جلسه اول

نسل های ذخیره و باز یابی اطلاعات

1- نسل اول-نسل فایل های ساده ترتیبی

ویژگی این نسل:

1- رسانه ی ذخیره سازی معمولا نوار بوده است

2- امکان دسترسی مستقیم وجود نداشت

3- خبری از نرم افزار واسط وجود نداشت

4- ساختار فیزیکی و منطقی فایل ها یکسان بود

5- برنامه های کاربردی مستقیما با جنبه های فیزیکی ذخیره سازی در ارتباط بودند

6- هرگونه تغییر در ساختار فایل ها سبب تغییر در برنامه های کار بردی می شد

7- اشتراک داده ها و همینطور تدابیر امنیتی وجود نداشت

2-نسل دوم-نسل شیوه های دستیابی

این نسل را باید پیدایش نرم افزارهای موسوم به شیوه های دستیابی و پیدایش دیسک دانست

Access Method: نرم افزاری است که به جنبه های فیزیکی محیط ذخیره سازی و عملیات این محیط می پردازد به نحوی که دیگر برنامه های کاربردی نیازی به پرداختن به این جنبه ها را ندارند.

ویژگی این نسل

1- نرم افزار واسط برای ایجاد فا یل ها

2- پردازش مستقیم امکان پذیر است

3- امکان دسترسی به رکوردها (نه فیلد ها) وجود داشت

4-پدیده افزونگی در حدی کم شد

3-نسل سوم-نسل (Data Base Management System: DBMS)

DBMS مانند حصاری نفوذ ناپذیر اطراف بانک اطلاعاتی را محاصره کرده و هر گونه دستیابی به بانک اطلاعاتی می بایست از طریق DBMS صورت گیرد تنها کسانی که می توانند دور از چشم DBMS به داده ها دسترسی داشته باشند مدیر و برنامه سازان مجاز بانک اطلاعات هستند.

DBA چه کسی است؟ تعیین کننده خط مشی کلی پایگاه داده ها است به عبارتی کسی است که مسئولیت طراحی و ایجاد پایگاه داده و تصمیماتی مانند مجوز استفاده کاربران از بانک،انجام تغییرات در بانک، و ... را به عهده دارد و جهت پیاده سازی تصمیماتش از DBP استفاده میکند.

نکته: DBA مسئول پیاده سازی نظرات DA یا مدیر داده ها است.

DA چه کسی است؟مسئول مدیریت منابع شامل طراحی،توسعه و نگهداری استانداردها، و خط مشی ها و روال ها و طراحی مفهومی و منطقی پایگاه داده ها را به عهده دارد

4- نسل چهارم- نسل بانک معرفت یا پایگاه شناخت(Knowledge Base)

در این نسل با بهره گیری از فایل های داده،منطق صوری، مفاهیم هوش مصنوعی،سیستم های خبره، پردازش زبان طبیعی، سیستم هائی ایجاد شد که قادرند از واقعیات ذخیره سازی شده به طور منطقی استنتاج کنند.

فرق مابین بانک های اطلا عاتی و بانک های معرفت چیست(فرق KB با DB)؟ بانک معرفت مجموعه ای است از واقعیت های ساده و قواعد عام که به طور صریح بیان شده اندر حالی که بانک اطلاعاتی مجمو عه ای است از تعدادی زیاد واقعیات ساده که به طور صریح بیان شده اند همراه با تعداد نسبتا کمی از قواعد عام که به طور ضمنی بیان شده اند.

چه نیازی به پایگاه داده (DB) می باشد؟ سازمان ها نیاز به مدیران خوب دارند، مدیران خوب جهت تصمیم گیری ها نیاز به اطلاعات خوب دارند، اطلاعات از داده های خام تولید می شوند،مدیریت داده ها شامل جمع آوری، ذخیره و باز یابی می بایست توسط DB انجام شود.

DB را می توان مانند یک کابینت الکترونیکی در نظر گرفت که شامل یک سری قفسه هائی است که مدیریت این قفسه ها توسط DBMS انجام می گیرد

تعاریف رایج برای داده:

1- نمایش پدیده ها و مفاهیم به صورت صوری و مناسب برای بر قراری ارتباط یا پردازش.

2- داده عبارت است از واقعیتی که می توان از آن واقعیت دیگری را استنباط کرد.

3- هر نمایشی اعم از کاراکتری یا کمیت های قیاسی که معنائی به آن قابل انتساب است.

4- به مقادیر صفات خاصه داده گویند.

موجودیت: به هر فرد، شئی یا مفهومی که می خواهیم راجع به آن اطلاعاتی داسته باشیم

رابطه(Relation ship): به ارتباط بین موجودیت ها در یک محیط عملیاتی گفته میشود،مثلا ارتباط بین دانشجویان و اساتید در محیط عملیاتی دانشگاه (مثلا دانشجوی A با چه اساتیدی در این ترم درس دارد).

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

رکورد: مجموعه ای از فیلد های مرتبط به هم است

فایل: مجموعه ای از چند رکورد است.

تعاریف رایج برای اطلاع:

1-اطلاع،داده پردازش شده می باشد

2-اطلاع،داده سازمان یافته ای است که شناختی رامنتقل می کند

3- اطلاع معنائی است که انسان از طریق یکسری قراردادها به داده منتسب می کند

نکته:داده همان مقدار واقعا ذخیره شده و اطلاع معنای داده است. یعنی اطلاع و داده باهم فرق دارند. اطلاع دارای خاصیت ارتباط دهندگی و انتقال دهندگی دارد،در حالی که داده این خواص را ندارد

تعاریف رایج برای پایگاه داده ها(DB):

- مجموعه ای است از داده های ذخیره شده (در مورد انواع موجودیت های یک محیط عملیاتی و ارتباط بین آنها) به صورت مجتمع و مبتنی بر یک ساختار خاص ،تعریف شده به صورت صوری با حداقل افزونگی،تحت کنترل متمرکز،مورد استفاده یک یا چند کاربر به طور اشتراکی و همزمان.
- مجموعه ای از داده های منطقا به هم مرتبط که برای پاسخ گوئی به نیاز های اطلاعاتی یک سازمان طراحی شده اند.

تعریف شده به صوری:یعنی سیستم به کاربران امکان بدهد تا داده های خود را آنگونه که خود می بینند ، به صورت انتزاعی و به دور از جنبه های پیاده سازی و نشست فیزیکی آنها را روی رسانه تعریف کنند.

مجتمع و مبتنی بر یک ساختار: به این معناست که کل داده های عملیاتی محیط مورد نظر در یک ساختار مشخص و به صورت یکجا ذخیره شده باشند،لازمه ی هر تجمعی وجود یک ساختار است.

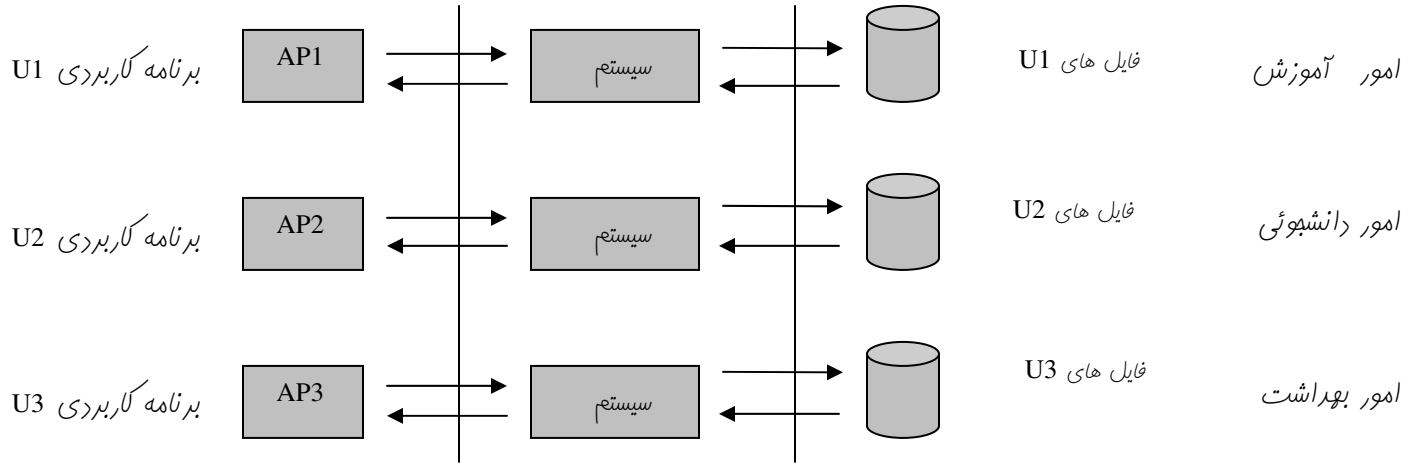
افزونگی: تکرار ذخیره سازی داده در تمام نمونه های مختلف یک نوع رکورد به عبارتی تکرار یک سری مقادیر در بیش از یک نقطه فایل

روش های ایجاد سیستم های کاربردی:

1. **فایلنگ(سنتی)**
2. **پایگاهی**

مثال : فرض کنید می خواهیم سیستم امور آموزش،امور دانشجوئی و امور بهداشت یک دانشگاه را مکانیزه کنیم، یک بار به روش کلاسیک و یکبار با استفاده از پایگاه داده مسئله را مکانیزه می کنیم.

الف. روش فایلنگ(سنتی)



در این روش هر یک ازبخش های سه گانه به طور جداگانه سیستم خاص خود را ایجاد می کنند(مثل شکل بالا)

معایب این روش:

- عدم وجود محیط مجتمع ذخیره سازی
- عدم وجود سیستم کنترل متمرکز
- عدم ضوابط ایمنی کارا

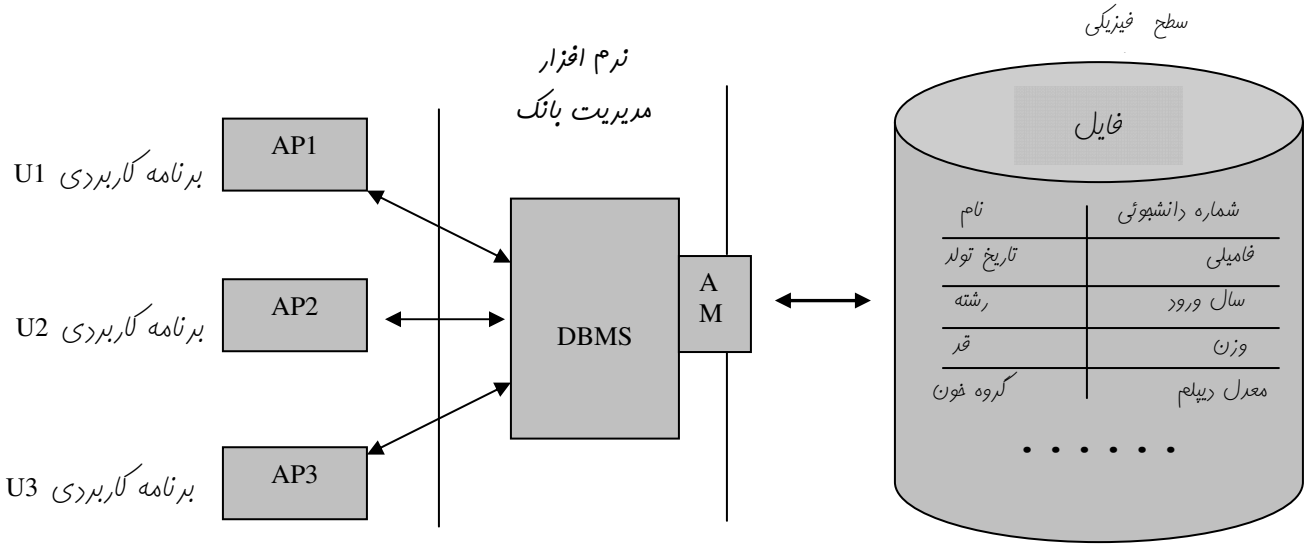
- عدم امکان اشتراکی شدن داده ها
- تکرار در ذخیره سازی اطلاعات
- مصرف نامناسب امکانات سخت افزاری و نرم افزاری
- حجم زیاد برنامه سازی
- وابسته بودن برنامه های کاربردی به محیط ذخیره سازی داده ها

ب. روش پایگاهی

مراحل کار در روش پایگا هی:

در این روش نیازهای اطلاعاتی تمامی قسمت ها مورد مطالعه قرار می گیردتا بتوان یک سیستم یکپارچه(integrated) طراحی کرد. داده های سازمان مدلسازی معنائی(SDM) می شوند و مشخصات سیستم یکپارچه تعیین میشود.برای سیستم مدیریت متمرکز از یک یا چند DBMS استفاده می شود. طراحی پایگاه داده ها در سطوح لازم انجام می شود و کاربران هر قسمت،پایگاه داده های خود را تعریف می کنند و با آن کار می کنند، در واقع در روش پایگاهی یک محیط ذخیره سازی واحد، مجتمع و اشتراکی، تحت کنترل متمرکز وجود دارد که کاربران بر اساس نیاز خاص خود، پایگاه خود را تعریف می کنند و هر کاربر تصور می کند که پایگاه خاص خود را دارد.

