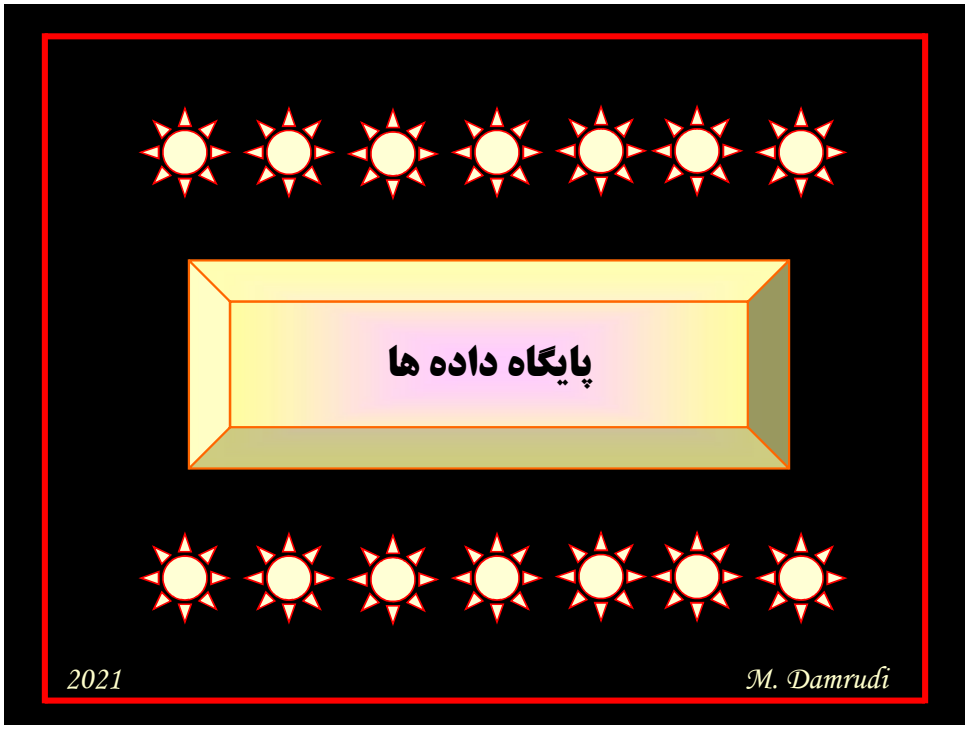


مقدمه

- فایل های اطلاعاتی که به نوعی به هم مرتبط هستند تشکیل بانک اطلاعاتی را می دهند.
- فایل مجموعه ای از رکوردها است.
- رکورد مجموعه ای از فیلدها است.
- فیلد کوچکترین جزء یک بانک اطلاعاتی است.
- نمایش پدیده ها، مفاهیم به صورت صوری و مناسب برای برقراری ارتباط یا پردازش، داده نامیده می شود. داده همان مقدار ذخیره شده است.
- داده پردازش شده را اطلاعات می نامند.
- مجموعه‌ای از داده‌های ذخیره شده و پایا، به صورت مجتمع(یکپارچه) (نه لزوما فیزیکی، بلکه حداقل به طور منطقی)، بهم مرتبط، با کمترین افزونگی، تحت مدیریت یک سیستم کنترل متمرکز، مورد استفاده یک یا چند کاربر از یک یا بیش از یک "سیستم کاربردی"، به طور همزمان و اشتراکی را پایگاه داده ها می نامند.

مقدمه

پردازش داده‌ها از دهه ۱۹۵۰ تا کنون فراز و نشیب‌های فراوانی داشته است.

نسلهای ذخیره و بازیابی اطلاعات:

- نسل اول نسل فایلهای ساده ترتیبی
- نسل دوم نسل شیوه های دستیابی (Access Method)
- نسل سوم سیستم مدیریت داده ها DMS
- نسل چهارم نسل DBMS (DataBase Management System)
- نسل پنجم نسل بانک معرفت یا پایگاه شناخت (Knowledge Base System)

نسل فایلهای ساده ترتیبی

- ✓ در این نسل رسانه خارجی معمولاً نوار بوده است.
- ✓ این نسل را می توان نسل بی نرم افزار واسط نیز نامید.
- ✓ در این نسل کاربران مستقیماً با محیط فیزیکی یا سخت افزار کامپیوتر تماس داشتند و داده‌ها را روی آنها ذخیره یا بازیابی می‌کردند.

نسل فایلهای ساده ترتیبی

1. ساختار فایلها ترتیبی است.
2. ساختار فیزیکی همان ساختار منطقی فایل است.
3. تنها روش پردازش فایلها، پردازش یکجا یا دسته‌ای (Batch Processing) است.
4. نرم افزار تنها عملیات ورودی / خروجی را انجام می دهد. نرم افزار واسطی برای مدیریت پردازش فایلها وجود ندارد.
5. طراحی ساختار فیزیکی فایلها هم، برعهده کاربر است.
6. هرگونه تغییر در ساختار داده ها و یا رسانه های ذخیره سازی سبب بروز تغییر در برنامه و بازنویسی و کامپایل آن می شود.
7. اشتراک داده ها مطرح نیست.
8. تکرار در ذخیره‌سازی داده ها (افزونگی) در بالاترین حد است.

نسل شیوه های دستیابی

- ✓ مهمترین ویژگی این نسل را باید پیدایش نرم افزارهای موسوم به «شیوه‌های دستیابی» و همچنین ایجاد رسانه‌های دستیابی مستقیم (یعنی دیسک) دانست.
- ✓ نرم افزار شیوه دستیابی، نرم افزاری است که به جنبه‌های فیزیکی محیط ذخیره سازی و عملیات در این محیط می‌پردازد. به نحوی که دیگر برنامه کاربر نیازی به پرداختن به این جنبه‌ها را ندارد.

نسل شیوه های دستیابی

1. نرم افزار واسط برای ایجاد فایلها با ساختارهای گوناگون بین برنامه های کاربردی و محیط ذخیره سازی وجود دارد.
2. امکان دستیابی ترتیبی و مستقیم به رکوردها (نه فیلدها) وجود دارد.
3. پردازش در محیط های بلادرنگ (Real Time) و برخط (On – Line) بسته به نوع سیستم عامل می تواند انجام شود.
4. ساختار فیزیکی و ساختار منطقی فایلها از یکدیگر جدا هستند ولی نه تا حدی که برنامه‌های کاربردی از محیط فیزیکی ذخیره سازی مستقل شوند. نرم افزاری برای مدیریت وجود نداشت.
5. تغییر در رسانه های ذخیره سازی بر روی برنامه های کاربردی تاثیر چندان ندارد.
6. ایمنی و حفاظت داده ها مطرح بوده ولی روشهای تامین امنیت و حفاظت ابتدایی هستند.
7. تکرار ذخیره‌سازی هنوز در حد نسبتاً بالایی وجود دارد.
8. برای پیاده‌سازی فایل با ارتباط خاصی بین انواع رکوردها (مثلاً ارتباط سلسله مراتبی) خود برنامه‌ساز باید ارتباطات را در برنامه‌اش بسازد.

سیستم مدیریت داده ها

- ✓ در این نسل نرم افزاری کامل تر از نرم افزار دستیابی به عنوان واسط بین برنامه های کاربردی و فایلهای محیط فیزیکی طراحی و ایجاد شد.
- ✓ در این نسل دریافتند که می‌توان برنامه‌های کاربردی را در قبال رشد فایلها (File Growth) مثلاً افزودن یک فیلد به یک نوع رکورد از یک فایل مصون نگاه داشت.
- ✓ تا قبل از این نسل برنامه‌های کاربردی فقط در قبال تغییرات سخت افزاری و رشد کمی فایلها (یعنی افزایش حجم داده های فایل) مصون بودند.

سیستم مدیریت داده ها

1. نرم افزار نسبتًا پیچیده ای به نام سیستم مدیریت داده‌ها واسط بین برنامه کاربردی و محیط فیزیکی ذخیره سازی است.
2. فایلهای منطقی متعددی می توانند از داده های فیزیکی مشترک بهره‌برداری کنند و این فایلها می‌توانند به هم مرتبط باشند.
3. میزان تکرار ذخیره سازی (افزونگی) کاهش یافته است.
4. داده‌های مشترک در کاربردهای متنوع به کار می‌روند.
5. صحت داده‌های ذخیره شده تا حدی تامین می‌شود.
6. آدرس دهی به داده‌ها در سطح فیلد یا گروهی از فیلدها امکان‌پذیر است.

نسل DBMS

- ✓ این نسل از اوایل دهه ۷۰ آغاز شد و هم اکنون نیز ادامه دارد.
- ✓ مهمترین خصیصه این نسل مستقل شدن برنامه‌های کاربردی (Application Program) از جنبه‌ها و خصوصیات محیط فیزیکی ذخیره‌سازی است که اصطلاحاً به آن استقلال داده‌ای فیزیکی (Physical Data Independence) می‌گویند.

نسل DBMS

1. امکان کنترل متمرکز روی تمام داده های عملیاتی وجود دارد و یکی از مزایای این کنترل متمرکز کاهش میزان افزونگی (Redundancy)در ذخیره سازی داده هاست.
2. نرم افزار پیچیده و جامع موسوم به سیستم مدیریت بانک اطلاعاتی (DBMS) واسط بین برنامه های کاربران و محیط داخلی و فیزیکی ذخیره سازی است. این نرم افزار امکان می دهد تا کاربران در یک محیط انتزاعی کار کنند و در عین حال به داده های ذخیره شده دستیابی داشته و عملیات موردنظر خود را انجام دهند.
3. سرعت دستیابی به داده ها بالا می باشد. ایمنی داده ها زیاد است و امکان استفاده اشتراکی از داده ها وجود دارد.
4. در این نسل مفهوم چند سطحی بودن ساختار داده ای و معماری چند سطحی ذخیره سازی مطرح و به تدریج بسط یافت این سطوح از پایین به بالا عبارتند از ساختار فیزیکی بانک، ساختار داخلی بانک، ساختار ادراکی بانک و ساختار خارجی بانک.

نسل بانک معرفت یا پایگاه شناخت

- ✓ این نسل به نسل Knowledge Base یا پایگاه شناخت (پایگاه دانش) معروف است.
- ✓ در این تکنولوژی، ضمن استفاده از تکنولوژی بانکهای اطلاعاتی با بهره گیری از منطق صوری (Formal Logic) سیستم های خبره (Expert System) مفاهیم هوش مصنوعی (Artificial Intelligence = AI) و پردازش زبان طبیعی (Natural Language) سیستمی طراحی و ایجاد می شود که قادر به استنتاج منطقی از داده های ذخیره شده است.
- ✓ تعریف Knowledge Base: مجموعه‌ای از واقعیت‌های ساده و قواعد عام که نشان‌دهنده بخشی از جهان واقعی (Real World) باشد.

DBMS

- اصلی ترین تفاوت نسل DBMS با نسل های قبلی وجود حصاری نفوذناپذیر به نام سیستم مدیریت بانک اطلاعاتی یا DBMS است که هرگونه دستیابی به داده‌ها می بایست از طریق آن انجام شود.
- میزان وابستگی برنامه ها به داده ها از هر نسل به نسل بعدی کاهش می یابد.

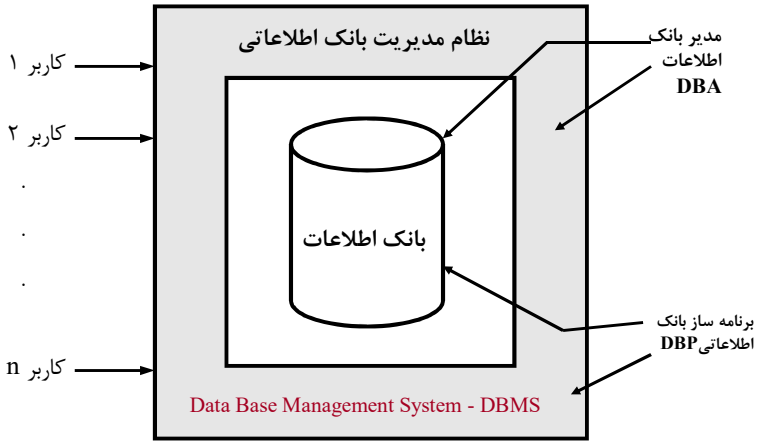
کاربران DBMS

- برای راهبری نرم افزار DBMS، دادن اطلاعات به آن (مثل اطلاعات امنیتی درباره اجازه استفاده کاربران از پرونده های مختلف) و همچنین انجام تغییرات به دو نوع نیروی انسانی نیاز است:

الف) مدیر بانک اطلاعات Data Base Administrator = DBA: که مسئولیت تصمیم گیری، نظارت و طراحی موارد مذکور را بر عهده دارد.

ب) برنامه ساز بانک اطلاعات Data Base Programmer= DBP: تصمیمات مدیر را پیاده سازی می‌کند.

نظام بانک اطلاعات



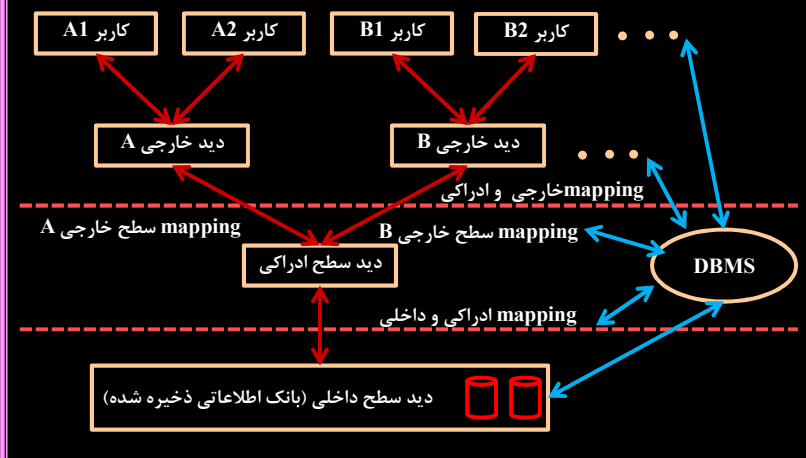
کاربران DBMS

- تنها کسانی که می‌توانند دور از چشم DBMS به داده‌ها دسترسی داشته باشند مدیر و برنامه سازان مجاز بانک اطلاعات هستند.
- مدیر و برنامه سازان مجاز می توانند از طریق نرم افزارهای کمکی (utility) نیز به بانک دسترسی داشته باشند.
- کاربر نهایی End User: کاربران نهایی کسانی هستند که از طریق برنامه‌های تهیه شده، داده‌ها را در حیطه نظارت DBMS دستکاری می‌نمایند.
- عناصر محیط پایگاه داده ها: نرم افزار- سخت افزار- کاربر- داده

معماری سیستم بانک اطلاعاتی ANSI /SPARC

- ✓ چگونه می توان اطلاعات گسترده داده را به شکلی ساده و به دور از پیچیدگیهای مربوط به رسانه ها و داده ها در اختیار کاربران قرار داد؟
- ✓ کمیته ANSI/SPARS معماری ۳ لایه را ارائه داد که بعدها یک لایه به آن افزودند. این معماری ۴ لایه یک معماری کاملتر است.
- ✓ معماری استاندارد پایگاه داده ANSI شامل سطوح زیر می باشند:
  - ❖ سطح خارجی (External)
  - ❖ سطح ادراکی (Conceptual)
  - ❖ سطح داخلی (Internal)

معماری سیستم بانک اطلاعاتی ANSI /SPARC



معماری سیستم بانک اطلاعاتی ANSI /SPARC

- ✓ دید خارجی یا External View
- ❖ دید خارجی، دید خاص هر گروه ازکاربران به داده‌های ذخیره شده در بانک اطلاعاتی است.
- ❖ یعنی اینکه هر کاربر چه قسمتهایی از بانک اطلاعات را اجازه دارد ببیند و چه کارهایی روی آن قسمتها می‌تواند انجام دهد.(امنیت)
- ❖ اصل اول بانک اطلاعات (قانون پنهان سازی اطلاعات): این اصل می‌گوید به هر کس همان مقدار اطلاعات بده که لازم دارد نه بیشتر
- ❖ هر گروه ازکاربران دید خود را دارند و همچنین چند کاربر می‌توانند دارای دید یکسانی باشند. دید خارجی نزدیک‌ترین سطح به کاربران نهایی است.
- ❖ تنها لایه ای است که به کاربران مربوط می شود.

معماری سیستم بانک اطلاعاتی ANSI /SPARC

- ✓ دید ادراکی عام Public Conceptual View
- ❖ لایه دوم لایه تصویر ادراکی عام است. تصویر ادراکی عام یعنی طراحی بانک اطلاعات بدون وابستگی به مدل خاص و پیاده سازی فیزیک خاص. این لایه را کاربر نهایی نمی‌بیند.
- ❖ تصویر ادراکی عام فقط در مرحله طراحی بانک اطلاعات مطرح می‌شود و توسط مدیر بانک طراحی می‌گردد. معروفترین تصویر ادراکی عام EER و NIAM و UML هستند.
- ❖ تصویر ادراکی عام فقط در مستندات وجود دارد و آنچه پیاده سازی شده و مورد استفاده قرار می گیرد تصویر ادراکی خاص است.

معماری سیستم بانک اطلاعاتی ANSI /SPARC

✓ دید ادراکی خاص Spec. Conceptual View

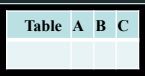
- این لایه، تصویر ادراکی خاص یا همان مدل منطقی است. یعنی اینکه داده‌ها به صورت منطقی چگونه کنار هم قرار می‌گیرند.
- مدل‌های مرسوم جدول، درخت، گراف و مانند این‌هاست.
- در سطح ادراکی خاص ارتباط موجودیتها و صفات خاصه، امنیت و جامعیت داده‌ها مطرح می‌گردد.
- طراحی این لایه به عهده مدیر بانک می‌باشد.
- فقط مدیر بانک و برنامه نویس هستند که این لایه برای آنها قابل استفاده است.

معماری سیستم بانک اطلاعاتی ANSI /SPARC

✓ دید داخلی Internal View

- در این سطح یا دید در واقع فایلهای محیط فیزیکی از نظر محتوا، ساختار و استراتژی دستیابی، تعریف می‌شوند.
- در شمای داخلی، انواع رکوردها، فایلها، صفات خاصه شاخص (استراتژی دستیابی)، نحوه نمایش و تشریح رکوردهای ذخیره شده در فایل، توالی رکوردها، تخصیص فضای ذخیره‌سازی برای داده‌ها، محل رکورد، فشردگی داده‌ای و تکنیکهای رمزگذاری داده‌ها تشریح می‌شوند.
- در یک سیستم بانک اطلاعاتی، کاربران اساساً به مسائل این سطح نمی‌پردازند.
- سطح داخلی نزدیکترین سطح به رسانه ذخیره‌سازی فیزیکی است.
- گذر از یک لایه به لایه بعدی از طریق تبدیل و توسط واحدهای DBMS به طور اتوماتیک انجام می شود.

معماری چهار لایه بانک اطلاعاتی

دیدهای کاربران مختلف (views)	کاربر ۱ ... کاربر ۲	تصویر خارجی
کل بانک بدون توجه به مدلی خاص		تصویر ادراکی عام
کل بانک در قالب مدل انتخابی		تصویر ادراکی خاص
کل بانک روی رسانه ها		تصویر فیزیکی

مدل‌های بانک اطلاعات

- مدل‌های قدیمی
  - سلسله مراتبی: پیاده‌سازی با درخت
  - مدل شبکه‌ای: پیاده‌سازی با گراف
- مدل سنتی (مدل رابطه‌ای)
- مدل‌های جدید (معنایی- فرا رابطه‌ای)
  - مدل شیءگرا
  - مدل تابعی
  - مدل منطقی
  - مدل استنتاجی

مدل قدیمی

- در مدل سلسله مراتبی (hierarchical model) از درخت استفاده می شود و بر مبنای رکورد پایه گذاری شده است. بین هر دو درکورد پشت سرهم ارتباط در یک مسیر درخت پیوند پدر فرزندی وجود دارد.
- در مدل سلسله مراتبی:
  - هر رکورد فرزند فقط یک رکورد پدر دارد.
  - تعدادی محدودیت جامعیت دارد.
  - مبنای ریاضی ندارد.
  - نمایش ارتباط چند به چند در آن مشکل است.
  - همواره جستجو باید از ریشه شروع شود.
  - برای رابطه های یک به چند مناسب است.
  - دستور بازیابی نسبت به مدل رابطه ای مشکل تر است.
  - سادگی نمایش مدل رابطه ای را ندارد.

مدل قدیمی

- در مدل شبکه‌ای (network model) ازگراف برای سازماندهی داده ها استفاده می شود و بر مبنای رکورد پایه گذاری شده است.
- در مدل شبکه‌ای:
  - هر گره فرزند می تواند بیشتر از یک گره پدر داشته باشد.
  - مبنای ریاضی ندارد.
  - در عملیات ذخیره سازی بی نظمی (anomoly) وجود ندارد.
  - امکان ناسازگاری داده ها کمتر از سلسله مراتبی است.
  - دستور بازیابی پیچیده تری نسبت به مدل های دیگر دارد.
  - قواعد جامعیت ذاتی دارد.
  - سادگی مدل رابطه ای را ندارد.
  - جستجوی رکورد نسبت به مدل های دیگر پیچیده تر است.

مدل رابطه‌ای

- در مدل رابطه‌ای داده‌ها به صورت رکوردهای مرتبط سازماندهی می‌شوند و بانک اطلاعات به صورت مجموعه‌ای از رابطه‌ها طراحی می‌شود.
- رابطه ها به صورت جدول پیاده سازی می شوند.
- منطق و دستور بازیابی ساده است.
- امکان نمایش ارتباطات یک به یک، یک به چند و چند به چند وجود دارد.
- نمایش ساده ای از نظر کاربر دارد.
- مبنای تئوریک قوی دارد.
- علل موفقیت مدل رابطه‌ای:
  - سادگی
  - پشتوانه‌ی تئوریک قوی

مدل جدید (معنایی– فرا رابطه ای)

- مدل رابطه ای پاسخگوی بسیاری از نیازهای ما نیست.
- اطلاعاتی مانند صوت و تصویر و multimedia را نمی توان به سادگی در قالب رکورد و رابطه قرار داد.
- از انواع این مدلها:
  - مدل شیءگرا (object-oriented)
  - مدل تابعی (functional)
  - مدل منطقی (logical)
  - مدل استنتاجی (deductive)
- مدل شی گرا حرف اول را می زند.

امنیت و جامعیت

- امنیت security:

محافظت از داده ها در برابر خطراتی از قبیل آتش سوزی و نیز جلوگیری از دسترسی غیر مجاز به آنها. راه های مخالفی برای جلوگیری از دستیابی غیر مجاز وجود دارد که متداولترین آنها استفاده از رمز و الگویتم های خاص جهت تغییر داده. البته همواره کسانی هستند که این رمزها و قفل ها را بگشایند.

