بسمه تعالی

اصول طراحی پایگاه داده
(بانک اطلاعاتی)

توجه:

این جزوه درسی توسط یکی از دانشجویان تایپ شده و صرفا جهت صرفه جویی و استفاده بهینه از زمان کلاس ارائه می گردد. در طول ترم مطالب تکمیلی و مثالهای دیگری علاوه بر مطالب این جزوه ارائه خواهد شد که باید توسط دانشجویان به این جزوه اضافه گردد و توصیه می گردد بصورت یکرو چاپ شود.

منابع آزمون میان ترم و پایان ترم شامل کلیه مطالب تدریس شده به همراه قسمت هایی از کتاب مرجع است که در سایت قرار خواهد گرفت، لذا نباید صرفا به این جزوه درسی اکتفا نمود.

تعریف پایگاه داده : جایی است که در آن دادهای مربوط به یک سیستم بصورت مجتمع و یکپارچه ، مبتنی بر یک ساختار مشخص و با حداقل افزونگی ذخیره شده وتحت کنترل یک سیستم متمرکز بوده و مورد استفاده یک یا چند کاربر بصورت همزمان و اشتراکی قرار می گیرد.

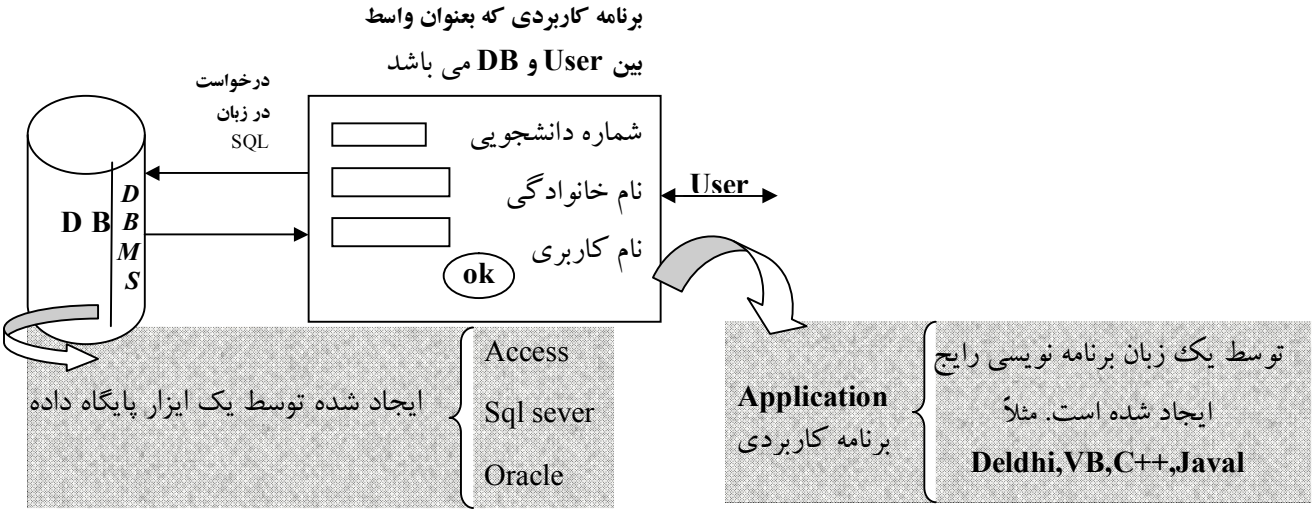
در ادامه برخی از اصطلاحات بکار رفته در این تعریف توضیح داده خواهد شد:

یکپارچگی پایگاه داده بیشتر از نظر منطقی اهمیت دارد نه لزوماً فیزیکی (فایلی). (بطور مثال داده های مربوط به هر کاربر درسایت **Yahoo** ممکن است در سرور های مختلفی قرار داشته باشد اما برای کاربران این قضیه قابل مشاهده نیست).

افزونگی به معنی **(Redundency)** تکرار بی رویه داده ها می باشد. مثلا اگر در یک سیستم لازم است اطلاعاتی درباره دانشجویان از قبیل کد دانشجویی، نام و نام خانوادگی ، کد ملی و غیره ذخیره شود، این اطلاعات فقط یکبار ذخیره شده باشد و در موارد مختلف قابل بازیابی باشد.

. در مورد ساختار مشخص نیز باید گفت هر پایگاه داده برای قرار دادن داده های مربوطه از ساختاری استفاده می کند که در صفحات بعد برخی از این ساختار ها معرفی خواهد شد. ساختار جدول نوعی از این ساختارهاست.

با توجه به تعاریف فوق، داده های ذخیره شده در یک یا چندین فایل نمی توانند پایگاه داده باشند چون اصولاً سیستم کنترل متمرکزی وجود ندارد که مفهوم یکپارچگی منطقی را ایجاد کند و یا درخواستها را گرفته و پاسخ دهد



با توجه به شکل فوق هر پایگاه داده شامل نرم افزاریست بنام **DBMS**
(*Data Base Management System*): سیستم مدیریت پایگاه داده .

مثلاً ابزار **SQL Server** که در درس آزمایشگاه پایگاه داده تدریس می شود دا رای **DBMS** است. مثال دیگر از **DBMS** نرم افزاریست که در دستگاههای خود پرداز **ATM** مسئول گرفتن درخواستها و دادن پاسخ است.

اجزاء اصلی بانک اطلاعاتی:

۱- **سخت افزار :** کلیه سخت افزارهایی که در سیستم بانک اطلاعاتی وجود دارند مانند سخت افزار ذخیره سازی داده ها (هارد دیسک)، سخت افزار پردازش داده ها (پردازشگر)ف سخت افزار انتقال داده ها (کانالهای ارتباطی مثل خط تلفن و فیبر نوری و...)

۲- **نرم افزار** : نرم افزارهایی که برای ایجاد بانک اطلاعاتی و کنترل داده ها وجود دارند . تمام ابزارهای ایجاد بانک اطلاعاتی از قبیل **Access, SQL Server , Oracle** دراین دسته قرار دارند.

۳- **کاربران :** دسته های مختلفی از کاربران در این دسته قرار دارند. مثالی از آن ها عبارتند از: ل **DA** (مدیر داه ها) –**DBA** (مدیر بانک اطلاعاتی)- **End user** (کاربر نهایی)

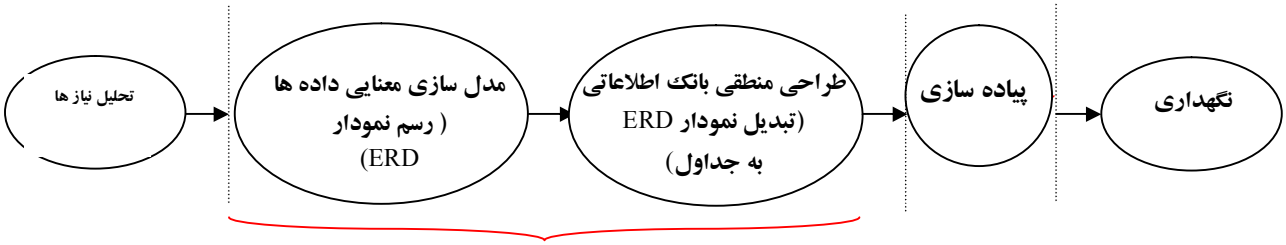
مدیر داده ها (**Data Administrator**) شخصی است که وظیفه آن تعیین سیاستهایی در مورد بانک اطلاعاتی است. همچنین داده هایی که باید در بانک اطلاعاتی ذخیره شود را مشخص می کند و معمولاً یک شخص می باشد .

مدیربانک اطلاعاتی (**DataBase Administrator**) که معمولاً یک تیم می باشند و وظیفه پیاده سازی بانک اطلاعاتی و سیاستهای تعیین شده از طرف مدیر داده ها را به عهده دارند.

کاربر نهایی(**End user**) افرادی هستند که نهایتاً از سیستم استفاده خواهند کرد.

۴- **داده ها** (**Data**) : تمام داده هایی که در سیستم مورد نظر به آنها نیاز داریم و باید به نحوی ذخیره شوند . این داده ها از هر سیستمی به سیستم دیگر تفاوت دارند و معمولاً توسط مصاحبه با مشتری و پرکردن پرسشنامه و یا روش های دیگر تحلیل نیازها مشخص می گردد و در قالب نموداری بنام **ERD** مشخص می گردد.

بنابراین با تحلیل نیاز ها که بعنوان اولین مرحله از فرآیند ایجاد بانک اطلاعاتی باید انجام شود،، سیستم مورد نظر بخوبی شناسایی شده و هرگونه کاستی و اشکالی در این مرحله ممکن است مراحل بعد را تحت شعاع خود قرار دهد. در شکل زیر مراحل مختل کار نشان داده شده است:



قسمتی از مباحث درس پایگاه داده ها

مباحث مربوط به آزمایشگاه پایگاه داده ها

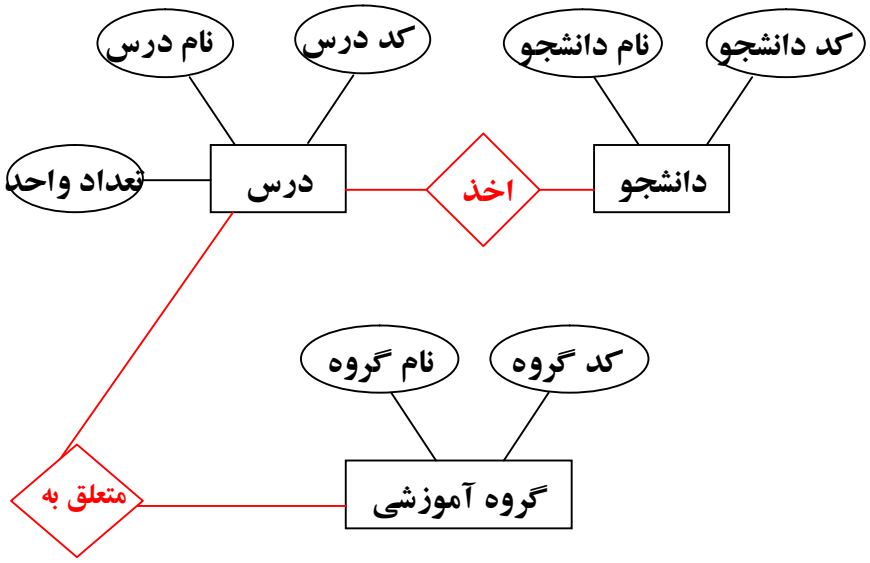
مدلسازی معنایی داده ها :

قدم اول برای اینکار شناسایی موجودیتهای سیستم می باشد. پس در مورد هر موجودیت ویژگی آن را بدست می آوریم و در قالب نموداری بنام **ERD** (نمودار روابط موجودیتها)
(Entity Relationship Digram) برخی از موجودیتها را به یکدیگر ارتباط می دهیم.

