد مي شود كه هر كلاس داراي خصوصيت و ويژگيهاي خاصي است سپس عناصري را كه مي خواهيم ذخيره كنيم را عضو كلاس ها قرار مي دهيم كه در اين صورت ارث بري از كلاسهاي مختلف به وجود مي آيد .
مدل رابطه اي و تعاريف مربوط به آن
مؤسس مدل رابطه اي يك رياضي دان به نام آقاي كاد است .

دامنه و ميدان: (توابع)

ميدان مجموعه اي است كه مقادير يك صنعت خاصه از آن گرفته مي شود .

به ميدان در لاتين Domain مي گويند .

تعريف رابطه :تعريف رابطه همان تعريف رياضي آن مي باشد كه داراي دامنه و برد است و چند تايي هايي مرتب را شامل مي شود . $y = f(x) y = \frac{x^2 - 4}{x + 2} x = 1 y = \frac{-3}{0} = 1$

Heading مجموعه اسامي صفات خاصه را Heading مي گويند .

Body مجموعه اي از چندتايي هاي مرتب كه متغيرند به عنوان مثال مي توان :

معمولHeading ثابت است ولي body متغير است .

هر رابطه از يك Heading و يك Body تشكيل مي شود.

رابطه هر زير مجموعه اي از ضرب كارتزين n ميدان است عنوان مهمي است اين ضرب كارتزين و گاهي اوقات به آن join نيز مي گويند چون اتصال را برقرار مي سازد .

ميدان Name را در ميدان family ضرب كرده ايم جواب حاصل را بگوييد ؟

كه تنها از اين موارد سه مورد را صحيح مي دانيم و همه را نياز نداريم .

در مدل رابطه اي هنگام پياده سازي هر رابطه اي جدول مي شود.

به هر سطر جدول تاپل topel ميگويند و به هر ستون از جدول يك صفت خاصه مي گويند.

مقايسه جدول و مدل رابطه اي از نظر عنوان كه مي توان يك رابطه را به جدول تبديل كرد درجه رابطه چيست ؟ تعداد صفحات خاصه رابطه (ستون ها) درجه رابطه است و هميشه ثابت است .

كاردينالیتي رابطه :تعداد تاپل هاي رابطه در يك لحظه از حيات آن كاردينالیتي رابطه گفته مي شود كاردينالیتي تغيير مي كند.

خصوصيات رابطه :

١ـ در رابطه تاپل تكراري وجود ندارد. اما در جدول ما ممكن است سطر تكراري داشته باشيم ٢ـ تاپل ها نظم ندارند . اما در جدول ما نظم داريم . اين ويژگي نشان مي دهد كه رابطه و جدول دقيقا يكسان نيستند زيرا سطرهاي جدول داراي ترتيب بالا و پايين هستند.

٣ـ صفات خاصه نظم ندارند در رابطه و اين ويژگي نشان مي دهد رابطه و جدول دقيقا يكسان نيستند زيرا ستونها در جدول به طور خودكار نظم دارند .

٤ـ همه مقادير صفت خاصه Ai تجزيه ناپذير هستند منظور از تجزيه ناپذيري اين است كه هرگز به دو اطلاعات تقسيم نشود . مقدار اتوميك : مقداري است ساده كه قابل تجزيه به مقادير ديگر نباشد مانند : تاريخ تولد كه اگر روز ، ماه و سال در يك ستون ذخيره شود اتوميك نيست اما اگر در سه ستون جدا ذخيره شود اتوميك است .

تعريف : رابطه اي كه داراي خصوصيت تجريه ناپذيري (اتوميك) باشد رابطه نرمال شده (1NF) است در تعريف تئوريك مدل رابطه اي همه رابطه ها بايد 1NF باشند.

ميدان :

ميدان مجموعه اي است از مقادير هم نوع كه مقادير يك صفت خاصه از آن گرفته مي شود مثلا براي شماره ملي تمام اعداد ١٠ رقمي مجاز است .

پياده سازي ميدان در بانكهاي اطلاعاتي به دليل پيچيدگي آن مشكل است و معمولا نرم افزارهاي DBMS بعضي از ميدانها را پياده سازي مي كند.

نقش ميدان در پايگاههاي اطلاعاتي :

۱ـ كنترل مقداري پرس و جوها :

مقادير يك صفت خاصه در طول حيات رابطه از مقادير ميدان گرفته مي شود به عبارت ديگر بايد در ميدان وجود داشته باشد (قاعده جامعيت خاص) مثلا : اين پرسش كه دانش آموزاني را استخراج كنيد كه نمره آنها ۲۱ است اشتباه است .

۲ـ كنترل سماتيك پرس و جوها

ممكن است دو صفت خاصه از يك نوع باشند (مثلا عددي باشد) اما هر كدام ماهيت متفاوت دارد مانند شماره دانشجويي و شماره شناسنام اين دو را نمي توان با هم مقايسه كرد مثلا سؤال زير فاقد سمانتيك است .

دانشجويي را استخراج كنيد كه شماره دانشجويي آنها كوچكتر از شماره شناسنامه آنها باشد ؟ كه اين سؤال معنا ندارد .

۳ـ تسهيل در پاسخگويي بعضي از پرس و جوها

در بعضي از پرس و جوها سؤال در مورد خود بانك و رابطه ها است مانند اينكه در كدام جدول اطلاعاتي در مورد دانشجويان وجود دارد اين سؤال بوسيله مفهوم ميدان و كاتالوگ سيستم جواب داده خواهد شد .

كليدها در مدل رابطه اي :

مفهوم مدل رابطه اي بر اساس كليدها گذاشته شده است كليدها در جداول مورد استفاده مي باشد .

۱ـ كليد كانديد : condidate key

مجموعه اي از صفت خاصه هاي (Ai,Aj, An) يا زيرمجموعه اي از ستونها در رابطه يا جدول كه دو خاصيت زير را داشته باشند كليد كانديد گفته مي شود .

خواص

الف) منحصر بودن (يكتايي) uniquenese :

منحصر بودن به اين معنا كه در هر لحظه از حيات رابطه يا جدول صفت خاصه هاي (Ai,Aj, ,An) بايد يكتا باشد و هيچ دوتاپلي تكرار نشود .

