پایگاه داده ها

**تعریف پایگاه داده**

دیتابیس یا پایگاه داده به مجموعه ای از داده های به هم مرتبط بدون کمترین افزونگی، (داده هایی که نه به صورت فیزیکی بلکه بصورت منطقی هستند) گفته می شود، که توسط یک سیستم متمرکز به نام DBMS مدیریت میشوند، بطوری که میتوان آنها را به اشتراک گذاشت و از داده های آن به صورت همزمان استفاده نمود.

**مهمترین ویژگی های یک دیتابیس:**

* مدیریت متمکز: مدیریتی که از یک طرف انجام میشود که از سراسر دنیا برقرار است.
* یکپارچگی: بطوری که یک تغییر در دیتابیس در تمام جهات صورت گیرد.
* کمترین افزونگی: بطوری که از تکرار بی رویه داده ها در دیتابیس جلوگیری شود.
* نداشتن تداخل
* مدیریت توسط یک سیستم به عنوان حصار نفوذ ناپذیر به نام DBMS
* قابلیت اشتراک گذاری به عنوان یک برنامه کاربردی برای کاربران

پایگاه داده یا بانک اطلاعاتی یک سیستم متمرکز برای ذخیره سازی داده های بسیار گسترده است، دلایلد مختلفی برای استفاده از آن وجود دارد:

یکی از مهم ترین دلایل آن می تواند جلوگیری از تکرار بیهوده داده ها باشد، همچنین می توان به تداخل یا Conflict اشاره کرد که باعث بهم زدن نظم یک پایگاه داده میشود.

قبل از پیاده سازی یک پایگاه داده بایستی یک شمای بصری از آن کشیده شود که به این شمای بصری ERD یا Entity Relationship Diagram گفته می شود.

**چند تعریف:**

**دیتا:** مواد خام بدون معنایی هستند که میتوانند به صورت اعداد یا حروف و کلمات باشند

**اطلاع:** دیتا های پردازش شده به عنوان یک اطلاعات شناخته میشن.

**دانش:** نوعی شناخت است که ما بر روی مجموعه از اطلاعات داریم.

مثلا وقتی عدد 880011 را بیان می کنم در ابتدا یک عدد بی معنیست، ما وقتی بیان می کنم که این شماره، شماره دانشجویی شماست در واقع به آن روح دادیم که یک داده با معنا بوجود آمد، و اگر بیان کنم که این شناسه دو رقم اولش سال ورودی شما را بیان می کند، در واقع شما شناختی نسبت به داده خود یعنی شماره دانشجوییتان، پیدا کرده اید.

**دو روش برای ایجاد دیتابیس وجود دارد:**

**روش فایلی و روش پایگاهی**

**روش فایلی** یک روش سنتی و ابتدایی برای نگهداری داده ها در قسمت های مختلف است، یک روشی اسان برای ذخیره کردن داده ها که به راحتی می توان از آنها بدون هیچ گونه برنامه ای مدیریت کننده استفاده نمود، اما معایب خاص خودش را دارد، با توجه به این سیستمی متمرکز ندارد تا داده ها در آن ذخیره شوند، باعث میشه نوع مدیریت آن به صورت جزیره ای و غیر متمرکز باشد، با تداخل همراه خواهیم بود، تکرار بی رویه داده ها زیاد خواهد شد، بروز رسانی داده ها سخت میشود، به راحتی نمیتوان مشخص کرد که هر دیتایی چه ارتباط دیگه ای دارد، در آن قادر به ایجاد ارتباط بین جداول نخواهیم بود و بسیاری از مشکلات دیگر.

**روش پایگاهی** یک روشی متمرکز است که توسط یک سیستم متمرکز به نام DBMS مدیریت میشود و مشخص می کند هر کاربری در آن قادر به انجام چه کار هایی است، یا اینکه چه کاربرانی می توانند وارد دیتابیس شوند، به راحتی میتوان بین جداول ارتباط برقرار کرد، داده ها با کمترین تداخل و افزونگی همراه خواهد بود و بسیاری از مزایای دیگر

اجزای اساسی یک پایگاه داده به ترتیب زیر است:

1. **کاربران Users**
2. **داده ها Data**
3. **سخت افزار Hardware**
4. **نرم افزار Software**

بعد از آشنایی با اجزای مهم یک دیتابیس بایستی انواع هر یک را بشناسید:

1. در دسته اول کاربران وجود دارند که سه نوع متداول از آنها در این نوشته شده است:
2. Database Administrator که همان مدیر پایگاه داده است، که به صورت پیش فرض تمام مجوز های عملیاتی Permission ها را دارای می باشد.
3. Programmer یا برنامه نویس مانند ادمین دیتابیس دارای اختیاراتی است.
4. End users کاربران نهایی که قرار است با دیتابیس در ارتباط باشند، به طوری که دارای اختیاراتی محدود شده توسط ادمین هستند.
5. در دسته دوم داده ها قرار دارند، که منظور آن تمام داده هایی است که در دیتابیس ذخیره شده و روزانه از آنها استفاده میشود که به دو دسته داده های سیستمی و کاربردی تقسیم میشوند. که مهم ترین آنها میتوان به System log ها و دیکشنری سیستم اشاره نمود. به طوری که از آنها به نام Meta data یاد می شود، یعنی: داده برای داده.
6. سخت افزار: بستری باید وجود داشته باشد که بتوان پایگاه داده را بروی آن قرار داد تا کاربران بتوانند به صورت متمرکز و بدون هیچ محدودیت مکانی و زمانی به آن دسترسی داشته باشند، به صورت کلی در این بخش سرویس دهنده ها حکم اصلی را دارند. در حالت کلی بستری که قابلیت پردازش، ذخیره سازی داده ها و ارتباط و رسانش از قابلیت های اصلی یک سخت افزار (بستر) پایگاه داده است.
7. نرم افزار دیتابیس یا Database Management System (DBMS) نرم افزاری که کاربران بتوانند به وسیله آن عملیات خود را انجام دهند، مانند MySQL یا MSSQL.

نکته اساسی اینجا به چشم می خورد که آیا همه کاربران میتوانند به یک نرم افزار دیتابیس (DBMS) دسترسی کامل داشته باشند؟

پاسخ این است خیر، دیتابیس یک حصار غیر قابل نفوذ دارد که فقط کسانی میتوانند به آن دسترسی داشته باشند که دارای مجوز های لازمه باشند. پس دو بحث به نام Authentication و Permission در اینجا وجود دارد.

**Authentication:** همه کاربرانی که توسط مدیردیتابیس تایید شده باشند میتوانند وارد پایگاه داده شوند، به شرطی که شرایط لازم برای احراز هویت و اعتبار سنجی را داشته باشند.

**Permission:** به معنای لغوی یعنی مجوز، بازم به سمت مدیر بانک اطلاعاتی بر می گردد، مدیر مشخص می کند هر کاربر بعد از Authenticate شدن میتواند از چه قابلیت هایی استفاده کند، یا به عبارتی دیگر قادر به انجام چه کارهایی هستند؟ مثلا بتواند جدولی بسازد، یا داده ها را فقط ببیند و یا مجوز نوشتن و تغییرات در اطلاعات را نداشته باشد.

پس از این تعاریف می توان گفت به صورت پیش فرض فقط دو کاربر Database Administrator و **Programmer** می توانند به صورت مستقیم به تمام دیتابیس و داده هایش دسترسی داشته باشند و تمام مجوز ها را دارا هستند، و این کاربران هستند که توسط مدیر یا برنامه نویس مشخص می شوند که چکاری می توانند انجام دهند یا به عبارتی دیگر محدود سازی شوند.