تعریف موجودیت (Entity): هرچیزی که می خواهیم در سیستم مورد نظر در مورد آن اطلاعات ثبت کنیم موجودیت نام دارد.. مثلاً در مورد سیستم دانشگاه موجودیتهایی مثل دانشجو، اساتید، دروس، تجهیزات و غیره وجود دارند که در مورد آنها با توجه به نیاز سیستم اطلاعاتی ذخیره می کنیم.
مثال:
دانشجو: کد دانشجو، نام دانشجو، رشته ، معدل و ...(صفات موجودیت یا **Attribute**)
درس : کد درس، نام درس، تعداد واحد، پیش نیازو...(صفات موجودیت یا **Attribute**)
استاد : کد استاد، نام استاد،رشته،درجه علمی، سابقه کار و...(صفات موجودیت یا **Attribute**)

موجودیت را با نماد مستطیل؛ صفات را بانماد دایره یا بیضی؛ و ارتباط را با نماد لوزی نشان می دهیم.

مثال: قسمتی از **ERD** سیستم دانشگاه



نکات زیر در شناسایی موجودیتها می تواند مفید باشد
الف) هر موجودیت معمولاً بیش از یک نمونه (**instance**) دارد. نمونه زمانی شکل می گیرد که به صفات یک موجودیت مقادیری اختصاص داده شود. بطور مثال نمونه ایی از دانشجو بصورت زیر است:

کد	نام	رشته	معدل
8873	**Mohammad**	**IT**	**19**

ب) هر موجودیت معمولاً در ارتباط با موجودیتهای دیگر می باشد. معمولاً موجودیت ایزوله (منفرد از بقیه) در نمودار نداریم.

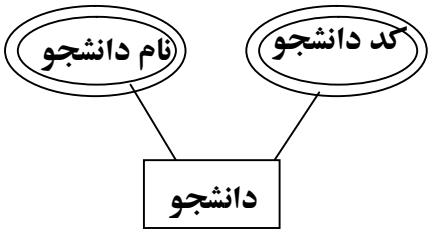
ج) معمولاً هر موجودیت بیش از یک صفت دارد.

حالتهای مختلفی در مورد صفات موجودیت وجود دارد که به آن اشاره می کنیم :

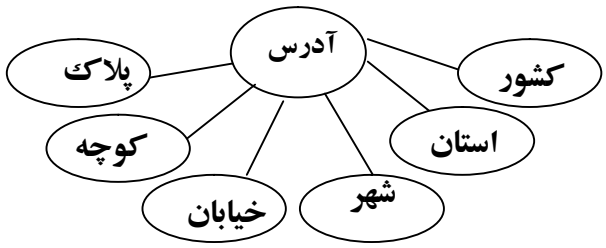
۱- **صفت شناسه:** صفت هایی هستند که مقادیر آن به ازای هر نمونه، یکتا (**unique**) هستند. یعنی مقادیر هیچ دو نمونه ایی از آن موجودیت در این صفت شناسه یکسان نیستندومعمولاً این صفت شناسه معرف آن موجودیت می باشد. مثل کد دانشجو برای موجودیت دانشجو یا کد ملی برای موجودیت دانشجو.صفت شناسه نمی تواند مقداری نداشته باشد و معمولاً هر موجودیت حداقل یک صفت شناسه دارد مگر اینکه موجودیت ضعیف باشد که بعداً به آن اشاره می کنیم. زیر صفت های شناسه معمولاً یک خط کشیده می شود. صفت شناسه نمی تواند هیچ مقدار پذیر باشد مانند کد دانشجویی و یا کد ملی دانشجو.

۲- **هیچ مقدار پذیر بودن** (Nullable): صفاتی هستند که می توانند به ازای برخی از موجودیتها مقداری نداشته باشند. مثل تلفن ، آدرس ، ایمیل.

۳- چند مقداری بودن (**Multivalue**) : صفاتی که می توانند به ازای برخی از نمونه ها چندین مقدار داشته باشند. مانند تلفن ، آدرس ، ایمیل، شماره حساب. **(**صفات چند مقداری را با دو دایره نشان می دهیم **)**



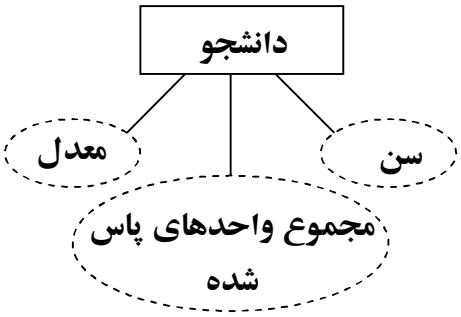
۴- **صفت مرکب** : صفتی که خود از اجزاء مختلفی تشکیل شده و ترکیب این اجزاء، آن صفت را می سازد. مثلاً در مورد آدرس که یک صفت مرکب است کافیست اجزائی از قبیل کشور ، استان ، شهر ، خیابان ، کوچه و پلاک مشخص شود.



❖ یک صفت می تواند هم مرکب باشد و هم چند مقداری.

۵- **صفت مشتق شده :** صفتی که مقادیر آن مستقیماً وارد نمی شود بلکه توسط یک فرمول و محاسبه ای روی مقادیر صفات دیگر بدست می آید مانند معدل ، مجموع واحد های پاس شده تا کنون و...

۶- صفت مشتق را با خط چین نشان می دهند.



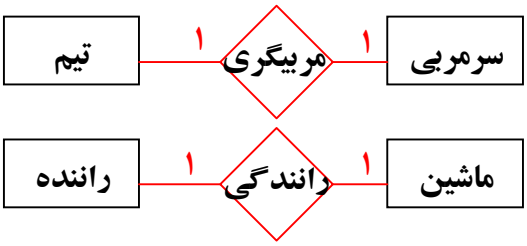
ارتباط بین موجودیتها :

ارتباط بین موجودیتها مشخصاتی دارد که در ادامه بررسی می کنیم

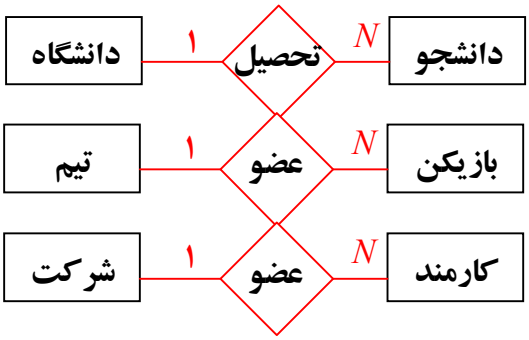
۱- **چندی ارتباط** (کاردینالیتی**cardinality**) : مشخص کننده تعداد نمونه های موجودیتهایی هستند که با یکدیگر ارتباط دارند.

حالتهای زیر وجود دارند:

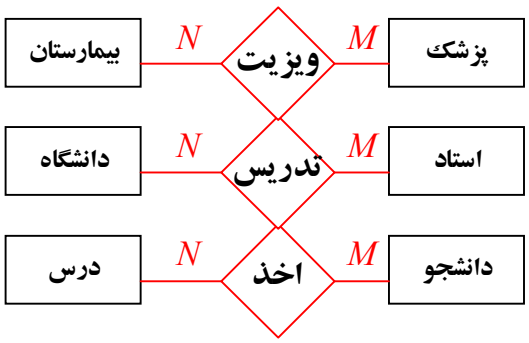
a) یک به یک (۱-۱) **one to one** : یعنی هر نمونه از موجودیت اول حداکثر با یک نمونه از موجودیت دوم ارتباط دارد و بر عکس.



b) یک به چند (**1-*N***) **one to many** : هر نمونه از موجودیت اول می تواند با چندین نمونه از موجودیت دوم ارتباط داشته باشد ولی هر نمونه از موجودیت دوم حداکثر با یک نمونه از موجودیت اول ارتباط دارد.

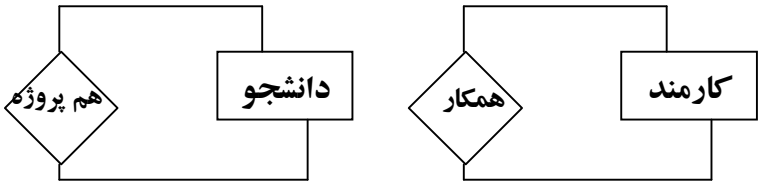


c) چند به چند(***N-M***) **many to many** : هر نمونه از موجودیت اول می تواند با چندین نمونه از موجودیت دوم ارتباط داشته باشد و برعکس.



۲- درجه ارتباط : مشخص کننده تعداد موجودیتهایی است که در آن ارتباط وجود دارد. به حالتهای زیر وجود دارند:

درجه یک: (**Unary**)



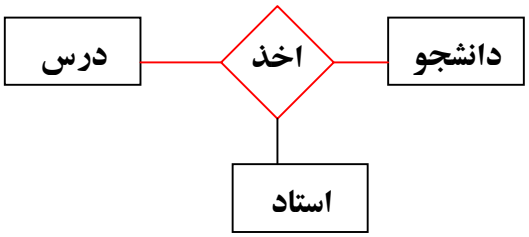
درجه دو:(**Binary**)

درجه سه: (**Ternary**)

در ارتباط درجه یک، نمونه هایی که ارتباط مورد نظر بین آنها برقرار است هر دو مربوط به یک موجودیت هستند.