- جامعیت integrity:

صحت داده‌ها و پردازش‌ها و پیروی از مقررات سیستم. مثلا موجودی حساب بانکی نباید منفی باشد و یا شخص نتواند بیش از موجودی از حسابش برداشت کند. نوعی جامیت به نام همخوانی (consistency) به این معناست که اقلام داده کل سیستم نباید با هم در تضاد باشند.

تراکنش (transaction)

- هر گونه برنامه‌ای که توسط کاربر در محیط بانک اطلاعات اجرا می‌شود تراکنش نام دارد.
- تراکنش همواره به DBMS تسلیم شده و DBMS در اعمال هرگونه کنترل، به تعویق انداختن و یا ساقط کردن آن تصمیم‌گیری می‌کند.
- هدف اصلی از این کنترل‌ها حفظ جامعیت و صحت بانک اطلاعات است.
- داده‌های بانک اطلاعات را مانا (persistent) می‌نامند زیرا برنامه‌ها می‌آیند و می‌روند اما داده‌ها می‌مانند.

خواص ACID

- برنامه های کاربران به عنوان تراکنش به سیستم مدیریت بانک اطلاعات تحویل می شود. بخشی از یک DBMS که وظیفه مدیریت تراکنش ها را بر عهده دارد، واحد مدیریت تراکنش (transaction management) نام دارد.
- چه کنترل‌هایی لازم است روی برنامه‌ها اعمال شود تا صحت و جامعیت بانک اطلاعات تضمین گردد؟ (Jim Gray 1981)
  - ✓ یکپارچگی (atomicity)
  - ✓ همخوانی ((consistency)
  - ✓ انزوا (isolation)
  - ✓ پایایی (durability)

یکپارچگی (atomicity)

- به معنی "همه" یا "هیچ"
- یا تمامی دستورالعمل‌های یک تراکنش باید اجرا شود و یا هیچکدام از آنها.
- مثال: تراکنشی برای انتقال مبلغی پول از حسابی به حساب دیگر.

  این تراکنش شامل دو بخش است.
- بخش اول، پول را از حساب اول برداشت می‌کند.
- بخش دوم، همان پول را به حساب دوم واریز می‌کند.

همخوانی (consistency)

- این خاصیت می‌گوید که تراکنش باید تمامی قوانین جامعیت بانک اطلاعات را رعایت کند. هر تراکنش اگر به تنهایی اجرا شود بانک اطلاعات را از حالتی صحیح به حالت صحیح دیگری منتقل می کند.
- تراکنش ممکن است دو نوع پایان داشته باشد: پایان ناموفق (سقوط abort)، پایان موفق (انجام commit)
- مثال: در برنامه انتقال پول اگر مبلغ برداشت شده با مبلغ واریز شده به حساب دیگر برابر نباشد تراکنش غلط است.

انزوا (isolation)

- اثر تراکنش‌های همروند روی یکدیگر چنان است که گویا هر کدام در انزوا هستند. (همروندی تراکنش‌ها کنترل می‌شود تا اثر مخرب روی هم نداشته باشند.)
- این عمل توسط بخشی از DBMS به نام واحد کنترل همروندی (concurrency control) انجام می‌شود.

پایایی (durability)

- تراکنش‌هایی که به مرحله انجام (commit) برسند، اثرشان ماندنی است و هرگز بطور تصادفی از بین نمی‌رود.
- دو عمل یکپارچگی و پایایی توسط واحدی از DBMS به نام واحد مدیریت بازگرد (recovery management) انجام می‌گیرد.

لغت نامه (data dictionary)

- در یک بانک اطلاعات اسامی زیادی مورد استفاده قرار می گیرد. از آنجائیکه افراد متفاوتی در یک مجموعه بانک اطلاعات درگیر هستند و اطلاعات در محلهای مختلف جغرافیایی پراکنده هستند مرجعی برای ایجاد یکنواختی و هماهنگی در نام داده ها و معنای آنها ضروری است. این مرجع لغتنامه داده ها نام دارد.
- در مرحله طراحی بانک اطلاعات هرگاه طراح برای مفهومی نامی برمی گزیند باید آن را در لغت نامه داده ها همراه با معنا و فرمت و ... وارد کند. در بانک های اطلاعاتی مدرن از نرم افزارهای ویژه ای برای اضافه کردن اسامی استفاده می شود که از اشتباهات زیر جلوگیری می کند.
  - ✓ یک نام با دو معنای مختلف (homonym)
  - ✓ دو نام برای یک مفهوم (synonym)

کاتالوگ سیستم(system catalog)

- استفاده از کاتالوگ سیستم باعث افزایش استقلال داده ای می شود. علاوه بر اسامی داده ها و مشخصات آنها اطلاعات دیگری در مورد بانک باید نگهداری شود. به عنوان مثال:
  - ✓ حق دستیابی افراد به داده های مختلف
  - ✓ تاریخ ایجاد و به روز درآوردن داده ها
  - ✓ تعداد نسخه های هر پرونده و ترتیب زمانی آنها
  - ✓ ایجاد کننده پرونده ها
  - ✓ اندازه هر جدول یا شئ
- کاتالوگ سیستم اطلاعات لغت نامه داده ها را نیز در بر می گیرد. در واقع، لغت نامه داده ها زیر مجموعه کاتالوگ سیستم است.
- اطلاعات موجود در کاتالوگ سیستم را به اصطلاح دادگان (meta data) به معنی داده در مورد داده می نامند. متاداده از دید کاربر خارجی پنهان است.
- شمای ادراکی و خارجی در کاتالوگ سیستم ذخیره می شوند.

تصویر ادراکی بانک اطلاعات (database schema)

- مجموعه ساختارهای طراحی شده در یک بانک اطلاعات بدون توجه به داده هایی که در آنها قرار می گیرند شمای بانک اطلاعات نام دارد.
- شمای یک بانک اطلاعات را جداول مورد استفاده در آن وستونهای آن جداول تشکیل می دهند. نوع داده هر ستون به شمای بانک مربوط می شود ولی تعداد سطرهای هر جدول ربطی به شمای بانک ندارد.
- در معماری بانک اطلاعات کلمه schema به جای لایه نیز به کار می رود.

استقلال داده ها

- مستقل بودن ذخیره سازی داده ها از کاربرد آنها.
- داده ها در قالب جدول قرار دارند. استفاده کاربر از جداول هیچ ربطی به نحوه ذخیره سازی داده ها ندارد. اگر تغییری در ذخیره سازی داده ها (تعویض نوع دیسک) شود برنامه های کاربردی هیچ تغییری نمی کنند. به این نوع استقلال، استقلال فیزیکی داده ها (physical data independence) می گویند. دیدهای کاربران و برنامه های کاربردی در قبال تغییرات در سطح داخلی مصونیت دارند.
- اگر جدولی چهار ستون داشته باشد و بعدا ستون پنجمی به آن اضافه شود، برنامه های سابق نیاز به دستکاری ندارند و به همان شکل قبلی قابل اجرا هستند. کافی است دیدهای (view) جدید با استفاده از ستون پنجم تعریف شود. این نوع استقلال، استقلال منطقی داده ها (logical data independence) نامیده می شود. دیدهای کاربران و برنامه های کاربردی در قبال تغییرات در سطح ادراکی مصونیت دارند.

زبانهای برنامه سازی بانک اطلاعات

- زبان میزبان (HL:Host Language) یکی از زبانهای برنامه سازی مانند C ،Java و ... می باشد. برای نوشتن یک برنامه کاربردی مورد استفاده قرار می گیرد و از یک زبان دیگر برای ارتباط با پایگاه داده کمک می گیرد.
- در بانک اطلاعات از زبانهای بیانی (declarative) که به آنها زبان پرس و جو نیز(query language) گفته می شود، استفاده می شود.
- در این زبان کافیست کاربر بگوید چه چیزی لازم دارد. اینکه چگونه اطلاعات استخراج می شود از دید کاربر مخفی است.
- برنامه سازی در زبانهای بیانی (declarative) ساده تر از زبانهای روالی (procedural) است زیرا کاربر از بیان روال معاف است.
- قدرت برنامه ساز در زبانهای بیانی محدود می شود زیرا دستورالعمل ها محدود هستند.

زبانهای برنامه سازی بانک اطلاعات

- زبانی که برای ارتباط با پایگاه داده ها استفاده می شود، زبان فرعی داده ها (DSL: Data Sub Language) نامیده می شود.
- در بانک اطلاعات به انواع زبان های زیر نیاز است:
  - زبان تعریف داده ها (DDL: Data Declaration Language)
  - زبان کنترل داده ها (DCL: Data Control Language)
  - زبان کار با داده ها (DML: Data Manipulation Language)
- DSL می تواند به صورت مستقل یا ادغام شده به کار رود.
- زبان HL و DSL می توانند به صورت صریح یا ضمنی ادغام شوند. در ادغام صریح دستورات DSL به صورت صریح در زبان میزبان نوشته می شود و در حالت ضمنی، دستوراتDSL به صورت تابع فراخوانده می‌شود. درحالت ادغام صریح، محیط برنامه سازی دارای دو زبان است که به دو کامپایلر نیاز دارد.

مزایای بانک اطلاعات

1. امکان مدل سازی داده های عملیاتی بر اساس قواعد خاص
2. ذخیره و بازیابی داده ها تحت یک کنترل متمرکز و کارا
3. شاخص گذاری و سایر امکانات ذخیره سازی کارا
4. اشتراک داده ها بین کاربران
5. کاهش افزونگی داده ها
6. سهولت دستیابی به داده ها
7. عدم وجود ناسازگاری
8. تامین جامعیت (integrity) و پیش گیری از تناقض و ابهام
9. تامین امنیت داده ها و اطلاعات
10. امکان ترمیم داده ها (recovery): برگشت سیستم به حالت قبلی درصورت وجود خطا
11. تامین استقلال داده ای در دو سطح فیزیکی و منطقی
12. بهینه سازی پرس و جو
13. دستیابی موازی به داده ها و اطلاعات

معماری پایگاه داده

- نحوه ترکیب اجزای سیستمی شامل حداقل یک پایگاه داده، یک DBMS، یک سیستم عامل و یک کامپیوتر با دستگاههای جانبی و تعدادی کاربر را معماری سیستم پایگاه داده ها می نامند.
  - متمرکز
  - نامتمرکز
    - معماری سیستم پایگاهی مشتری-خدمتگزار(سرویس دهنده- سرویس گیرنده)
    - معماری سیستم پایگاهی توزیع شده (مبادله پیام)
    - معماری با پردازش موازی (سخت افزار مشترک)
    - معماری چند پایگاهی
    - سیستم پایگاهی همراه

معماری پایگاه داده

- معماری سیستم های پایگاهی مشتری-خدمتگزار
- هر معماری که در آن قسمتی از پردازش را یک برنامه، سیستم یا ماشین انجام دهد و انجام قسمت دیگر از برنامه را از برنامه، سیستم یا ماشین دیگر بخواهد. وظیفه ای که باید "سیستم" انجام دهد به دو دسته تقسیم می شود:
  - دسته ای که انجام آن بر عهده خدمتگزار است.
  - دسته ای که انجام آن بر عهده مشتری است.
- مزایای معماری مشتری-خدمتگزار
  - تقسیم پردازش
  - کاهش ترافیک شبکه (در معماری حول شبکه)
  - استقلال ایستگاه های کاری
  - اشتراک داده ها

معماری پایگاه داده

- معماری سیستم پایگاهی توزیع شده
- از ترکیب دو تکنولوژی پایگاه داده ها و شبکه معماری توزیع شده حاصل می شود. مجموعه ای از چند پایگاه داده به هم مرتبط و توزیع شده روی یک شبکه که از نظر کاربران پایگاه واحدی است.
- ویژگی های معماری سیستم پایگاهی توزیع شده
  - مجموعه ای است از داده های منطقاً مرتبط و اشتراکی.
  - بعضی از بخشها ممکن است به صورت تکراری (در چند نسخه) در کامپیوترها ذخیره شده باشند.
  - کامپیوترها از طریق یک شبکه بهم مرتبط اند.
  - داده های ذخیره شده در هر کامپیوتر تحت کنترل یک DBMS است.
  - DBMS در هر کامپیوتر می تواند برنامه های کاربردی محلی را به صورت اتوماتیک اجراء کند.
  - هر DBMS حداقل در اجراء یک برنامه کاربردی سرتاسری مشارکت دارد.

معماری پایگاه داده

- مزایای معماری سیستم پایگاهی توزیع شده
  - سازگاری و هماهنگی با ماهیت سازمان های نوین.
  - کارایی بیشتر در پردازش داده ها به ویژه در پایگاه داده های بزرگ.
  - دستیابی بهتر به داده ها.
  - اشتراک داده ها.
  - افزایش پردازش موازی.
  - کاهش هزینه ارتباطات.
  - تسهیل گسترش سیستم.
  - استفاده از پایگاه داده های از قبل موجود.
- معایب معماری سیستم پایگاهی توزیع شده
  - پیچیدگی طراحی سیستم.
  - پیچیدگی پیاده سازی.
  - کاهش کارایی در برخی موارد.
  - هزینه بیشتر.
  - مصرف حافظه بیشتر.

معماری پایگاه داده

- معماری با پردازش موازی
- تعداد زیادی تراکنش در ثانیه و به صورت موازی اجرا می شود.
- طرح های معماری با پردازش موازی
  - معماری با حافظه مشترک
  - معماری با دیسکهای مشترک
  - معماری با اجزاء مشترک

تصویر ادراکی عام

EER

- در سال ۱۹۷۶ یک دانشجوی دکتری کامپیوتر (Chen) مدل ER (Entity Relationship) برای طراحی بانک اطلاعات پیشنهاد کرد. در طول زمان پیشرفت کرد (EER: Extended ER)
- پدیده یا موجودیت (entity): نمایانگر چیزهایی است که در بانک اطلاعات وجود خارجی دارد و یا به تصویر در می آیند. موجودیت ها دارای مشخصاتی هستند که به آنها صفت (attribute) گفته می شود.
- ارتباط (relationship) پدیده ها را به هم پیوند می دهد و چگونگی در ارتباط قرار گرفتن آنها را با یکدیگر بیان می کند.

EER

تصویر ادراکی عام

- موجودیت (Entity): مفهوم کلی شیئ، چیز، پدیده و به طور کلی هر آنچه که می‌خواهیم در موردش اطلاع داشته باشیم و شناخت خود را در موردش افزایش دهیم.
- انواع موجودیت در دانشگاه عبارتند از: دانشجو، درس، استاد، کارمند، ....
- در رابطه به تشخیص موجودیت سه ضابطه وجود دارد:

1. معمولا نمونه‌هایی متمایز از یکدیگر دارند.
2. معمولا بیش از یک صفت دارد و کاربر به مجموعه‌ای از اطلاعات در مورد آن نیاز دارد.
3. معمولا حالت کنشگری (فاعلیت) یا حالت کنشپذیری (مفعولیت) دارد.

EER

تصویر ادراکی عام

- موجودیت مستقل و وابسته
- موجودیت مستقل (قوی)، موجودیتی که مستقل از هر موجودیت دیگر و به خودی خود، در یک محیط مشخص مطرح باشد(موجودیت درس).
- موجودیت وابسته (ضعیف)، موجودیتی که وجودش وابسته به یک نوع موجودیت دیگر است (موجودیت عضو خانواده برای موجودیت کارمند).

- تعریف صفت
- خصیصه یا ویژگی یک نوع موجودیت است. هر نوع موجودیت مجموعه‌ای از صفات دارد. هر صفت یک نام، یک نوع و یک معنای مشخص دارد.

EER

تصویر ادراکی عام

انواع صفت

- مقدارهیچ پذیر
- کلیدی
- ساده و مرکب
- تک‌مقداری و چندمقداری
- ذخیره شده و مشتق

EER

تصویر ادراکی عام

- صفت مقدار هیچ پذیر: اگر مقدار یک صفت دریک یا بیش از یک نمونه از موجودیت برابر با مقدار هیچ باشد، آن صفت هیچ مقدار پذیر است. نمره دانشجویان که تا پایان ترم مقداری ندارد.
- صفات کلیدی (کلید): یک یا چند صفت که در یک پدیده منحصر به فرد (یکتا) هستند.
- صفت ساده: از لحاظ معنایی تجزیه‌نشدنی است. مثل تعداد واحد در پدیده درس که معمولا یک رقمی است.
- صفت مرکب: صفتی است که هم خودش معنادار است و هم بخشهائی از آن. صفت مرکب صفتی است که از چند صفت ساده تشکیل شده است. مثل صفت نام استاد و دانشجو که ترکیب نام و نام خانوادگی است. آدرس هم یک صفت ترکیبی است.

EER

تصویر ادراکی عام

- صفت تک مقداری: صفت ممکن است تک مقداری باشد مثل نام. هر استاد بیش تر از یک نام ندارد. حداکثر یک مقدار برای یک موجودیت وجود دارد.
- صفت چند مقداری: صفت ممکن است چند مقداری باشد مثل مدرک تحصیلی. برای یک موجودیت چند مقدار وجود دارد.
- صفت ذخیره شده: صفتی که مقادیر آن در پایگاه داده ذخیره شده باشد.
- صفت مشتق: صفتی که در پدیده وجود خارجی ندارد ولی در صورت لزوم می توان آن را بدست آورد. در واقع، صفتی که در پایگاه داده ذخیره نشده است و از طریق داده های ذخیره شده می توان آن را بدست آورد. مثل صفت سن که می توان آن را از روی تارخ تولد بدست آورد یا صفت معدل که از روی نمرات محاسبه می شود.

EER

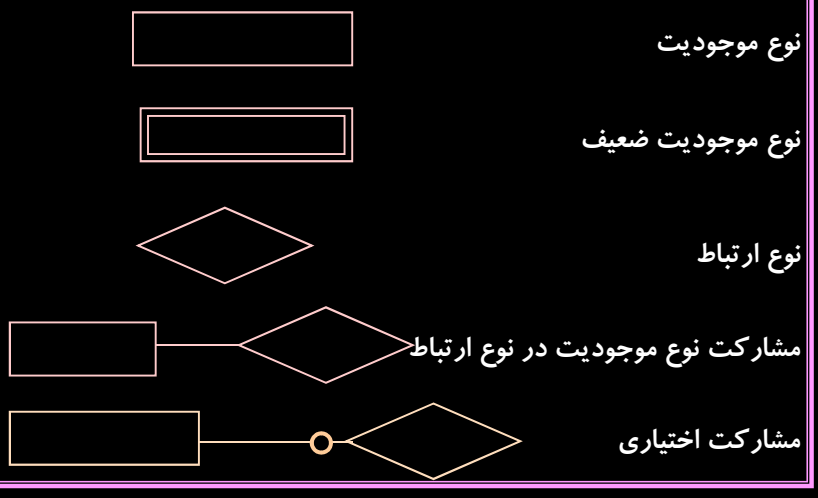
تصویر ادراکی عام

نمادهای رسم نمودار ER

- مدل کلی پایگاه داده ها در بالاترین سطح انتزاع است.
- نموداری است که سه مفهوم اساسی مدل ER، یعنی نوع موجودیت، صفت و ارتباط نمایش داده می‌شوند.
- درجه ارتباط: تعداد پدیده هائی (موجودیت هایی) است که در آن ارتباط مشارکت دارند. درجه ارتباط عدد صحیحی کوچکتر از ۵ است.
- پدیده یک نوع (type) است وعضوهای آن یک مجموعه را تشکیل می‌دهند.

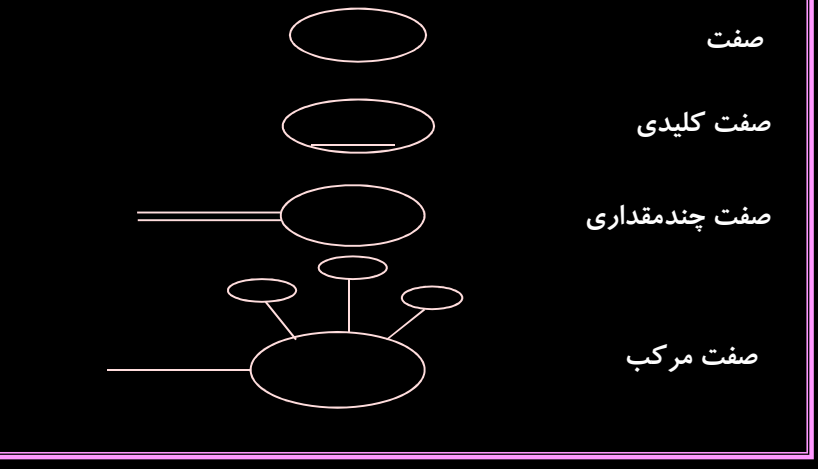
EER

تصویر ادراکی عام



EER

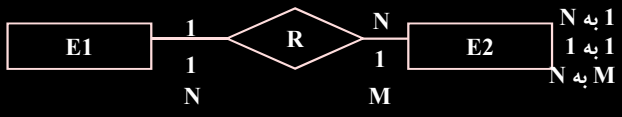
تصویر ادراکی عام



EER

تصویر ادراکی عام

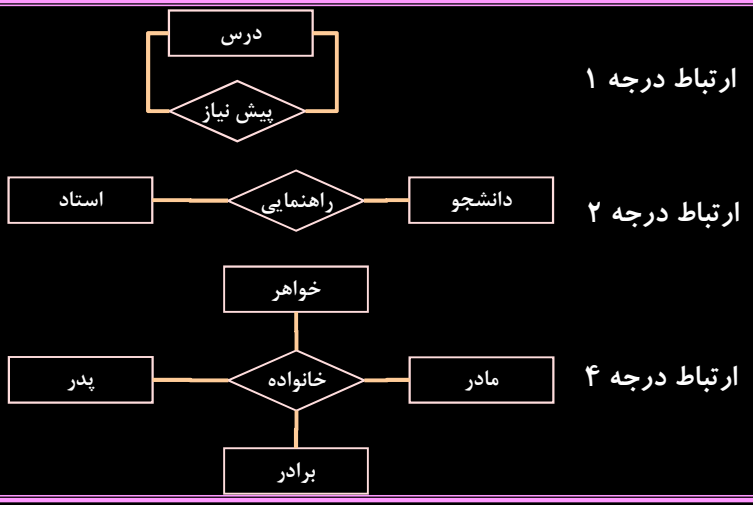
صفت مشتق



درجه ارتباط
1 به N
1 به 1
N به M

EER

تصویر ادراکی عام



ارتباط درجه ۱

ارتباط درجه ۲

ارتباط درجه ۴

EER

تصویر ادراکی عام

- اتصال (connectivity)
- ماهیت نوع ارتباط عبارتست از چگونگی تناظر بین دو مجموعه نمونه‌های آن دو نوع موجودیت. ارتباط از نظر اتصال سه نوع است.

