اصول طراحی
پایگاه داده ها

فهرست مطالب

فهرست اشکال

پیشگفتار:

انسان همواره در زندگی روزانه اش نیاز به اطلاعات[1] داشته است.به بیان دیگر انسان اطلاعاتی از حقایق خام که به داده ها معروفند را جمع آوری می کند و بر اساس آن داده ها یا اطلاعات جدید تولید و از آنها استفاده می نماید.برای بهره برداری مجدد از داده ها در فعالیتهای روزانه و انتقال به آیندگان و دیگران لازم است آنها ذخیره و بازیابی مجدد گردند.اگر حجم اطلاعات کم و اندک باشد ، سیستمهای ابتدایی مفید خواهد بود و لیکن با فزونی حجم اطلاعات می بایست آنها بر اساس یک سیستم و ساختار مشخص نگهداری شوند.به بیان دیگر داده ها با حجم زیاد هنگامی مفید و کاراترند که در یک پایگاه داده ها[2] ذخیره شوند.موضوع مطالب گنجانده شده این دوره شامل دو بخش:۱- طراحی پایگاه داده ها که شامل مفاهیم و روابط است و ۲- زبان پرس و جوی SQL که دستورات زبان تعریف داده و زبان دستکاری داده و زبان کنترل داده با ذکر مثال معرفی گردیده شده اند.

---

[1] Information
[2] Database

بخش ۱

عنوان:

طراحی پایگاه داده ها

مفاهیم اساسی بانک اطلاعاتی:

Data

یک سری داده های خام می باشد که از آنها اطلاعات ساخته می شود.

Information

اطلاعات به معنی دانشی است که از طریق خواندن، مشاهده و آموزش به دست می آید. اطلاعات داده هایی هستند که جمع آوری شده و پردازش می گردند تا شکلی با مفهوم تولید نمایند.بدین منظور کامپیوترها و سایر ماشینهای مرتبط با آنها داده ها را به روشهای گوناگون پردازش می کنند تا اطلاعات مفیدی تولید نمایند.

Entity

به هر چیزی که باید اطلاعاتی در مورد آن ذخیره کنیم Entity یا موجودیت می گویند.مثلاً در سیستم دانشگاه دانشجو، استاد و... موجودیت هستند.

صفت خاصه

به هر کدام از اجزاء اطلاعات یک فرد یا یک شئ (مثل نام ،نام خانوادگی ، شماره دانشجویی)، صفت خاصه گویند .صفت خاصه در هر مکانی ممکن است متفاوت باشد .

سلسله مراتب سازمان داده ها و اطلاعات در نشست فیزیکی بر روی رسانه ذخیره سازی

در نشست فیزیکی داده ها و اطلاعات بر روی رسانه ذخیره سازی ، سازمان داده ها و اطلاعات در یک سلسله مراتب شش سطحی تجلی می یابد که سطوح آن به ترتیب بیت، کاراکتر، فیلد، رکورد ، فایل و پایگاه داده ها می باشد.هر سیستم اطلاعاتی دارای سلسله مراتب سازمان داده ها و اطلاعات می باشد و هر سطح در راستای سلسله مراتب ، از ترکیب عناصر پایین تر به وجود می آید.

- ### کاراکتر یا نویسه

یک کاراکتر به صورت گروهی از بیتها نمایش داده می شود و پیکر بندی آن مبتنی بر یکی از سیستمهای کدگذاری اسکی یا ابسیدیک می باشد.همانگونه که بیت واحد اولیه حافظه است کاراکتر نیز واحد اولیه جهت درک انسان می باشد.

- **فیلد یا میدان**

فیلد یا میدان که عنصر داده ها نیز نامیده می شود پایین ترین سطح یک واحد منطقی در سلسله مراتب داده ها و اطلاعات می باشد به عبارتی دیگر همان مفهوم صفت خاصه است.

- **رکورد یا سابقه**

رکورد یا سابقه شرح یک رویداد نظیر تولد ، یا یک اقدام نظیر فروش یا رزرو جا در هواپیما و یا شرح یک قلم کالا نظیر قطعه یدکی و یا مشخصات یک فرد نظیر مشتری بانک می باشد. به مجموعه ای یا تعدادی فیلد رکورد گفته می شود.

- **فایل یا پرونده**

یک فایل مجموعه ای از رکوردهای بهم مرتبط می باشد.مثلاً پرونده اخذ جرائم شامل سابقه های کلیه خودروهایی است که دارای خلاف می باشند.

- **پایگاه داده یا بانک اطلاعاتی**

مجموعه ای از فایلها یا پرونده ها که به نحوی منطقاً بهم مرتبط می باشند پایگاه داده یا بانک اطلاعاتی نامیده می شوند.بانک اطلاعاتی مجموعه ای است از داده های ذخیره شده در مورد انواع موجودیت ها یا انواع entity های یک محیط عملیاتی و ارتباط بین آنها به صورت مستمر و مبتنی بر یک ساختار تعریف شده به صورت صوری با حداقل افزونگی تحت کنترل متمرکز و مورد استفاده یک یا چند کاربر به طور اشتراکی وهمزمان.

تاریخچه ای از بانک اطلاعاتی

نسل اول :فایل های ساده ترتیبی

ویژگیها:

۱- نوار مغناطیسی وجود داشت .

۲- فایل منطقی و فیزیکی یکی بوده است.

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

برای انجام عملیات بهنگام سازی ، الزاماً فایل دیگری ایجاد و تغییرات را در آن وارد می کردند، نسخه قدیمی را به عنوان فایل پدر ، فایل پدربزرگ و... نامگذاری می نمودند.

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

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

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

افزونگی:یعنی تکرار ذخیره سازی

نسل دوم: شیوه های دست یابی[۳]

در این نسل حافظه های جانبی از نوع دیسک اختراع شد به همین دلیل شیوه دست یـابی از حالت تر تیبی خارج شده به صورت مستقیم شد.

ویژگیها:

۱- مستقیم شدن دست یابی
۲- وجود نرم افزار a.m
۳- جدا شدن فایل منطقی و فیزیکی
۴- عدم وجود نرم افزار مدیریت
۵- افزونگی بالا

نسل سوم: سیستم مدیریت Data

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

ویژگیها :

۱- نرم افزار مدیریت بوجود آمد.
۲- فایل های منطقی متعددی از یک فایل فیزیکی به دست آمد.
۳- اشتراک داده ها یعنی افزونگی دیگر وجود نداشت.

نسل چهارم: DBMS

در نسل چهارم استقلال داده به طور کامل رعایت می شود. در این نسل برنامه جـامعی بـه نـام DBMS به وجود آمد که وظیفه آن مدیریت بانک اطلاعاتی است و واسطی است بین برنامه های کاربردی و محیط فیزیکی. مهمترین ویژگی این نسل چند سطحی شدن معمـاری بانـک اسـت . Oracle, Access, SQL Serverوغیره همگی DBMS هستند.

نسل پنجم:knowledge base یا هوش مصنوعی که در این زمینه کامپیوتر نیز قابلیت آموزش را دارد.

مثلاً:به جای نوشتن یک دستور پرس و جو جهت گزارش گیری از دانشجویان معدل الـف، کـاربر دستور زیر را بنویسد:لیست تمامی دانشجویان معدل الف را نمایش بده.

---

[۳] access method

مزایای سیستم بانک اطلاعاتی

۱. مدلینگ داده های عملیاتی بر اساس ساختار آن ها

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

۲. وحدت ذخیره سازی کل داده های عملیاتی

منظور از وحدت ذخیره سازی داده ها این است که همه داده های مربوط به یک **Entity** در یک ساختار ذخیره می شود.

۳. اشتراکی شدن داده ها

سیستم **DBMS**[۴] امکان می دهد که کاربران از داده های ذخیره شده به طور اشتراکی استفاده کنند و هر کاربر دید خارجی مخصوص به خود داشته باشد ، در صورتی که یک فایل فیزیکی بیشتر وجود ندارد .

۴. کاهش میزان افزونگی

وحدت ذخیره سازی باعث کاهش افرونگی می شود .

۵. تعدد شیوه های دست یابی و دست یابی آسان به داده ها

وجود سطح ادراکی و نگاشت های آن بین سطوح خارجی و داخلی باعث می شود برای دست یابی به داده از شیوه های متعدد دست یابی استفاده شود شیو های دست یابی را مدلها تعیین می کنند.

۶. عدم وجود ناسازگاری داده ها[۵]

منظور از ناسازگاری این است که افزونگی در بانک وجود داشته باشد و زمان به هنگام سازی ، بعضی از داده ها را ویرایش کنیم و مابقی بدون ویرایش بماند. مثلا معدل دانشجو در ۲ جای مختلف ذخیره شود و هنگام تغییر کردن فقط یکی از آنها را تغییر دهیم. با حذف افزونگی ناسازگاری نیز از بین می رود.

۷. تأمین جامعیت

بی نقص بودن داده ها را در بانک جامعیت می گویند.مفهوم میدان جامعیت را تأمین می کند مثلا نمره منفی برای یک دانشجو نشان دهنده نقص بانک است یا در مورد دانشجویی که ثبت نام هویتی نداشته اما انتخاب واحد داشته است .

---

[۴] Data Base Management System
[۵] Inconsistency

۸. تأمین استقلال داده ای

دو نوع استقلال داده وجود دارد استقلال داده فیزیکی و منطقی.

۹. حفظ محرمانگی اطلاعات:

برای حفظ محرمانگی اطلاعات می توان از روشهای مختلفی مانند :

کنترل دست یابی ها (دیدهای خارجی) یا ذخیره سازی داده ها به طور رمـزی و یـا رمزگـذاری روی بانک استفاده کرد .

۱۰. تسریع در دریافت پاسخ پرس و جوها

معایب بانک اطلاعاتی

۱. به دلیل متمرکز بودن داده ها ممکن است آنها به خطر بیفتند و راه حل آن هم پشتیبان گیری است .
۲. به دلیل متمرکز بودن داده ها ممکن است جامعیت داده ها به خطر بیفتد .
۳. ممکن است نیاز به سخت افزار اضافه باشد .
۴. برنامه نویسی و پیاده سازی تمام مفاهیم بانک اطلاعاتی پیچیده است .

اصول و مبانی بانک اطلاعاتی

به دو روش می توان بانک اطلاعاتی ایجاد کرد.

۱- روش فایلی(کلاسیک)
۲- روش بانکی

روش فایلی(کلاسیک)

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

مثال:فایل CUSTOMER دارای ۵ رکورد است.هر رکورد دارای ۶ فیلد به نامهای Agent, C-Zip, C-phone, C-name, A-phone و R-Date می باشد.

شماره	C-name	C-phone	C-Zip	Agent	A-phone	R-Date
۱	احمدی	۶۵۸۲۵۰	۰۳۱	آزادی	۶۱۵۸۱۲	۷۵/۰۲/۰۳
۲	سعادتجو	۷۱۳۸۱۲	۰۳۱	عسگری	۷۱۳۲۱۲	۷۵/۰۸/۰۵
۳	سعادتجو	۶۱۵۸۲۲	۰۳۱	آزادی	۶۱۵۸۱۲	۷۵/۰۵/۲۱
۴	قاسمی	۶۱۵۸۲۱	۰۲۱	آزادی	۶۱۵۸۱۲	۷۵/۰۲/۱۵

۵ اکبری ۶۱۵۲۱۵ ۰۶۸۱ حسینی ۶۱۵۸۳۳ ۷۵/۱۱/۱۰

A-phone= تلفن نمایندگی ،R-Date = تاریخ تمدید،C-phone =تلفن مشتری،C-Zip =پیش کد مشتری ، C-name= نام مشتری

برای گزارشات مورد نیاز اداره فروش به شرح زیر نوشته می شود:

- تهیه گزارش از خلاصه عملکرد ماهیانه بر حسب نوع و مقدار بیمه فروخته شده به وسیله نمایندگیها.(چنین گزارشی امکان تجزیه و تحلیل میزان کارائی هر نمایندگی را فراهم می سازد.)
- تهیه گزارشی ماهیانه از مشتریانی که بیمه نامشان باید تمدید شود.
- تهیه نامه هایی برای مشتریان که حاوی خلاصه ای از پوششها و تشویقها باشد.
- تهیه گزارشهایی تحلیلی از نسبتهای نوع بیمه های فروخته شده.

برنامه های دیگری برای تولید گزارشات جدید با گذشت زمان نوشته خواهد شد.هر فایل در سیستم ، برنامه مربوط به خودش را برای ذخیره و بازیابی و اصلاح داده ها استفاده می نماید.هر فایل متعلق به شخص یا اداره ایست که متولی ایجاد آن بوده است.هر چه سیستم فایلی رشد و بزرگ شود تقاضا برای برنامه نویسی افزایش و نیاز به رایانه ای بزرگتر و پیچیده تر خواهد شد.

نقدی بر سیستم فایلی

۱- مدیریت داده ها در سیستمهای فایلی

برای ساده ترین بازیابی اطلاعات نیاز فراوان به برنامه نویسی به زبان نسل سوم می باشد.در یک زبان نسل سوم ، برنامه نویس لازم است معین کند چه باید انجام شود و چگونه انجام شود.برنامه نویسی درزبانهای نسل سوم مثل کوبول،بیسیک،پاسکال و فرترن نیاز به مهارت بالا دارد.برنامه نویس باید روش دسترسی به فایلهای مختلف و اجزای سیستم را تعریف نماید.هر چه سیستم فایلی پیچیده تر شود ، روش دسترسی به اطلاعات پیچیده تر و منجر به سیستمی غیر عملی تر خواهد شد.

هر چه تعدا فایلها در سیستم گسترده تر شود مدیریت سیستم دشوارتر خواهد شد.هر فایل یک مدیریت سیستم فایل مستقل دارد که از اعمالی چون ایجاد فایل، اضافه کردن داده ها به فایل ، حذف داده ها ، اصلاح داده ها و لیست نمودن محتوای فایل تشکیل شده است.

۲- وابستگی ساختاری و داده ای

تغییر در ساختار فایل مانند حذف و اضافه یک فیلد نیازمند اصلاح کلیه برنامه هایی است که از آن فایل استفاده می کنند.حتی تغییر در یک خصیصه یک فیلد داده مانند تغییر فیلدی از صحیح به اعشاری نیاز به تغییر در کلیه برنامه هایی که دسترسی به آن فایل دارند.

۳– تعریف فیلدها

از نقطه نظر کاربر رکوردی بهتر است که نیازمندیهای گزارشاتش را پیش بینی کرده باشد و فیلدهای فایل قابل تفکیک باشند.داشتن یک شناسه یکتا و منحصر به فرد برای هر رکورد لازم است.مثلاً فایل CUSTOMER ممکن است چندین مشتری به نام محمدی باشد.در نتیجه فیلدی به نام C-Account که شامل حساب مشتری است و یکتا و منحصر بفرد می باشد ، نیاز است.قابل ذکر است که مشکل بیان شده تنها منحصر به سیستم فایلی نمی باشد.

۴–داده های تکراری

در محیط سیستم فایلی مشکل است داده ها در یک جا جمع گردند، داده های یکسان در محلهای مختلف ذخیره می شوند.برای مثال، داده های نمایندگی ممکن است در فایل پرسنلی ، در فایل فروش و در فایل حسابداری نگهداری شوند.چنین تکرار داده بدترین نوع اشتباه است چون یافتن و تصحیح آن بسیار سخت است.

روش بانکی

در این روش وحدت ذخیره سازی وجود دارد ولی هر کاربر دید خاصی از داده ها دارد .تغییر کردن فایل تأثیری بر روی برنامه های کاربردی ندارد به دلیل اینکه فقط یک فایل به وجود می آید. داده ها مجتمع هستند و مهمترین قسمت سیستم DBMS است.

روشهای ارتباط بین فایلها در یک پایگاه داده

توانایی در حفظ و نگهداری روابط بین فایلها یکی از مقدورات سیستم مدیریت پایگاه داده می باشد.هنگامی که یک رکورد داده ها از یک فایل با یک رکورد داده ها از فایل دیگر بهم مرتبط شوند فایلها از ارتباط با یکدیگر برخوردارند.

در یک پایگاه داده فایلها می توانند به سه روش زیر بهم مرتبط شوند:

۱- ارتباط یک به یک

۲- ارتباط یک به چند و چند به یک

۳- ارتباط چند به چند

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

ارتباط یک به یک

به عنوان مثال، اگر در یک آموزشگاه آزاد برای بهبود وضعیت تحصیلی دانش آموزان معلمان خصوصی اختصاص داده شوند به گونه ای که هر دانش آموز از تدریس یک معلم خصوصی بهره بگیرد، در این حالت ارتباط بین فایل دانش آموزان و فایل معلمان و معلمان خصوصی یک رابطه یک به یک می باشد.

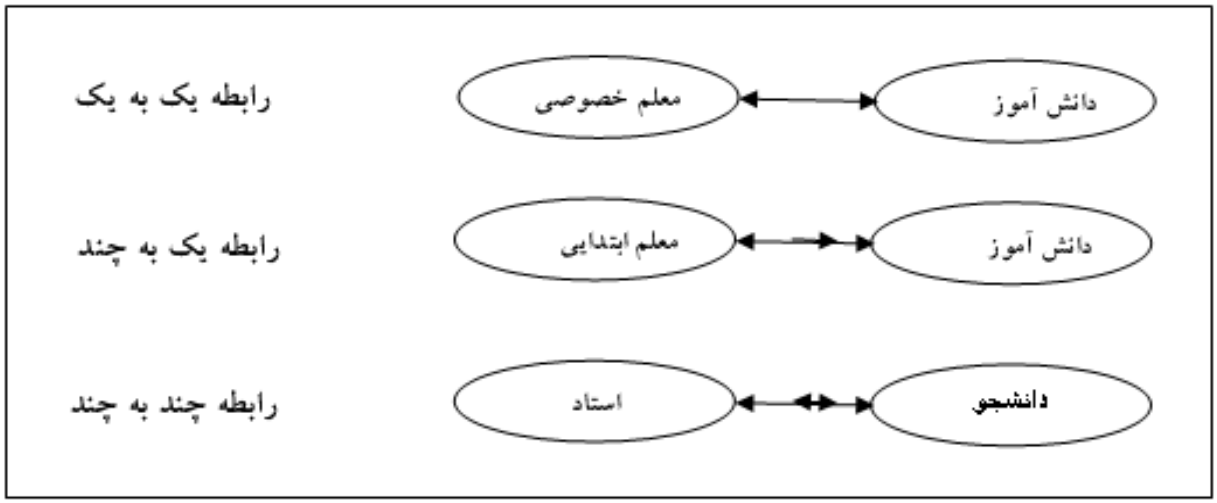
ارتباط یک به چند و چند به یک

اگر در یک دبستان یک معلم تمام دروس ابتدایی را به دانش آموزان یک کلاس آموزش دهد ارتباط بین فایل معلمان و فایل دانش آموزان یک ارتباط یک به چند می باشد و از طرف دیگر چون هر دانش آموز فقط یک معلم دارد ارتباط بین فایل دانش آموزان و فایل معلمان یک ارتباط چند به یک می باشد.

ارتباط چند به چند

در دانشگاه هر دانشجو از تدریس استادان مختلف بهره می گیرد و هر استاد دانشجویان زیادی را آموزش می دهد بدین لحاظ ارتباط بین فایل دانشجویان و فایل استادان یک ارتباط چند به چند می باشد.

روابط ذکر شده در شکل زیر به تصویر کشیده شده است.



شکل۱- انواع ارتباط

این شکل نشان می دهد که چگونه می توان روابط بین فایلها را به تصویر کشید.هر پیکان در شکل نشانگر روابط بین فایلهاست.هر پیکان با یک فلش منفرد یا یک زوج فلش فایلها را بهم مرتبط می سازد.فلش منفرد اشاره به رابطه "به یک"و زوج فلش اشاره به "به چند" دارد.

اجزا و عناصر محیط بانک

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

- یک ماشین محاسبه گر که معمولاً کامپیوتر است.
- یک سخت افزار ذخیره سازی
- در صورت امکان سخت افزار ارتباطی (برای بانکهای شبکه ای)

۲- نرم افزار

- نرم افزارهای کاربردی

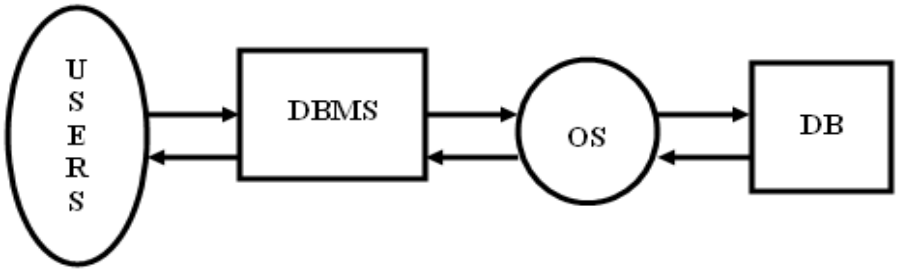
نرم افزاری است که کاربر باید برای تماس با سیستم بانک اطلاعاتی از آنها استفاده کند.

- نرم افزار سیستمی

۱- نرم افزار سیستمی مخصوص بانک(DBMS) ۲- نرم افزار سیستم عمومی(سیستم عامل)

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

a. کاربران با امکانات سیستم یکجا[۶]
b. کاربران با امکانات پایانه ای (کاربر بر خط)[۷]
c. اداره کننده پایگاه داده ها (DBA)[۸]



ارتباط ساده بین کاربران،سیستم مدیریت پایگاه داده ها ، سیستم عامل و پایگاه داده ها کاربران با امکانات سیستم یکجا معمولاً از یک زبان سطح بالای برنامه سازی برای انجام عملیات مورد نظرشان مثل تعریف داده ها و ذخیره و بازیابی داده ها استفاده می کنند.

کاربران با امکانات پایانه ای یک زبان پرس و جو در اختیار دارند که به کمک آنها داده هایشان را تعریف و با آنها کار می کنند.

اداره کننده پایگاه داده ها فردی است که مسئولیت ایجاد، پیاده سازی و نگهداری پایگاه داده ها را در محیط عملیاتی به عهده دارد.