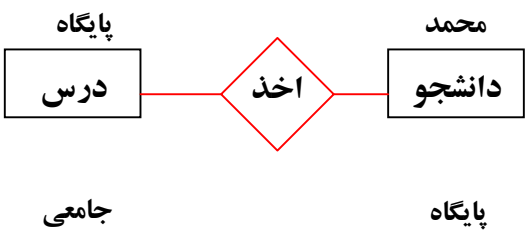
نکته : برخی از ارتباطات همزمان به سه موجودیت مربوط می شوند مانند مثال زیر

یک ارتباط درجه ۳

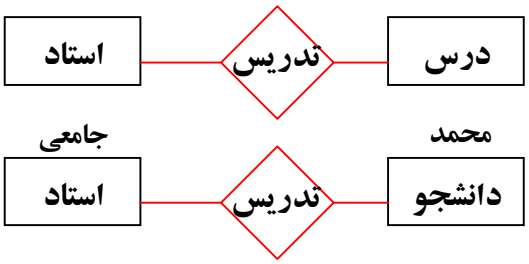


نکته : یک ارتباط درجه سه حاوی اطلاعات بیشتری نسبت به سه ارتباط درجه دو است. مثلاً جمله زیر:

" دانشجویی بنام محمد درسی بنام پایگاه را با استاد جامعی اخذ کرده است." از شکل فوق نتیجه می شود اما از شکل زیر نتیجه نمی شود.



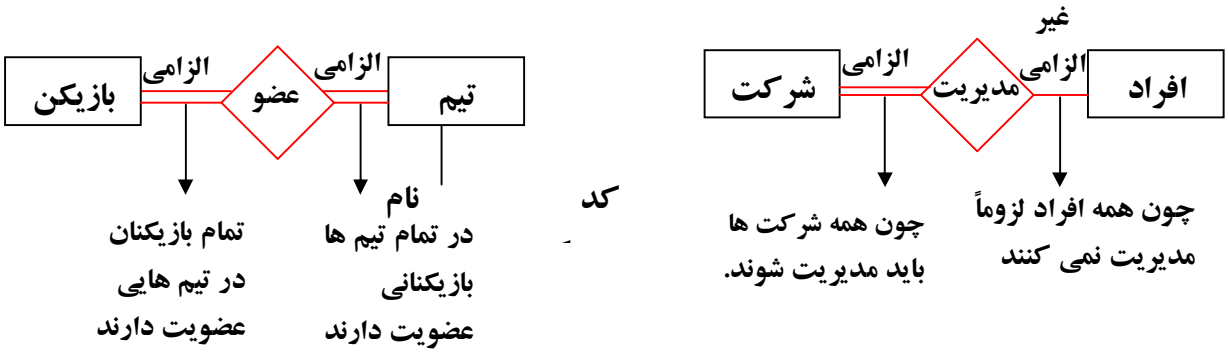
سه ارتباط درجه ۲



دام حلقه ایی (circular trap) : اصولاً دام به معنی تفسیر نادرست از نمودار **ER** می باشد که یکی از آنها دام حلقه ایست یعنی اگر از شکل دوم تفسیری مشابه شکل اول داشته باشیم دچار دام حلقه ایی شده ایم.

نکته : برخی از ارتباطات هم مثل موجودیتها می توانند دارای صفات باشند این در مواقعی پیش می آید که صفتی داشته باشیم و این صفت همزمان به موجودیتهای شرکت کننده در آن ارتباط مربوط باشد نه بطور مستقل به تک تک آنها.مثل نمره یک دانشجو در یک درس که با استاد خاصی اخذ شده است.

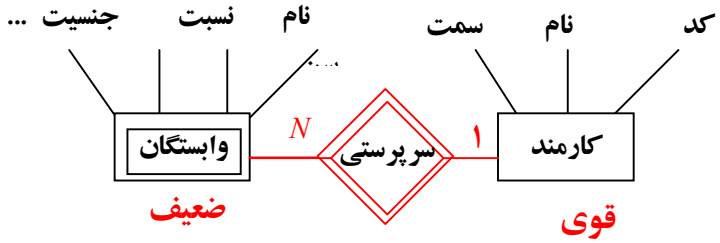
۳- نوع مشارکت موجودیت ها در ارتباط : مشارکت هر موجوریت در ارتباط می تواند الزامی یا غیر الزامی باشد. مشارکت الزامی یعنی تمام نمونه های آن موجودیت در آن ارتباط شرکت کرده باشند حتی اگر یک نمونه از آن موجودیت در آن ارتباط شرکت نکند مشارکت غیر الزامی است. مشارکت الزامی را با دو خط از موجودیت به ارتباط وصل می کنیم. مثال:



نکته: خود ارتباط هم می تواند دارای صفت باشد (زمانی که صفتی داشته باشیم که بطور همزمان به چند موجودیت مربوط باشد)

مثال:

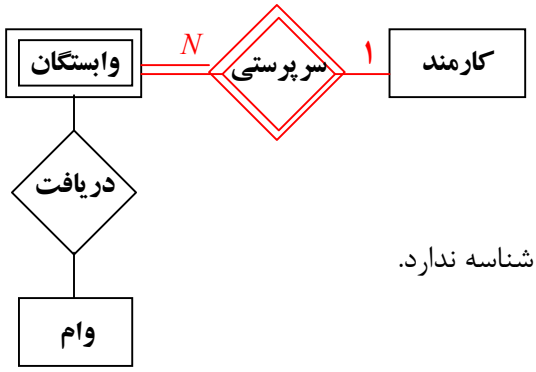
موجودیت ضعیف (**Weak Entity**) : موجودیتی است که وجودش وابسته به یک موجودیت دیگر (موجودیت قوی) باشد. یعنی با حذف موجودیت قوی از سیستم، نیازی به نگهداری نمونه های موجودیت ضعبف مربوطه نمی باشد. موجودیت ضعیف را با مستطیل دو خطی نمایش می دهیم و ارتباط بین موجودیت قوی و ضعیف را با لوزی دو خطی نمایش می دهیم. مثال:



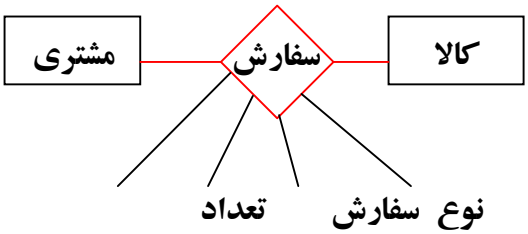
ضعیف **قوی**

چون با حذف کارمند نیازی به نگهداری اطلاعات وابستگان آن نمی باشد

نکته : قوی و ضعیف بودن نسبی است یعنی یک موجودیت ممکن است نسبت به یک موجودیت دیگر ضعیف باشد ولی نسبت به موجودیت دیگری نباشد.

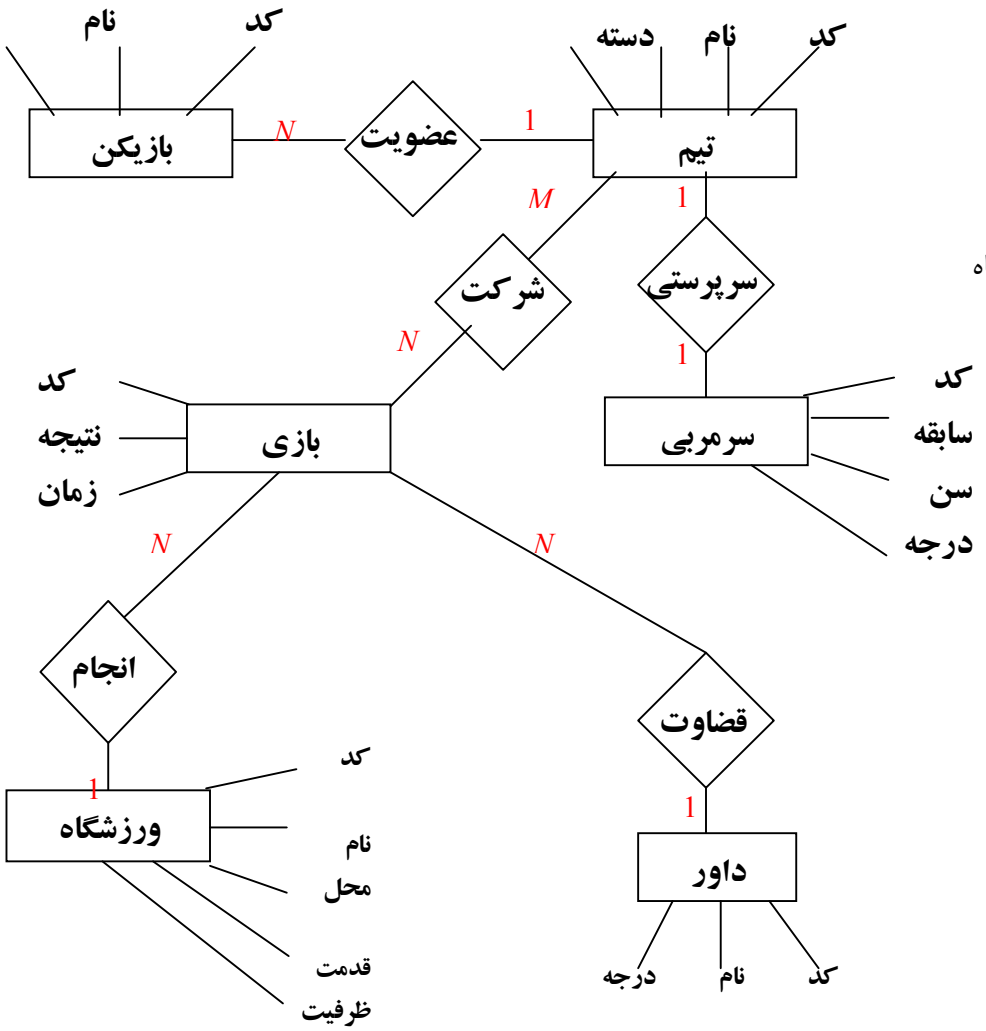


نکته : موجودیت ضعیف معمولاً از خود صفت شناسه ندارد.



