*In the name of Allah*

*2007*

# *Basic Information In Database*


Mohammad

www.mamizade.ir

4/25/2007

# بانک های اطلاعاتی رابطه ای ( مفاهیم و تعاریف)

- موجودیت (Entity)

به هر چیزی (شی ، شخص ، محل و ...) که می خواهیم در یک سیستم راجع به آن اطلاعاتی را جمع آوری ، پردازش و نگهداری نمائیم ، یک موجودیت گفته می شود . تعریف فوق ، متداولترین برداشت اولیه از موجودیت می باشد . مجموعه موجودیت های یک سیستم ، ساختار اطلاعاتی آن سیستم را مشخص می کند . هر موجودیت شامل اجزاء و المان هائی است که آن موجودیت را توصیف می کند که به آنها خصیصه و یا **Attribute** گفته می شود . هر موجودیت بسته به این که در سیستم مورد مطالعه چه میزان اطلاعات راجع به آن می خواهیم داشته باشیم ، شامل حداقل یک و یا چند خصیصه خواهد بود. از آنجا که هر موجودیت راجع به یک موضوع به خصوص می باشد ، بنابراین یک ارتباط منطقی بین کلیه خصایص موجودیت وجود خواهد داشت .در واقع ، تمام خصائص یک موجودیت توصیف کننده آن موجودیت خواهد بود . برای روشن شدن موضوع بد نیست به نمونه مثال ذیل توجه نمائید :

– موجودیت مشتری شامل خصلت های نام مشتری ، آدرس مشتری ، تلفن مشتری و ... است .

– موجودیت سفارش شامل خصلت های شماره سفارش ، تاریخ سفارش ، نام مشتری ، کالای سفارش شده ، تعداد کالای سفارش شده و ... است

همانگونه که در مثال فوق مشاهده گردید ، تمام خصلت های موجودیت مشتری توصیف کننده یک مشتری و تمام خصلت های موجودیت سفارش توصیف کننده یک سفارش می باشند .

- کلید (Key)

هر رخداد از یک موجودیت را باید بتوان به وسیله یک و یا ترکیبی از چند خصیصه آن به صورت یکتا شناسائی نمود . به تعبیر دیگر ، هر یک از رخدادهای یک موجودیت باید یکتا باشد ، در غیر اینصورت تغییر و یا حذف یک رخداد از موجودیت (در مثال فوق یک مشتری) غیر ممکن خواهد بود . از اینرو از بین خصلت های یک موجودیت یک و یا ترکیبی از چند خصیصه به عنوان کلید آن موجودیت انتخاب می شود . این خصلت (و یا ترکیب خصلت ها) باید بتواند یکتائی هر رخداد از موجودیت را تضمین نماید . در موجودیت سفارش مثال فوق ، خصلت شماره سفارش می تواند بعنوان کلید انتخاب شود .

توضیح : در برخی از موارد در یک موجودیت چندین کلید وجود دارد که به هر یک از آنها یک **Candidate Key** یا **Alternate Key** گفته می شود .

در برخی از حالات نمی توان در یک موجودیت هیچ کاندیدی برای کلید یافت ، مانند موجودیت مشتری در مثال فوق . در این موجودیت هیچیک از خصلت ها و یا هیچ ترکیبی از آنها نمی تواند صد درصد تضمین کننده یکتائی آن باشد (با اینکه احتمال وجود دو مشتری هم نام در یک آدرس و با یک شماره تلفن بسیار کم است ، اما باز هم احتمال وقوع دارد) . در چنین مواردی مجبور هستیم یک خصلت به موجودیت اضافه کنیم تا تضمین کننده یکتائی رخدادهای آن باشد . در مثال فوق با اضافه کردن خصلت کد مشتری به موجودیت مشتری ، می توان یکتائی آن را تضمین نمود . به این نکته دقت شود که بسیاری از خصلت های یک موجودیت در کنترل سیستم نیست و از خارج به سیستم تحمیل می گردد . به عنوان مثال ما نمی توانیم تعیین کنیم که نام مشتری های سازمان تکراری نباشد . اما عدم تکراری بودن خصلت هائی که خود ما ایجاد نموده ایم را می توان تضمین کرد ( نظیر کد مشتری که توسط سیستم و یا سازمان مربوطه تولید می شود ) .

- کلید اصلی (Primary Key)

از بین کلیدهای یک موجودیت (Candidate Key) ، می بایست یک کلید را به عنوان کلید اصلی انتخاب نمود . معیارهای مختلفی در این انتخاب دخیل هستند ، اما معمولا" بهترین کلیدی که معرف مفهوم و ماهیت موجودیت باشد به عنوان کلید اصلی انتخاب می گردد .