---

۶ Batch Users
۷ End Users or On-line Users
۸ DataBase Administrator

هر سه دسته فوق کاربرانی هستند که با مفاهیم و اصول دانش و فن کامپیوتر آشنا بوده ، قادر به برنامه سازی و کار با کامپیوتر هستند و مبانی سیستمهای پایگاه داده ها را می شناسند.اما کاربران دیگری نیز وجود دارند که اساساً اطلاعاتی از دانش و فن کامپیوتر و تکنولوژی ذخیره و بازیابی ندارند.این گونه کاربران باید نیازهای اطلاعاتی خود را به کمک سیستم داده ها مرتفع سازند.برای این منظور لازم است بوسیله طراح پایگاه داده ها نرم افزارهای واسطه ای تهیه شود.

این واسطه ها برای کاربران این امکان را فراهم می اورند که در یک محیط دوستانه بوسیله منو با سیستم کار کنند.یعنی بتوانند داده های خود را وارد کنند ، سپس پاسخ سئولات خود را از سیستم بگیرند.

۴-**داده ها** : داده ها شامل حقایق ذخیره شده در مورد انواع موجودیت ها در یک محیط عملیاتی و ماورای داده ها برای نشان دادن ارتباط بین موجودیتها می باشند که به صورت مجتمع و مبتنی بر یک ساختار داده معین با حداقل افزونگی و تحت کنترل متمرکز تعریف شده بصورت اشتراکی و همزمان مورد استفاده کاربران سیستم قرار می گیرند.

محیط عملیاتی بصورت ساده به هر مؤسسه یا سازمان علمی، فرهنگی و یا تجاری که در نظر است رایانه ای گردد، گفته می شود.برای مثال یک کارخانه، یا بانک یا بیمارستان یا دانشگاه می تواند یک محیط عملاتی باشد.

معماری سیستم پایگاه داده

مدیریت سیستم پایگاه داده ها از تجمع داده عملیاتی و مجموعه برنامه هایی که اجازه می دهد چندین کاربر بصورت اشتراکی به داده ها دسترسی یابند ، تشکیل شده است.هدف اصلی مدیریت پایگاه داده ها فراهم آوردن یک دید انتزاعی از داده ها برای هر یک از کاربران سیستم می باشد به طوری که هر کاربر احساس نماید که تنها اطلاعات مورد نیاز خودش در سیستم وجود دارد.

در واقع سیستم باید جزئیات چگونگی ذخیره سازی یا نگهداری داده ها را از کاربر پنهان سازد.این امر به وسیله یک معماری سه سطحی توسط ANSI[۹] پیشنهاد گردیده است ، تحقق می یابد.
معماری پایگاه داده در سه سطح کلی به نام سطح داخلی ، سطح مفهومی و سطح خارجی می تواند تجلی یابد.

- سطح داخلی یا فیزیکی:سطحی است که به محیط فیزیکی ذخیره سازی از همه نزدیک تر است، یعنی سطحی که با روشی که در آن داده ها به مفهوم واقعی ذخیره می شوند مرتبط است.
- سطح مفهومی (ادراکی) :سطحی است بین دو سطح داخلی و خارجی.در این سطح داده هایی که واقعاً در یک پایگاه داده ها ذخیره شده است و ارتباطهای بین آنها توصیف می شود.

---

[۹] American National Standard Institute

- **سطح خارجی: سطحی است که به کاربران از همه نزدیک تر می باشد و تنها قسمتی از کل پایگاه داده ها است که مورد نظر کاربر خاص می باشد.**

با توجه به تعریف رکورد دانشجو از محیط عملیاتی دانشگاه هر یک از سطوح فوق را می توان به صورت زیر توصیف نمود:

```
Type Student=Record
    Name: char [۳۰];
    Major: char [۵];
    Address: char [٤۰];
    Avg: real;
End;
```

در سطح فیزیک رکورد Student می تواند بعنوان بلاکی از محلهای حافظه پیاپی توصیف شود.در سطح ادراکی ، رکورد بصورت تعریف نوع به صورت فوق بیان می شود.در سطح خارجی برای مثال برای پستچی که تنها نیاز به اسم و آدرس دارد، دیدی که شامل اسم و آدرس دانشجو است تعریف می شود و برای کارشناسان مسئول وام که نیازمند به اسم ، رشته و معدل دانشجو می باشند، دیدی که شامل تنها اسم ، رشته و معدل دانشجو می باشند، تعریف می شود.

معماری یک سیستم پایگاه داده در شکل ۲ به تصویر کشیده شده است.

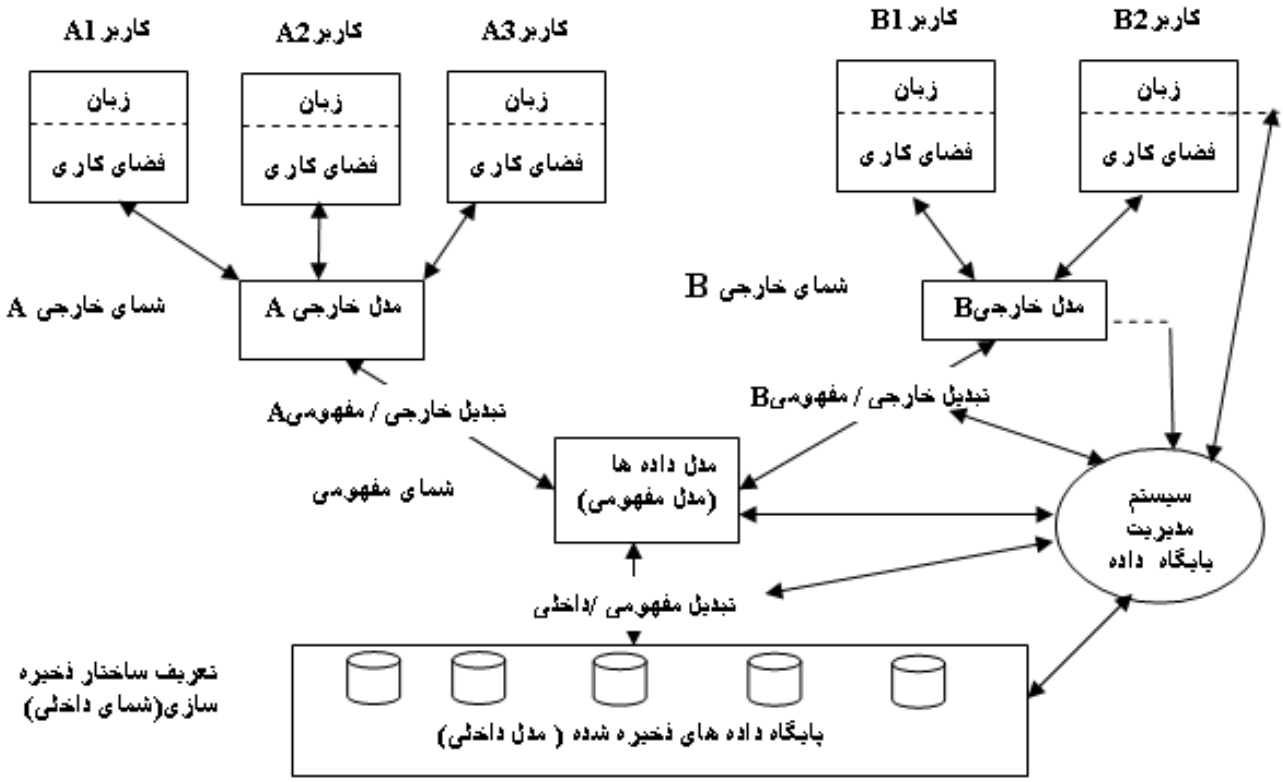
در لایه اول معماری ،کاربران یا برنامه نویسان کاربردی هستند یا کاربرانی هستند که از طریق پایانه های راه دور به سیستم دسترسی دارند و نیاز به بخشی از پایگاه داده دارند. هر کاربری از یک زبان میزبان [۱۰] استفاده می کند که می تواند یکی از زبانهای برنامه سازی سطح بالا باشد و از یک زبان فرعی داده ها [۱۱] که زیر مجموعه ای از زبان مرتبط با ذخیره و بازیابی اطلاعات در پایگاه داده می باشد و در زبان میزبان نهفته شده است و یا به صورت مستقل است.

برای هر کاربر فضای کاری وجود دارد که به عنوان یک ناحیه دریافت یا انتقال تمام داده ها که بین کاربر و پایگاه داده رد و بدل می شود عمل می کند.جهت یک برنامه نویس کاربردی این فضای کاری تنها یک ناحیه ورودی/ خروجی می باشد.جهت یک کاربر پایانه این فضای کاری ممکن است مرکب از حافظه کاری باشد که به پایانه یا شاید صفحه نمایش اختصاص داده شده باشد.همانگونه که بیان گردید یک کاربر منفرد بطور کلی تنها علاقمند به استفاده از بخشی از پایگاه داده ها می باشد.علاوه بر آن دید کاربر از آن بخش با مقایسه با روشی که در آن داده ها بطور فیزیکی ذخیره می شوند تا حدودی حالت انتزاعی دارد.

---

[۱۰] Host Language

[۱۱] DSL=Data Sublanguage



شکل ۲- معماری سیستم پایگاه داده ها

اجزای تشکیل دهنده معماری سیستم پایگاه داده ها

معماری سیستم پایگاه داده ها از اجزای زیر تشکیل می گردد.

- **زبان میزبان**
- **زبان فرعی داده ها**
- **مدل خارجی**
- **مدل مفهومی**
- **مدل داخلی**
- **تبدیل خارجی/ مفهومی**
- **تبدیل مفهومی / داخلی**
- **سیستم مدیریت پایگاه داده ها**
- **فرهنگ داده ها**

زبان میزبان

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

زبان برنامه نویسی DSL [۱۲]

یک زبان سطح بالاست (مثلاً زبان SQL)که معمولاً به صورت میهمان در کنار HL قرار می گیرد و به وسیله آن می توان:

الف) داده ها را تعریف کرد(DDL [۱۳])

ب) با داده ها کار کرد (DML[۱۴])

ج) داده ها را کنترل کرد (DCL [۱۵])

مدلها و شماها و تبدیلها

سطح خارجی

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

سطح مفهومی یا ادراکی

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

در این سطح DSL مفهومی وجود دارد یعنی طراح به وسیله DDL بانک را طراحی می کند و برای کنترل کردن داده ها از DCL استفاده می کند. در این سطح طراح بانک از چگونگی ذخیره سازی و محیط گرافیکی مستقل است .طراح باید مدل بانک یا همان ساختار داده را مشخص کند.مثلاً در مدل رابطه ای ساختار آن جدول است ، یعنی هر موجودیت را با یک جدول معرفی می کنیم ،مثل جدول اطلاعات دانشجو

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

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

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

---

[۱۲] زبان فرعی داده ها (Data Sublanguage)
[۱۳] Data Definition Language
[۱۴] Data Manipulation Language
[۱۵] Data Control Language

تبدیل ساختار:

1. **تبدیل خارجی / مفهومی**
2. **تبدیل مفهومی/ داخلی**

تبدیل خارجی / مفهومی: این تبدیل رابطه بین یک مدل خارجی خاص و مدل داده ها را تعریف می کند.

تبدیل مفهومی/ داخلی : رابطه بین مدل داده ها و پایگاه داده های ذخیره شده را تعریف می کند.

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

سیستم مدیریت پایگاه داده(DBMS)

سیستم مدیریت پایگاه داده نرم افزاری است که تمام دستیابیها به پایگاه داده ها را عملی می سازد.به عبارتی سیستم مدیریت پایگاه داده مجموعه ای از برنامه هایی است که واسط بین کاربران و امکانات سیستم کامپیوتر می باشد.از نقطه نظر مفهومی روی می دهد به شرح زیر است:

۱- یک کاربر با بهره گیری از زبان داده های خاصی مثل SQL درخواست دستیابی می نماید.

۲- DBMS درخواست را دریافت و تفسیر می کند.

۳- DBMS به نوبه خود شمای خارجی/ مفهومی، شمای مفهومی تبدیل مفهومی / داخلی و تعریف ساختاری ذخیره سازی را مورد بررسی قرار می دهد.

۴- DBMS عملیات مورد نیاز را بر روی پایگاه داده های ذخیره شده انجام می دهد.

مسئولیتهای مدیر پایگاه داده :

۱- اتخاذ تصمیم در مورد محتوای اطلاعاتی پایگاه داده ها

۲- اتخاذ تصمیم در مورد ساختار ذخیره سازی و راهبرد دستیابی

۳- عهده داری نقش رابط بین پایگاه داده ها و کاربران

۴- تعریف کنترلهای مجاز بودن و رویه های معتبر سازی

۵- تعریف یک راهبرد جهت پشتیبان و ترمیم

۶- نظارت بر عملکرد و پاسخگویی به تغییرات در نیازمندیها

تفاوت بین DA با DBA

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

مدیر بانک DBA : مدیر بانک اطلاعاتی یک شخص فنی است که مسئول پیاده سازی تصمیمات مدیر داده است.

مدل سازی داده ها

نمایشی از واقعیتها مدل یا الگو نامیده می شود.برای روشن شدن مفهوم مدل فرض کنید کارفرمایی در نظر دارد یک مجتمع مسکونی بنا نماید.بدین لحاظ از یک مهندس معمار درخواست می کند طرح و نقشه ساختمانی مورد نیاز را تهیه نماید.مهندس معمار با بازدید از محل احداث مجتمع و دریافت خواسته های کارفرما ، نقشه اجرایی کلی و تفصیلی ساختمان را طراحی و ترسیم می نماید.

این نقشه ها مدل منطقی ساختمان را تشکیل می دهند.هنگامی که مهندس معمار از طراحی مدل منطقی رضایت حاصل نمود، یک مدل سه بعدی کوچکی به نام ماکت می سازد و به کارفرما ارائه می نماید.این ماکت مدل فیزیکی ساختمان است.در راستای آماده سازی مدل فیزیکی ساختمان ممکن است مهندس معمار طی نشست های مختلف و بحث با کارفرما بارها به مدلهای منطقی خود مراجعه نماید تا مدل مطلوب و مورد نظر را تهیه نماید.زمانی که کارفرما از مدل رضایت نمود، پیاده کردن نقشه و احداث آن آغاز می شود.

گامهای طراحی پایگاه داده

در چرخه حیات طراحی پایگاه داده ها چهار مرحله متمایز به صورت زیر وجود دارند:

۱- تجزیه و تحلیل نیازمندیها

۲-طراحی مفهومی

۳-طراحی پیاده سازی

۴-طراحی فیزیکی

تجزیه و تحلیل نیازمندیها

طی این مرحله جامعیت و عملکرد مورد نظر کاربرد پایگاه داده ها مشخص می گردند. بدین ترتیب تجزیه و تحلیل نیازمندیها باید جنبه های دوگانه کاربرد مورد نظر را مورد توجه قرار دهد:

- **نیازمندیهای اطلاعاتی سازمانی که پایگاه داده ها برای آن طراحی و ساخته می شود.**

• نیازمندیهای پردازش داده ها یعنی نیازمندیهای برنامه هایی که با پایگاه داده ها در تعامل هستند.

این کار بطور معمول با انجام مصاحبه با کاربران نهایی مورد نظر و برنامه نویس سیستم که از پایگاه داده های تحت طراحی استفاده خواهند نمود انجام می گیرد.تهیه مستندی به زبان رسمی (یا نیمه رسمی) که بعنوان مبنایی جهت مراحل بعدی در فرایند طراحی پایگاه داده ها حائز اهمیت می باشد.

طراحی مفهومی

طراحی مفهومی مستقل از سیستم می باشد. پس از آنکه شمای مفهومی تکمیل گردید طراحی پیاده سازی یا طراحی منطقی می تواند انجام گیرد.در این گام ، طراحی شمای مفهومی به مدل پیاده سازی سیستم پایگاه داده های مورد استفاده تبدیل می گردد.مثلاً مدل رابطه ای،مدل شبکه ای، مدل شیء گرایی و...

بطور قابل ملاحظه ای متداول ترین مدل مفهومی مورد استفاده جهت طراحی پایگاه داده ها مدل ER یا مدل ارتباط موجودیتها[16] می باشد.

مفاهیم اساسی مدل ER

ویژگیهای ساختاری اساسی مدل ارتباط موجودیت به صورت زیر می باشد:

• **موجودیت ها**: نشانگر شیءهای فیزیکی یا انتزاعی در جهان واقعی می باشند و نوع موجودیت مفهوم کلی فرد، شیء یا پدیده است که می خواهیم درباره آن اطلاعاتی داشته باشیم.یک موجودیت یک شیء در جهان واقعی می باشد که از تمام شیء های دیگر متمایز است مثلاً هر فرد در یک شرکت موجودیت است.هر موجودیت مجموعه ای از خاصیتها را داراست و مقادیر مجموعه معینی از خاصیتها بطور منحصر به فرد آن فرد را متمایز می سازد.یک موجودیت ممکن است به صورت ذاتی باشد یعنی فضایی را اشغال می کند و قابل لمس است مانند یک شخص یا یک کتاب یا ممکن است انتزاعی باشد مانند وام یا تعطیلی

• **رابطه ها**: نشانگر رابطه های درونی ما بین موجودیتها در جهان واقعی هستند.

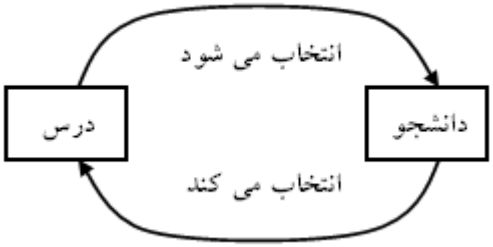
• **صفات خاصه:** ویژگیهای موجودیتها یا رابطه ها را شرح می دهند.به بیان دیگر صفت خاصه ویژگی جداساز یک نوع موجودیت از نوع دیگر یا عامل متمایز کننده رابطه ها از یکدیگر است.

نمودار ER(Entity relationship Diagram)

[16] Entity relationship model

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

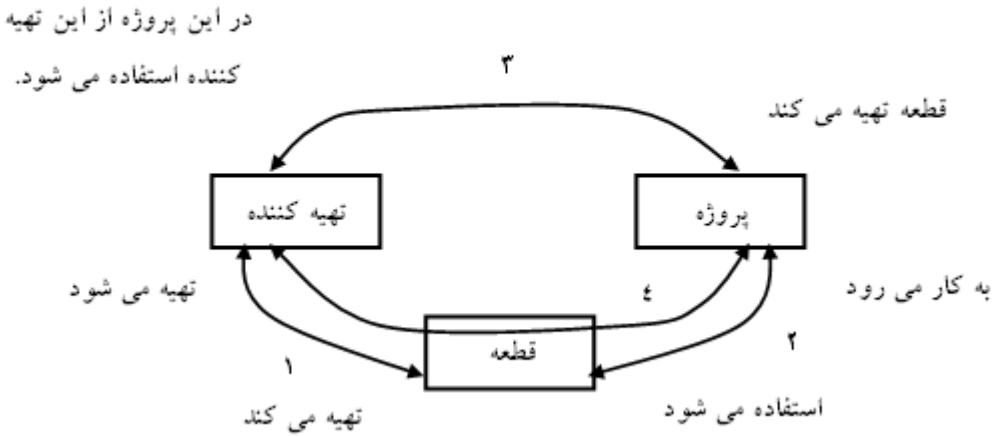
مثال۱:ارتباط دو موجودیت دانشجو و درس می تواند به صورت زیر باشد:



هر ارتباطی بین موجودیتها مفهوم یا سمانتیک(semantic) خاصی است.به بیان دیگر حـاوی بـار اطلاعاتی مخصوصی است که هر ارتباط دیگری فاقد آن است.سمانتیک مستتر در هر ارتبـاط بایـد بـه نحوی در بانک ذخیره شود.

تذکر:همیشه لزومی ندارد که یک ارتباط حتماً بین دو نوع موجودیت باشد ممکن است بین بـیش از دو موجودیت یک ارتباط وجود داشته باشد.

مثال۲: بین موجودیت های تهیه کننده (S)، قطعه (P) و پروژه (J) ارتباطات زیـر را مـی تـوان در نظـر گرفت:



توجه کنید از سه ارتباط ۱و۲و۳ همیشه نمی توان ارتباط ۴ را نتیجه گرفت به بیان دیگر سمانتیک موجود در سه ارتباط ۱و۲و۳ همان سمانتیک ۴ را تشکیل نمی دهد.

مثال۳: سمانتیک ۱: S۲ قطعه P۳ را تهیه کرده است.

سمانتیک ۲: قطعه P۳ در پروژه J۴ به کار رفته است.

سمانتیک ۳: S۲ برای پروژه J۴ قطعه تهیه کرده است.

از سه عبارت بالا نمی توان همیشه نتیجه گرفت که "s۲ قطعه p۳ را برای پروژه J۴ تهیه کرده است."به چنین حالاتی که گاه باعث بروز اشتباه در استنتاج اطلاع می شوند دام ارتباطی یا دام الحاقها (Connection trap) می گویند.

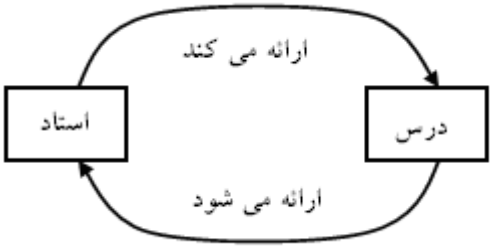
نکته:ممکن است یک موجودیت با خودش ارتباط داشته باشد.

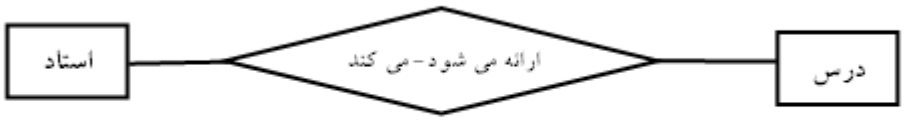
مثال ۴:

این بدین معناست که "یک قطعه از قطعه یا قطعات دیگر ساخته شده است."

تذکر:در کتابهای جدید ارتباط را داخل لوزی ترسیم می کنند.