قسمتی از نمودار ER سیستم مسابقات فوتبال

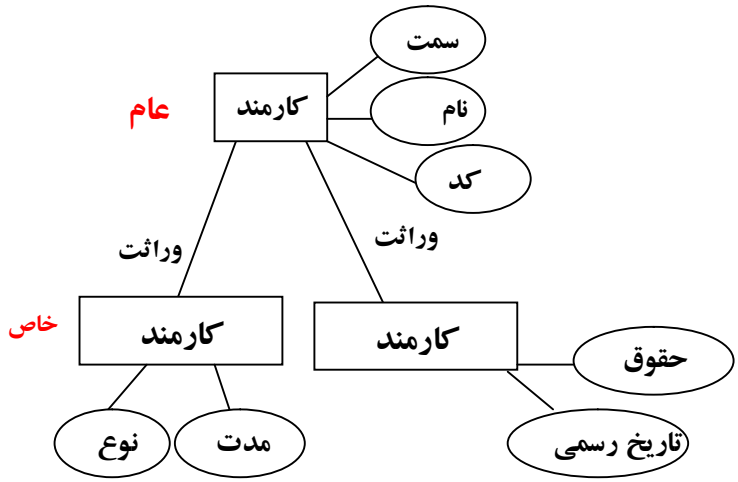
- تیم
- بازیکن
- مربی
- داور
- بازی
- ورزشگاه



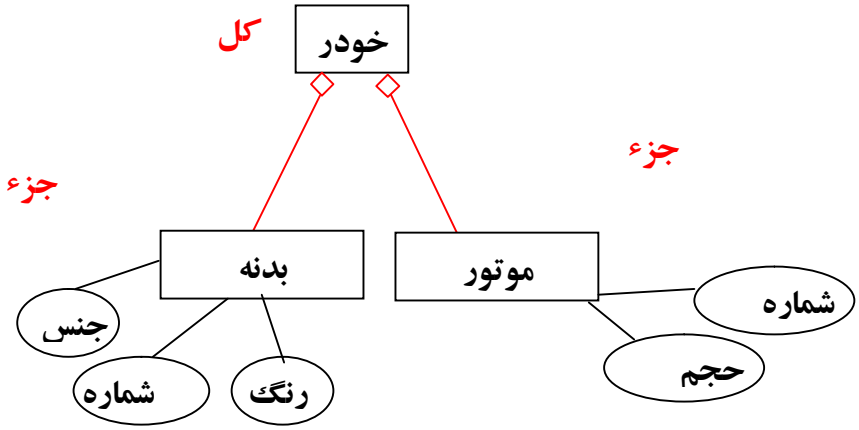
نمودار EER (**نمودار** ER **گسترش یافته**): نمودار**ER** که تاکنون بررسی کردیم برای نشان دادن مباحثی مثل وراثت و غیره کمبودهایی داشت و با اضافه کردن این نمادها این کمبود برطرف شد و **EER** نام گرفت.(**E** اول از کلمه **Extended** گرفته شده است)

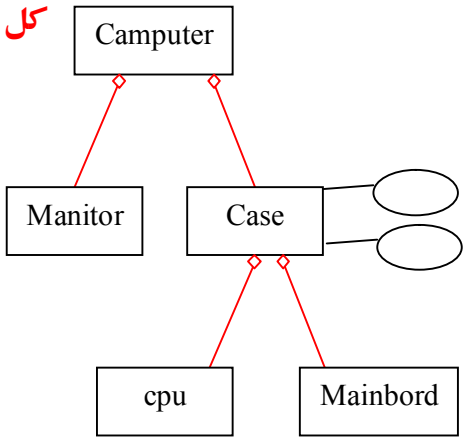
رابطه وراثت (عام- خاص): رابطه ایست بین یک موجودیت عام و انواع آن (موجودیت خاص). این رابطه زمانی لازم است که چندین نوع از یک موجودیت وجود داشته باشد که هریک از این انواع صفات خاص خودش را داشته باشند و بعلاوه دارای یکسری صفات مشترک باشند که در موجودیت عام آنها قرار گرفته و آنها این صفات را به ارث می برند.

مثالی از رابطه وراثت(**Inheritance**)



رابطه کل- جزء : رابطه ایست بین یک موجودیت و اجزاء آن که معمولاً فیزیکی است. موجودیت کل صفات خاص خود را دارد و موجودیتهای جزء هم صفات خاص خود را دارند. این ارتباط با یک لوزی کوچک در اطراف موجودیت کل نمایش داده می شود.





تمرین: یک سیستم انتخاب نموده و در مورد آننمودار **ERD** را رسم نمایید.

مثالی از این سیستم ها می تواند بصورت زیر باشد:

سیستم بیمارستان : که دارای موجودیت هایی از قبیل بیمار- پزشک - پرستار- تجهیزات- اتاق عمل می باشد.
سیستم کتابخانه : که دارای موجودیت هایی از قبیل کتاب- عضوها- پرستل می باشد
سیستم بانک : که دارای موجودیت هایی از قبیل کارمندان – مشتری ها – حساب ها – وام ها – شعبه ها می باشد.

ساختارهای داده ایی : ساختار داده ای جائیست که در آن می توان داده ها را ذخیره نمود. رایج ترین آنها ساختار رابطه ایی (جدولی) می باشد که تمامی ابزارهای امروزی از این ساختار پشتیبانی می کند و در این درس به این ساختار می پردازیم . ولی ساختارهای دیگری از قبیل سلسله مراتبی و شبکه ایی نیز قبلاً وجود داشت که در آنها به جای جدول از رکورد هایی استفاده می کردند و هر رکورد از یکسری فیلد(**field**) تشکیل شده بود و به تعداد نمونه های لازم باید رکورد ایجاد می شد . مثال

رکوردهای دانشجو

شماره دانشجویی	نام	رشته
۸۴۵۱	**Ali**	فناوری
۸۴۵۲	**Reza**	نرم افزار

رکوردهای درس

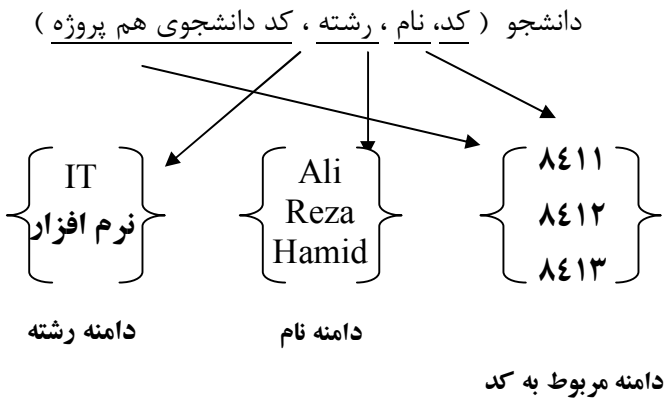
کد درس	نام درس	واحد
۱۱	پایگاه داده	۳
۱۱	پایگاه داده	۳

در مدل سلسله مراتبی فقط ارتباط یک به یک و یک به چند پشتیبانی می شد ولی در مدل شبکه ایی ارتباط چندبه چند هم قابل پیاده سازی بود.
مدل رابطه ایی (جدولی) : زیر مجموعه ایست از حاصلضرب دکارتی (کارتزین) مقادیر صفتهای آن رابطه .

یادآوری: **ضرب دکارتی دو مجموعه** A **و** B **بصورت زیر است:**

A={A1,A2}
B={B1,B2}
A*B={A1B1,A1B2, A2B1,A2B2}

مثال



تعرف دامنه (Domain): برای هر صفت از رابطه می توان مجموعه مقادیری را از پیش تعیین کرد که به آن دامنه گفته می شود. یعنی دامنه جائیست که هر صفت از رابطه از آنجا مقدار می گیرد. در مثال قبل دامنه مربوط به هریک از صفات مشخص شده اند که فقط از آنجا می توانند مقدار بگیرند و انتساب مقداری خارج از دامنه آنها مجر به خطا می گردد. رابطه ایی که روی این دامنه ها تعریف شده می تواند بصورت زیر باشد

کد	نام	رشته	کد
۸۴۱۱	**Ali**	**IT**	۸۴۱۳
۸۴۱۲	**Reza**	نرم افزار	۸۴۱۱

یک رکورد ؛ سطر؛ تاپل Tupple

- رکوردها نمونه هایی از موجودیت مربوطه هستند که به برخی از صفات آن مقداری منتسب شده است.
- در واقع پایگاه داده تشکیل شده است از یکسری جداول که با یکدیگر ارتباط دارند و این جداول از نمودار **ERD** بدست آمده اند.

کلید ها در مدل رابطه ایی : یکسری کلیدهایی در مدل رابطه ایی وجود دارند که عبارت اند از :

سوپرکلید یا ابرکلید	**Super key**
کلید کاندید	**Candidate Key**
کلید اصلی	**Primary Key**
کلید فرعی (کلید بدیل)	**Alternative Key**
کلید خارجی	**Foreign Key**

سوپر کلید (Super key): صفت یا مجموعه ایی از صفت ها که یکتایی مقدار داشته باشند(**Unique** باشند).

کددانشجو☑ نام دانشجو☒ کد،نام☑ کد ملی☑

کلید کاندید (**Candidate Key**): صفت یا مجموعه ایی از صفت ها که اولاً یکتایی مقدار داشته باشند (یعنی سوپرکلید باشند) و دوماً خاصیت کمینگی داشته باشند. خاصیت کمینگی (**Minimality**)یعنی با کمترین ترکیب ممکن یکتایی را برقرار کرده باشد.