- وابستگی تابعی (Functional Dependency)

وابستگی تابعی مفهومی است که مابین خصلت های یک موجودیت تعریف می گردد . به این معنی که می گوئیم خصلت **A** با خصلت **B** وابستگی تابعی دارد ، در صورتیکه به ازای هر مقدار مشخص از خصلت **B** بتوان مقدار مشخص و یکتائی از خصلت **A** را بدست آورد ، اما عکس آن ممکن است صادق نباشد . در موجودیت مشتری مثال قبل ، به ازای هر کد مشتری می توان نام او را بدست آورد در این صورت می گوئیم خصلت نام مشتری با خصلت کد مشتری وابستگی تابعی دارد . اما عکس آن صادق نیست چرا که به ازای یک نام مشتری مشخص ، نمی توان یک کد مشتری یکتا استخراج نمود (دو مشتری مختلف می توانند نام یکسان داشته باشند ، در این حالت یک نام مشتری ممکن است متناظر با دو و یا حتی چند کد مشتری باشد).

- انواع رابطه بین خصلت های یک موجودیت

بین خصلت های یک موجودیت سه نوع رابطه وجود دارد :

- رابطه یک به یک (One To One) : در حالتی اتفاق می افتد که خصلت **A** وابستگی تابعی به خصلت **B** داشته باشد و خصلت **B** نیز وابستگی تابعی به خصلت **A** داشته باشد . در این حالت هر دو خصلت **A** و **B** کاندیدای کلید شدن می باشند.

- رابطه یک به چند (Many One To) : اگر خصلت A وابستگی تابعی به خصلت B داشته باشد و عکس آن صادق نباشد ، یک ارتباط از نوع یک به چند وجود خواهد داشت . در این حالت ، خصلت B کاندید کلید شدن است و خصلت A صرفا" یکی از توصیف گرهای موجودیت محسوب می گردد .

- رابطه چند به چند (Many To Many) : اگر دو خصلت هیچکدام وابستگی تابعی به یکدیگر نداشته باشند آنگاه رابطه بین آنها چند به چند خواهد بود . در این حالت هیچیکدام از آنها کاندید کلید شدن نبوده (ممکن است ترکیب آنها کاندید کلید شدن باشد) و صرفا" توصیف کننده موجودیت خواهند بود .

- **هنجار سازی (Normalization)**

هنجار سازی ، فرآیندی است که طی آن یک موجودیت جهت به حداقل رسانی نابهنجاری های بوجود آمده در خلال تغییرات اعمال شده بر روی رخدادهای یک موجودیت مورد بررسی و تبدیل قرار می گیرد. اگر این فرآیند به طور صحیح بر روی یک موجودیت اعمال نگردد ، آنگاه نمی توان هیچ تضمینی در خصوص حفظ یکپارچگی اطلاعات آن موجودیت ارائه داد . فرآیند هنجار سازی به دلیل اهمیت و گستردگی آن در مقاله ای جداگانه تشریح خواهد شد.

- **نا بهنجاری**

به پیامدهای ناخواسته تغییر اطلاعات نابهنجاری گفته می شود .

- **Relation**

موجودیت ها در مدل منطقی داده های سیستم مورد بحث و بررسی قرار می گیرند و پس از طی فرآیند هنجارسازی در مرحله فیزیکی به صورت ماتریسهای دوبعدی مشتمل بر سطرها (رخدادهای مختلف یک موجودیت) و ستون ها (خصلت های مختلف آن موجودیت) تعریف می گردند . هر یک از این ماتریس ها را یک ارتباط یا Relation می نامند که در مدل فیزیکی معمولا" آنها را با نام جدول (Table) معرفی می کنند . همانطور که پیش از این اشاره شد تمام خصلت های یک موجودیت با یکدیگر ارتباط منطقی داشته و معرف آن موجودیت می باشند ، از اینرو به این جداول ارتباط می گویند .

- **Tuple**

هر یک از رخدادهای مختلف یک موجودیت را یک Tuple می گویند که در مدل فیزیکی معمولا" از آنها با نام ردیف (Row) و یا رکورد (Record) نام برده می شود . بنابراین Tuples ، ردیف های جدول دو بعدی هستند که آن را به عنوان Relation و یا Table می شناسیم .

- **Attribute**

هریک از خصلت های مختلف یک موجودیت را **Attribute** می نامند ( نظیر کد مشتری ) .
معمولا" در مدل فیزیکی به جای **Attribute** از فیلد (**Field**) و یا ستون (**Column**) استفاده می شود . بنابراین **Attributes** ، ستون های جدول دو بعدی هستند که آن را به عنوان **Relation** و یا **Table** می شناسیم .