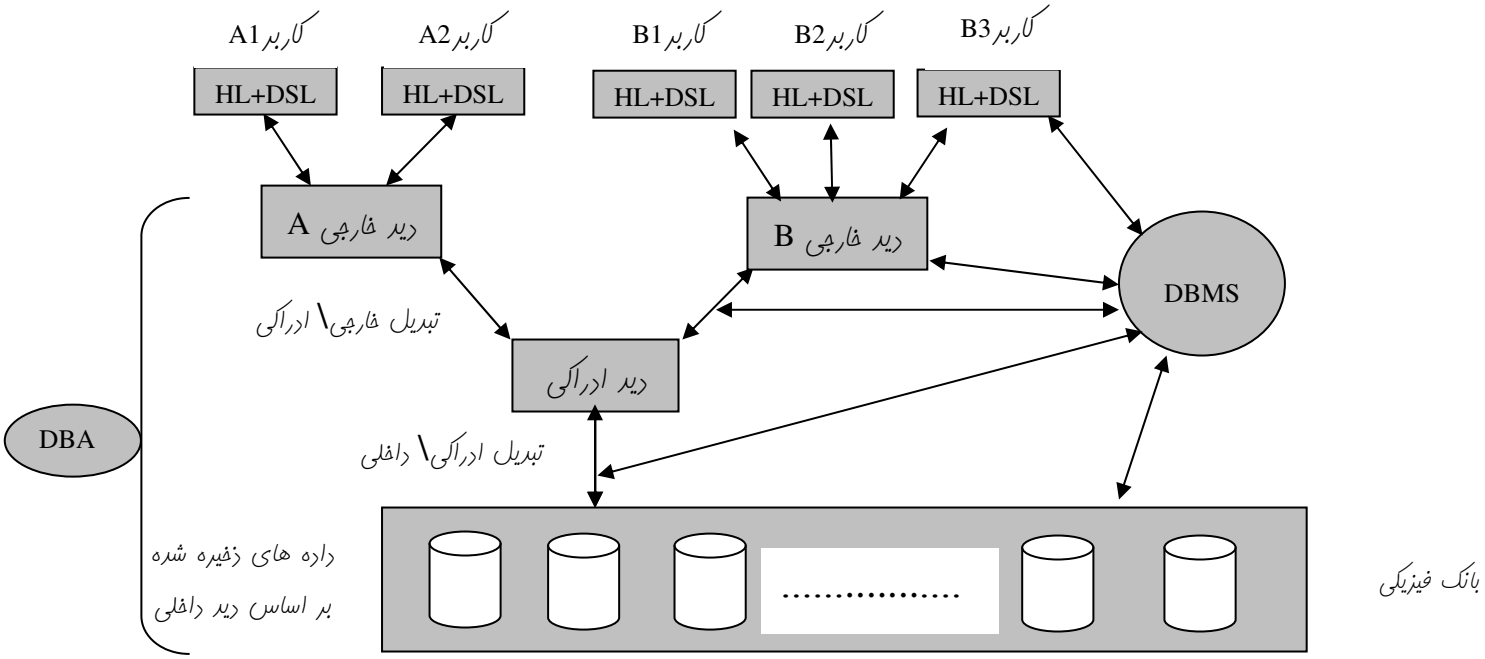
- کاربران در روش پایگاهی به طور همزمان از سیستم استفاده می کنند.
- در روش پایگاهی نسبت به داده های ذخیره شده؛ تنوع و کثرت دید وجود دارد.



پایان جلسه اول

جلسه دوم

معماری پایگاه داده ها



سه نوع دید در این معماری داریم .

الف) **دید خارجی یا دید کار بران**(External view)**:** دید خاص کاربر است از داده های ذخیره شده در بانک، هر کاربر دید خاص خود را دارد، همچنین چند کاربر می توانند دید یکسانی باشند، مثل A_1 و A_2 که دید خارجی A را دارند، همانند دید ادراکی دید خارجی نیز برای معرفی شدن نیاز به یک ساختار یا مدل داده ای دارد.

ب) **دید ادراکی یا مفهومی**(Conceptual view)**:** دید طراح بانک است از داده های ذخیره شده در بانک.یعنی داده های انواع موجودیت ها، و ارتباط بین آنها، آنگونه که طراح می بیند. دید طراح دیدی است جامع همه دید های کاربران(اجتماع دید های خارجی) و در عین حال متفاوت با هر یک از دید ها

ج) **دید داخلی یا فیزیکی**(Internal View)**:** به جنبه های ذخیره سازی محیط فیزیکی مانند نوع رکورد، نوع شاخص گذاری و ... گویند که همان بحث ذخیره باز یابی است

انواع نگاشت یا (Mapping)

الف) نگاشت های خارجی \ ادراکی:

سبب میشود تغییرات در پایگاه داده از دید کاربران مخفی بماند. به عنوان مثال اگر جدولی دارای سه ستون باشد و ستون چهارمی به آن اضافه شده باشد این جزئیات از دید کاربر مخفی بماند و بنابراین می توان ستون اضافه شده را از طریق برنامه کاربردی جدید(دید جدید) در اختیار کاربران مورد نظر قرار داد.

هدف: حفظ استقلال داده ای منطقی(مخفی ماندن تغییرات در سطح ادراکی از کاربران)

ب) نگاشت ادراکی \ داخلی: این نگاشت سبب می شود تغییر در رسانه یا ساختار فایل از سطح مفهومی مخفی بماند، در این نگاشت هدف حفظ استقلال داده ای فیزیکی است، که به معنای مصون ماندن تغییرات رسانه ذخیره سازی یا همان محیط فیزیکی از دیدکاربران می باشد.

نکته: زبان میزبان (HL)**:**زبان میزبان می تواند یکی از زبان های سطح بالا مانند پاسکال، C ، ایدا و ... باشد

نکته: DBMS ای که تعداد HL های مورد پذیرش آن زیاد باشد مطلوب تر است چون مو جب تنوع کاربردها و کاربران می شود.

زبان داده ای فرعی (DSL):یک زبان بیانی می باشد که میهمان یک زبان سطح بالا می باشد .

زبان بیانی:این زبان ها بر خلاف زبان های رویه ای هستند که در آن کاربر می گوید چه می خواهد ولی رویه انجام کار را بیان نمی کند ، برعکس C و پاسکال که رویه ای هستند و کاربر باید رویه انجام کار را بگوید.

انواع DSL : ادغام شده – ادغام نشده

در DSL ادغام شده DSL میهمان یک HL می باشد مانند استفاده ازSQL در دلفی.

در DSL ادغام نشده ، DSL به طور مجزا استفاده می شود مانند FoxPro یا Access

اجزای DSL

دستور های این زبان به سه قسمت تقسیم می شوند

الف) دستورات تعریف داده ها (Data definition Language:DDL)

ب)دستورات کنترل داده ها(Data control language:DCL)

ج) دستورات عملیات(پردازش)روی داده ها (Data Manipulation Language:DML)

معماری بانک اطلاعات به صورت خلاصه

تصویر خارجی	کاربر 1 کاربر 2 ..... کاربر n	دید های کار بران مختلف(Views)
تصویر ادراکی عام	نمو دار های EER ،NIAM و موجودیت 1 ارتباط موجودیت 2	کل بانک بدون توجه به مدل خاصی
تصویر ادراکی خاص	مدل شئی گرا مدل شبکه ای مدل سلسله مراتبی مدل جدولی A B C	کل بانک در قالب مدل انتخابی
تصویر فیزیکی	.....	کل بانک روی رسانه

در واقع در یک بانک اطلاعات که در مرحله ی بهره برداری است تصویر ادراکی عام فقط در حد یک سری Document می باشد.

Data Dictionary: هر گاه طراح بانک برای مفهومی نامی را انتخاب کند معنای آن را همراه با فرمت آن در DD یاد داشت می کند .

مثلا اگر برای یک جدول نامی انتخاب می شود معنای آن را همراه با فرمتش در DD ذخیره می کنند .

نکته:به اطلاعات موجود در دیکشنری داده ها اصطلا ما فرا داده یا دادگان (Meta data) می گویند که به معنی داده در مورد داده است.

کاتالوگ سیستم: در بردارنده اطلاعاتی همچون مجوز دستیابی کاربران- تاریخ ایجاد داده ها ، تاریخ آخرین ویرایش ، تعداد پرونده ها ، اندازه جدول یا شئی و ... می باشد.

نکته: لغت نامه داده ها زیر مجموعه کاتالوگ سیستم است ولی به دلیل کاربرد ویژه آن مجزا شده و برای کار با آن نرم افزار خاصی طراحی شده است.

شمای بانک اطلاعاتی: تشریح کلی پایگاه داده ها را شمای بانک اطلاعاتی گویند. به عبارت دیگر ساختار های بانک را بدون در نظر گرفتن محتویات آن ، شمای بانک اطلاعاتی گویند مثلا تعداد جداول ، تعداد فیلد ها به شمای بانک مربوط می شود. تعداد سطرها از آنجا که به تعداد داده ها مربوط است به شمای بانک مربوط نمی شود

درستی داده ها و پردازش در پایگاه داده ابعاد مختلفی دارد.

امنیت(Security)

جامعیت (integrity)

امنیت یعنی حفظ پایگاه داده ها در مقابل خطراتی از قبیل آتش سوزی و جلوگیری از دستیابی غیر مجاز کاربران .

جهت برآورده شدن امنیت می توان از تکنیک هائی همچون کلمه عبور در رمزنگاری مثلا تبدیل موجودی حساب های بانکی به مقدار منفی و محاسبه موجودی واقعی به هنگام نیاز استفاده کرد.

جامعیت: یعنی صحت داده ها و پردازش ها و پیروی از مقررات سیستم . نوعی از جامعیت بنام سازگاری یا consistency معروف است به طوری که اقلام داده ها و نسخه های مختلف نباید باهم در تضاد باشند به عنوان مثال موجودی واقعی نباید منفی باشد یا به عنوان مثال دیگر اگر فیلدی یا فیلد هائی در یک رکورد خاص در یک محل تغییر کردند، همان فیلدها در جاهای دیگر نیز تغییر کنند .

تراکنش:

به هر برنامه ای که در یک محیط بانک اطلاعاتی اجرا شود تراکنش گویند یک تراکنش مجموعه ای از دستورات read, Write, Commit Abort میباشد .اگر کلیه دستورات یک تراکنش به درستی انجام شود،گوئیم تراکنش اجرای موفق داشته است(Commit) در غیر این صورت Abort شده است.

فرق تراکنش با برنامه های که در محیط غیر بانکی اجرا می شوند در این است که تراکنش همواره تسلیم DBMS میشود و DBMS در اعمال هر گونه تصمیمی از جمله به تعویق انداختن اجرا و ساقط کردن آن آزادی عمل دارد.

هر تراکنش می بایست 4 اصل زیر را بر آورده کند تا صحت و جامعیت بانک اطلاعاتی بر قرار شود.

1- یکپارچگی (Atomicity) 2- سازگاری (Consistency) 3- مجزا بودن (Isolation) 4- پایائی(Durability)

یکپارچگی:

به قانون هیچ یا همه معروف است یا تمام دستورات تراکنش اجرا شوند یا هیچ کدام از آنها نباید اجرا شود.

مثال: تراکنشی را در نظر بگیرید که می خواهد مبلغی را از حسابی بر روی یک سیستم برداشته و به حساب دیگری بر روی سیستم دیگر انتقال دهد. اگر بعد از برداشت مبلغ از حساب اول ارتباط با سیستم دوم قطع شود مطابق این اصل می بایست موجودی به حساب اول برگردانده شود.

سازگاری:

دستورات یک تراکنش می بایست صحیح باشند، به عبارتی دستورات یک تراکنش چنان باشند که سیستم را از یک حالت صحیح به حالت صحیح دیگری ببرند. مثلا شخصی نتواند بیش از موجودی خوداز حسابش برداشت کند .

مجزا بودن:

دستورات هر تراکنش می بایست طوری باشد که کوئی هر تراکنش در انزوا اجرا می شود به عبارتی دستورات یک تراکنش اثر مخرب روی دیگر تراکنش ها نداشته باشد(در بحث تراکنش های همروند این بحث وجود دارد) . همروند توسط بخشی از DBMS به نام واحد کنترل همروندی کنترل می شود.

پایائی:

بدین معناست که اثرات تراکنش هائی که به مرحله Commit رسیده اند پایدار و ماندنی باشد. به عنوان مثال تراکنشی که مبلغی را به حسابی واریز کرده ، حتی در صورت وقوع آتش سوزی در آن شعبه از بانک ، مشتری متضرر نشود، بنا براین می بایست قبل ازاعلان پایان اجرای موفق (Commit) نتایج یا تغییرات در جاهای دیگری هم ثبت شوند . یکپاچگی و پایائی توسط بخشی از DBMS به نام مدیریت باز گرد (Recovery Management) اعمال می شود .

مدل ادراکی عام:

پایگاه داده بدون در نظر گرفتن جنبه های پیاده سازی یا ارائه بانک اطلاعاتی در قالب مدلی بدون نگرانی در مورد دغدغه های پیاده سازی.

دید ادارکی خاص: در این دید جزئیات پیاده سازی بانک اطلاعاتی بررسی می شود، مثلا می توان از مدل رابطه ای (ایجاد جدول) یا شبکه ای برای پیاده سازی بانک اطلاعاتی استفاده کرد

مدل رابطه – موجودیت ER (Entity-Relationship:) :

هدف از ER بیان پایگاه داده فارغ از جزئیات پیاده سازی است که اولین بارتوسط چن در دانشگاه MIT ارائه شد که بعدها بنام EER (Extend ER)نامیده شد .

الف) موجودیت را با مستطیل نمایش می دهند

ب) صفت را با بیضی نمایش می دهند

ج) رابطه را با لوزی نمایش می دهند

د) رابطه بین صفت و موجودیت را با یک خط نمایش می دهند

پایان جلسه دوم

جلسه سوم

کلید:

کلید عبارت است از یک یا چند صفت خاصه که در یک موجودیت منحصر به فرد باشد. مثلا در موجودیت دانشجو شماره دانشجوئی کلید است . چون هر دانشجو شماره یکتا دارد. ولی نام نمی تواند کلید باشد.

☐ در نمودار EER زیر صفت یا صفات کلیدی یک خط می کشند.

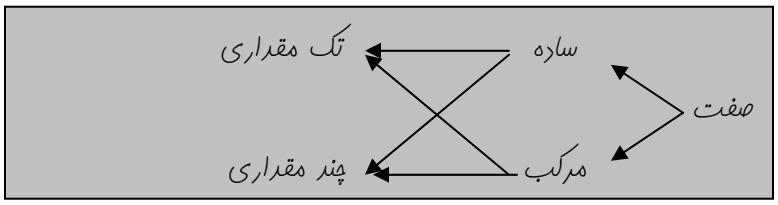
صفت ساده و مرکب:

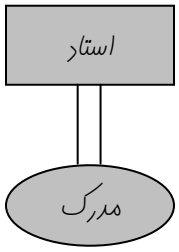
بعضی از صفت ها ساده هستند مثل شماره دانشجوئی ولی بعضی از صفت ها مرکب (تجزیه پزیر) هستند مثل آدرس که خود از صفت های شهر ، خیابان ، کوچه ، و پلاک تشکیل یافته است . در واقع صفت مرکب صفتی است که هم خودش معنی دار است و هم بخش هائی از آن .در بانک اطلاعاتی رابطه ای (جدولی) صفت مرکب نداریم.