کدملی ،کددانشجو☒ نام دانشجو☒ کدونام☒ کد دانشجو☑

باوجود اینکه یکتایی دارد کمینگی ندارد چون کد به تنهایی کافی بود و نام اضافیست.

مثال:

جدول تولید

کد تولید کننده کد کالا تعداد

QTY	P#	S#
۱۰	P1	S1
۲۰	P2	S1
۳۰	P1	S2

با فرض اینکه (**s#,p#**) تکراری نداریم به سوالات زیر پاشخ دهید؟
(یعنی هر تولید کننده هر کالایی را یکبار تولید کرده)

آیا **S#** کلید کاندید است؟خیر
آیا **P#** کلید کاندید است؟خیر
آیا(**S# ,P#**)کلید کاندیداند؟بله
آیا(**S# ,P#,QTY**)کلید کاندیداند؟خیر

جدول فوق یک کلید کاندید دارد ولی جدول قبلی دو کلید کاندید دارد که یکی کد دانشجو و دیگری کد ملی است.

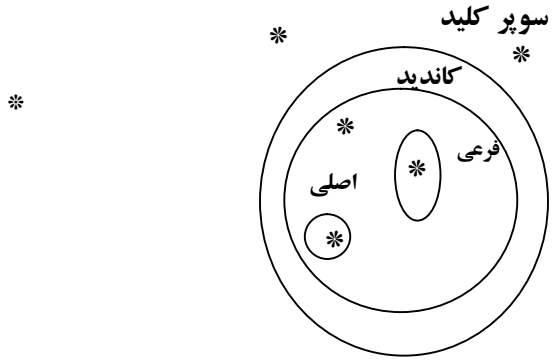
کلید اصلی (Primary Key) : یکی از کلیدهای کاندید که توسط طراح انتخاب می شود و معیار این انتخاب می تواند کوتاه بودن کلید و یا رایج بودن آن باشد . مثلاً در متال اول از بین دو کلید کاندید کد دانشجو در سیستم دانشگاه رایج تر است و بعنوان کلید اصلی انتخاب می گردد.

نکته: کلید اصلی نماینده بقیه اطلاعات نمونه موجودیت هاست . مثلاً در سیستم دانشگاه در صورتی که کد یک دانشجو را به سیستم بدهیم میتوانیم بقیه اطلاعات آن دانشجو را بدست آوریم.

کلید فرعی(Alternative Key) : کلیدهای کاندیدی که به عنوان کلید اصلی انتخاب نشده اند کلید فرعی می باشند. کلید فرعی مثال اول: : کد ملی

- هر جدول حتماً باید یک کلید اصلی داشته باشد.
- هر جدول حداقل باید یک کلید کاندید داشته باشد و می تواند حتی بیش از یک کلید داشته باشد.

شکل زیر موقعیت کلید های مختلف را نسبت به یکدیگر نشان می دهد:



کلید خارجی : صفت یا ترکیبی از صفتهای در صورتی در یک جدول کلید خارجی است که دقیقاً همین ترکیب در جدول دیگری از پایگاه داده به عنوان کلید اصلی تعریف شده باشد .
کلید اصلی (**P.k**) و کلید خارجی (**F.k**) عامل ارتباط بین جداول می باشد. (**Relationship**)
مثال:

در هنگام پیاده سازی باید بین جداولی که از نظر منطقی به یکدیگر مربوط می شوند ارتباط برقرار کرد و این کار با توجه به نمودار **ER** رسم شده و طبق اصولی که بعداً خواهیم گفت انجام می گیرد.(مبحث مربوط به تبدیل نمودار **ER** به رابطه ها).

- هر جدول حتماً یک کلید اصلی دارد(نه کمتر نه بیشتر)
- هرجدول می تواند یک یا چند کلید خارجی داشته باشد.
- مقادیر کلید اصلی نمی توانند تکراری و یا **Null** (خالی) باشند.
- مقادیر کلید خارجی می تواند تکراری و یا **Null** باشد که البته این به منطق جدول مربوط می شود.

قوانین جامعیت (Integrity Rules: قوانین جامعیت ، قوانینی هستند که برای حفظ سازگاری و درستی بانک اطلاعاتی تعریف می شوند. برخی از این قوانین بطور اتوماتیک توسط ابزارها تعریف می شوند و وجود دارند از قبیل قانون جامعیت موجودیتی و ارجاعی که بعداً توضیح خواهیم داد. برخی قوانین باتوجه به سیستم مورد نظر باید تعریف شوند که اصطلاحاً به آنها قوانین کاربری گفته می شود و می توانند از یک سیستم به سیستمی دیگر ممکن است متفاوت باشد.
مثلاً در یک دانشگاه ممکن است نمره قبولی بین صفر تا بیست و در دانشگاهی دیگر بین صفر تا صد باشد.
مثالهای دیگر از قوانین کاربری عبارت اند از:
حداقل نمره قبولی ۱۲ است . سن افراد نمی تواند منفی باشد. موجودی افراد نمی تواند کمتر از ۵/۰۰۰ باشد و غیره.
قوانین جامعیت موجودیتی (**Entity Integrity Rules**): هر جدول حتماً باید یک کلید اصلی داشته باشد و این کلید نمی تواند مقادیر تکراری داشته باشد و یا **Null** باشد. حتی در صورت چند صفتی بودن کلید، هیچ جزئی از آن نمی تواند **Null** باشد .

قانون جامعیت ارجاعی(**Referential Integrity Rule**): مقادیری که در یک جدول برای کلید خارجی وارد می شوند باید قبلاً برای کلید اصلی مربوطه در جدول دیگر وارد شده باشند.

☑ حذف آبشاری (**Cascade Deleting**): یعنی با حذف سطر مربوط به مقداری از کلید اصلی از یک جدول تمامی مقادیر مرتبط به آن در جداول دیگر که به عنوان کلید خارجی تعریف شده به طور اتوماتیک پاک می شود. مثلاً باحذف سطر اول از جدول کالا سطر اول و سوم از جدول سفارش هم حذف خواهد شد.

به روز رسانی آبشاری(**Cascade updating**): یعنی با تغییر سطر مربوط به مقداری از کلید اصلی از یک جدول تمامی مقادیر مرتبط به آن در جداول دیگر که به عنوان کلید خارجی تعریف شده به طور اتوماتیک تغییر می کند. مثلاً با تغییر **P1** در جدول کالا تمام **P1** ها در جدول سفارش هم تغییر خواهند کرد.

جدول وام Borrow

وام	کد مشتری	کد وام
Amount	**C#**	**L#**
۵۰/۰۰۰	۱۱	۱
۳۰۰/۰۰۰	۱۲	۲
۱۰۰/۰۰۰	۱۲	۳

جدول مشتری Customer

City	**Cname**	**C#**
Tehran	**Ali**	۱۱
Karaj	**Reza**	۱۲
Tehran	**Hamid**	۱۳

جدول حساب Deposit

کد شعبه	موجودی	کد مشتری	شماره حساب
B#	**Balance**	**C#**	**A#**
A	۱۰۰/۰۰۰	۱۱	۱۱۱
B	۲۰۰/۰۰۰	۱۲	۱۱۲
A	۳۰۰/۰۰۰	۱۳	۱۱۳

جدول شعبه Branch

شهر شعبه	نام شعبه	کد شعبه
Bcity	**Bname**	**B#**
Karaj	**Vanak**	**A**
Tehran	**Tajrish**	**B**
Tehran	**7tir**	**A**

جداول نمونه فوق را بعنوان برخی از جداول پایگاه داده بسیستم بانک معرفی می کنیم و در ادامه برای اجرای و بدست آوردن خروجی، این جداول را در نظر می گیریم.

جبر رابطه ایی (Relation Algebra): جبر رابطه ایی برای بیان مفاهیمی است که مربوط به نوشتن درخواستها روی بانک اطلاعاتی بصورت جبری و تئوری می باشد و بیشتر جنبه تئوری داشته و قابل اجرا نیست. این مفاهیم در **SQL** قابل اجرا هستند. جبر رابطه ایی از یکسری عملگرهایی تشکیل شده که روی جداول انجام می گیرند و حاصل خروجی آنها باز هم یک جدول خواهد بود. بنابراین می توان حاصل یک عملگر را روی جداول به عنوان جدول جدیدی در نظر گرفت و آن را برای یک عملگر دیگر به کار برد.

عملگر **Celect** یا گزینش : δ (نام)

شرط ها

برای جدا کردن سطرهایی می باشد که شرط مورد نظر را دارند .

مثال : درخواستی بنویسید که اطلاعات مربوط به حساب افرادی را نشان دهد که در شعبه ایی بنام **A** حساب دارند و موجودی آنها از ۲۰۰/۰۰۰ بیشتر است .

δ(Deposit)
B#="A" And balance>200/000

خروجی

B#	Balance	c#	A#
A	**300/000**	**13**	**13**

عملگر **Project**(پرتو): برای جدا کردن ستون ها می باشد. π (نام جدول)

نام ستون

مثال: درخواستی بنویسید که از جدول حساب کد افراد را نشان دهد.

π (**Deposit**) خروجی

C#

C#
۱۱
۱۲
۱۳

تمرین: درخواستی بنویسید که کد افراد ی را نشان دهدکه در شعبه **A** حساب دارند و موجودی آنها بیش از ۲۰۰/۰۰۰ است.

عملگر اجماع(**Union**): (رابطه طرف دوم) **U** (رابطه طرف اول)

مقادیری را نشان می دهد که حداقل در یکی از طرفین باشند.

مثال: درخواستی بنویسید که افرادی را نشان دهد که حساب دارند و وام گرفته اند.

π(Deposit) U π(Borrow)

c#	c#		c#
11	11	خروجی	11
12	12	⟶	12
13	12		12

عملگر اشتراک (**Intersect**): رابطه طرف دوم ∩ رابطه طرف اول

مقادیری را نشان می دهد که در هر دو طرف هستند.

مثال: درخواستی بنویسید که افرادی را نشان دهد که هم حساب دارند و هم وام گرفته اند.