شکل زیر یک **Relation** را به همراه اجزاء آن نشان می دهد :

| Attribute 4 شماره کارمندی | Attribute 3 جنسیت | Attribute 2 سن | Attribute 1 نام | |
|---|---|---|---|---|
| ۱۱۱۱۱ | F | ۱۸ | محمد | Tuple 1 |
| ۲۲۲۲۲ | F | ۱۹ | علی | Tuple 2 |
| ۳۳۳۳۳ | F | ۲۰ | رضا | . |
| ۴۴۴۴۴ | F | ۲۱ | فرهاد | . |
| ۵۵۵۵۵ | F | ۲۲ | بهروز | . |
| ۶۶۶۶۶ | F | ۲۳ | کاوه | Tuple 6 |

یک **Relation** به همراه اجزاء آن

- **ارتباط (Relationship)**

منظور ارتباط بین دو **Relation** و یا جدول است که بر اساس برابری فیلدهای یکسان در هر جدول تعریف و دارای انواع مختلفی است . ( به دلیل اهمیت و گستردگی ، در مقاله ای جداگانه تشریح خواهد شد) . این ارتباط ها در مدل منطقی مابین موجودیت ها (خصوصا" موجودیت های نرمال شده ) تعیین می گردند و به آن **Entity Relation** یا **ER** سیستم می گویند . مدل **ER** سیستم توسط ابزارهای مستند سازی جهت درک بهتر مدل داده ای سیستم ترسیم می گردد که به آنها **ERD** می گویند .

پس از تشریح برخی از مفاهیم اولیه و در عین حال مهم بانک های اطلاعاتی رابطه ای ، به اختصار می توان گفت که یک بانک اطلاعات رابطه ای مجموعه ای از رابطه ها (**Relations**) و یا جداول به همراه تمامی ارتباط هائی (**Relationship**) است که بین آنها وجود دارد . هر بانک اطلاعاتی در خصوص یک سیستم مورد نظر طراحی و ایجاد می گردد ، اما در برخی از سازمان های بزرگ که بین سیستم های مختلف آن ارتباط وجود دارد (نظیر سیستم پرسنلی ، حقوق و دستمزد و مالی و ...) ممکن است بانک های اطلاعاتی با یکدیگر تجمیع و پس از طی فرآیند یکپارچه سازی به صورت یک بانک اطلاعاتی جامع و یکپارچه برای آن سازمان تعریف و ایجاد گردد .

امروزه سیستم های مدیریتی بانک های اطلاعاتی رابطه ای مختلفی وجود دارد که هر یک ویژگی ها و قابلیت هایی خاص خود را دارند . به این سیستم ها و یا نرم افزارها اختصارا" RDBMS گفته می شود . MS ACCESS، MS SQL ، ORACLE ، SYBASE ، نمونه هائی متداول در این زمینه می باشند . تمامی سیستم های مدیریت بانک های اطلاعاتی رابطه ای به منظور ارائه قابلیت های خود و استفاده از آنها از زبان مشترکی که به آن SQL ( برگرفته شده از Structured Query Language ) گفته می شود ، استفاده می نمایند . تمامی نیازها و انتظارات کاربران از بانک های اطلاعاتی نظیر جستجوی اطلاعات ، ایجاد ، تغییر و یا حذف اطلاعات حتی ایجاد بانک اطلاعاتی و یا سایر اجزاء مرتبط با آن توسط زبان فوق تعریف و تحویل RDBMS داده خواهد شد تا پس از بررسی بر روی بانک اعمال گردد.

## نرمال سازی بانک های اطلاعاتی

قبل ار مطالعه این مطلب پیشنهاد می گردد به دلیل ضرورت آشنائی خوانندگان با مفاهیم بانک های اطلاعاتی رابطه ای ، مقاله "بانک های اطلاعاتی رابطه ای : مفاهیم و تعاریف " ، مطالعه گردد .

نرمال سازی ( Normalization ) یا به تعبیری هنجار سازی فرآیندی است در رابطه با بانک های اطلاعاتی که با دو هدف عمده زیر انجام می شود :

- کاهش افزونگی اطلاعات ، به این معنی که اطلاعات فقط در یک مکان (جدول) ذخیره و در تمام بانک با استفاده از روابط منطقی تعریف شده (RelationShip) قابل دسترسی باشد .
- حفظ یکپارچگی اطلاعات ، به این معنی که اعمال تغییرات بر روی اطلاعات ( نظیر ایجاد ، بهنگام سازی و حذف ) در یک مکان انجام و به دنبال آن آثار تغییرات در تمام بانک مشاهده گردد . برای روشن شدن مفهوم یکپارچگی بد نیست به مثال ذیل توجه نمائید :

فرض کنید در یک بانک اطلاعاتی دارای دو موجودیت کتاب و نویسنده باشیم . هر یک از موجودیت های فوق دارای المان های اطلاعاتی (Attribute) مختص به خود می باشند . به عنوان نمونه موجودیت "کتاب" دارای المان اطلاعاتی نام نویسنده و موجودیت "نویسنده " دارای المان های اطلاعاتی متعددی نظیر نام نویسنده ، آدرس نویسنده و ... باشد . در صورتی که در موجودیت "کتاب" یک رخداد (رکورد) ایجاد نمائیم بدون اینکه نام نویسنده آن را در موجودیت "نویسنده" ایجاد کرده باشیم ، دچار یک ناهمگونی اطلاعات خواهیم شد .