صفت تک مقداری یا چند مقداری:

بعضی از صفات چه ساده وچه مرکب فقط میتوانند یک مقدار را اتخاذ کنند که به این صفات ، صفات تک مقداری گفته می شود . مانند شماره دانشجوئی که نمیتواند بیش از یک مقدار داشته باشد. این صفات در نمودار ER به صورت معمول نمایش داده می شوند.

صفاتی وجود دارند که میتوانند چندین مقدار بگیرند مانند صفت مدرک در موجودیت استاد که می تواند شامل مقادیر لیسانس ، فوق لیسانس و یا دکترا را در خود بگیرد. صفات چند مقداری در نمودار ER به صورت دو خط به موجودیت وصل می شوند





صفت مشتق:

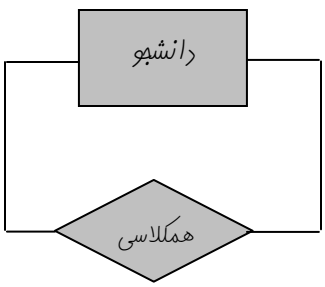
صفتی است که وجود خارجی ندارد به عنوان مثال صفت سن ، برای محاسبه صفت سن میتوان صفت تاریخ تولد را در نظر گرفت واز روی این صفت سن را محاسبه نمود. صفت مشتق را در نمودار ER با نقطه چین به موجودیت مورد نظر وصل میکنند.

درجه ارتباط:

درجه ارتباط برابر تعداد موجودیت هائی است که در آن ارتباط مشارکت دارند. معمولا درجه ارتباط **1** یا **2** و یا **3** است و درجات بالاتر به ندرت استفاده می شوند.

☐ در ارتباط درجه یک فقط یک نوع موجودیت شرکت دارد (شکل 1)

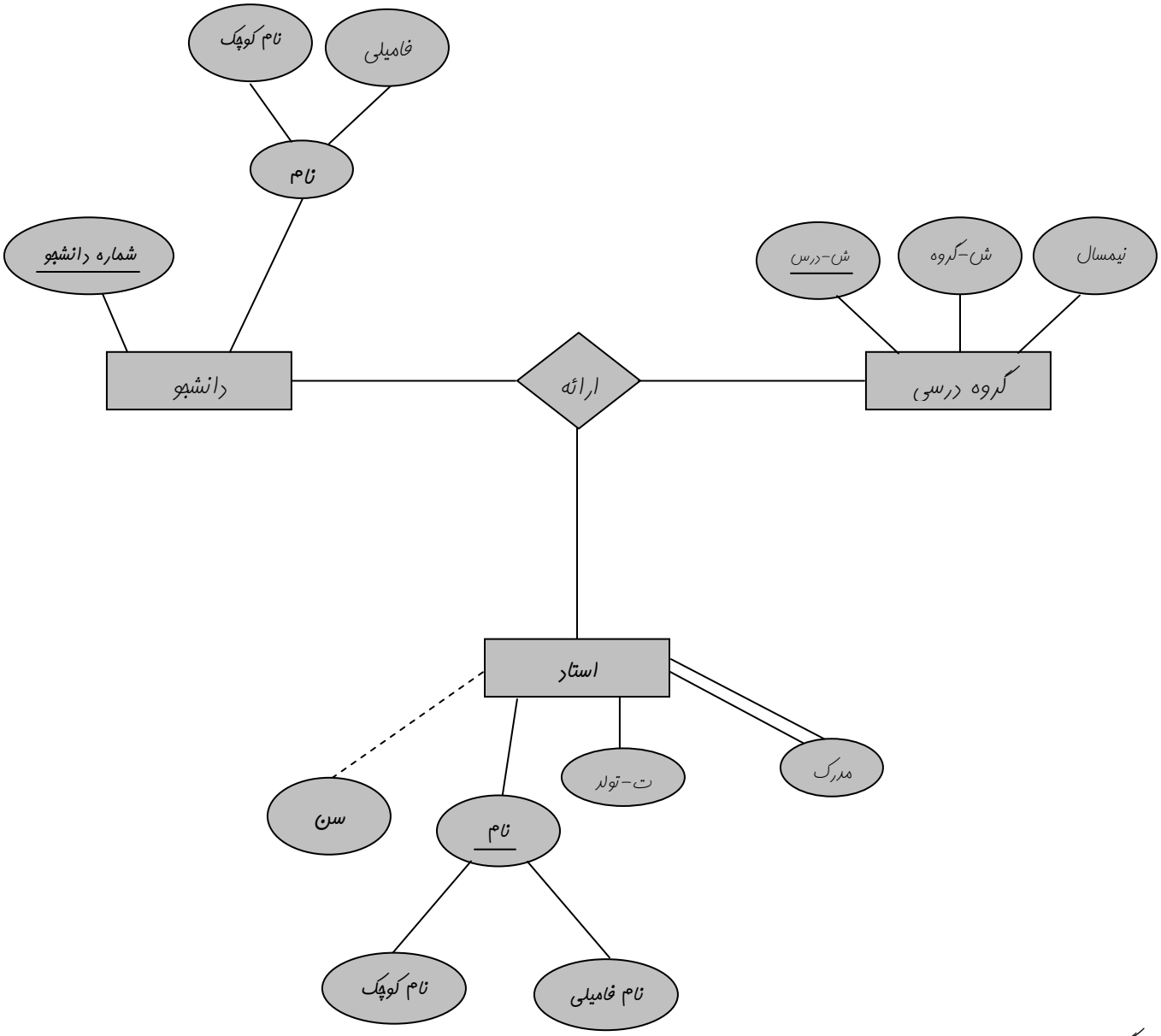
☐ در ارتباط درجه دو ،دونوع موجودیت وجود دارد(شکل 2)



شکل 1

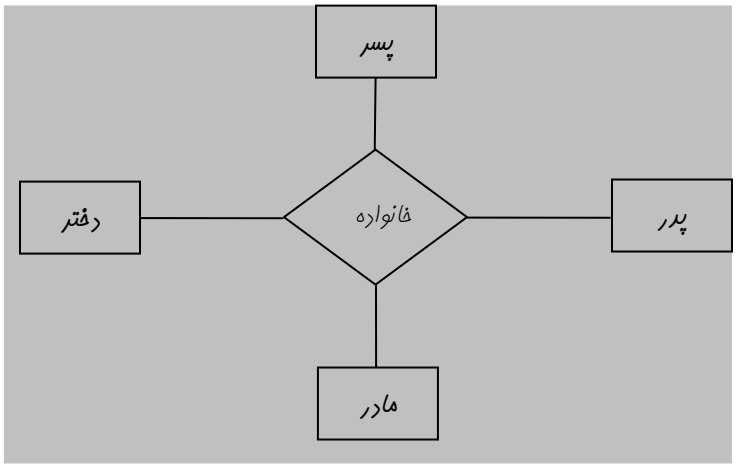
شکل 2

☐ در ارتباط درجه 3 ، سه نوع موجودیت در ارتباط شرکت دارند.



تصمیم گیری در مورد صفت مشتق به عهده طراح است به عنوان مثال صفت معدل برای فارغ التحصیلان غیر مشتق می باشد چون تغییر نمی کند ولی برای دانشجویان بهتر است معدل یک صفت مشتق باشد زیرا مرتبا با گزراندن دروس بیشتر تغییر میکند.

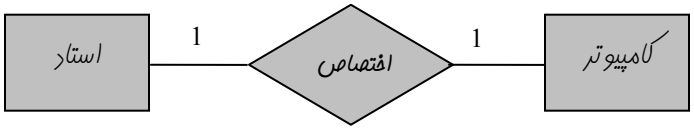
☐ در ارتباط درجه **4**، چهار نوع موجودیت در ارتباط شرکت دارند



ارتباط از نظر نوع اتصال(Connectivity)برسه نوع است1-1 . 1-M . N-M

منظور از اتصال در واقع تعداد نمونه های شرکت کننده در ارتباط است

- **در یک دانشگاه ممکن است هر استاد یک کامپیوتر اختصاصی داشته باشد**



- **یک استاد چندین دانشجو را راهنمائی می کند**

کاردینالیتی(Cardinality): بیانگر حداقل و حداکثر تعداد موجودیت هائی است که در یک ارتباط شرکت دارند. در شکل فوق(0,10) یعنی یک استاد ممکن است راهنمای هیچ دانشجوئی نباشد و حداکثر ده دانشجو را راهنمائی کند . همچنین در شکل فوق (0,1) یعنی یک دانشجو ممکن است استاد راهنما نداشته باشد و حداکثر توسط یک استاد راهنمائی شود

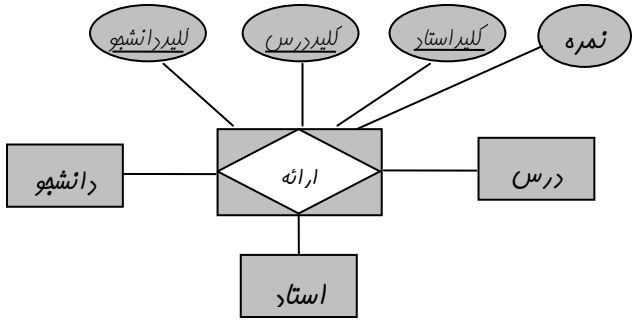
شرکت اجباری و اختیاری موجودیت در ارتباط:

اگر وجود یک پدیده در ارتباط الزامی نباشد گوئیم آن موجودیت وجودش در ارتباط اختیاری است و مقابل پدیده و کنار رابطه علامت ○ می گذاریم.

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

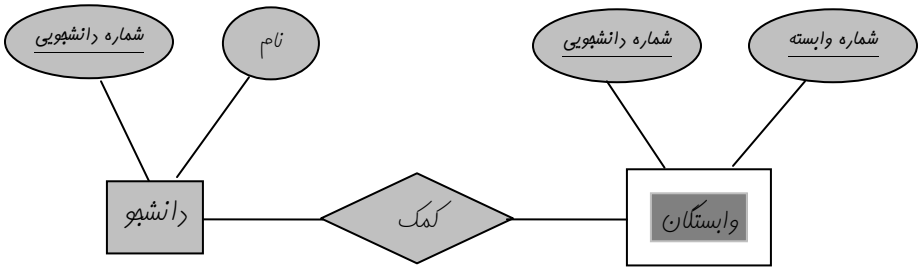
در این شکل تدریس استاد اختیاری است- اگر دایره نداشته باشیم ارتباط اجباری است

صفت در ارتباط: ارتباط ها نیز می توانند صفت داشته باشند . مثلا صفت نمره در نمودار بانک اطلاعات دانشگاه می تواند صفت ارتباط ارائه باشد شاید تصور شود که نمره مربوط به پدیده دانشجو یا درس است. ولی این تصور غلط است زیرا یک دانشجو چند نمره (در دروس مختلف) ویک درس نیز چند نمره (برای دانشجویان مختلف) دارد. از طرف دیگر ممکن است دانشجویی در درسی از یک استاد نمره 7 ودر ترم بعد از استادی نمره 14 گرفته باشد . بنابراین صفت نمره را باید به ارتباط ارائه که سه پدیده دانشجو ، درس و استاد را به هم مرتبط می کند نسبت داد. چنین ارتباطهایی با یک لوزی درون مستطیل نشان داده می شوند و کلید آنها کلیدهای همه پدیده های مربوطه را شامل می شود مانند شکل



وابستگی وجودی(existence dependency):

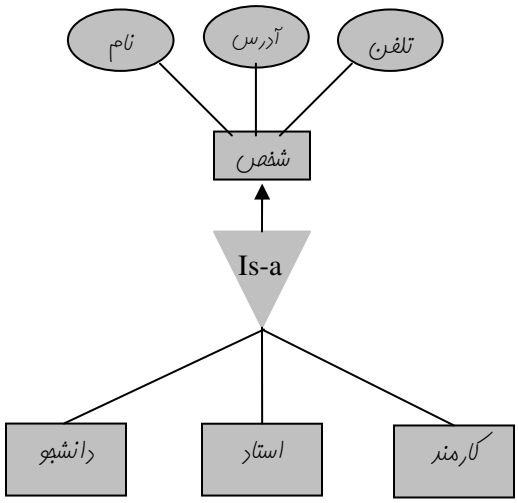
اگر در یک بانک اطلاعاتی وجود یک موجودیت وابسته به موجودیت دیگری باشد که در صورت حذف و تغییر موجودیت اصلی این موجودیت نیز تغییر کند ، این نوع وابستگی را وابستگی وجودی گفته و به پدیده وابسته، موجودیت ضعیف(Weak entity) گویند. که موجودیت ضعیف کلید موجودیت اصلی را دربردارد تا هر گونه تغییر یا حذف در موجودیت اصلی به موجودیت وابسته اعمال شود . موجودیت وابسته با دو تا مستطیل نمایش داده می شود.



اشتراک صفت (ارث بری)(IS-a):

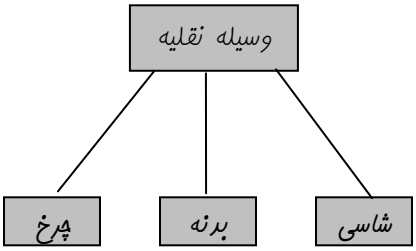
در بسیاری از موارد موجودیت ها در یک بانک صفات مشترکی دارند

مثال: در دانشگاه تمامی افراد اعم از استاد ، دانشجو و کارمند دارای صفاتی مثل نام ، تلفن و آدرس هستند . برای جلوگیری از تکرار بی رویه می توان ارتباطی از نوع ارث بری به صورت زیر تعریف کرد.



رابطه تجمعی(Aggrigation):

اگر چند تا موجودیت با هم یک موجودیت دیگر شکل دهند به رابطه این موجودیت ها رابطه تجمعی گویند.



پایان جلسه سوم

جبر رابطه ای

جبر:

- نوع داده ها و عملگر های رو ی آنها
- به مجمو عه ای از قوانین و عملگرها که امکان پردازش جداول را فراهم می سازند، جبر رابطه ای می گویند
- نوع داده ها در جبر رابطه ای فقط **رابطه** است،یعنی ورودی و خروجی تمامی عملگر ها رابطه می باشد.