مثال ۵:نمودار روبرو معادل نمودار زیر است:





تعداد نگاشتها

تعداد نگاشتها بیانگر تعداد موجودیتهایی است که موجودیت دیگری از طریق مجموعه رابطه ها می تواند با آنها پیوند برقرار کند.تعداد نگاشتها مفیدترین عامل محدود کننده جهت شرح مجموعه های رابطه های دوتایی است.جهت یک مجموعه رابطه دو تایی R بین مجموعه های موجودیت A وB تعداد نگاشتها باید به یکی از صورتهای زیر باشد:

۱- یک به یک ۲- یک به چند ۳- چند به یک ۴- چند به چند

- **یک به یک**

یک موجودیت درA با حداکثر یک موجودیت در B پیوند برقرار می کند و یک موجودیت در B با حداکثر یک موجودیت در A پیوند برقرار می کند.

- **یک به چند**

یک موجودیت درA با هر تعدادی موجودیت در B پیوند برقرار می کند .

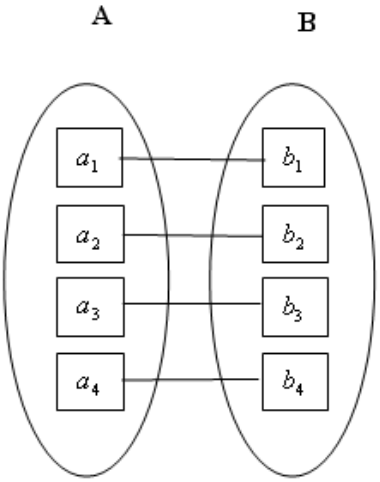
چند به یک

یک موجودیت درB با هر تعدادی موجودیت در A پیوند برقرار می کند.

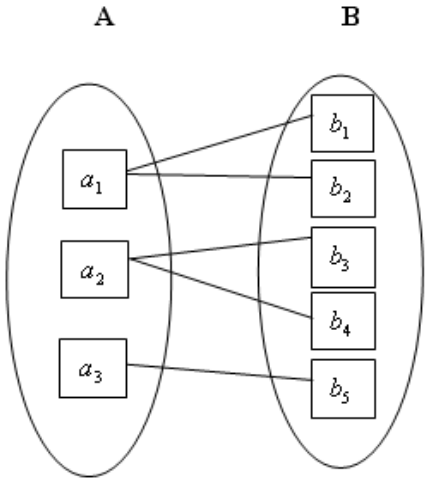
- **چند به چند**

یک موجودیت درA با هر تعدادی موجودیت در B و یک موجودیت درB با هر تعـدادی موجودیـت در A پیوند برقرار می کند.

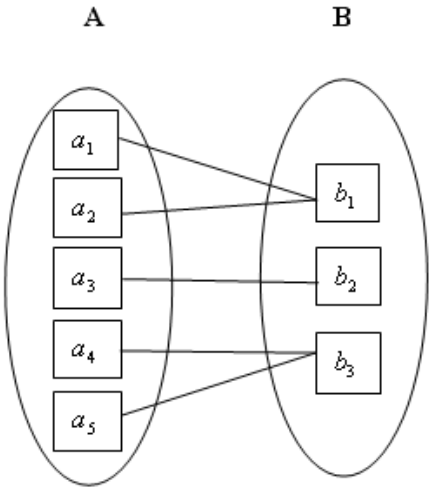
انواع رابطه در شکلهای زیر نشان داده می شود.



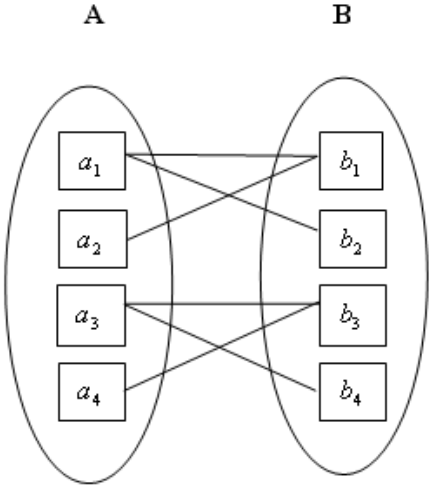
رابطه یک به یک



رابطه یک به چند



رابطه چند به یک



رابطه چند به چند

شکل ۳- انواع ارتباط

بدیهی است تعداد نگاشتهای مناسب جهت یک مجموعه رابطه های خاص بستگی به وضعیت جهان واقعی دارد که توسط مجموعه ها مدل سازی می شود.به عنوان مثال پس انداز کننده(depositor) را در یک رابطه یک به چند در نظر می گیریم به گونه ای که یک مشتری ممکن است چند حساب داشته باشد ولی هر حساب فقط به یک مشتری اختصاص می یابد.

نکته:در ترسیم نمودار در ترسیم نمودار ER درجه ارتباط (Relationship Degree) می تواند یک به یک(۱:۱)، یک به چند(۱: n) چند به یک (n:۱) یا چند به چند (n:n) باشد.

مثال ۶:ارتباط یک به یک

در این شکل هر استاد یک درس و هر درس فقط توسط یک استاد ارائه می شود البته ممکن است استادی اصلاً درس نداشته باشد یا درسی توسط هیچ استادی این ترم ارائه نگردد.

مثال ۷ : ارتباط چند به یک

در این شکل چند استاد ممکن است یک درس را ارائه کنند ولی هر استاد فقط یک درس را ارائه می کند.

مثال ۸ : ارتباط چند به چند

در این شکل هر درس ممکن است توسط چند استاد ارائه شود و هر استاد ممکن است چند درس مختلف را ارائه کند.

در شکلهای فوق شرکت موجودیتها در ارتباط اختیاری بود یعنی ممکن بود استادی درسی ارائه نکند و یا درسی این ترم ارائه نشود.در نمودار ER برای اینکه شرکت اجباری شود علامت به جای روی خط ارتباط ، داخل مستطیل موجودیت ترسیم می شود.

در این شکل هر اتاد باید درسی را ارائه کند (فقط یک درس) و هر درس فقط توسط یک اســتاد ارائه می شود (البته ممکن است درس خاصی ارائه نشود) ولی استادها نمی توانند درس ندهند.

نکته: ارتباط بین موجودیتها به تعبیری خود یک نوع موجودیت است زیرا با توجه به تعریف موجودیــت (پدیده ، شیء یا چیزی که می خواهیم در موردش اطلاع داشته باشیم) وجود ارتباط نیز پدیده ای است که باید در مورد آن اطلاعات در بانک داشته باشیم.

پس بانک اطلاعاتی به تعبیری مجموعه ای از اطلاعات در مورد موجودیتهای یک محیط عملیاتی و ارتباط بین آنها می باشد.

نمودار EER

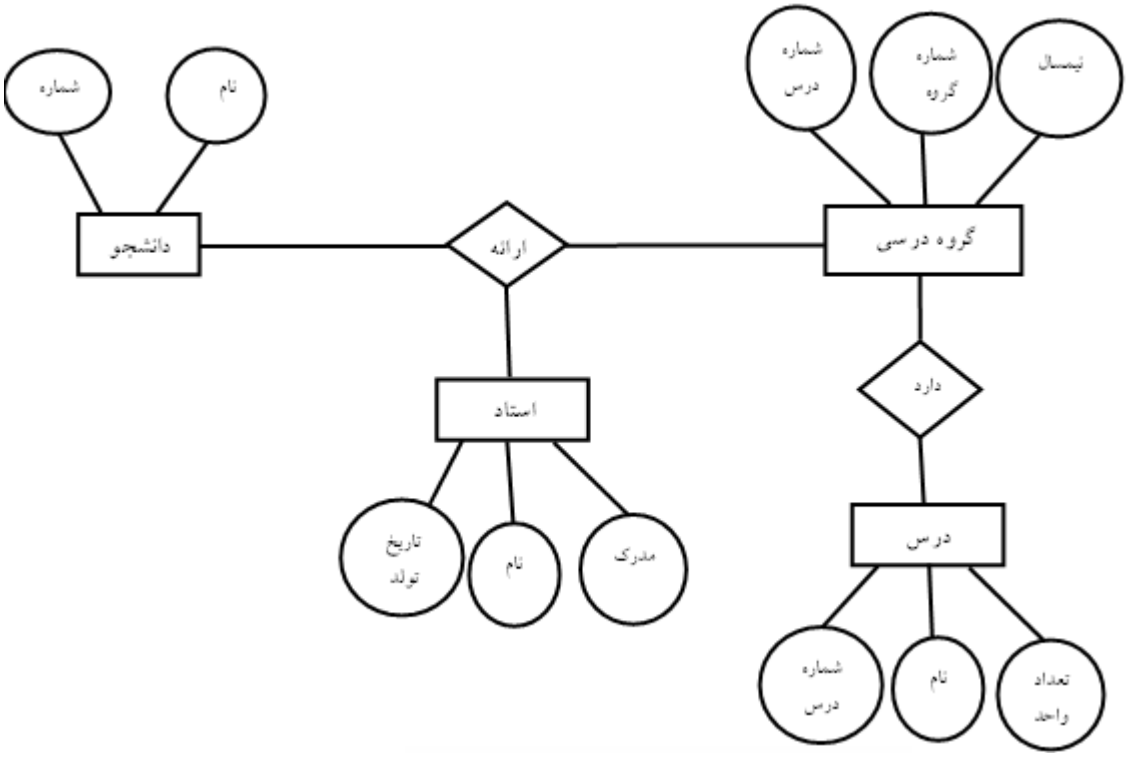
در سال ۱۹۷۶ چن(chen) از دانشگاه MIT مدل ER را جهت طراحی بانک پیشنهاد کرد.این مدل در طول زمان پیشرفت کرد و بنام Extended ER=EER معروف گردید.

چنین نموداری حاوی مؤلفه های زیر می باشد:

- **مستطیل ها که نشانگر مجموعه موجودیتها می باشند.**
- **بیضی ها که نشانگر صفات خاصه می باشند.**
- **لوزی ها که نشانگر مجموعه های رابطه ها می باشند.**

خط ها که صفات خاصه را به مجموعه موجودیتها و مجموعه ای از موجودیتها را به مجموعه های رابطه ها پیوند می دهند.

مثال ۱۱:



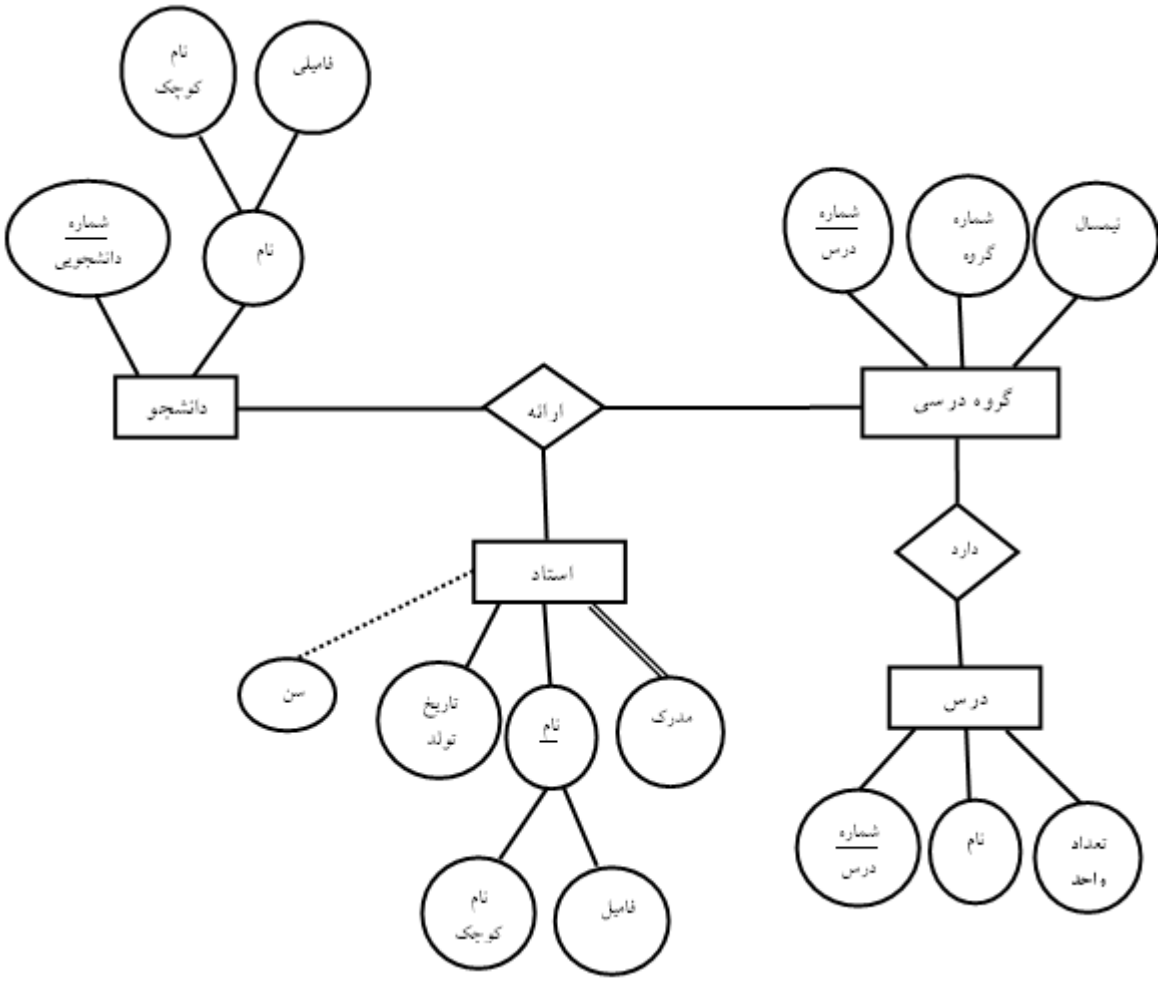
انواع صفت در نمودار EER

الف)صفت کلیدی:کلید عبارت است از یک یا چند صفت که در یک موجودیت منحصر به فرد باشد.مثلاً در موجودیت دانشجو شماره دانشجویی کلید است.چون هر دانشجو یک شماره یکتا دارد.ولی نام نمی تواند کلید باشد.گاهی اوقات یک صفت تنها نمی تواند کلید باشد بلکه مجموعه ای از دو یا چند صفت با همدیگر کلید می شوند.مثلاً نام و شماره شناسنامه هر یک به تنهایی کلید نیست ولی هر دو با هم کلید می شوند.برای مشخص کردن کلید یک موجودیت زیر آن صفت خط می کشیم.(....،نام پدر،نام،شماره) دانشجو

ب)صفت ساده و مرکب:بعضی از صفتها ساده هستند مثل شماره دانشجویی ولی بعضی از صفتها مرکب (تجزیه پذیر) هستند مثل آدرس که خود از صفتهای شهر،خیابان، کوچه و پلاک تشکیل یافته است.در واقع صفت مرکب صفتی است که هم خودش معنی دار است و هم بخشهایی از آن.در بانک اطلاعاتی رابطه ای (جدولی) صفت مرکب نداریم.

در نمودار EER صفات ترکیبی را در دو سطح می کشیم.

مثال ۱۲:



می توان اجزاء صفات مرکب را داخل پرانتز نوشت.

مثل: ((نام کوچک و فامیلی)نام و شماره دانشجویی) دانشجو

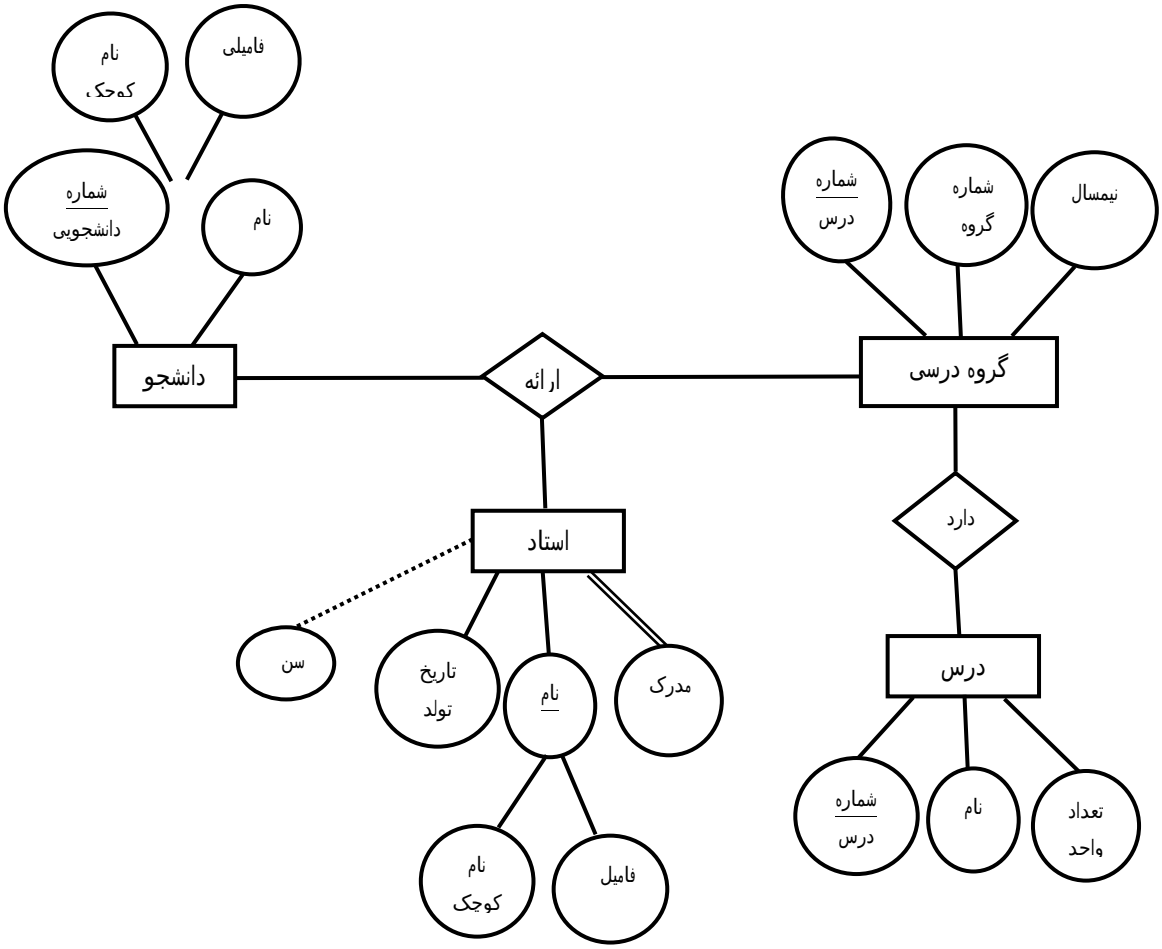
ج) صفت خاصه تک مقداری یا چند مقداری

مثلاً در موجودیت استاد نام تک مقداری است چون هر استاد فقط یک نام دارد ولی صفت مدرک چند مقداری است چون استاد ممکن است چندین مدرک داشته باشد.صفتهای چند مقداری را در مدل EER با دوخط ترسیم می کنیم.در مدل رابطه ای صفت چند مقداری نداریم.

د) صفت مشتق:صفت مشتق صفتی است که به کمک صفتهای دیگر می توان آن را محاسبه کرد.مثلاً سن استاد یک صفت مشتق است که با توجه به تاریخ تولد قابل محاسبه می باشد.تصمیم گیری در مورد صفت مشتق به عهده طراح است مثلاً معدل کل برای دانشجو بهتر است مشتق باشد زیرا مرتباً با گذراندن دروس بیشتر عوض می شود ولی برای فارغ التحصیل معدل کل بهتر است بخشی از پدیده باشد.

صفت مشتق در نمودار EER بصورت خط چین ترسیم می شود.در شکل مثال ۱۲ سن استاد صفت مشتق است.

مثال ۱۲:


نام
کوچک
فامیلی
شماره
دانشجویی
نام
دانشجو
ارائه
شماره
درس
شماره
گروه
نیمسال
گروه درسی
دارد
استاد
سن
تاریخ
تولد
نام
مدرک
نام
کوچک
فامیل
درس
شماره
درس
نام
تعداد
واحد

مدل داده منطقی که از مدل داده ادراکی منتج می شود ، به تعریف و تشریح موجودیتها و روابط بین آنها در چارچوب سیستم مدیریت پایگاه داده ها می پردازد.

تعریف و تشریح پایگاه داده ها برای اجرایی خاص ، در یک سیستم مدیریت پایگاه داده ها به وسیله زبان فرعی داده ها "شما" گفته می شود.

به عبارت دیگر "شما" دید منطقی و کامل از داده هاست و اگر قسمتی از داده ها تعریف شوند "زیر شما" نامیده می شود.

در مدل داده منطقی، مدلهای گوناگونی وجود دارد که در حال حاضر مدل داده سلسله مراتبی ، مدل داده شبکه ای و مدل داده رابطه ای رایج تر است.

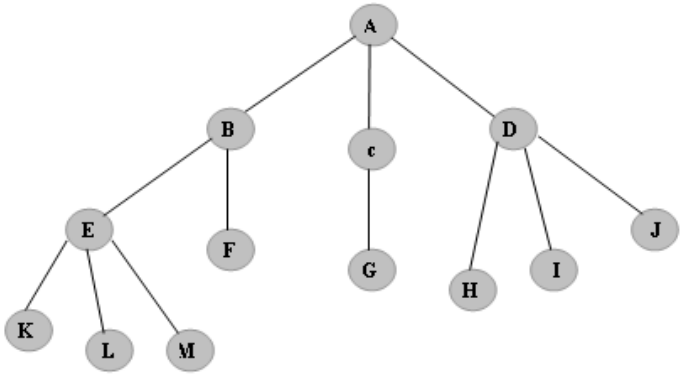
مدلهای بانک اطلاعاتی

به ساختار داده ای که توسط طراح بانک در سطح ادراکی تعیین می شود مدل بانک گفته می شود به طور کلی ۴ مدل وجود دارد .

۱ـ ساختار سلسله مراتبی

این ساختار قدیمی داده ها و ارتباط بین آنها را به کمک یک درخت نمایش می دهد . روش سلسله مراتبی برای ارتباطات یک به چند بین موجودیتها مناسب است .هر گره از درختواره که نمایشگر یک موجودیت است بوسیله رکورد و پیوند دو گره که ارتباط بین دو موجودیت را نشان می دهد بوسیله پیوند یا به وسیله مسیر سلسله مراتبی که پیمایش پیش ترتیب درختواره می باشد ، پیاده سازی می گردد. مسیر سلسله مراتبی به مسیری که از ریشه شروع و از چپ ترین حرکت آغاز شود ، گفته می شود.