**معماری یک پایگاه داده Architecture of Database**

به طورکلی 4 سطح در معماری پایگاه داده وجود دارد:

|  |
| --- |
| **دید خارجی** |
| **دید مفهومی عام** |
| **دید مفهومی خاص** |
| **دید فیزیکی** |

**دید خارجی:** در واقع بعضی از کاربران ممکن است در بعضی از تصاویر اشتراکی داشته باشند. مثلا یک کاربری فقط می تواند دیتا های داخل دیتابیس را فقط بخواند و اجازه هیچ تغییری را ندارد، کاربردیگری دقیقا به همان دیتابیس دسترسی تغییر هم دارن، در واقع در مورد دی که هر دو کاربر مشترک هستند خواندن داده های داخل دیتابیس است.

**دید مفهوم عام،** در واقع همان شمای بصری است که به صورت ERD نوشته شده است، قبل از پیاده سازی یک دیتابیس بایستی به وسیله نمودار ها و دیاگرام ها یک دیتابیس طراحی شود.

**دید مفهوم خاص،** در واقع بعد از طراحی دیتابیس بر روی کاغذ نوبت به پیاده سازی آن می رسد که در این تقسیم بندی به نام دید مفهوم خاص است.

**دید فیزیکی،** همان بستری است که دیتابیس ما بر روی آن قرار می گیرد.

**مدل ها قدیمی:**

**مدل سلسله مراتبی:** که یک ارتباط یک به یک را تعیین میکرد و ساختار xml داشت

**مدل شبکه ای:** مدلی که یک ارتباط یک به چند و بلعکس را تشکیل می داد. که دو مفهوم رکورد ها و مجموعه اعضای مالک را مورد اهمیت قرار می داد.

**مدل تخت:** همچون آرایه دو بعدی که یک جدول با ستون رکورد هایی را بیان میکرد. (روش ناکارآمد)

که باعث افزایش تکرار بی رویه داده های میشد.

**مدل رابطه ای** براساس رابطه ها طراحی میشه

چند تعریف:

**داده گان یا متا دیتا، داده برای داده:**

یعنی برای داده هایی که در دیتابیس تعریف کردیم، داده های دیگری تعریف کنیم. که به دو دسته تقسیم میشود.

**لغتنامه داده ها:**

در واقع هر نامی که در دیتابیس در مورد ساخت یک تیبل و پایگاه نوشته میشود، در لغتنامه داده های دیتابیس هم قرار میگرد، که یک زمانی پیش نیاد که از نام های تکراری در دیتابیس خود استفاده کرده باشیم.

**کاتالوگ سیستم:**

هر اتفاقی که در سیستم دیتابیس رخ میدهد و وضعیت های مختلف در کاتالوگ سیستم دیتابیس ذخیره میشود در واقع همان لاگ فایل در سیستم عامل است.

**امنیت:**

حفاظت و نگهداری از داده ها در برابر آسیب های طبیعی مانند آتش سوزی به همین خاطر تهیه نسخه پشتیبان در بازه های زمان مشخص لازم است.

حفاظت و نگهداری از داده ها در برابر آسیب های غیر طبیعی مانند دسترسی غیر مجاز افراد به دیتابیس و استفاده از الگوریتم های مختلف رمزنگاری و هش کردن و غیره

(هش کردن غیر قابل بازگشته اما اسینک کریپت قابل بازگشت)

**دسترسی به دیتابیس توسط AAA**

**Authentication** احراز هویت، یعنی همون کاربری که ادعا میکنه کاربر این سیستم است.

**Authorization** اعتبارسنجی: مشخص می کنیم کاربر به چه قسمت هایی دسترسی داشته باشه.

**Accounting**: مشخص می کنیم کاربر به چه قسمت هایی چه مدتی دسترسی داشته باشد.

**جامعیت**

یعنی که داده ها و اطلاعاتی که دریافت می کنیم صحیح و سازگار باشد.

مثلا از حساب شما مبلغی کم میشود و همزمان شما بایستی مبلغ کم شده را در حساب خود مشاهده کنید در غیر این صورت یک تداخلی در جامعیت حساب شما رخ داده است.

**تراکنش یا تنزاکشن**

هر برنامه ای که در محیط دیتابیس توسط کاربر اجرا میشود به آن تراکنش گفته میشود. که مدیریت این بخش بر عهده DBMS است تا یکسری وظایف را در صورت نیاز انجام دهد تا جامعیت حفظ شود.

هر تراکنش یا تنزاکشن باید چهار ویژگی را داشته باشد:ACID

**یکپارچگی** **یا** **atomicity**

یعنی یا همه دستور عمل ها بایستی انجام بشه یا هیچ کدوم انجام نشه. مثلا اگر قراره پولی از حسابی به حساب دیگر منتقل شود باید این عملیات تا انتها پیش برود و نیمه کاره نمونه، یا کامل انجام بشه یا کامل فیلد بشه.

**سازگاری یا consistency**

هر ترنزاکشنی باید تمام قوانین دیتابیس رو رعایت کند، مثلا موقعی که پولی واریز می کنیم از سمت ما کم شده باشد و به مقصد اضافه شده باشد، اگه مشکلی نداشت میگن عملیات commit شده و اگر مشکلی داشت میگن abort شده.

**انزوا یا isolation**

ممکن است چند فرایند به طور همزمان بخواهند داده ای را در دیتابیس تغییر دهند به همین خاطر همروندی پیش می آید، پس ما باید این همروندی را حذف کنیم تا دیتابیس به انزوا بر نخورد.

**پایداری durability**

یعنی که اثارات تراکنش ها هیچ وقت از بین نمی رود، مثلا اگه پولی در حسابی ریخته شده هیچ وقت اثر آن از بین نمی رود حتی اگر بانک آتش گیرد.

موضوع بعدی در مورد ساختار دیتابیس است مانند ایجاد جدول، حذف جدول و کلیه ساختار دیتابیس که به آن **data schema** گفته میشود.

استقلال دیتا یا data dependency

که به دوصورت فیزیکی و منطقی

به صورت فیزیکی اگر در نحوه ذخیره سازی دیتا تغییری ایجاد کند برنامه کاربری نباید تغییر کند

به صورت منطقی اگر فیلدی اضافه یا حذف شوند نباید در برنامه کاربردی تاثیری بگذارد.

زبان دیتابیس به صورت بیانی است که شما منظورتون رو به روش ساده می رسانید و نتیجه را دریافت می کنید.

به دو دسته تقسیم میشود:

**DDL Data Definition Language** که مربوط به پياده سازي و تعريف يک ديتابيس حذف و تغييير در آن است.

**DML Data Manipulation Language** که مربوط به مدريت ديتا ها از قبل حذف و بازيابي و اضافه و خلاصه هر چيزي که مربوط به ديتا هاي ديتابيس ميشه.

**فصل دوم**

گام اول تو پياده سازي ديتابيس در واقع مدل سازي معنايي به صورت شکلي است، که بعدا از طريق همون مي شه به صورت حقيقي آنرا پياده سازي کرده.

براي مدل سازي بصري از نمودار ER استفاده ميشه، که در آن صفات و موجوديت ها و ارتباط اونا براي ما مهم هستن.

**موجوديت**: هر چيزي که تو ديتابيس بخوايم اطلاعات ازش داشته باشيم موجوديت گفته ميشه،

موجوديت ها در دانشگاه

دانشجو – درس – استاد – گروه درسي

**اولين ويژگي موجوديت**:

تعداد موجوديت ها در اکثر موقع معلوم نيست! (ثبتنام دانشجوها)

**دومين ويژگي موجوديت**:

معمولا چن تا صفت داره داره (شماره دانشجويي، نام و نام خانوادگي، شماره تلفن و غيره)

صفات یا Attributes که در مورد هر موجودیت صحبت می کند، صفت هاي يه موجوديت هميشه مطعلق به خودشه، مثلا معدل صفت يک موجوديت نيست.چون مقدارش متغير هستش، پس بايد صفت مقدارش ثابت باشه. اما صفت عامل ارتباط دهندست.

**دسته بندي هاي صفت ها:**

1. **ساده و يا مرکب:**

زير صفت نداره، مثلا نام دانشجويي زير صفت نداره قابل تجزيه نيست.