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

1- عملگر های ساده
- **1**- گزینش(Select ، restrict **یا** σ) *
- **2**-پرتو(project **یا** Π) *

2-عملگر های مجمو عه ای
- **1**- اجتماع($\cup$) *
- **2**-اشتراک($\cap$)
- 3-تفاضل(**−**) *

3- عملگر های پیوند
- **1**-ضرب دکارتی(**X**) *
- **2**-پیوند طبیعی(∞)
- **3**-نیم پیوند($\propto$)
- **4**-پیوند شرطی(X_θ)
- **5**-فرا پیوند

4- عملگر های دیگر
- **1**- نامگذاری(ρ) *
- **2**- شیفت
- **3**- انتساب
- **4**-جایگزینی(**←**)
- **5**- تقسیم(**÷**)

عملگر های اصلی:

عملگر هائی هستند که جهت انجام یک سری از عملیات نیاز به آنها حتمی است(آنهائی که علامت ستاره خورده اند)

عملگر های غیر اصلی عملگرهائی هستند که آنها را به کمک عملگر های اصلی میتوان انجام داد و جهت سهولت کار هستند.

دامنه(Domain) : مجموعه مقادیری است که یک صفت خاصه می تواند اتخاذ کند. به عنوان مثال اگر یک صفت خاصه از نوع **Integer** دو بایتی باشد مقادیری که می تواند اتخاذ نماید 32768 الی 32768- می باشد (دامنه)

رابطه: زیر مجموعه ای است از ضرب دکارتی چند دامنه

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

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

مثال: اگر داشته باشیم D2:Integer و D1: String آنگاه هرمجموعه ای که عضو هایش زوج های مرتب (D1, D2) باشند یک رابطه است، بهترین راه نمایش و پیاده سازی رابطه به وسیله جدول است.

D1: String	D2: Integer
علی	10
رضا	20
...	...

تاپل: به هر کدام از سطر های جدول یک تاپل گویند

تاپل(Tuple):

به عضو (3,4) از رابطه R (بالا) یک تاپل گویند. پس تاپل به اعضاء رابطه گفته می شود. به عبارت دیگر تاپل مجموعه ای است از مقادیر صفات خاصه.

کاردینالیتی رابطه: تعداد تاپل های رابطه در یک لحظه از حیات آن، کاردینالیتی رابطه نام دارد و در طول حیات رابطه متغییر است .

تناظر بین مفاهیم رابطه ای و مفاهیم جدول:

رابطه = جدول	درجه = تعداد ستون ها
تاپل = سطر	کاردینالیتی = تعداد سطر ها
صفت = ستون	میدان = مجموعه مقادیر ستون

کلید:

صفت خاصه یا ترکیبی از صفات خاصه که در تمام تاپل های یک مجموعه منحصر به فرد باشد.

ابرکلید(Super Key:s.k)

یعنی هر ترکیبی از صفت ها که خاصیت کلید داشته باشد. این تنها نوع کلید است که الزاما خاصیت حداقلی (Minimality) نیست. یعنی زیر مجمو عه ای از آن هم ممکن است کلید باشد. مثلا «شماره دانشجوئی» و «نام دانشجو-شماره دانشجوئی» هردو ابر کلید هستند.

کمینگی اجزائی(Minimality):

یعنی اگر صفت خاصه ای یا ترکیبی از صفات خاصه کلید باشند، هیچ زیر مجموعه ای از آنها (به جز زیر مجمو عه ای که مساوی خودشان باشد) کلید نباشد. به عبارت دیگر با حذف هر یک از اجزاء A_i ، A_j و... A_k ، یکتائی مقدار از بین می رود.

کلید کاندید(Candidate Key:C.K):

کلیدی است که دارای خاصیت Minimality است. یک رابطه ممکن است چند کلید کاندید داشته باشد

کلید اصلی(Primary Key:P.k):

کلید کاندیدی است که توسط طراح بانک انتخاب و معرفی می شود. هر رابطه ای حتما کلید اصلی دارد، چون هر رابطه ای حداقل یک کلید کاندید دارد.

کلید فرعی یا بدیل(Alternative:A.k):

هر کلید کاندید غیر از کلید اصلی را کلید فرعی می نامند. به عبارتی یکی دیگر از کلید های کاندید است که برای برخی کاربرد ها انتخاب می شود. طراح می تواند در شمای ادراکی هم کلید اصلی را معرفی نماید و هم کلید فرعی را.

به عنوان مثال اگر در بانک اطلاعاتی دانشجویان، شماره دانشجوئی کلید اصلی باشد در صورتی که بخواهیم مشخصات دانشجویان را بر اساس نام نمایش دهیم، صفات نام و شماره شناسنامه می توانند جهت سهولت کار به عنوان کلید فرعی در نظر گرفته شوند.

کلید خارجی(Foreign Key:F.k):

کلید خارجی در رابطه ای مانند R_i ، صفت خاصه یا صفات خاصه ای است که در رابطه دیگر مانند R_j، کلید اصلی یا فرعی باشد

کلید خارجی برای نمایش ارتباطات بین انواع موجودیت ها بکار می رود و تنها کلیدی است که مقدار آن می تواند Null باشد.

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

در جدول روبرو (S#,P#,J#) کلید کاندید رابطه است. زیرا هیچ یک از صفات خاصه به تنهائی یا دو به دو یکتائی مقدار ندارند.

با آنکه نام دانشجو و شماره دانشجوئی با همدیگر به صورت یکتا تمام دانشجویان را از یکدیگر متمایز می سازند ولی نام دانشجو در این بین زائد است و شماره دانشجوئی برای این منظور کفایت می کند لذا (نام و شماره دانشجوئی) برای مجموعه دانشجویان کلید کاندید نیست. در این حالت به (نام و شماره دانشجوئی) ابر کلید یا Super Key گفته می شود.

مثال های این فصل بر مبنای جدول صفحه بعد ارائه می شوند.

prof:

pname	Office	esp	Degree	Clg#
میر شمسی	4	کامپیوتر	فوق لیسانس	10
ابو طالبی	3	مواد	دکتری	6
قربانی	12	کامپیوتر	دکتری	10
اشرفیزاده	8	شیمی	دکتری	5
هاشمی اصل	10	کامپیوتر	فوق لیسانس	10
جلالی	5	برق	دکتری	7
شیدفر	3	ریاضی	دکتری	1
حسنی	2	ریاضی	دکتری	1
جاهد مطلق	1	کامپیوتر	دکتری	10
ذاکر	4	فیزیک	دکتری	2
مفتون	1	زبان	دکتری	3
صادقیان	3	صنایع	دکتری	4

Stud:

S#	Sname	City	avg	Clg#
71133848	محمدی	تهران	17.24	10
72130502	وکیلی	اصفهان	14.06	10
72203305	علینقیزاده	مشهد	16.42	1
73120504	کمانی	یزد	17.56	4
73166801	احمدی	کرمان	15.44	5
74182532	جوادی	تهران	16.8	5
74209836	حسین زاده	تبریز	12.2	6

Crs:

C#	cname	unit	Clg#
10172	شبیه سازی	3	10
10174	مدار منطقی	3	10
12100	معارف 1	2	12
12564	ریاضی عمومی 1	4	1
51516	شیمی آلی	3	5
71203	کنترل خطی	3	7

Clg:

Clg#	clgname	city	pname
1	ریاضی	تهران	حسنی
10	کامپیوتر	تهران	جاهد مطلق
11	معماری	یزد	نقره کار
2	فیزیک	مشهد	ذاکر
3	زبان	مشهد	مفتون
4	صنایع	تهران	صادقیان
5	شیمی	تهران	اشرفی زاده
6	مواد	تبریز	ابوطالبی
7	برق	تهران	جلالی

sec:

sec#	c#	s#	term	pname	score
1724	10172	71133848	761	هاشمی اصل	14.5
1516	51516	74182532	752	اشرفی زاده	17
1747	10174	71133848	752	میرشمسی	15.75
1747	10174	72130502	752	میرشمسی	12.5
1748	10172	72203305	761	قربانی	16.25

عملگر گزینش:

گزینش سطر هائی از جدول را انتخاب می کند. نام جدول جلو علامت گزینش در پرانتز و شرط انتخاب زیر آن می آید. همه ستون های آن جدول در خروجی می آید.

مثال: *تمام ستون های جدول دانشجو که شهر آنها کلمه یزد را نشان می دهد و شماره دانشکده آنها 4 است.*

دستور:

$$\sigma_{City = \text{"یزد"} \wedge clg\# = 4}(Stud)$$

خروجی:

S#	Sname	City	avg	Clg#
73120504	کمانی	یزد	17.56	4

مثال: *با استفاده از عملگر جبر رابطه ای دستوری بنویسید که مشخصات دانشجویان تهرانی را نمایش دهد.*

$$\sigma_{City = \text{"تهران"}}(Stud)$$

S#	Sname	City	avg	Clg#
71133848	محمدی	تهران	17.24	10
74182532	جوادی	تهران	16.8	5

عملگر پرتو:

ستون هائی از جدول را انتخاب می کند و هیچ گونه شرطی اعمال نمی شود. درخروجی پرتو سطر های تکراری حذف می شوند. نام جدول جلو علامت پرتو در پرانتز و ستون های انتخاب شده زیر آن می آید.

مثال: *ستون های شماره دانشجو، نام دانشجو و کد دانشکده دانشجو از جدول دانشجو.*

$$\Pi_{s\#, sname, c\lg\#}(stud)$$

73166801	احمدی	5
71133848	محمدی	10
72130502	وکیلی	10
72203305	علینقیزاده	1
73120504	کمانی	4
S#	**sname**	**Clg#**
74182532	جوادی	5
74209836	حسین زاده	6

مثال: شهرهای مختلف محل تولد دانشجو

دستور: $\Pi_{city}\ (stud)$

خروجی:

City
تهران
اصفهان
مشهد
یزد
کرمان
تبریز

☜ از ترکیب گزینش و پرتو می توان اطلاعات بیشتری از جداول به دست آورد.

مثال: ستون های شماره دانشجوئی، نام، کد دانشکده و میانگین،دانشجویانی که معدل آنها بالای 15 است.

دستور: $\Pi_{s\#,sname,c\lg\#,avg}\ (\sigma_{avg>15}\ (stud))$

دستور معادل: $\sigma_{avg>15}(\Pi_{s\#,sname,c\lg\#,avg}(stud))$

S#	Sname	avg	Clg#
71133848	محمدی	17.24	10
72203305	علینقیزاده	16.42	1
73120504	کمانی	17.56	4
73166801	احمدی	15.44	5
74182532	جوادی	16.8	5

عملگر های مجموعه ای

عملگر های اجتماع ، اشتراک و تفاضل همان معنای خود در تئوری مجموعه ها حفظ کرده اند. ورودی هر کدام دو صفت و خروجی آنها یک رابطه است. روابط ورودی باید همتا(Same arity) باشند،یعنی

- تعداد صفت های دو رابطه(ستون های دو جدول) مساوی باشد.
- صفت ها به ترتیب دارای دامنه یکسان باشند.

☜ به عنوان مثال دو رابطه **stud** و **prof** ، **same arity** نیستند چون شرط دوم را ندارند.

مثال: دستوری بنویسید که لیست نام همه افرادی را که در دانشکده ها هستندرا نمایش دهد.

حل: این افراد یا استاد هستند یا دانشجو. بنابراین باید نام دانشجویان و نیز نام اساتید را جداگانه لیست و سپس با هم اجتماع کرد.(اسامی تکراری حذف خواهد شد)

دستور: $\Pi_{sname}(stud) \cup \Pi_{pname}(prof)$

☜ اگر کاردینالتی R_1 برابر n و کاردینالیتی R_2 برابر m باشد، آنگاه کاردینالیتی $R_1 \cup R_2$:

- حداقل زمانی است که یکی از رابطه ها زیر مجموعه دیگری باشد و برابر $\max(n,m)$ خواهد شد.

- حداکثر زمانی است که تاپل مشترک ندارند و برابر $n+m$ است

مثال: لیست نام اساتیدی که رئیس دانشکده نیستند.

حل: ابتدا نام اساتید را پیدا می کنیم و سپس نام روسای دانشکده ها را از آن تفریق می کنیم

دستور: $\Pi_{pname}(prof) - \Pi_{pname}(c\lg)$

خروجی:

pname
میرشمسی
قربانی
هاشمی اصل
شید فر

اگر کاردینالتی R_1 برابر n و کاردینالیتی R_2 برابر m باشد، آنگاه کاردینالیتی $R_1 - R_2$:

- حداقل زمانی است که $R_1 \subseteq R_2$ و برابر صفر است.
- حداکثر زمانی است که تاپل مشترک ندارند و برابر n است

مثال: لیست اسامی اساتید و دانشجویان همنام.

حل: کافی است اشتراک اسامی دانشجویان و اساتید را پیدا کنیم، یعنی دستور : $\Pi_{sname}(stude) \cap \Pi_{pname}(prof)$

اگر کاردینالتی R_1 برابر n و کاردینالیتی R_2 برابر m باشد، آنگاه کاردینالیتی $R_1 \bigcap R_2$:

- حداقل زمانی است تاپل مشترک ندارند و برابر صفر است.
- حداکثر زمانی است که $R_1 \subseteq R_2$ یا $R_2 \subseteq R_1$ و برابر $\min(n,m)$ است.

عملگر های پیوند

الف: ضرب دکارتی

از گرانترین عملگر های بانک رابطه ای است که زمان و فضای زیادی می خواهد و تا حد امکان باید از آن اجتناب کرد.حاصل ضرب دو رابطه، رابطه ای است که تاپل هایش از الصاق هر یک از دو تاپل دو رابطه بدست می آیند. به عبارت دیگر در $R_1 \times R_2$، هر سطر R_1 را پشت سر تمام سطر های R_2 قرار می دهیم.

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

نکته: ضرب دکارتی در ریاضیات مجموعه ها خاصیت جابه جائی ندارد یعنی($A \times B \neq B \times A$) ولی در جبر رابطه ای چون ترتیب ستون ها مهم نیست، خاصیت جا به جائی دارد، یعنی ($A \times B = B \times A$)

ب: پیوند شرطی(ضرب دکارتی شرطی)($\theta-$ JOIN)

این عملگر، زیر مجموعه ای است از ضرب دکارتی که شرط θ روی سطر های آن اعمال شده باشد. ستون های خروجی معادل ستون های ضرب دکارتی است. در بعضی کتاب ها آن را به صورت X_θ نمایش داده اند که θ شرط مورد نظر می باشد.

مثال. نام و شماره درسی که توسط استاد قربانی ارئه شده است.