π(Deposit) ∩ π(Borrow)

c#	c#		c#
11	11	⟶	11
12	12	خروجی	12
13	12		

عملگر تفریق (**Minus**) : طرف دوم – طرف اول

مقادیری را نشان می دهد که در طرف اول هستند ولی در طرف دوم نیستند.

مثال: درخواستی بنویسید که افرادی را نشان دهد که حساب دارند ولی وام نگرفته اند.

π(Deposit) - π(Borrow)

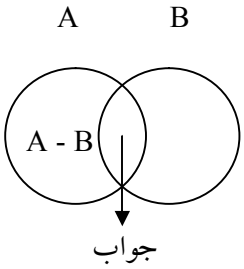
c#	c#		c#
11	11	خروجی	11
12	12	⟶	
13	13		

نکته : شرط لازم برا ی استفاده از سه عملگر اجتماع ، اشتراک و تفریق، سازگار بودن دو رابطه دو طرف این عملگرهاست. شرط سازگاری این است که تعداد ستونهای دو طرف برابر و نظیر به نظیر هم معنی باشند.

π(Deposit) U π(Borrow)
C#,A#
C#,L#

هم معنی نیست پس ازگاری ندارد

مثال : نتیجه عبارت **A – (A - B)** چیست ؟



۱) **A – B**
۲) **A U B**
۳) **A ∩ B**
۴) **A**

عملگر تقسیم (Divide): شرط تقسیم جدول **A** بر جدول **B** این است که ستونهای **B** زیر مجموعه ستونهای **A** باشند و نتیجه این تقسیم ستونهای غیر مشترک از جدول **A** خواهد بود که مقادیر این ستون در صورتی نشان داده
می شود که با تمامی مقادیر **B** ترکیب شده باشند.
مثال:

جدول **B**

جدول**A**

X	K	Z
X1	K1	Z1
X1	K2	Z2
X2	K1	Z1

÷

K	Z
K1	Z1
K2	Z2

=

X1

در تقسیم مشترک ها از بین می رود و سپس آنهایی را می نویسیم که با کل داده های جدول دوم ترکیب شده اند.
مثال: نتیجه تقسیم **S** و **P** چیست؟

S

S#	Sname
A1	Sn1
B1	Sn2
A1	Sn3
B2	Sn1

÷ **p**

S#
A1

= **S/P**

عملگر ضرب دکارتی (Times) : این عملگر دو یا چند جدول را با یکدیگر ترکیب می کند. به این صورت که ستون های دو جدول را کنار هم قرار می دهد و حتی در صورت وجود ستون مشترک آنها را بصورت تکرای نمایش می دهد و تمامی سطرهای جداول اول با تمامی سطرهای جداول دوم ترکیب می شوند.

جدول R1

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

جدول R2

B	D
B1	D1
B2	D2

حاصلضرب دو جدول R1 **و** R2

A	B	C	B	D
A1	B1	C1	B1	D1
A1	B1	C1	B2	D2
A2	B2	C2	B1	D1
A2	B2	C2	B2	D2
A3	B3	C3	B1	D1
A3	B3	C3	B2	D2

معمولاً از ضرب زمانی استفاده می شود که بخواهیم شرط خاصی را که همزمان به بیش از یک جدول مربوط می شود را چک کنیم.

متال: درخواستی بنویسید که شماره حساب افرادی را نشان دهد که مقدار وام آنها از موجودی آنها بیشتر است.(**Amount>Balance**)

$$\pi_{\text{Deposit.A\#}}\left(\delta_{\text{Deposit.Balance<Borrow.Amount and Deposit.c\# = borrow.c\#}}(\text{Deposit} < \text{Borrow})\right)$$

B#	Balance	C#	A#	Amount	C#	L#
A " "	100/00 " "	11 11 11	111 111 111	50/000 300/000 100/000	11 12 12	1 2 3
B " "	200/000 " "	12 12 12	112 112جواب 112	50/000 300/000 100/000	11 12 12	1 2 3
A " "	300/000 " "	13 13 13	113 113 113	50/000 300/000 100/000	11 12 12	1 2 3

متال:

درخواستی بنویسید که موجودی و شماره حساب افرادی را نشان دهد که در شهر تهران زندگی می کنند و در شعبه ایی با کد **A** حساب دارند.

$$\pi\Big(\quad \delta_{\text{Customer .city = "Tehran" And Deposit .B# ="A" And Customer .C # = Deposit# C}}(\text{Costumer* Deposit})\quad\Big)$$

Deposit .A#, Deposit . Balance

خروجی

⟶

A #	Balance
11	100/000
113	300/000

بهینه سازی درخواست : در مثال بالا فرض کنید هر دو جدول دارای سه هزار سطر می باشند حاصلضرب آنها نه میلیون سطر خواهد داشت که در این نه میلیون سطر باید سه شرط فوق چک شود و این خیلی زمان بر است. برای بهینه کردن این درخواست می توان تاحد امکان ابتدا شرط ها را روی جدول مربوطه اعمال نمود و بعد ضرب را انجام داد.

$$\pi\Big(\delta\ \Big(\ \delta_{\text{city = "Tehran"}}(\text{Costumer})\ *\ \delta_{\text{B# = "A"}}(\text{Deposit})\ \Big)\ \Big)$$

Customer .C # = Deposit# C

Deposit .A#, Deposit . Balance

عملگر پیوند طبیعی(Natural) ∞ : شبیه ضرب دکارتی عمل می کند با این تفاوت که اولاً شرط برابری ستون های مشترک را با خود دارد و دوماً ستونهای تکراری را یکبار نشان می دهد.

مثال : حل مثال قبل با پیوند طبیعی

$$\pi\Big(\delta_{\text{city = "Tehran}}(\text{Costumer})\ \infty\ \delta_{\text{B# = "A"}}(\text{Deposit})\ \Big)$$

Deposit .A#, Deposit . Balance

سوال: رابطه **M** دارای سه سطر و پنج ستون و رابطه **N** دارای دو سطر و سه ستون می باشد. رابطه حاصل از ضرب دکارتی آنها چند سطر و چند ستون دارد؟

مثال: رابطه ایی که نتیجه پیوند طبیعی دو رابطه **P, S** می باشد چند سطر و چند ستون دارد؟

۱) سه سطر و چهار ستون ۲) چهارسطرو سه ستون ۳) سه سطر و سه ستون ۴) چهارسطروچهارستون

رابطه **P**

P #	Type	S #
12	K	۰۰۲
15	H	۰۰۳
17	K	002

رابطه **S**

S #	S Name
۰۰۲	S N1
۰۰۲	S N2
۰۰۳	S N3

عملگر ← انتساب (جایگزینی) :

عملگر انتساب حاصل یک عبارت را در متغییر جدیدی از نوع رابطه (جدول) می ریزد تا در عبارتهای بعدی برای سادگی نوشتن از رابطه جدید استفاده شود. مثلا: **R3←R1*R2**

مثال: در رابطه زیر از چه عملگرهایی استفاده شده است ؟

$$MP1 \leftarrow \pi_{S\#}\left(\delta_{Color="Red"}(S \infty P)\right)$$

جواب: از عملگرهای پیوند طبیعی؛ گزینش (انتخاب)؛ پرتو(**Project**) ؛ انتساب (جایگزین)

عملگر نیم پیوند (**Semi join**) $\propto$: این عملگر شبیه پیوند طبیعی عمل می کند با این تفاوت که یک دستور π (**project**) هم برای جدا کردن کلیه ستونهای طرف اول با خود دارد.

نکته : عملگر فوق خاصیت جابجایی ندارد.

مثال: کدام یک از عملگرهای زیر خاصیت تعویض پذیری (جابجایی) ندارد؟

۱) $\div ,\cup ,-$ ٭ ۲) $\propto ,-,\div ,\cap$ ۳) $\cup ,\cap ,x,\infty$ ۴) $\propto ,\div ,-$

تمرین: در جدول زیر مشخص کنید کدامیک از عملگر ها خاصیت جابجایی دارند.

عملگر	جابجایی
δ	✓
π	✘
$\cup$	....
$\cap$	....
$-$	....
$\times$	....
∞	....
$\propto$	....
$\div$	....

زبان SQL (Structured Query Language) :

برای نوشتن درخواستهای قابل اجرا روی بانک اطلاعاتی (**Query**) از این زبان استفاده می شود. **SQL** بُعد عملی جبر رابطه ایست و شامل دستوراتی است که با ترکیب آنها می توان درخواست مورد نظر را نوشت.
SQLیک زبان استاندارد است و تمامی ابزارهای ایجاد پایگاه داده ها از آن پشتیبانی می کنند.(مثل **Access** ، **SQL Server**، **Oracle** و **My SQL**) . همچنین می توان برای نوشتن یک برنامه کاربردی در بین دستورات زبان مورد نظر از دستورات **SQL Server** استفاده نمود. زمانی این کار لازم است که بخواهیم برنامه ایی بنویسیم که با پایگاه داده هم تعامل دارد.

مثال:

---- دستورات زبان میزبان (**Host Language**)
---- **VB**

---- دستورات زبان **SQl** به عنوان زیر زبان داده ها (**Data SubLanguage**)
---- **SQL**

---- دستورات زبان میزبان (**Host language**)
---- **VB**

دستور **Select** : این دستور داده های درون جداول مورد نظر را بازیابی می کند.

Select نام ستون ها

From نام جداول

where شرط ها

مثال: درخواستی بنویسید که نام و کد مشتری های تهرانی را نشان دهد.

```
Select C#,CName
From custumer
where city='Tehran'
-----------------------------
```

خروجی

C#	Name
۱۱	Ali
۱۳	Hamid

نکته: در قسمت **Select** می توان روی ستونها محاسبات را نیز انجام داد. مثلاً موجودی افراد را بـا مبلـغ ۵/۰۰۰ اضافه تر نمایش دهد و یا میزان مالیاتی را که هر شخص باید بپردازد را به اندازه یک دهم موجودی آن نشان داد. در این مواقع بهتر است به این ستون محاسباتی ، یک نام مستعار(**Alias**)نسبت دهیم . در غیر اینصورت نام ایـن ستون را در خروجی **expr** قرار می دهد که به معنی عبارت (**Expression**). برای در نظر گرفتن نام مستعار از کلمه **as** استفاده می کنیم.