1. **صفت هاي کليدي:**

صفت يا ترکيبي از صفات که منحصر به فرد باشن، مثل کد ملي يا شماره داشنجويي

1. **صفت هاي تک مقداري يا چند مقداري**

صفتي که يا يه مقدار داره، يا چند مقداره. مثلا يک دانشجويي يک نام و نام خانوادگي دارد ولي ميتواند چند مدرک داشته باشد. (در ER يا ارتباط دو خطه يا بيزي با برد دوتايي)

1. **صفت هاي مشتق يا غير مشتق**

صفتي است که حضور خارجي نداره، يعني از ديگر صفات مشخص ميشن. مثال سال ورود داشنجو از دانشجو گرفته ميشه، یا مثلا میشه از روی کد ملی متوجه شد که موجودیت ما متولد چه شهریه، مثلا تاریخ تولد تجزیه شود می توان سن آنها را محاسبه نمود.

نکته صفت مشتق به صورت نقطه چین نمایش می دهند.

**ارتباط**

رابطه بين موجوديت ها رو ارتباط ميگن، که با لوزي نمايش داده ميشود.

تقسيم ارتباطات

از نظر درجه ارتباطي

به تعداد موجوديت هايي که در ارتباط شرکت ميکنن

**اتصال**

**يک به يک**

**يک به چند**

**چند به چند**

**حد ارتباط**

تعداد موجوديت هايي که ميتونن تو ارتباط شرکت کنن.

**شرکت اجباري يا اختياري**

يک دانشجو ممکنه استاد راهنماي نداشته باشه به همين خاطر با يک دايره تو خالي نمايش داره ميشه

**مومجوديت ضعيت**

وجودش وابسه به موجوديت قويه.، در صورتی که موجودیت قوی از جدول پاک شود، موجودیت ضعیف هم پاک خواهد شد.

**انواع کليد ها**

**ابر کلید یا سوپر کی**

هر صفت يا ترکيبي از صفات که **خاصيت کليدي** و **يونيک** داشته باشن، مانند **کد ملي**، يا **شماره** **دانشجوي** يا ترکيبي از شماره داشنجوي با نام دانشجوي

**کليد کانديد:**

جزو ابر کليد که کمينه (کمترين تعداد صفت ممکن تشکيل شده باشه) باشد.(هيچ چيزي نميشه حذف کرد)

**جدايي**

ترکيب نام دانشجويي و کد ملي، نمي تواند به صورت کانديد باشد، چون اگر کد ملي رو حذف کنيم، نام دانشجويي ديگه صفت کليدي نيست.

**کليد خارجي**

صفتي که تو يه رابطه کليد کانديده و تو جدولي ديگه براي ارتباط.

**قواعد جامعیت**

**درون رابطه ای**:

هر رابطه ای (جدولی) باید کلید کاندید مخصوص به خود را داشته باشد.

**ارجاعی**:

کلیدی است که در یک جدول به عنوان کلید اصلی (کلید کاندید) است، و در جدولی دیگر به عنوان یک صفت ساده جهت ارتباط بین دو رابطه (جدول) است.

**دامنه ای**:

یک فیلد مقادیرش را باید از دامنه ای بگیرد، یا به تعریفی دیگر، مقدار هر فیلدی تابعی از دامنه آن است، مثلا شماره دانشجویی نمیتواند حروف داشته باشد، یا کد ملی هم همینطور، نام و نام خانوادگی دانشجویی نمیتواند عددی باشد.

**اگر یکی از این قواعد وجود نداشته باشد، قواعد جامعیت نقض می شود.**

**تبدیل نمودار های ER به جدول:**

نکاتی برای این تبدیل وجود دارد:

**اگر ارتباط به صورت چند به چند درجه یک بود ، برای هر موجودیت یک جدول ایجاد میکنیم، و برای ارتباط هم یک جدول، که درجدول ارتباط باید صفت کلیدی دو موجودیت حضور داشته باشد.**

**مثلا ارتباط دانشجو با درس:**

درس

**پیش نیاز**

**درس** (**کد** **درس**، واحد درس، نام درس)

**پیش نیاز** (**ای دی**، کد درس، کد درس پیش نیاز)

|  |  |  |
| --- | --- | --- |
| جدول درس | | |
| کد درس | نام درس | تعداد واحد |
| 1108 | سیستم های عامل | 2 |
| 1110 | آزمایشگاه سیستم عامل | 1 |

|  |  |  |
| --- | --- | --- |
| جدول پیش نیاز | | |
| ای دی | کد درس | کد درس پیش نیاز |
| 1 | 1110 | 1108 |

ارتباط یک به چند درجه یک:

کارمند

**مدیر**

|  |  |  |
| --- | --- | --- |
| جدول کارمند | | |
| کد کارمندی | نام کارمند | مدرک کارمند |
| 1108 | علیرضا سلطانی | دیپلم نرم افزار |
| 1110 | سجاد ایوبی | دیپلم نرم افزار |

|  |  |  |
| --- | --- | --- |
| جدول مدیریت | | |
| ای دی | کد کارمندی که تحت مدیریت | کد فردی مدیر |
| 1 | 1108 | 1130 |
| 2 | 1110 | 1130 |

کارمند (کد کارمندی، نام کامل کارمند، مدرک)

مدیر (کد مدیر، کد افرادی که تحت مدیریت هستند)

**ارتباط یک به یک دانشجو با یک همگروه**

دانشجو

**همگروه**

باید برای هر کدام، چه موجودیت چه ارتباط، جدول درست کنیم.

دانشجو (کد دانشجویی ، نام و نام خانوادگی ،؛ مقطع)

همگروهی (**ایدی**، کد دانشجو، به همره کد همگروه)

|  |  |  |
| --- | --- | --- |
| جدول مشخصات دانشجویان | | |
| ***شماره دانشجویی*** | نام و نام خانوادگی | مقطع |
| 98123456789 | علیرضا سلطانی نشان | کاردانی |
| 98321654987 | یاسین داووده | کاردانی |
| جدول همگروه | | |
| ***ایدی*** | شماره دانشجویی | شماره همگروه |
| 1 | 98123456789 | 98321654987 |
| 2 | 987564123 | 98123987456 |

نکته در جدول همگروه، رکورد های تکراری حذف میشه، یعنی اگه 1 و 2 با هم گروه باشن، نیازی به بیان 2و 1 اصلا نیست.

**ارتباط یک به چند درجه دو:**

ارتباط بین، استاد راهنما و دانشجویان،

نکته، یک دانشجو می تواند یک استاد راهنما داشته باشد، اما یک استاد می تواند چند دانشجو داشته باشد.

برای همچین نمودار هایی بایستی برای هر موجودیتی یک جدول به همراه کلید کانید منحصر به فرد تعریف شود، و کلید اصلی موجودیت طرف یک، به عنوان کلید خارجی وارد موجودیت طرف N میشود.