با توجه به اهداف فوق می توان گفت که فرآیند نرمال سازی از ناهنجاری های بوجود آمده به دلیل بروز تغییرات در بانک جلوگیری خواهد نمود . با اعمال فرآیند نرمال سازی ، یک بانک اطلاعاتی کارآ و مطمئن را خواهیم داشت .

فرآیند نرمال سازی ، فرم های متفاوتی دارد که انواع متداول آن به شرح ذیل است :

- فرم اول نرمال سازی NF۱
- فرم دوم نرمال سازی NF۲
- فرم سوم نرمال سازی NF۳
- فرم بویس کد نرمال سازی BCNF
- فرم چهارم نرمال سازی NF٤

### فرم اول نرمال NF۱

موجودیت و یا جدولی در فرم اول نرمال است که تمامی المان های اطلاعاتی آن ( منظور **Attribute** است ) یکتا و یا اصطلاحا" **atomic** باشند . برای روشن شدن این موضوع فرض کنید دارای موجودیتی با نام "فاکتور فروش " باشیم .

| فاکتور فروش |
| --- |
| شماره فاکتور(کلید اصلی) |
| تاریخ فاکتور |
| کد مشتری |
| نام مشتری |
| کالای ۱ |
| تعداد کالای ۱ |
| قیمت واحد کالای ۱ |
| . |
| . |
| کالای n |
| تعداد کالای n |
| قیمت واحد کالای n |

با مشاهده موجودیت فوق متوجه این موضوع خواهیم شد که المان های کالا ، تعداد کالا و قیمت واحد کالا بیش از یک مرتبه در موجودیت وجود داشته و اصطلاحا" یک گروه تکرار را تشکیل می دهند . برای اجرای مدل فیزیکی این موجودیت ناچار خواهیم بود در طراحی جدول آرایه ای به طول ثابت ( به عنوان نمونه با ده عضو ) تعریف و در آن به ترتیب کالای ۱ تا ۱۰ را تعریف نمائیم .

مشکل : طراحی فوق ما را با دو مشکل عمده روبرو خواهد ساخت : اول این که کارائی بانک اطلاعاتی پائین خواهد آمد (اگر در آینده تعداد کالاهای فاکتور فروش بیش از ۱۰ کالا باشد ، آنگاه مجبور خواهیم بود طراحی جدول مربوطه و متعاقب آن نرم افزارهائی که از آن استفاده می کنند را تغییر دهیم ) و مشکل دوم این که بسیاری از فاکتورها لزوما" دارای ۱۰ کالا نیستند و بنابراین محتوی بسیاری از فیلدها در جدول فوق خالی (دارای ارزش **Null**) خواهد ماند و حجم زیادی از فضای دیسک هدر خواهد رفت .

راه حل : برای حل این مشکل کافی است تمامی گروه های تکرار و یا آرایه ها را از موجودیت خارج کرده و به موجودیت دیگری منتقل نمائیم . در چنین مواردی ، کلید اصلی موجودیت اول را به عنوان بخشی از کلید اصلی موجودیت جدید قرار داده و با تلفیق یکی دیگر از آیتم های اطلاعاتی موجودیت جدید که تضمین کننده یکتا بودن رکوردهای آن موجودیت ( جدول ) است ، کلید اصلی موجودیت ایجاد می گردد . بدین ترتیب ، یک ارتباط بین موجودیت پدر و فرزند بر اساس کلید اصلی موجودیت پدر برقرار خواهد شد . مجددا" به موجودیت "فاکتور فروش " مثال قبل پس از تبدیل به فرم اول نرمال توجه نمائید :

| ردیف های فاکتور فروش | | فاکتور فروش |
|---|---|---|
| شماره فاکتور(قسمت اول کلید اصلی)<br>کالا (قسمت دوم کلید اصلی)<br>تعداد<br>قیمت واحد | ارتباط بین موجودیت پدر و فرزند بر اساس کلید اصلی موجودیت پدر (فاکتور فروش) | شماره فاکتور(کلید اصلی)<br>تاریخ فاکتور<br>کد مشتری<br>نام مشتری |

به طور خلاصه می توان گفت که هدف از فرم اول نرم سازی حذف گروه های تکرار و آرایه ها از موجودیت یا جدول است . فرآیند فوق ، می بایست بر روی تمامی موجودیت های بانک اطلاعاتی اعمال گردد تا بتوان گفت بانک اطلاعاتی نرمال شده در فرم اول است .