حل: نام و شماره درس در جدول "درس" آمده است. این جدول با جدول "استاد" ارتباط ندارد(هیچ کدام کلید خارجی در دیگری ندارد). اما با جدول "گروه درس" ارتباط دارد، پس می توان از پیوند شرطی این دو جدول استفاده کرد به صورت زیر.

$$\Pi_{cname, crs.c\#}(crs\ X_{pname = \text{قربانی} \ \wedge crs.c\# = \sec.c\#}\ \text{sec})$$

خروجی:

C#	cname
10172	شبیه سازی

کاردینالیتی:

- حداقل زمانی است که هیچ شرطی نداریم $\min = 0$
- حداکثر زمانی است که تمام شرط ها باشند.

ج.پیوند طبیعی(natural join):

مشابه پیوند شرطی است با تفاوت های زیر

☜ خودبخود شرط تساوی روی همه ستون های همنام دو جدول اعمال می شود. یعنی فقط سطرهائی را از دو جدول انتخاب می کند، که همه ستون های همنام آن دو جدول مقادیر مساوی داشته باشند. در صورتی که دو جدول ستون همنام نداشته باشند، نتیجه پیوند طبیعی معادل ضرب دکارتی است.

☜ستون های تکراری فقط یکبار در خروجی ظاهر می شوند. از آنجا که فقط مقادیر مساوی آنها انتخاب می شود، نیازی به تکرار ستون یا نقطه گزاری (مثلا sec.c#) نیست.

مثال: نام و شماره دروسی که توسط استاد قربانی ارائه شده است.

حل. ابتدا شماره دروس استاد قربانی را از جدول "گروه درسی" انتخاب می کنیم، سپس شماره و نام همه درس ها را از جدول درس انتخاب می کنیم، آنگاه آنها را پیوندطبیعی می کنیم به شکل زیر.

$$\Pi_{c\#}\ \sigma_{pname = \text{قربانی}}\ ((\sec) \infty (\Pi_{c\#, cname}(crs))$$

مثال. مشخصات کامل رؤسای دانشکده ها

حل.

دستور

$$(\Pi_{pname}(c\lg)) \infty\ prof$$

قسمتی از خروجی.

pname	office	esp	degree	Clg#
حسنی	2	ریاضی	دکتری	1
جلالی	5	برق	دکتری	7
اشرفی زاده	8	شیمی	دکتری	5

کاردینالیتی:

- حداقل زمانی است که حداقل یک ستون همنام داشته باشیم ولی مقدار یکسان نداشته باشند(min=0)
- حداکثر زمانی است که هیچ ستون مشترکی نداشته باشیم(max=n*m)

د.عملگر نیم پیوند(Semi join):

این عملگر مشابه پیوند طبیعی است با این تفاوت که فقط ستون های جدول اول را می دهد.

مثال: خروجی دستور زیر چیست.

$$\sigma_{c\lg\#=1 \wedge term=771}(crs \propto \sec)$$

حل: ظاهرا این دستور مشخصات دروسی را می دهد(بدون مشخصات گروه آنها) که در ترم اول سال 77 (کد 771) در دانشکده 1 ارائه می شود، ولی واقعا چنین نیست. دستور فوق غلط است زیرا ستون ترم مربوط به جدول crs نیست و پس از نیم پیوند حذف می شود پس نمی توان آن را در شرط کنجاند. یکی از پاسخ های صحیح چنین است.

$$(\sigma_{c\lg\#=1} crs) \propto (\sigma_{term=771} \sec)$$

در رابطه با نیم پیوند باید به نکات زیر توجه کرد.

☜ ممکن است تعداد سطر های خروجی به مراتب کمتر از پیوند طبیعی باشد زیرا با کنار رفتن چند ستون، سطر های تکراری پدید می آیند و حذف می شوند.

☜ بر خلاف سایر عملگر های این بخش،ترتیب جداول ورودی در نیم پیوند مهم است($x \propto y \neq y \propto x$) زیرا همیشه ستون های جداول اول را می دهد

کاردینالیتی:

- حداقل زمانی است که ستون های همنام مقدار مشترک ندارند(min=0)
- حداکثر زمانی است که تمام مقادیر ستون های همنام مشترک باشند(max=n)

عملگر های دیگر:

عملگر تغییر نام($\rho_b a$)

نام جدید b را روی جدول a گذاشته می شود. محدوده آن در همان دستوری است که ذکر شده است. در واقع b اشاره گری به a است.

مثال: نام اساتیدی که دفتر کارشان مشترک است.

$$prof \ X_{prof.office=p.office \wedge prof.pname \neq p.pname} (\rho_p (\Pi_{pname,office}(prof))$$

ابتدا ستون های نام استاد و دفتر او از جدول استاد جدا و با نام p نام گذاری می شود. سپس سطر هائی از $prof$ که با p دفتر کارشان یکسان است، ولی نام متفاوتی دارند انتخاب می شوند. باید توجه داشت که همه اساتید با خودشان هم اتاق هستند و اگر شرط $prof.pname \neq p.pname$ پاسخ غلط خواهد بود.

عملگر انتساب(جایگزینی)

با علامت $\leftarrow$ جدول حاصل از دستورات ذخیره می شود تا در ادامه مورد استفاده قرار گیرد. اگر دستوری طولانی باشد می توان با استفاده از جایگزینی، آن را در چند مرحله نوشت. پاسخ پرس وجوها در این موارد از چند دستور تشکیل می شود.

مثال. مشخصات دروسی که توسط استاد قربانی ارائه شده است. (sec)

$$temp \leftarrow \Pi_{c\#} \sigma_{pname = \text{قربانی}}$$

$$temp \propto crs$$

عملگر تقسیم($R_1 \div R_2$)

☜ تمام ستون های R_2 در R_1 می بایست موجود باشند(شمای R_2 می بایست زیر مجموعه شمای R_1 باشد).

☜ ستون های خارج قسمت تقسیم، همان ستون هائی از R_1 است که در R_2 نباشند.

☜ نتیجه همان خارج قسمت، تاپل هائی از R_1 است که به ازای تمام تاپل های R_2 تکرار شده است

مثال. با توجه به R_1 و R_2 خروجی $(R_1 \div R_2)$ چیست.

R_1

S#	cname
10	منطقی
20	منطقی
10	شبیه سازی
30	ذخیره
30	منطقی
10	ذخیره
30	شبیه سازی
20	شبیه سازی
40	شبیه سازی

R_2

cname
منطقی
شبیه سازی
ذخیره

$R_1 \div R_2 \Rightarrow$

S#
10
30

مثال. با توجه به R_1 ، R_2 و R_3 خروجی $(R_1 \div R_2)$ و $(R_1 \div R_3)$ چیست.

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

R_1

P#
P1
P2
P3
P4
P5
P6

R_2

P#
P2
P4

R_3

$R_1 \div R_2 \Rightarrow$

S#
S1

$R_1 \div R_3 \Rightarrow$

S#
S1
S4

مثال. دستور جبر رابطه ای بنویسید که نام و شماره دانشجویانی را برهر که تمام درس های ارئه شده توسط میر شمسی را گرفته اند.

$T_1 \leftarrow \Pi_{\text{sec\#}, c\#}(\sigma_{pname = \text{میرشمسی}} (\text{sec}))$ همه درس های میر شمسی

$T_2 \leftarrow (stud \infty \text{sec})$ تمام دانشجویانی که تمامی درس ها را گرفته اند

$\Pi_{sname, c\#}(T_2 \div T_1)$ تمام دانشجویانی که تمامی درس های میر شمسی را گرفته اند.

☜ تقسیم زمانی استفاده می شود که همه(تمام) حالات یک موضوع را بررسی کنیم.

مثال. نام و شماره دروسی را که توسط همه دانشکده ها ارائه می شود.

$temp \leftarrow \Pi_{c\lg\#}(c\lg)$ تمامی دانشکده ها

$crs \div temp$ جواب

اعمال حذف،اضافه و تغییر رابطه ها

اضافه کردن به کمک عملگر های $\cup$ و $\leftarrow$ انجام می گیرد.

مثال. جهت اضافه کردن (82105200,"رضائی","شیراز",16.5,5) به جدول stud دستورات جبر رابطه ای به صورت زیر است

$stud \leftarrow stud \cup (82105200, "رضائی", "شیراز", 16.5, 5)$

مثال. جهت اضافه کردن درس جدیدی با شماره C101 با نام "بانک اطلاعات نامتمرکز" که چهار واحدی است و توسط دانشکده مهندسی کامپیوتر با کد 10 ارائه می شود، به جدول crs دستورات جبر رابطه ای به صورت زیر است.

$crs \leftarrow crs \cup (C100, "بانک اطلاعات نامتمرکز", 4, 10)$

حذف کردن با دستورات $-$ و $\leftarrow$ قابل انجام است.

مثال. برای حذف دانشجویی با مشخصات 73120504 از جدول stud دستورات جبر رابطه ای به صورت زیر است.

$$stud \leftarrow stud - \sigma_{s\# = 73120504}(stud)$$

مثال. دستور زیر تمام دانشجویانی که معدلی کمتر از 18 را از جدول good_stud حذف میکند.

$$good\_sud \leftarrow good\_stud - \sigma_{avg<18}(good\_stud)$$

تغییر داده های جدول.

☜ با این عمل نه سطری به جدول افزوده می شود و نه از آن حذف می شود، بلکه داده های سطر های موجود تغییر می کند.

☜ می توان با دستور های گزینش و جایگزینی این عمل را انجام داد.

مثال. افزودن یک واحد به همه دروس $\sigma_{unit \leftarrow unit+1}(crs)$

مثال. تغییر نام باختران به کرمانشاه در جدول دانشجو. $\sigma_{city \leftarrow "کرمانشاه"}(\sigma_{city = "باختران"}(stud))$

توسعه عملگرهای جبر رابطه ای

1- توسعه تصویر(project): به معنای قرار دادن دستورات محاسباتی در عملگر تصویر می باشد.

مثال. حساب بانکی(account)

Customer#	Customer name	blance	With draw	br-name
1	محمودی	$1000	$500	ملی
2	ایمانی	$5000	$3000	سپه
3	رضائی	$4500	$4000	مسکن
4	کریمی	$7000	$3000	ملی

نمایش مقدار باقی مانده حساب مشتری $\Pi_{customer\#, (blance - withdraw)}(ac)$

☜ عمل $blance - withdraw$ برای هر تاپل انجام می گیرد.

☜ برای دادن نام جدید به ستون حاصل از $blance - withdraw$ از دستور as استفاده می کنیم

$$\Pi_{customer\#, (blance - withdraw)\, as\, Rem}(ac)$$

customer	Rem
1	$500
2	$2000
3	$500
4	$3000

☜ در این حالت رابطه ایجاد شده دارای دو ستون با نام های $Rem, customer\#$ خواهد بود که Rem همان باقیمانده حساب مشتریان را نمایش می دهد.

2- توابع تجمعی (aggregate function):

☜ این توابع مجموعه ای از داده ها را گرفته و یک داده را به عنوان خروجی می دهند این توابع عبارتند از

$sum(), avg(), count(), Max(), Min()$

☜ تابع $avg()$ میانگین چند داده را برمیگرداند و تابع $count()$ عمل شمارش را انجام می دهد

☜ شکل کلی استفاده از توابع تجمعی در جبر رابطه ای به صورت زیر است .

$$G_1, G_2, \ldots G_n \zeta^{(R_1)}_{F_1(A_1)\ldots F_n(A_n)}$$

R_1 : رابطه

A_i : صفت خاصه

$F_j(A_i)$: یک تابع تجمعی بر روی صفت خاصه A_i می باشد.

G_i : صفت خاصه ای که تاپل هایش بر اساس آن گروه بندی می شوند.

مثال. دستور $\zeta^{(ac)}_{sum(blance)}$ مجموع موجودیهای تمام مشتریان را از رابطه (ac) می دهد که خروجی دستور یک رابطه می باشد.

مثال. با استفاده از عملگرهای جبر رابطه ای دستوری بنویسید که مجموع موجودیهای افراد را در شعب مختلف نمایش دهد.

جواب. $br-name\,\zeta^{(ac)}_{sum(blance)\,as\,sb}$

خروجی:

Br-name	sb
ملی	$8000
سپه	$5000
مسکن	#4500

اگر تعداد افرادی که در شعب مختلف سپرده گذاری کرده اند را بخواهیم، از تابع تجمعی $count()$; استفاده میکنیم که صفت خاصه ورودی آن $custmer\,name$ خواهد بود.

$$br-name\,\zeta^{ac}_{count(custmer\,name)\,as\,cc}$$

خروجی

Br-name	cc
ملی	2
سپه	1
مسکن	1

☜ در توابع تجمعی اگر بخواهیم مقادیر تکراری یکبار حساب شوند از دستور distinct استفاده مینمائیم، مثلا در صورتی که بخواهیم از رابطه ac نام ها بدون تکرار نمایش داده شوند به این صورت عمل می نمائیم $\zeta^{ac}_{count-distinct(cuustmer\,name)}$

پیوند بیرونی(**extend join**):

انواع پیوند بیرونی عبارتند از :

پیوند بیرونی چپ(⟕)

پیوند بیرونی راست(⟖)

پیوند کامل(⟗)

پیوند بیرونی چپ(⟕):

این پیوند شامل تمام تاپل هائی است که از پیوند طبیعی R_1 و R_2 تشکیل می شوند به اضافه تاپل هائی از R_1 که در پیوند طبیعی R_1 و R_2 ذکر نشده اند، به جای ستون هائی از R_2 که در R_1 وجود ندارند **null** قرار داده می شود.

مثال.

Br#	Br-name	add
1	ملی	آزادی
2	ملی	انقلاب
3	صادرات	آزادی
4	مسکن	انقلاب
5	تجارت	ولیعصر

R_1

Cu#	Cu-name	Br#
100	محمدی	1
101	رضائی	5
205	موسوی	3
300	کریمی	2
400	موسوی	2
401	ایمانی	6

R_2

Br#	Br-name	add	Cu#	Cu-name
1	ملی	آزادی	100	محمدی
2	ملی	انقلاب	300	کریمی
2	ملی	انقلاب	400	موسوی
3	صادرات	آزادی	205	موسوی
5	تجارت	ولیعصر	101	رضائی
4	مسکن	انقلاب	null	null

$R_1 ⟕ R_2$

پیوند بیرونی راست $R_1 ⟖ R_2$

*این پیوند در واقع شامل تاپل هائی است که در پیوند طبیعی R_1 و R_2 تشکیل می شوند بعلاوه تاپل هائی از R_2 که در پیوند طبیعی ذکر نشده اند، در این صورت با ذکر تاپل ها از R_2 به جای تاپل های غیر همنام در R_1 نیز **Null** قرار می گیرد.*

پیوند راست مثال قبلی به صورت زیر خواهد بود.