یک به یک 1:1 (هر استاد یک کامپیوتر اختصاصی دارد)

یک به چند 1:N (هر استاد می تواند استاد راهنمای چند دانشجو باشد)

چند به چند N:M (هر درس می تواند چند پیش نیاز داشته باشد و هر درس می تواند پیش نیاز چند درس دیگرباشد)

EER

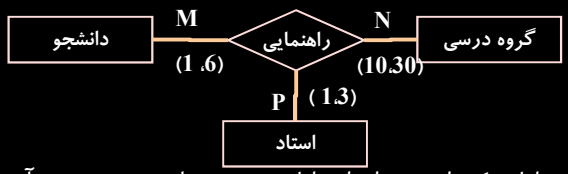
تصویر ادراکی عام

- حد (cardinality)
- تعداد نمونه هایی از یک موجودیت که با تعداد نمونه هایی از موجودیت دیگر ارتباط دارند.
- حد داخل پرانتز نوشته می شود. (حد بالا، حد پایین)

EER

تصویر ادراکی عام

- اتصال و حد برای ارتباط های ۳ و ۴ تایی گاهی معنی دار و گاه مبهم است.



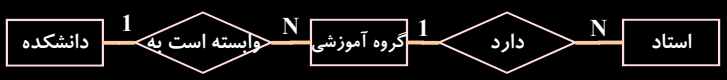
- استنباط غلطی که از نمودار ارتباط موجودیتها به وجود می آید دام یا تله ارتباطی گویند.
- ابهامی که در اینجا وجود دارد این است که مثلا حدود (۳۰ ، ۱۰) مربوط به دانشجو است یا استاد. ارتباط های ۳ تایی و ۴ تایی را به چند ارتباط دو تایی تبدیل می کنند تا ابهامات از بین برود.

EER

تصویر ادراکی عام

- دام یک چندی
- زمانی که ارتباطی بین چند موجودیت وجود داشته باشد، اما مسیر ارتباطی مبهم باشد. درنمودار زیر پاسخ به این سوال که استاد X در کدام گروه عضویت دارد ممکن نیست.

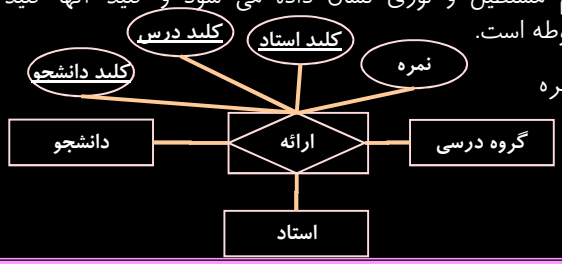
- که به صورت زیر باید اصلاح شود:



EER

تصویر ادراکی عام

- شرکت اجباری یا اختیاری در ارتباط
- شرکت اجباری در ارتباط به طور عادی نمایش داده می شود اما شرکت اختیاری در خط اتصال پدیده به ارتباط یک دایره تو خالی می کشد. مشارکت در یک ارتباط اجباری است اگر همه نمونه های آن موجودیت در آن ارتباط شرکت کنند.
- نیاز به صفت در ارتباط
- ارتباط با ادغام مستطیل و لوزی نشان داده می شود و کلید آنها کلید همه پدیده های مربوطه است.
- مانند صفت نمره



EER

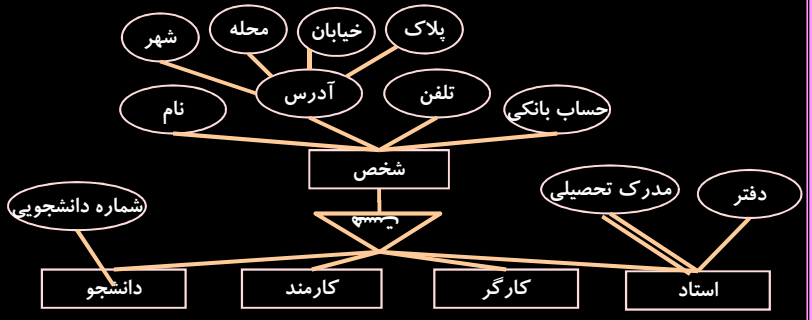
تصویر ادراکی عام

- وابستگی وجودی (existence dependency)
- ممکن است وجود یک موجودیت وابسته به وجود یک موجودیت دیگر باشد. وابستگی را وابستگی وجودی و موجودیت وابسته را موجودیت ضعیف (weak entity) می نامند.
- موجودیت ضعیف کلید پدیده اصلی را به ارث می برد.

EER

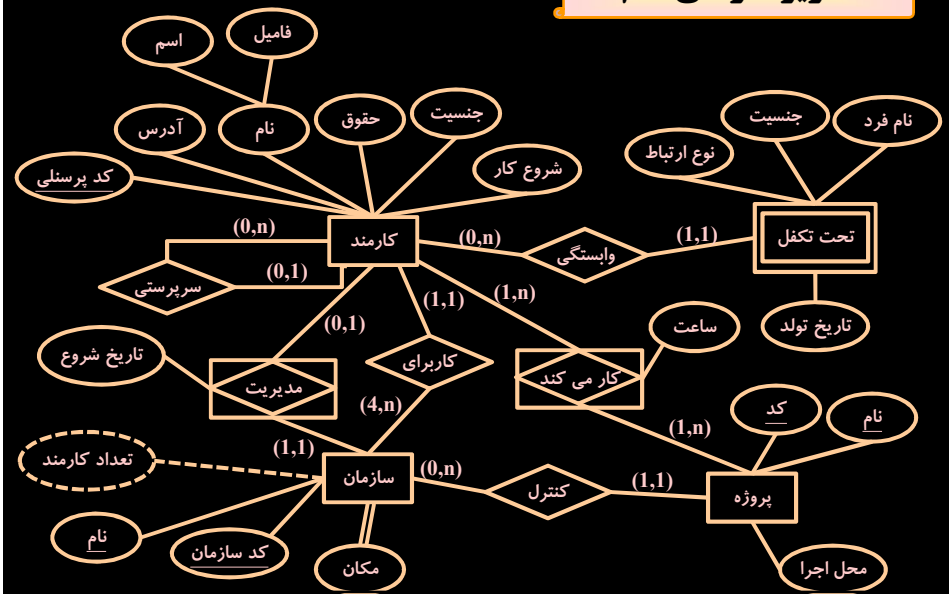
تصویر ادراکی عام

- اشتراک صفت (ارث بری): برای جلوگیری از تکرار بی رویه مشخصات مشترک، ارتباطی از نوع ارث بری تعریف می شود. یک موجودیت مشخصات موجودیت دیگر را به ارث می برد و نیازی به تکرار ندارد. این ارتباط is-a (هست) و با مثلث نمایش داده می شود.



EER

تصویر ادراکی عام



EER

تصویر ادراکی عام

- مدیریت و سرپرستی از کارمندان است.

۱- نگاشت انواع پدیده های عادی

کارمند

کدپرسنلی	اسم	فامیل	آدرس	حقوق	جنسیت	شروع کار

سازمان

کد سازمان	نام

پروژه

کدپروژه	نام	محل اجرا

EER

تصویر ادراکی عام

۲- نگاشت انواع پدیده های ضعیف

تحت تکفل

کدپرسنلی	نام فرد	جنسیت	تاریخ تولد	نوع ارتباط

کلید خارجی

۳- نگاشت انواع ارتباط دودوئی یک به یک

کارمند

کدپرسنلی	اسم	فامیل	آدرس	حقوق	جنسیت	شروع کار

مدیر سازمان و سازمان

کلید خارجی

سازمان

کد سازمان	نام	کد پرسنلی مدیر	شروع کار مدیر

EER

تصویر ادراکی عام

۴- نگاشت انواع ارتباط دودوئی یک به چند

کارمند

کد سرپرست	کد سازمان	شروع کار	جنسیت	حقوق	آدرس	فامیل	اسم	کدپرسنلی
کلید خارجی	کلید خارجی							

یک سازمان چند کارمند دارد. هر کارمند به یک سازمان مربوط است. هر کارمند یک سرپرست دارد.

پروژه

کد سازمان	محل اجرا	نام	کدپروژه
کلید خارجی			

هر سازمان می تواند چند پروژه داشته باشد. هر پروژه مربوط به یک سازمان است.

EER

تصویر ادراکی عام

۵- نگاشت انواع ارتباط دودوئی چند به چند

کار بر روی

ساعت	کدپروژه	کدپرسنلی
	کلید خارجی	کلید خارجی

هر کارمند برای چند پروژه کار می کند. هر پروژه توسط چند کارمند انجام می شود.

۶- نگاشت صفت های چندمقداری

مکان سازمان

مکان	کد سازمان
	کلید خارجی

مدل رابطه ای

- دامنه (domain): مجموعه تمام مقادیر ممکن صفت است.

D_{city}: {c2,c3} $D_{s\#}$: {s1,s2,s3}

- رابطه (relation): زیرمجموعه ای از ضرب دکارتی چند دامنه است. اگر داشته باشیم D_1 و D_2 آنگاه ضرب دکارتی $D_1 \times D_2$ معادل جدولی است که تمام ترکیب های ممکن این دامنه را دربر گیرد.

S#	name	status	city
s1	n1	20	c2
s2	n2	10	c3
s3	n3	30	c3

مدل رابطه ای

- عنوان: مجموعه ای ثابت از صفات خاصه

H_s:{S#, name,status,city}

- بدنه: مجموعه ای است متغیر در زمان از tuple ها

B_s:{(s1,n1,20,c2), (s2,n2,10,c3),…}

- زوج مرتب (تاپل tuple): ارتباط مجموعه ای از مقادیر در یک رابطه است. هر سطر یک جدول معادل یک تاپل است. کاردینالیتی همان تعداد سطرها است.
- میدان: مجموعه ای نامدار از مقادیر همنوع که یک یا بیشتر از یک صفت از آن مقدار می گیرند.
- درجه رابطه: به تعداد میدانهای یک رابطه درجه رابطه گویند.

مدل رابطه ای

- ویژگیهای رابطه عبارتند از:

۱- رابطه تاپل تکراری ندارد.

یک مجموعه عناصر تکراری ندارد. (عناصر مجموعه بدنه تاپل ها هستند)

۲- تاپلها نظم ندارند.

عناصر مجموعه نظم ندارند. (عناصر مجموعه بدنه تاپلها هستند)

۳- صفات رابطه نظم مکانی ندارند.

عناصر مجموعه نظم ندارند. (عناصر مجموعه عنوان صفات هستند)

۴- تمام صفات تک‌مقداری هستند.

۵- تمام صفات اتومیک (تجزیه ناپذیر) هستند. (در صورت تجزیه اجزا بی معنی هستند)

مدل رابطه ای

- کلید (key): هررابطه دارای شناسه ای است که کلید نامیده می شود.
- کلید یک رابطه تمام عضوهای آن را به صورتی منحصر به فرد مشخص می کند.
- کلید تکراری وارد جدول نشود.
- در هر رابطه حداقل یک کلید وجود دارد.
- اجزای کلید باید حداقل باشند.
- مقدار تهی (NULL) برای کلید وارد نشود.

مدل رابطه ای

❖ انواع کلید

۱- ابرکلید (super key): یعنی هر ترکیبی از صفت ها که خاصیت کلید داشته باشد. این تنها نوع کلید است که کمینه (minimal) نیست یعنی زیرمجموعه ای از ان هم ممکن است کلید باشد. «شماره دانشجویی» و «نام دانشجو و شماره دانشجویی»

۲- کلید کاندید (candidate key): هر ترکیبی از صفتهاست که کلید کمینه باشد. یک رابطه ممکن است چند کلید کاندید داشته باشد.

۳- کلید اصلی (primary key): یکی از کلیدهای کاندید است که توسط مدیر بانک اطلاعات انتخاب شده است. نقش و اهمیت این کلید نسبت به بقیه بیشتر است.

مدل رابطه ای

❖ انواع کلید

۴- کلید فرعی (alternative key, secondary key): یکی دیگر از کلیدهای کاندید است که برای برخی کاربردها انتخاب شود. در یک رابطه می توان چند کلید فرعی تعریف کرد.

۵- کلید جانشین: هر کلید کاندیدی که کلید اصلی نباشد.

۶- کلید خارجی (foreign key): صفتی است در یک رابطه که در رابطه دیگری کلید اصلی (یا فرعی) است و برای برقراری ارتباط بین دو رابطه استفاده می شود.

score(شماره دانشجویی، کد درس، نمره)

course(کد درس، نام، تعداد واحد، دانشگاه ارائه دهنده)

مدل رابطه ای

DBMS این خاصیت ها را کنترل می کند.

- قواعد جامعیت (integrity)

در مدل رابطه ای قواعدی باید وجود داشته باشد تا بر اساس آن جامعیت بانک تضمین گردد. صحت، دقت و سازگاری داده‌های ذخیره‌شده در پایگاه در تمام لحظات. در مدل رابطه ای سه نوع جامعیت مورد تاکید قرار می گیرد:

۱- جامعیت دامنه ای (domain integrity): تمام صفات در تمامی رابطه ها از نوع دامنه خود باشد (عدد اعشاری برای شماره دانشجویی پذیرفته نشود)

۲- جامعیت درون رابطه ای (intra-relation integrity): هر رابطه به تنهایی صحیح باشد. عضو تکراری نداشته باشد و کلیدهایش درست باشند و کلیدها دارای مقادیر تهی یا تکراری نباشد.

۳- جامعیت ارجاع (referential integrity): کلید خارجی درست تعریف شده باشد. کلید خارجی یک رابطه حتما در رابطه دیگر کلید باشد و مقداری که به کلید خارجی داده می شود در جدول دیگر وجود داشته باشد.

جبر رابطه ای

- جبر رابطه ای قوی ترین مبنای تئوریک مدل رابطه ای است.
- کاربردهای جبر رابطه ای:
  - ۱- بازیابی داده‌ها
  - ۲- ذخیره‌سازی داده‌ها
  - ۳- تعریف انواع رابطه‌های مشتق
  - ۴- تعریف قواعد برای کنترل پایگاه داده‌ها
  - ۵- تعریف داده‌ها به عنوان حیطه بعضی عملیات کنترل همروندی تراکنشها
  - ۶- ضابطه تشخیص کامل بودن زبانهای رابطه‌ای
- جبر رابطه ای مجموعه ای از عملگرهای رابطه ای به عنوان عملگر (Operator) و رابطه ها به عنوان عملوندها (Operand) می باشد و یک رابطه را به عنوان نتیجه بر می گرداند.
- ورودی و خروجی کلیه عملوندها از نوع جدول است.
- R1 Operator R2 → R3

جبر رابطه ای

- عملگرها در جبر رابطه ای:

۱- عملگرهای ساده :

- گزینش (select) σ
- پرتو (Project) Π

۲- عملگرهای مجموعه ای:

- اجتماع (Union) $\cup$
- اشتراک (Intersect) $\cap$
- تفاضل (Minus) $-$

جبر رابطه ای

۳- عملگرهای پیوند:

- ضرب دکارتی (Times) $\times$
- پیوند شرطی $(\theta - join)$ $\times_\theta$
- پیوند طبیعی $(Natural - join)$ ∞
- نیم پیوند $(Semi - join)$ $\propto$

۴- عملگرهای خاص:

- نامگذاری (Rename) ρ_b^a
- جایگزینی $\leftarrow$
- تقسیم (Divide) $\div$

جبر رابطه ای

- عملگرها به دو دسته عملگرهای اصلی و اضافی تقسیم می شوند. هر عملگر اضافی را می توان با ترکیب عملگرهای اصلی جایگزین کرد
- عملگرهای اضافی برای ساده تر شدن کار استفاده می شود.
- به عنوان مثال اشتراک را می توان با فرمول زیر جایگزین کرد:

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

- عملگرهای اصلی
- $- \quad \times \quad \Pi \quad \cup \quad \sigma$
- عملگرهای اضافی
- $\div \quad \rho_b^a \quad \propto \quad \infty \quad \times_\theta \quad \cap$

جبر رابطه ای

- جداول نمونه (پایگاه داده دانشگاه)

Stud (s#, sname, city, avg, clg#)
دانشجو (شماره دانشکده، معدل کل، شهر محل تولد، نام، شماره)
Prof (pname, office, esp, degree, clg#)
استاد (شماره دانشکده، مدرک تحصیلی، تخصص، دفتر کار، نام)
Crs (c#, cname, unit, clg#)
درس (شماره دانشکده ارائه دهنده، تعداد واحد، نام، شماره)
Sec (sec#, c#, s#, term, pname, score)
گروه درس (نمره، نام استاد، ترم، شماره دانشجو، شماره درس، شماره گروه)
Clg (clg#, clgname, city, pname)
دانشکده (نام رئیس، نام شهر، نام دانشکده، شماره)

جبر رابطه ای

Stud

S#	Sname	city	avg	Clg#
71133848	محمدی	تهران	17.25	10
72130502	وکیلی	اصفهان	14.06	10
72203305	علینقی زاده	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	17.07	6

Crs

C#	Cname	Unit	Clg#
10172	شبیه سازی	3	10
10174	مدار منطقی	3	10
12100	معارف ۱	2	12
12564	ریاضی عمومی ۱	4	1
51515	شیمی عالی	3	5
71203	کنترل خطی	3	7

Sec

Sec#	S#	C#	Term	Score	Pname
1724	71133848	10172	761	14.5	هاشمی اصل
1747	71133848	10174	752	15.75	میرشمسی
1747	72130502	10174	752	12.5	میرشمسی
1748	72203305	10172	761	16.25	قربانی
1516	74182532	51516	752	17	اشرفی زاده

جبر رابطه ای

Clg

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

Prof

Pname	office	esp	degree	Clg#
ابوطالبی	3	مواد	دکتری	6
اشرفی زاده	8	شیمی	دکتری	5
جاهد مطلق	1	کامپیوتر	دکتری	10
جلالی	5	برق	دکتری	7
حسنی	2	ریاضی	دکتری	1
ذاکر	4	فیزیک	دکتری	2
صادقیان	3	صنایع	دکتری	4
قربانی	12	کامپیوتر	دکتری	10
مفتون	1	زبان	دکتری	3
میرشمسی	4	کامپیوتر	فوق لیسانس	10
نقره کار	3	معماری	دکتری	11
هاشمی اصل	10	کامپیوتری	فوق لیسانس	10

جبر رابطه ای

- عملگرهای ساده: گزینش (select)
- فقط شامل یک عملوند می باشد که از نوع رابطه می باشد. روی سطر های جدول اعمال می شود و می تواند از روی سطرها، سطرهایی را که شرط مشخص شده را داشته باشند انتخاب کند. می توانیم چندین شرط داشته باشیم. این شروط باید در مورد یک رابطه باشد.
- مثال: مشخصات دانشجویان یزدی را پیدا کنید.

$$\sigma_{City="\text{یزد}"}(Stud)$$

- حاصل گزینش:

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

- مثال: مشخصات دانشجویان یزدی دانشکده ی شماره ۴ را پیدا کنید.

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

جبر رابطه ای

- عملگرهای ساده: پرتو (project)
- فقط شامل یک عملوند می باشد که از نوع رابطه می باشد. روی ستون های جدول اعمال می شود و می تواند ستون یا ستون های مشخص شده را از جدول ورودی جدا کند.
- مثال: ستون های شماره ی دانشجویی، نام دانشجو و کد دانشکده را از جدول دانشجو انتخاب کنید.

$$\Pi_{S\#,Sname,Clg\#}(Stud)$$

- حاصل پرتو:

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

جبر رابطه ای

- عملگرهای ساده: پرتو (project)
- مثال: شهرهای محل تولد دانشجویان را مشخص کنید.

$$\Pi_{City}(Stud)$$

city
تهران
اصفهان
مشهد
یزد
کرمان
تبریز

- حاصل پرتو:
- نکته: باید دقت داشته باشید که استفاده از عملگر project باعث حذف تکرار خواهد شد به عنوان مثال در مثال قبلی ممکن است چندین دانشجو از یک شهر وجود داشته باشد اما این عملگر تکرار ها را حذف کرده و هر شهر فقط یک بار در خروجی آورده خواهد شد.

جبر رابطه ای

- مثال: ستون های شماره دانشجویی، نام، کد دانشکده و معدل دانشجویانی که معدل آنها بالای ۱۵ است.
- در این مثال هم از عملگرproject و هم از عملگر select استفاده می شود.

1 $\Pi_{S\#,Sname,Clg\#,Avg}(\sigma_{Avg>15}(Stud))$

2 $\sigma_{Avg>15}(\Pi_{S\#,Sname,Clg\#,Avg}(Stud))$

در این مثال هر دو پاسخ صحیح می باشد

S#	Sname	avg	Clg#
71133848	محمدی	17.25	10
72203305	علینقی زاده	16.42	1
73120504	کمانی	17.56	4
73166801	احمدی	15.44	5
74182532	جوادی	16.8	5
74209836	حسین زاده	17.07	6

جبر رابطه ای

- مثال: ستون های شماره دانشجویی، نام و کد دانشکده دانشجویانی که معدل آنها بالای ۱۵ می باشد.
- در این مثال هم از عملگرproject و هم از عملگر select استفاده می شود. اما این مساله که کدام یک زودتر انجام شود اهمیت دارد.

$$\Pi_{S\#,Sname,C\lg\#}(\sigma_{Avg>15}(Stud))$$

- ورودی select، Stud است که یکی از ستونهای آن Avg است و دستور select می تواند شرط مورد نظر را اعمال کند. سپس project ستونهای مورد نظر را از جدول حاصل از نتیجه select را انتخاب می کند. اما پاسخ زیر درست نیست.

$$\sigma_{Avg>15}(\Pi_{S\#,Sname,C\lg\#}(Stud))$$

- در این حالت خروجی project شامل ستون Avg نیست در نتیجه شرط select که روی ستون Avg است اعمال نمی شود.

جبر رابطه ای

- عملگرهای مجموعه ای
- هنگام استفاده از این عملگر ها باید شرط همتا بودن (same arity) بین عملوندهای آن برقرار باشد یعنی:
  ۱. عملوند ها تعداد ستونهای یکسانی داشته باشند.
  ۲. ستونهای متناظر در عملوند ها هم نوع (هم دامنه) باشند.
- به عنوان مثال دستور زیر صحیح نمی باشد:

$$Stud \cup Prof$$

- با اینکه تعداد ستونهای یکسانی دارند، اما ستونهای متناظر آنها هم دامنه نیستند. به عنوان مثال s# با pname هم دامنه نیستند.