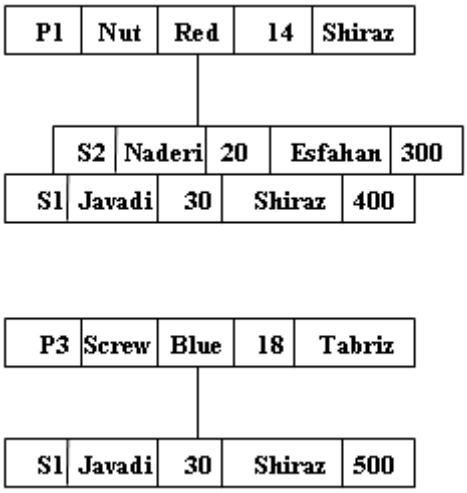
به عبارت دیگر پایگاه داده های سلسله مراتبی ، مجموعه ای از رکوردها هستند و به گونه ای مرتب شده اند که مانند یک ساختار درختواره مشاهده می شوند.



شکل ۵- ساختار سلسله مراتبی

در این ساختار هر پدر می تواند چندین فرزند داشته باشد ، اما هر فرزند می تواند تنها یک پدر داشته باشد.

شکل زیر یک مدل سلسله مراتبی پایگاه داده های تهیه کنندگان و قطعات را نشان می دهد.



شکل ۶- مثال ساختار سلسله مراتبی

ویژگیهای ساختار سلسله مراتبی

۱- از دید کاربر وضوح دارد ولی وضوح آن به حد مدل رابطه ای نیست.

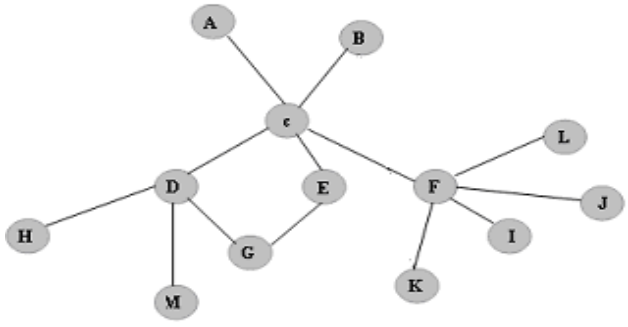
۲- ارتباط بین داده ها به کمک درخت واره نشان داده می شود.بنابراین برای محیطهای ارتباط طبیعی یک به چند مناسب است.

۳- عملگر بازیابی به سادگی عملگرهای مدل رابطه ای نیستند و در عمل ذخیره سازی و حذف دارای مشکل می باشند.

۴- از مبانی تئوریک ریاضی همانند مدل رابطه ای برخوردار نیست.

۲ـ مدل شبکه ای

در این ساختار که به آن ساختار پلکس (Plax)گفته می شود برای نمایش داده ها و ارتباط بین آنها از گراف استفاده می کنند. این مدل برای نمایش ارتباطات چند به چند بسیار مناسب است .در ساختار شبکه ای موجودیتها به کمک انواع رکوردها و ارتباطات به کمک پیوندهای بین رکوردها نمایش داده می شود.این ساختار تکامل یافته ساختار سلسله مراتبی است.در اینجا هر فرزند می تواند بیش از یک پدر داشته باشد.



شکل ۸- مدل شبکه ای

خواص مدل شبکه ای

۱- از دید کاربر وضوح ندارد.

۲- ارتباط یک به چند دوسویه بین دو نوع موجودیت وجود دارد.

۳- عملگر بازیابی پیچیده تر از مدل سلسله مراتبی است.

۴- این ساختار برای محیطهای دارای ارتباط یک به چند دو سویه مدل مناسبی است.

۳– مدل شیء گرایی

در این روش برای هر موجودیت یک کلاس یا طبقه ایجاد می شود که هر کلاس دارای خصوصیت و ویژگیهای خاصی است سپس عناصری را که قرار است ذخیره شوند عضو کلاس ها قرار می دهیم که در این صورت ارث بری از کلاسهای مختلف به وجود می آید .

۴ـ مدل رابطه ای

این مدل طراحی موجودیتها را به صورت جدول نمایش می دهد. بنابراین بانک تشکیل شده از مجموعه جداول. برای برقراری ارتباط بین جداول ، جداول دیگری را طراحی می کنیم(رابطه ها همان جداول هستند). در این مدل برای بازیابی اطلاعات فقط به عملگر سطریاب نیاز است . هر جدول ماتریسی است که شامل تعدادی ردیف و ستون می باشد.هر ردیف یک نمود موجودیت و هر ستون یکی از ویژگیهای موجودیت است.در واقع یک جدول شامل مجموعه ای از نمود یک موجودیت است که از این بین دیدگاه همان فایل می باشد.

ویژگیهای مدل رابطه ای

۱ـ از دید کاربر دارای وضوح است .

۲- داده ها و ارتباطات بین آنها با مکانیزم واحد نشان داده می شود.

۳- عملگر بازیابی نسبتاً ساده است.

۴- از مبانی تئوریک ریاضی برخوردار است.

۵- برای طراحی رابطه ها به صورت مطلوب دارای ابزار تئوریک است.

شکل ۷ یک مدل رابطه ای پایگاه داده های تهیه کنندگان و قطعات را نشان می دهد.

S

S#	SNAME	STATUS	CITY
S1	Javadi	30	Shiraz
S2	Naderi	20	Esfahan
S3	Pooya	10	Esfahan

SP

S#	P#	QTY
S1	P1	400
S1	P2	300
S1	P3	500
S2	P1	300
S2	P2	400
S3	P2	200

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	14	Shiraz
P2	Bolt	Green	18	Esfahan
P3	Screw	Blue	18	Tabriz
P4	Screw	Red	15	Shiraz

شکل ۷- مثال مدل رابطه ای

داده ها در سه جدول S (تهیه کنندگان)،P(قطعات) و SP (حمل یا ارسال کالا) سازماندهی شده اند.

جدول S برای هر تهیه کننده یا تولید کننده کالا شامل شماره تهیه کننده (S#)، نام خانوادگی تهیه کننده (SNAME)،کد وضعیت(STATUS) و نام شهر(CITY) محل کار است.

جدول P برای هر قطعه شامل شماره تهیه کننده (S#)،شماره قطعه (P#)،نام قطعه(PNAME)، رنگ (COLOR)، وزن(WEIGHT) و نام شهر (CITY) محل نگهداری قطعه است.

جدول SP برای هر بار ارسال کالا شامل شماره تهیه کننده (S#)، شماره قطعه (P#) و مقدار کالای فرستاده شده (QTY)است.فرض بر این است که وضعیتها به گونه ای است که در هر زمان جهت یک ترکیب معینی از تهیه کننده/قطعه بیش از یک ارسال کالا وجود دارد.

هر یک از سه جدول مشابه یک فایل ترتیبی هستند که در آن سطرها متناظر با رکوردهای فایل و ستونها متناظر با فیلدهای رکورد می باشند.به عنوان مثال جدول P از چهار رکورد که هر کدام پنج فیلد دارند تشکیل یافته است.دامنه مشترک جداول یا رابطه های S و SP شماره های تهیه کنندگان است و دامنه مشترک P و SP شماره های قطعه است.همچنین دامنه های مشترک P و S نامهای شهرهاست.

محاسن

۱- تأمین اشتراک داده ها

در پایگاه داده های رابطه ای ، مانند پایگاه داده های شبکه ای و سلسله مراتبی ، تنها یک مخزن داده ها وجود دارد.

۲- تأمین استقلال داده ای و ساختاری

مدل داده رابطه ای این فرصت را فراهم می سازد که خصوصیات فیزیکی نحوه ذخیره داده ها را کاملاً فراموش کنیم و تنها آنچه را مشاهده می نماییم، متمرکز شویم.

۳– تأمین زبان پرس و جوی کاربر موردی

یکی از دلایل اختصاص بازار به خود در مدل داده رابطه ای وجود امکانات پرس و جوی بسیار قوی و انعطاف پذیر می باشد.در بیشتر پایگاه داده های رابطه ای ، زبان پرس و جو، زبان پرس و جوی ساختیافته (SQL) می باشد.

۴– تأمین واسطه های دوست کاربر

هر پایگاه داده رابطه ای مجهز به SQL دارای سه بخش می باشد:واسط، مجموعه جداول در پایگاه داده ها و موتور.واسط ممکن است شامل منوها،اعمال پرس و جویی،تولید کنندگان گزارشات وغیره باشد.در حالت کلی ، واسط به کاربر اجازه می دهد با داده ها ارتباط برقرار نماید.

۵– مدیریت و نگهداری داده ها در سیستم پایگاه داده ها

مهمترین کار پایگاه داده ها ، نگهداری سیستم می باشد که RDBMS آن را بدون اطلاع کاربران انجام می دهد

معایب

چون سیستم مدیریت پایگاه داده های رابطه ای، پیچیدگیهای ساختار فیزیکی سیستم را از دید طرحان و کاربران مخفی می نماید، پس نیاز به کامپیوتری با قدرت بالا برای اجرای کلیه کارهای محول شده به RDBMS دارد.در نتیجه سیستم مدیریت پایگاه داده های رابطه ای کند تر از دیگ سیستمها می باشد.

مدل رابطه ای و تعاریف مربوط به آن

سیستم مدیریت پایگاه داده های رابطه ای بر اساس مدل داده رابطه ای که دکتر کاد در سال ۱۹۷۰ بر مبنای نظریه روابط در ریاضی تعریف نمود،پایه گذاری شده است. در حال حاضر متداولترین مدل پایگاه داده ها مدل رابطه ای می باشد.سیستم های مدیریت پایگاه داده ها که مبتنی بر مدل رابطه ای داده ها هستند نقش برجسته ای را در بازار جهانی نرم افزار بازی می کنند.

لذا یک پایگاه داده رابطه ای مجموعه ای از روابط است که هر رابطه به صورت یک جدول نمایش داده می شود.هر جدول شامل نمودهای یک موجودیت می باشد و دارای ستون ها و ردیف هایی است

که به ترتیب ویژگی ها و تاپل های یک رابطه نامیده می شوند.هر ستون یک برچسب دارد که نام ویژگی گفته می شود.هر رابطه دارای یک نام منحصر به فرد می باشد.

تعریف رابطه

تعریف رابطه همان تعریف ریاضی آن می باشد که دارای دامنه و برد است و چند تایی هایی مرتب را شامل می شود . به عبارتی رابطه از دو قسمت عنوان و پیکر تشکیل می شود.

هر جدول شامل سه دسته اطلاعات می باشد:

۱- مجموعه نام ستون ها یا ویژگی ها که عنوان یا Header جدول نامیده می شود.

عنوان[۱۷] مجموعه ای است ثابت از اسامی صفات خاصه $A_i \ldots\ldots, A_3, A_2, A_1$ به نحوی که صفت خاصه A_i دقیقاً متناظر با میدان D_i باشد.(در حقیقت همان عنوان فیلدها می باشد.)

۲- مجموعه مقادیر یک ستون یا یک ویژگی که دامنه گفته می شود.

میدان مجموعه ای است که مقادیر یک صفت خاصه از آن گرفته می شود .

$$D_{stn} = \{S_1, S_2, S_3, S_4\}$$

۳- مجموعه ردیف ها یا تاپل های جدول که به بدنه[۱۸] یا بسط جدول معروف می باشد.

مجموعه ای است متغیر در زمان از تاپل ها (تاپل=چند تایی).در واقع همان مجموعه رکوردهایی است که در بانک ذخیره می شوند.

معمولاًHeading ثابت، اما body متغیر می باشد.

یک بانک اطلاعاتی از دید کاربر یک جدول است.

دید رابطه ای	دید کاربر
رابطه	جدول
تاپل	سطر
صفت خاصه	ستون
میدان	مقادیر مجاز یک ستون

---

[۱۷] Heading
[۱۸] Body

نقش میدان در پایگاههای اطلاعاتی

۱ـ کنترل مقداری پرس و جوها

مقادیر یک صفت خاصه در طول حیات رابطه از مقادیر میدان گرفته می شود. به عبارت دیگر باید در میدان وجود داشته باشد (قاعده جامعیت خاص) مثلاً : این پرسش که دانش آموزانی را استخراج کنید که نمره آنها ۲۱ است اشتباه است .

۲ـ کنترل سمانتیک پرس و جوها

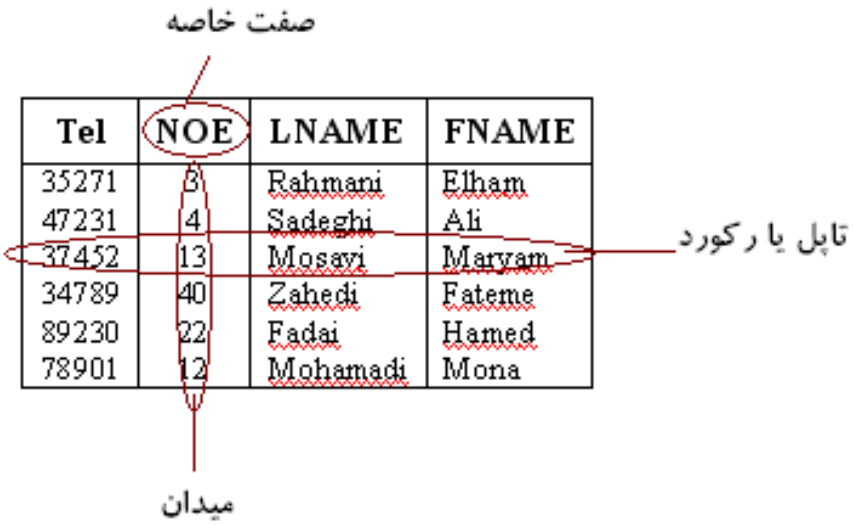
ممکن است دو صفت خاصه از یک نوع باشند (مثلا عددی باشد) اما هر کدام ماهیت متفاوت دارد مانند شماره دانشجویی و شماره شناسنامه این دو را نمی توان با هم مقایسه کرد مثلا سؤال زیر فاقد سمانتیک است .

دانشجویی را استخراج کنید که شماره دانشجویی آنها کوچکتر از شماره شناسنامه آنها باشد ؟ که این سؤال معنا ندارد .

۳ـ تسهیل در پاسخگویی بعضی از پرس و جوها

در بعضی از پرس و جوها سؤال در مورد خود بانک و رابطه ها است مانند اینکه در کدام جدول اطلاعاتی در مورد دانشجویان وجود دارد.

مثال:جدول Studentرا در نظر بگیرید.این جدول با ۶ ردیف و چهار ستون می باشد.هر ردیف نمودی از یک دانشجوی خاص است که نام،نام خانوادگی،شماره دانشجویی و شماره تلفن را در چهار ستون به نامهای Tel,NOE,Lname وFname توصیف می نماید.



Tel	NOE	LNAME	FNAME
35271	3	Rahmani	Elham
47231	4	Sadeghi	Ali
37452	13	Mosavi	Maryam
34789	40	Zahedi	Fateme
89230	22	Fadai	Hamed
78901	12	Mohamadi	Mona

چون هر رابطه از مجموعه عنوان و مجموعه بسط تشکیل شده است .لذا طبق قوانین نظریه رابطه ها در ریاضی خصوصیات زیر را برای یک رابطه می توان تعریف نمود.

- **کاردینالیتی رابطه.**تعداد تاپل های رابطه در یک لحظه از حیات آن ، کاردینالیتی رابطه نام دارد. کاردینالیتی رابطه، در طول حیات رابطه متغیر است. کاردینالیتی رابطه **Student** ، ۶ است.
- **درجه رابطه**.تعداد ویژگیهای ،رابطه درجه رابطه نام دارد.به عبارت دیگر کاردینالیتی مجموعه شما یا عنوان ، همان درجه رابطه است.درجه رابطه **Student** چهار است.
- **عدم تکرار در رابطه** .چون بسط رابطه، یک مجموعه است ، عناصر تکراری نمی تواند داشته باشد.
- **فاقد نظم در تاپلهای یک رابطه**.چون عناصر یک مجموعه طبق تعریف در ریاضی فاقد نظم می باشند، پس هیچ لزومی ندارد که تاپلها بر اساس یک صفت خاصه مرتب باشند.
- **فاقد نظم در ویژگیهای یک رابطه**.عنوان رابطه یک مجموعه طبق تعریف ریاضی فاقد نظم است.بنابراین **Student(Tel,noe,Fname,Lname)** با**Student(Tel,noe,Lname, Fname)** برابر است.
- **تجزیه ناپذیر بودن مقادیر هر ویژگی رابطه**.همه مقادیر صفت خاصه **Ai** تجزیه ناپذیر(اتومیک) هستند منظور از تجزیه ناپذیری این است که هرگز به دو اطلاعات تقسیم نشود . یعنی در تلاقی هر سطر و ستون جدول دقیقاً یک نقدار نه مجموعه ای از مقادیر وجود دارد.مانند : تاریخ تولد که روز ، ماه و سال در یک ستون ذخیره می شود.

کلیدها و قواعد جامعیت داده ای

مشاهده نمودید که هر موجودیت از شمای ادراکی بوسیله یک جدول در مدل داده رابطه ای پیاده سازی می شود و پیاده سازی ارتباط بین جدولها بوسیله تکرار یک ویژگی از یک جدول در جدول دیگر انجام می شود.

کلید وسیله ای است که به ما کمک می کند روابط بین موجودیتها را تعریف کنیم.

توانایی در تعیین اینکه چگونه موجودیتهای درون یک مجموعه موجودیت معینی و رابطه های درون یک مجموعه رابطه معینی متمایز هستند حائز اهمیت می باشد.بطور مفهومی موجودیتها و رابطه های منفرد متمایز هستند.با این وجود از نظر تصویری از پایگاه داده ها اختلاف بین آنها باید بر حسب صفات خاصه آنها بیان گردد.مفهوم کلید به ما اجازه چنین تمایزاتی را می دهد.

۱– کلید کاندید[۱۹]

به مجموعه ای از صفت خاصه های **(Ai,Aj, An)** یا زیرمجموعه ای از ستونها در رابطه یا جدول که دو خاصیت زیر را داشته باشند کلید کاندید گفته می شود .

الف) منحصر بودن (یکتایی[۲۰])

---

[۱۹] condidate key

[۲۰] uniquenese

منحصر بودن به این معنا که در هر لحظه از حیات رابطه یا جدول صفت خاصه های (Ai,Aj, An) باید یکتا باشد و هیچ دوتاپلی تکرار نشود .

ب) کاهش ناپذیری[۲۱]

یعنی اگر یکی از عناصر کلید کاندید مثلاً Aj را حذف کنیم منحصر بودن خود را از دست بدهد همچنین نباید فیلد اضافه داشته باشد.

نکته: هر رابطه باید حداقل یک کلید کاندید داشته باشد.

رابطه تمام کلید

در بدترین حالت ممکنه همه صفت خاصه یک رابطه به عنوان کلید می شوند به چنین رابطه ای All-key می گویند.

۲ـ کلید اصلی[۲۲]

کلید اصلی کلید کاندیدی است که توسط طراح بانک انتخاب می شود. طراح بانک باید از بین کلیدهای کاندید فقط یکی را برای کلید اصلی معرفی کند دو عامل برای این انتخاب مهم است :

الف) نقش واهمیت کلید اصلی نسبت به سایرکلیدهای کاندید مثلا : شماره نظام پزشکی در تشخیص پزشکان . کلید انتخاب شده نسبت به سایر کلیدهای کاندید پاسخگوی نیازها باشد.

ب) کوتاهترین کلید کاندید را به عنوان کلید اصلی انتخاب می کنند. (نه از لحاظ تعداد صفات خاصه بلکه از نظر طول رشته بایتی)

۳ـ کلید بدیل یا فرعی[۲۳]

هر کلید کاندید به غیر از کلید اصلی را می گویند .

مورد استفاده کلید فرعی

---

[۲۱] minimality
[۲۲] Primary key
[۲۳] Alternate key

اگر برای یک بانک کلید اصلی شماره دانشجویی باشد و از ما بخواهند لیست مرتب شده بر اساس نام خانوادگی را نمایش دهید از کلید اصلی استفاده نمی کنیم و می توان یک کلید فرعی دیگر استفاده کرد تا بر اساس نام خانوادگی مرتب شوند.

۴ـ کلید خارجی[۲۴]

کلیدی است که در یک رابطه کلید اصلی و در رابطه دیگری کلید فرعی است و برای ارتباط بین دو رابطه به کار می رود.

صفت خاصه x از رابطه R۲ کلید خارجی است اگر x در رابطه R۱ کلید اصلی باشد .

تکلیف : برای کتاب خانه یک بانک طراحی کنید که شامل مشخصات کتاب ها(نام ، عنوان،ناشر،سال انتشار و...)، اعضای کتابخانه(نام،شماره دانشجویی،رشته و...)،کارمندان کتابخانه (نام،کد پرسنلی،نوع همکاری و...)،امانت کتاب (تاریخ امانت، تاریخ برگشت و...)باشد.کلید اصلی را در هر رابطه مشخص نمایید.

جامعیت بخشی به پایگاه داده ها

در مدل رابطه ای پایگاه داده ها برای تضمین بی نقصی پایگاه داده ها و جامعیت بخشیدن به آن باید قواعدی حاکم باشند.این قواعد به دو گروه خاص و عام طبقه بندی می گردند.

قواعد جامعیت خاص

قواعد خاص قواعدی هستند که در یک سیستم مشخصی پیش بینی و حاکم می گردند. می توان گفت این قواعد همان مبحث میدان می باشد .به عنوان مثال اگر جدول تهیه کنندگان قطعات و ارسال کنندگان آنها را در نظر بگیرید چنانچه یک محدود کننده جامعیتی به عنوان قاعده برای QTY (مقدار کالای فرستاده شده) جدول SP به صورت زیر تعریف شود:

"مقدار QTY همیشه بزرگتر از صفر و کمتر از دو هزار می باشد."وجود چنین قاعده ای در بانک ایجاب می کند که سیستم از انجام هر گونه عملی که سبب شود مقدار QTY خارج از این طیف بشود،جلوگیری کند.