### فرم دوم نرمال NF۲

موجودیتی در فرم دوم نرمال است که اولا" در فرم اول نرمال باشد و ثانیا" تمامی آیتم های (Attribute) غیر کلیدی آن وابستگی تابعی به تمام کلید اصلی موجودیت داشته باشند نه به بخشی از آن .همانگونه که از تعریف فوق استنباط می گردد ، فرم دوم نرمال سازی در خصوص موجودیت هائی بررسی و اعمال می شود که دارای کلید اصلی مرکب هستند ( بیش از یک جزء ) . بنابراین در مثال فوق موجودیت "فاکتور فروش " به خودی خود در فرم دوم نرمال است ولی موجودیت "ردیف های فاکتور فروش " که دارای کلید اصلی مرکب است ، نیاز به بررسی دارد .

مشکل : در صورتی که موجودیت در فرم دوم نرمال نباشد ، آنگاه با تغییر اطلاعات قسمت های غیروابسته به تمام کلید ، این تغییرات در یک رکورد اعمال می شود ولی تاثیری بر روی سایر رکوردها و یا جداول نخواهد داشت . در مثال فوق با تغییر محتوی قیمت واحد در موجودیت "فاکتور فروش " ، قیمت واحد کالا در یک فاکتور فروش اصلاح می گردد اما در سایر فاکتورها اعمال نخواهد شد .

راه حل : برای حل این مشکل کافی است موجودیت جدیدی ایجاد نمائیم و کلید اصلی آن را برابر با آن بخش از کلید اصلی موجودیت مورد بررسی که دارای المان های وابسته به آن است قرار دهیم ، سپس تمام المان های اطلاعاتی وابسته تابعی به این کلید را از موجودیت مورد بررسی خارج کرده و به موجودیت جدید منتقل نمائیم . در این حالت بین موجودیت جدید ایجاد شده و موجودیت نرمال شده ، بر اساس کلید اصلی موجودیت جدید ایجاد شده یک ارتباط پدر فرزندی تعریف خواهد شد . دقت کنید که بر عکس نرمال سازی فرم اول ، در این جا موجودیت موردبررسی فرزند بوده و موجودیت جدید پدر خواهد بود .
به مثال فوق برمی گردیم و فرم دوم نرمال سازی را بر روی آن اعمال می نمائیم . موجودیت "فاکتور فروش" دارای کلید مرکب نیست پس در فرم دوم نرمال بوده و نیاز به بررسی ندارد ، اما موجودیت "ردیف های فاکتور فروش" نیاز به بررسی دارد . در این موجودیت آیتم اطلاعاتی "قیمت واحد" وابستگی تابعی به آیتم کالا دارد که بخشی از کلید است نه کل کلید ، پس لازم است تا این موجودیت را تبدیل به فرم دوم نرمال نمائیم . بدین منظور موجودیتی به نام "کالا" ایجاد کرده ، کلید اصلی آن را برابر کالا قرار داده و

آیتم قیمت واحد را از موجودیت ردیف های فاکتور فروش خارج نموده و به این موجودیت منتقل می نمائیم. مثال فوق پس از تبدیل به فرم دوم نرمال به شکل ذیل خواهد بود :

| ردیف های فاکتور فروش | | فاکتور فروش |
|---|---|---|
| شماره فاکتور(قسمت اول کلید اصلی)<br>کالا (قسمت دوم کلید اصلی)<br>تعداد | ارتباط بین موجودیت پدر و فرزند بر اساس کلید اصلی موجودیت پدر (فاکتور فروش) | شماره فاکتور(کلید اصلی)<br>تاریخ فاکتور<br>کد مشتری<br>نام مشتری |
| ارتباط بین موجودیت پدر و فرزند بر اساس کلید اصلی موجودیت پدر (کالا) | | |
| کالا | | |
| کالا (کلید اصلی)<br>قیمت واحد | | |

### فرم سوم نرمال NF۳

موجودیت و یا جدولی در فرم سوم نرمال است که اولا" در فرم دوم نرمال بوده و ثانیا" تمام آیتم های غیر کلید آن وابستگی تابعی به کلید اصلی داشته باشند ، نه به یک آیتم غیر کلید .