**مثال:**

استاد راهنما

**راهنمایی می کند**

دانشجو

**حل:**

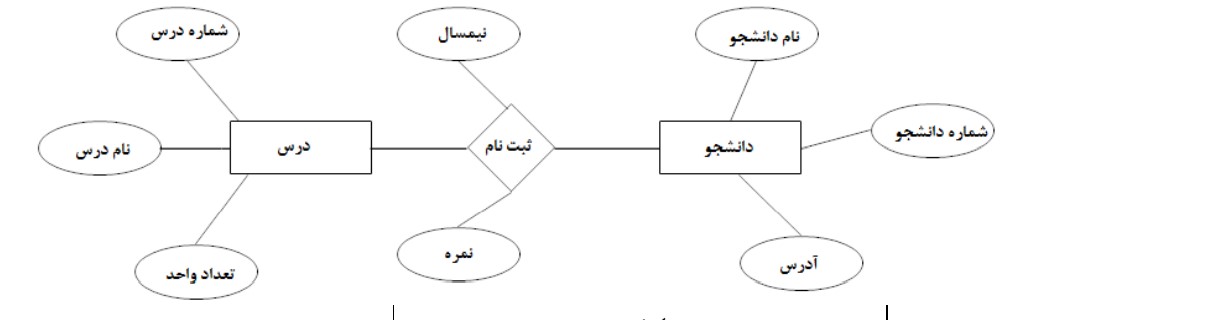
استاد راهنمای (***شماره استادی***، نام استاد، مدرک)

دانشجو (***شماره دانشجویی***، نام و نام خانوادگی، آدرس، مقطع، شماره استاد راهنما)

سوال: چرا برای رابطه جدول تعین نکردیم؟ چون ممکن بود هر دانشجویی چند استاد انتخاب کند با اینکه شرط این بود هر استادی فقط می تواند چند دانشجو داشته باشد، نه هر دانشجو چند استاد!

**ارتباط چند به چند درجه دوم:**

برای هر موجودیت یک جدول، برای هر رابطه هم یک جدول.

بطور کلی این نمودار به این شکل است:

درس (***کد درس***، نام درس، واحد درس)

دانشجو (***کد دانشجویی***، نام و نام خانوادگی دانشجو، مقطع، آدرس)

ثبت نام( ***ایدی***، شماره دانشجویی،کد درس، نیم سال، نمره)

**ارتباط دارای موجودیت ضعیف:**

برای هر موجودیت، یک جدول و کلید اصلی موجودیت قوی، به عنوان کلید خارجی وارد جدول موجودیت ضعیف می شود.

**موجودیتی که دارای صفت چند مقداری است:**

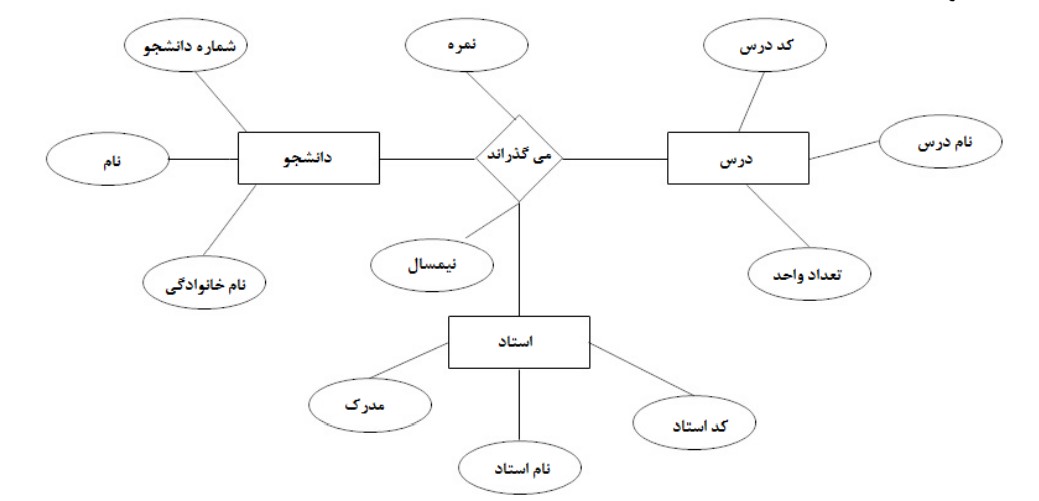
برای موجودیت اصلی یک جدول درست میکنیم، و برای فیلدی که حاوی چند مقدار است، یک جدول مجزا تشکیل می دهیم.

آژانس مسافر بری (کد آژانس، نام ، آدرس،)

جدول شماره تلفن (آیدی، کد آژانس، شماره تلفن)

**ارتباط با درجه بیشتر از 2**

برای هر موجودیت، یک جدول برای هر ارتباط هم یک جدول.

دانشجو (***شماره دانشجویی***، نام و نام خانوادگی)

درس (***کد درس***، نام درس، تعداد واحد)

استاد (***کد استاد***، مدرک، نام استاد)

رابط(آیدی، شماره دانشجویی، کد درس، کد استاد، معدل، نیم سال)

**جبر رابطه ای Relationship Algebra**

مبنای ریاضیاتی مد رابطه ای است.

**سطریاب یا عملگر گزینش:**

برای پیمایش بین سطر ها همانند دستور SELECT در SQL مادر جبر رابطه ای از علامت σ استفاده میکنیم. مانند:

σ city = “Tehran” ^ stud\_name=”Alireza” (stud)

**ستون یاب یا عملگر پرتو Project**

برای انتخاب ستون های یک جدول از عملگر پرتو استفاده میکنند، و در این قسمت فقط نام ستون و جدول در می شود، شرطی نیاز نیست نوشته شود.

Πنام ستون یا ستون ها (نام جدول)

Π city , stud \_name (stud)

**نکته: در عملگر پرتو، سطر های تکراری وجود ندارد.**

ترکیب رابطه:

**مثال**

**ستون های شماره دانشجویی، نام دانشجویانی که معدل آنها بالای 17 است، با استفاده از جبر رابطه ای بنویسید.**

Π student-num, sname (σavg>17 (stud))

**کد SQL آن**

**SELECT student\_num , s\_name FROM stud WHERE avg>17;**

عملگر های مجموعه ای

عملگر های اجتماع، اشتراک و تفاضل معنای خود را در تئوری مجموعه ها حفظ کرده اند.

**نکته:** ورودی هر کدام دو رابطه و خروجی هر کدام یک رابطه است.

**نکته:** رابطه های ورودی باید همتا باشند، شرط همتایی :

1. **تعداد صفت های دو رابطه یا همان ستون های دو جدول یکسان باشند.**
2. **صفت ها به ترتیب دارای دامنه های یکسان باشند.**

**سوال:**

**ليست نام اساتيدی که رئیس دانشگاه نیستند؟**

**Πpname(Prof) - Πpname(clg)**

**لیست اسامی دانشجویان اساتید؟**

**Πpname(Prof) ∩ Πpname(clg)**

**عملگر پیوند**

زمانی استفاده می شود که بخواهیم دو جدول رو با هم پیوند بزنیم.

**ویژگی ها**

1. بسیار پرکاربرد و قدرمتند اند.
2. سربار اجرایی بالا (زمان و حافظه)

**ضرب دکارتی ×**

وقت گیر و بسیار زمان بر است، استفاده به خودی خود ندارد.

**پیوند شرطی Theta join**

**× دستور معادل سوال: نام و شماره دروسی که توسط استاد قربانی ارائه میشود؟**

**Πcrs.cname,crs.c#,sec.pname (sec × crs) on pname=”قربانی” ^ src.c# = sec.c#**

**هزینه این روش بالاست، به دلیل سربار زیاد.**

**پیوند طبیعی natural join**

جزء علمگرهای اصلی جبر رابطه ای نیست، اما معروف ترین و پرکاربرد ترین آن است.

****وقتی دو تا جدول رو می خوایم پیوند کنیم، پیوند روی همه ستون های هم نامش صورت می گیرد، و ستون های تکراری فقط یکبار نمایش داده میشوند.