ب) كاهش ناپذيري minimality :

توضيح ب) يعني اگر يكي از عناصر كليد كانديد مثلا Aj را حذف كنيم منحصر بودن خود را از دست بدهد همچنين نبايد فيلد اضافه داشته باشد هر رابطه بايد حداقل يك كليد كانديد داشته باشد در بدترين حالت ممكنه همه صفت خاصه صفت خاصه ها (فيلدها) كليد رابطه اي مي شوند به چنين رابطه اي All-key مي گويند.

اين جدول داراي دو كليد كانديد است يكي ساده و ديگري مركب در كتاب CjD كليد كانديدهاي ازدواج و طلاق بحث شده است .

۲ـ كليد اصلي : Primary key

كليد كانديدي كه توسط طراح بانك انتخاب مي شود كليد اصلي گفته مي شود طراح بانك بايد از بين كليدهاي كانديد فقط يكي را براي كليد اصلي معرفي كند دو عامل براي اين انتخاب مهم است :

الف) نقش و اهميت كليد اصلي نسبت به ساير كليدهاي كانديد

مثلا : شماره نظام پزشكي در تشخيص پزشكان .

ب) كوتاهترين كليد كانديد را به عنوان كليد اصلي انتخاب مي كنند.

ج) كليد اصلي تنها مكانيزم نشان دهي آدرس يا آدرس دهي در سطح تاپل است به همين دليل وجود كليد اصلي الزامي است .

۳ـ کلید بدلیل Alternate key :

کلید بدیل : هر کلید کاندید به غیر از کلید اصلي را مي گويند .

٤ـ کلید خارجي : foreign key

صفت خاصه x (يعني ستون) از رابطه R2 کلید خارجي است اگر x در رابطه R1 کلید اصلي باشد .

تکليف : براي يك کتاب خانه يك بانك طراحي کنيد که کتاب ها اعضاي کتابخانه و ناشران و کارمندان کتابخانه ذخيره مي شوند بايد جدولها باشد صفات خاصه باشد و کليد اصلي را مشخص کنيم و تاريخ رفت و برگشت کتاب هم نياز است ؟

کليد خارجي حتما نبايد کليد اصلي باشد در جدول خودش ولي در جدول ديگر بايد کليد اصلي باشد کليد خارجي امکاني است براي ارجاء از يك رابطه به رابطه ديگر به وسيله کليد خارجي جدولها به يکديگر پيوند زده مي شوند.

نکته : کليد خارجي تنها امکان ايجاد ارتباط بين جداول نيست بلکه وجود هر صفت خاصه مشترك بين دو جدول عاملي است براي ارتباط جدولها

قواعد جمعيت در مدل رابطه اي :

قواعد جامعيت در بانك intigrity Rvle

در مدل رابطه اي قواعدي بايد وجود داشته باشد که بر اساس آنها جامعيت و بي نقصي بانك کنترل و تضمين شود .

قواعد جامعيت خاص :

قواعدي هستند که در يك سيستم مشخص و خاص وضع مي شوند مي توان گفت اين قواعد همان مبحث ميدان مي باشد مانند نمره يك درس که بين صفر تا ۲۰ است اين قواعد بستگي به سيستم خاصي ندارد و در همه بانکها مطرح هستند .

قواعد جامعيت عام موجوديتي :

این قاعده یعني این که هیچ بخشي از کلید اصلي نباید داراي مقدار تهي (Null Value) باشد مقدار تهي Null valud مقدار خاصي است که مي تواند تهي باشد منظور همان جاي خالي است یا صفر یا به طور کلي هر مقدار که در میدان آن فیلد نباشد . دلیل این قاعده این است که کلید اصلي نقش یك نمونه اي از موجودیت را دارد و هر نمونه از موجودیت را از روي شناسه اش که کلید اصلي است مي شناسد اگر این شناسه تهي باشد به این معنا است که این نمونه از موجودیت وجود ندارد و تضاد به وجود مي آید .

قواعد جامعیت عام ارجاعي :

اگر صفت خاصه کلید خارجي باشد در رابطه R2 به طوري که در رابطه R1 کلید اصلي باشد آن گاه صفت خاصه در رابطه R2 الف) مي تواند مقدار تهي داشته باشد به شرط اینکه خودش در رابطه R2 کلید اصلي نباشد
ب) اگر مقدار تهي نداشته باشد حتما باید مقداري داشته باشد که در سطري از رابطه R1 کوجود باشد این دو قاعده در تمام حیات یك رابطه باید رعایت شود.

قاعده جامعیت و سیستم DBMS:

الف) DBMS سیستم مدیریت بانك از انجام عملیاتهایي که باعث مي شود قواعد جامعیت نقض شود جلوگیري مي کند .
ب) سیستم مدیریت بانك DBMS در پي انجام یك عمل روي بانك به خاطر رعایت قاعده جامعیت آن عمل را در تمام رابطه ها منتشر مي کند.
ج) سیستم مدیریت بانك DBMS مي تواند با حذف Tple از یك رابطه صفت خاصه اي را که از طریق آن تاپل ها از رابطه هاي دیگر به این تاپل رجوع مي کنند را نال Null مي کند.
مدل رابطه اي باید حداقل داراي سه جنبه اساسي باشد :

۱ـ ساختاري داده اي (جدول)
۲ـ قواعد جامعيت بايد داشته باشد .
۳ـ حداقل داراي عملكردهايي مانند :گزينش ـ پرتو و پيوند براي كار با داده ها داشته باشد .

عملگرهای جبر رابطه ای

جبر رابطه ای:

به مجموعه ای از قوانین و عملگرهاکه امکان پردازش جداول را فراهم می سازند جبر رابطه ای می گو یند.
بانک مدل رابطه ای دارای زبان کار با داده هاست dslو زبان کار با داده ها مجموعه ای است از عملگرها که بر روی رابطه ها(جداول) کار می کنند این عملگرها رابطه ها را به عنوان عملوند گرفته ویک رابطه را به عنوان نتیجه برمی گرداند.
عملگرها:

۱- عملگرهای مجموعه ای :

اجتماع-اشتراک- تفاضل-ضرب کارتزین

۲- عملگرهای رابطه ای ساده:

گزینش(محدودیت)-تصویر-الحاق-تقسیم

به شرطی یک رابطه را با رابطه ی دیگر سازگار گویند که هم درجه باشند یعنی تعداد خانه های جداول یکسا ن باشند.

عملگرهای مجموعه ای بر روی رابطه ی r2وr1 در صورتی معنی دارد که رابطه ی r2وr1

سازگار باشندیعنی درجه ی انهایکسان بوده وصفت خاصه ی iام از هر دو رابطه از یک میدان باشند.