Br#	Br-name	add	Cu#	Cu-name
1	ملی	آزادی	100	محمدی
2	ملی	انقلاب	300	کریمی
2	ملی	انقلاب	400	موسوی
3	صادرات	آزادی	205	موسوی
5	تجارت	ولیعصر	101	رضائی
6	null	null	401	ایمانی

پیوند بیرونی کامل $R_1 ⟗ R_2$

این پیوند اجتماع دو پیوند بیرونی چپ و راست می باشد، در مثال قبل $R_1 ⟗ R_2$ به صورت زیر خواهد بود.

Br#	Br-name	add	Cu#	Cu-name
1	ملی	آزادی	100	محمدی
2	ملی	انقلاب	300	کریمی
2	ملی	انقلاب	400	موسوی
3	صادرات	آزادی	205	موسوی
5	تجارت	ولیعصر	101	رضائی
4	مسکن	انقلاب	null	null
6	null	null	401	ایمانی

محاسبه کاردینالیتی:

$$cardinality \qquad R_1 ⟕ R_2 \begin{cases} \min = n \\ \max = n*m \end{cases}$$

حالت min *زمانی است که دو رابطه حداقل یک ستون همنام داشته باشند.ولی در ستون های همنام دارای مقادیر مساوی نباشند، در این حالت تنها سطر های R_1 به رابطه اضافه می شودکه همان n است.*

حالت max *زمانی است که دو رابطه هیچ ستون همنامی نداشته باشند که معادل ضرب دکارتی خواهد بود.*

$cardinali\ R_1 \propto R_2$ $\begin{cases}\min = m \\ \max = n*m\end{cases}$ حالت min زمانی رخ می دهد که ستون های همنام دو رابطه مقادیر یکسان نداشته باشند، که در این حالت تنها سطرهای R_2 ذکر خواهند شد، که برابر m می باشد

$cardinali\ R_1 \supset\propto R_2$ $\begin{cases}\min = n+m \\ \max = n*m\end{cases}$ حالت min زمانی رخ می دهد که ستون های همنام دو رابطه مقادیر یکسان نداشته باشند، که در این حالت سطرهای R_1 و R_2 ذکر خواهند شد، که مجموع آنها $n+m$ می باشد

بهینه سازی پرس جوها(Query Optimization)

قواعد بهینه سازی

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

مثال. بهینه $\sigma_{unit=3}(crs \infty \sec) \equiv (\sigma_{unit=3}(crs)) \infty \sec$

در سمت راست(در حالت بهینه) ابتدا عمل گزینش انجام می گیرد، و در نهایت دروس سه واحدی با sec پیوند طبیعی داده می شوند ، در این حالت تعداد تاپل های کمتری در ترکیب $crs, \sec$ وجود خواهد داشت که سبب صرفه جوئی در حافظه خواهد شد.

2. شرط های ترکیبی را تبدیل به شرط های متوالی کنید. این روش باعث بهینه سازی از نظر زمانی خواهد شد.

مثال. $\sigma_{unit=3 \wedge c\lg\#=10}(crs) \equiv \sigma_{unit=3}(\sigma_{c\lg\#=10}(crs))$

3. پرتو را زودتر انجام دهید(ولی دیرتر از گزینش). این کار باعث صرفه جوئی در حافظه می شود.

$$\Pi_{pname,clg\#}(prof \infty clg) \equiv \Pi_{pname,c\lg\#}(prof) \infty \Pi_{pname,c\lg\#}(clg)$$

☜ در پیوند طبیعی باید مراقب باشیم تا در هنگام بهینه سازی، ستون های همنام را از دست ندهیم .

4. استفاده از هم ارزی در جهت بهینه سازی.

$$R_1 \infty R_2 \equiv R_2 \infty R_1 \qquad R_1 \cup R_2 \equiv R_2 \cup R_1$$

$$R_1 \infty (R_2 \infty R_2) \equiv (R_1 \infty R_2) \infty R_3 \qquad R_1 \cap R_2 \equiv R_2 \cap R_1$$

مثال. اگر سایز جدول crs را 100 و سایز جدول sec را 10001 فرض کنیم ثابت کنید $\sec \infty crs$ می تواند صد برابر سریعتر از $crs \infty \sec$ باشد.

حل. این دو دستور با یکدیگر بسیار متفاوتند! زیرا سایز دو جدول بسیار متفاوت است، جدول crs بسیار کوچکتر از جدول sec می باشد، فرض کنید جدول crs در حافظه کش جا بگیرد، در این صورت الگوریتم های دو راه حل بالا را بررسی میکنیم.

الگوریتم 1. $crs \infty \sec$

برای هر سطر جدول crs {

برای هر سطر جدول sec {

مقایسه کن

انتخاب کن

}}

الگوریتم 2 . $\sec \infty crs$

برای هر سطر جدول sec {

برای هر سطر جدول crs {

مقایسه کن

انتخاب کن

{

{

در الگوریتم 1 باید سطر های بسیار زیاد جدول sec را به دفعات وارد حافظه اصلی کنیم و مقایسه و انتخاب را انتخاب کنیم، به عبارت دیگر تعداد دستیابی به دیسک به اندازه حاصل ضرب سایز دو جدول است.

در الگوریتم 2 هر سطر sec فقط یک بار به حافظه اصلی می آید، زیرا جدول crs به طور کامل در حافظه کش است. تعداد دستیابی به دیسک به اندازه سایز جدول sec است

قواعد جامعیت در رابطه ها:

- جامعیت دامنه ای(Domain Integrity)
- جامعیت درون رابطه ای(inter Relation Integrity)
- جامعیت ارجاعی(Referentiol Integrity)

جامعیت دامنه ی: مقادیری که به صفات یک رابطه داه می شوند، از نوع دامنه آن صفات باشند،به عنوان مثال ، شماره دانشجوئی که به صورت عدد صحیح تعریف شده است، مقدار اعشاری را قبول نکند و همچنین مقادیر کلید ها تهی و تکراری نباشند.

جامعیت درون رابطه ای: به این معناست که هر رابطه به تنهائی درست تعریف شده باشد، به طوری که عضو تکراری نداشته باشد و کلید هایش به درستی معین شده باشند.

جامعیت ارجاع. در این جامعیت باید صفتی که به عنوان کلید خارجی تعریف می شود، در رابطه دیگر کلید اصلی یا فرعی باشد، و مقادیری که به کلید اصلی داده می شوند، در رابطه های دیگر موجود باشند(در رابطه هائی که با آن رابطه در ارتباط هستند)، مثلا می خواهیم درسی با شماره 1000 را به رابطه sec اضافه نمائیم، که این امکان وجود ندارد زیرا در جدول sec (گروه درسی)تنها درس های موجود در crs می توانند قرار بگیرند، و درسی با این شماره در جدول crs وجود ندارد.

- **حساب رابطه ای دامنه ای** (*Domain* Re*lationship Calculas*)

شکل کلی این حساب به این صورت می باشد $\{< c_1, c_2, \ldots, c_n > | \; p(c_1, c_2, \ldots, c_n, c_{n+1}, \ldots)\}$

و بدین معناست که ستون های c_1 تا c_n را بده اگر شرط p برقرار باشد.

☜برای بیان تعلق ستون ها به یک رابطه از عضویت استفاده می کنیم که با $\in$ نشان داده می شود، همچنین در هنگام استفاده از عضویت باید دقت کرد که شرط **same arity** بودن میان ستون های ذکر شده و رابطه داده شده ، رعایت شود

☜. مثلا اگر در بیان شرط داشته باشیم $< c_i \ldots c_j > \in R$ (ستون های c_i تا c_j عضو رابطه R هستند) و در تعریف نام ستون ها یا همان $< c_1 \ldots c_n >$، نام برخی از ستون های ذکر شده در عضویت موجود نباشند، توسط صور وجودی ($\exists$)و یا صور عمومی $(\forall)$ نام آن ستون ها ذکر خواهد شد.

☜ در قسمت شرط ($p()$) با استفاده از سور وجودی ($\exists$) و عمومی ($\forall$) می توان علاوه بر متغیر ها، ثابت هائی را هم ذکر کرد، به شرطی که از نوع دامنه های متناظر با رابطه ذکر شده باشند.

☜ ترکیب شرط ها با $\wedge, \vee, \neg$ و $\Rightarrow$ (معادل "و"، "یا"، "نه"، "نتیجه می دهد") انجام می شود.

☜ جهت ترکیب رابطه ها از ستون های همنام و متغیر های همنام استفاده می گردد.

☜ عدم تعلق با منفی کردن شرط(استفاده از $\in$ و $\neg$) انجام می شود و علامت عدم تعلق($\notin$) تعریف نشده است (زیرا با طبیعت بانک اطلاعاتی همخوانی ندارد)

☜ از هم ارزیهای مجموعه ای می توان در این ترکیب ها استفاده کرد.

$$(R_1 - R_2) \equiv (R_1 \wedge \neg R_2) \quad \leftarrow (R_1 - R_2) \equiv (R_1 \bigcap \overline{R}_2)$$

$$(R_1 \Rightarrow R_2) \equiv (\neg R_1 \vee R_2) \quad \leftarrow (R_1 \Rightarrow R_2) \equiv (\overline{R}_1 \bigcup R_2)$$

مثال. با استفاده از دستورات حساب رابطه ای دامنه ای مشخصات دانشجویانی را که دارای معدل بالاتر از 15 هستند را بدست آورید.

توضیح: در این دستور تنها به رابطه $stud$ نیاز داریم، چون تمام مشخصات دانشجویان با شرایط مورد نظر مد نظر است. در قسمت نام ستونها یا همان نام متغیر ها، نام تمام ستون های رابطه را می آوریم، یعنی خواهیم داشت $< s, sn, c, ave, c\lg >$، در قسمت شرط عنوان می نمائیم این ستون ها از رابطه $stud$ می باشند و شرط هم همان $ave > 15$ است.

$$\{< s, sn, c, ave, c\lg > | < s, sn, c, ave, c\lg > \in stud \wedge ave > 15\}$$

مثال. شماره دانشجوئی دانشجویانی را که معدل زیر 10 دارند را با استفاده از حساب رابطه ای دامنه ای نمایش دهید.

دستور $\{< s > | < s > \in stud \wedge ave < 10\}$ اشتباه می باشد زیرا در رابطه عضویت نام ستون هائی که ذکر می کنیم باید از نظر تعداد و دامنه با ستون های رابطه مورد نظر یکسان باشند یا به عبارتی دارای خاصیت $same\,arity$ باشند در حالیکه $< s >$ تنها یک ستون است و رابطه $stud$ دارای 5 ستون می باشد.در این حال برای رفع مشکل بایستی نام دیگر ستون های رابطه که در عضویت ذکر نشده اند، توسط سور وجودی بیان نمائیم تا خاصیت $same\,arity$ را رعایت کرده باشیم پس داریم $\{< s > | \exists s, c, ave, c\lg(< s, sn, c, ave, c\lg >) \in stud \wedge ave < 10\}$

مثال.نام و تخصص رؤسای دانشکده ها را با استفاده از حساب رابطه ای دامنه ای نمایش دهید.

دستور $\{< p, e > | \exists c, cn, ci(< c, cn, ci, p > \in c\lg) \wedge \exists o, d, cl(< p, o, e, d, cl > \in prof)\}$

این پرس و جو به دو جدول استاد و دانشکده نیاز دارد. متغیر مشترک p دو جدول را به هم پیوند می دهد. متغیر های کمکی هر دو جدول جداگانه تعریف شده اند. به محل باز و بسته شدن پرانتز ها دقت شود. متغیر ها را باید در نزدیکترین محلی که مورد استفاده قرار می گیرند تعریف کرد، و حوزه عملکرد آنها را با پرانتز مشخص نمود. حوزه عملکرد متغیر های خروجی سراسر دستور است.

پایان جلسات چهارم،پنجم،ششم

جلسه هفتم

SQL (Structured Query Language)(زبان پرس و جو ساخت یافته)

- این زبان مبتنی بر جبر رابطه ای و حساب رابطه ای دامنه ای می باشد، این زبان در سال 1970 توسط شرکت IBM ارائه گردید.
- در سال 1986 استاندارد هائی توسط کمیته استاندارد آمریکا(ANSI) و کمیته استاندارد اروپا(ISO) بر آن اعمال شد.

زبان SQL شامل موارد زیر می باشد.

DDL : زبان تعریف داده که SQL با استفاده از آن می تواند انواع دامنه و شماهای داده را تعریف کند.

DML : زبان دستکاری داده که SQL توسط این نوع دستورات، داده ها را عوض می کند، دستوراتی مانند *Delete , Update , Insert , Select* از این نوع هستند.

- این زبان شامل دستوراتی است که می تواند قوانین جامعیت را به پایگاه داده ارائه کند، همچنین شامل دستوراتی است که می توان شروع و خاتمه تراکنش را مشخص کرد.

دید(view): دستوراتی که توسط آنها می توان رابطه(جداول) مجازی ایجاد کرد.

- این زبان شامل دستوراتی جهت اعطای مجوز است که می توان به کاربران مختلف امتیازات خاصی را اعطا کرد

انواع داده در SQL (Data type):

Char(n) : رشته ثابتی به طول n می باشد.

Varchar(n) : رشته ای متغیر با طول حداکثر n می باشد.

int : تعریف عدد صحیح.

Small int : تعریف عدد صحیح کوچک(فضای آن به اندازه نیمی از فضای int می باشد).

numeric(p,d) : داده‌ی عددی با طول میدان p رقم و قسمت اعشاری d رقم(عدد حقیقی).

Double Real : نوع داده اعشاری با دقت مضاعف.

Float(n) : اعداد ممیز شناور با دقت n رقم اعشار.

Date : بیانگر تاریخ می باشد و دارای ترتیب "ماه-روز-سال" می باشد.

Time : بیانگر زمان است و دارای فرمت "ثانیه:دقیقه:ساعت" می باشد.

Timestamp : ترکیبی از تاریخ و زمان با دقت میکرو ثانیه با نمایش 20 رقم دهدهی بدون علامت(yyyymmddhhmmssnnnnnn)

تابع extract :

این تابع جهت بدست آوردن فیلدهای خاص از داده های ذکر شده می باشد؛ به عنوان مثال اگر d_1 نام یک داده از نوع Date باشد آنگاه دستور زیر تنها سال را از این نوع داده بر می‌گرداند. Extract(year from d_1)

- به جای year می‌توان از month ، day استفاده کرد.