همان سوال قبلی با حل پیوند طبیعی:

**Π**sec.c#(σ pname=”قرباني”(crs)) ∞ **Π**crc.cname,crs.c#(crs)

**سوال: مشخصات کامل روسای دانشکده ها؟**



**یعنی صورت سوال از ما روئسای مشترک در دو جدول رو میخواد:**

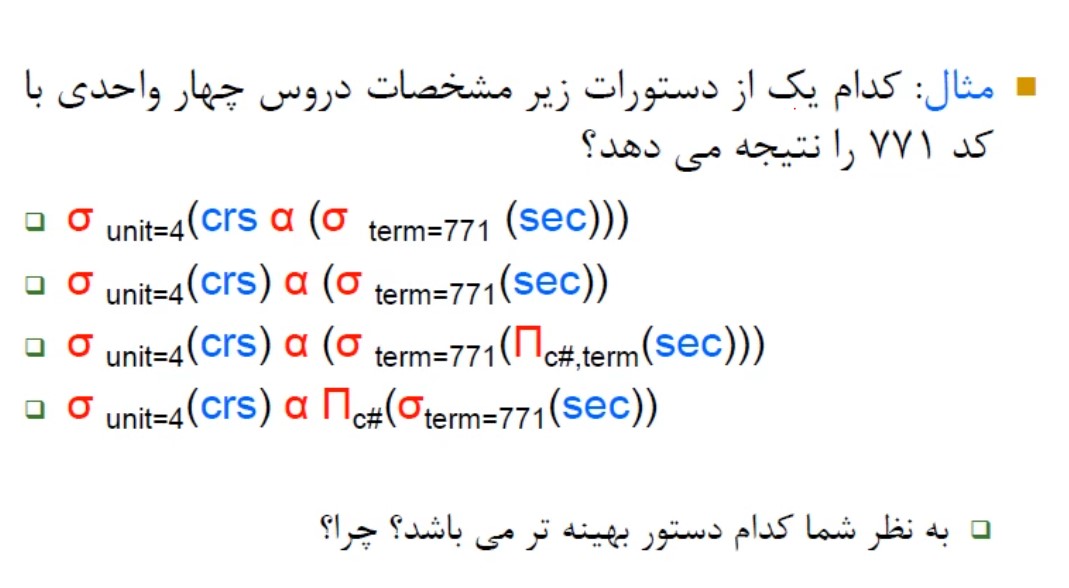
**Π**pname(clg) ∞ Prof



**بهینه سازی جبر رابطه ای**

بهینه سازی پرس و جو یعنی تغییر دستور به صورتی که از نظر زمان اجرا و فضای مورد نیاز بهینه شود.

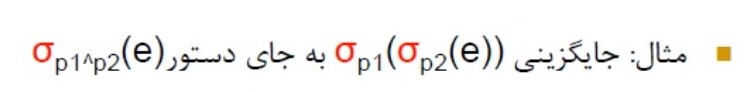
نکته مهمی که در جبر رابطه ای وجود دارد این است، که میتوانیم آنها را بهینه سازی کنیم.

1. **سوال: به نظر شما قواعد بهینه سازی در مدل رابطه ای باید توسط کاربر لحاظ شود یا DBMS خود بهینه سازی را انجام میدهد؟**
2. DBMS در هنگام اجرای پرسوجوهای ما (در زمان کامپایل دستورات) این بهینه سازی را انجام می دهد.
3. بهینه سازی در زمان اجرا با استفاده از معدل سازی و یکسری آمار دیگر توسط DBMS صورت میگیرد.

پاسخ: هر کدام از این دستورات یک کار را انجام میدهند، مهم این است که کدام بهینه تر است!

در زیر قواعد بهینه سازی دستور العمل ها را می آموزیم:

**قواعد بهینه سازی دستور العمل ها**

* اول از همه بايد از سيگما یا SELECT در رابطه استفاده شود.
* شرط های ترکیبی و گنگ را ساده کنید و به صورت متوالی نمایش دهید
* پروتو را زود تر انجام بدیم، یعنی بعد از سیگما
* عملگر های ترکیبی، مجموعه ای و پیوندی را در انتها استفاده کنیم

**پس ترتیب به این صورت شد**

**اول سیگما، بعد پرتو، اخر سر عملگر های ترکیبی.**

با توجه به این تعریفات گزینه شماره 4 بهینه تر است.

اگه جدول بسیار بزرگ در یک دیتابیس وجود داشته باشه، باعث میشه سرعت بانک کاهش پیدا کنه.

**به روز کردن در آوردن داده ها**

تغیر جداول در غالب جبر رابطه ای، عبارتست از:

1. اضافه کردن داده به جدول
2. حذف داده از جدول
3. تغییر داده از جدول

سوال: **آیا اضافه و حذف یک ستون و یا تغییر دامنه آن به جبر رابطه ای مربوط است؟ چرا؟**

خیر، زیرا به مدل ادراکی عام مربوط میشود.

اضافه کردن رکورد در جبر رابطه ای :

Clg 🡸 clg U {3, “Computer”, “Tehran”, “Akbari”}

حذف رکورد از جدول:

Clg 🡨 clg – sigma city=”Tehran” (clg)

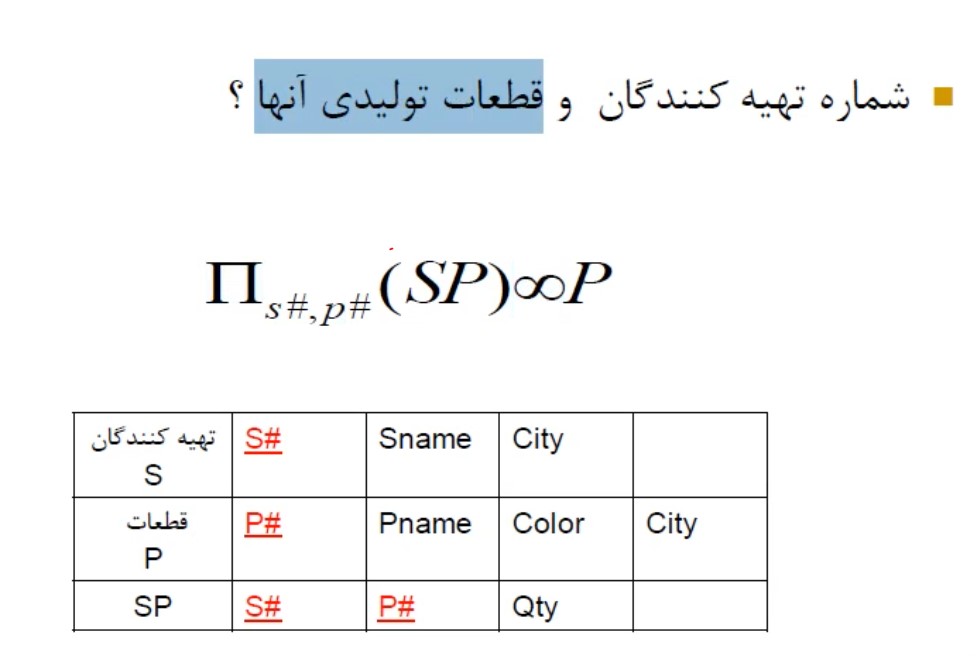
تغییر داده های جدول

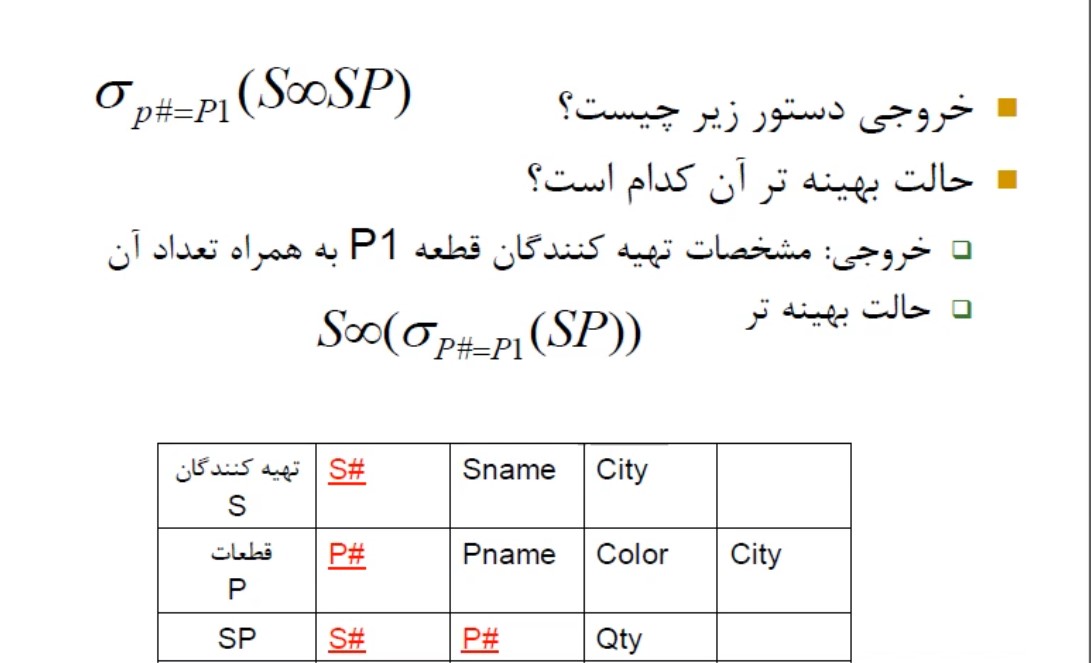
مثال: افزایش یک واحدی شماره اتاق استاید دانشکده ها:

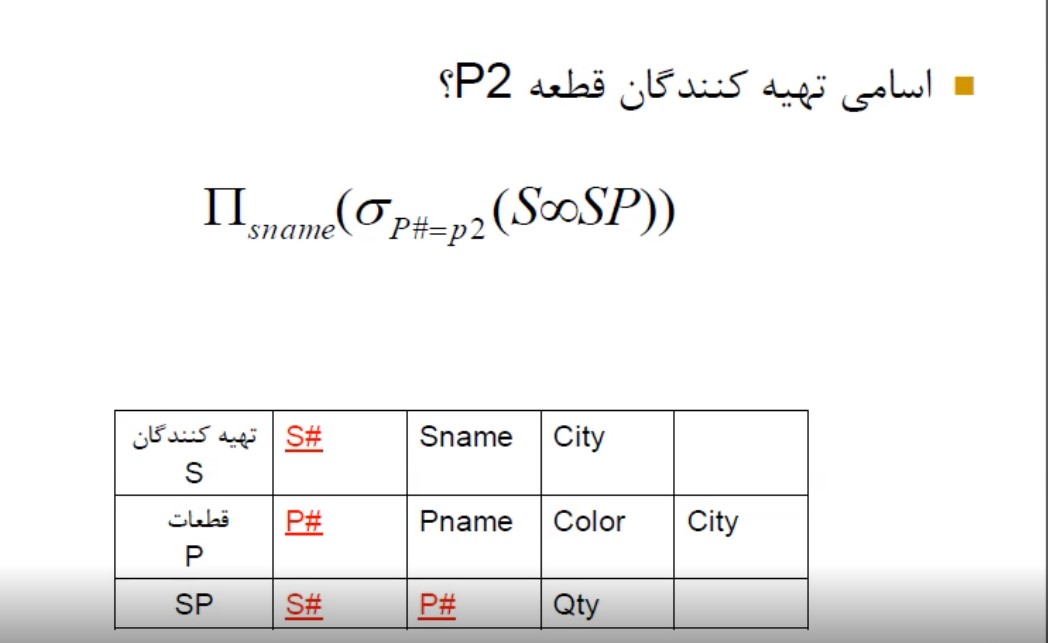
Sigma office🡨 office+1(prof)

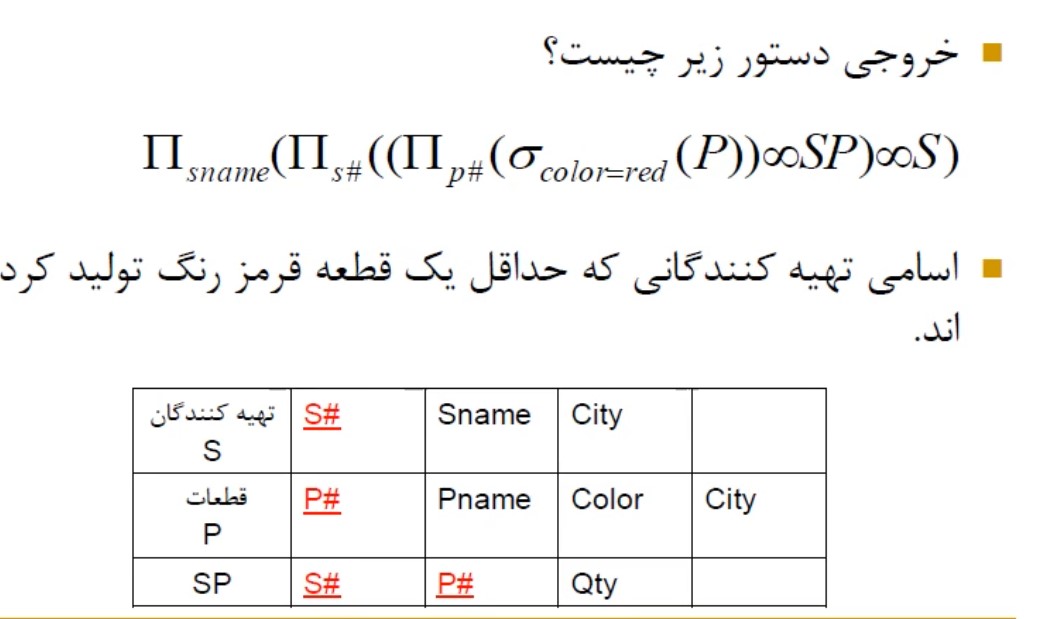
مثال: تغيير رئيس دانشگاه مشهد به اکبری

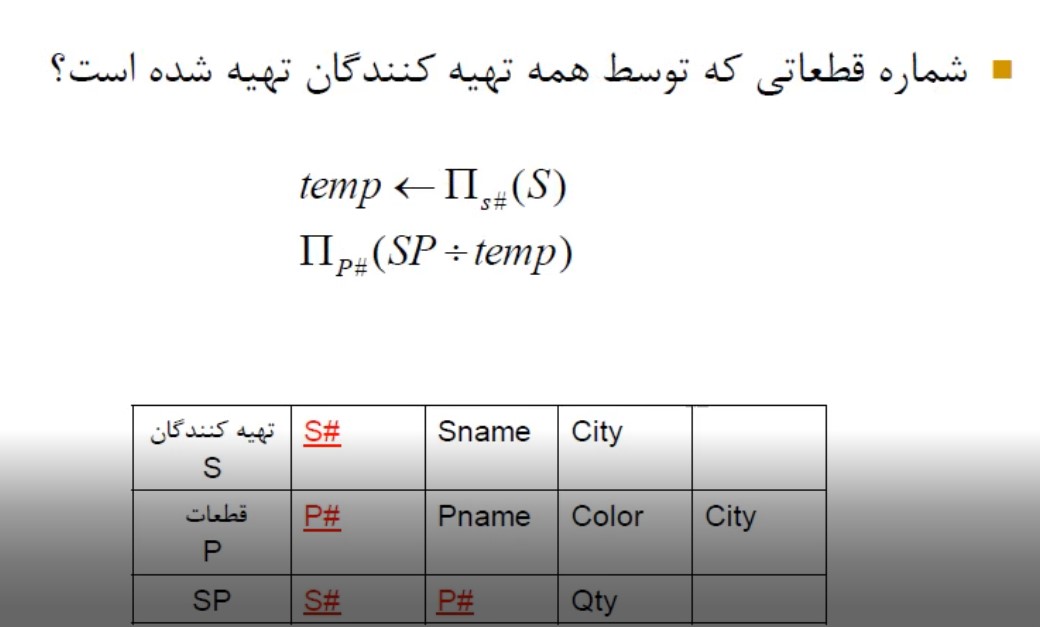
Sigma profName🡨”اکبري” (sigma city=”مشهد”(clg))