مثال: درخواستی بنویسید که کد و مالیات هر شخص را نشان دهد. (مالیات را به اندازه یک دهم موجودی در نظر بگیرید)

```
Select c# , balance * 0.1 as TAX
From Deposit
```

نکته : در صورتی که بخواهیم تمام ستون های یک جدول را در خروجی نمایش دهیم بجای ذکر نام آنها از علامت * استفاده می کنیم.

مثال: درخواستی بنویسید که شماره حساب ، موجودی ، کد شعبه و کد افرادی را نشان دهد که موجودی آنها از ۱۵۰۰۰۰ بیشتر است.

```
Select *
From Deposit
Where balance>150000
```

خروجی

A#	Balance	B#	C#
۱۱۲	۲۰۰/۰۰۰	B	۱۲
۱۱۳	۲۰۰/۰۰۰	A	۱۳

نکته : در صورتی که بخواهیم مقادیر تکرار ی یک ستون فقط یکبار نمایش یابد از کلمه کلیدی **Distinct** در قسمت **Select** و قبل از نام ستون اول استفاده می کنیم.

مثال : درخواستی بنویسید که کد افرادی را نشان دهد که وام گرفته اند. (کد افرادی که وام گرفته اند یکبار بیشتر نشان ندهد)

```
Select Distinct c#
From Borrow
```

خروجی

C#
۱۱
۱۲

نکته: برای ساختن شرطهای پیشرفته تر از عملگر های زیر می توان در قسمت **where** استفاده نمود:

Between	مابین بودن
Not Between	مابین نبودن
Is Null	پوچ بودن
Is Not Null	پوچ نبودن
Like	شبیه بودن
Not Like	شبیه نبودن

مثال : برنامه بنویسید که کد افرادی را نشان دهد که موجودی آنها بین ۱۲/۰۰۰ تا ۲۲۰/۰۰۰ باشد.

```
Select c#
 From Deposit
  Where balance between 120/000 and 220/000
```

--

خروجی

C#

۱۲

- عملگر **like** شبیه بودن مقادیر رشته ایی را با یک الگو بررسی می کند. در این عملگر علامت٪ به معنی هر تعداد کاراکتر و علامت _ (**Under line**) به معنی فقط یک کاراکتر می باشد.

مثال : درخواستی بنویسید که کد و نام افرادی را نشان دهد که نام آنها با **A** شروع می شود.

```
Select c#
 From Custumer
Where CName Like 'A%'
```

- اگر بخواهیم سه حرفی باشد و با با حرف **A** شروع شود عبارت **"A--"** را باید بنویسیم.

مثال: درخواستی بنویسید که اسم کسانی را نشان دهد که حرف سوم آنها **B** باشد.

```
Select Cname
  From Custumer
Where CName Like '_ _B%'
```

- **استفاده از توابع تجمعی در قسمت** Select **: این توابع عبارت اند از:**

Count	تعداد مقادیر (یکی از کاربرها برای شمارش سطرها)
Avg	میانگین مقادیر
Sum	مجموع مقادیر
Min	کمترین مقادیر
Max	بیشترین مقادیر

مثال : درخواستی بنویسید که مجموع و میانگین موجودی ها را نشان دهد.

Select Sum(balance) as "مجموع**" , Age (balance) as "** میانگین **"**
From Deposit

خروجی

میانگین	مجموع
۲۰۰/۰۰۰	۶۰۰/۰۰۰

جدول زیر را در نظر بگیرید ؛ اگر بخواهیم بیشترین موجودی یا میانگین موجودی را به ازای هر شعبه جداگانه حساب کنیم باید نتایج را بر اساس **B#** گروه بندی کنیم.(پس گاهی اوقات لازم است بعد از استفاده از توابع تجمعی عمل گروه بندی هم انجام شود) این بار دستور *نام ستون* ***group by*** انجام می شود.

مثال: جدول زیر را در نظر بگیرید:

B#	Balance	C#	A#
A	۱۰۰/۰۰۰	۱۱	۱۱۱
B	۲۰۰/۰۰۰	۱۲	۱۱۲
A	۳۰۰/۰۰۰	۱۳	۱۱۳
B	۴۰۰/۰۰۰	۱۴	۱۱۴

سوال: درخواستی بنویسید که مجموع موجودی هر شعبه را به همراه کد آن شعبه نشان دهد.

Select B #, Sum (balance) as مجموع
From Deposit
Group by B#

خروجی

B#	مجموع
A	۴۰۰/۰۰۰
B	۶۰۰/۰۰۰

نکته : در قسمت **Where** از هر شرطی می توان استفاده نمود بجز شرط هایی که با توابع تجمعی ساخته می شوند. اینگونه شرط ها را باید بعد از **Group by** در قسمت **Having** نوشت.

مثال: درخواستی بنویسید که بیشترین موجودی را به ازای هر شعبه و در کنار آن شعبه به شرطی نمایش دهد که از ۳۵۰/۰۰۰ کمتر نباشد . (طبق جدول مثال قبل)

تذکر: در صورتی که بخواهیم خروجی **Select** بر اساس ستون خاصی بصورت صعودی (**Ascending**) و یا نزولی (**Descending**) مرتب شده باشد، از دستور زیر به عنوان آخرین دستور **Select** استفاده می کنیم.

Order by نام ستون نوع مرتب سازی

(نوع مرتب سازی می تواند **Ascending** یا **Descending** باشد که به ترتیب با کلمه **ASC** یا **DESC** مشخص می کنیم

نکته: ترتیب اجزای دستور **Select** بصورت زیر است.

Select……
From ……
Where ……..
Group by……..
Having……..
Order by……..

پیوند جداول:

همانطور که در بحث جبر رابطه ایی مطرح شد در برخی از درخواستها نیاز به پیوند جداول می باشد که یکی از انواع آنها ضرب دکارتی است در**SQL** برای برای انجام عمل ضرب کافیست اسم جداول را در دستور **Select** و در قسمت **From** وارد کنیم و شرط ضرب را علاوه بر شرهای مسأله در قسمت **Where** وارد کنیم .

مثال: درخواستی بنویسید که شماره حساب و کد مشتریهایی را نشان دهد که مقدار وام آنها از موجودی آنها بیشتر باشد.

Select Deposit. A # , Deposit. C#
From Deposit , Borrow
Where Deposit . balance < Borrow . AmountAnd deposit . C# = Borrow. C#

پیوند درونی (Inner join) : شبیه ضرب دکارتی عمل می کند با این تفاوت که در قسمت **From** بجای کاما باید عبارت "**Inner Join** " را بنویسیم . همچنین شرط ضرب را به همراه کلمه **ON** در قسمت **From** وارد کنیم تا فقط سطرهایی را پیوند دهد که این ظرط را دارند . پس در واقع سریعتر از ضرب عمل می کنند.

Select اسم ستون ها
From جدول۱ **Inner join** جدول ۲ **ON** شرط پیوند
Where..... شرط ها

حل مثال قبل با پیوند درونی:

Select Deposit . A # , deposit . c #
From Deposit inner join borrow on deposit . c # = borrow . c #
Where deposit . balance < borrow . amount

مثال : درخواستی بنویسید که موجودی افرادی که در شعبه ونک حساب دارند را نشان دهد.

Select Deposit . balance
From Deposit inner join branch ON deposit . B # = branch . B #
Where branch . bname ='Vanak'

دستور Insert : برای وارد کردن داده ها (سطرها) به جداول می باشد.

insert into نام جداول
(نام ستون ها)
Values
(مقادیر)

مثال : درخواستی بنویسید که یک مشتری جدید بنام مهدی و کد ۱۷ و شهر تهران به جدول مشتری اضافه کند.

```
Insert into  customer
(c # , Cname , City)
Values
(17 , ' Mehdi",  'Tehran')
```

نکته : هر دستور **Insert** با هر بار اجرا فقط می تواند اطلاعات را روی یک جدول وارد کند.

دستور **update** : این دستور باعث انجام تغییرات و به روز رسانی داده های جدول می گردد.

Update نام جدول

Set تغییرات

Where شرط ها

مثال: درخواستی بنویسید که به موجودی افرادی که در شعبه ایی با کد **A** حساب دارند مبلغ **5/000** اضافه کند.

```
Update deposit
Set balance = balance +
                                    5000
Where b # = 'A'
```

مثال: درخواستی بنویسید که به موجودی افرادی که در شعبه ایی بنام **Vanak** حساب دارند ۵۰۰۰ اضافه کند.
نکته : شرط مورد نظر به یک جدول دیگر (**branch**) مربوط می شود . در این مواقع از **Select** تو در تو در قسمت استفاده می کنیم (فیلد مشترک دو جدول **B#** می باشد)

```
Update deposit
 Set balance = balance + 5000
 Where b# in  ( select b#
                  From branch
                   Where bname = 'vanak')
```

دستور حذف برخی از سطرها : دستور **Delete** سطرهایی را حذف می کند که شرایط خاصی دارند . الگوی آن بصورت زیر است.

Delete From نام جدول

Where شرط ها

مثال: درخواستی بنویسید که از جدول مشتری افراد تهرانی را حذف کند.

```
Delete From Customer

Where City='Tehran'
```

نکته: این دستور می تواند یک جدول را **Update** کند.

مثال: دستوری بنویسید که اطلاعات مربوط به شعباتی که افراد ی با موجودی بیش از ۲۰۰/۰۰۰ در آنجا حساب دارند را حذف کند(از جدول **Branch**)

```
Delete From Branch
Where B# In (select B # from Deposit where Balance>200/000 )
```

مثال: درخواستی بنویسید که اطلاعات مربوط به حساب افرادی که وام گرفته اند را حذف کند.