تعریف داده جدید.

در حالت کلی به این شکل می باشد

```
creat type Data-name as Readsy type(s)
```

مثال.

```
create type color as ("Read","grean","blue")
```

در دستور صفحه قبل نوع داده **color** تعریف می‌شود و برای نوع این متغیر از داده های آماده **blue, Green, Red** استفاده می‌گردد با این تعریف مقدار داده **color** می تواند یکی از این سه مورد گفته شده باشد.

□ در f_1 **color** ، f_1 از نوع **color** تعریف می شود و می تواند یکی ز مقادیر **blue, Green, Red** را اختیار کند.

مثال. **creat type i as numeric(5,2)**

نوع داده ای بنام **i** تعریف می گردد که متغیر هاتی از نوع این داده می توانند اعدادی با 5 رقم صحیح و 2 رقم اعشار باشند.

حذف نوع داده تعریف شده.

صورت کلی به این شکل می باشد. **Drop type data name**

مثال. **Drop type color** داده ای با نام **color** حذف می شود.

□ محدوده نوع دستوراتی که تعریف می کنیم تنها در میان دستورات می باشد

□ دستور معادل **Creat type** دستور **Creat Domain Domain−name Domain−value** می باشد.

مثال. **Creat Domain i** (1,…,100) نوع داده ای بنام **i** تعریف کرده ایم که تنها مقادیر بین 1 تا 100 را می پذیرد.

□ دستور معادل **Drop type** دستور **Drop Domain domain−name** می باشد. مثلا با **Drop Domain i** نوع داده **i** حذف می شود.

تعریف رابطه(جدول)

صورت کلی به این شکل می باشد. **Creat table table−name**$(A_1d_1 \dots A_nd_n)$

A_1 تا A_n: نام صفات خاصه یا فیلد ها می باشد

d_1 تا d_n: به ترتیب دامنه صفات خاصه A_1 تا A_n می باشد.

مثال. دستور **Creat table** T_1**(ssn int, name char(30))** جدول مقابل را ایجاد می کند.

T_1

ssn	Name
⋮	⋮

□ شکل توسعه یافته دستور قبل به صورت

```
Creat table table−name (A1d1 … Andn)
[constraint 1]
⋮           ⋮
[constraint m]
```

می باشد، که موارد اضافه شده محدودیت ها یا قید های اعمال شده بر روی رابطه می باشند.

تعریف کلید اصلی در رابطه : برای تعریف کلید اصلی دو روش وجود دارد

1. تعریف کلید اصلی بعد از تعریف صفت خاصه با استفاده از واژه **primary key** (مثال **1** صفحه بعد)

از این روش زمانی استفاده می شود که کلید اصلی ساده باشد و ترکیبی نباشد(اتمیک باشد)

2. تعریف کلید اصلی در قسمت محدودیت ها : از این روش زمانی استفاده می شود که کلید اصلی ترکیبی باشد(ترکیبی از چند صفت باشد).

البته کلید اصلی ساده را نیز می توان از این طریق تعریف کرد.(مثال **2** صفحه بعد)

مثال1.

```
Creat table t1(ssn int primery key,name char(30))
```

مثال2.

```
Creat table t2(lname char[20],fname char[20],degree char[20])
primery key(lname,fname)
```

دستور بالا صفت ترکیبی $fname,lname$ را به عنوان کلید اصلی می گیرد، که این کار بعد از تعریف فیلد ها (صفات خاصه) و در قسمت محدودیت ها صورت می گیرد.

قید `Not Null` : اگر این قید(محدودیت) جلوی یک فیلد ظاهر شود، بدین معناست که مقدار این فیلد نمی تواند تهی باشد.

مثال. $creat\ \ table\ \ t_3(id$ int $not\ null\ \ ,\ \ name\ \ char(30))$

قید `chek ( p )` :این قید در قسمت محدودیت ها ذکر می شود و بدین معناست که شرط p باید در جدول ذکر شده بر آورده شود.

مثال.

```
Creat table mark(sn int primary key , grade numerice(4,2))
check((grade >= 00.00) and (grade < +20.00))
```

قید $Uniqe(A_i \cdots A_j)$: این قید همراه صفت خاصه و یا در محدودیت ها ذکر می شود،و به این معناست که مقادیر صفات خاصه A_i تا A_j باید یکتا باشند.از این دستور در تعریف کلید کاندید استفاده می گردد.

مثال. $Creat\ table\ t(f_1\ \ char[20]\ ,\ f_2\ \ char[30]\ ,\ f_3$ int $,uniqe(f_1,f_2))$

در دستور بالا مشخص می گردد که ترکیب مقادیر f_1 و f_2 نباید تکراری باشند.

پیاده سازی جامعیت ارجاعی: برای پیاده سازی این جامعیت در قسمت محدودیت ها از دستور زیر استفاده می کنیم

```
foreign key (A i ) References Reference – Table
[on Delete casecade]
[on update casecade]
```

☐ $Reference-table$ جدول مرجع است که در آن کلید خارجی به عنوان کلید اصلی می باشد

☐ اگر مقدار کلید خارجی در جدول مرجع حذف شود، و از $[on\ \ Delete\ \ casecade]$ استفاده نشود **DBMS** جلوی حذف را می گیرد ولی اگر از $[on\ \ Delete\ \ casecade]$ استفاده شود عمل حذف در جدول مراجع کننده به صورت آبشاری انتشار می یابد.

☐ $[on\ \ update\ \ casecade]$ به این معناست که تغییرات بر روی کلید خارجی در جدول مرجع به جدول رجوع کننده به صورت آبشاری سرایت می کند ولی اگر این عبارت ذکر نشود **DBMS** جلوی تغییرات را می گیرد، البته می توان به جای این دو عبارت از عبارات $Set\ \ Defult\ ,Set\ \ Null$ استفاده کرد، با استفاده از این دستورات با حذف و یا تغییر در کلید خارجی در جدول مرجع، به جای آن کلید در جداول مرتبط مقدار **NUll** یا مقدار پیش فرض قرار داده می شود.

مثال .

```
creat table stud(sn int , name varchar(30) , city varchar(40) , ave numeric(4,2) , clg int)
primery key(sn) , foreign key(clgn) , reference clg
on Delet casecade
on update casecade
```

در دستور بالا رابطه **stud** ایجاد شده و در قسمت صفات خاصه، صفات یا فیلد ها همراه با دامنه شان آمده اند، و در قسمت محدودیت ها، **sn** از نوع کلید اصلی بیان شده است، $clg\ n$ به عنوان کلید خارجی تعریف شده که رابطه مرجع برای $clg\ n$(رابطه ای که $clg\ n$ در آن کلید اصلی است) ، clg بیان شده است و در آخر با بیان عبارات $on\ update\ casecade\ ,on\ \ delete\ \ casecade$ امکان حذف و بروز رسانی داده ها در تمام جداول مرتبط با جدول مرجع فراهم شده است

حذف جدول: دستور $Drop\ \ table\ \ table-name$ جدول را به طور کامل حذف می کند و حتی شمای آن نیز باقی نخواهد ماند.

مثال. $Drop\ \ table\ \ stud$

تغییر شمای جدول:

☐ دستور $Alter\ \ table\ \ table-name\ \ add\ \ A_i d_i$، فیلد A_i با دامنه d_i را به رابطه ای که نام آن در $table-name$ ذکر شده اضافه می کند

☐ دستور $Alter\ \ table\ \ table-name\ \ delete\ \ A_i$، فیلد A_i را از رابطه $table-name$ حذف می کند

☐ دستور $Alter\ \ table\ \ table-name\ \ modify\ \ A_i d_i$، فیلد A_i را از رابطه $table-name$ به دامنه جدید d_i تغییر می دهد.

مثال.

```
creat  table  t(ssn int , name char(30))
Alter  table  t  add city char(50)
Alter  table  t  delete name
```

در خط اول دستور جدولی به نام t با دو فیلد $ssn, name$ تشکیل می گردد

t

ssn	name
⋮	⋮

در خط دوم نام شهر($city$) به فیلدهای جدول t اضافه می گردد.

ssn	name	city
⋮	⋮	⋮

درخط سوم فیلد $name$ از شمای جدول حذف می گردد.

ssn	city
⋮	⋮

دستورات واقعی محیط SQL:

دستور **select**: این دستور دارای سه قسمت است به این شکل

```
select  part1  from  part2
[where  part3]
part4
```

می باشد.

این دستور ترکیبی از عملگر های تصویر($part_1$)، ضرب دکارتی($part_2$) و گزینش($part_3$) می باشد.

☐ $part_1$ تمام فیلد های جداول ذکر شده در $part_4, par_3, part_2$ را بر آورده می کند.

مثال . $select\ \ *\ \ from\ \ stud$

دستور بالا تمام تاپل های جدول **stud** با ذکر نام ستون هایش می باشد به این شکل

sn	name	city	ave	clg
⋮	⋮	⋮	⋮	⋮

☐ $part_1$ میتواند نام فیلد خاصی باشد، مثلا با دستور $select\ \ sn, ave\ \ from\ \ stud$ تمام تاپل های ستون های sn و ave نمایش داده می شود.

□ دستور **select** به طور عادی مقادیر تکراری در یک ستون یا فیلد را نمایش می دهد، برای جلوگیری از وارد شدن مقادیر تکراری از دستور **Distinct** قبل از نام فیلد استفاده می نمائیم

مثال1 .دستور **select pname from sec** تمام مقادیر ستون **pname** را بدون توجه به تکراری بودن ارائه می دهد.

مثال2 .دستور **select Distinct pname from sec** نام های تکراری را تنها یک بار نمایش می دهد.

□ در part1 میتوانیم برای یک فیلد نام مستعار داشته باشیم به این شکل

نام مستعار **as** نام فیلد(صفت خاصه)

مثال . **select s#, city as shahr from stud**

S#	shahr
⋮	⋮

□ در part1 می توان توابع محاسباتی قرار داد

مثال . **select sn, ave*1.2 from stud**، مقادیر **ave** در 1.2 ضرب می شوند و در همان فیلد قرار می‌گیرند

جلسه هشتم

☐ در part1 می توان از توابع تجمعی نیز استفاده کرد، توابعی مانند $Min\ ()\ ,\ Max\ ()\ ,\ Sum\ ()\ ,\ avg\ ()\ ,\ count\ (A_i)\ ,\ count\ (*)$

☐ تمام توابع تجمعی به جزء count(*) مقادیر تکراری را حساب می کنند، البته مقادیر تکراری در $Min()\ ,\ Max()$ تاثیری ندارد. بنابراین اگر بخواهیم از مقادیر تکراری صرف نظر کنیم از کلمه Distinct استفاده می نمائیم،این کلمه با تمام توابع تجمعی به جزء count(*) قابل استفاده است.به شکل زیر نام تابع تجمعی $(distinct\ \ A_i)$

☐ در مقابل کلمه Distinct می توان از All َ استفاده کرد تا مقادیر تکراری نیز حساب شوند. البته پیش فرض تمام توابع تجمعی به جزء count()، All می باشد.

☐ نتیجه توابع تجمعی در part1 به part 4 نیز بستگی دارد.

مثال:

```
Select    Avg (avge )   from    stud
```

دستور بالا میانگین معدل ها را با محاسبه مقادیر تکراری نشان می دهد.

مثال.

```
Select   Avg(Distinct  avge)  from   stud
```

دستور بالا با استفاده از کلمه distinct ازمحاسبه مقادیر تکراری جلوگیری شده است.

مثال.

```
Select    Max (avge )   from    stud
```

این دستور در بین معدل ها بیش ترین معدل را حساب می کند.

مثال.

```
select    count(*)    from    stud
```

تابع count(*) مقدار تاپل های یک رابطه را می شمارد(سطر های Null را نیز می شمارد)

☐ توابع تجمعی فقط می توانند در لیست Select و Having ظاهر شوند.

Part3:

در این قسمت شرایطی را قرار می دهیم که می بایست بر آورده شوند، شرایط میتوانند با استفاده از عملگرهای رابطه ای مانند $=\ ,\ <>\ ,\ \geq\ ,\ \leq\ ,\ >\ ,\ <$,(برای مقایسه رشته ها) بیان شوند، برای ترکیب این شرایط از عملگر های منطقی مانند $Not\ ,\ OR\ ,\ And$ استفاده می شود.

مثال. Query (درخواستی) بنویسید که مشخصات دانشجویانی را برهر که معدل آنها بالاتر از پانزده است

```
select  *  from  stud
where  avge > 15
```

مثال. Query (درخواستی) بنویسید که مشخصات دانشجویان تهرانی را برهر که معدل آنها بالاتر از پانزده است

```
select  *  from  stud
where  (city ='تهران ') And (avge > 15)
```

مثال. Query (درخواستی) بنویسید که نام و شماره دانشجوئی دانشجویان تهرانی را برهر که معدل آنها بالاتر از پانزده است

```
select  sn , sname  from  stud
where  (city ='تهران ') And (avge > 15)
```

عملگر Link:

کار این عملگر تطبیق الگو می باشد که برای مقایسه رشته ها بکار میرود، کاراکتر های عمومی این عملگر percent (٪) و under line (_) می باشد،

☐ عملگر Link به بزرگی و کوچکی حروف حساس است.

☐ در عملگر **Link** علامت درصد(٪) به جای مجموعه ای از کاراکتر ها و علامت زیر خط(_) به جای یک کاراکتر می آید.

مثال. **Query**(درخواستی) بنویسید که نام دانشجویانی را که اسم شهرشان به "ان" ختم می شود را برهر.

```
select sname from stud
where city link "٪ان"
```

ابتدای رشته، هر رشته ای می تواند باشد، ولی بایستی در پایان "ان" داشته باشد.

مثال. **Query**(درخواستی) بنویسید که مشخصات دانشجویانی را که حرف اول نامشان را نمی دانیم ولی به "حمدی" ختم می شود را برهر.

```
select * from stud
where sname link "_حمدی"
```

از آنجا که مطمئن هستیم در اول رشته یک کاراکتر داریم پس از (_) استفاده می نمائیم، در صورتی که از وجود کاراکتر مطمئن نباشیم از (٪) استفاده می نمائیم.