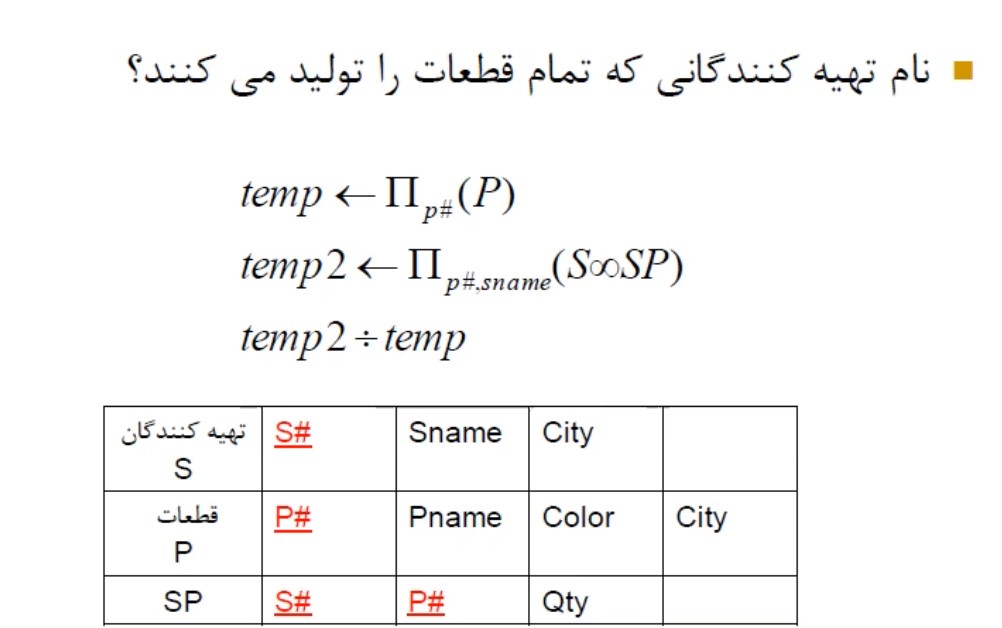
چند مثال:

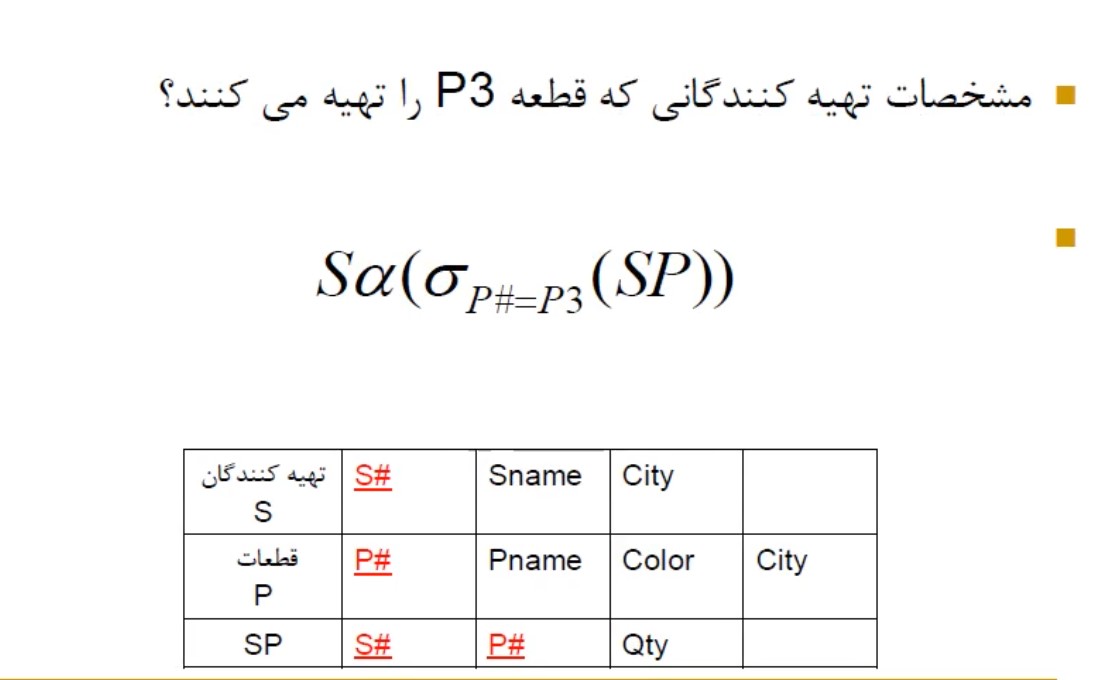
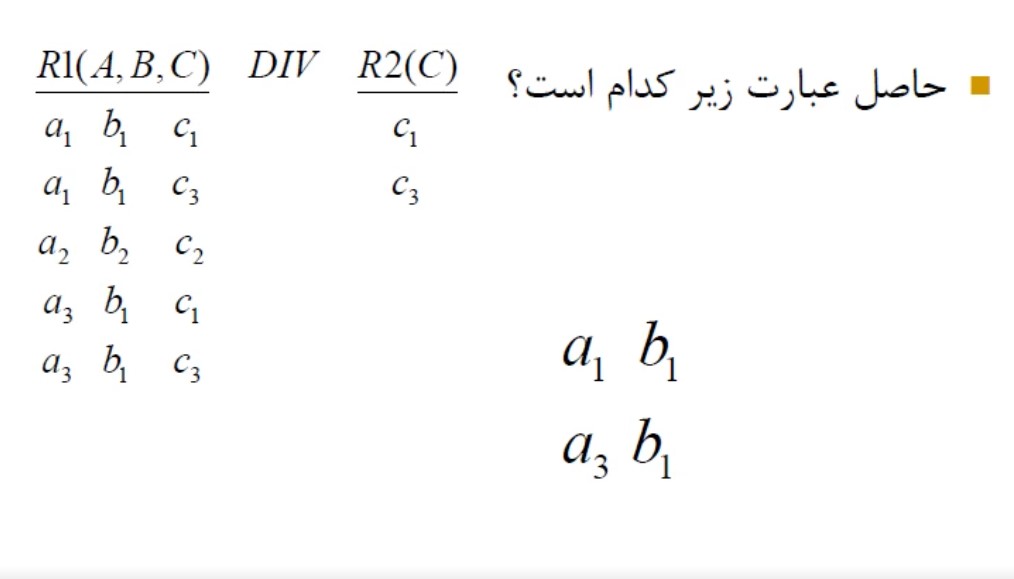












**SQL Server**

موقعی که می خواهیم دیتابیسی را در SQL Server ایجاد بکنیم، دو فایل بصورت پیشفرض با ایجاد دیتابیس ایجاد خواهند شد، اولی فایلی با پسوند MDF (Master Data File) که خود دیتابیس مورد نظر ماست و دیگری فایلی مربوط به Log LDF (Log Data File) است، که تمام وضعیت و اتفاقاتی که در سیستم دیتابیس رخ میدهد را در خود ذخیره میکند، به نوعی جعبه سیاه دیتابیس ماست.

برخی از نکات در هنگام ایجاد جدول با فیلد هایی:

در قسمت Identity دو بخش وجود دارد به نام identity seed که منظور از آن است که از چه عددی شروع شود، و identity Increment منظور آن است که چن تا چن تا به مقدار Auto increment اضافه شود.

**ایجاد جدول به وسیله کد**

Use <you database>

Go

Create table tbl (

Id int(10) identity(1, 1) primary key,

…..)

**برای اضافه کردن رکور از کد خط زیر استفاده میکنیم:**

Insert into <tbl>(all filed) values(all values fix with all filed)

نکته مهمی که در این میان وجود داره،این است که ما در هنگام اضافه کردن رکورد میتوانیم نام تمام فیلد های جدول مورد نظر را وارد نکنیم و فقط مقادیر خود را بیان کنیم **(\*البته به تعداد فيلدهايمان\*)**، اما اگه فیلدهای خود را بیان کردیم، باید به تعداد همان فیلدها مقادیر وارد شوند، مثلا اگه من سه فیلد را در نظر گرفتم، نیام از 4 مقدار یا 2 مقدار استفاده کنم، باید به تعداد فیلدهایی که بیان کردم، مقادیر را وارد کنم.