مشکل : در صورتی که موجودیتی در فرم سوم نرمال نباشد ، آنگاه با تغییر آیتم یا آیتم های اطلاعاتی غیر وابسته به کلید اصلی در یک رکورد، تغییرات در سایر رکوردها اعمال نخواهد شد و دچار دوگانگی اطلاعات خواهیم شد (مثلا" یک مشتری با دو نام متفاوت) .

راه حل : کافی است آیتم های غیر کلیدی به هم وابسته را به موجودیت جدیدی منتقل و کلید اصلی موجودیت جدید را تعیین نمائیم ، آنگاه کلید اصلی موجودیت جدید را در موجودیت نرمال شده به عنوان یک کلید خارجی (Foreign Key) در نظر گرفت . در موجودیت "فاکتور فروش" مثال فوق آیتم نام مشتری وابستگی تابعی به آیتم کد مشتری دارد که خود یک آیتم غیر کلید است بنابر این باید نرمال سازی فرم سوم در خصوص آن اعمال شود . شکل ذیل نحوه انجام این کار را نشان می دهد :


## فرم بویس کد نرمال BCNF

فرم بویس کد دارای مفهوم جامع تری نسبت به فرم دوم و سوم نرمال است . در فرم دوم و سوم نرمال بحث بر سر وابستگی تابعی آیتم های غیر کلیدی به کلید اصلی است . اما در فرم بویس کد ، موجودیتی در فرم بویس کد نرمال است که اولا" در فرم اول نرمال بوده و ثانیا" تمام المان های غیر کلیدی آن کاملا" وابسته تابعی به یک کلید باشند و نه چیز دیگر . نکته حائز اهمیت در این فرم این است که بحث بر سر وابستگی

تابعی با یک کلید است نه فقط کلید اصلی. مفهوم فوق در خصوص موجودیت هائی که دارای چندین کلید هستند (Alternate Key) مطرح می شود .

### فرم چهارم نرمال NF٤

این فرم در خصوص موجودیت هائی است که ارتباط بین المان های آن یک ارتباط چند ارزشه و یا چند به چند باشد . به عنوان مثال ، موجودیت کلاس درس می تواند شامل چندین دانش آموز و چندین معلم باشد. در چنین مواردی ارتباط بین معلم و دانش آموز یک ارتباط چند به چند می باشد . در این حالت با ایجاد یک موجودیت رابط مابین موجودیت های مذکور، مشکل ارتباط چند به چند حل خواهد شد (بسیاری از سیستم های مدیریت بانک های رابطه ای نظیر MSSQL از رابطه چند به چند پشتیبانی نمی نمایند ، یعنی نمی توان بین دو جدول یک رابطه چند به چند ایجاد نمود). معمولا" تمام المان های موجودیت رابط ایجاد شده بخشی از کلید اصلی است .

### خلاصه

نرمال سازی فرم های دیگری نیز دارد که به دلیل نادر بودن و خاص بودن آنها در این مقاله به آنها اشاره نشده است . آنچه در خصوص نرمال سازی عمومیت دارد تا فرم سوم آن است ، یعنی در هنگام طراحی بانک های اطلاعاتی حتما" می بایست فرآیند نرمال سازی تا فرم سوم را انجام داد .
فرآیند نرمال سازی یک فرآیند تکراری (Recursive) است یعنی پس از هر مرحله نرمال سازی که منجر به ایجاد موجودیت های جدید می گردد ، فرآیند را باید از ابتدا تا انتها بر روی موجودیت های تازه ایجاد شده نیز اجرا نمود.

## طراحی بانک های اطلاعاتی ( مبانی مدل سازی)

طراحی پایگاه داده و ایجاد نمودار ارتباط موجودیت ها (ERD) یکی از مهمترین بخش های چرخه حیات توسعه یک نرم افزار است که در برخی موارد از آن به عنوان مهمترین بخش نیز نام برده می شود . مدل صحیح و به هنگام (Up To Date) اطلاعات می تواند به عنوان مهمترین ابزار مرجع برای مدیران بانک اطلاعاتی (DBAs) ، پیاده کنندگان نرم افزار و سایر اعضاء تیم توسعه دهنده نرم افزار باشد . فرآیند ایجاد مدل داده به تیم توسعه دهنده کمک می کند تا به پرسش های مطرح شده توسط کاربران نهائی سیستم پاسخ دهند .همچنین طراحی کارا و موثر پایگاه داده به تیم توسعه دهنده این امکان را می دهد تا سیستم را از همان

ابتدا در فرم مناسب پیاده سازی نمایند . ساخت سیستم با کیفیت فوق الذکر این امکان را به تیم توسعه دهنده خواهد داد تا زمان کلی انجام پروژه را کاهش دهند ، که در واقع این امر موجب کاهش هزینه های توسعه پروژه نیز خواهد شد .