۱- اجتماع دو رابطه:

STIDED	نام
100	علي
107	حسن
105	رضا
109	اكبر
108	سارا

STIDED	نام
100	علي
105	رضا
108	سارا

جدول اجتماع =

نام	STIDED
علي	100
رضا	105
سارا	108
اكبر	109
حسن	107

تاپلهای در عمل اجتماع unionحذف می شود.

۲- عملگر اشتراکintersect:

اشتراک دو رابطه رابطه ای است که تاپلهایش در هر دو رابطه وجود داشته باشد.

STIDED	نام
100	علي
107	خسن
105	رضا
109	اكبر

STIDED	نام
100	علي
105	رضا
108	سارا

جدول اشتراک دو جدول بالا

STIDED	نام
100	علي
105	رضا

۳-عملگر تفاضل minus :

تفاضل دو رابطه رابطه ای است که تاپلهایش در رابطه ی اول موجود باشد اما در رابطه ی دوم وجود نداشته باشد.

STIDED	نام
100	علي
107	حسن
108	رضا
109	اكبر

STIDED	نام
100	علي
105	رضا
107	حسن
108	سارا
109	مريم

جدول تفاضل =

STIDED	نام
109	مريم
108	سارا

عملگرهای اجتماع و اشتراک خاصیت جابه جایی و شرکت پذیری دارنداما تفاضل این خاصیت ها را ندارد.

۴-عملگر ضرب کارتزین:times

X	5	A	ALI	13	
y	12	B	REZA	15	
z	8	C	ALI	9	
w	13				

X	5	A	ALI	13	
X	5	B	REZA	15	
X	5	C	ALI	9	
y	12	A	ALI	13	
y	12	B	REZA	15	
y	12	C	ALI	9	
Z	8	A	ALI	13	
Z	8	B	REZA	15	
Z	8	C	ALI	9	

ضرب دو جدول صفحه ی قبل

W	13	A	ALI	13
W	13	B	REZA	15
W	13	C	ALI	9

حاصل ضرب دو رابطه رابطه ای است که تاپلهایش از الصاق هر یک از دو تاپل دو رابطه به دست می اید.

عملگر ضرب کارتزین زمان وفضای زیادی می خواهد به همین دلیل تا حد امکان باید از ان اجتناب کرد اگر رابطه یr1دارایmسطروnستون باشد ورابطه یr2دارایxتا سطروyتا ستون باشد انگاه r1*r2 دارایm*x سطروn*y ستون است.

عملگر ضرب کارتزین در مدل رابطه ای خاصیت جابه جایی دارد زیرا در مدل رابطه ای ترتیب ستون ها مهم نیست همچنین ضرب کارتزین دارای خاصیت شرکت پذیری است.

عملگر گزینش یا محدودیت restrict به عنوان select شناخته میشود.

این عملگر طبق یک شرط تعدادی از تاپلهای یک رابطه را برمی گرداند

S#	P#	Qty
s1	p1	300
s1	p2	200
s1	p3	400
s1	p4	200
s1	p5	100
s1	p6	100
s2	p1	300

جدول sp

s2	p2	400
s3	p2	200
s4	p2	200
s4	p4	300
s4	p5	400

جدول تولیدکنندگان

S#	s name	status	city
s1	sn1	20	c2
s2	sn2	10	c3
s3	sn3	30	c3
s4	sn4	20	c2
s5	sn5	30	c1

جدول قطعات

P#	P name	color	weight	city
p1	nut	red	12	c2
p2	bolt	green	17	c3
p3	screm	blue	17	c4
p4	screw	red	14	c2
p5	cam	blue	12	c3
p6	cog	red	19	c2

عملگر تصویر یا project:

این عملگر تعدادی از صفات خاصه ی یک رابطه را انتخاب می کند وستون یاب است.

این عملگر تاپلهای تکراری را حذف می کند زیرا خروجی عملگر projectنیز یک رابطه است و در رابطه تاپل تکراری وجود ندارد.

همچنین صفات خاصه نمی توانند تکراری باشند (ستونها نمی توانند تکراری باشند).

عملگر پیوندjoin :

کد	نام	معدل
100	علي	12.5
102	رضا	14
105	سارا	13.5
107	حسن	15

Code	Telephon
100	5555
102	45101
103	5610101
107	23107

پیوند دو جدول بالا

کد	نام	Avg	Code	Telephon
102	رضا	14	100	5555
105	سارا	13.5	100	5555
105	سارا	13.5	102	45101
105	سارا	13.5	103	5610101
107	حسن	15	100	5610101
107	حسن	15	102	5610101
107	حسن	15	103	5610101

کد	نام	Avg	code	telephon
100	علي	12.5	100	5555
102	رضا	14	102	45101
107	حسن	15	107	23107

پیوند تساوی =

این عملگر (پیوند) ابتدا دو جدول را که دارای صفت خاصه ی مشترکی هستند رادر هم ضرب کارتزین می کنند سپس به کمک عملگر مقایسه ای join حاصل را فیلتر می کند.

به وسیله ی دو عملگر ضرب کارتزین و عملگر select می توان join را شبیه سازی کرد اگر فیلد مشترک وجود داشته باشد نمی توان join کرد.

عملگر join که مقایسه ی ان مساوی باشد به ان join تساوی می گویند . معمولآ پیوند تساوی در بانکهای رابطه ای به کار می رود.

پیوند طبیعی:

اگر در پیوند تساوی صفت خاصه ی مشترک تکرار نشود به ان پیوند طبیعی می گویند .

از اینجا به بعد هر جا join استفاده شود منظور join طبیعی می با شد.

منظور از r1 join r2یعنی پیوند طبیعی r1 وr2 براساس صفت خاصه ی مشترک شان اگر r1وr2 چند صفت مشترک داشته باشند انگاه به صورت زیر می نویسیم:

R1 join r2 over یکی از صفات خاصه های مشترکشان

عملگر join خاصیت شرکت پذیری دارد.