☐ تنها در صورتی که مطمئن باشیم کاراکتر یا کاراکتر هائی داریم به تعداد کاراکتر ها از (_) استفاده خواهیم کرد.

: part4

عناصری که در این قسمت می توانند قرار بگیرند عبارتند از: **order by , Having , Group by** و عملگر های مجموعه ای شامل عضویت، مقایسه مجموعه ها، تست مجموعه ها؛ تست رابطه یکتا، تست مجموعه خالی و ...

مثال. **Query**(درخواستی) بنویسید که نام و شماره دانشجویان ممتاز هر دانشکده را نمایش دهد.

برای حل این **Query** می بایست جدول بر اساس شماره دانشکده گروه بندی شود و سپس از هر گروه بیش ترین معدل انتخاب شود. که جهت گروه بندی از **Group by** استفاده می شود.

☐ همواره فیلدی که در **Group by** ذکر می گردد باید در **Part1** نیز بیاید.

```
Select sn , sname , Max(avge) as avgemax , clgn , from stud
Group by (clgn)
```

sn	sname	avgemax	clgn
⋮	⋮	⋮	⋮

☐ در صورتی که از **Group by** استفاده نشود، از بین تمام معدل ها بیشترین معدل انتخاب می شود و تنها یک خروجی داریم.

مثال. **Query**(درخواستی) بنویسید که معدل دانشجویانی را که از معدل کل دانشجویان کمترند(میانگین معدل دانشجویان) را نمایش دهد.

این دستور خطای ساختاری دارد زیرا در قسمت مقایسه نمی توان توابع تجمعی داشت.

```
select avge from stud
where avge < Avg(avge)
```

☐ زمانیکه می خواهیم نتیجه توابع تجمعی را مقایسه کنیم می بایست از کلمه **Having** استفاده نمائیم دستور صحیح به صورت زیر خواهد بود.

```
select avge from stud
Having avge < Avg(avge)
```

: Order by

اگر بخواهیم خروجی **select** بر اساس یک یا چند صفت خاصه مرتب باشد از **Order by** استفاده می نمائیم.

مثال. **Query**(درخواستی) بنویسید که مشخصات دانشجویان را بر اساس نام دانشجویان به صورت مرتب نمایش دهد.

```
select * from stud
Order by (sname )
```

□ پیش فرض دستور Order by به صورت صعودی است اما می توان نوع مرتب کردن را بیان نمود.

$$Order\ by\ (sname)\begin{cases} asc & \text{صعودی} \\ desc & \text{نزولی} \end{cases}$$

مثال. Query (درخواستی) بنویسید که مشخصات دانشجویان مرتب شده را بر اساس نام دانشجو برهد، در مورد دانشجویان همنام، ابتدا دانشجوئی قرار بگیرد که شماره دانشجوئی بیشتری دارد.

```
select * fom  stud
order  by  (sname , sn desc)
```

در دستور بالا مشخصات دانشجویان ابتدا بر اساس نام به طور صعودی(پیش فرض) مرتب می شود، و در صورتی که دو نام یکسان وجود داشته باشد، بر اساس شماره دانشجوئی (sn) که به طور نزولی(desc) است مرتب خواهند شد.

ترتیب اجرای دستورات:

```
select   part1  from  part2
where   part3
Group  by
Having
Order  by
```

ابتدا شرط ذکر شده در $part_3$ بررسی می شود، و تاپل هائی که واجد این شرایط باشند انتخاب می شوند، سپس گروه بندی انجام میشود، و بخش Having بر روی آنها اعمال می گردد، و در آخر ستون های مورد نیاز در $part_1$ با توجه به Order by (در صورت وجود) نمایش داده میشود.

ترتیب اجرا $part_3$, Group by , Having , {Order by , $part_1$}

مثال. Query (درخواستی) بنویسید که مشخصات دانشجویان هر شهر را که معدل آنها کمتر از معدل کل دانشجویان آن شهر می باشد و در دانشکده شماره ده درس می خوانند را نمایش دهد.

چون اعمال به ترتیب انجام می گیرد، معدل ها با معدل میانگین هر شهر مقایسه می گردد.

```
select  *  from  stud
where  c lg n = 10
group by  city
having  (avge < Avg(avge))
```

تست تهی بودن یک صفت :

☐کلمه *Is Null* برای تست تهی بودن و *Is Not Null* برای تهی نبودن استفاده میگردد.

مثال: **Query** بنویسید که نام وشماره دانشجویی دانشجویان را بدهد که در مقابل شهر آنها مقداری وارد نشده است.

```
select sn , sname from stud
where city Is NUll
```

Nested Query (*Query* تودرتو)

اگردر part3 یا part4 ، دستور select داشته باشیم به چنین **Query** هایی **Query** تو در تو گوییم و به **Query** های ذکر شده دراین تست ها *sub Query* گوییم.

تست عضو مجموعه بودن :

برای این کار از کلمات *IN (sub Query)* و *Not IN (sub Query)* استفاده میگردد ، از آنجایی که *In* و *Not in* بروی یک Sub Query تست میشوند. پس در part3 و part4 از دستور *select* استفاده میگردد.

مثال : **Query** بنویسید که نام درسهایی را بدهد که نمره دانشجویان درآن درس بزرگتر از 15 باشد .

```
select cname from crs
where c# IN (select c# from sec
where score > 15)
```

در **subQuery** که در دستور بالا داریم از جدول sec شماره درسهایی بدست می آید که نمره کسب شده در آنها بزرگتر از 15 است، حال اگر شماره درسی از جدول crs عضو این مجموعه بدست آمده باشد یعنی آن درس ارائه شده ونمره کسب شده در ان بیشتر از 15 است.

☐دستور **Query** قبل را می توان با استفاده از پیوند طبیعی نیز نوشت، به این ترتیب که دروسی که ارائه شده اند از پیوند طبیعی *crs , sec* حاصل میگردد و می توان شرط *score > 15* را بروی آن اعمال نمود .

☐ اگردر part1 نام دو رابطه ذکر گردد، حاصلضرب دکارتی دو رابطه محاسبه میگردد ، برای پیوند طبیعی باید شرطی را مبنی بر مقایسه ستونهای همنام دو رابطه بنویسیم .

1.حاصلضرب دکارتی دو رابطه محاسبه میگردد.

2.با تست شرط مساوی بودن ستونهای همنام

3.پیوند طبیعی حاصل میگردد.

```
select cname from crs , sec
where (crs.c#= sec.c#)
And (score>15)
```

مثال. **Query** بنویسید که مشخصات دانشجویانی رابدهد که در شهرهای،تهران،اصفهان،یزد و تبریز زندگی نمیکنند.

در **Query** نوشته شده شهر ها از رابطه *stud* که عضو مجموعه ذکر شده نیستند، همراه با مشخصات دانشجویان آنها داده می شود.

```
select * from stud
where city Not IN("تبریز"،"یزد"،"اصفهان"،"تهران" )
```

مقایسه مجموعه ها:

مثال. **Query** بنویسید که مشخصات دانشجویانی را بدهد که میانگین معدل آنها بزرگتر از معدل تمام دانشجویان دانشکده شماره 5 باشد.

```
select * from stud
where avge > all (select avge from stud where c lg n = 5)
```

از **subQuery** رابطه ای بدست می آید که معدل تمام دانشجویان دانشکده شماره 5 در آن می باشد. حال می توان معدل هر یک از دانشجویان را از طریق *all >* با تمام این مقادیر مقایسه کرد. و آنهائی را که معدل شان از تمام معدل های موجود در رابطه بدست آمده از **subQuery** بزرگتر است بدست آورد.

> all	بزرگتر از همه
< all	کوچکتر از همه
<> all	مخالف همه
= all	مساوی همه

> som	بزرگتر از حداقل یکی
< som	کوچکتر از حداقل یکی
<> som	مخالف با حداقل یکی
= som	مساوی با حداقل یکی

☐ *all* <> معادل است با *NOT IN*

☐ *som* <> معادل است با *IN*

مثال. *Query* بنویسید که مشخصات دانشجویانی را برهر که معدل آنها بیشتر از معدل حداقل یکی از دانشجویان دانشکده شماره 5 باشد.

تست رابطه های خالی(*Not Exist*) و غیر خالی(*Exist*):

☐ از *Exist* و *Not Exist* برای تست غیر خالی و خالی بودن یک *Query* یا **subQuery** استفاده می کردد.

مثال. نام دانشکده هائی که درسی ارائه می دهند.

مثال. *Query* بنویسید که مشخصات درس هائی را برهر که تا به حال ارائه شده اند.

```
select * from stud
where c# NOT IN (select C# from sec)
```

عملگر تقسیم :

برای پیاده سازی تقسیم در *SQL* مستقیما عملگری وجود ندارد ولی می توان با تابع تجمعی *count()* پیاده سازی نمود.

مثال. *Query* بنویسید که شماره دانشجویانی را برهر که همه درس ها را گرفته اند.

```
Select sn from Sec
Group by (Sn)
Having (count (Distinct C#)) = (select Count(C#) from Crs)
```

در این *Query* رابطه *Sec* بر اساس شماره دانشجوئی گروه بندی می گردد و با استفاده از تابع *Count()* تعداد درس هائی را که در هر گروه و در واقع برای هر دانشجو موجود است شمرده می شود، حال اگر مقدار این درس ها با مقدار تمام درس ها برابر باشد، یعنی دانشجو تمام درس ها را گرفته است. از *Distinct* به این خاطر استفاده شده که ممکن است دانشجوئی درسی را افتاده باشد، و دوباره آن را بگیرد.

مثال. نام اساتیدی که همه درس ها را ارائه کرده اند.

```
Select pname from Sec
Group by (pname)
Having (Count (Distinct C#)) = (Select Count(C#) from Crs)
```

مثال.مشخصات اساتیدی که همه درس ها را ارائه کرده اند.

در *Query* مثال قبل نام اساتیدی که همه درس ها را ارائه کرده بودند را بدست آوردیم حال با مقایسه نام اساتید با نام اساتید موجود در *Query* مثال قبل بدین صورت که اگر نام استادی در آن *Query* باشد یعنی تمام درس ها را ارائه کرده است و می توان از رابطه *prof* مشخصات کامل اساتید را بدست آورد.

```
Select * from prof
where pname IN (Select pname from Sec
Group by (pname)
Having (Count (Distinct C#)) = (Select Count(C#) from Crs))
```

مثال. نام اساتید با مدرک دکتری که همه درس ها را ارائه کرده اند.

در مثال بالا کافی است در شرط *where* عبارت (" دکتری"*And , degree*) اضافه شود.

دستورات اضافه کردن داده به جدول:

صورت کلی.

```
insert into table − name
values (v1 ⋯ vn)
```

دستور بالا مقادیر v_1 تا v_n را به رابطه $table-name$ اضافه می نماید و در واقع یک تاپل به رابطه اضافه میکند.

مثال.

```
insert into stud
values (743622 ,'کریمی' ,'مشهد' ,13.22 , 10)
```

در دستور بالا مشخصات ذکر شده به ترتیب به هر یک از ستون های جدول Stud اضافه می شوند.

☐ در دستور insert مشخصات v_1 تا v_n به گونه ای در جدول قرار می گیرند که v_1 به اولین ستون و v_n به آخرین ستون اضافه می شود.

حال در صورتی که بخواهیم، مشخصاتی که وارد می کنیم به درستی وارد ستون های مربوطه خود شوند از فرمت زیر استفاده می کنیم

با استفاده از این فرمت هر یک از مقادیر v_i تا v_j به ترتیب وارد ستون های A_i تا A_j میشوند

```
insert into table − name (Ai ,⋯ , Aj)
values (vi ,⋯ , vj)        1 ≤ i, j ≤ n
```

مثال.

```
insert into stud (city , sn , avge , sname)
values ('تهران' , 7425306 , 14.25 , 'کریمی')
```

☐ می توان خروجی دستور select را در یک جدول ذخیره کرد یا به تاپل های یک جدول اضافه نمود.

مثال.

```
insert into good − student
select * from stud
where avge > 17
```

$good-stud$ رابطه ای است همانند رابطه Stud، ولی دانشجویانی که معدل بالای 17 هستند در آن قرار می دهیم.

دستور حذف کردن:

صورت کلی.

```
Delete from table − name
[where conditions]
```

مثال. Delete from stud

این دستور تمام مقادیر جدول Stud را حذف می نماید، ام شمای جدول همچنان باقی است

☐ دستور Drop table محتویات آن را همراه با شمای آن حذف خواهد کرد.

☐ در صورتی که بخواهیم تنها مقادیر خاصی از جدول حذف شوند از شرط استفاده می کنیم مثلا به این شکل

```
Delete from stud
where avge > 17
```

دستور بهنگام سازی(Update):

صورت کلی.

```
Update table − name set Ai = vi ,⋯, Ak = vk
[where conditions]
```

☐ در بهنگام سازی رکوردها موجودند و تنها مقادیر صفات خاصه آن را تغییر می دهیم.

مثال. Query بنویسید که نام درس شیمی آلی را به شیمی معدنی تغییر دهد.

```
Update Crs set cnamee = 'شیمی آلی'
Where Cname = 'شیمی معدنی'
```

مثال. دستوری بنویسید که معدل دانشجویان کمتر از 14 را در ضریب 1.2 و بزرگتر از 14 را در ضریب 1.1 ضرب کند.

اگر ابتدا معدل دانشجویانی که معدل کمتر از 14 دارند در 1.2 ضرب کنیم ممکن است معدل آنها مقداری بیشتر از 14 گردد، و سپس دوباره معدل آنها را چون مقداری بیش از 14 شده در 1.1 ضرب خواهد شد. بنابراین ابتدا باید معدل های بیشتر از 14 را در 1.1 ضرب نمائیم و سپس معدل های کمتر از 14 را در 1.2 ضرب شوند.

```
Update stud Set avge=
Case
Where avge>14 Then avge=avge*1.1
else avge=avge*1.2
end
```

□ می توان در عبارت Case برای مقایسه چند دستور از else when نیز استفاده نمود.

مثال. Query بنویسید که نام درس ریاضی عمومی 1 به ریاضیات پایه و تعداد واحد آن را برابر 3 قرار دهد.

```
Update Crs Set cname=' ریاضیات پایه ',unit=3
where cname='ریاضیات 1'
```

□ در صورتی که از شرط where استفاده نشود، نام تمام دروس به ریاضیات پایه و تعداد واحد آنها به 3 تغییر می کند.