با توجه به موارد فوق ، شعار طراحی خوب و جامع پایگاه داده این است که :

اول اندازه بگیر و بعد قیچی کن

طراحان خوب و خبره بانک های اطلاعاتی ، مبانی و اصول نرمال سازی پایگاه داده را همواره در خلال طراحی به خاطر داشته و آن را به کار خواهند گرفت . همانطور که در مبحث نرمال سازی بانک های اطلاعاتی به تفصیل بیان شد ، نرمال سازی فرآیندی در خلال طراحی پایگاه داده است که با چهار هدف عمده ذیل دنبال می شود :

- به حداقل رسانی افزونگی اطلاعات
- به حداقل رسانی تغییر ساختار اطلاعات
- به حداقل رسانی I/O سرور به منظور کاهش تعداد تراکنش ها (Transactions)
- و در نهایت حفظ یکپارچگی اطلاعات

برای طراحی بانک اطلاعاتی نرم افزار و مدل سازی آن می بایست اصول و تکنیک های ذیل را مد نظر داشت و از آنها استفاده نمود .

موجودیت (Entity) ، مجموعه ای از چیزهائی است که مربوط به بانک اطلاعاتی سیستم مورد نظر می باشد و یا به تعبیر دیگر هر آنچه که می خواهید در سیستم راجع به آن اطلاعات جمع آوری و نگهداری نمائید را شامل می شود . در مدل فیزیکی ، موجودیت تبدیل به جدول (Table) می شود .

خصلت (Attribute) یکی از مشخصه های توصیفی و یا مقداری موجودیت می باشد . در مدل فیزیکی یک خصلت به یک ستون (Column) و یا فیلد (Field) تبدیل می شود .

کلید اصلی (Primary Key) خصلت و یا ترکیبی از خصلت ها در یک موجودیت است که تضمین کننده یکتا بودن هر رخداد از موجودیت می باشد . خصلت یا خصلت های کلید اصلی نمی توانند فاقد ارزش باشند (NULL) و معمولا" کمتر تغییر می کنند . معمولا" سعی می شود جهت انتخاب کلید اصلی از خصلت هائی استفاده شود که کارائی بیشتری داشته و بهترین معرف موجودیت باشند (کارائی یک فیلد از

نوع **Integer** به مراتب بیشتر از فیلدی از نوع **Char** است ) . در صورتیکه نتوان در یک موجودیت خصلت یا خصلت هائی برای کلید اصلی شدن یافت ، آنگاه کلیدهای دستی برای این کار را ایجاد می کنیم که به آنها کلید **Artificial** می گویند .

ارتباط ( **Relationship**) ، ارتباط منطقی بین دو موجودیت است . یک ارتباط در واقع نشان دهنده قوانین کاری حاکم بر پروژه و اطلاعات آن است که معمولا" به صورت جملات فعلی توصیف می گردد . مثل ارتباط بین موجودیت کارمند و دپارتمان که به صورت جمله ذیل بیان می شود :
"کارمند شاغل است در دپارتمان" در این مثال ارتباط بین موجودیت کارمند و دپارتمان با جمله "شاغل است" توصیف میگردد .
دو نوع ارتباط می تواند بین موجودیت ها وجود داشته باشد :

- ارتباط یک به چند (**One To Many**) در این نوع ارتباط ، هر رخداد از موجودیت والد با چندین رخداد در موجودیت فرزند ارتباط دارد . به عنوان مثال چندین کارمند می توانند در یک دپارتمان شاغل به کار باشند .
- ارتباط چند به چند (**Many To Many**) . در این نوع ارتباط ، چند رخداد از یک موجودیت با چند رخداد از موجودیت دیگر ارتباط دارند . به عنوان مثال اگر یک کارمند بتواند در چند دپارتمان شاغل به کار باشد ، آنگاه ارتباط بین موجودیت کارمند و دپارتمان یک ارتباط چند به چند است . ارتباط چند به چند در طراحی پایگاه داده پذیرفته شده نیست چراکه علاوه بر افزونگی اطلاعات موجب عدم یکپارچگی اطلاعات نیز می گردد ، از اینرو باید این ارتباط طبق فرم چهارم نرمال سازی تبدیل به دو ارتباط یک به چند شود . همانطور که در مقاله نرمال سازی بانک های اطلاعاتی اشاره گردید برای حل این مشکل کافی است یک موجودیت واسط که به آن موجودیت **XREF** می گویند ایجاد و خصلت های کلید اصلی هردو موجودیت را به این موجودیت رابط منتقل نمود . با این عمل هریک از موجودیت های اصلی به عنوان والد این موجودیت رابط تلقی شده و یک ارتباط یک به چند بین آنها برقرار خواهد شد. در نتیجه یک ارتباط چند به چند تبدیل به دو ارتباط یک به چند خواهد شد . لازم به ذکر است که بسیاری از سیستم های مدیریت بانک های اطلاعاتی رابطه ای ( نظیر **MS SQL Server**) از ارتباط چند به چند پشتیبانی نمی کنند .