عملگر تقسیمdivdby:

این عملگر رابطه ی r1 از درجه ی m+n را بر رابطه ی r2 از درجه ی n تقسیم می کند و خارج قسمت رابطه ایی است از درجه ی m(n تا صفت مشترک بین r1وr2 با میدانهای یکسان وجود دارد)و به صورت زیر نوشته می شود

R3= r1 divdby r2

فرض کنید r1 دارای دو صفت خاصه ی yوx باشدr2 نیز دارای یک صفت خاصه یy باشد در این صورت r3که حاصل تقسیم انها است دارای یک صفت خاصه ی x است که در ازای هر x در رابطه یr3 مقداری از r1*x باشد به طوری که به ازای تمام مقادیر y از رابطه ی r2 تاپل (yوx) در رابطه r1 وجود داشته باشد.

X	Y
رضا	5
علي	7

جدول تقسیم =

X
حسن

Y
5
7
3

3	حسن
7	رضا
5	حسن
7	حسن

زبان SQL

در بيشتر نرم افزارهاي ارايه شده براي بانك هاي اطلاعاتيDBMS قسمت تعريف داده هاي جدايي از قسمت پرس وجو است يعني طراح يا برنامه نويس به راحتي جداول كه همان رابطه ها هستند را طراحي مي كنند.
اما در SQL طراحي شده در بعضي از نرم افزارها دستوراتي وجود دارد برنامه نويس مي تواند جداول و فيلدهاي ان را بوسيله دستورات SQL ايجاد كند .

SQL

يك زبان استاندارد براي كاركردن بر روي بانك هاي اطلاعاتي رابطه اي است اين زبان در سال 1970 در شركت IBM طراحي شده است زبان SQL طراحي شده همه قوانين و رابطه موجود در مدل رابطه اي را پوشش نمي دهند .

نكته:

در SQL براي كار با داده ها چهار دستور بيشتر وجود ندارد.
(SELECT,INSERT,DELET,UPDATE)

سوال :

شماره ی وضعيت تهيه كنندگان ساكن شهر C2 را بدست آوريد؟

ازجدول S كه ساكن C2 اين مثل PROJECT عمل مي كند. S#STATUS

⟶

SELECT(S#STATUS)
FROME S
WHERE CITY='C2'

نكته :

دستور Select در SQL تلفيقي است از Select و Project در جبر رابطه اي است .

سؤال

کد تهیه کنندگاني که قطعه تولید مي کنند بدست آورید؟

Select s# s# SP

From sp s1

S1 s3

مثال

قطعاتي را استخراج کنید که رنگشان قرمز نباشد؟

SELECT *

FROM P

WHERE COLOR <>"RED"

مثال

شماره قطعه و وزن ونام قطعه بر اساس گرم استخراج کنید بطوریکه وزن آنها بیشتر از ۱۲باشد؟

SELECT (P#, PNAME, WEIGHT) 1000

FROM P

WHERE WEIGHT >12

مثال

شماره وضعیت تهیه کنندگان ساکن C2بر اساس مرتب شده بر حسب وضعیت آنهانشان دهید؟

SELECT S# , STATUSE

FROM S

WHERE CITY='C2'

ORDERY BY STATUS DESC

Order by

مي توانيم در دستور selectبعد از اعمال شرط با دستور ordery byداده ها انتخاب شده را مرتب كردبر اساس نزولي و صعودي

ASC

اگردستور صعودي باشد از اين دستور استفاده مي كنيم

DESE

اگر دستورنزولي باشد از اين دستور استفاده مي كنيم

نكته :

پيش فرض بصورت صعودي استواگر ننويسيم اشكال ندارد

مثال

كد توليد كنندگاني را استخراج كنيد كه قطعه P2يا3Pرا توليد مي كند وبراساس تعداد

توليد نزولي مرتب كنيد.

```
SELECT   DISTINCT   S#   QTY
FROM       SP
WHERE   P#='P2'    OR    P#='P3'
ORDERBY  2  DESE
```

نحوه كار SQL:

SQLاينگونه عمل مي كند كه در هنگام اجرا با دستور SELECTبصورت سطر ياب است

SQLبراي بدست آوردن جوابها جداول را سطر به سطر چك مي كند به همين دليل SQLزير جوابش تهي مي شود موقعي كه از عملگرANDاستفاده مي كنيم زيرايك سطر خاص در جدولSPستون

#Pنمي تواند هم P2وp3باشد براي حل اين مسئله بايد از select هاي تودر تواستفاده كرد.

```
SELECT      P#,QTY
FROM        SP
WHERE   P#='P2' AND   P#='P3'
```

مثال

مشخصات قطعاتي را به دست آوريد كه وزن آنها بين ۱۹ - ۱۶ باشد و بر حسب رنگ و وزن آنها را مرتب كنيد.

```
Select p# (p name,color,weight,city)
Form  p                                   مرتب سازي بر اساس۲ ستون
  Where  weight  between 16        خود ۱۶و۱۹ هم در نظر مي گيرد
  and 19
Order by  color,weight              اول براساس رنگ بعد براساس وزن
```

نكته:

هر گاه از دستور order by استفاده شود ديگر از ستاره استفاده نمي كنيم چون بايد تك تك بنويسيم

Between:

عملگر بين راانجام مي دهد و Not Between عملگر بين آنها نباشد.

مثال:

مشخصات قطعاتي رابه دست آوريد كه وزن آنها مساوي يكي از مقادير ۱۷و۱۶و۱۲ باشد.

```
Select *
From  p
Wherr  weight  In[12,16,17]
```

: In

عمل براي مجموعه ها استفاده مي شود و در صورتي كه فيلد مورد نظر در مجموعه باشد انتخاب مي شود عملگر In شبيه چند Or است Not in يعني در اين مجموعه نباشد و عكس بالا عمل مي كند .

مثال:

تهيه كنندگاني را كه بيابيد كه حرف اول آنها A باشد ؟

در محيط داس:

علامت؟ جايگزين يك حرف است

علامت * جايگزين چند حرف است

```
Select *
 From s
Where    s name  like  'A%'
```

در SQL

_ جايگزين يك حرف است. مثل (_ _ _) سه حرفي

باشدB چند حرفي و حرف دومش 'B%-' جايگزين چند حرف است .%

پرس وجوهاي بر اساس پيوند جداول:

سوال:

تمام تركيبهاي تهيه كنندگان و قطعات را بدست آوريد كه از يك شهر باشند .

```
Select *
From s,p
Where   s.city=p.city
```

ضرب در SQL از دستور SELECTاستفاده مي كنيم بطوري كه در قسمت FROME نام دو يا چند جدول را مي نويسيم .