جبر رابطه ای

- عملگرهای مجموعه ای: اجتماع، اشتراک، تفاضل
- مثال: لیست نام افرادی که در دانشکده هستند.

$$\Pi_{Sname}(Stud) \cup \Pi_{Pname}(Prof)$$

- مثال: لیست نام دانشجویان هم نام با اساتید.

$$\Pi_{Sname}(Stud) \cap \Pi_{Pname}(Prof)$$

- مثال: لیست نام دانشجویان غیر هم نام با اساتید.

$$\Pi_{Sname}(Stud) - \Pi_{Pname}(Prof)$$

- نکته: باید دقت داشته باشید که ترتیب عملوندها در اجتماع و اشتراک مهم نیست اما در تفاضل مهم است:

$$A \cup B = B \cup A$$
$$A \cap B = B \cap A$$
$$A - B \neq B - A$$

جبر رابطه ای

- عملگرهای پیوند: عملگر ضرب دکارتی یا کارتزین
- یکی از گرانترین عملگرهای بانک رابطه ای است. زمان و فضای زیادی را اشغال می کند. این عملگر دارای دو عملوند میباشد، جدول حاصل از این عملوند مشخص کننده کلیه حالت های ممکن از تلفیق این دو جدول خواهد بود. ممکن است این دو جدول شامل ستونهای همنام باشند در این حالت برای شناسایی ستون مورد نظر از نقطه گذاری استفاده خواهیم کرد.
- در نتیجه ضرب دکارتی دو جدول، جدولی است که ستونهایش همه ستونهای دو جدول و سطرهایش تمام ترکیب های ممکن سطرهای آن دو جدول است. به عنوان مثال:

$$Crs \times Clg$$

- کلیه حالتهای ممکن از تلفیق دو جدول Clg و Crs را به ما خواهد داد.

جبر رابطه ای

- عملگرهای پیوند: عملگر ضرب دکارتی یا کارتزین
- زمانی که برای هر یک مسئله به بیش از یک جدول نیاز داشته باشیم یکی از راه حل ها استفاده از عملگرهای پیوند می باشد.
- مثال: نام و شماره دروسی که توسط استاد قربانی ارائه می شود را مشخص کنید. $\Pi_{cname,crs.c\#}(\sigma_{pname="قرباني" \wedge crs.c\#=sec.c\#}(crs \times sec))$
- نکته: اکثرا زمانی دو جدول را با همدیگر الحاق خواهیم کرد که با همدیگر ارتباط مستقیم داشته باشند. یعنی کلید اصلی یک جدول داخل جدول دیگر به عنوان کلید خارجی قرار داده شده باشد به همین دلیل در مثال قبل، بعد از ضرب دکارتی شرطی که قرار داده شده است برابر بودن کلید خارجی با کلید اصلی می باشد. (crs.c# = sec.c#)

جبر رابطه ای

- عملگرهای پیوند: پیوند شرطی (Teta join)
- پیوند شرطی همانند ضرب دکارتی می باشد با این تفاوت که در پیوند شرطی، شرطی نیز محقق می شود، پس از انجام ضرب این شرط بر روی جدول بدست آمده اعمال شده و نهایتا سطر یا سطرهایی باقی می ماند که شرط مشخص شده را دارا هستند.
- مثال: نام و شماره دروسی که توسط استاد قربانی ارائه می شود را مشخص کنید. $\Pi_{cname,crs.c\#}(crs \underset{pname="قرباني" \wedge crs.c\#=sec.c\#}{\times} sec)$
- ابتدا جدول بزرگی شامل همه ترکیب های ممکن سطری و ستونی دو جدول تشکیل می گردد و سپس بخش کوچکی ار آن به عنوان خروجی انتخاب و بقیه دور ریخته می شود. راه حل بسیار گرانی است که با استفاده از پیوند طبیعی و دستور جایگزینی حل کرد.

جبر رابطه ای

- عملگرهای پیوند: پیوند طبیعی (Natural join)
- از عملگرهای اصلی جبر رابطه ای نیست ولی از معروفترین و کارآمدترین آنهاست. پیوند طبیعی با ضرب دکارتی و پیوند شرطی تفاوتهای زیر را خواهد داشت:

۱. به صورت خود کار شرط تساوی روی ستونهای هم نام اعمال می شود. در صورتی که دو جدول ورودی ستون هم نام نداشته باشند نتیجه پیوند طبیعی برابر ضرب دکارتی خواهد بود.

۲. ستونهای تکراری، حذف خواهند شد و دیگر نیازی به نقطه گذاری نخواهد بود.

۳. پیوند طبیعی بر خلاف ضرب دکارتی و پیوند شرطی عملگر گرانی نیست. ضرب دکارتی و پیوند شرطی عملگرهای گرانی می باشند چون اولا سرعت پایینی دارند و ثانیا به حافظه زیادی نیاز دارند.

جبر رابطه ای

- عملگرهای پیوند: پیوند طبیعی (Natural join)
- مثال: نام و شماره دروسی که توسط استاد قربانی ارائه می شود را مشخص کنید.

1 $\prod_{cname,c\#}(\sigma_{pname="قرباني"}(crs \infty \sec))$

2 $\prod_{cname,c\#}(crs \infty(\sigma_{pname="قرباني"}(\sec)))$

3 $\prod_{cname,c\#}(crs) \infty(\prod_{c\#}(\sigma_{pname="قرباني"}(\sec)))$

جبر رابطه ای

- عملگرهای پیوند: نیم پیوند (Semi-join)
- نیم پیوند همانند پیوند طبیعی می باشد با این تفاوت که ترتیب جداول ورودی مهم است. جدول حاصل فقط شامل ستون های عملوند سمت چپ (اول) خواهد بود.
- مثال: $Stud \propto Sec$
- Stud شامل اطلاعات کلیه دانشجویان می باشد و Sec شامل اطلاعات انتخاب واحد دانشجویان می باشد. پس از انجام نیم پیوند از بین کل دانشجویان، اطلاعات دانشجویانی باقی می ماند که انتخاب واحد کرده اند ولی اطلاعات دانشجویانی که ثبت نام کرده اند اما انتخاب واحد نکرده اند در جدول نتیجه آورده نخواهد شد.
- نکته: باید توجه داشته باشید که $A \propto B \neq B \propto A$

جبر رابطه ای

- عملگرهای پیوند: نیم پیوند (Semi-join)
- مثال: نام و شماره دروسی که توسط استاد قربانی ارائه می شود را مشخص کنید.

$$\Pi_{cname,c\#}(crs) \propto (\sigma_{pname="قرباني"}(sec))$$
$$\Pi_{cname,c\#}(crs) \propto (\Pi_{c\#}(\sigma_{pname="قرباني"}(sec)))$$

- مشخصات کامل رؤسای دانشکده ها.

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

جبر رابطه ای

- عملگرهای پیوند: نیم پیوند (Semi-join)
- خروجی $clg \propto prof$ دستور چیست؟
- دو جدول clg و prof در ستونهای clg# و pname با همدیگر مشترک می باشندو این ستونها در این جداول باید با همدیگر برابر باشند. در نتیجه حاصل اطلاعات اساتیدی می باشد که رییس دانشکده خودشان می باشند.
- مثال: خروجی دستور روبرو چیست؟ $\sigma_{term=771 \wedge clg\#=1}(crs \propto sec)$
- ظاهراً جواب این دستور مشخص کننده دروسی می باشد که در ترم اول سال ۷۷ در دانشکده شماره ۱ ارائه شده است اما جواب صحیح نمی باشد. چون جدول حاصل از $crs \propto sec$ شامل ستون ترم نمی باشد.
- پاسخ صحیح $\sigma_{term=771}(crs) \propto \sigma_{clg\#=1}(sec)$

جبر رابطه ای

- عملگرهای خاص: نام گذاری (Rename)
- عملگر نامگذاری با علامت ρ_b^a مشخص می شود با این کار نام b نیز بر روی جدول a گذاشته می شود. زمانیکه مجبور باشیم در دو طرف عملگر join از یک جدول استفاده کنیم، با استفاده از این عملگر می توان نام یکی از عملوندها را به صورت موقت تغییر داد. به عنوان مثال اگر بخواهیم اساتیدی که دفتر کارشان مشترک باشد را پیدا کنیم خواهیم داشت:

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

- ابتدا ستونهای نام استاد و دفتر او را از جدول استاد جدا با ρ نامگذاری می شود. سپس سطرهایی از prof که با ρ دفتر کار یکسانی دارند ولی نام متفاوتی دارند انتخاب می شوند.

جبر رابطه ای

- عملگرهای خاص: جایگزینی
- از این عملگر استفاده خواهیم کرد تا بتوانیم جدول حاصل از دستورات را برای استفاده های بعدی ذخیره کنیم.
- درصورتی که دستورات طولانی باشد می توان با استفاده از جایگزینی آن را در چند مرحله نوشت.
- مثال: اطلاعات دروسی را مشخص کنید که توسط استاد قربانی تدریس می شوند.

$$temp \leftarrow \prod_{c\#}(\sigma_{pname="قرباني"}(sec))$$
$$crs \infty temp$$

جبر رابطه ای

- عملگرهای خاص: تقسیم (Divide)
- کاربرد عملگر تقسیم زمانی می باشد که بخواهیم همه حالتهای یک اتفاق را بررسی کنیم. به عنوان مثال:
- دانشجویانی که همه درس های استاد تبریزی را انتخاب کردند.
- اساتیدی که در همه دفاتر کار کرده اند.
- درسهایی که توسط همه دانشکده ها ارائه می شوند.
- برای پاسخگویی به پرسش های فوق باید از عملگر تقسیم استفاده کنیم. هنگام استفاده از این عملگر اولین کار مشخص کردن مقسوم علیه می باشد. مقسوم علیه بخشی از مساله می باشد که شامل شرط همه است. سپس مقسوم را بر مقسوم علیه تقسیم کرده تا خروجی که شامل صفت های باقیمانده است بدست آید.

جبر رابطه ای

- عملگرهای خاص: تقسیم (Divide)
- هنگام استفاده از این عملگر باید نکات زیر رعایت شوند:

۱. صفات موجود در مقسوم‌علیه باید زیر مجموعه‌ای از صفات مقسوم باشد.

۲. جدول حاصل شده از تقسیم شامل صفات مقسوم به غیر از صفات مشترک آن با مقسوم علیه خواهد بود.

- مثال: دانشجویانی که همه درسهای استاد تبریزی را انتخاب کرده اند.
- $stud \div temp$ اشتباه است زیرا صفت c# در stud وجود ندارد.
- خروجی دارای ستونهای s# و Sname خواهد بود.

$$temp \leftarrow \prod_{c\#}(\sigma_{pname="تبریزی"}(\text{sec}))$$
$$\prod_{s\#,sname,c\#}(stud \infty \text{sec}) \div temp$$

جبر رابطه ای

- بهینه سازی پرس و جو
- برای یک مساله ممکن است جواب های زیادی وجود داشته باشد اما سعی خواهیم کرد از بین جوابها جواب بهینه را انتخاب کنیم.
- دستوری را انتخاب خواهیم کرد که سرعت بالا و مصرف حافظه ی کمتری داشته باشد.
- برای بهینه سازی سعی می کنیم تا حد ممکن تا قبل از join کردن اندازه جداول را کوچک کنیم، این کوچکتر کردن می تواند از لحاظ تعداد ستون یا از لحاظ تعداد سطر انجام شود.

جبر رابطه ای

- بهینه سازی پرس و جو
- مثال: دروس ۴ واحدی که در ترم اول سال ۷۷ (۷۷۱) ارائه شده اند.

1 $\sigma_{unit=4}(crs \propto (\sigma_{term=771}(sec)))$

2 $\sigma_{unit=4}(crs) \propto (\sigma_{term=771}(sec))$

3 $\sigma_{unit=4}(crs) \propto (\sigma_{term=771}(\prod_{rerm,c\#}(sec)))$

4 $\sigma_{unit=4}(crs) \propto (\prod_{c\#}(\sigma_{term=771}(sec)))$

- جوابها از جواب شماره ۱ تا جواب شماره ۴ به ترتیب بهینه می شوند.

جبر رابطه ای

- بهینه سازی پرس و جو
- قواعد بهینه سازی
- تا حدی که ممکن است سعی کنیم عملگر select را قبل از بقیه عملگر ها استفاده کنیم.
- شرطهای ترکیبی را به شرطهای ترتیبی (متوالی) تبدیل کنید به عنوان مثال می توانیم $\sigma_{p1\wedge p2}(e)$ را با $\sigma_{p1}(\sigma_{p2}(e))$ جایگزین کرد.
- تا حد ممکن عملگرproject را زودتر انجام دهیم ولی و دیرتر از select.
- عملگرهای ترکیبی مانند عملگرهای مجموعه ای و پیوند در انتها بیایند.

جبر رابطه ای

- بهینه سازی پرس و جو
- عملگرهای معادل
- می توان برای بهینه سازی از دستورات معادل استفاده کرد، به عنوان مثال $crs \infty sec$ از لحاظ جواب معادل $sec \infty crs$ می باشد اما کارایی آنها با هم برابر نیست، به عنوان مثال فرض کنید جدول crs جدول کوچکی می باشد و می توانیم کل آن را یکجا روی حافظه داشته باشیم اما جدول sec جدول کوچکی نمی باشد و می توانیم بخش کوچک آن را روی حافظه داشته باشیم.
- هنگام join کردن سعی خواهیم کرد بجای اینکه یک جدول کوچک را با یک جدول بزرگ join کنیم، یک جدول بزرگ را با یک جدول کوچک join خواهیم کرد.

جبر رابطه ای

- بهینه سازی پرس و جو
- عملگرهای معادل

$$A \infty B = B \infty A$$
$$(A \infty B) \infty C = A \infty (B \infty C)$$
$$A \times B = B \times A$$
$$(A \times B) \times C = A \times (B \times C)$$
$$A \cup B = B \cup A$$
$$(A \cup B) \cup C = A \cup (B \cup C)$$
$$A \cap B = B \cap A$$
$$(A \cap B) \cap C = A \cap (B \cap C)$$
$$\sigma_p (A \cap B) = \sigma_p (A) \cap \sigma_p (B)$$
$$\sigma_p (A - B) = \sigma_p (A) - B = \sigma_p (A) - \sigma_p (B)$$

جبر رابطه ای

- به روز درآوردن داده ها
- تغییر داده های جداول در سه بخش انجام می شود.
- افزودن داده به جداول - Insert
- حذف داده ها از جداول - delete
- تغییر داده های جداول - update

جبر رابطه ای

- به روز درآوردن داده ها
- اضافه کردن داده به جداول
- اضافه کردن داده به جداول با استفاده از U و ← انجام می شود.
- مثال: می خواهیم اطلاعات درس جدیدی به نام database را به جدول crs اضافه کنیم:

$$crs \leftarrow crs \cup \{"C14","database",3,4\}$$

- در این مثال با یک دستور فقط یک سطر به جدول اضافه می شود.
- جدول goodStud را در نظر بگیرید که فقط شامل صفات sname و avg می باشد. اگر بخواهیم اطلاعات دانشجویان ممتاز دانشکده شماره ۱۰ از جدول Stud را به این جدول اضافه کنیم خواهیم داشت:

$$goodstud \leftarrow goodstud \cup (\prod_{sname,avg}(\sigma_{avg\geq 17 \wedge clg\#=10}(stud)))$$

جبر رابطه ای

- به روز درآوردن داده ها
- حذف داده ها از جداول
- این کار در جبر رابطه ای با استفاده از تفاضل و جایگزینی انجام می شود. با این کار ممکن است یک یا چند سطر از جدول حذف شود یا ممکن است هیچ سطری از جدول حذف نشود.
- مثال: دانشجویانی که معدل آنها کمتر از ۱۸ می باشد از جدول goodstud حذف کنید:

$$goodstud \leftarrow goodstud - (\sigma_{avg<18}(goodstud))$$

- چند بار استفاده از یک جدول در این دستور مشکلی ایجاد نمی کند زیرا عملیات مختلف آن به ترتیب انجام می شود.

جبر رابطه ای

- به روز درآوردن داده ها
- تغییر داده های جداول
- این کار در جبر رابطه ای با استفاده از گزینش و جایگزینی انجام می شود.
- مثال: افزودن یک واحد به تمام دروس

$$\sigma_{unit \leftarrow unit+1}(crs)$$

- مثال: تغییر تعداد واحد درس database از ۳ به ۴

$$\sigma_{unit \leftarrow 4}(\sigma_{cname="database"}(crs))$$

حساب رابطه‌ای

- حساب رابطه‌ای، با جبر رابطه‌ای منطقا معادل است، یعنی برای هر عبارت جبر رابطه‌ای، یک عبارت معادل در حساب رابطه‌ای وجود دارد و برعکس. تفاوت آنها این است که جبر رابطه‌ای، دستوری است، اما حساب رابطه‌ای توصیفی است.
- حساب رابطه ای دونوع تاپلی و میدانی (دامنه ای) دارد.
- حساب تاپلی: در این حساب مفهوم مهمی به نام متغیر تاپلی وجود دارد که تنها مقادیر مجازش، تاپلهای رابطه هستند.
- حساب میدانی: در این حساب، متغیر میدانی وجود دارد که از یک میدان مقدار می‌گیرد. در این حساب یک شرط اضافی به نام شرط عضویت وجود دارد. بجای متغیر تاپلی، متغیر میدانی (domain) داریم.

حساب رابطه‌ای

- در حساب رابطه ای تاپلی دو سور وجود دارد:
- ۱- سور وجودی: به صورت $\exists T(f)$ نوشته می‌شود، به این معنا که حداقل یک مقدار برای متغیر T وجود دارد به نحوی که fبه "درست" ارزیابی شود.
- ۲- سور همگانی: به صورت $\forall T(f)$ نوشته می‌شود. یعنی به ازاء تمام مقادیر متغیر T، fبه "درست" ارزیابی می‌شود.
- در حساب رابطه ای دامنه ای شرط عضویت به صورت زیر نوشته می شود:
- $R\ (A_1{:}v_1\ ,\ A_2{:}v_2,\ \dots\ ,\ A_n{:}v_n)$
- R نام رابطه، A_i نام صفت، v_i یک مقدار از میدان و یا یک لیترال است. وجود این شرط به مفهوم "درست" ارزیابی می شود اگر و تنها اگر تاپلی در R داشته باشد که مقادیر داده شده برای صفات را داشته باشد.

زبان پرس و جوی SQL

- زبان SQL (Structured Query Language) پیاده سازی آزادی از جبر رابطه ای است که بعضی از عملگرها مثل تقسیم را نمی پوشاند.
- SQL زبانی بیانی (declarative) است یعنی کاربر آنچه را لازم دارد با دستورات محدودی بیان می کند و سیستم آن را تفسیر می کند.
- زبان SQL بخش های زیر را دارد:
- Data Definition Language (DDL): create, drop
- Data Manipulation Language (DML): delete, insert, update, select
- Data Control Language (DCL): grant, revoke

زبان پرس و جوی SQL

- انواع داده (رشته)

Data type	Description	Max size	Storage
char(n)	Fixed width character string	8,000 characters	Defined width
varchar(n)	Variable width character string	8,000 characters	2 bytes + number of chars
varchar(max)	Variable width character string	1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string	2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string	4,000 characters	Defined width x 2
nvarchar	Variable width Unicode string	4,000 characters	
nvarchar(max)	Variable width Unicode string	536,870,912 characters	
ntext	Variable width Unicode string	2GB of text data	
binary(n)	Fixed width binary string	8,000 bytes	
varbinary	Variable width binary string	8,000 bytes	
varbinary(max)	Variable width binary string	2GB	
image	Variable width binary string	2GB	

زبان پرس و جوی SQL

- انواع داده (عددی)

Data type	Description	Storage
bit	Integer that can be 0, 1, or NULL	
tinyint	Allows whole numbers from 0 to 255	1 byte
smallint	Allows whole numbers between -32,768 and 32,767	2 bytes
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
decimal(p,s)	Fixed precision and scale numbers.	5-17 bytes
numeric(p,s)	Fixed precision and scale numbers.	5-17 bytes
smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
float(n)	Floating precision number data from -1.79E + 308 to 1.79E + 308.	4 or 8 bytes
real	Floating precision number data from -3.40E + 38 to 3.40E + 38	4 bytes

زبان پرس و جوی SQL

- انواع داده (تاریخ و زمان)

Data type	Description	Storage
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	6-8 bytes
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 bytes
date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset	8-10 bytes
timestamp	Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable	

زبان پرس و جوی SQL

❖ تعریف بانک اطلاعات

```
create Database   اسم پایگاه داده
on primary
(name = اسم فایل (منطقی),
filename = ‘اسم فیزیکی و مسیر.mdf‘,        اطلاعات فایل
size=اندازه (به مگابایت),
maxsize=حداکثر اندازه (پس از داده) ,
filegrowth=واحد افزایش)
LoG on
(name= logاسم فایل ,                       اطلاعات log
filename=‘اسم فیزیکی و مسیر.ldf‘,
size= MB,
Maxsize= MB,
filegrowth=MB)
```

زبان پرس و جوی SQL

❖ تعریف بانک اطلاعات

```
create Database  Test
on primary
(name = logictest,
filename = 'c:\physictest.mdf',
size=10,
maxsize= 30,
filegrowth=5)
LoG on
(name= logictestlog,
filename = 'c:\logictest.ldf',
size=10,
maxsize= 20,
filegrowth=5 )
```

برای درست کردن جداول در فایل ها نام منطقی به کار می رود.

زبان پرس و جوی SQL

- حذف پایگاه داده

Drop database نام پایگاه داده

Drop database Test

- توضیحات تک خطی

--This is single line comment

- توضیحات چند خطی

```
/*
This is multiple line comment
*/
```

- علائم

<	<=	>	>	>=	<> or !=

زبان پرس و جوی SQL

- تعریف جداول

```
create table نام جدول
(نام ستون  نوع متغیر [Not NULL],
  .
  .
[primary key (نام ستون جدول)],
[unique (نام ستون)],
[foreign key (نام ستون)
references نام جدول مبدا (نام ستون مبدا)]
                [on delete cascade]
                [on update cascade]
[check (محدودیتها)])
```

یکتا بودن مقادیر ستون

ارجاع به

زبان پرس و جوی SQL

- تعریف جداول

```
create table student
(stNo smallint,
 stName char(20) Not NULL,
 stLastname char(80),
 primary key (stNo))
```

زبان پرس و جوی SQL

- دستور check برای بیان قوانین جامعیتی است.

```
create table course
(cNo smallint Not NULL,
 cName char(30) Not NULL,
 unit smallint Not NULL,
 primary key (cNo),
 unique (cName),
 check (unit>0 AND unit<5))
```

cNo	cName	unit
134	C++	3
135	Database	3
421	English	2
531	Person	2

اسامی جدولها باید منحصربفرد باشد، اسامی ستونهای یک جدول منحصربه فرد باشند.
علامت # در SQL استاندارد پذیرفته نیست ولی در بسیاری از پیاده سازیها پذیرفته می شود.

زبان پرس و جوی SQL

- تغییر جدول (تغییر نوع)

➢ Alter table نام جدول
alter column تغییر اسم ستون

Alter table course
alter column cName char(40)

- تغییر جدول (حذف ستون)

➢ Alter table نام جدول
drop column اسم ستون

Alter table course
drop column unit

زبان پرس و جوی SQL

- تغییر جدول (اضافه نمودن ستون)

➢ Alter table نام جدول
ADD نوع اسم ستون

Alter table course
ADD state char(10)

- حذف جدول

➢ drop table نام جدول

- شرح جدول از کاتالوگ سیستم برداشته می شود.
- تمام شاخص ها و دیدهای ایجاد شده روی جدول از بین می رود.
- تمام کلیدهای خارجی تعریف شده روی جدول از بین می روند.