قواعد عام

قواعد عام، حاکم بر پایگاه داده های رابطه ای می باشند مشروط بر اینکه DBMS انتخاب شده قادر به اعمال چنین قواعدی باشد.در سیستم پایگاه داده های رابطه ای دو قاعده جامعیت به نامهای قاعده جامعیت موجودیتی و قاعده جامعیت ارجاعی حاکم می باشند.

---

[۲۴] foreign key

قواعد جامعیت عام موجودیتی

این قاعده یعنی این که هیچ بخشی از کلید اصلی نباید دارای مقدار تهی باشد. دلیل این قاعده این است که کلید اصلی نقش یک نمونه ای از موجودیت را دارد و هر نمونه از موجودیت را از روی شناسه اش که کلید اصلی است می شناسند اگر این شناسه تهی باشد به این معنا است که این نمونه از موجودیت وجود ندارد و تضاد به وجود می آید.

قواعد جامعیت عام ارجاعی

حاکم نمودن این قاعده بر پایگاه داده ها برای این است که غالباً می خواهیم اطمینان حاصل نماییم که یک مقدار که جهت مجموعه معینی از صفات خاصه در یک رابطه ظاهر می شود برای مجموعه معینی از صفات خاصه در رابطه دیگری نیز ظاهر می شود.این قاعده ناظر بر کلید خارجی است.دو رابطه R_1 و R_2 را در نظر بگیرید و A_i صفت خاصه ای از R_1 باشد که در آن رابطه نقش کلید اصلی را بر عهده دارد و این صفت خاصه کلید خارجی رابطه R_2 نیز باشد.مقدار این صفت خاصه در تاپلی از رابطه R_2 می تواند به صورت تهی باشد لیکن در صورت داشتن مقدار باید این مقدار در رابطه R_1 وجود داشته باشد.

طراحی پایگاه داده های رابطه ای

هدف نهایی از سیستمهای پایگاه داده ها آن است که طراحی و ساخت نرم افزارهای کـاربردی را آسان تر ، ارزان تر، سریع تر و قابل انعطاف تر سازد. سیستم پایگـاه داده هـا مخزنـی از داده هـا را در اختیار دارد که برای داده پردازی یک سازمان مورد نیاز است. داده ها باید دقیق، اختصاصی و محفوظ از خسارت باشند.باید به گونه ای سازماندهی شوند که نرم افزارهای کاربردی گوناگون با نیازمندیهای داده ای مختلف بتوانند داده ها را به کار گیرند.

گامهای نه گانه فرآیند طراحی پایگاه داده های رابطه ای

۱- تشخیص مبنا جهت نیازمندیهای پایگاه داده ها
۲- تعریف نیازمندیهای وظیفه مندی و عملکردی پایگاه داده ها
۳- تشخیص فقره های داده ها
۴- جداسازی عناصر داده ها از فایل ها
۵- ساخت فرهنگ داده ها
۶- جمع آوری عناصر داده ها در فایل ها
۷- تشخیص ویژگیهای بازیابی هر فایل
۸- تشخیص رابطه های بین فایل ها
۹- طراحی و ساخت شما برای سیستم مدیریت پایگاه داده های مورد استفاده

همیشه یک گام دهمی وجود دارد و آن تکرار مجدد نه گام یاد شده تا حصول اطمینـان نتیجـه نهـایی مطلوب است.

۱– تشخیص مبنا جهت نیازمندیهای پایگاه داده ها

هنگامی که آماده طراحی یک پایگاه داده ها می باشیم باید مأموریتی به ما واگذار شده باشـد و دارای هدفی باشیم.شخصی درخواست نموده است که چیـزی را بـه صـورت خودکـار یـا مکـانیزه در آوریم.پیوسته پرسشهایی در این زمینه مطرح می گردد.درخواست چگونه مطرح گردید؟هنگامی که راه حل مسئله در خواست گردید تا چه حد موضوع روشن شده بود؟آیا اینک مسئله حل شده است؟آیا راه حل موجود به صورت خودکار یا مکانیزه در آمده است؟آیا هرگز شخص دیگری در یک محیط متفـاوت مسئله مشابهی را حل نموده است؟

مبنا جهت پایگاه داده ها مسئله ای را که باید حل شود،موجود بودن منابعی که در حـل مسـئله می توانند مورد استفاده قرار گیرند و رویکردهای احتمالی به حل مسئله را مشـخص مـی سـازد.بـرای

تعریف مبنا با راه حل موجود کار را آغاز می کنیم.اگر راه حل موجود وجود نـدارد کـار را بایـد از هـیچ شروع کرد.

مثلاً در یک پایگاه داده پرسنلی ،مشخصات می تواند به صورت زیر باشد:

مسئله:ردگیری وضعیت کارکنان درون بخشهایی که کار می کنند از آن جمله حقوق های آنان و تاریخی که استخدام شده اند،گزارش وضعیت پـروژه هـای اختصـاص داده شـده بـه بخشـها و ثبـت مشخصات کارکنانی که بر روی پروژه ها کار می کنند.

لیست منابع موجود حاوی اطلاعاتی نظیر مستندات منبع(مثلاً کارتهای زمانی کارکنان)،تعامل با سایر پایگاه داده ها ، پرسنل و غیره می باشد

۲– تعریف نیازمندیهای وظیفه مندی و عملکردی پایگاه داده ها

گام بعدی پالایشی از گام نخستین است.بر اساس مبنا ، نیازمندیهای پایگاه داده ها را تعریف می کنیم.لازم است نیازمندیهای وظیفه مندی و نیازمندیهای عملکردی به تفکیک تعیین گردند.

نیازمندیهای وظیفه مندی بیانگر نوع داده هایی است که پایگاه داده ها در بر خواهـد داشـت.در این نیازمندیها باید هر چیزی را که در مورد وظایفی که باید پشتیبانی شوند می دانیم مسـتند سـازی کنیم و در تشخیص اطلاعاتی که پایگاه داده ها باید حاوی آن باشد ژرف نگری داشته باشیم.

نیازمندیهای عملکردی،بسامدها،سرعتها،مقادیر و اندازه ها را (چند وقت به چند وقت،تا چقـدر سریع،چند تا) که پایگاه داده ها باید پشتیبانی نماید شرح می دهد.

به عنوان مثال در یک سیستم پرسنلی برخی از نیازمندیها به صورت زیر خواهد بود:

نیازمندیهای وظیفه مندی

۱- سیستم با استفاده از شماره و نام بخش ، داده های مربوط به بخشهای درون سـازمان را ثبـت و گزارش خواهد کرد.

۲- سیستم با استفاده از شماره کارمندی، داده های مربوط به کارمندان را ثبـت و گـزارش خواهـد کرد. رکورد هر کارمند حاوی نام کارمند، حقوق و تاریخ استخدام خواهد بود.

۳- سیستم با استفاده از شماره پروژه، داده های مربوط به سازمان را ثبت و گزارش خواهد کرد.هـر رکورد مربوط به پروژه حاوی نام پروژه ، تاریخ شروع و پایان پروژه و تعـدا سـاعات کـاری کـه بودجه صرف آن شده است می باشد.

نیازمندیهای عملکردی

۱- سیستم تا ده بخش را پشتیبانی خواهد کرد.

۲- سیستم تا صد کارمند پشتیبانی خواهد کرد.

۳- بازیابی داده های ذخیره شده در مورد یک کارمند،پروژه یا بخش به صورت On line خواهد بود و در پاسخ به وارد کردن شماره کارمندی، شماره پروژه یا شماره بخش توسط کاربر خواهد بود. سیستم در ظرف سه ثانیه از ورود شماره کنترلی مرتبط داده های مربوطه را منتقل خواهد کرد.

۳- تشخیص فقره های داده ها

بعد از اینکه نیازمندیهای پایگاه داده ها نوشته شدند می توان شروع به تبدیل این نیازمندیها به عناصر متمایز داده ها که جهت خودکار نمودن سیستم مناسب هستند نمود.به روشی جهت یادداشت کردن و سازمان دادن فقره داده هایی که تشخیص می دهید نیاز داریم.از روی مبنا و نیازمندیهای تعیین شده هر رجوعی به چیزی را که به نظر فقره داده می آید باید استخراج نمود و نام آن را و هر چیزی که درباره اش می دانیم یادداشت کنیم.می توان با بیرون کشیدن اسمها از کار در دست انجام آغاز به کار نمود.هر اسم بالقوه یک فقره داده ها می باشد.

مثلاً لیست زیر فقره داده های سیستم نمونه پرسنلی است:

Employee,departments,salary,date-hired,Projects,Schedules,hours-budgeted,hours-worked,department-number, department-name,progect_name,progect_number,....

این لیست به عنوان یک مجموعه اولیه فقره داده هایی که در پایگاه داده های سیستم پرسنلی که در حال طراحی آن می باشیم ذخیره و گزارش خواهد شد می تواند به کار رود.

۴- جداسازی عناصر داده ها از فایل ها

در اینجا باید از قضاوت و بینش خود در قالب یک طراح پایگاه داده ها بهره بگیریم.به فقره های داده ها که جمع آوری نموده ایم باید نظری بیفکنیم.کدامیک از این فقره های داده ها به صورت عناصر منفرد داده ها به نظر می رسند و کدامیک بیشتر به صورت مجموعه سازمان یافته منطقی عناصر به نظر می رسند.با بررسی آنها مشخص خواهد شد که کدام فقره های داده ها هستند و کدام نیستند.مثلاً **date** مشخص است که یک عنصر داده هاست.اما **assignment** چندان روشن نیست که عنصر داده هاست.**department-number** یک عنصر داده است در حالی که **department** یک موجودیت داده هاست که ممکن است توسط تعدادی عناصر داده ها احتمالاً از آن جمله **department-number** نشان داده شود.

۵- ساخت فرهنگ داده ها

فرهنگ داده ها به عنوان مؤلفه اساسی در طراحی پایگاه داده ها تعریف و شناخته می شود. در حقیقت فرهنگ داده ها در جایگاه خود پایگاه داده ای است که حاوی داده هایی درباره داده ها می

باشد. بجای اینکه فقط داده های خام داشته باشد، حاوی شرحهایی در مورد سایر موجودیتهای سیستم می باشد. فرهنگ داده ها دائماً به روز نگه داشته می شود.

پس از آنکه تشخیص دادیم که عناصر داده ها کـدام هسـتند مـی تـوانیم فرهنـگ داده هـا را بسازیم.باید هر چیزی را که در مورد هر عنصر داده ها مشخص است یا می تواند تعیین شود جمع آوری نمود.حد اقل چیزی که نیاز است این است که اندازه و نوع داده ها را بدانیم.

باید خواص فیزیکی تمام عناصر داده هایی را که باید در پایگاه داده ها ذخیره شوند به گونـه ای روشن تعریف کنیم.

این مثال نشان می دهد که چگونه ساختمان داده های سطح پایین تـر داده فـاکتور مشـتری را تشکیل می دهند.

داده ها	عناصر
اطلاعات مشتری	فاکتور مشتری
اطلاعات فاکتور	
لیست کالا	
نام مشتری	اطلاعات مشتری
آدرس مشتری	
شماره فاکتور	اطلاعات فاکتور

نام فروشنده

شماره حساب فروشنده

تاریخ فاکتور

کد فرآورده **لیست کالا**

مقدار سفارش شده

شرح

قیمت فرآورده

یک نمودار از ساختمان داده ها در فرهنگ داده ها

عناصر اصلی فرهنگ داده ها ، جریان های داده ها و داده های ذخیره شده هستند.جریان های داده ها نشانگر داده های در حال حرکت است و عاملی است که توسط آن اقلام داده ها از یک گره پردازشی به گره پردازشی دیگر نقل مکان می یابند.

یک ساختمان داده ها از یک یا چند قلم داده تشکیل می گردد که در کل یک مفهوم منطقی را منتقل می نماید، یا پدیده ای را تشریح می نماید.مثلاً مشخصات دانشجو ساختمانی از داده هاست که از شماره دانشجویی،نام،نام خانوادگی، نام پدر ، تاریخ تولد و... تشکیل می گردد و هر کدام از این عناصر تشکیل دهنده ساختمان داده ها عنصر داده ها یا قلم داده ها یا فقره داده ها یا فیلد و یا میدان نامیده می شوند.

بعضی از عناصر داده ها نظیر نام پدر نمی توانند به عناصر کوچکتری تفکیک شوند.ولی برخی مثل تاریخ تولد به عناصر کوچکتری به صورت روز، ماه و سال تفکیک پذیرند.

مثال:

مشخصات دانشجو=شماره دانشجویی

+نام

+نام خانوادگی

+نام پدر

+تاریخ تولد

+(محل صدور شناسنامه)

محتوای دیکشنری داده ها

۱- نام ساختار داده ای
۲- نام موجودیت ها و ارتباط بین آنها
۳- نام صفات خاصه هر نوع موجودیت و نوع و رنج آنها
۴- شماهای ادراکی
۵- شماهای خارجی
۶- شماهای داخلی
۷- مشخصات فنی کاربران و چگونگی حق دستیابی آنها به داده ها
۸- تراکنش هایی که باید روی بانک انجام بشود.
۹- مشخصات برنامه های غیر کاربردی که برای تماس با بانک نوشته شده اند.
۱۰- مشخصات گزارشاتی که باید از بانک گرفته شود.
۱۱- واحدهای اندازه گیری
۱۲- تاریخ ایجاد داده ها و مکانیسم ورود داده ها به بانک
۱۳- ارتباط بین برنامه های کاربردی و داده ها

در واژه نامه داده ای حداقل نام و خصوصیات هر ویژگی از جداول موجود در سیستم تعریف می شود.به عبارت دیگر واژه نامه داده ای شامل ماورای داده ها (داده درباره داده) می باشد.

مثال:

نام جدول	نام ویژگی	نوع	طول	الزامی	Pk/Fk
EMPLOEE	ENO	INT	۴	Y	PK
"	ENAME	CHAR	۱۲	Y	
"	MGR	INT	۴	N	FK
DEPT	DNAME	CHAR	۱۰	Y	Pk
"	FLOOR	INT	۱	n	
ITEM	INAME	CHAR	۸	Y	Pk
"	TYPE	CHAR	۱	N	
"	COLOR	CHAR	۶	N	
SUPPLY	SUPPLIER	CHAR	۲	Y	PK
"	DEPT	CHAR	۱۰	N	

۶– جمع آوری عناصر داده ها در فایل ها

هنگامی که عناصر داده ها را از مجموعه داده ها جدا می سازیم مجموعه ها باقی مـی ماننـد.بـر روی آن مجموعه ها باید کار کرد چون احتمال دارد آنها فایلهایی باشند.

می توان بر لیست نیازمندیهای خود نظری افکند تا مشاهده نماییم که کدام یک از اقلام بالا در پایگـاه داده های مورد نظر به صورت فایل خواهند بود.

مثلاً فایل **employees** شامل عناصر داده های زیر خواهند بود:

Employee_number, employee_name, date, salary, department_number

۷– تشخیص ویژگیهای بازیابی هر فایل

هدف از تشخیص نیازمندیهای بازیابی پایگاه داده ها این است که بتـوانیم تصـمیم بگیـریم کـه کدام فیلدها باید کلید اصلی یا اولیه باشند و کدام فیلدها کلید ثانوی باشند.

۸– تشخیص رابطه های بین فایل ها

در یک پایگاه داده های رابطه ای فایل ها معمولاً هنگامی که حاوی عنصر داده های مشترک مـی باشند بهم مرتبط هستند.

۹– طراحی و ساخت شما برای سیستم مدیریت پایگاه داده های مورد استفاده

در این گام آگاهی از زبان شمای سیستم مدیریت پایگـاه داده هـای مـورد اسـتفاده لازم مـی باشد،زیرا در این گام نمودارها و آنچه را یادداشت کرده ایم باید به قالبی که سیسـتم مـدیریت پایگـاه داده ها درک می کند تبدیل می گردد.

سیستم بانک رابطه ای

مدل رابطه ای باید حداقل دارای سه جنبه اساسی باشد :

۱ـ ساختاری داده یی رابطه ای (جدول)

۲ـ قواعد جامعیت

۳ـ امکانات کار با داده ها :مجموعه ای از عملگرهایی که در کادر مفاهیم سـاختار داده یـی عمـل مـی کنند.چون ساختار داده یی رابطه است لذا عملگرها باید چنان باشند که روی این مفهوم ریاضـی عمـل

نمایند.این عملگرها همان عملگرهای جبر رابطه ای هستند (یا عملگرهای محاسبات رابطه ای با عملکرد معادل عملگرهای جبر رابطه ای).

عملگرهای جبر رابطه ای

به مجموعه ای از قوانین و عملگرها که امکان پردازش جداول را فراهم می سازند جبر رابطه ای می گویند.

این عملگرها رابطه ها را به عنوان عملوند گرفته ویک رابطه را به عنوان نتیجه برمی گرداند.

جبر رابطه ای، مجموعه روشها برای ساختن جداول جدید از جداول موجود می باشد.جبر رابطه ای برای بیان روشهای ساخت جدولهای جدید از نظریه رابطه ها و جبر مجموعه ها استفاده نموده است.

از این لحاظ مجموعه عملگرهای جبر رابطه ای به دو دسته کلی زیر تقسیم می شوند:

۱- عملگرهای مجموعه ای مانند:اجتماع[۲۵]،اشتراک[۲۶]،تفاضل[۲۷]،ضرب کارتزین[۲۸]

۲- عملگر های خاص مانند: گزینش[۲۹] ، پرتو[۳۰] ، پیوند[۳۱] و تقسیم[۳۲]

عملگر های پیوند، تقسیم و اشتراک عملگر های مرکب می باشند و می توان به کمک سایر عملگر های که به عملگر های ساده موسومند، نوشته شوند.

این عملگرها نه تنها برای انجام عمل بازیابی بکار می روند، بلکه ایجاد عباراتی که مقادیر جملات تشکیل دهنده آن ، رابطه ها هستند، را نیز امکان پذیر می سازند.و از این رهگذر ، مبنای لازم برای مطالعه و تحقیق درباره جنبه های مختلف سیستم بانک اطلاعاتی را تأمین می کنند.این جنبه ها عبارتند از: طراحی بانک،بهینه سازی پرس و جو، تعریف دید، سازماندهی مجدد بانک و ایمنی جامعیت بانک.

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

عملگرهای مجموعه ای بر روی رابطه ی R_1 و R_2 در صورتی معنی دارد که رابطه ی R_1 و R_2 سازگار باشند یعنی درجه آنهایکسان بوده وصفت خاصه ی i ام از هر دو رابطه از یک میدان باشند.

عملگرهای مجموعه ای

---

۲۵ Union
۲۶ Intersection
۲۷ Difference
۲۸ Cartesian Procuct
۲۹ Select
۳۰ Project
۳۱ Join
۳۲ Division

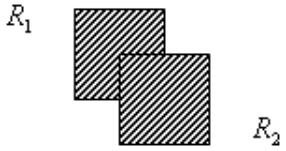
• اجتماع دو رابطه

برای اینکه عملیات اجتماع $r \cup s$ معتبر باشد دو شرط باید صادق باشند:

۱- تعداد صفات خاصه رابطه های r و s برابر باشند.

۲- دامنه های i امین صفت خاصه r و i امین صفت خاصه s برای تمام مقادیر i برابر باشند.

تاپلهای تکراری در عمل اجتماع(union) حذف می شود.



$$R = R_1 \ UNION \ R_2$$

مثال: اجتماع دو جدول زیر را بدست آورید:

نام	STIDED
علی	۱۰۰
حسن	۱۰۷
رضا	۱۰۵
اکبر	۱۰۹
سارا	۱۰۸

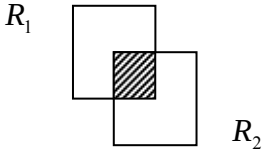
نام	STIDED
علی	۱۰۰
رضا	۱۰۵
سارا	۱۰۸

پاسخ:اجتماع دو جدول فوق به صورت زیر می باشد.

نام	STIDED
علی	۱۰۰
رضا	۱۰۵
سارا	۱۰۸
اکبر	۱۰۹
حسن	۱۰۷

• عملگر اشتراک

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



$$R = R_1 \ INTERSECT \ R_2$$

مثال:اشتراک دو جدول زیر را بدست آورید:

نام	STIDED
علی	۱۰۰
رضا	۱۰۵

سارا	۱۰۸

نام	STIDED
علی	۱۰۰
حسن	۱۰۷
رضا	۱۰۵
اکبر	۱۰۹

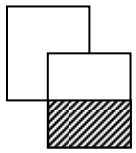
پاسخ:جدول اشتراک دو جدول بالا

نام	STIDED
علی	۱۰۰
رضا	۱۰۵

- **عملگر تفاضل minus**

تفاضل دو رابطه رابطه ای است که تاپلهایش در رابطه اول موجود باشد اما در رابطه ی دوم وجود نداشته باشد.با نماد "-" نمایش داده شده و حاصل r-s را بدست می آورد.

R_1



R_2

$$R = R_1 \ MINUS \ R_2$$

نام	STIDED
علی	۱۰۰
رضا	۱۰۵
حسن	۱۰۷
سارا	۱۰۸
مریم	۱۰۹

نام	STIDED
علی	۱۰۰
حسن	۱۰۷
رضا	۱۰۸
اکبر	۱۰۹

پاسخ: جدول تفاضل دو جدول بالا

نام	STIDED
مریم	۱۰۹
سارا	۱۰۸

- **عملگر ضرب کارتزین**

عملیات حاصلضرب دکارتی که با نماد "*" نشان داده می شود به ما اجازه می دهد تا اطلاعاتی را از هر دو رابطه با هم ترکیب کنیم.حاصلضرب دکارتی رابطه های r_1 و r_2 را به صورت $r_1 \times r_2$ می نویسیم.