کلید خارجی (**Foreign Key**) . هرگاه خصلت(های) کلید اصلی موجودیت والد در موجودیت فرزند وجود داشته باشد (بر اساس ارتباط تعریف شده بین دو موجودیت) آنگاه این خصلت ها در موجودیت فرزند

، کلید خارجی نامیده می شوند . در واقع نمی توان هیچ رخدادی در موجودیت فرزند (که دارای کلید خارجی است) ایجاد نمود که رخداد مربوط به آن (بر اساس محتوای خصلت کلید خارجی) قبلا" در موجودیت والد ایجاد نشده باشد . آنگونه که از توصیف فوق استنباط می شود کلید خارجی تضمین کننده یکپارچگی اطلاعات در داخل پایگاه داده است چرا که باعث می شود که هیچ فرزند بدون والدی در بانک اطلاعاتی نداشته باشیم .

ارتباط (Relationship) بین دو موجودیت به دو مدل ذیل دسته بندی می گردد :

- ارتباط تعریف شده (identifying Relationship) . اگر کلید اصلی جدول والد بخشی (یا تمام) از کلید اصلی جدول فرزند باشد و یا به تعبیر دیگر بخشی از کلید اصلی موجودیت فرزند کلید خارجی نیز باشد ، در این حالت ارتباط مابین این دو موجودیت از نوع تعریف شده است .
- ارتباط تعریف نشده (Non-Identifying Relationship) ، برخلاف مورد فوق اگر کلید اصلی جدول والد در جدول فرزند وجود داشته باشد اما نه به عنوان بخشی از کلید اصلی آن و صرفا" به عنوان یک خصلت غیر کلید ، در این حالت ارتباط بین این دو موجودیت از نوع تعریف نشده می باشد . ارتباط تعریف نشده خود دارای دو حالت متفاوت به شرح ذیل است :

  mandatory non-identifying relationship ، زمانی است که خصلت کلید خارجی در موجودیت فرزند نتواند فاقد ارزش باشد (Not Allow NULL)

  relationship non-mandatory non-identifying ، زمانی است که خصلت کلید خارجی در موجودیت فرزند بتواند فاقد ارزش باشد (Allow NULL)

Cardinality ، به ما در فهم بیشتر ماهیت ارتباط مابین موجودیت والد و فرزند کمک می کند . جهت تشخیص Cardinality یک ارتباط کافی است به سئوال ذیل پاسخ داده شود :

" چه تعداد رخداد از موجودیت فرزند مرتبط است با هر رخداد از موجودیت والد؟ "

چهار نوع Cardinality مختلف به شرح ذیل وجود دارد :

- Zero or Many One To به این معنی که هر رخداد از موجودیت والد با هیچ و یا چند رخداد از موجودیت فرزند مرتبط است . به این نوع Common Cardinality می گویند.
- One Or Many One To به این معنی که هر رخداد از موجودیت والد با حداقل یک و یا چند رخداد از موجودیت فرزند مرتبط است . به این نوع P Cardinality می گویند .

- **Zero Or One One To** ، به این معنی که هر رخداد از موجودیت والد با هیچ و یا تنها یک رخداد از موجودیت فرزند مرتبط است . به این نوع **Z Cardinality** می گویند .
- **Exactly N One to** ، به این معنی که هر رخداد از موجودیت والد باید با **N** رخداد از موجودیت فرزند مرتبط باشد . به این نوع **N Cardinality** می گویند .

## خلاصه

- طراحی خوب بانک اطلاعاتی می تواند به تیم توسعه دهنده نرم افزار در کاهش زمان انجام پروژه و هزینه های آن کمک کند .
- طراحی بانک اطلاعاتی و مدل سازی آن به تیم توسعه دهنده نرم افزار کمک خواهد کرد تا درک بهتر و عمیقتری نسبت به نیازمندیهای کاربران نرم افزار پیدا کرده و در نتیجه نرم افزاری را توسعه دهند که در برگیرنده قوانین کاری و خواسته آنها باشد .
- یکی از اهداف اصلی طراحی بانک اطلاعاتی و مدل سازی آن ، مستقل بودن آن از پلت فرم است ، بنابر این اختیار انتخاب محیط و پلت فرم پیاده سازی فیزیکی پایگاه داده با تیم توسعه دهنده بوده و در ماحصل کار هیچ تغییری ایجاد نخواهد کرد .