زبان پرس و جوی SQL

- وارد کردن داده

➢ INSERT INTO نام جدول
VALUES (لیست مقادیر)

```
INSERT INTO course
    VALUES (56, 'c', 2)
```

- اگر مقادیر متناظر با جدول نباشند و یا بعضی مقادیر وارد نشود (NULL)

➢ INSERT INTO [(ستونها)]
VALUES (لیست مقادیر)

```
INSERT INTO course (cNo,unit,cName)
    VALUES (33,4, 'Pascal')
```

زبان پرس و جوی SQL

- جداول نمونه

Number

sNo	sName	city	avg	born	course
86001	Kamali	C2	14	62	C++
86002	Kiani	C1	17	64	Database
86003	Javadi	C3	18	63	C++
86004	Niazi	C2	15	62	English
86005	Keyvani	C3	19	65	Persian

course

cNo	cName	unit
1	C++	3
2	Database	3
3	English	2
4	Persian	2
5	Data	3

زبان پرس و جوی SQL

- استخراج اطلاعات

- select [All] [Distinct] نام ستونها from نام جدولها
  [where شرایط] [ORDER BY نام ستونها] [Having شرایط] [Group by نام ستونها]
- select * from course
- select cName from course
- select cName, unit from course
- نکته: همیشه نتیجه دستور select یک جدول است.

زبان پرس و جوی SQL

- استخراج اطلاعات

- select Distinct unit from course
  تکرارها را چاپ نمی کند. پیش فرض ALL است.
- select sName, avg*100 from Number
  عنوان ستونهای خروجی: sName,no column name
- select sName, avg*100 as avg100 from Number
  عنوان ستونهای خروجی: sName,avg100
- با استفاده از where می توان دامنه انتخاب را محدودتر کرد.

زبان پرس و جوی SQL

- استخراج اطلاعات
  - select sName, city from Number where city='C3'
  - select sNo, avg from Number where avg>15
  - select * from Number where born != 62
  - select * from Number where born <> 62
  - select * from Number where city IS NULL
  - select sNo, sName, (born+2) as secondYear from Number

زبان پرس و جوی SQL

- استخراج اطلاعات
  - عملگرهای منطقی: and, or, Not
  - مقدار یک عبارت منطقی را برعکس می کند (not)
  - در صورتی که هر دو عبارت درست باشد (and)
  - در صورتی که هر دو عبارت یا یکی از آنها درست باشد (or)
  - select * from Number where avg=17
  - select * from Number where not avg=17
  - select * from Number where avg>=17 or city<>'C2'
  - select * from Number where avg=15 and city='C2'

زبان پرس و جوی SQL

- استخراج اطلاعات
- برای انتخاب مقادیر از like استفاده می شود. کوچک و بزرگ بودن در این دستور مهم است.
- مقدار دقیق یک جزء پایگاه داده را نمی دانید.
- select * from Number where sName like ‘Kiani’
- % : هر کاراکتری و به هر مقداری
- _ : تنها یک کاراکتر
- select * from Number where sName like ‘K%‘
- select * from Number where sName like ‘_a%'

زبان پرس و جوی SQL

- استخراج اطلاعات
- اگر بخواهید خروجی با ترتیب خاصی مرتب شود از این قسمت استفاده می شود.
- select * from Number

  where born>62 order by born asc
- پیش فرض صعودی (asc) است.
- select sName, avg from Number

  where city=‘C2‘ order by avg desc

زبان پرس و جوی SQL

- استخراج اطلاعات
- اتصال دو رشته ||

```
select sName || city as Ncity
                from Number
```

- Between: انجام بعضی از اعمال با استفاده از این اپراتور راحت تر است.

```
select  * from Number where (born Between 62 and 63)
select  * from Number where born >= 62 and born<=63
```

زبان پرس و جوی SQL

- استخراج اطلاعات
- iN:

```
select  * from Number where (born iN(62,63,64))
select  * from Number where (born = 62 or born=63 or born=64)
select  sName, avg, born from Number where (born not in (62,63))
```

- ۱۰۰۰ رکورد اول را نشان می دهد:

```
select top 1000 * from Number
```

زبان پرس و جوی SQL

- استخراج اطلاعات
  - ضرب دکارتی در جداول
  - select * from course, Number (اتصال دو جدول)
  - select *,* from course, Number (خروجی دوبرابر)
  - cross join
  - select course.cNo,course.cName,Number.sName, Number.course from course, Number
    where course.cName=Number.course
  - select course.cNo,course.cName,Number.sName, Number.course from course inner join Number
    on cName=course

زبان پرس و جوی SQL

- استخراج اطلاعات
  - select cNo, cName, sName, course from course inner join Number on course.cName=Number.course
  - هر رکورد جدول course را با تمام رکوردهای جدول Number می نویسد.
  - select cNo, cName, sName, course from course Left outer join Number on cName=course
  - درصورتیکه outer نباشد اگر شرط برقرار باشد می نویسد، اما اگر بخواهیم همه course ها را بنویسد و اگر معادل در Number نداشت Null بگذارد از Left استفاده می شود.

زبان پرس و جوی SQL

- استخراج اطلاعات

- select cNo, cName, sName, course from course Right outer join Number on cName=course
- درصورتیکه بخواهیم تمام رکوردهای Number نوشته شود حتی اگر شرط برقرار نباشد معادل آن در course، Null قرار دهد از Right outer join استفاده می شود.
- select cNo, cName, sName, course from course full outer join Number on cName=course
- select course cross join Number
- معادل است با
- select * from course,Number

زبان پرس و جوی SQL

- استخراج اطلاعات

- select * from course, Number where course.cName=Number.course and Number.avg != 14
- union: اجتماع دو جدول (رابطه) است. برای اجتماع دو ستون باید سازگار باشد به این معنی که نوع داده یکسانی داشته باشند.
- select cName from course
  Union
  Select sName from Number
- تمام نامها را بدون وجود تکرارها نمایش می دهد.

زبان پرس و جوی SQL

- استخراج اطلاعات

```
select cName from course
Union all
select sName from Number
```

- تمام نامها را به همراه تکرارها نمایش می دهد.

```
select cName, unit from course
Union
select sName, born from Number
```

- نام ستون اول را قرار می دهد.

زبان پرس و جوی SQL

- توابع گروهی

- Count: تعداد سطرهایی را که در شرط where صدق می کند، بر می گرداند.

```
select  Count(*) from Number where city ='C2'
select  Count(Distinct sName) from Number
```

- تکرارها را نمی شمارد.

```
select  Count(sName) Num_k from Number
    where sName like 'K%'.
```

زبان پرس و جوی SQL

- توابع گروهی
- Sum: مجموع مقادیر موجود در یکی از ستونهای رکوردهای انتخابی را بازمی گرداند.
- select Sum(unit) as Total_unit from course
- select Sum(unit) Total_unit from course
- select Sum(avg) Total_avg from Number
  where born>62
- select Total_avg=Sum(avg) from Number
  where born>62
- select Sum(avg), Sum(born) from Number

زبان پرس و جوی SQL

- توابع گروهی
- Max: بزرگترین مقدار موجود در یک ستون را برمی گرداند.
- select Max(avg) from Number
- select sName from Number
  where avg=(select Max(avg) from Number)
- این تابع روی مقادیر غیرعددی نیز به کار می رود.
- select Max(sName) from Number
- به z نزدیکتر باشد.

زبان پرس و جوی SQL

- توابع گروهی
  - Min: کوچکترین مقدار موجود در یک ستون را برمی گرداند.
  - select Max(avg), Min(born) from Number
  - select sNo from Number
    where avg <(select Max(avg) from Number)
  - شماره دانشجویانی که معدل آنها از شاگرد اول کمتر باشد.

زبان پرس و جوی SQL

- توابع گروهی
  - AVG: میانگین مقدار موجود در یک ستون را برمی گرداند. مانند تابع Sum روی اعداد کار می کند.
  - select AVG(born)/ AVG(avg) from Number
  - select sNo from Number
    where born>=(select AVG(born) from Number)
  - شماره دانشجویانی که سال تولد آنها از میانگین سن دانشجویان مساوی یا بیشتر باشد.

زبان پرس و جوی SQL

- ABS(): قدر مطلق عدد مورد نظر را بازمی گرداند.
- SQRT(): جذر عدد مورد نظر را بازمی گرداند.
- Power(a,b): عدد اول را به توان عدد دوم رسانده و نتیجه را باز می‌گرداند.
- UPPER(): رشته را به معادل بزرگ کاراکترها تبدیل می کند.
- Lower(): رشته را به معادل کوچک کاراکترها تبدیل می کند.
- Len(): طول رشته را برمی گرداند.

➢ select sName, len(sName) from Number

زبان پرس و جوی SQL

- Group by با پنج تابع گروهی کار می کند.

➢ Select born, Sum(avg) from Number

Group by born

born	Sum(avg)
62	29
63	18
64	17
65	19

زبان پرس و جوی SQL

- Group by با پنج تابع گروهی کار می کند.

- Select pNo, Sum(QTY) from Test
  Group by pNo

sNo	pNo	QT Y
S1	P1	100
S2	P1	400
S1	P3	300
S3	P3	700
S1	P2	100
S2	P2	400

زبان پرس و جوی SQL

- Having مانند where است با این تفاوت که where روی سطرها تاثیر می گذارد اما having روی گروه ها اثر دارد و با Group by همراه می شود. Group by روی نتیجه having اعمال می شود.

- Select born, Sum(avg) from Number
  Group by born
  Having Count(*)>1

born	Sum(avg)
62	29

زبان پرس و جوی SQL

- نام، شهر و میانگین معدل هایی را نشان دهید که میانگین معدل آن شهر بیشتر از ۱۷ باشد.

```
Select city, Avg(avg) from Number
    Group by city
    Having Avg(avg)>=17
```

city	Avg(avg)
C1	17
C3	18.5

زبان پرس و جوی SQL

- Exists: عملگر صریحی برای اشتراک و تفاضل دو رابطه وجود ندارد ولی می توان آنها را به کمک Exists شبیه سازی کرد.

- درسهایی که حداقل یک دانشجو در آن درس افتاده باشد.

```
Select cName from Course
    where Exists( Select sName from Number
        where Course.cName=Number.course
        AND avg<10)
```

زبان پرس و جوی SQL

- Delete: برای حذف داده ها به کار می رودکه می تواند سطر یا سطرهایی را حذف کند.

```
Delete from Course
        where cName='Data'
```

- اگر بخواهیم کل اطلاعات جدول را حذف کنیم.

```
Delete from Course
```

- همه اطلاعات جدول را پاک می کند. (سریعتر کار می کند)

```
Truncate Table نام جدول
```

زبان پرس و جوی SQL

- تفاوت Delete، Drop و Truncate
  - Drop: جدول و محتویات آن را حذف می کند.
  - Delete: محتوای سطرها را پاک می کند اما عملیات را در فایل log می نویسد و می توان در صورت نیاز اطلاعات را recovery کرد.
  - Truncate: سطرها را به کلی پاک می کند اما عملیات را در فایل log نمی نویسد. سریع تر عمل می کند اما امکان recovery وجود ندارد.

زبان پرس و جوی SQL

- Update: برای تغییر داده ها در جدول استفاده می شود.

➢ Update نام جدول

set نام ستون=مقدار جدید , ...

where شرایط

- مثال:

➢ Update Course

set cNo=10

where cNo= 4

زبان پرس و جوی SQL

- در SQL سه نوع جدول وجود دارد:

1. جداول اصلی (base tables): با دستور create ایجاد می شود و می توان به آنها دسترسی داشت و یا تغییر داد.
2. جداول میانی (intermediate tables): که از دسترسی کاربران به دورند و توسط سیستم در ضمن انجام دستورهای پیچیده ایجاد می شوند و سپس از بین می روند. این جداول نه قابل دسترسی هستند و نه می توان آنها را تغییر داد.
3. جداول مجازی (views): برخلاف دو نوع دیگر، وجود فیزیکی و خارجی ندارند ولی می توان به آنها دسترسی داشت. هر view یک select است.

هدف اصلی از جداول مجازی ایجاد جداول خلاصه از اطلاعات موجود و محدود کردن دید کاربران است.

زبان پرس و جوی SQL

- جداول مجازی
- ✓ در view، دید باید ستونهایی از جداول موجود را در کنار هم چید و نمایش داد.
- ✓ با create view تعریف می شوند.
- ➢ Create view Myview(Names, Marks) as
  select sName, avg from Number
- ✓ می توان ستونهای جدول را با نام های جدید نامگذاری کرد.
- ✓ جداول مجازی تصویری از جداول حقیقی هستند و توسط سیستم به جداول اصلی مرتبط می شوند و وجود خارجی ندارند.

زبان پرس و جوی SQL

- جداول مجازی
- ✓ سیستم هر گونه استخراج را از جدول اصلی انجام می دهد اما کاربر می تواند مستقیما از جدول مجازی پرس و جو کند.
- ✓ با drop view از بین می روند.
- ➢ Drop view Myview
- ✓ مشکل اصلی جداول مجازی به روز درآوردن آنهاست. درصورتی می توان جداول مجازی را به روز درآورد که تغییرات مورد نظر روی جدول اصلی بدون اشکال و ابهام باشد.
- ✓ می توان روی جدول مجازی (دید) دید دیگری ایجاد کرد.
- ✓ برای انجام عملیات ذخیره‌سازی در دید از دستورات سه‌گانه INSERT، UPDATE و DELETE استفاده می‌شود.

زبان پرس و جوی SQL

❖ مزایای دید

- ✓ تامین‌کننده محیط انتزاعی برای کاربران سطح خارجی
- ✓ تامین‌کننده پویایی بالا در تعریف پایگاه توسط کاربر
- ✓ تسهیل‌کننده واسط کاربر برنامه‌ساز با پایگاه
- ✓ امکانی است برای کوتاه‌نویسی یا ماکرونویسی پرسشها
- ✓ تامین‌کننده اشتراک داده‌ای
- ✓ تامین‌کننده نوعی مکانیسم خودکار ایمنی داده‌ها
- ✓ تامین‌کننده استقلال داده‌ای فیزیکی و منطقی
- ✓ امکان تعریف شیئ با اندازه‌های مختلف

❖ معایب دید

- ✓ ایجاد اضافه کاری در سیستم برای انجام تبدیل خارجی/ادراکی
- ✓ عدم امکان انجام عملیات ذخیره‌سازی در بسیاری از دیدها و در نتیجه ایجاد محدودیت برای کاربر

زبان پرس و جوی SQL

❖ ایجاد شاخص

create [unique][clustered] index نام شاخص
on نام جدول (ستون [ترتیب],...)

ASC: صعودی (پیش فرض)
DESC: نزولی
unique:
باعث می شود شاخص تکراری نداشته باشیم.
clustered:
داده های جدول به صورت فیزیکی تغییر می کنند.
فقط یک index از این نوع می توانیم داشته باشیم.
DBMS در دستوراتی مثل select از آن استفاده می کند.

زبان پرس و جوی SQL

- ایجاد شاخص

create index x on course (cNo, cName DESC, unit)

- حذف شاخص

Drop index نام جدول .نام شاخص

Drop index course.x

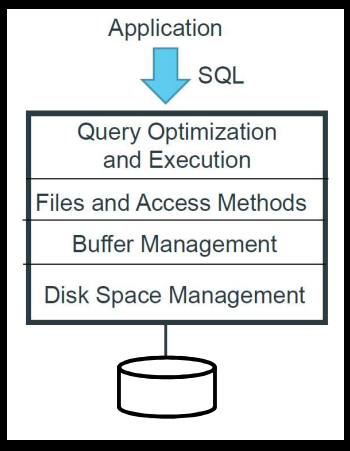
زبان پرس و جوی SQL

- GRANT و REVOKE
- سرپرست پایگاه داده می تواند به کاربران مجوزهایی مانند بازیابی، درج رکورد، حذف رکورد، تغییر رکورد، ایجاد جدول، حذف جدول، اضافه کردن فیلد، حذف فیلد، ایجاد شاخص، حذف شاخص، اجرای تابع مشخص شده و ... را به کاربران اعطا کند یا لغو کند. برای اعطا مجوز از دستور GRANT و برای لغو آن از دستور REVOKE استفاده می شود.
- اعطای مجوز به بازیابی به mina
- GRANT SELECT ON Student TO mina
- اعطای مجوزبازیابی به tina که می‌تواندمجوز خود را به کاربران دیگر بدهد
- GRANT SELECT ON Student TO tina WITH GRANT OPTION

زبان پرس و جوی SQL

- کاربر tina دستور زیر را اجرا می کند.
- GRANT SELECT ON Student TO nina;
- برای پس گرفتن امتیاز از tina دستور زیر اجرا می شود.
- REVOKE SELECT ON Student FROM tina;
- اگر در انتهای دستور بالا قید CASCADE اضافه شود آنگاه فقط mina می تواند از جدول Student انتخاب SELECT کند. زیرا لغو مجوز به صورت آبشاری خواهد بود. از آنجایی که tina به nina مجوز داده است، مجوز nina هم لغو می شود.
- با GRANT امتیاز انجام یک عمل روی جدول مبنا و مجازی (دید) را می توان اعطا کرد.

شاخص گذاری indexing



- **ساختار داخلی یک DBMS نمونه**
- **لایه اول: query به زبان SQL را دریافت و پس از parse کردن نحوه اجرا را مشخص می کند.**
- **لایه دوم: برای لایه بالاتر مفهوم فایل را ایجاد می کند.**
  - **فایل مجموعه ای از صفحات است.**
  - **هر صفحه حاوی یک یا چند رکورد است.**
- **لایه سوم: وظیفه انتقال صفحات از حافظه جانبی به حافظه اصلی را به عهده دارد.**
- **لایه چهارم: مدیریت فضای روی دیسک را به عهده دارد.**
  - **توابع allocate، deallocate ، read و write صفحات**

شاخص گذاری indexing

- ذخیره سازی داده روی حافظه ثانویه
- در پایگاه داده حجم زیادی از اطلاعات ذخیره می شود.
- داده ها باید ماندگار باشند.
- داده ها در حافظه اصلی جا نمی شود.
- انتقال صفحه از حافظه جانبی به حافظه اصلی و بالعکس پرهزینه ترین عملیات درDBMS است.
  - ✓ تعداد Page IO باید کمینه شود.
- داده ها را روی حافظه ثانویه ذخیره می شوند و هر زمان پروسس لازم باشد به حافظه اصلی منتقل می شوند. (واحد انتقال داده = صفحه، پارامترDBMS است)
  - ✓ دیسک: مهمترین ابزار ذخیره سازی با دسترسی تصادفی
  - ✓ نوار (Tape): دسترسی به صورت متوالی است (مناسب برای backup)

شاخص گذاری indexing

- در لایه فایل فرض می کنیم، هر رکورد یک rid دارد، که به صورت یکتا آدرس صفحه ای که رکورد را شامل می شود، دارد.
- اگر لایه فایل فضای جدید نیاز داشته باشد، به لایه Disk Space Management، درخواست یک صفحه جدید می دهد.
- اگر لایه فایل یک صفحه را نیاز نداشته باشد، آن را از طریق Disk Space Management، آزاد می کند.
- در لایه فایل، داده های ذخیره شده، به صورت مجموعه ای از رکوردها نمایش داده می شود.
- عملیات در سطح فایل: ایجاد فایل، حذف فایل، درج رکورد، حذف رکورد، اسکن(Scan): دسترسی به همه رکوردها
- لایه فایل، رکوردها را به صورت مجموعه ای از صفحات ذخیره می کند.
- لایه فایل، صفحه های مربوط به هر رابطه ای را می شناسد.
- لایه فایل، فضای خالی هر صفحه مربوط به یک رابطه را می شناسد.

شاخص گذاری indexing

- **ساختار Heap**
- ساده ترین ساختار فایل است.
- رکوردها به صورت تصادفی در صفحات مربوطه ذخیره می شوند.
- با داشتن rid می توان صفحه مربوطه به رکورد مورد نظر را پیدا کرد.

- **ساختار شاخص گذاری شده**
- نوعی ساختار که کارایی برخی از عملیات جستجو را بهبود می دهد.
- هر شاخص برای یک کلید جستجو (Search key) طراحی می شود.
- برای یک فایل واحد می توان چند نوع شاخص گذاری داشت که هر کدام کلید جستجوی خاص خودش را دارد.
- مثال: اطلاعات کارمندان
  - ✓ ذخیره سازی می تواند به صورت شاخص روی تاریخ تولد باشد.
  - ✓ یک شاخص اضافی روی حقوق ایجاد کرد.

شاخص گذاری indexing

- **دو مفهوم:**
  - ✓ رکورد های ذخیره شده در فایل ایندکس (Data Entry)
  - ✓ رکوردهای حاوی داده: مانند رکوردهای کارمندان در مثال قبل (Data Record)

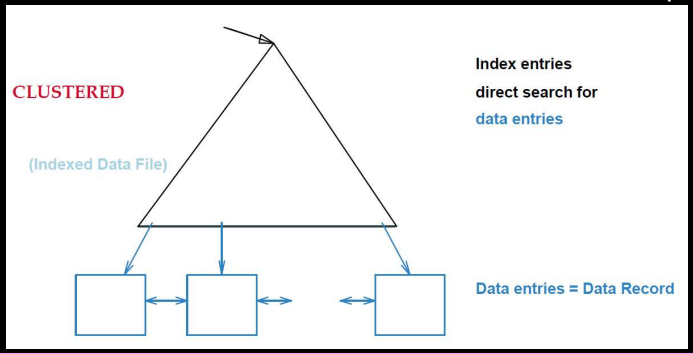
- با داشتن data entry می توان به data record دسترسی پیدا کرد.
- شاخص به گونه ای ساخته می شود که یافتن data entry با یک شرط مشخص در آن کارا باشد.
- پس از یافتن data entry، data record ها را پیدا می کنیم.

شاخص گذاری indexing

- **سه روش برای نگهداری شاخص وجود دارد:**
  - ✓ روش خوشه بندی شده data entry = data record (با کلید k)
  - ✓ data entry=(k,rid) (rid شامل id رکورد data record است با کلید k)
  - ✓ data entry=(k,rid-list) (rid-list لیست id های data records است با کلید k)
- در روش اول data record بر اساس کلید جستجو شاخص در فایل قرار می گیرند.
- در روش دوم و سوم data entry ها به data record ها آدرس می دهند.
- در روش اول: یک فایل داریم که data record ها به صورت شاخص گذاری شده نگهداری می شوند.
- در روش دوم و سوم: فایل شاخص از فایل data record ها مجزا و مستقل است.