X	۵
y	۱۲

z	۸
w	۱۳

A	ALI	۱۳
B	REZA	۱۵
C	ALI	۹

ضرب دو جدول بالا

X	۵	A	ALI	۱۳
X	۵	B	REZA	۱۵
X	۵	C	ALI	۹
y	۱۲	A	ALI	۱۳
y	۱۲	B	REZA	۱۵
y	۱۲	C	ALI	۹
Z	۸	A	ALI	۱۳
Z	۸	B	REZA	۱۵
Z	۸	C	ALI	۹
W	۱۳	A	ALI	۱۳
W	۱۳	B	REZA	۱۵
W	۱۳	C	ALI	۹

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

عملگر ضرب کارتزین زمان وفضای زیادی می خواهد به همین دلیل تا حد امکـان بایـد ازآن اجتناب کرد اگر رابطه r۱ دارایm سطروn ستون باشد ورابطه یr۲ دارایx سطروy سـتون باشـد آنگـاه $r_1 \times r_2$ دارای $m \times x$ سطرو $n + y$ ستون است.عملگر ضرب کارتزین در مدل رابطه ای خاصیت جابه جایی دارد زیرا در مدل رابطه ای ترتیب ستون ها مهم نیست همچنین ضرب کارتزین دارای خاصـیت شـرکت پذیری است.

عملگرهای اشتراک ، اجتماع و ضرب خاصیت شرکت پذیری دارند:

$$(T_1 \cup T_2) \cup T_3 = T_1 \cup (T_2 \cup T_3) = T_1 \cup T_2 \cup T_3$$

$$(T_1 \cap T_2) \cap T_3 = T_1 \cap (T_2 \cap T_3) = T_1 \cap T_2 \cap T_3$$

$$(T_1 \times T_2) \times T_3 = T_1 \times (T_2 \times T_3) = T_1 \times T_2 \times T_3$$

عملگرهای خاص

- **عملگر گزینش**

عملگر گزینش یا محدودیت(restrict) به عنوان select شناخته می شود.این عملگـر طبـق یـک شرط تعدادی از ردیفهای یک رابطه را برمی گرداند.گـزاره بـه صـورت انـدیس Select نشـان داده مـی شود.رابطه شناسه در داخل پرانتزی که به دنبال Select نوشته می شود قرار می گیرد.

به طور کلی با بهره گیری از عملگرهای $=,\neq,<,\leq,>,\geq$ و مقایسه هایی در گزاره انتخاب انجام می گیرند. علاوه بر آن می توانیم با بهره گیری از عملگرهای منطقی $\wedge$ و $\vee$ چند گزاره را با هم ترکیب کنیم.و به صورت یک گزاره بنویسیم.

به عنوان مثال جدول sp را در نظر بگیرید.دستور $Select_{Qty>300}(Sp)$ تاپلهایی را بر می گرداند که مقدار QTY آنها بیشتر از ۳۰۰ باشد.

جدول sp

S#	P#	Qty
s۱	p۱	۳۰۰
s۱	p۲	۲۰۰
s۱	p۳	۴۰۰
s۱	p۴	۲۰۰
s۱	p۵	۱۰۰
s۱	p۶	۱۰۰
s۲	p۱	۳۰۰
s۲	p۲	۴۰۰
s۳	p۲	۲۰۰
s۴	p۲	۲۰۰
s۴	p۴	۳۰۰
s۴	p۵	۴۰۰

نتیجه به صورت زیر می باشد.

S#	P#	Qty
s۱	p۳	۴۰۰
s۲	p۲	۴۰۰
s۴	p۵	۴۰۰

مثال:دستور $Select_{Qty>300 \wedge P\#=P2}(Sp)$ را در نظر بگیرید. خروجی شامل تاپل زیر است:

S#	P#	Qty
s۲	p۲	۴۰۰

- **عملگر تصویر یا project**

این عملگر زیر مجموعه ای عمودی از یک رابطه را استخراج می کند. این عملگر تاپلهای تکراری را حذف می کند.همچنین صفات خاصه نمی توانند تکراری باشند .برای این کار آن صفات خاصه ای را که می خواهیم در لیست خروجی باشد به صورت اندیس در زیر دستور Project می نویسیم.فرض

کنید قرار است تمام شماره های قطعات از جدول SP را لیست کنیم اما نیازی به سایر اطلاعات نداریم.دستور $Project_{S\#}(SP)$ تمام شماره قطعات را از جدول SP بدون تکرار لیست می کند و خروجی به صورت زیر می باشد:

S#
s۱
s۲
s۳
s٤

- **عملگر پیوند یا join**

غالباً ساده سازی بعضی از پرس و جوها که نیاز به حاصلضرب دکارتی دارند مطلوب است.به طور معمول یک پرس و جو که در بر گیرنده یک حاصلضرب دکارتی است شامل یک عملیات انتخاب بر روی نتیجه حاصلضرب دکارتی می باشد.

عمل پیوند دو رابطه، روی صفت خاصه ای از دو رابطه انجام می شود که از یک میدان باشند.حاصل عمل پیوند رابطه R_1 روی صفت خاصه x با رابطه R_2 روی صفت خاصه y ، مجموعه تاپلهایی است که از

الصاق تاپل r_1 از رابطه R_1 و تاپل r_2 از رابطه R_2 به نحوی که: $r_1.x \ op \ r_2.y$ باشد، بدست می آیند. به عنوان مثال اگر بنویسیم $S1 = join\ S2\ over\ code$ دو رابطه S۱ و S۲ روی صفت خاصه code با هم پیوند می شوند و تاپلهایی از دو رابطه بهم الصاق می شوند که مقدار code در آنها یکسان باشد.

S۱

معدل	نام	code
۱۲.۵	علی	۱۰۰
۱۴	رضا	۱۰۲
۱۳.۵	سارا	۱۰۵
۱۵	حسن	۱۰۷

S۲

code	Telephone
۱۰۰	۵۵۵۵
۱۰۲	۴۵۱۰۱
۱۰۳	۵۶۱۰۱۰۱
۱۰۷	۲۳۱۰۷

پیوند دو جدول صفحه قبل بر اساس کد

S۱.code	نام	معدل	S۲.code	telephone
۱۰۰	علی	۱۲.۵	۱۰۰	۵۵۵۵
۱۰۲	رضا	۱۴	۱۰۲	۴۵۱۰۱

۱۰۷	حسن	۱۵	۱۰۷	۲۳۱۰۷

- **عملگر تقسیم(divdby)**

عملیات تقسیم که با نماد(÷) نشان داده می شود مناسب پرس و جوهایی است که حاوی عبارت "برای تمام " می باشد.

این عملگر رابطه ی r۱ از درجه m+n را بر رابطه r۲ از درجه n تقسیم می کند و خارج قسمت رابطه ای است از درجه m. (n تا صفت مشترک بین r۱ و r۲ با میدانهای یکسان وجود دارد) و به صورت زیر نوشته
می شود:

r۳= r۱ dived by r۲

فرض کنید r۱ دارای دو صفت خاصه ی x و y باشد r۲ نیز دارای یک صفت خاصه y باشد در این صورت r۳ که حاصل تقسیم آنها است دارای یک صفت خاصه x است که در ازای هر x در رابطه ی r۳ مقداری از r۱*x باشد به طوری که به ازای تمام مقادیر y از رابطه ی r۲ تاپل (x و y)در رابطه r۱ وجود داشته باشد.
مثال:دو جدول r۱ و r۲ به صورت زیر موجود هستند حاصل تقسیم r۱ بر r۲ را بدست آورید.

r۱

X	Y
رضا	۵
علی	۷
حسن	۳
رضا	۷
حسن	۵
حسن	۷

r۲

Y
۵
۷
۳

جدول تقسیم = **r۳**

X
حسن

بخش ۲

عنوان:

زبان پرس و جوی SQL

SQL

زبان SQL برای پردازش ، ذخیره و بازیابی داده در پایگاه داده های رابطه ای استفاده می شود و حدوداً شامل ۳۰ دستور می باشد. SQL زبان استاندارد و جامع پیاده‌سازی، مدیریت، نگهداری و کار با بانکهای اطلاعاتی می‌باشد که تقریباً توسط تمام بانکهای اطلاعاتی کوچک و بزرگ مانندAccess ، SQL Server، Oracle و DB۲ پشتیبانی می‌شود. طراحان و افرادی که بنوعی با بانکهای اطلاعاتی سروکار دارند و همچنین برنامه نویسانی که از این بانکها استفاده می‌کنند هرکدام باید تا اندازه‌ای با این زبان آشنایی داشته باشند.

این زبان به تنهایی کلیه نیازهای برنامه نویسان پایگاه داده ها را تأمین نمی نماید و یک زبان برنامه نویسی مکمل جهت تهیه منوها و الگوریتمهای مختلف مورد نیاز می باشد.اکثر نرم افزارهای مدیریت پایگاه داده ها(DBMS) ، زبانهای نسل چهارم را بعنوان مکمل زبان SQL معرفی می نمایند و این امکان را فراهم می سازند که از داخل نرم افزار نسل چهارم کلیه دستورات SQL را اجرا نمایند.همچنین دستورات SQL در ساختارهای نسل سوم مانند C ، پاسکال و غیره ادغام می شود و دستورات SQL را بصورت تعاملی اجرا می نماید.

برای نشان دادن دستورات SQL در قالب شبه برنامه نویسی ، از قوانین زیر تبعیت می شود:

۱- کلمه هایی با حروف بزرگ ، قسمت اصلی دستور SQL می باشند و حروف کوچک ، توسط برنامه نویسان تعیین می شوند.

در مثال زیر کلمه های SELECT و FROM قسمت اصلی دستورSQL می باشند و کلمه های name و student توسط برنامه نویس تعیین می شود.

```
SELECT name
FROM student
```

۲- از گزینه های داخل نماد"[]" ، می توان در صورت نیاز استفاده نمود.اگر گزینه های مندرج در نماد "[]" با "|" جدا شده باشند، هیچکدام و یا یکی از گزینه ها باید انتخاب شوند.

در مثال زیر فقط می توانید حداکثر یک و یا هیچکدام از گزینه ها را انتخاب نمایید:

```
[database|data structure|network]
```

اگر گزینه ها با نماد"،" جدا شده باشند، می توان هیچ، یک و یا تعدادی از گزینه های مورد نیاز را انتخاب نمود.مثال:

```
[Database, data structure, network]
```

۳- از گزینه های داخل نماد"{}"، حداقل باید یک گزینه انتخاب شود.اگر گزینه های مندرج در آن با نماد"|" جدا شده باشند ، باید فقط و فقط یک گزینه انتخاب شود.در مثال زیر باید یک گزینه انتخاب گردد.

```
{database|data structure|network}
```

اگر گزینه ها با نماد "،" جدا شده باشند، باید حداقل یک و یا تعدادی از گزینه ها انتخاب شوند.مثال:

{Database, data structure, network}

۴- اگر نماد "..." بکار رود ، تکرار گزینه امکان پذیر است.

معرفی SQL و دستورات عمومی آن

توسط دستورات SQL می توان درون یک بانک اطلاعاتی پرس و جو کـرده (Query) و نتیجـه را برگرداند. بانک اطلاعاتی شامل آبجکتی به نام جدول (Table) می باشد.رکوردها در بانکهای اطلاعات در جداول ذخیره می گردند.جدول شامل سطر و ستون می باشد.

پر کاربرد ترین دستورات SQL شامل موارد زیر است :

SELECT : استخراج یک داده از بانک اطلاعاتی

UPDATE: به روز رسانی یک داده درون بانک

DELETE :پاک کردن یک داده از بانک اطلاعاتی

INSERT :وارد کردن یک داده جدید به بانک اطلاعاتی

همچنین در SQL می توان داده هایی نیز تعریف کرد :

CREATE TABLE :ایجاد یک جدول جدید

ALTER TABLE: تغییر دادن یک جدول

DROP TABLE :پاک کردن یک جدول

CREATE INDEX : ایجاد یک اندیس

DROP INDEX : پاک کردن یک اندیس (کلید جستجو)

معرفی داده ها در زبان SQL

زبان معرفی داده ها برای ایجاد پایگاه داده ها و جداول مختلفی که قبلاً در طراحی پایگاه داده ها تعریف شده است، بکار گرفته می شود.

ایجاد پایگاه داده

قبل از تعریف جداول و تراکنش های یک پایگاه داده، لازم است یک پایگاه داده ایجاد گردد.

با استفاده از قالب زیر ، می توان یک پایگاه داده ایجاد کرد.در دستور زیر ،database-name نام پایگاه داده می باشد.

```
CREATE DATABASE database-name
  [ON {DEFAULT [=size] device-name[=size]}
  [,device-name [=size]]…]
```

در مثال زیر یک پایگاه داده بنام HOSPITAL ایجاد می گردد.

```
CREATE  DATABASE  hospital
```

ایجاد جدول

پس از ایجاد پایگاه داده ، باید جداول و ویژگیهای هر جدول معرفی شوند.این جداول معمولاً رابطه هایی هستند که در مرحله طراحی پایگاه داده نرمال سازی شده اند.فرمت دستور ایجاد جدول به صورت زیر است:

```
CREATE TABLE Table_Name
(Field۱ Type۱[NOT NULL|NULL]
,Field۲ Type۲ [NOT NULL|NULL]
.
.
Fieldn Typen [NOT NULL|NULL],
PRIMARY KEY (Pre-Field),
FOREIGN KEY (Pre-Field) REFERENCES table-name);
```

درREFERENCES نام جدولی را مشخص می کند که به آن مراجعه می نماید.در این کاربرد Table_Name نام جدولی است که باید ایجاد شود و Field۱ ، Field۲ ،...وFieldn نام فیلدهایی را مشخص می کنند که باید ایجاد شوند و Type۱،Type۲ ،...Typen نوع فیلدهارا مشخص می کنند. مقدار Not Null مشخص می کند که مقدار این فیلد نمی تواند خالی باشد و Pre-Field نام فیلد کلید اصلی یا خارجی است.
انواع داده ها در جدول زیر آمده است.

نوع	هدف
Bit(n)	n بیت از حافظه را در نظر می گیرد.
Decimal(n,n)	یک عدد دهدهی با دقت n و مقیاس n در نظر می گیرد.
Integer	یک عدد صحیح در نظر می گیرد.
SmallInt	یک عدد صحیح با طول کوتاه در نظر می گیرد.
Number(n)	یک عدد صحیح n رقمی در نظر می گیرد.
Real(n)	یک عدد اعشاری با دقت معمولی در نظر می گیرد. (n دقت است.)
Float(n)	یک عدد اعشاری با دقت بالاتردر نظر می گیرد.(n دقت است.)
Double Precision	یک عدد اعشاری با دقت بسیار بالا در نظر می گیرد.(n دقت است.)
Date Time	برای نگهداری تاریخ و زمان به کار می رود،عناصر DateTime عبارتند از Year(سال) ،Month(ماه)،Day(روز)،Hour(ساعت)،Minute (دقیقه)وSecond(ثانیه) است.

Date	**تاریخ را نگهداری می کند.**
Time	**زمان را نگهداری می کند.**
Char(n)	**n کاراکتر ثابت در نظر می گیرد.**
Varchar(n)	**کاراکترهایی با طول متغیر نگهداری می کند که حداکثر می تواند n بایت داشته باشد.**

دستورات زیر را در نظر بگیرید:

```
Create Table St۱
(StNo Char(۱۰) Not Null Primary key,Fname Varchar(۱۵) Not Null,Lname Varchar(۲۰) Not
Null,Date_s date Null)
Primary Key(StNO));
```

این دستور جدولی به نام St۱ با ساختار زیر ایجاد می کند:

نام فیلد	نام فیلد (فارسی)	نوع	وضعیت	کلید اصلی
StNo	شماره دانشجویی	کاراکتر ۱۰	نمی تواند خالی باشد.	بلی
Fname	نام	کاراکتری با طول متغیر حداکثر ۱۵ کاراکتر	نمی تواند خالی باشد.	خیر
Lname	نام خانوادگی	کاراکتری با طول متغیر حداکثر ۲۰ کاراکتر	نمی تواند خالی باشد.	خیر
Dat_s	تاریخ شروع به کار	تاریخ	می تواند خالی باشد.	خیر

ایجاد شاخص

با توجه به توانایی و کاربرد شاخص برای جستجو داده در یک یا چند جدول ، زبان SQL امکاناتی برای انجام شاخص بر روی یک یا چند ویژگی را ارائه می نماید.قالب ایجاد یک شاخص به شکل زیر است:

```
CREATE [UNIQUE] INDEX Index_Name
On Table_Name(Field۱,Field۲,…,Fieldn)
```

در این کاربرد ،Index_Name نام فایل ایندکسی است که باید ایجاد شود.Table_Name نام جدولی را مشخص می کند که باید برای آن ایجاد شود و Field۱,Field۲و...Fieldn نام فیلدهایی است که باید ایندکس بر اساس آن فیلدها ایجاد شود.با ایجاد شاخص بر روی هر ویژگی ، یک جدول شاخص ساخته می شود که از طریق این جدول ، جستجو انجام می گیرد.

به عنوان مثال دستورات زیر را در نظر بگیرید:

```
Create Index name_Idx
On St۱(Lname)
```

این دستور ایندکسی به نامname_Idx بر اساس فیلد Lname مربوط به جدول St۱ ایجاد می کند.

Lname	pointer
rajabi	
sadeghi	
mehrabani	

Lname	Dat_s	Fname	StNo
rajabi	۸۳/۱/۹	sahar	۸۳۲۲۷۶۱۲۳
mehraban	۸۳/۲/۷	nima	۸۳۲۲۷۵۱۲۰
sadeghi	۸۵/۱۲/۱۰	sara	۸٤۱۲۳٤٥٦۷

ارتباط جدول ایندکس و جدول اصلی

حذف ایندکس

دستور DROP INDEX برای حذف فایل Index که ساخته ایم به کار می رود و قالب کلی دستور به صورت زیر می باشد:

```
DROP INDEX Index_Name
```

Index_Name نام فایل اینکسی است که باید حذف گردد.

تغییر نوع داده در جداول ساخته شده

برای تغییر ساختار جداولی که قبلاً بوسیله دستور CREATE ساخته شده اند، می توان از دستور ALTER استفاده نمود.قالب این دستور در زیر نشان داده شده است:

```
ALTER TABLE table-name
        MODIFY(column-namenew characteristic)
```

مثال:

ALTER TABLE Student
MODIFY (StNo Varchar (۸)NOT NULL)

این دستور ، نوع فیلد StNo جدول Student را تغییر می دهد.

اضافه کردن یک ستون به جدول ساخته شده:

ALTER TABLE Table Name
ADD (column-name data type [NOT NULL| NULL]

به مثال زیر برای اضافه کردن یک ستون به جدول patient به نام محل تولد(BIRTH_PLACE) توجه نمایید:

ALTER TABLE patient
ADD (birth-place CHAR (۵))

حذف یک جدول

برای حذف جدول ،از دستور DROP استفاده می شود.قالب دستور حذف جدول به صورت زیر می باشد:

DROP TABLE Table Name

Table Name نام جدولی است که باید حذف گردد.

برای مثال برای حذف جدول patient ، از دستور زیر استفاده می شود:

DROP TABLE patient

مدیریت داده ها در زبان SQL

پس از تعریف داده ها و تعیین ساختار جداول در بخش قبل، در این بخش مدیریت و پردازش داده به زبان SQL ارائه می شود.

بازیابی داده ها

جهت بازیابی داده از یک یا چند جدول از دستور SELECT استفاده می شود که تمام رکوردها یا تعدادی از رکوردهای جدول را بازیابی می کند. هدف نهایی جستجو و یافتن اطلاعات مفید می‌باشد. به این عمل یعنی جستجوی اطلاعات در بانک اطلاعاتی Query نیز گفته می‌شود. اکثر دستورات زبان SQLنیز در همین راستا مورد استفاده قرار می‌گیرند .چون مهمترین و پرکاربردترین دستور را می‌توان دستور SELECT قلمداد کرد.شکل کلی این دستور به صورت زیر است:

SELECT $A_1, A_2, ..., A_n$
$r_1, r_2, ..., r_n$ **FROM**
WHERE P

هر A_i نشانگر یک صفت خاصه و هر r_i نشان دهنده یک رابطه و P یک گزاره است.

یا به طور کامل تر به صورت زیر به کار می رود:

SELECT [predicate] {*| Table.*| [Table.] Field۱ [as Alias۱] [, Table.] Field۲ [As alias۲] [,…]}
FROM table-name
[WHERE [شرط]]
[GROUP BY grouping-column]
[ORDER BY {ASC DESC}]
[HAVING search-condition]

Predicate می تواند یکی از مقادیر زیر را بپذیرد:

- **All**. این مقدار تمام رکورد ها را بازیابی می کند.
- **Distinct**. این مقدار رکوردهایی با داده های تکراری را حذف می کند.(در فیلدهای انتخاب شده در دستور select)
- **Distinct Row**. این مقدار داده ها را در تمام رکوردهای تکراری حذف می کند.برای این کار کل یک رکورد با رکورد دیگر مقایسه می گردد،چنانچه این رکوردها برابر باشند،در نمایش حذف خواهند شد.
- **Topen[perecent]**.این مقدار ، تعدادی از رکوردها یا درصدی از رکوردها را بازیابی می کند.

پارامتر*.تمام فیلدهای جدول یا جدول ها را مشخص می کند.

پارامتر Table:جدولی را تعیین می کند که فیلدها باید از آن جدول انتخاب و بازیابی شوند.

پارامتر های Field۱ ،Field۲ و... فیلدهایی را مشخص می کنند که باید داده ها از آن ها بازیابی شوند.

پارامتر های Alias۱ و Alias۲ و... عباراتی را مشخص می کنند که باید در عنوان ستون ها به جای نام فیلد چاپ شوند.(به عنوان مثال عناوین فیلدهای فارسی)

FROM:در این بخش از دستور SQL ، نام جدول یا جداول را تعیین می کنند.در صورتیکه نام بیش از یک جدول در این بخش بکار رود، عمل پیوند انجام می گیرد.

پارامتر Table-name:جدول یا جدولهایی است که داده ها باید از آن ها بازیابی شوند.

WHERE : در این بخش شرایط و محدودیتهای بازیابی داده ها تعیین می شود.محدودیت مندرج در این قسمت شامل اپراتورهای ریاضی، منطق و غیره می باشند.

عملگرهای ریاضی:این اپراتورها عبارتند از:+ ، - ، * و /

عملگر || برای الصاق دو رشته کاراکتری استفاده می شود.مثلاً با دستور NAME ||FAMILY دو متغیر رشته ای با هم ترکیب می شود.