دستور ایجاد جدول :

Create table نام جدول

نام ستون نوع ستون (اندازه) **Primary key,**

نام ستون نوع ستون (اندازه) **)**

مثال : جدولی با چهار فیلد بصورت زیر ایجاد کنید.

Code	Name	Age	Score

Create table student
Code int (10) Primary Key,
Name char (10),
Age float,
Score float)

مثال : دستوری بنویسید که سطرهایی را از جدول مشتری حذف کند که مربوط به افرادی است که در شعبه ایی با کد **A** حساب دارند.

Delete from customer
Where c# in (select c# from deposit
Where b# ='a')

دستور تغیرات در ساختارجدول:

Alter table نام جدول

نوع نام ستون **Add**

نام ستون **Drop column**

نوع جدید نام ستون **Alter column**

مثال: دستوری بنویسید که به جدول **Branch** ستونی بنام **Street** اضافه کند.

Alter Table Branch
Add street nvarchar (20)

مثال: دستوری بنویسید که ستون **C#** را در جدول مشتری از **int** به **Float** تغییر دهد.

Alter table customer
Alter column c# float

دستور حذف کلید سطرهای جدول:

Truncate table نام جدول

مثال: دستوری بنویسید که همه سطرهای جدول وام را حذف کند.

Truncate table Borrow

دستور حذف جدول**:** این دستور کل ساختار جدول را به همرا تمام داده های آن حذف می کند.

Drop table نام جدول

مثال: دستوری بنویسید که جدول **Student** را حذف کند.

Drop Table Student

وابستگی تابعی **Functional Dependency (F.D)**

صفت یا مجموعه صفات **y** به صفت یا مجموعه صفات **x** وابستگی تابعی دارند اگر و فقط اگر به ازای هر مقـدار **x** حداکثر یک مقدار برای **y** وجود داشته باشد. وابستگی تابعی را با نمار روبرو نشان می دهیم: $X \xrightarrow{F.D} Y$

وابستگی تابعی کامل Full Functional Dependency (F.F.D):

صفت یا مجموعه صفات **y** به صفت یا مجموعه صفات **x** وابستگی تابعی کامل دارند اگر و فقط اگر وابستگی تابعی داشته باشد و در صورت ترکیبی بودن **X** به کل آن وابسته باشد ولی به اجزائ آن وابسته نباشد.

مثال: رشته ⟵ کد دانشجو، نام دانشجو

با وجود اینکه **F.D** دارد ولی **F.F.D** ندارد چون رشته به کد وابسته است و این یعنی وابستگی جزئی.

مثال: نمره $\xrightarrow{FFD}$ (کد دانشجو، کد درس)

نمره به تنهایی به کد دانشجو وابسته نیست . همچنین به تنهایی به کد درس وابسته نیست . پس وابستگی جزئی وجود ندارد و وابستگی کامل است .

نرمال سازی رابطه ها(Normalization):

به فرآیندی گفته می شود که روی پایگاه داده انجام می شود و منجر می شود پایگاه داده به جداولی تبدیل شود که در آنها تا حد امکان از افزونگی (**Redundency**) یعنی تکرار بی رویه داده ها و بی نظمی (**Anomaly**)جلوگیری شده است.

اساس کار نرمال سازی تجزیه جداول است یعنی با شکستن یک جدول به چند جدول باعث نرمال تر شدن جداول

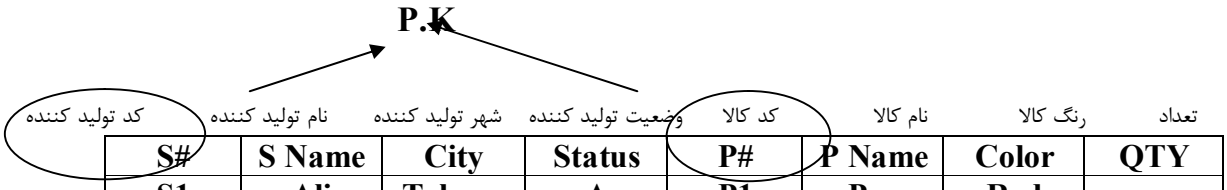
می گردد. سطوح مختلفی برای نرمال ساز یوجود دارد که عبارتند از:

افزایش سطح نرمال سازی ↓

1NF (First Normal Form)
2NF (Second Normal Form)
3NF (Third Normal Form)
BCNF (Boyce- Codd Normal Form)
4NF (Fourth Normal Form)
5Nf (Fifth Normal Form)

هرچه عمل نرمال سازی بیشتر انجام شود افزونگی کمتر می شود ولی چون تعداد جداول بیشتر خواهد شد، باعث افزایش پیچیدگی می شود. بنابراین بهتر است عمل نرمال سازی تا سطحی انجام شود که افزونگی و پیچیدگی هر دو در سطح قابل قبول باشند و معمولاً این سطح **3NF** است.

شرایط 1NF: هر جدولی که کلید اصلی آن مشخص شده باشد و ستون چند مقداری هم نداشته باشد حداقل در سطح **1NF** است مانند جدول زیر:



S#	S Name	City	Status	P#	P Name	Color	QTY
S1	Ali	Tehran	A	P1	Pen	Red	۲۰
S1	Ali	Tehran	A	P2	Bag	Black	۳۰
S2	Reza	Karaj	B	P1	Pen	Red	۴۰

جدول فوق با وجود اینکه **1NF** است مشکلاتی دارد از جمله افزونگی و بی نظمی که در ادامه توضیح داده می شوند.

بی نظمی در درج اطلاعات: بطور مثال نمی توان فقط اطلاعات زیر را در مورد یک تولید کننده در جدول وارد نمود. (**S3,Hamid,Tabriz,C**) زیرا به همراه آن اطلاعات مربوط به تولید کننده **S2** هم حذف می شود.

- در فرآیند نرمال سازی وابستگی های تابعی یک جدول را در یک گراف به نام گراف وابستگی ها نمایش می دهند.

برخی از وابستگی های این گراف بدیهی است . بطور مثال درمورد جدول فوق داریم :

S# ⟶ S Name

یا

S# ⟶ City

P# ⟶ P Name

ولی برخی از وابستگی های دیگر را باید در صورت مسأله مشخص کرد. به عنوان مثال جمله زیر:

هر شهری یک وضعیت دارد؛ یعنی **City ⟶ Status**

همچنین جمله زیر:

هرتولید کننده ایی هر کالایی را یکبار تولید کرد؛ یعنی به ازای هر **S#** و**P#** یک تعداد**QTY** وجود دارد.

مثال: **(S#,P#) ⟶ QTY**

شرایط 2NF: هر جدولی که اولاً در سطح **1NF** باشد و دوماً وابستگی جزئی آن در مشاهده نشود در سطح **2NF** است ؛ یعنی هیچ صفت غیر کلیدی به جزئی از کلید وابسته بلکه وابستگی تابعی کامل باشد.

با تعریف فوق و با توجه به گراف وابستگی های فوق جدول قبل **2NF** نیست.

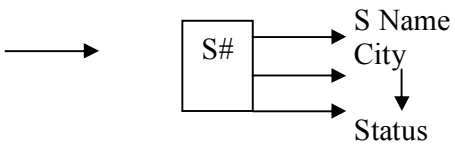
مراحل تبدیل 1NF **به** 2NF :

۱) هر جزئی از کلید را به همراه وابسته های آن در یک جدول جداگانه قرار می دهیم.

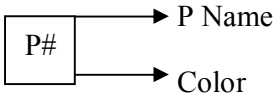
۲) صفات باقیمانده را با کل کلید در جدول دیگر قرار می دهیم .

تبدیل به **2NF**

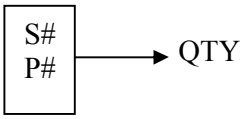
جدول **R1**



جدول **R2**



جدول **R3**



شرایط 3NF :

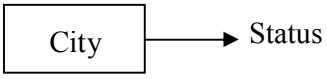
جدولی در سطح **3NF** است که اولاً **2NF** باشد و دوماً وابستگی گذری (**Transitive Dependency**) در آن وجود نداشته باشد. وابستگی گذری یعنی دو صفت غیر کلید به یکدیگر وابسته باشند. با این تعریف جدول **R1** نمی تواند**3NF** باشد، چون صفت غیر کلید **Status** به صفت غیر کلید**City** وابسته شده است. ولی جداول **R3**و**R2** هر دو **3NF** هستند.

مراحل تبدیل 2NF **به** 3NF:

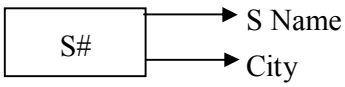
۱) هرصفتی که باعث وابستگی گذری شده است را به همراه وابسته های آن در یک جدول قرار می دهیم.

۲) کلید اصلی را با صفات باقیمانده و همچنین صفاتی که باعث وابستگی گذری شده انـد در جـدولی دیگـر قرار می دهیم.

جدول **R1-1**



جدول **R1-2**

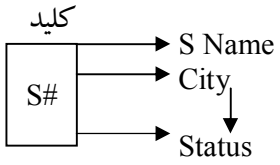


شرایط BCNF : جدولی در سطح **BCNF** است که در آن هر دترمینال کلید کاندید باشد.
تعریف دترمینال: اگر **y** به **x** وابسته باشد به **x** دترمینال گفته می شود.

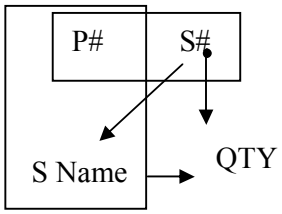
دترمینال

$$\mathbf{X} \longrightarrow \mathbf{y}$$

مثال: آیا جدولی که گراف وابستگی آن بصورت زیر است **BCNF** است؟



مثال: جدول روبرو در چه سطح نرمالی قرار دارد.



آیا **2NF** است؟ بله (چون صفت غیر کلیدی نداریم که به جزئی از کلید وابسته شده باشد)
آیا **3NF** است؟ بله(چون وابستگی گذری در آن وجود ندارد)
آیا **BCNF** است؟ خیر(چون **S#** دترمینال است ولی به تنهایی کلید نیست)