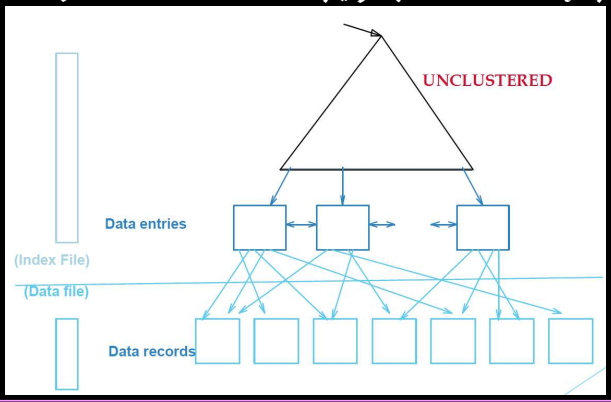
شاخص گذاری indexing

- **نمایش شاخص خوشه گذاری شده**
- **ترتیب data entry ها با ترتیب data record ها یکسان یا نزدیک به هم است.**



شاخص گذاری indexing

- نمایش شاخص خوشه گذاری نشده
- ترتیب data entry ها با ترتیب data record ها متفاوت است.



شاخص گذاری indexing

- برای یک فایل با چند شاخص فقط یکی از شاخص ها می تواند از نوع اول باشد و بقیه شاخص ها از نوع دوم و سوم هستند.
- اگر یک فایل به گونه ای باشد که ترتیب data entry ها شبیه ترتیبdata recordها باشد شاخص clustered نامیده می شود.
- شاخص های نوع اول clustered هستند و شاخص های نوع دوم و سوم clustered نیستند.

شاخص گذاری indexing

- **دسته بندی query ها با توجه به محدوده جستجو**
- Range Search Query: جستجوی محدوده
- نام دانشجویان با معدل بین ۱۲ تا ۱۷ را نمایش بدهد.
  - Select SName from Number where avg between 12 and 17
- Equity Search Query: جستجوی برابری
- نام دانشجویان با معدل ۱۵را نمایش بدهد.
  - Select SName from Number where avg =15

شاخص گذاری indexing

- **شاخص خوشه بندی شده (Clustered Index)**
- با استفاده از شاخص خوشه بندی شده، می توان کارایی Range Search Query را به صورت چشم گیری افزایش داد.
- رکوردهایی که در جواب صدق می کنند در مجاورت هم ذخیره شده اند.
- با انتقال تعداد محدودی از صفحات به حافظه، به جواب می رسیم.
- برای یک فایل حداکثر یک شاخص خوشه گذاری شده می توان داشت.
- به تعداد دلخواه می توان شاخص خوشه گذاری نشده داشت.
- اگر شاخص خوشه بندی نباشد؟
  - ممکن است هر رکورد جواب، در یک صفحه مجزا قرار بگیرد.
  - در بدترین حالت به تعداد رکوردهای جواب باید صفحه به حافظه منتقل شود.
    - ✓ IO cost بسیار بالا

شاخص گذاری indexing

- **شاخص primary و Secondary**
- دو نوع تعریف وجود دارد:
- تعریف اول: شاخص های تعریف شده روی Primary Key شاخص های Primary گفته می شوند. در غیر این صورت شاخص Secondary است.
- تعریف دوم: شاخص های نوع ۱ را Primary index و شاخص های نوع ۲و ۳ را Secondary index است.

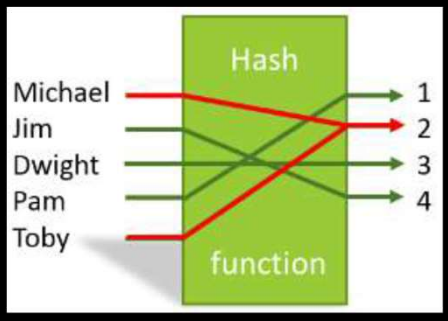
- **شاخص Unique**
- اگر در یک شاخص گذاری search key) duplicate یکسان) نداشته باشیم به شاخص مربوطه Unique Index گویند.

شاخص گذاری indexing

- **شاخص گذاری داده ها**
- دو روش کلی برای شاخص گذاری داده استفاده می شود:
  - HASHING ✓
  - TREE ✓
- هدف: با داشتن مقدار کلید جستجو چگونه شماره صفحه رکوردهای جواب را پیدا کنیم؟
- رکوردهای دانشجویانی را بیابید که نام آنها علی باشد.

شاخص گذاری indexing

- **شاخص گذاری مبتنی بر Hash**
- با استفاده از تابع Hash می توانیم رکورد ها را به گونه ای قرار دهیم که با داشتن کلید جستجو، با سرعت بالا به صفحه مربوط به رکورد دسترسی پیدا کنیم.



شاخص گذاری indexing

- **شاخص گذاری مبتنی بر Hash**
- فرض کنید مجموعه رکورد ها به شرح زیر باشد.
- رکوردها را به صورت خوشه بندی شده، با استفاده از تابع Hash زیر در فایل مرتب کنید.
- باقی مانده سال تولد به عدد ۳

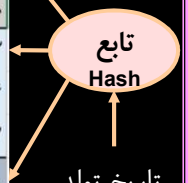
ورودی تابع: تاریخ تولد
خروجی تابع: یک عدد صحیح بین ۰ تا ۳

شماره	نام	نام خانوادگی	تاریخ تولد	حقوق
۱	تقی	تقوی	۱۳۵۵	۲۰۰۰
۲	اکبر	حسینی	۱۳۶۵	۱۵۰۰
۳	علی	اکبری	۱۳۴۵	۱۲۰۰
۴	حسن	حسنی	۱۳۳۰	۱۰۰۰
۵	اکبر	اکبری	۱۳۶۲	۲۵۰۰
۶	حسن	اکبری	۱۳۷۰	۸۰۰
۷	علی	حسنی	۱۳۴۵	۷۵۰
۸	حسن	تقوی	۱۳۴۴	۹۲۰

شاخص گذاری indexing

- **شاخص گذاری مبتنی بر Hash**
- آیا نوع شاخص خوشه بندی شده است؟.

شماره	نام	نام خانوادگی	تاریخ تولد	حقوق	باقی مانده تاریخ تولد به ۳
۲	اکبر	حسینی	۱۳۶۵	۱۵۰۰	۰
۵	اکبر	اکبری	۱۳۶۲	۲۵۰۰	۰
۸	حسن	تقوی	۱۳۴۴	۹۲۰	۰
۳	علی	اکبری	۱۳۴۵	۱۲۰۰	۱
۴	حسن	حسنی	۱۳۳۰	۱۰۰۰	۱
۷	علی	حسنی	۱۳۴۵	۷۵۰	۱
۱	تقی	تقوی	۱۳۵۵	۲۰۰۰	۲
۶	حسن	اکبری	۱۳۷۰	۸۰۰	۲



شاخص گذاری indexing

- **شاخص گذاری مبتنی بر Hash**
- رکوردهای فایل قبل را بر اساس حقوق هم شاخص گذاری کنید.
- آیا حقوق مضربی از ۲۰۰ است؟
- ورودی تابع: حقوق
- خروجی تابع: بلی، خیر
- تعداد Bucketها:۲

- آیا می توان شاخص خوشه بندی شده ایجادکرد؟
- دو کلید جستجو داریم:
  - ✓ تاریخ تولد (خوشه بندی شده)
  - ✓ حقوق

شاخص گذاری indexing

- شاخص گذاری مبتنی بر Hash

شاخص گذاری indexing

- شاخص گذاری مبتنی بر Hash
- تعداد فیلد های کلید جستجو می تواند بیش از یک باشد.
- می توان همزمان روی دو یا چند فیلد شاخص گذاری انجام داد.
- استفاده عملی:
  - ✓ به صورت عملی استفاده از شاخص گذاری Hashing محدود است.
  - ✓ تنها برای Equity Search Query مناسب است و برای Range Search Query مناسب نیست.

شاخص گذاری indexing

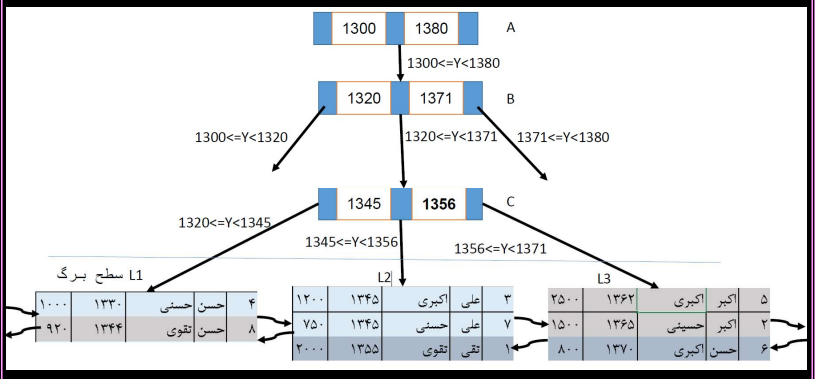
- **شاخص گذاری مبتنی بر Hash**
- مجموعه رکورد ها به چند Bucket تقسیم می شوند.
- هرBucket شامل یک صفحه اصلی و در صورت نیاز، چند صفحه اضافی است.
- نگاشت کلید جستجو به Bucket با استفاده از تابع Hash انجام می شود.
- اگر شماره Bucket را بدانیم، یافتن صفحه مورد نظر با تعداد محدودی IO امکان پذیر است.
- در هنگام Insert هر رکورد در Bucket مربوط به خود درج می شود.
- در هنگام Search مقدار کلید جستجو را با استفاده از تابع Hash به شماره Bucket تبدیل می کند.

شاخص گذاری indexing

- **شاخص گذاری مبتنی بر Tree**
- ساختار درختی یک راهکار جایگزین برای ساختن index است.
- در این روش Data Entry ها با توجه به کلید جستجو، به صورت مرتب (sorted) ذخیره می شوند.
- از ساختار درختی برای هدایت جستجو استفاده می شود.

شاخص گذاری indexing

- **شاخص گذاری مبتنی بر Tree**
- در مثال اطلاعات کارمندان، نمونه شاخص گذاری درخت



شاخص گذاری indexing

- **شاخص گذاری مبتنی بر Tree**
- در لایه انتهایی که لایه برگ است data entry ها قرار می گیرند.
- تعداد disk I/O هایی که در یک جستجو اتفاق می افتد برابر است با
  - عمق (فاصله ریشه تا برگ) برگ که شامل اولین پاسخ است.
  - به علاوه تعداد برگ هایی که پاسخ ها را دارند.
- اگر ساختار شاخص درخت B^+ باشد، تمام مسیرها از ریشه تا یک برگ یکسان است. به دلیل اینکه این درخت balanced است.
- هزینه جستجو در یک درخت B^+ کمتر از هزینه جستجوی باینری روی فایل مرتب شده از رکورد ها است. به دلیل اینکه هر گره داخلی (غیربرگ) می تواند تعداد زیادی اشاره گر داشته باشد که ارتفاع درخت را با استفاده از آن کاهش می دهیم.
- ارتفاع درخت B^+ در عمل حداکثر ۳ یا ۴ است.

شاخص گذاری indexing

- **شاخص گذاری مبتنی بر Tree**
- میانگین تعداد زیر نودهای یک نود غیر برگ fan out نامیده می شود.
- اگر هر نود غیربرگ n فرزند داشته باشد، درخت با ارتفاع h تعداد n^h صفحه برگ دارد.
- اگر درخت ۱۰۰ میلیون صفحه برگ داشته باشد، درخت با ارتفاع چهار، صد میلیون برگ داشته باشد. با ۴ I/O می توانیم صفحه مورد نظر را در درخت B^+ پیدا کنیم.
- درخت باینری برای همین فایل نیاز به $\log_2 100000000$ بالای ۲۵، I/O نیاز دارد.

شاخص گذاری indexing

- **مقایسه ساختارهای فایل**
- مقایسه کارایی عملیات مختلف با استفاده از ساختارهای متنوع
- فرض شده است شاخص ها بر اساس کلیدهای (حقوق و سال تولد) ایجاد شدند.
- ساختارهای در نظر گرفته شده:
  - ✓ Heap
  - ✓ فایل مرتب شده (sorted)
  - ✓ شاخص خوشه بندی شده درختی B^+
  - ✓ ساختار Heap با شاخص غیرخوشه ای درختی B^+
  - ✓ ساختار Heap با شاخص غیرخوشه ای Hash
- تاکید این مبحث بر اهمیت انتخاب ساختار فایل است.

شاخص گذاری indexing

- **مقایسه ساختارهای فایل**
- عملیاتی که روی فایل ها انجام می شود به شرح زیر است:
- Scan: در این فعالیت هدف، انتقال همه رکوردها (صفحات فایل) از حافظه جانبی به buffer است. در این عمل overhead مربوط به CPU برای هر رکورد وجود دارد تا رکورد در بافر قرار گیرد.
- Search with Equality Selection (Equity search): جستجو با شرط برابری با کلید جستجو، صفحات مربوطه باید بازیابی شوند.
  - ✓ مثال: همه رکوردهایی که کارمند متولد ۱۳۴۵ و درآمد ۷۵۰ تومان باشد.
- Search with Range Selection (Range Query): جستجوی محدوده روی کلید جستجو
  - ✓ مثال: همه رکوردهایی که شامل کارمندان متولد سال ۱۳۶۰ به بعد باشد.

شاخص گذاری indexing

- **مقایسه ساختارهای فایل**
- عملیاتی که روی فایل ها انجام می شود به شرح زیر است:
- Insert a Record: صفحه فایلی که رکورد باید در آن قرار گیرد مشخص می شود. از حافظه آورده شده (واکشی)، اصلاح می شود، سپس صفحه اصلاح شده در حافظه نوشته شود.
- Delete a Record: حذف یک رکورد با استفاده از rid آن رکورد. صفحه ای که حاوی رکورد است شناسایی می شود، از روی دیسک آورده، اصلاح کرده، سپس روی دیسک نوشته می شود.

شاخص گذاری indexing

- مقایسه ساختارهای فایل

شرح پارامتر	نام پارامتر	مقدار متداول
تعداد صفحات برای ذخیره سازی رکورد ها (بدون اتلاف فضا)	B	Size/8k
تعداد رکورد در هر صفحه	R	-
متوسط زمان خواندن/ نوشتن یک صفحه	D	15 MS
متوسط زمان لازم برای پردازش یک رکورد	C	100 NS
متوسط زمان لازم برای اجرای تابع Hash روی یک رکورد	H	100 NS
تعداد فرزندان نودهای غیر برگ دردرخت B^+	F	100

شاخص گذاری indexing

- بررسی ساختار Heap
- Scan:
  - ✓ B(D+RC)
  - ✓ B صفحه باید به حافظه منتقل شود: BD
  - ✓ و هر صفحه شامل R رکورد است که برای پروسس هررکورد C زمان لازم است: BRC
- Equity Search:
  - ✓ اگر جستجو روی کلید کاندید انجام شود: 0.5B(D+RC)
  - ✓ اگر جستجو جواب نداشته باشد و یا جستجو روی یک صفت غیرکلید باشد: B(D+RC)
- Range Query:
  - ✓ همه فایل را برای یافتن پاسخ باید جستجو کنیم به دلیل اینکه رکوردهای پاسخ ممکن است هرجایی در فایل قرار داشته باشند: B(D+RC)

شاخص گذاری indexing

- **بررسی ساختار Heap**
- Insert:
  - ✓ کافی است رکورد به انتهای فایل اضافه شود. صفحه آخر را به حافظه منتقل می کنیم، رکورد را می نویسیم؛ صفحه را به دیسک منتقل می کنیم: 2D+C
- Delete:
  - ✓ هزینه جستجو رکورد و انتقال صفحه به علاوه C+D
  - ✓ اگر rid را داشته باشیم: صفحه به راحتی توسط id رکورد قابل بازیابی و مستقیما قابل خواندن است. هزینه جستجو رکورد و انتقال صفحه به حافظه: D
  - ✓ اگر rid را نداشته باشیم : هزینه باید به روش های موجود محاسبه شود. هزینه حذف بستگی به تعداد رکوردها دارد.

File Type	Scan	Equity Search	Range Search	Insert	Delete
Heap	B(D+RC)	0.5B(D+RC)	B(D+RC)	2D+C	Search +C+D

شاخص گذاری indexing

- **بررسی ساختار Sorted**
- Scan:
  - ✓ همه صفحه ها باید بررسی شوند: B(D+RC)
- Equity Search:
  - ✓ جستجو روی کلید ترکیبی(حداقل اولین کلید) انجام شود. $\log B$ جستجو برای یافتن صفحه هدف، $\log R$ جستجو برای یافتن رکورد: $D \log B + C \log R$
  - ✓ اگر بیش از یک رکورد جواب داشته باشیم، به دلیل مرتب بودن مجاور هم هستند و هزینه بازیابی همه رکوردها برابر با یافتن اولین رکورد به علاوه هزینه خواندن همه رکوردها به ترتیب است. معمولا، همه رکوردها در یک صفحه قرار می گیرند.
- Range Query:
  - ✓ اگر جستجو روی کلید ترکیب انجام شود، هزینه شامل یافتن اولین رکود جواب و در صورت لزوم واکشی کردن صفحات بعدی است. هزینه به رکوردهای جواب بستگی دارد $D \log B + C \log R + RC \ \#matched\ records$.

شاخص گذاری indexing

- **بررسی ساختار Sorted**
- Insert:
  - ✓ محل درست درج رکورد در فایل مشخص می شود. رکوردهای بعد از رکورد جدید یک رکورد جابجا شوند: 2(0.5B(D+RC))
- Delete:
  - ✓ محل درست حذف رکورد در فایل مشخص می شود. رکوردهای بعد از رکورد جدید برای از بین بردن فضای آزاد جابجا شوند: هزینه با insert برابر است. هزینه جستجو به علاوه B(D+RC).

File Type	Scan	Equity Search	Range Search	Insert	Delete
Heap	B(D+RC)	0.5B(D+RC)	B(D+RC)	2D+C	Search +C+D
Sorted	B(D+RC)	$D \log B + C \log R$	$D \log B + C \log R$ + RC #matched records	Search+ B(D+RC)	Search+ B(D+RC)

شاخص گذاری indexing

- **بررسی ساختار Clustered B^+ tree**
- مطالعات تجربی نشان داده است که حدود ۶۷ درصد صفحات اشغال می شوند و به طور متوسط یک سوم حجم صفحات خالی می ماند. بنابراین، اگر داده به صورت فشره B باشد، تعداد صفحات داده های فیزیکی حدود ۱.۵B است.
- Scan:
  - ✓ به دلیل اینکه تمام صفحات باید بررسی شوند: 1.5B(D+RC)
- Equity Search:
  - ✓ جستجو روی کلید ترکیبی(حداقل اولین کلید) انجام شود.
  - ✓ جستجو برای یافتن صفحه هدف، $\log_2 R$ جستجو برای یافتن $\log_F 1.5B$ رکورد: $D \log_F 1.5B + C \log_2 R$
  - ✓ اگر بیش از یک رکورد جواب داشته باشیم، به دلیل مرتب بودن مجاور هم هستند و هزینه بازیابی همه رکوردها برابر با یافتن اولین رکورد به علاوه هزینه خواندن همه رکوردها به ترتیب است.

شاخص گذاری indexing

- بررسی ساختار Clustered B^+ tree
- Range Query:
  - ✓ اگر جستجو روی کلید ترکیب انجام شود، هزینه شامل یافتن اولین رکود جواب و در صورت لزوم واکشی کردن صفحات بعدی است. (با استفاده از pointer های روی برگ ها: لینک های next و previous در سطح برگ). هزینه به رکوردهای جواب بستگی دارد: $D \log_F 1.5B + C \log_2 R + RC \#matched\ records$
- Insert:
  - ✓ صفحه برگ جایگاه درج رکورد مشخص شود. در بیشتر موارد فضای خالی براب درج شدن یک رکورد جدید وجود دارد، در غیر این صورت ساختار B Tree باید روز شود که به ندرت اتفاق می افتد. هزینه برابر با هزینه جستجو و یک نوشتن است: $D \log_F 1.5B + C \log_2 R + D$

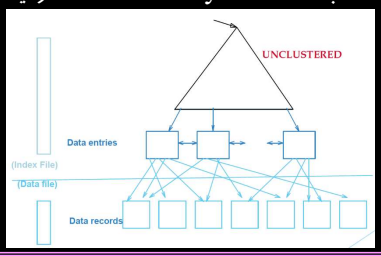
شاخص گذاری indexing

- بررسی ساختار Clustered B^+ tree
- Delete:
  - ✓ جستجوی رکورد، حذف رکورد از صفحه و نوشتن صفحه اصلاح شده. مشابه Insert است: $D \log_F 1.5B + C \log_2 R + D$

File Type	Scan	Equity Search	Range Search	Insert	Delete
Heap	B(D+RC)	0.5B(D+RC)	B(D+RC)	2D+C	Search+C+D
Sorted	B(D+RC)	D log B + C log R	D log B + C log R + RC #matched records	Search+ B(D+RC)	Search+ B(D+RC)
Clustered Tree	1.5B(D+ RC)	D $\log_F$ 1.5B+C $\log_2$ R	D $\log_F$ 1.5B+C $\log_2$ R + RC#matched records	D $\log_F$ 1.5B+ C $\log_2$ R + D	D $\log_F$ 1.5B+ C $\log_2$ R + D

شاخص گذاری indexing

- **بررسی ساختار Unclustered B$^+$ tree**
- تعداد برگهای صفحه بستگی به data entry ها دارد.
- فرض می شود که هر data entry در شاخص یک دهم اندازه یک data record است: data entry= 0.1 data record
- تعداد صفحه های برگ در شاخص 0.15B = 0.1(1.5B) است.
- اگر ۶۷ درصد فضای شاخص اشغال شده باشد. تعداد data entry ها در یک صفحه 6.7R = 10(0.67R).



شاخص گذاری indexing

- **بررسی ساختار Unclustered B$^+$ tree**
- Scan:

✓ اگر بخواهیم به صورت مرتب شده به رکورد ها دسترسی پیدا کنیم، بهتر است که شاخص را فراموش کنیم و فایل را مستقیم بررسی کنیم. تمام data entry ها با هزینه 0.15B(D+0.67RC) خوانده می شوند. برای هر data entry باید data record واکشی شود که به ازای هر رکورد یک I/O داریم. شاخص خوشه بندی نشده است و هر data entry در صفحه برگ ممکن است در صفحه متفاوتی در فایل اشاره کند. هزینه این مرحله $RB\ (C + D)$ که زیاد است.

✓ روش ساده این است که فایل را در یک الگوریتم دو مرحله ای که در هر مرحله یک خواندن و یک نوشتن دارد مرتب کنیم. بنابراین هزینه مرتب سازی فایل با B صفحه، 4B است که بسیار کمتر است.