عملگرهای مقایسه ای: این اپراتورها بعد از دستورWHERE بکار گرفته می شوند و شامل موارد زیر می باشند:

مساوی	=	کمتر یا مساوی	<=
کمتر	<	بزرگتر یا مساوی	>=
بزرگتر	>	نامساوی	<> یا =~

رابطه های منطقی عبارتند از : AND,OR و NOT

پارامتر GROUP BY: رکوردهایی را گروهبندی می کند که مقادیر آن ها در یک فیلد یا چند فیلد یکسان باشد. نام فیلدهایی که باید رکوردهایی بر اساس آن گروهبندی شوند، در جلوی دستور GROUP BY قرار می گیرند.حداکثر می توان ۱۰ گروه را مشخص کرد.

پارامترORDER BY :رکوردها را به ترتیب نزولی یاصعودی مرتب می کند.فیلدهایی که باید رکوردها بر اساس آن ها مرتب شوند،در جلوی پارامتر Order by قرار می گیرند.

ASC :اگر ترتیب صعودی باشد از این دستور استفاده می کنیم.

DESE:اگر ترتیب نزولی باشد از این دستور استفاده می کنیم.

نکته : پیش فرض بصورت صعودی است واگر ننویسیم اشکال ندارد.

پارامترGroup By, Having :

همیشه دستور Group by , Having با هم به کار می روند و باعث می شوند که طبق شرایطی بعضی از گروه ها حذف شوند و مشخص می کند که کدام یک از رکوردهای گروهبندی شده ظاهر شوند . شرط تعریف شده در جلوی کلمه Having قرار می گیرد.نقش Having در Group by مثل Where در سطر است .

مثال:

بعنوان مثال اگر در بانک اطلاعاتی Student بخواهیم اسم و فامیل تمام دانشجویان را مشاهده کنیم باید پرس و جویی به شکل زیر بنویسیم:

```
SELECT name, family
FROM students
```

در صورتی که بخواهیم تمام فیلدهای یک جدول را ببینیم می‌توانیم بجای نوشتن اسم تمام فیلدها فقط از یک کاراکتر ستاره استفاده کنیم. کاراکتر ستاره بمعنی تمام فیلدهای یک جدول می‌باشد. مثلاً دو دستور زیر با هم معادلند:

```
SELECT *
SELECT name, family, age, gpa
```

مثال:

اگر بخواهیم مشخصات اساتیدی که بیش از چهل سال سن دارند را ببینیم باید پرس و جویی بصورت زیر بنویسیم:

```
SELECT *
FROM teachers
WHERE  age>۴۰
```

مثال:

اگر بخواهیم مشخصات دانشجویانی که نام آنها علی می‌باشد و سن آنها نیز کمتر از پانزده سال است را ببینیم باید پرس و جویی زیر را اجرا کنیم:

```
SELECT *
FROM students
WHERE age < ۱۵ AND name = 'ali'
```

در مثال بالا دو مطلب جدید وجود دارد نخست آنکه در زبان SQL رشته متنی را باید داخل کوتیشن (') قرارداد. بنابراین برای معرفی کلمه‌ای بنام علی باید آنرا بصورت 'ali' نوشت، زیرا در این حالت این کلمه یک ثابت رشته‌ای (متنی)بحساب می‌آید. نکته دیگر آنکه می‌توان شرطهای مختلف را توسط AND ، OR و NOT با همدیگر ادغام کرده و شرطهای پیچیده‌تری را بدست آورد AND ، OR و NOT هر سه از کلمات کلیدی زبان SQL می‌باشند.

دستور ORDER BY

دستور ORDERBY جهت Sort کردن رکوردهای نمایش داده شده مورد استفاده قرار می‌گیرد. با این دستور می‌توان مشخص کرد که رکوردهایی که قرار است نمایش داده شوند برحسب کدام فیلد باید مرتب شوند. مثال: برای مشاهده کردن مشخصات اساتیدی که سن آنها بیشتر از ۲۹ سال می‌باشد و در ضمن لیست خروجی بترتیب اسم فامیل نیز مرتب شده باشد باید Query زیر را اجرا کرد.

```
SELECT *
FROM teachers
ORDER BY family
```

در این حالت افراد بترتیب اسم فامیل خود لیست خواهند شد (از A تا Z).در صورتی که بخواهیم ترتیب Sort شدن برعکس شود (از Z تا A) می‌توان پس از دستور ORDERBY از کلمه کلیدی DESCاستفاده کرد، مانند مثال زیر:

```
SELECT *
FROM teachers
ORDER  BY name DESC
```

برای دستور ORDER BY می‌توان چند فیلد تعریف کرد، در این صورت این دستور عمل Sort کردن را با در نظرگرفتن اولین فیلد انجام خواهد داد در صورتی که چند رکورد دارای مقدار مشابهی در این فیلد باشند ملاک مرتب سازی آنها فیلد دومی خواهد بود که در دستور ORDER BY ذکر شده است، در صورتی که این فیلد نیز دارای مقادیر مشابهی باشد ملاک تصمیم گیری فیلد سوم خواهد بود والی آخر.

مثال: در Query زیر نام اساتیدی که بیش از ۳۰ سال دارند برحسب فامیل آنها مرتب می‌شود در صورتی که چند استاد دارای اسم فامیل یکسانی باشند ملاک مرتب شدن، اسم کوچک آنها خواهد بود.

```
SELECT *
FROM teachers
WHERE age > ۳۰
ORDERBY family, name
```

اپراتورهای خاص :اپراتورهای مخصوص زبان SQL بشرح زیر می باشند:

BETWEEN: این اپراتور برای ایجاد محدودیت یک ویژگی بین دو شرط می باشد.

مثال:

مشخصات قطعاتی را به دست آورید که وزن آنها بین ۱۹ - ۱۶ باشد و بر حسب رنگ و وزن آنها را مرتب کنید.(خود ۱۶و۱۹ هم در نظر می گیرد)

```
SELECT  p# (pname,color,weight,city)
FROM  p
WHERE  weight   BETWEEN ۱۶ AND ۱۹
ORDER BY color,weight
```

نکته:هر گاه از دستور order by استفاده شود دیگر از ستاره استفاده نمی کنیم چون باید تک تک بنویسیم.

IS NULL: این محدودیت جهت شناختن ویژگی هایی که مقدار آنها مساوی NULL می باشد به کار گرفته می شود.

مثال:

مشخصات قطعاتی را بیابید که وزن آنها مساوی NULL باشد.

```
SELECT *
FROM P
WHERE Weight IS NULL
```

EXIST: این محدودیت جهت شناسایی ویژگیهایی است که مقدار آنها NULL نباشد.

مثال: مشخصات قطعاتی را بیابید که وزن آنها خالی نباشد.

```
SELECT *
FROM P
WHERE Weight EXIST
```

IN: این محدودیت جهت شناختن ویژگیهایی است که مقدار آنها با یکی از مقادیر آرایه انتخاب شده مساوی باشد. مجموعه‌ای که بادستور IN بکار می‌رود می‌تواند دهها عضو داشته باشد و نوع اعضا نیز می‌تواند رشته‌ای، عددی یا ... باشد. البته واضح است که تمام اعضا باید هم نوع باشند.

مثال:

مشخصات قطعاتی رابه دست آورید که وزن آنها مساوی یکی از مقادیر ۱۷و۱۶و۱۲ باشد.

```
SELECT *
FROM P
WHERE weight IN[۱۲,۱۶,۱۷]
```

مثال:

فرض کنید می‌خواهیم مشخصات تمام دانشجویانی را که نام آنها، علی یا رضا نباشند را پیدا کنیم:

```
SELECT *
FROM students
WHERE name NOT IN ('ali', 'reza')
```

در صورتی که NOT را از برنامه بالا حذف کنیم مشخصات تمام دانشجویانی که نام آنها علی یا رضا می‌باشند نمایش داده خواهد شد.

LIKE:این محدودیت جهت انتخاب ویژگیهایی که حروف مشابه دارند بکار می رود.نمونه ها را با بهره گیری از دو کاراکتر خاص تعریف می کنیم.

درصد%:کاراکتر درصد با هر زیر رشته ای مطابقت می کند.

خط اتصال (-):کاراکتر - با هر کاراکتری مطابقت می کند.

نمونه ها از حساسیت موردی برخوردارند،یعنی کاراکترهای حروف بزرگ با کاراکترهای حروف کوچک مطابقت نمی کنند یا برعکس.

مثال:

"for%"هر رشته ای که با for آغاز شود.

"%age%" با هر رشته ای که حاوی age به عنوان زیر رشته باشد مطابقت می کند مثلاً با agency ،language،agelong .

"---" با هر رشته ای که دقیقاً سه کاراکتر داشته باشد مطابقت می کند.

"٪---" با هر رشته ای که حداقل سه کاراکتر داشته باشد مطابقت می کند.

برای نمونه هایی که شامل کاراکترهای نمونه ای خاص(یعنی % , -) می باشد SQL اجازه استفاده از کاراکتر گریز که اسلش معکوس(\) می باشد را می دهد.کاراکتر گریز بلافاصله قبل از کاراکتر نمونه خاص به کار برده می شود تا نشان دهد که کاراکتر نمونه خاص همانند یک کاراکتر معمولی با آن عمل می شود.با استفاده از واژه کلیدی escape کاراکتر گریز را برای مقایسه like تعریف می کنیم.به عنوان مثال نمونه های زیر را در نظر بگیرید:

"\"escape"\%cd%"ab\" Like با تمام رشته هایی که با "%cd%"ab" آغاز می گردند مطابقت می کند.

"\"escape"%cd\\ab" Like با تمام رشته هایی که با "%cd\ab" آغاز می گردند مطابقت می کند.

به جای تطابق از عملگر مقایسه ای not like جهت عدم تطابق استفاده شود.

مثال: می‌خواهیم مشخصات دانشجویانی را ببینیم که اسم فامیل آنها به 'زاده' ختم می‌شود مانند علیزاده، محمد زاده و ... برای این منظور باید Query زیر را نوشته و اجرا کنیم:

```
SELECT *
FROM family LIKE '% زاده'
```

کاربرانی که با Access کار می‌کنند باید سطر آخر را به صورت زیر تغییر دهند:

```
WHERE family LIKE '* زاده '
```

مثال: می خواهیم نام و سن تمام اساتیدی که اسم آنها با' پ ' شروع می‌شود، نمایش داده شود.

```
SELECT name, age
FROM teachers
WHERE name LIKE 'پ%'
```

بنابراین مشخصات افرادی که نام آنها مثلاً پدرام یا پیمان باشد در خروجی لیست خواهد شد .

استفاده از توابع

در دستور SELECT علاوه بر تعریف فیلدها می‌توان از عبارتهای ریاضی و یا توابع استاندارد SQLنیز استفاده کرد.

مثال: اگر مالیات بردرآمد پنج درصد باشد Query زیر نام اساتید و مالیاتی را که هر کدام می‌پردازند مشخص می‌کند .

```
SELECT family, name, salary*۵/۱۰۰
FROM teachers
```

علاوه بر عبارتهای ریاضی می‌توان از توابع استاندارد SQL نیز استفاده کرد، تعدادی از این توابع عبارتند از:

تابع COUNT : تعداد رکوردهای یک جدول را نشان می دهد.

تابع SUM : مجموع یک فیلد عددی را برمی‌گرداند.

تابع AVG : میانگین یک فیلد عددی را برمی‌گرداند.

تابع MIN :مینیمم یک فیلد عددی را برمی‌گرداند.

تابع MAX : ماکزیمم یک فیلد عددی را برمی‌گرداند.

مثال: اگر بخواهیم تعداد اساتید و مجموع حقوقهایی را که به آنها پرداخت شده است ببینیم می‌توانیم از Query زیر استفاده بکنیم.

```
SELECT COUNT (name), SUM (Salary)
FROM teachers
```

مثال:تعداد تولید شده قطعه P۱ را به دست آورید؟

```
SELECT    SUM (QTY)
FROM    Sp
WHERE   P# =’P۱’
```

با AS می توان نام جدید به ستون خروجی داد.

```
SELECT MIN(QTY) AS  MIN_QTY
FROM S
```

تذکر:در صورت لزوم توابع فوق را می توان با کلمه Distinct نیز به کار برد.در این حالت داده های تکراری در نظر گرفته نمی شوند (برای MIN و MAX داده های تکراری اهمیتی ندارند).مقادیر NULL قبل از اجرای این توابع حذف شده و روی این توابع تأثیری ندارند.

مثال:تعداد تهیه کنندگان شهر C۲ چند تاست؟

```
SELECT COUNT(S#)
FROM S
WHERE City='C۲'
```

مثال:تعداد شهرهای موجود در جدول P چند تاست؟

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

```
SELECT COUNT(DISTINCT city) AS CT#
FROM  P
```

نکته: تابع ویژه COUNT(*) برای شمارش سطرهای جدول است.در این تابع نمی توان از DISTINCT استفاده کرد و این تابع سطرهای NULL را نیز می شمارد.اگر جدول تهی باشد، این تابع صفر بر می گرداند.

مثال:

مشخص سازیر چند تهیه کننده P۲ را تهیه کرده اند؟

```
SELECT COUNT(*)
FROM  SP
WHERE P#='P۲'
```

Having و Group By در SELECT

مثال:

کل مقدار تهیه شده از هر قطعه را همراه با شماره قطعه نشان دهید:

```
SELECT P#,SUM(Qty)
FROM SP
GROUP BY P#
```

مثال:برای هر قطعه تهیه شده ، شماره قطعه، کل تعداد و ماکزیمم تعداد تهیه شده از آن را بدون در نظر گرفتن S۱ بدهید.

```
SELECT P#,SUM(Qty),MAX(Qty)
FROM SP
WHERE S# ~='S۱'
GROUP BY P#
```

تذکر:صفتی که گروهبندی روی آن انجام می شود حتماً باید در خروجی بیاید.having همراه با grup by استفاده می شود.نقش having در گروه مانند نقش where در سطر است.به عبارت دیگر از having برای در نظر گرفتن گروهها استفاده می شود.

مثال:

شماره قطعه تمام قطعاتی که توسط بیش از یک تهیه کننده تهیه شده اند را بیابید:

```
SELECT P#
FROM SP
GROUP BY P#
HAVING COUNT(*)>۱
```

Query های چند جدولی

در SQL امکان نوشتنQuery های چند جدولی نیز وجود دارد. در این حالت اطلاعات چند جدول بطور همزمان مورد جستجو قرار می‌گیرد و حتی امکان مقایسه فیلدهایی از یک جدول با فیلدهایی از جدول دیگر نیز وجود دارد. اگر در بین جداولی که در Query شرکت داده می‌شوند فیلدهای هم نام وجود داشته باشد باید نام آن فیلدها را Fully qualified کرد بدین معنی که ابتدا اسم جدول و سپس اسم فیلد را ذکر کرد. بین اسم جدول و اسم فیلد نیز باید از یک کاراکتر نقطه ('.') استفاده کرد.

پرس و جوهای مبتنی بر پیوند جدولها

پیوند نوعی پرس و جو است که طی آن عمل بازیابی از بیش از یک جدول انجام می پذیرد.

مثال:خروجی دستور زیر چیست؟

```
SELECT S.s#, S.city, P.p#, P.city
FROM S,P
WHERE S.city=P.city
```

مثال:

```
SELECT S.*,P.*
FROM S,P
WHERE S.city=P.city
```

دستور فوق تمام فیلدهای دو جدول را در صورت برقراری شرط می نویسد.

مثال: اگر بخواهیم لیست اساتید و دانشجویانی که دارای ارتباط فامیلی هستند را ببینیم می‌توانیم از Queryزیر استفاده بکنیم:

```
SELECT *
FROM students, teachers
WHERE students.family = teachers. Family
```

یا اگر بخواهیم لیست اساتیدی را مشاهده کنیم که سن آنها از سن برخی از دانشجویان کمتر است، می‌توانیم Query زیر را اجرا کنیم:

```
SELECT teachers.family, teachers.name
FROM students, teachers
WHERE teachers.age < students.age
```

Select تو در تو

با استفاده از دستورات SQL می توان عملیات پیچیده ای را با ترکیب دستورات ساده انجام داد.

مثال: نام تهیه کنندگانی را بیابید که قطعه P۱ را تهیه می کنند.

این پرس و جو را می توانیم به کمک عملیات پیوند به صورت زیر تنظیم کنیم:

```
SELECT S.sname
FROM S,SP
WHERE S.S#=SP.S#  AND SP.P# ='p۲'
```

ولی یک راه دیگر استفاده از SELECT متداخل است:

```
SELECT sname
FROM S
WHERE S# IN (SELECT S#  FROM SP WHERE P# ='p۲')
```

مثال:نام تهیه کنندگانی را استخراج کنید که وضعیت آنها از همه تولید کنندگان ساکن c۲ بزرگتر باشد.

```
SELECT  sname
FROM  S
WHERE   status  >All (SELECT   status  FROM  S
                      WHERE     city='c۲' )
```

مثال بعد تفاوتش با این مثال این است که بالایی با تک تک مقایسه می کند ولی پایین یک عدد می شود.

```
SELECT  sname
FROM     S
 WHERE  status   > (  SELECT Avg (status) FROM  S)
```

مثال:شماره تهیه کنندگانی را به دست آورید که مقدار وضعیت آنها کوچکتر از مقدار میانگین وضعیت هاست.

```
SELECT     S#
FROM   S
WHERE   Status  < (SELECT   AVG  (Status)
```

عملگرهای مجموعه ای در select

عملگرهای مجموعه ای تعریف شده در جبر رابطه ای در SQL پیاده سازی شده اند.عمل اجتماع با دستور UNION ، عمل اشتراک با INTERSECT و عمل تفاضل با EXCEPT پیاده سازی شده است.

در اینجا هم باید همتایی داده ها رعایت شود. یعنی باید تعداد ستونها و همچنین دامنه هـای سـتونهای دو جدول با هم برابر باشند.

مثال: شماره قطعاتی را بیابید که یا وزن آنها بیش از ۱۶ باشد یا توسط S۲ تهیه شده باشند یا هر دو شرط را دارا باشند.

```
SELECT P#
FROM P
WHERE weigh>١٦
UNION
SELECT P#
FROM  SP
 WHERE S#=S٢
```

EXISTS در SELECT

فرم کلی آن به صورت exists (select * from…) است.چنین عبارتی به مقدار "درست" ارزیابی می شود، اگر و فقط اگر مجموعه حاصل از ارزیابی پرس و جوی داخلی select * from… تهی نباشد.
در واقع هر پرس و جویی که با استفاده از IN قابل تنظیم باشد با استفاده از exists نیز قابل تنظیم است ولی عکس این معنا، درست نیست.Exists وجود سطر و not exists عدم وجود سطر را بررسی می کند:

مثال:اسامی تهیه کنندگان قطعه P۲ را بیابید:

```
SELECT  sname
FROM S
WHERE exists (SELECT * FROM SP
                WHERE S#=S.S# AND P#='P٢')
```

دستور Select برای ایجاد جدول

با این دستور می توان به صورت زیر جدول جدیدی را از فیلدهای جدول موجود ایجاد کرد:

```
SELECT [predicate] {*| Table.*| [Table.] Field١ [as Alias١] [,...]
INTO TableName
FROM Tableexpr [,…] ] IN ExternalDatabase]
[Where [ شرط]]
[Group by …]
[Order by…]
[having…]
```

پارامترهای این دستور همانند پارامترهای دستورات Select است و پارامتر TableName نام جدولی است که باید ایجاد گردد.
به عنوان مثال دستورات زیر را در نظر بگیرید:

```
SELECT Fname,lname INTO St١
FROM student
WHERE  stNo>="١١"
ORDER BY lname
```

این دستور ، جدول St۱ را از فیلدهای نام و نام خانوادگی جدول Student ایجاد می کند و رکوردهایی را که شماره دانشجویی آن ها بزرگ تر یا مساوی "۱۱" باشد ، بر حسب نام خانوادگی مرتب کرده و در جدول St۱ کپی می نماید.

ورود داده

در زبان SQL با استفاده از دستور INSERT داده های مورد نظر در یک یا چند جدول وارد می شوند.قالب این دستور به صورت زیر می باشد:

```
INSERT INTO Table-Name (Field۱,Field۲,…,Fieldn)
VALUES(value۱,value۲,…,valuen)
```

پارامتر Table-Name نام جدولی را مشخص می کند که باید داده ها به آن اضافه بشود. پارامترهای Field۱,Field۲,…,Fieldn فیلدهایی را مشخص می کنند که داده ها باید در آن قرار گیرند و value۱,value۲,…,valuen مقادیری را مشخص می نمایند که باید در فیلدهای جدول قرار گیرند.به عنوان مثال: دستورات زیر را در نظر بگیرید:

```
INSERT INTO Student(StNo,Fname,Lname)
VALUES (۱۵,' رضا',' رحمانی')
```

این دستور رکوردی را به جدول Student اضافه می کند و مقادیر"۱۵" ، "رضا" و" رحمانی" را به ترتیب در فیلدهای شماره دانشجویی،نام و نام خانوادگی آن ها قرار می دهد.

مثال:

قطعه P۸ را با مشخصات ('P۸,Pn۸,'Pink',۱۴,'C۸') در جدول P درج کنید.

```
INSERT INTO P
VALUES (P۸,Pn۸,'Pink',۱۴,'C۸')
```

تذکر:چون لیست فیلدها نوشته نشده است منظور تمام فیلدهاست.

اضافه کردن داده ها از جدولی به جدول دیگر

با استفاده از دستور Insert می توان رکوردهای یک جدول را به جدول دیگری اضافه نمود.دستور Insert به صورت زیر به کار می رود:

```
INSERT INTO Table-Name۱ (Field۱,Field۲,…,Fieldn)
SELECT [*|(Field۰۱, Field۰۲,…, Field۰n)
FROM Table-Name۰۱
[شرط WHERE]
```

در این کاربرد،Table-Name۱ نام جدولی را مشخص می کند که رکوردها باید به آن اضافه شوند. Field۱,Field۲,…,Fieldn نام فیلدهایی از جدول Table-Name۱ هستند که باید مقادیر فیلدهایی Field۰۱,

Field۰۲,..., Field۰n از جدول Table-Name۰۱ به آن اضافه گردند و Table-Name۰۱ نام جدولی را تعیین می کند که باید رکوردها از آن به جدول Table-Name۱ اضافه گردند.