مثال:

نام توليد كننده نام قطعه توليد شده آن را به دست آوريد؟

```
Select   S. s name  ,P. p name
From     s,p, sp
Where    s .s# =sp. s#  and  p. p# =sp. p#
```

نام گذاري مجازي براي جداول :

```
Select    ABC ,  S name
From   S ,  ABC
Where    ABC, Status>10
```

هراسمي كه در اين قسمت مي نويسيم با قسمت Select.بايد يكي باشد

نام مجازي كه وجود ندارد و فضا هم اشغال نمي كند

*Select يك جدول در خودش ضرب شود حتما" بايد يك اسم مجازي From S,S برايش در نظر بگيريم.

تمام جفت نام تهيه كنندگاني را بيابيد كه از يك شهر باشند ؟

```
Select    S name ,S name
From    S,S
Where
```

غلط است اين گونه نوشتن پس براي رفع اين اشكال از نام گذاري مجازي استفاده مي كنيم كه اينجا نمي توانيم تعريف كنيم

SELECT تو در تو :

نام تهيه كنند گاني را بيابيد كه قطعه P1 را توليد مي كند ؟

```
Select     S.S name
From     S,SP
Where     SP.P# ='P1'  and  S.S# =SP. P#
```

```
Select       S name
From      S
Where      S#    IN (Select  S#
         From  Sp
  Where      P# =
                                                    'P1')
```

مثال:

نام تهيه كنندگاني را به دست آوريد كه قطعه قرمز توليد مي كند؟

```
Select       S.S name
From      S
Where      S#      IN ( Select   S#
                        From    SP
  Where   P#) IN ( Select    P#
                                  From    P
                                 Where     Color =' red')
```

تهي بودن سطر جداول:

```
Select     *
From     P
Where      Weight  is null
```

شماره تهيه كنندگاني را بيابيد كه همشهري S1 باشند؟

```
Select    S#
From    S
Where   City =( Select  City
           From  S
                  Where   S# ='S1')
```

شماره تهیه کنندگاني را که وضعیتشان از وضعیت S2 بزرگتر باشد؟

```
Select   S#
From  S
Where   Status >( Select  Status
          From   S
               Where   S# =’S2’)
```

توابع ستوني در SELECT :

تابع	MIN	MAX	COUNT	AVG	SUM

مثال:

تعداد تولید شده قطعه P1 را به دست آورید؟

```
Select    Sum (QTY)
From    Sp
Where   P# =’P1’
```

شماره تهیه کنندگاني را به دست آورید که مقدار وضعیت انها کوچکتر از مقدار میانگین وضعیت هاست؟

```
Select    S#
From   S
Where   Status  < ( Select   AVG  (Status)
                        From   S)
```

تعداد شهرهاي موجود در جداول P چند است ؟

```
Select   Count(city)
From   P
Where   Count (*)
From   P
Select   Count (Distinct city)
From   P
```

تعداد تولید کنندگان قطعه P2 را بهدست آورید؟

```
Select    Count (*)
From    P
Where    P# ='P2'
```

هرگاه از توابع استفاده مي كنيم ديگر نمي توانيم ستونهاي ديگر را در Select كنيم .چون جواب آن يك عدد مي شود و ديگر نمي توانيم در يك ستون جدول ديگر تركيب كنيم براي اين كار مي توانيم با Group by انجام دهيم.

```
Select     S# , SUM (QTY)
From    SP
Where    S# , ='S1'
```

غلط است

: HAVING , GROUP BY

شماره قطعه و مقدار توليد شده اي ان را براي هر مقدار به دست آوريد؟

```
Select     P# , sum (QTY)
From    SP
Group by    P#
```

شماره هر توليد كننده را با تعداد قطعه توليدي ان را به دست آوريد ؟

```
Select     S# , SUM (QTY)
From    SP
Group by   S#
Order by   2  Dese
```

:GROUP BY , HAVING

همیشه دستور Group by , Having با هم به كار مي روند و باعث مي شوند كه طبق شرايطي بعضي از گروه ها حذف مي شود .

دليل اين كه Where مي گذاريم مي خواهيم بعضي سطرها را حذف كنيم.

نقش Having در Group by مثل Where در سطر است .

مثال :

کد تولید کنندگاني را به دست آورید که بیش از ۲قطعه تولید مي کند ؟

```
Select     S#  ,  Count (*)
From     SP
Group by   S#
Having      count (*) > 2
```

مثال:

شماره تهیه کنندگاني را که همه قطعات را تولید کرده اند را به دست آورید؟

```
Select     S#
From     SP
Group by   S#
Having   Count(*) = (Select  Count ( P# )
                                From     P)
```

براي همه قطعه تولید شده شماره قطعه ,کل تعداد, ماکسیم تعداد تهیه شده از آن را بدون در نظر گرفتن تولیدات S1 به دست آورید؟

```
Select   P# , Sum (qty) , max(qty)
Form     Sp
Where  S# <> 's1'
Group by  p#
```

سور وجودي یا Exits :

سور وجودي شبیه ضرب دکارتي است وبمعناي وجودداشتن است.

مثال:

اسامي تهیه کنندگان قطعه P2 را بدست آورید.

```
Select   S#
Form     S
```

```
Where   S#   In  (Select  S#
                  From     Sp
                        Where  P# =’2’)
```

روش دوم:

```
Select    S name
From    S
Where  exit   (Select   *
                        From     Sp
                        Where    S.S# = Sp. S#      And   Sp.
P#)
```

روش سوم:

```
Select    S. S name

From    S, Sp
Where   S. S# = Sp. #    and  Sp. P# = ‘P2’
```

مثال:

اسامي تهيه كنندگاني را بدست آوريد كه همه قطعات را توليد مي كند ؟

```
Select    S name
From     S
Where    not Exists    (Select   *
                           From     P
                           Where    not   exists (Select    *
                                                  From     SP
                                                  Where  Sp  S# =
S  S #   and  Sp. P # =p .p  )
```

نکته:Giving درا

مثال:

کد توليدکنندگاني را بدست آوريد که P1 را توليد مي کند ولي P2 را توليد نمي کند.

```
Select  s#
From  sp   A
Where   a. P# =’p1’  and  not  exits  (select   *
                                        Form   sp  B
                                         Where  B. p# =
‘P2’      and  B. S# = A.S#)
```

عملگرها union

شماره قطعاتي را بدست آوريد که وزن آنها بيشتر از ١۶ باشد يا توسط S2 توليد مي شود

```
Select     P#
From       P
Where      Weigh  E>16
Union
Select     P#
From       Sp
Where      S# = ‘S2’
```

دو رابطه را در صورتي مي توان اجتماع کردکه با

يکديگر سازگار باشند يعني الف) تعداد ستون هاي آنها يکسان باشد ب) ستون Ai رابطه اول با ستون Aiرابطه دوم هم ميدان باشد.