شاخص گذاری indexing

- بررسی ساختار Unclustered B^+ tree
- Equity Search:
  - ✓ یافتن اولین صفحه شامل data entry یا data entry ها: $D \log_F 0.15B$
  - ✓ یافتن اولین data record با استفاده از درخت دودویی: $C \log_2 6.7R$
  - ✓ هزینه کل : $D \log_F 0.15B + C \log_2 6.7R + D$
  - ✓ اگر بیش از یک رکورد جواب داشته باشیم، رکوردها مجاور هم نیستند و هزینه بازیابی همه رکوردها برابر با هزینه یافتن اولین رکورد به علاوه یک I/O به ازای هر رکوردی که بازیابی می شود. هزینه در این حالت هزینه به تعداد رکوردها بستگی دارد.

شاخص گذاری indexing

- بررسی ساختار Unclustered B^+ tree
- Range Query:
  - ✓ هزینه شامل یافتن اولین رکورد جواب و در صورت لزوم واکشی کردن صفحات بعدی است. (با استفاده از pointer های روی برگ ها: لینک های next و previous در سطح برگ). برای هر data entry یک I/O لازم است تا data record به دست آید. با افزایش رنج انتخابی هزینه گران می شود. اگر ده درصد رکوردها در شرایط انتخابی (پاسخ) قرار دارند بهتر است از شاخص استفاده نشود.
  - ✓ یافتن اولین صفحه شامل data entry یا data entry ها:
  - ✓ یافتن اولین data record با استفاده از درخت دودویی:
  - ✓ به تعداد صفحه های match شده، در index، I/O نیاز داریم.
  - ✓ به عنوان یک قاعده کلی اگر ۱۰ درصد از data record شرایط انتخاب را برآورده کند، بهتر است همه data record ها را بازیابی کرده، آنها را مرتب کرده و سپس مواردی را که انتخاب را برآورده می کند، نگه داریم.

شاخص گذاری indexing

- بررسی ساختار Unclustered B^+ tree
- Insert:
  - ذخیره شدن داده در فایل heap : 2D+C
  - جستجو و به روز رسانی Index (جستجو درج کردن data entry در شاخص): $D\log_F 0.15B + C\log_2 6.7R + D$
  - جستجو : $D\log_F 0.15B + C\log_2 6.7R$ درج: D
- Delete:
  - جستجو data record در فایل و data entry در شاخص: $D\log_F 0.15B + C\log_2 6.7R + D$
  - نوشتن صفحات اصلاح شده در فایل و در شاخص: 2D

شاخص گذاری indexing

- بررسی ساختار Unclustered B^+ tree
- جدول هزینه ها ی ساده شده

File Type	Scan	Equity Search	Range Search	Insert	Delete
Heap	BD	0.5BD	BD	2D	Search + D
Sorted	BD	D log B	D log B + # matching pages	Search+ BD	Search+ BD
Clustered Tree	1.5BD	$D\log_F 1.5B$	$D\log_F 1.5B$+ # matching pages	Search+D	Search + D
Unclustered Tree	BD(R+0.15)	$D(1+\log_F 0.15B)$	$D(\log_F 0.15B$ + # matching records)	$D(3+\log_F 0.15B)$	Search + 2D

شاخص گذاری indexing

- بررسی ساختار Heap file با Unclustered Hash Index

- فرض می شود که هر data entry در شاخص یک دهم اندازه یک data record است: data entry= 0.1 data record
- در فایل های static hash، ۸۰ درصد صفحه ها اشغال شده اند. برای اینکه از overflow ها در زمان افزایش و گسترش فایل جلوگیری شود.
- تعداد صفحات لازم برای data entry ها: 1.25(0.1B)=0.125B
- تعداد data entry ها در یک صفحه 8R = (0.8R)10.

شاخص گذاری indexing

- بررسی ساختار Heap file با Unclustered Hash Index

- Scan:
  - ✓ هزینه بازیابی data entry ها: 0.125B(D+8RC)
  - ✓ برای هر data entry یک هزینه اضافه (I/O) برای واکشی data record منطبق وجود دارد. هزینه این مرحله: BR(D+C)
  - ✓ هزینه زیاد است و از hash برای scan استفاده نمی شود.

شاخص گذاری indexing

- **بررسی ساختار Heap file با Unclustered Hash Index**
- Equity Search:
  - ✓ یافتن صفحه شامل data entry ها (یافتن Bucket): H
  - ✓ با فرض اینکه هر bucket یک صفحه دارد هزینه بازیابی صفحه data entry: D
  - ✓ اگر فرض شود بعد از بررسی نصف رکوردها در صفحه data entry پیدا می شود هزینه اسکن صفحه : $0.5(8RC) = 4RC$
  - ✓ هزینه واکشی data record: D
  - ✓ هزینه کل: H+2D+4RC
  - ✓ این هزینه کمتر از شاخص درختی است.
  - ✓ اگر بیش از یک رکورد جواب داشته باشیم، رکوردها مجاور هم نیستند و هزینه بازیابی همه رکوردها برابر با هزینه یافتن اولین رکورد به علاوه یک I/O به ازای هر رکوردی که بازیابی می شود. هزینه در این حالت هزینه به تعداد رکوردها بستگی دارد.

شاخص گذاری indexing

- **بررسی ساختار Heap file با Unclustered Hash Index**
- Range Query:
  - ✓ استفاده از hash index کمکی نمی کند.
  - ✓ کل فایل باید جشتجو شود. هزینه: BR(D+C)
- Insert:
  - ✓ درج داده در فایل heap: 2D+C
  - ✓ جستجو و به روز رسانی Index (جستجو صفحه و درج کردن data entry در شاخص): H+2D+C
- Delete:
  - ✓ جستجو data record در فایل و data entry در شاخص: H+2D+4RC
  - ✓ نوشتن صفحات اصلاح شده در فایل و در شاخص: 2D

شاخص گذاری indexing

- بررسی ساختار Heap file با Unclustered Hash Index
- جدول هزینه ها ی ساده شده

File Type	Scan	Equity Search	Range Search	Insert	Delete
Heap	BD	0.5BD	BD	2D	Search + D
Sorted	BD	D log B	D log B + # matching pages	Search+ BD	Search+ BD
Clustered Tree	1.5BD	$D \log_F 1.5B$	$D \log_F 1.5B$+ # matching pages	Search+D	Search + D
Unclustered Tree Index	BD(R+0.15)	$D(1 + \log_F 0.15B)$	$D(\log_F 0.15B +$ # matching records)	$D(3 + \log_F 0.15B)$	Search + 2D
Unclustered Hash Index	BD(R+0.125)	2D	BD	4D	Search + 2D

شاخص گذاری indexing

- مقایسه ساختارهای فایل
- ساختار Heap:
  - ✓ کارایی ذخیره سازی خوب، اسکن و درج رکورد سریع، جستجو و حذف کند
- ساختار مرتب شده (sorted):
  - ✓ کارایی ذخیره سازی خوب، درج و حذف رکوردها کند، جستجو سریعتر از فایلهای heap
- ساختار Clustered:
  - ✓ درج و حذف کارآمد، جستجوها سریعتر از فایل های مرتب شده، البته یک فایل مرتب شده می تواند سریعتر باشد هنگامی که تعداد زیادی رکورد به صورت متوالی بازیابی می شوند
- ساختار Unclustered (با شاخص Hash و tree):
  - ✓ جستجو و درج و حذف سریع، اسکن و جستجو با تعداد زیادی match کند، hash در Equity Search کمی سریعتر اما عدم پشتیبانی از Range Query

شاخص گذاری indexing

- **انتخاب Index و تنظیم کارایی**
- انتخاب نوع index می تواند در کارایی سیستم پایگاه داده بسیار موثر باشد.
- انتخاب index با توجه به workload انجام می شود.
- با استفاده از Index کارایی بازیابی اطلاعات بیشتر اما هزینه درج بیشتر است.
- Hash index فقط برای Equity Search بهینه است.
- Tree index، پشتیبانی هر دو جستجو (Equity Search و Range Query)
- Hash و Tree : پشتیبانی از درج، حذف و به روز رسانی به صورت کارا

شاخص گذاری indexing

- **مفهوم workload**
- عملیات و پرس و جوهای مرسوم و با رخداد زیاد در سیستم چیست؟
- مثال: سیستم آموزشی دانشگاه:
  - ✓ عملیات: درج استاد(۲) ، درج دانشجو(۱۰۰) ، درج درس (۱)
  - ✓ جستجو: دانشجو با شماره دانشجویی (۱۰۰)، دانشجو با نام و نام خانوادگی (۸۰)، دانشجو با کد ملی (۱۰)

شاخص گذاری indexing

- **انتخاب های طراحی**
- برای کدام رابطه های پایگاه داده شاخص تعریف کنیم.
- چه نوع Index استفاده شود:
  - ✓ Hash یا tree
  - ✓ Clustered یا Unclustered
- روش طراحی indexes
  - ✓ Query های مهم سیستم را شناسایی کنید.
  - ✓ نحوه اجرای Query را در نظر بگیرید.
  - ✓ کارایی را با اضافه کردن index جدید بررسی کنید.
- حداکثر یک index به صورت clustered داریم، شاخصی را انتخاب کنید که بیشترین کارایی را دارد.

شاخص گذاری indexing

- **انتخاب های طراحی**
- اضافه کردن index جدید کارایی insert و update را کاهش می دهد.
- Attribute هایی که به دفعات در شرط where قرار می گیرند، کاندیدهای خوبی برای index شدن هستند.
  - ✓ شرط برابری: Hash index
  - ✓ شرط محدوده: Tree index
  - ✓ استفاده از clustering کارایی جستجو محدوده را افزایش می دهد.
- انواع Index
  - ✓ single Attribute: رکوردها برحسب یک صفت شاخص گذاری شده اند.
  - ✓ multi Attribute: اگر در شرط where چند صفت با هم رخ دهند می توان شاخص را به صورت multi attribute ایجاد کرد.
  - ✓ indexهایی را انتخاب کنید که کارایی تعداد بیشتری از queryها را افزایش دهد.

شاخص گذاری indexing

- مقایسه single Attribute و multi Attribute

شماره دانشجویی	نام	نام خانوادگی	دانشکده
۱	علی	حسنی	ریاضی
۲	محمود	حسنی	ریاضی
۳	حسن	حسنی	ریاضی
۴	حسین	حسنی	ریاضی
۵	علی	اکبری	فیزیک
۶	محمود	اکبری	فیزیک
۷	حسن	اکبری	فیزیک
۸	حسین	اکبری	کامپیوتر
۹	علی	توکلی	کامپیوتر
۱۰	محمود	توکلی	کامپیوتر
۱۱	حسن	توکلی	کامپیوتر
۱۲	حسین	توکلی	کامپیوتر

- مثال: اگر نام و نام خانوادگی دانشجو همزمان در شرط where آورده شود، در این صورت پیشنهاد می شود multi-attribute index ساخته شود.

Select * from STT where stname ='ali' and stfamily = ' hassani'

- Data entry ها در این صورت ابتدا بر روی نام خانوادگی و سپس بر روی نام شاخص گذاری شده

شاخص گذاری indexing

- مقایسه single Attribute و multi Attribute

شماره دانشجویی	نام	نام خانوادگی	دانشکده
۱	علی	حسنی	ریاضی
۲	محمود	حسنی	ریاضی
۳	حسن	حسنی	ریاضی
۴	حسین	حسنی	ریاضی
۵	علی	اکبری	فیزیک
۶	محمود	اکبری	فیزیک
۷	حسن	اکبری	فیزیک
۸	حسین	اکبری	کامپیوتر
۹	علی	توکلی	کامپیوتر
۱۰	محمود	توکلی	کامپیوتر
۱۱	حسن	توکلی	کامپیوتر
۱۲	حسین	توکلی	کامپیوتر

single index

شماره دانشجویی	نام	نام خانوادگی	دانشکده
۵	علی	اکبری	فیزیک
۶	محمود	اکبری	فیزیک
۷	حسن	اکبری	فیزیک
۸	حسین	اکبری	کامپیوتر
۹	علی	توکلی	کامپیوتر
۱۰	محمود	توکلی	کامپیوتر
۱۱	حسن	توکلی	کامپیوتر
۱۲	حسین	توکلی	کامپیوتر
۱	علی	حسنی	ریاضی
۲	محمود	حسنی	ریاضی
۳	حسن	حسنی	ریاضی
۴	حسین	حسنی	ریاضی

شاخص گذاری indexing

- مقایسه single Attribute و multi Attribute

شماره دانشجویی	نام	نام خانوادگی	دانشکده
۱	علی	حسنی	ریاضی
۲	محمود	حسنی	ریاضی
۳	حسن	حسنی	ریاضی
۴	حسین	حسنی	ریاضی
۵	علی	اکبری	فیزیک
۶	محمود	اکبری	فیزیک
۷	حسن	اکبری	فیزیک
۸	حسین	اکبری	کامپیوتر
۹	علی	توکلی	کامپیوتر
۱۰	محمود	توکلی	کامپیوتر
۱۱	حسن	توکلی	کامپیوتر
۱۲	حسین	توکلی	کامپیوتر

multi index

شماره دانشجویی	نام	نام خانوادگی	دانشکده
۷	حسن	اکبری	فیزیک
۸	حسین	اکبری	کامپیوتر
۵	علی	اکبری	فیزیک
۶	محمود	اکبری	فیزیک
۱۱	حسن	توکلی	کامپیوتر
۱۲	حسین	توکلی	کامپیوتر
۹	علی	توکلی	کامپیوتر
۱۰	محمود	توکلی	کامپیوتر
۳	حسن	حسنی	ریاضی
۴	حسین	حسنی	ریاضی
۱	علی	حسنی	ریاضی
۲	محمود	حسنی	ریاضی

شاخص گذاری indexing

- مقایسه single Attribute و multi Attribute
- در شاخص گذاری multi Attribute تقدم صفت ها در where مهم است.
- در query زیر شاخص گذاری مفید است:

```
Select * from STT where stfamily = 'اکبری' and stname ='علی'
```

- در query زیر شاخص گذاری مفید نیست:

```
Select * from STT where stname ='علی'
```

شاخص گذاری indexing

- **کلید جستجوی مرکب**
- اگر جستجو به صورت Sal=400 && DOB=1330 باشد:
  - ✓ کارایی index روی (sal,DOB) بیشتر از کارایی روی sal و DOB است.
- اگر جستجو به صورت 300<Sal<400 && 1330<DOB<1340 باشد:
  - ✓ ایندکس خوشه بندی شده روی (sal,DOB) یا (DOB,sal) بهترین انتخاب است.
- اگر جستجو به صورت 300<Sal<400 && DOB=1330 باشد:
  - ✓ ایندکس خوشه بندی شده روی (DOB,sal) بهترین انتخاب است.

شاخص گذاری indexing

- **پاسخ به Query فقط با دسترسی به index**
- در برخی موارد اگر index مناسب داشته باشیم می توانیم بدون دسترسی به تاپل های جداول پایگاه داده، به query پاسخ دهیم.

Select DOB, count(*) from Employee group by DOB

- ✓ Index on DOB

Select AVG(sal) from Employee DOB = 1330 and sal between 3000 and 5000

- ✓ Index on "DOB, sal"

افزونگی داده ها data redundancy

- ✓ افزونگی یعنی تکرار بی رویه داده ها
- ✓ در بانک اطلاعات رابطه ای تکرار داده ها تنها راه برقراری ارتباط بین جداول است و از آن به عنوان کلید خارجی یاد می شود. تکرار بیش از این بی رویه است و افزونگی نام دارد. هنگامیکه جداول را ادغام می کنیم همواره افزونگی به میزان بزرگترین جدول رخ می دهد. این رخداد دو زیان دارد:
  - ❖ هدر دادن فضا
  - ❖ پایین آمدن سرعت

بی نظمی anomoly

- ✓ وجود افزونگی باعث بی نظمی در تغییر داده ها می شود.
- ✓ با ادغام کردن جداول، وارد کردن اطلاعات، حذف کردن و به روز درآوردن اطلاعات بسیار سخت می شود.
- ✓ کار کنترل مخصوصا کنترل جامعیت ارتباطی به معنای عام سخت می شود و پرس و جو ها مرتکب خطا می شوند.

مقادیر تهی NULL values

- ✓ با ادغام جداول از نشان دادن بعضی اقلام اطلاعاتی بدون استفاده از مقادیر تهی ناتوانیم.
- ✓ مقادیر تهی علاوه بر اینکه جای زیادی را اشغال می کنند مشکلات دیگری را نیز باعث می شوند.

وابستگی تابعی functional dependency

- ✓ افزونگی و وابستگی ارتباط تنگاتنگ دارند. تئوری وابستگی از مفاهیم عمده بانک اطلاعات است.
- ✓ وابستگی انواع مختلف دارد:
  - ❖ وابستگی تابعی functional dependency
  - ❖ وابستگی پیوندی join dependency
  - ❖ وابستگی چندمقداری multi-valued dependency

وابستگی تابعی functional dependency

- ✓ تعریف: اگر A و B دو مجموعه صفت در رابطه R باشند آنگاه وابستگی تابعی $A \rightarrow B$ برقرار است اگر برای تمام رابطه ها در R به ازای هر مقدار A فقط یک مقدار B داشته باشیم. (A کلید کاندید)
- ✓ در این تعریف نکات زیر قابل توجه است:

1. وابستگی تابعی باید برای تمام رابطه ها درست باشد یعنی از معنی و ذات آن صفت ها سرچشمه بگیرد نه از موارد خاص در یک یا چند رابطه.
2. وابستگی تابعی برای تعریف محدودیتهای بانک اطلاعات نیز به کار می رود.

وابستگی تابعی functional dependency

- تعریف: $A \rightarrow B$ را وابستگی تابعی کامل (full functional dependency یا FFD) می گوئیم اگر B به هیچ زیرمجموعه محض (proper subset) از A وابسته نباشد.
- تعریف: اگر برای تمام صفت های B در R داشته باشیم $A \rightarrow B$ آنگاه A را ابر کلید R می نامند و به صورت $A \rightarrow R$ نمایش می دهند. اگر این وابستگی از نوع FFD باشد آنگاه A کلید کاندید R است.
- تعریف: اگر B زیرمجموعه ای از A باشد آنگاه همواره $A \rightarrow B$. این وابستگی تابعی را بدیهی (trivial FD) می نامیم.

وابستگی تابعی functional dependency

- در جدول sec وابستگی تابعی زیر مشاهده می شود.
- (sec#,c#,s#,term)→pname
- این وابستگی از نوع FFD نیست زیرا داریم:
- (sec#,c#,term)→pname
- این وابستگی FFD است

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

وابستگی تابعی functional dependency

- در یک بانک باید از همه وابستگی های آن مطلع باشیم. تعدادی از وابستگی ها قابل شناسایی است و باید سایر وابستگی ها رو بدست آورد.

$$R=(S,T,U,V,W)$$
$$F=\{S\to T, V\to SW, T\to U\}$$

- مثال: فرض کنیم داریم:
- آنگاه منطقاً می توان وابستگی های تابعی زیر را نیز به دست آورد و نتیجه گرفت که V کلید کاندید R است.

$S\to U$ (زیرا $S\to T, T\to U$)
$V\to S$
$V\to W$
$V\to T$ (زیرا $V\to S, S\to T$)
$V\to U$ (زیرا $V\to S, S\to U$)

$$F^+=\{S\to T, V\to SW, T\to U, S\to U, V\to T, V\to U\}$$

وابستگی تابعی functional dependency

- **مجموعه پوششی وابستگی**
- **تعریف: اگر F یک مجموعه از وابستگی های تابعی باشد آنگاه مجموعه تمام وابستگی های تابعی که از آن منتج می شود را مجموعه پوششی Fمی نامیم و با F^+ نمایش می دهیم.**
- **آقای آرمسترانگ در سال ۱۹۷۴ روشی برای استخراج پوششی ارائه داد او ثابت کرد با اعمال مکرر سه قاعده زیر می توان به تمام وابستگی های منتج دست یافت و هیچ وابستگی اضافی نیز تولید نمی شود :**

وابستگی تابعی functional dependency

1. بازتاب (*reflexivity*) :اگر B زیر مجموعه A باشد آنگاه $A \to B$
2. افزایش (*augmentation*) :اگر $A \to B$ و C صفت باشد آنگاه $AC \to BC$
3. انتقال (*transitivity*) : اگر $A \to B$ و $B \to C$ آنگاه $A \to C$

✓ بعد ها دیگران قواعد دیگری را بیان کردند که کار را سهولت بخشید.

4. اجتماع (*union*) : اگر $A \to B$ و $B \to C$ آنگاه $A \to BC$
5. تجزیه (*decomposition*) : اگر $A \to BC$ آنگاه $A \to B$ و $A \to C$
6. ترکیب (*composition*) : اگر $A \to B$ و $C \to D$ آنگاه $AC \to BD$

وابستگی تابعی functional dependency

❖ مجموعه وابستگی بهینه

✓ با اعمال قواعد فوق وابستگی زیادی بدست می آید که بعضا تکراری و اضافی هستند. در اینجا روشی برای حذف اینگونه وابستگی ها و رسیدن به مجموعه وابستگی بهینه ارائه داده می شود.

✓ تعریف : دو مجموعه وابستگی تابعی F_1 و F_2 را معادل (equivalent) می نامند اگر مجموعه پوششی آنها برابر باشند یعنی $F_1^+ = F_2^+$

وابستگی تابعی functional dependency

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

۱. سمت راست هر وابستگی فقط یک صفت باشد.
۲. هر صفتی که F^+ را تغییر نمی دهد از سمت چپ حذف شود.
۳. وابستگی های تکراری و اضافی حذف شود.

وابستگی تابعی functional dependency

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

$$R=(U,V,W,X,Y,Z)$$
$$F=\{U\to XY, X\to Y, XY\to ZV\}$$

حل : ابتدا F^+ را می یابیم و سپس آن را بهینه می کنیم .

$$F^+=\{U\to XY, X\to Y, XY\to ZV, U\to ZV\}$$

قاعده 1 روی وابستگی اول :

$$F^+=\{U\to X, U\to Y, X\to Y, XY\to ZV, U\to ZV\}$$

قاعده 2 روی $XY\to ZV$ با توجه به $X\to Y$:

$$F^+=\{U\to X, U\to Y, X\to Y, X\to ZV, U\to ZV\}$$

حال قاعده 1 روی سایر وابستگی ها :

$$F_{opt}=\{U\to X, U\to Y, U\to Z, U\to V, X\to Y, X\to Z, X\to V\}$$

وابستگی تابعی functional dependency

کلید های کاندید

اگر مجموعه ای از صفت ها را ATTR و مجموعه وابستگی تابعی آنها و تعدادی صفت دیگر را F بنامیم آنگاه الگوریتم زیر، مجموعه تمام صفت های وابسته بهATTR را می دهد. (بهتر است ابتداF را بهینه کنیم).