مثال: دستورات زیر را در نظر بگیرید:

```
INSERT INTO St۱(StNo,Fname,Lname)
SELECT (StNo,Fname,Lname)
FROM Student
WHERE StNo<="۱۲"
```

این دستور مقادیر شماره دانشجویی،نام و نام خانوادگی رکوردهایی را از جدول Student به جدول St۱ اضافه می کند که شماره دانشجویی آن ها کوچک تر یا مساوی"۱۲"باشد.
اگربخواهیم همه فیلد ها را مقدار دهی کنیم نیازی به نوشتن نام فیلد ها نیست .

تغییر مقدار داده:

دستور UPDATE برای ویرایش کردن یا اصلاح مقادیر جدول به کار می رود .برای یافتن ویژگی مورد نظر باید از طریق کلید اصلی رکورد مورد نظر جستجو شده و سپس عمل تغییر را اعمال نمایید.فرمت این دستور به صورت زیر استفاده می شود:

```
UPDATE Table_Name
SET Field۱=Value۱, Field۲=Value۲,..., Fieldn=Valuen,
[شرط WHERE]
```

در این کاربرد ،Table_Name نام جدولی است که باید اطلاعات آن اصلاح شود و Field۱ ، Field۲ ،...و Fieldn نام فیلدهایی را مشخص می کند که مقادیرValue۱ ، Value۲ ،...وValuen باید در آن ها قرار گیرد.به عنوان مثال دستورات زیر را در نظر بگیرید:

```
UPDATE Student
SET  StNo="۱۲", Lname"بابلی"
WHERE  StNo=' '
```

این دستور ، مقادیر "۱۲" و"بابلی" را به ترتیب در فیلدهای StNo و Lname جدول Student قرار می دهد که مقدار StNo آنها خالی باشد.

مثال :در جدول P به وزن قطعاتی که قرمز رنگ هستند ۵ کیلو اضافه کنید .

```
UPDATE   P
SET Weight = Weight +۵
WHERE  P. color = 'red'
```

حذف داده

به وسیله دستورDELETE می توان بعضی از سطر ها را طبق شرط های خاصی حذف کرد.قالب کلی این دستور به صورت زیر می باشد:

```
DELETE FROM Table_Name
[WHERE شرط]
```

یا

```
DELETE  FROM Table_Name۱
[WHERE Field۱][(SELECT  Field۲ FROM Table_Name۲)
 [WHERE شرط ]])
```

در این کاربرد Table_Name۱ نام جدولی است که باید رکوردهـای آن حـذف شـوند، Field۱ ، فیلـدی از Table_Name۱ است که باید با Field۲ ازTable_Name۲ مقایسه گردد و Table_Name۲ ، جدولی است کـه شرط بر اساس آن تعریف می گردد.

مثال: دستورات زیر را در نظر بگیرید:

```
DELETE FROM Student
WHERE StNo="۱۵"
```

رکوردی از جدول Student را حذف می کند که شماره دانشجویی آن "۱۵" باشد.

مثال:

```
DELETE FROM Grade
SELECT StNo =[SELECT stNo FROM Student]
 WHERE Lname ="رضا"
```

رکوردهایی را از جدول Grade حذف می کند که نام خانوادگی آن ها در جدول Student ،" رضا" باشد.

نرمال تر سازی رابطه ها

یکی از مهترین فرآیند ها در طراحی بانک اطلاعاتی نرمالسازی است . یک سئوال خیلی مهم برای طراحی بانک این است که با توجه به موجودیت ها و ارتباط بین آنها چند تا جدول باید طراحی کرد و در هر جدول چه فیلدهای باید درست کنیم نرمال سازی به این سئوال ها جواب می دهد . فرض کنید فیلد Status را از جدول S به جدول SP منتقل کنیم در این صورت فیلد Status برای تولید کنندگان به دفعات تکرار می شود. می توانیم جداول S وSP وP را در یک جداول بریزیم در این صورت بعضی سوالات ما نیازی به جوین و پیوند طبیعی ندارد .

ترکیب این سه جدول مشکلاتی رادر بر دارد البته باعث میشود که ضرب کارتزین و عمل جوین انجام نشود.

مشکلات:

الف) افزونگی داده ها (DATA REDUNDANCY) هر تولید کننده یا هر قطعه چندین بار نام آن تکرار شده است .

ب) وجود افزونگی در جدول باعث آنو مالی در تغییر داده ها می شود مثلا تغییر دادن شهر S1 مستلزم جستجو درتمام جدول است .

ج) برای نشان دادن بعضی از اطلاعات از مقادیر تهی NULL VALUE استفاده شده است.

وابستگی تابعی (FD[۳۳])

در تئوری نرمال تر سازی،مفهوم وابستگی تابعی از اهمیت خاصی برخوردار است.صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی دارد و می نویسیم: $R.X \rightarrow R.Y$ اگر و فقط اگر در طول حیات رابطه ، به هر مقدار از X در رابطه R ، دقیقاً یک مقدار Y از رابطه R متناظر باشد.

مثال:در بانک تهیه کنندگان - قطعات ، هر یک از صفات خاصه Sname ، Status و City از رابطه با صفت خاصه S# از همین رابطه ، وابستگی تابعی دارد زیرا به هر مقدار از S# در این رابطه فقط یک قدار از Sname، یک مقدار از Status و یک مقدار از City متناظر است:

$S.S\# \rightarrow S.Sname$

$S.S\# \rightarrow S.Status$

$S.S\# \rightarrow S.City$

یا بطور خلاصه می توان نوشت: $S.S\# \rightarrow S.(Sname, Staus, City)$

S#	Sname	Status	City
s۱	sn۱	۲۰	c۲
s۲	sn۲	۱۰	c۳
s۳	sn۳	۳۰	c۳

[۳۳] function dependency

جدول تولیدکنندگانS

s۴	sn۴	۲۰	c۲
s۵	sn۵	۳۰	c۱

جدول قطعات

P#	Pname	color	weight	city
p۱	nut	red	۱۲	c۲
p۲	bolt	green	۱۷	c۳
p۳	screm	blue	۱۷	c۴
p۴	screw	red	۱۴	c۲
p۵	cam	blue	۱۲	c۳
p۶	Cog	red	۱۹	c۲

جدول sp

S#	P#	Qty
s۱	p۱	۳۰۰
s۱	p۲	۲۰۰
s۱	p۳	۴۰۰
s۱	p۴	۲۰۰
s۱	p۵	۱۰۰
s۱	p۶	۱۰۰
s۲	p۱	۳۰۰
s۲	p۲	۴۰۰
s۳	p۲	۲۰۰
s۴	p۲	۲۰۰
s۴	p۴	۳۰۰
s۴	p۵	۴۰۰

در جدول SP ستون Qty به ستون S#وابستگی تابعی ندارد زیرا ممکن است در ازاء یک S# مثلا"S1چند تا Qty دیده شود در جدول SP ستون Qty به هر دو ستون S#,P# وابستگی تابعی دارد.
نکته:اگر x کلید اصلی باشد حتی کلید کاندید باشد هر صفت خاصه دیگر از این رابطه الزاما" با xوابستگی تابعی دارد.

وابستگی تابعی کامل(FFD[۳۴])

صفت خاصه Y از رابطه R با صفت خاصه X از رابطه R وابستگی تابعی کامل دارد اگر Y با X وابستگی تابعی داشته باشد ولی با هیچ یک از زیر مجموعه های X وابستگی تابعی نداشته باشد.صفت خاصه X الزاماً باید مرکب باشد.
مثال :در رابطه S ،صفت خاصه City با صفت خاصه مرکب (S#,Status) وابستگی دارد:

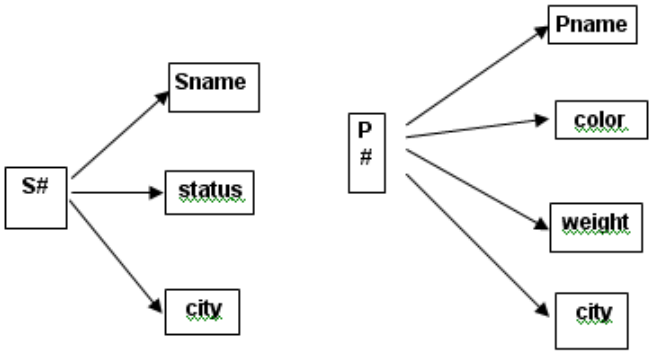
$$(S\#, Status) \rightarrow City$$

ولی این وابستگی کامل نیست، زیرا داریم: $S\# \rightarrow City$

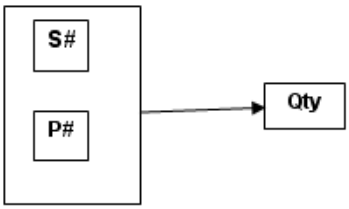
یعنی City با یکی از زیر مجموعه های (S#,Status)،وابستگی تابعی دارد.در رابطه SP ستون Qty به صفات خاصه S#,P# وابستگی تابعی دارد یعنی در ازاء هر S#,P# فقط یک Qty وجود دارد اما ستون Qty به هر
یک ازستون های S#,P#جداگانه وابستگی تابعی ندارد بهمین دلیل وابستگی تابعی کامل بین S#,Qty, P#وجود دارد . $(S\#, P\#) \rightarrow Qty$

نمودار وابستگی تابعی

این نمودارها وابستگی تابعی یک بانک را نشان می دهند:



[۳۴] full function dependency



سطوح نرمال سازی

سطوح مختلف نرمال سازی به شکل مقابل است.

۱NF

۲NF

۳NF

BCNF

٤NF

٥NF

DR NF

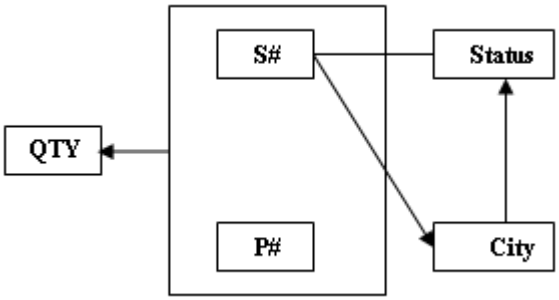
فرم ۱NF

رابطه R , ۱NF است اگر تمام صفات خاص آن اتومیک باشند یعنی تجزیه پذیر نباشند مثلاً جدولی که یک فیلد تاریخ دارد که خود از سه فیلد کوچکتر (سال , ماه , روز) تشکیل می شود INF نیست.

این تعریف صرفاً می گوید که هر رابطه نرمالی،۱NF است.رابطه ای که فقط۱NF باشد ساختاری دارد که به دلایلی چند نامطلوب است.برای نشان دادن جنبه های نامطلوب رابطه ۱NF فرض می کنیم که اطلاعات مربوط به تهیه کنندگان و قطعات به جای آنکه در دو رابطه مجزای S و SP باشند، در رابطه واحدی وجود داشته باشند بنام first.

$$first(S\#, Status, City, P\#, QTY)$$

در این مثال محدودیتی را اعمال می کنیم:صفت خاصه Status با صفت خاصه City وابستگی تابعی دارد.یعنی وضعیت یک تهیه کننده از روی شهر او تعیین می شود.کلید اصلی این رابطه ، صفت خاصه مرکب(S#,P#) است.شکل زیر نمودار FD های رابطه را نشان می دهد:



این رابطه در عملیات ذخیره سازی آنومالی دارد.مثلاً نمی توان ایـن واقعیـت را کـه تهیـه کننـده خاصی در شهر ساکن است در بانک درج کرد تا زمانی که ندانیم چه قطعه ای تهیه کرده است.

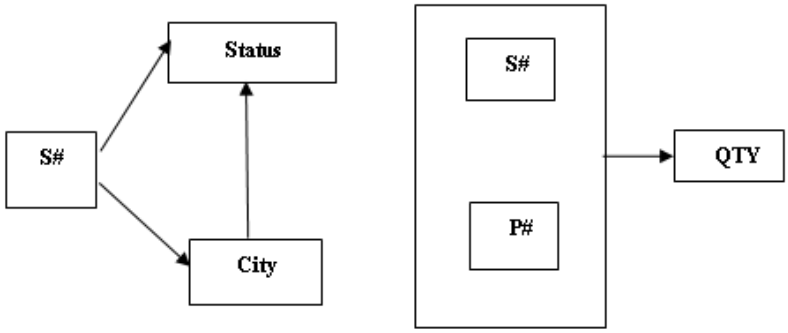
دلیلش این است که مقدار کلید اصلی را نداریم (کلید اصلی رابطه first صفت خاصه مرکـب (#P و #S) است.

برای رفع مشکلات آنومالی باید رابطه first را به دو رابطه زیر تبدیل کنیم

$\sec ond(S\#, Status, City)$

$SP(S\#, P\#, QTY)$

نمودار وابستگی این دو رابطه چنین است:



فرم دوم نرمالسازی ۲NF

رابطهR , NF ۲ است اگر فقط ۱NF باشد و هر صفت خاصی غیر کلید با کلید اصلی وابستگی تابعی کامل داشته باشد.

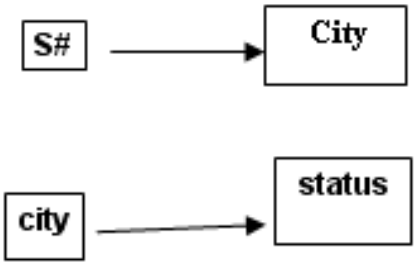
رابطه های second و SP هر دو در صورت دوم نرمال هستند.کلید ایـن رابطـه هـا بـه ترتیـب #S و (S#,P#) است.جدول ۲NF,first نیست زیرا city, status فقط به#S وابسته اند و ترکیـب #S,P# کـه همـان کلید اصلی است وابستگی تابعی ندارد.

اما رابطه second هنوز آنومالیهایی دارد مثلاً نمی توان این اطلاع را که یک شهر خـاص دارای یـک مقدار مشخص وضعیت است درج کرد.زیرا مقدار کلید اصلی نداریم.برای از بین بـردن مشـکل جـدول second آن را به رابطه هایی تجزیه می کنیم.

یک تجزیه ممکن رابطه (S#,City,Status) چنین است:

$$\begin{cases} SC(S\#, City) \\ CS(City, Status) \end{cases}$$

نمودار وابستگی های این دو رابطه به صورت زیر می باشد:



برای نرمال سازی باید نمودار وابستگی را رسم کنیم.

راهنمایی :برای نرمال سازی سطح ۲

۱- هر بخش از کلید اصلی را که صفت وابسته دارد با آن صفتها کنار می گذاریم.

۲- کل کلید اصلی را با صفتهای باقیمانده کنار هم می گذاریم.

۳- صفتهای که سایر وابستگی ها را نتیجه می دهند در مرحله ۲ تکرار شوند. (وابستگی درونی)

تعریف وابستگی با واسطه

اگر صفت خاصه B از رابطه R با صفت خاصه A از همین رابطه وابستگی داشته باشد و صفت خاصه C از همین رابطه با صفت خاصه B وابستگی داشته باشد و A با B وابستگی نداشته باشد آنگاه C با A وابستگی با واسطه دارد.

$$\begin{matrix} A \to B \\ B \to C \end{matrix} \Rightarrow A \to C$$

مجموعه پوششی وابستگی

اگر f یک مجموعه از وابستگی های تابعی باشد.آنگاه مجموعه تمام وابستگی های تابعی را که از آن نتیجه می شود مجموعه پوششی f می نامییم و آن را با f^+ نمایش می دهیم.

قواعد برای استخراج مجموعه پوششی

- بازتاب reflexivity

اگر B زیر مجموعه A باشد آنگاه $A \to B$ وابسته است.

- افزایش augmentation

اگر $A \to B$ و C یک صفت باشد $AC \to BC$

- انتقال transitivity

اگر $A \to B$ و $B \to C$ آنگاه $B \to C$

- **اجتماع**

اگر $A \to B$ **و** $A \to C$ **آنگاه** $A \to BC$

- **ترکیب composition**

اگر $A \to B$ **و** $C \to D$ **آنگاه** $AC \to BD$

- **تجزیه decomposition**

اگر $A \to BC$ **باشد آنگاه** $\begin{matrix} A \to B \\ A \to C \end{matrix}$

با استفاده از این قوانین می توان رابطه پوششی را بدست آورد.

قواعد زیر برای بهینه کردن یک مجموعه وابستگی است.

۱- سمت راست هر وابستگی فقط یک صفت باشد.

۲- هر صفتی که f^+ **را تغییر نمی دهد از سمت چپ حذف شود (فیلدهایی که وابستگی نمی آورند حذف شوند.)**

۳- وابستگی های تکراری و اضافی حذف شوند.

مثال : یک بانک اطلاعاتی با فیلدهای زیر و این روابط موجود است.مراحل بهینه سازی را انجام دهید.

$$R = \{u, v, w, x, y, z\}$$
$$= \{u \to xy, x \to y, xy \to zv, u \to zv\}$$
$$= \{u \to x, u \to y, x \to y, xy \to zv, u \to zv\}^*$$
$$\begin{cases} x \to y \\ xy \to zv \end{cases} \Rightarrow x \to zv \Rightarrow \begin{cases} x \to z \\ x \to v \end{cases}$$
$$^* = \{u \to x, u \to y, x \to y, x \to zv, u \to zv\}$$
$$= \{u \to x, u \to y, x \to y, x \to z, x \to v, u \to z, u \to v\}$$

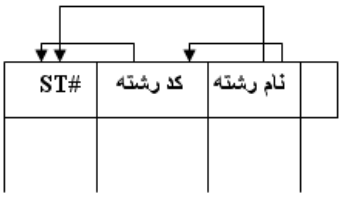
صورت سوم نرمال سازی۳NF

۱- رابطه در۲NF است.

۲- وابستگی انتقالی نداشته باشد.

وابستگی انتقالی

یک وابستگی در مجموعه صفت غیر کلیدی به یکدیگر. زیرا هر یک از آنها به کلید اصلی وابسته است.هدف ۳NF این است که وابستگی های انتقالی را بردارد. شکل زیر مثالی از وابستگی انتقالی را نشان می دهد.



مثال:بانک زیر را در نظر بگیرید.با توجه به مباحث نرمالسازی ایرادهای وارده بر آن را در پایین مشخص نموده ایم!

نمره	شماره دانشجویی	نام استاد	نام درس	واحد	کد دانشکده	شماره درس
۱۷	۱۰۰۱	رحمانی	پایگاه داده ها	۳	۱	۱۰۲۴
۱۴	۱۰۰۲	رحمانی	پایگاه داده ها	۳	۱	۱۰۲۴
۱۳.۵	۱۰۰۳	رحمانی	پایگاه داده ها	۳	۱	۱۰۲۴
۱۹	۱۰۰۴	رحمانی	پایگاه داده ها	۳	۱	۱۰۲۴

۱- وجود افزونگی داده ها در جدول فوق (واحد، شماره درس، نام استاد، کد دانشکده، نام درس).نیازی به وجود این صفات خاصه در یک بانک نیست.بلکه می توان آنها را در بانکهای مربوط به خود قرار داد و از طریق فیلد رابطه ای این مشکل را حل کرد.

۲- بی نظمی در این بانک برای وارد کردن نمره دانشجو چون باید تمام مشخصات وارد شود و در غیر اینصورت در عملیات جستجو دچار مشکل می شویم.

۳- مقادیر تهی است یا در ادغام جداول به ناچار برخی از صفات خاصه تهی می شود که هنگام جستجو کردن باعث مشکل خواهد شد.

وابستگی چند مقداری $A_1A_2...A_n \rightarrow B_1B_2...B_n$

اگر برای هر دو رکورد u وt که در تمام مقادیر A_j مشترک هستند رکورد دیگری مانند v وجود داشته باشد که:

۱- در مقادیر A با u وt مشترک باشند.

۲- در مقادیر B با t مشترک باشند.

۳- در تمام ستونهای دیگر R با u مشترک باشند.

مثال:جدول زیر دارای وابستگی چند مقداری است.

$\leftarrow A \rightarrow$		$\leftarrow B \rightarrow$		
Pname	**Col_name**	**city**	**Crs_name**	**Text**
احمدی	کامپیوتر	تهران	ساختمان داده ها	جعفر نژاد
احمدی	برق	مشهد	ساختمان داده ها	جعفر نژاد
احمدی	کامپیوتر	تهران	پایگاه داده ها	رانکوهی
احمدی	برق	مشهد	پایگاه داده ها	رانکوهی
احمدی	کامپیوتر	تهران	برنامه نویسی C++	هربرت شیلد
احمدی	برق	مشهد	برنامه نویسی C++	هربرت شیلد

صورت چهارم نرمال سازی۴NF

رابطه R در ۴NF است در صورتی که اگر وابستگی چند مقداری $A_1A_2...A_n \rightarrow B_1B_2...B_n$ در R وجود داشته باشد،آنگاه $A_1A_2...A_n$ ابر کلید R می باشد.(اگر برای تمام صفات B دررابطه R داشته باشیم $A \rightarrow B$ آنگاه A را ابر کلید R می نامیم)

مثال: فرم ۴NF را بر روی جدول صفحه قبل اعمال کنید.(جداول یکی شامل ستونهای A وB دیگری خود A با سایر ستونها ست.)

$R_1(Pname\ ,col\ \_\ name\ ,city)$

$R_2(Pname\ ,crs\ \_\ name\ ,text)$

Pname	Col_name	city
احمدی	کامپیوتر	تهران
احمدی	برق	مشهد

Pname	Crs_name	Text
احمدی	ساختمان داده ها	جعفر نژاد
احمدی	پایگاه داده ها	رانکوهی
احمدی	برنامه نویسی C++	هربرت شیلد

فرم BCNF:

تبدیل کردن جداول به ۳NF باعث افزونگی می شود که افت کارایی سیستم را به دنبال دارد زیرا در این حالت باید عمل پیوند داده ها به دفعات استفاده شود.فرم ۳NFدر موارد زیر ممکن است مشکل بوجود آورد:

الف)وقتی رابطه دارای چند کلید کاندید باشد.

ب)وقتی که کلیدهای کاندید رابطه مرکب باشد.

ج)وقتی که کلید های کاندید با یکدیگر اشتراک صفت داشته باشند.

جدولی در BCNF است که ستونهای آن ، فقط به کلید کاندیدش وابستگی تابعی داشته باشد.