اين عملگر تکراري ها را حذف ميکند مگر اينکه از

Union All استفاده کنیم تکرار ان حذف نمي شود

نکته:

فقط اجتماع داريم ولي اشتراك نداريم چون از Select هاي تودرتو استفاده مي كتيم.

عملگر DELETE :

به وسيله اين دستور مي توان بعضي از سطر ها را طبق شرط هاي خاصي حذف كرد بر خلاف دستور Select كه بر روي جدول تغييراتي اعمال نمي كرد دستور Delete سطر هارا حذف مي كند.

مثال:

قطعاتي را كه وزنشان كمتر از ١٢را حذف كنيد ؟

```
.Delete
From     P
Where   Weight  < 12
```

مثال:

توليد به دست آمده از ساكنان C2 را حذف كنيد ؟

```
Delete
From    SP
Where  S#   IN  ( Select   S#
                        From    S
                            Where   City ='C2')
```

كل جدول SP را حذف كنيد؟

```
Delete
From    SP
```

دستور INSERT :

با این دستور مي توان اطلاعات را در جدول درج کرد به وسیله یك تاپل یا مجموعه از تاپلها این کار انجام مي شود .

P# P name Color City Weight

```
Insert
Into P( P# , City , Weight)
Values (P8 , C4 , 13)
```

اگربخواهیم همه فیلد ها را مقدار دهي کنیم نیازي به نوشتن نام فیلد ها نیست .

```
Insert
Into P
Where ( P9 , P ns , Red , C3 , 14)
```

مثال:

قطعات قرمز رنگ را در جدول P’ بریزید.

```
Insert into p’
        (select *
            From p
              Where p . color=’color’)
```

Create table

توسط این دستور مي توانیم جدول بسازیم

```
Create table R1(
        P# Smallint not nau
        Qty integer
         Primary Key P# )
```

مثال:

کد هر قطعه و تعداد تولید شده از آن را در جدول R1 بریزید؟

```
Create table R1(
        P# Small int not null
```

```
        Qty   integer;
        Primary   Key  (P#)
        Insert
        Info   R1 ( P#, Qty)
Select     P#, Sum (Qty)
    From      SP
    Group   by   P#
```

Up date (بهنگام کردن) :

این دستور براي ويرايش کردن جدول.

مثال :

در جدول P قطعاتي که قرمز رنگ هستند و نشان را ۵ کيلو اضافه کنيد .

```
Update   P
Set  Wight = Weight +5
Where  P. color = 'red'
```

مثال:

رنگ قطعه P2 را به زرد تقيير داده و شهر آن را ناشناخته کنيد.

```
Update     P
Set   Color  'Yen now'  , city = 'nun'
Where    P# = 'P2'
```

مثال:

تعداد قطعات توليد شده ساکنان شهر C2 را ۱۰ واحد کاهش دهيد.

```
Update      Sp
Set     Qty = Qty- 10
Where     Sp .S#   In ( Select   S#
                        From    S
                          Where    City = 'C2')
```

کد توليد کنندگان S1 بهSR تقيير دهيد؟

```
Update        S
Set   S# = 'Sp'
Where     S# = 'S'
Update     Sp
Set       S# ='S2 '
Where   S# = 'S'
```

Drop table : حذف جدول

Alter table

اين دستور جدولي كه در قبل ساخته ايم را مي توانيد ساختار آن را ويرايش كند.

```
Alter  table   Sp
Modify     (S#   integer )
Alter   table     Sp
Add    S name   Char  15
```

Create index

مثال:

يك index بر روي نام تهيه كنندگان ايجاد كنيد.

```
Crete    index    A b c      On    S(S name)
```

Drop Index

براي حذف Index كه ساخته ايم

```
Drop Index   A b c
```

Any

برابر است با IN

```
Select  p name
```

Form p

Where weight= Any(11 , 17)

All

براي هر يك از مقادير

مثال

نام تهيه كنندگاني را استخراج كنيد كه وضعيت آنها از همه توليد كنندگان ساكن c2بذرگتر باشد.

Select s name

From s

Where status >All (select status from s

Where city=’c2’)

تفاوتش با اين مثال اين است كه بالايي با تك تك مقايسه مي كند ولي پايين يك عدد مي شود

Select S name

Form S

Where status > (select Avg status from S)

وابستگي تابعي FD (function dependency)

صفت خاصه yاز رابطه Rبا صفت خا صه xاز رابطه Rوابستگي تابعي دارد اگر وفقط اگر در طول حيات رابطه به هر مقداري از xدر رابطه rدقيقا" يك مقدار از رابطه ق متناظر باشد.

مثال

در جدول Sستون cityوابستگي تابعي دارد وبه ستون S# زيرا در ازاء هر S#فقط يك cityوجود دارد.

در جدول SPشتون Qtyبه ستون #Sوابستگي تابعي ندارد زيرا ممكن است در ازاء يك #Sمثلا"S1"چند تا Qtyديده شود در جدول SPستون Qty به هر دو ستون #S,#Pوابستگي تابعي دارد.

نكته

اگر xكليد اصلي باشد حتي كليد كانديد باشد هر صفت خاصه ديگر از اين رابطه الزاما" با xوابستگي تابعي دارد.

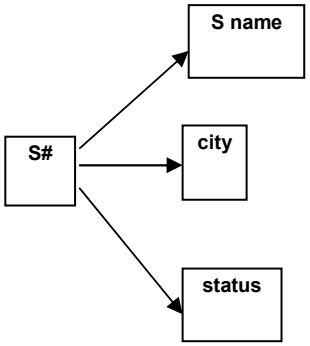
وابستگي تابعي كامل (full function dependency) FFD

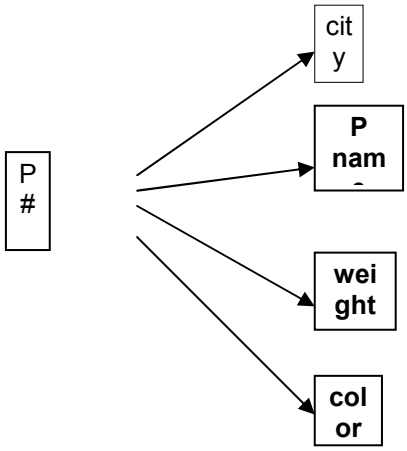
صفت خاصه yاز رابطه Rبا صفت خا صه xاز رابطه Rوابستگي تابعي كاملدارد اگرy با xوابستگي تابعي داشته باشد ولي با هيچ يك از زير مجموعه ها xوابستگي تابعي نداشته باشد.