$ATTR^+ = ATTR$

تکرار کن

برای هر $X \to Y$ در F

اگر X زیر مجموعه $ATTR^+$ باشد آنگاه

$ATTR^+ = ATTR^+ UY$

تا زمانیکه $ATTR^+$ دیگر تغییر نکند .

وابستگی تابعی functional dependency

با استفاده از این الگوریتم می توان کلید های کاندید را به دست آورد. بطور کلی باید به نکات زیر توجه کرد.

1. هر کلید کاندید شامل مجموعه ای از صفتهائی است که در سمت چپ پیکانها می آیند.
2. کلید کاندید باید کمینه باشد، یعنی زیر مجموعه ای از آن خاصیت کلیدی نداشته باشد.
3. بانک اطلاعات ممکن است چند کلید کاندید داشته باشد.
4. کلید کاندید ممکن است در یک یا چند صفت مشترک باشند.

وابستگی تابعی functional dependency

مثال : اگر $R = (S, T, U, V, W, X, Y)$

$F = \{S \to T, V \to SW, T \to U, SX \to Y\}$

آنگاه

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

چون V^+ و SX^+ هر کدام بیشترین تعداد صفت ها را می دهند می توان از آنها شروع کرد. اگر از V^+ شروع کنیم خواهیم داشت $VXY \to R$ اما چون کلید کاندید باید کمینه باشد پس $VX \to R$ (زیرا $X \to Y$ و $VX \to S$)

اگر از SX^+ شروع کنیم خواهیم داشت $SXVW \to R$. پس از کمینه کردن S و W حذف می شوند و خواهیم داشت $VX \to R$ که همان کلید قبلی است.

وابستگی تابعی functional dependency

- هر گاه تعداد صفت ها در بانک زیاد و وابستگی آنها گسترده باشد، می توان با رسم نمودار وابستگی به فهم بهتر آن بانک اطلاعات کمک کرد. در این نمودار، صفتهای ترکیبی در مستطیل قرار می گیرند و پیکانی از آنها به هر یک از صفتهای وابسته به آن رسم می شود.
- برای مجموعه صفتها و وابستگان آنها نیز مستطیل های دیگر و پیکانهای دیگری رسم می گردد. معمولا پیکانهایی که وابستگی به کلید اصلی رانشان می دهند در بالای صفت ها و سایر پیکانها زیر آنها کشیده می شوند. ابتدا باید مجموعه بهینه وابستگی ها را به دست آوریم و آنگاه به رسم نمودار وابستگی مبادرت کنیم.

وابستگی تابعی functional dependency

مثال: در بانک اطلاعات زیر ابتدا همه کلید های کاندید را بیابید و سپس نمودار وابستگی را رسم کنید.

$$R=(U,V,W,X,Y,Z,O,P,Q)$$

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

حل :ابتدا مجموعه بهینه معادلFرا پیدا می کنیم.

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

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

سپس مجموعه صفتهای وابسته به تمام مجموعه صفتهای چپ پیکانها را می یابیم.

$$U^{+}=\{U\}\Rightarrow\{U,V,X,Q\}$$

$$UP^{+}=\{U,P\}\Rightarrow\{U,P,V,X,Q,O,Y\}\Rightarrow\{U,P,V,X,O,Q,Y,Z\}$$

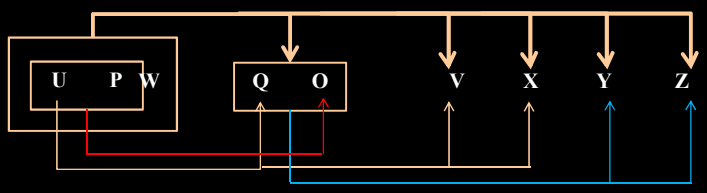
$$OQ^{+}=\{O,Q\}\Rightarrow\{O,Q,Y,Z\}$$

وابستگی تابعی functional dependency

بیشترین صفتها را UP^{+}می دهد تنها صفتی که از UP^{+}خارج است W است.

پس UPWکلید کاندیدی می باشد یعنی UPW→ R

کلید کاندید دیگری به دست نمی آید ، زیرا از QO نمی توان به P و U رسید(کلید کاندید باید کمینه باشد). نمودار وابستگی این بانک اطلاعات در شکل زیر مشاهده می شود.



نرمال سازی normalization

- همواره این سوال در ذهن طراحان بانک اطلاعات مطرح می شود که آیا آنچه ما ارائه داده ایم بهترین است؟ در مدل رابطه ای، روشی برای پاسخگوئی به این سوال دارد که آن را نرمالسازی می نامند. با استفاده از این متدولوژی می توان جداول خوبی طرای کرد و یا جداول موجود را بهتر نمود. در نرمال سازی قدم های زیر برداشته می شود:

۱. از جداول موجود یا لیستهای خروجی یا مشخصات سیستم شروع کن.
۲. داده ها و ارتباط ها وابستگی ها را شناسائی کن.
۳. ترجیحا نمودار وابستگی را رسم کن.
۴. جداول را قدم به قدم به فرم های نرمال تا سطح لازم تبدیل کن.

نرمال سازی normalization

- نرمال سازی در عمل یعنی پیروی از یک سری فرم های نرمال که منجر به تجزیه جداول می شوند. آقای کاد سه فرم نرمال معرفی کرد که به 1NF، 2NF و 3NF معروفند. شخص دیگری به نام بویس همراه با ایشان فرم نرمال دیگری به نام BCNE تعریف کرد. این چهار فرم نرمال بر مبنای وابستگی تابعی تعریف شده اند.
- دیگران وابستگی های دیگر و فرمهای نرمال دیگر مانند 4NF و 5NF معرفی کردند که نادر هستند و در موارد ویژه ای کاربرد دارند. فرم های نرمال هرکدام مشکلی را که در فرم قبلی وجود دارد حل می کنند. فرم 1NF درواقع ملزومات مدل رابطه ای را بیان می کند.

نرمال سازی normalization

- مزایای نرمال ترسازی
  - ✓ ارائه یک طراحی بهتر و واضح‌تر با کمترین اختلاط اطلاعات
  - ✓ کاهش بعضی انواع افزونگی
  - ✓ کاهش بعضی آنومالیها
  - ✓ تسهیل اعمال بعضی قواعد جامعیت
- معایب روش نرمالترسازی
  - ✓ بروز فزونکاری در سیستم در عمل بازیابی
  - ✓ ۲- ایجاد نوعی افزونگی از نوع افزونگی در سطح ادراکی
  - ✓ ۳- زمانگیر بودن فرآیند نرمالترسازی (محیط عملیاتی بزرگ و تعداد رابطه‌ها)
  - ✓ ۴- تصمیم‌گیری دشوار در اثر تعدد تجزیه‌ها

نرمال سازی normalization

- تعریف : جدولی در 1NF است که:

۱. همه کلید های آن تعریف شده باشند.

۲. تمام صفتهای آن به کلید اصلی وابستگی تابعی داشته باشند.

۳. صفتهای آن از دامنه تو در تو nested domain نباشد.

منظور از شرط سوم این است که صفات ترکیبی نداشته باشیم، یعنی مثلا یا صفتی به نام آدرس بدون توجه به بخشهای آن (شهر و ...) داشته باشیم یا بخشهای آن را بدون توجه به کل.

نرمال سازی normalization

❖ 1NF

✓ مثال (الف:جداول مقدماتی)

$$crs = \{c\#, cname, unit, c\lg\#\}$$
$$f1 = \{c\# \rightarrow cname, unit, c\lg\#, (cname, unit) \rightarrow c\lg\#\}$$

$$c\lg = (c\lg\#, c\lg name, city, pname)$$
$$f2 = \{c\lg\# \rightarrow c\lg name, city, pname\}$$

$$\text{sec} = (c\#, \text{sec}\#, s\#, pname, term, score, time, place)$$
$$f3 = \{(s\#, c\#, term) \rightarrow \text{sec}\#, score, (\text{sec}\#, c\#, term) \rightarrow pname, time, place\}$$

نرمال سازی normalization

❖ 1NF

✓ مثال (ب: کلیدهای کاندید)

✓ جداول crs و clg دارای یک کلید کاندید هستند که همان کلید اصلی است.

✓ کلید جدول sec را پیدا می کنیم:

کلید اول

$$(s\#, c\#, term)+ = (s\#, c\#, term, \text{sec}\#, score)$$
$$= (s\#, c\#, term, \text{sec}\#, score, pname, time, place)$$

در نتیجه $(s\#, c\#, term)$ کلید کاندید اول جدول sec است .

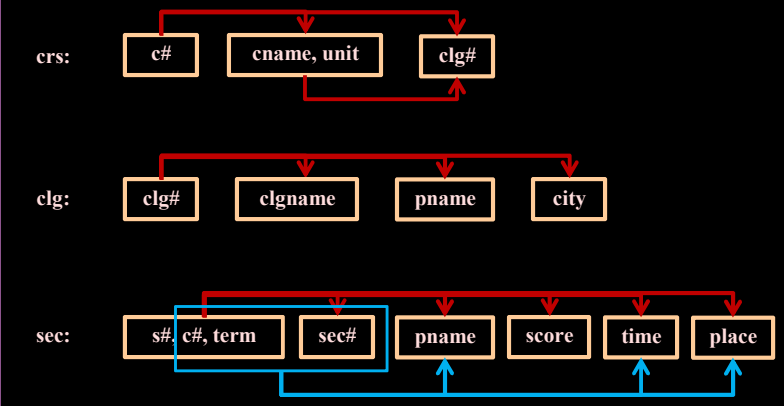
کلید دوم

$$(\text{sec}\#, c\#, term)+ = (\text{sec}\#, c\#, term, pname, time, place)$$

با توجه به کلید اول ، باید s# را به این مجموعه افزود که(s#,sec#,c#,terme) به دست می آید .این کلید دیگر کمینه نیست و با حذف sec# همان کلید اول به دس می آید که تنها کلید کاندید و کلید اصلی جدول است .

نرمال سازی normalization

1NF



❖ سه جدول در1NF هستند زیرا کلید اصلی دارند و همه صفتهای هر کدام به کلید اصلی وابسته است.

نرمال سازی normalization

❖ تعریف : جدولی در 2NF است که:

۱. در 1NF باشد.

۲. صفتهای آن به زیرمجموعه های کلید اصلی وابستگی نداشته باشند (وابستگی جزئی یا partial dependency)

❖ تبدیل جدول 1NF به جدول 2NF:

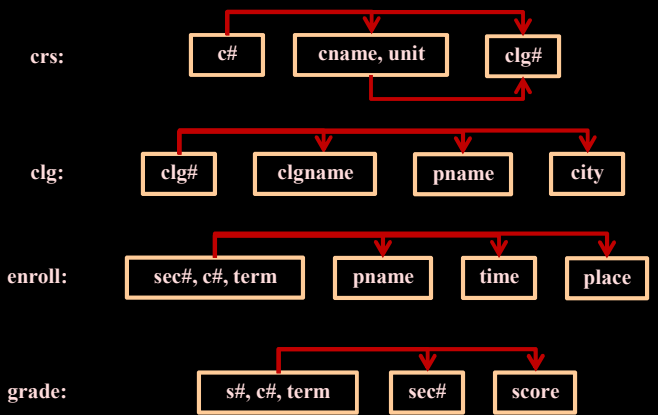
۱. هر بخش از کلید اصلی را که ایجاد وابستگی جزئی کرده است، با همه وابسته های آن کنار هم بگذار

۲. کل کلید اصلی را با صفت های باقیمانده کنار هم بگذار

۳. صفت های کلیدی را به عنوان کلید خارجی به ۲ اضافه کن

نرمال سازی normalization

2NF



نرمال سازی normalization

- تعریف : جدولی در 3NF است که:

۱. در 2NF باشد.

۲. وابستگی انتقالی (وابستگی های غیرکلیدی) نداشته باشد.

- تبدیل جدول 2NF به جدول 3NF:

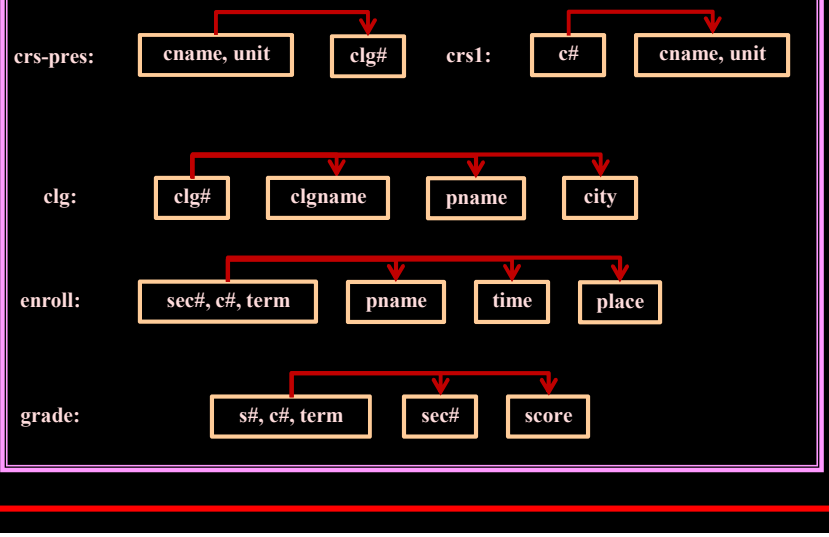
۱. مجموعه صفاتی را که وابستگی انتقالی ایجاد کرده است، با همه وابسته های آن کنار هم بگذار

۲. کلید اصلی را با صفتهای باقیمانده کنار هم بگذار

۳. صفت های کلیدی را به عنوان کلید خارجی به ۲ اضافه کن

نرمال سازی normalization

3NF



نرمال سازی normalization

- از 3NF تا BCNF
- 3NF با جداولی که هر سه شرط زیر را دارند ممکن است مشکل داشته باشد. در این صورت باید این نوع جداول را تا سطح بیشتری نرمال سازی نمود. این سه شرط عبارتند از :

۱. جدول دارای حداقل دو کلید کاندید باشد
۲. این کلید های کاندید ترکیبی باشند
۳. این کلید های ترکیبی ، صفت های مشترکی داشته باشند

نرمال سازی normalization

- از 3NF تا BCNF
- مثال: جدول دانشجو در یک موسسه کوچک که دانشجوی همنام ندارد (در این صورت هم شماره دانشجوئی کلید کاندید است و هم نام دانشجو):

$$stud(s\#, sname, address, avg)$$
$$candidate \quad key(s\#)$$
$$candidate \quad key(sname)$$

- وابستگی:

$$s\# \rightarrow stud$$
$$sname \rightarrow stud$$

- این جدول در 3NF هست. از سه شرط بالا فقط شرط ۱ را داراست. بنابراین نیاز به نرمال سازی بیشتر ندارد

نرمال سازی normalization

- از 3NF تا BCNF
- مثال: جدول " درس – دانشجو " در همین موسسه:

$$crs\_stud(s\#, sname, c\#, score)$$
$$candidate \quad key(s\#, c\#)$$
$$candidate \quad key(sname, c\#)$$

- وابستگی ها:

$$(s\#, c\#) \rightarrow crs\_stud$$
$$(sname, c\#) \rightarrow crs\_stud$$
$$s\# \rightarrow sname$$
$$sname \rightarrow s\#$$

- این جدول هر سه شرط بالا را داراست. بنابراین ممکن است نیاز به نرمال سازی بیشتر داشته باشد. داده های زیر نشان می دهد که این جدول دارای افزونگی است و برای برطرف نمودن آن نیاز به نرمال سازی بیشتر دارد.

نرمال سازی normalization

- از 3NF تا BCNF
- می توان این جداول را به دوجدول grade (s#,c#,score), st(s#,sname) شکست (در جدول grade می توان به جای s# ازsname نیز استفاده کرد.)

crs_stud

S#	sname	C#	Score
S1	علی	C1	17
S1	علی	C2	12
S1	علی	C3	19
S1	علی	C4	20
S1	علی	C5	10

افزونگی

نرمال سازی normalization

- از 3NF تا BCNF
- مثال: جدول امتحان شامل شماره دانشجوئی، شماره درس، رتبه دانشجو در درس (غیر تکراری):

$$Exam(s\#, subj\#, rank)$$
$$candidate \quad key(s\#, subj\#)$$
$$candidat \quad key(subj\#, rank)$$

- وابستگی ها:

$$(s\#, subj\#) \rightarrow Exam$$
$$(subj\#, rank) \rightarrow Exam$$

- این جدول در 3NF هست و هر سه شرط را نیز داراست، ولی نیاز به نرمال سازی بیشتر ندارد (با در نظر گرفتن داده هایی می توان دید که این جدول افزونگی ندارد).
- اگر جدولی در فرم BCNF باشد نیاز به شکستن ندارد.

نرمال سازی normalization

- تعریف : جدولی BCNF است که ستون های آن فقط به کلید های کاندیدش وابستگی تابعی داشته باشند.
- در مثال های ۱ و ۳ ، وابستگی به غیر کلید های کاندید وجود ندارد ولی در مثال ۲ این نوع وابستگی وجود دارد(sname→s#, s#→sname)
- توجه به نکات زیر در موردBCNF ضروری است :

1. برخلاف فرم های نرمال دیگر BCNF بدون استفاده از 3NF و فرم های قبلی نرمال تعریف می شود. غالبا می توان بانک اطلاعات را با استفاده از تعریفBCNF در یک قدم نرمال کرد و نیازی به تعریف وابستگی هایی از قبیل وابستگی انتقالی نیست. بنابراین BCNFجامعترین تعریف نرمال سازی بر مبنای وابستگی تابعی را به طور مستقل ارائه می دهد.
2. در مواردی، نرمال سازی تاسطح BCNFلازم نیست و بهتر است از تبدیل جدول فرم 3NF به BCNF خوددار ی کرد.

نرمال سازی normalization

- وابستگی چندمقداری (MVDیا multivalued dependency)
- وابستگی چند مقداری نوعی وابستگی بین دو مجموعه مستقل از صفتهاست که از وابستگی تابعی عام تر است. این حالت در مثال زیر مشاهده می شود.
- مثال : جدول دانشجو شامل نام دانشجو، اساتیدی که دانشجو با آنها درس دارد و وامهایی که دانشجو دریافت کرده و تاریخ آن وام ها.
- در این مثال، نام اساتید دانشجو و وام های دریافت شده توسط دانشجو از یکدیگرمستقل هستند یعنی وابستگی تابعی ندارند. اگر یک دانشجو چند استاد داشته باشد و چند وام دریافت کرده باشد، افزونگی داده ایجاد می شود.

نرمال سازی normalization

- **وابستگی چندمقداری (MVDیا multivalued dependency)**

Sname	Prof	Loan	Date
حمید	حق جو	مسکن	1381
حمید	حق جو	ضروری	1383
حمید	جاهد	مسکن	1381
حمید	جاهد	ضروری	1383
حمید	حق جو	ضروری	1384

- در جدول فوق مشاهده می شود که نام اساتید و وام های حمید تکرار شده اند(افزونگی). این در حالی است که جدول فوق تا سطح BCNF نرمال سازی شده است زیرا هیچگونه وابستگی معنی دار به جز وابستگی به کلید اصلی، که شامل تمام صفت ها است ، وجود ندارد.

نرمال سازی normalization

- **وابستگی چندمقداری (MVDیا multivalued dependency)**

این نوع وابستگی و فرم نرمال مربوط به آن به صورت های زیر تعریف می شود :

- **تعریف : هر گاه دو ارتباط مستقل بین مجموعه صفت های یک رابطهR مثل A:B وA:C وجود داشته باشد، وابستگی چند مقداری در را بطه R برقرار است که به دو صورت زیر نشان داده می شود: $A \rightarrow\rightarrow B \mid C$ یا $A \rightarrow\rightarrow B , A \rightarrow\rightarrow C$**
- **وابستگی چند مقداری به شکل های متفاوت در منابع مختلف تعریف شده است. یک تعریف دیگر هم در زیر آمده است:**
- **تعریف : در رابطه$R(A,B,C)$ داریم$A \rightarrow\rightarrow B \mid C$ اگر هر مقدارA به طریقی به مقادیر B وابسته باشد که بهC ارتباطی پیدا نکند.**
- **مثال : در جدول فوق وابستگی های چند مقداری زیروجود دارند:**

$$sname \rightarrow\rightarrow prof , sname \rightarrow\rightarrow loan, data$$

یا

$$sname \rightarrow\rightarrow prof \mid loan.date$$

نرمال سازی normalization

- وابستگی چندمقداری (MVDیا multivalued dependency)
- تعریف :یک وابستگی چند مقداری A→→ B در رابطه R را بدیهی (trivial) گویند اگر یکی از دو شرط زیربرقرار باشد:
  1. $B \subset A$
  2. $A \cup B = R$
- اگر در یک وابستگی چند مقداری هیچ کدام از دو شرط فوق برقرار نباشد، به آن وابستگی چند مقداری غیر بدیهی(nontrivial) گویند. در جدول فوق هر دو وابستگی چند مقداری، غیر بدیهی هستند.
- تعریف : رابطه R در 4NF است در صورتیکه اگروابستگی چند مقداری غیر بدیهی A→→ B در R وجود داشته باشد آنگاه A ابرکلید R باشد. جداولی که دارای وابستگی چند مقداری هستند را می توان به صورت زیر تجزیه کرد و افزونگی را از بین برد.

نرمال سازی normalization

- وابستگی چندمقداری (MVDیا multivalued dependency)
- جدول اول شامل مجموعه صفت هایB,A
- جدول دوم شامل مجموعه صفت هایC,A
- مثال :جدول قبل به دو جدول زیر تجزیه می گردد و افزونگی از بین می رود.

sname	Loan	Date
حمید	مسکن	1381
حمید	ضروری	1383
حمید	ضروری	1384

sname	Prof
حمید	حق جو
حمید	جاهد

نرمال سازی normalization

- وابستگی پیوندی (JDیا join dependency)
- در مواردی نمی توان جدولی را بطور صحیح به دو جدول تجزیه نمود ولی تجزیه آن به سه جدول یا بیشتر امکان پذیراست (شرایط صحت تجزیه در زیر آمده است). چنین جداول دارای وابستگی پیوندی هستند. این مفهوم ابتدا با مثال بیان می شود و سپس همراه با فرم نرمال مربوطه تعریف می گردد.
- مثال :جدول ABC زیر را که در آن هر سه ستون روی همدیگر کلیدکاندید را تشیل می دهند می توان به سه جدول Cو Bو A تجزیه نمود ولی تجزیه آن بطور صحیح به دو جدول امکان ندارد.
- تعریف : اگرRیک رابطه و ستون های هر یک از رابطه های A,B,…,P زیر مجموعه ستون های Rباشند آنگاه R دارای وابستگی پیوندی روی A,B,…,P است اگر و تنها اگر داشته باشیم R = A∞B∞…∞P
- تعریف : جدولR در 5NF است اگر و تنها اگر فقط به کلید های کاندیدش وابستگی پیوندی داشته باشد.

نرمال سازی normalization

- وابستگی پیوندی