**برای پاک کردن یک جدول از دستور**

Drop table <your table>

**برای پاک کردن داده های یک جدول**

Truncate table <Your table>

Delete from <Your table>

**ممکنه بخواهیم رکورد ها را فیلتر کنیم و بگیم مثلا دو رکورد اول را انتخاب کن**

Select top(2) \* from p

select top (50) percent\* from p order by weight asc

نکته

اگر ما بخوایم از هر نوع چیزی، محاسبه ای رو انجام بدیم، مثلا بخوایم مجموع کالاهای تولید شده هر محصول، باید از Group By استفاده کنیم، زمانی که کالای خاصی را نمی خوایم!

کوئری مجموع هر کدام از محصولات تولید شده را بنویسید:

Select PN, sum(qty) from SP Group by PN

**توابع محاسباتی در WHERE نمی آیند.**

شرط گروه بندی Having است.

select Sn, sum(qty) from SP group by Sn having SUM(Qty) > 20

برای بدست آوردن تفاضل بین دو جدول از Except استفاده خواهیم کرد:

elect \* from P where P.Pid = (

select P.Pid from P

except

select R.Pid from R)

**با استفاده از All میتوانیم یک چیز را در برابر همه چیز بررسی کنیم.**

**با استفاده از any میتونیم یک چیز را با هر چیز استفاده کنیم.**

**معدل >= any در حقیقت in است.**

**طریقه نوشتن یک view**

Create view “name” as your query

**طریقه نوشتن یک procedure**

Create procedure ‘procedure name’ @var type, @var type …. as begin query end

**برای اجرا یک procedure نوشته شده**

Exec ‘procedure name’ with argument

Trigger ها همانند Procedureها و Viewها یک شئ هستند که در programmability ذخیره میشن دقیقا جزء زبان DML یا Data Manipulation Language هستند.

نکته های بسیار مهمی در این رابطه وجود دارد:

مثلا

برای معرفی متغیر درهنگام تریگر باید از declare @var type استفاده بشه.

**Inserted** یک جدول سیستمی در دیتابیس مربوطه هست که ما میتوانیم در تریگر به آن دسترسی داشته باشیم، یعنی وقتی شما اخیرا آخرین insertی که در جدول داشته باشید، در جدول inserted بطور موقت قرار گرفته است. این مسئله در updated و deleted نیز وجود دارد.

نکته حاز اهمیت دیگر این است که باید توجه داشته باشیم که با تریگر میتوانیم تغییرات مختلفی در تعریف و دسترکاری خود جداول داشته باشیم یعنی میتونیم DML و DDL انجام دهیم.

برای داشتن تغییرات بروی جداول و بطور کلی ایجاد و حذف و ویرایش از for در تریگر استفاده میکنیم.

برای داشتن تغییرات دستکاری روی دادا ها از After و before استفاده خواهیم کرد.

سوال ممکن است این گونه باشد:

تریگری بنویسید که دسترسی به ساختتن و حذف کردن جدول A را داشته باشند:

create trigger nochange on A

for create\_table, drop\_table, alter\_table

as

print 'access denied'

rollback

مثال دوم:

تریگری بنویسید که در یک جدول به نام alog بنویسید که چه تغییراتی DML در جدول A رخ میدهد.

create trigger t2 on Dbtest.A

after insert, delete, update

as

declare @op nvarchar(50)

if exists (select \* from inserted)

begin

if exists (select \* from deleted)

set @op='updated'

else

set @op='inserted'

end

else

set @op='deleted'

insert into alog values(suser\_sname(), getDate(), 'A', @op)

همین مثال برروی دیتابیس university در لوکال

use university

go

create trigger log\_of\_changes on S

after insert, update, delete

as

if exists(select \* from inserted)

begin

declare @op nvarchar(50)

if exists(select \* from deleted)

set @op='Updated successful'

else

set @op='Inseted succssfull'

end

else

set @op='Deleted successful'

insert into studentLog values(SUSER\_SNAME(), GETDATE(), 'Students', @op)

چند نکته در هنگام ایجاد یک جدول:

درهنگام ایجاد یک جدول میتوانیم یکسری محدودیت ها را اعمال کنیم

**Create table test (**

**Id int identity(1, 1) primary key,**

**Username nvarchar(50) unique,**

**Score float,**

**City nvarchar(50) default N’تهران’,**

**Check(score>=0 and score<=20)**

**);**

**چند نکته که در پیشتر ذکر نشده است:**

**هنگام ایجاد دیتابیس در GUI یک پنجره بنام General وجود دارد** که در آن مشخص شده که چه فایل هایی در هنگام ایجاد یک DB ایجاد خواهند شد مانند MDF و LDF.

در قسمت بعدی در مورد نوع فایل ها میگوید، MDF نوعش رکوردی و LDF از نوع LOG.

در قسمت بعدی در مورد گروه یا file group است که به صورت پیش فرض فایل master در primary قرار داره و فایل لاگ در not applicable.

قسمت بعدی initial size است که میتوانیم اندازه های پرونده ها مشخص کنیم.

قسمت بعدی که مهمه path یا مسیر ذخیره سازی دیتابیس شماست.

**صفحه بعدی در مورد گزینه ها و options است:**

**مهمترین نکته درون آن، این است** که نوع دیتایی که قصد داریم در db وارد کنیم را مشخص می کنیم، یعنی قسمت collation اگه فارسی یا چینی یا عربی و غیره باشه در آن بایستی ذکر شود اما چند تا سرکلمه در آن وجود دارد مانند:

**CI:** **Case** **Insensitive یا CS: Case sensitive یا KS یا WS: Width Sensitive**

**و قسمت مهم بعدی** در مورد Compatibility level هستش، همان طور که از اسمش مشخصه، مشخص میکنه که این دیتابیس از چه MS های دیگر بایستی پشتیبانی کند!

**\*\*\* ما میتوانیم در قسمت security یعنی پایین database یک هویت جدید بسازیم\*\*\***

در پنجره Login – New در قسمت General نوع ورود را مشخص میکنیم که ویندوزی باشد یا سروری؟

در قسمت server roles قوانین را مشخص میکنیم.

در قسمت User Mapping نشان میدهیم که به چه dbهایی دسترسی داشته باشد.

نکته در هنگام کاپي کردن فايل ديتاي db

حتما باید دیتابیس مربوط Detach بشود.

روش های گرفتن بک آپ از db

یا میتونیم از گزینه task -> backup و restore استفاده کنیم

یا متونیم فایل masterا، در فولدر مربوط را کاپی کنیم که اول باید Detach باشد، و بعد task -> attach میکنیم.

برای این job میسازیم که طی یک فرایند یکسری کار ها صورت بگیره.

فعال یا غیر فعال کردن هر job از طریق job activity monitor صورت میگیرد.

**آزمایشگاه پایگاه داده**

دیتابیس سلسله مراتبی، به صورت سطحی میباشد، وقتی تعداد موجودیت ها زیاد شود، سطح این دیتابیس هم زیاد میشود. (سیستم درختی ارتباط بچه و مادر)

دیتابیس شبکه ای ارتباط بین موجودیت ها رو به صورت لینک بیان میکنه، در این نوع از دیتابیس ما دارای multiple parent بر خلاف دیتابیس سلسله مراتبی.

تمامی موارد بالا برای زمانی بود که امکان دیتابیس های رابطه ای وجود نداشت. اما بعد از تعریف دیتابیس رابطه ای در حقیقت داده ها در ساختار جدولی معرفی و نمایش داده میشود و تحت عنوان مدل رابطه ای مدیریت میشوند.

زبان یکپارچه UML