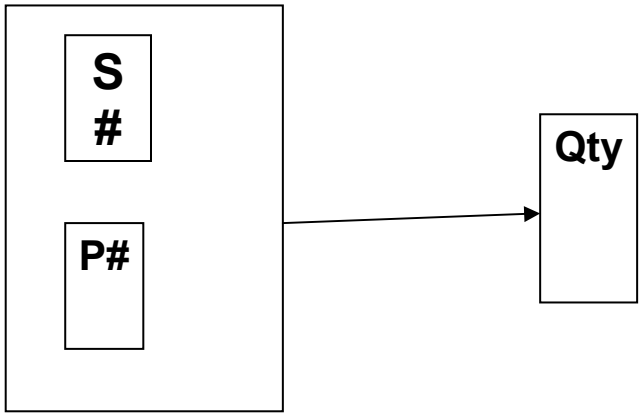
ستون Qtyدر جدول SPبه صفات خاصه #S,#Pوابستگي تابعي دارد يهني در ازاء هر #S,#Pفقط يك وجود دارد اما ستون Qtyبه هر يك از ستون هاي #S,#Pجداگانه وابستگي تابعي ندارد بهمين دليل وابستگي تابعي كامل بين,#Qty,S, #Pوجود داردادما در مثال فاميلي به كد ملي وشماره دانشجويي وابستگي دارد اما چون فاميلي به تك تك فيلدها كد ملي و شماره دانشجويي نيز وابسته است بنابرين كامل نيست.

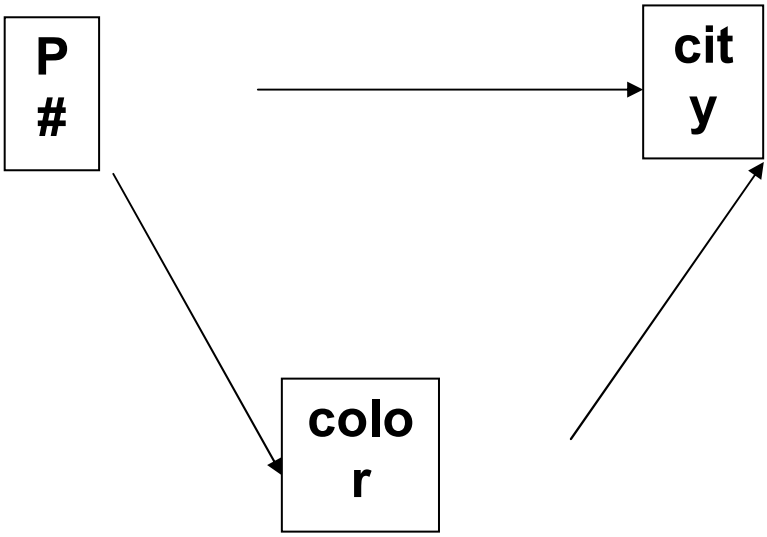
نمودار وابستگي تابعي

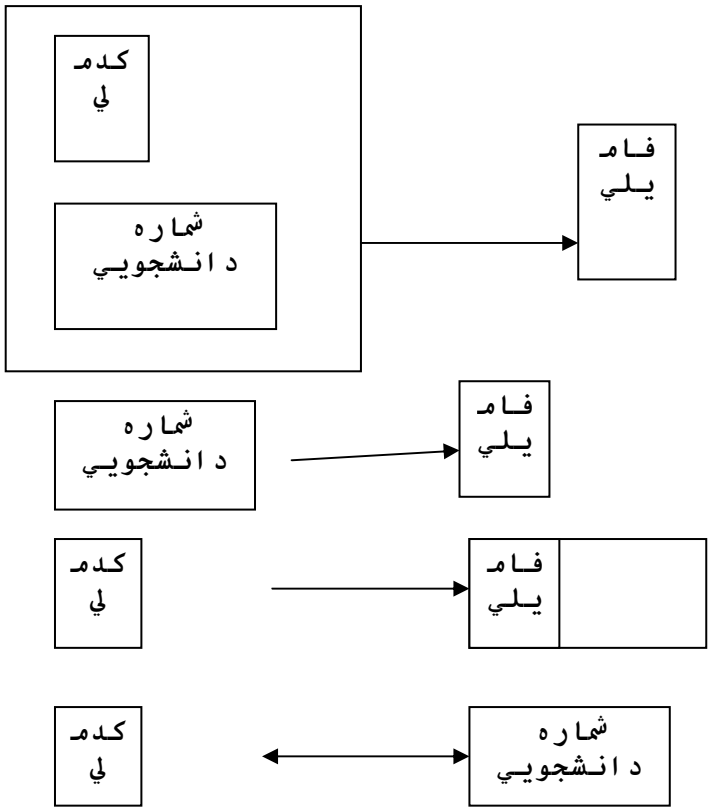
اين نمودارها وابستگي تابعي يك بانك را نشان مي دهند




P #
cit y
P nam
wei ght
col or


S #
P#
Qty


P #
cit y
colo r



نرما لسازي:

يكي از مهترين فرآيند ها در طراحي بانك اطلاعاتي فرمان سازياست . يك سئوال خيلي مهم براي طراحي بانك اين است كه با توجه به موجوديت ها و ارتباط بين آنها چند تا جدول بايد طراحي كرد و در هر جدول چه فيلدهاي بايد درست كنيم فرمان سازي براين سئوال ها جواب مي دهد . فرض كنيد در جداول sp فيلد STATUS را از جدول S به جدول SP

منتقل کنیم در این صورت فیلد STATUS براي توليد كنندگان به دفعات تكرار ميشود .

مي توانيم جداول SP را در يك جداول بريزيم در اين صورت بعضي سوالات ما نيازي به جوين و پيوند طبيعي ندارد .

تركيب اين سه جدول مشكلاتي رادر بر دارد البتهد باعث ميشود كه ضرب كارتزين و عمل جوين انجام نشود .

مشكلات:

الف) افزونگي داده ها (DATA REDUNDANCY) هر توليد كننده يا هر قطعه چندين بار نام آن تكرار شده است .

ب) وجود افزونگي در جدول باعث آنو مالي در تقيير داده ها مي شود مثلا تقيير دادن شهر S1 مستلزم جستجو درتمام جدول است .

ج) براي نشان دادن بعضي از اطلاعات از مقادير تهي NULL VALUE استفاده شده است.

سطوح نرمال سازي:

سطوح مختلف نرمال سازي به شكل مقابل است.

1NF

2NF

3NF

BCNF

4NF

5NF

DR NF

نکته:

رابطه هايي كه I NFهستند تعدادي شان 2NF هستند و از بين آنها تعدادي 3NF هستند . رابطه ايي كه 3NF باشد حتما 1NF و2NF هم است

فرم INF:

رابطه R , INF است اگر تمام صفحات خاص آن اتوميك باشند يعني تجزيه پذير نباشند مثلا جدولي كه يك فيلد تاريخ دارد كه خود از سه فيلد كوچكتر (سال , ماه , روز) تشكيل مي شود INF نيست

نكته

در ACCSS تمام جداول INF هستند .:

فرم 2NF:

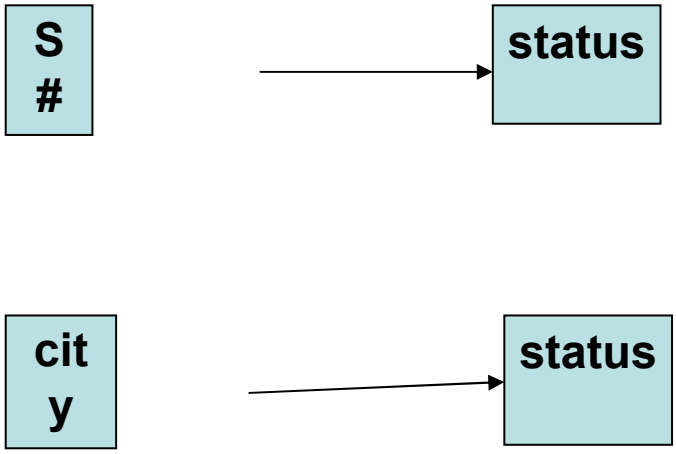
رابطهR , 2 NF است و اگر فقط INF باشد و هر صفت خاصي غير كليد يا كليد اصلي وابستگي تابعي كامل داشته باشد.

فرض كنيد جدولي به صورت زير داريم:

در اين جدول تركيب S#,P#كليد اصلي است فرض مي كنيم وضعيت يك تهيه كننده از روي شهر آن تعيين مي شود يعني statusبه cityوابسته است.

جدول 2NF,firstنيست زيرا city, statusفقط به S#وابسته اند و تركيب S#,P#كه همان كليد اصلي است وابستگي تابعي ندارد.

براي اينكه يك جدول كه 2NFنيست2NFشود آن را به جداول ديگري تجزيه مي كنيم جدول firstرا به دو جدول زير تجزيه مي كنيم



در جدول Secondاگر فیلدstatusتغییر کند باید فیلد city نیز تغییر کند و بر عکس بنابرین عمل ویرایش در این جدول آنومالي دارد

.

فرم 3NF:

رابطه 3NF,Rاست اگر وفقط اگر همه صفات خاصه غیر کلید اصلي:

الف)متقابلا" به یکدیگر وابسته نباشد.

ب) با کلید اصلي رابطه Rوابستگي تابعي کامل داشته باشد.

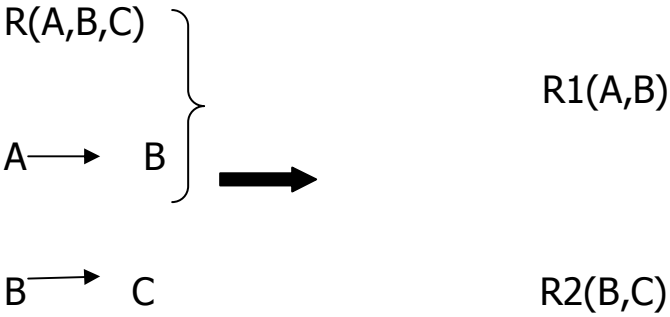
جدول Second, (3NF) نیست زیرا فیلدهايstatus, cityهر کدام به S#وابستگي تابعي کامل دارد اما خودشان نیزبه هم وابسته اند

براي اینکه جدول 3NF secondشود آن را به دو جدول تجزیه مي کنیم به صورت زیر

قضیهheath:

رابطه Rکه داراي سه مجموعه صفت A,B,Cاست را در نظر بگیرید اگر A → BوA → Cوابسته باشد آنگاه مي توانیم رابطه Rرا به دو رابطه R1(A,B),R2(A,C)تجزیه کنیم.

قضیه ریسانن:



قضیه ریسانن جداول را 3NFمي كند.

فرم BCNF:

تبدیل کردن جداول به 3NFباعث افزونگي جداول مي شود که باعث افت کارایي سیستم در عمل بازیابي اطلاعات مي شود زیرا در این حالت باید عمل پیوند داده ها به دفعات استفاده شود.

فرم 3NFدر موارد زیر ممکن است مشکل بوجود آورد:

الف)وقتي رابطه داراي چند کلید کاندید باشد.

ب)وقتي که کلیدهاي کاندید رابطه مرکب باشد.

ج)وقتي که کلید هاي کاندید بایکدیگر اشتراك صفت داشته باشند.

دترمینان :

هر صفت خاصه ایي که صفت خاصه دیگر با آن وابستگي تابعي کامل داشته باشد دترمینان گفته
مي شود.

یعني Aبه B وابستگي تابعي کامل دارد در این صورت ⟶

A B

به A دترمينان مي گويند.

رابطه R BCnfاست اگر و فقط اگر هر دترمينان كليد كانديد باشد يعني تمام صفات خاصه به كليد كانديدها وابسته باشند .
كيلد اصلي خود يكي از كليد هاي كانديد است .
رابطه اي كه 3NFنباشد BCNFنيز نخواهد بود و هر رابطه اي كه BCNFباشد حتما" 3NFاست
اگر در يك جدول دو صفت خاصه كه كليد كانديد نباشد ولي وابستگي تابعي داشته باشند جدول BCNF نخواهد بود .
BCNFازتجزيه جداول جلوگيري مي كند .