数据库范式那些事

**简介**

      数据库范式在数据库设计中的地位一直很暧昧，教科书中对于数据库范式倒是都给出了学术性的定义，但实际应用中范式的应用却不甚乐观，这篇文章会用简单的语言和一个简单的数据库DEMO将一个不符合范式的数据库一步步从第一范式实现到第四范式。

**范式的目标**

      应用数据库范式可以带来许多好处，但是最重要的好处归结为三点：

      1.减少数据冗余（这是最主要的好处，其他好处都是由此而附带的）

      2.消除异常（插入异常，更新异常，删除异常）

      3.让数据组织的更加和谐…

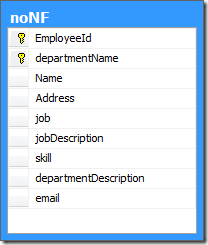
       但剑是双刃的，应用数据库范式同样也会带来弊端，这会在文章后面说到。

**什么是范式**

      简单的说，范式是为了消除重复数据减少冗余数据，从而让数据库内的数据更好的组织，让磁盘空间得到更有效利用的一种标准化标准，满足高等级的范式的先决条件是满足低等级范式。(比如满足2nf一定满足1nf)

**DEMO**

      让我们先从一个未经范式化的表看起,表如下：

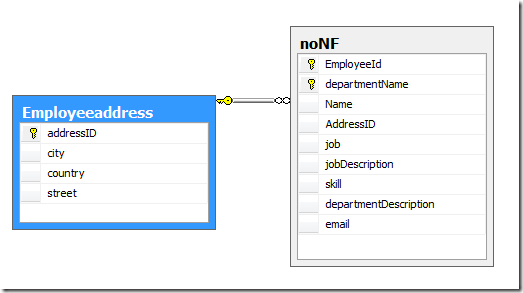
[](http://images.cnblogs.com/cnblogs_com/CareySon/WindowsLiveWriter/ebfdc5eb7fff_14F19/0nf_2.png)

先对表做一个简单说明，employeeId是员工id,departmentName是部门名称，job代表岗位，jobDescription是岗位说明，skill是员工技能，departmentDescription是部门说明，address是员工住址

**对表进行第一范式(1NF)**

*如果一个关系模式R的所有属性都是不可分的基本数据项，则R∈1NF。*

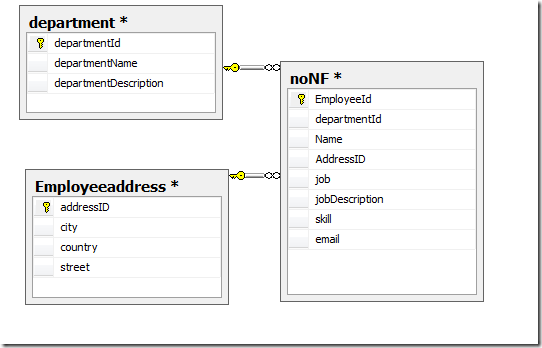
简单的说,第一范式就是每一个属性都不可再分。不符合第一范式则不能称为关系数据库。对于上表，不难看出Address是可以再分的，比如”北京市XX路XX小区XX号”，着显然不符合第一范式，对其应用第一范式则需要将此属性分解到另一个表,如下:

[](http://images.cnblogs.com/cnblogs_com/CareySon/WindowsLiveWriter/ebfdc5eb7fff_14F19/1nf_2.png)

**对表进行第二范式(2NF)**

*若关系模式R∈1NF，并且每一个非主属性都*[*完全函数依赖*](http://baike.baidu.com/view/228997.htm)*于R的码，则R∈2NF*

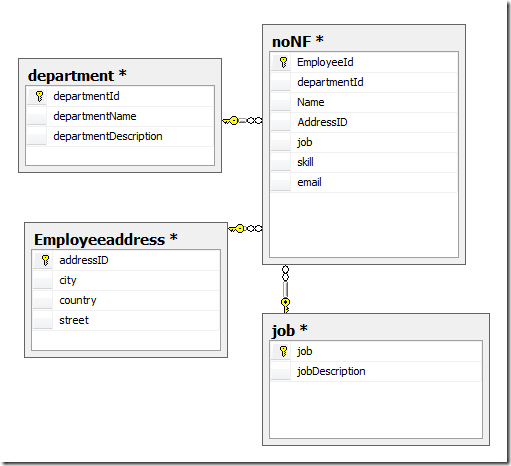
简单的说，是表中的属性必须完全依赖于全部主键，而不是部分主键.所以只有一个主键的表如果符合第一范式，那一定是第二范式。这样做的目的是进一步减少插入异常和更新异常。在上表中，departmentDescription是由主键DepartmentName所决定，但却不是由主键EmployeeID决定，所以departmentDescription只依赖于两个主键中的一个，故要departmentDescription对主键是部分依赖，对其应用第二范式如下表：

[](http://images.cnblogs.com/cnblogs_com/CareySon/WindowsLiveWriter/ebfdc5eb7fff_14F19/3nf_2.png)

**对表进行第三范式(3NF)**

[*关系模式*](http://baike.baidu.com/view/68347.htm)*R<U，F> 中若不存在这样的码X、属性组Y及非主属性Z（Z ⊆ Y）, 使得X→Y，Y→Z，成立，则称R<U，F> ∈ 3NF。*

简单的说，第三范式是为了消除数据库中关键字之间的依赖关系，在上面经过第二范式化的表中，可以看出jobDescription(岗位职责)是由job(岗位)所决定，则jobDescription依赖于job,可以看出这不符合第三范式，对表进行第三范式后的关系图为：

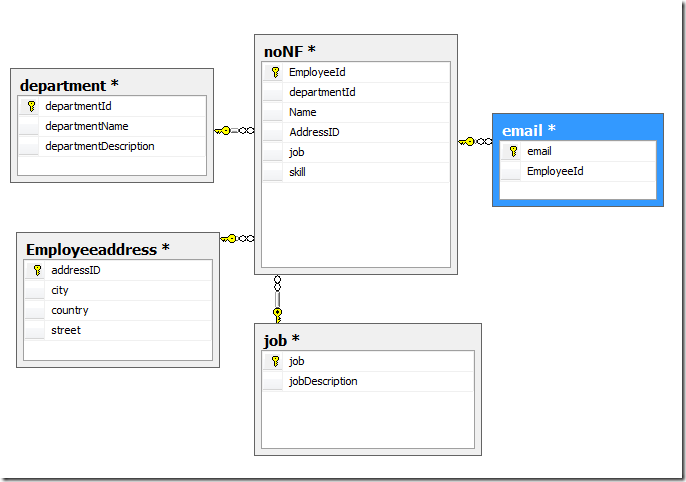
[](http://images.cnblogs.com/cnblogs_com/CareySon/WindowsLiveWriter/ebfdc5eb7fff_14F19/3nf1_2.png)

上表中，已经不存在数据库属性互相依赖的问题，所以符合第三范式

**对表进行BC范式(BCNF)**

*设*[*关系模式*](http://baike.baidu.com/view/68347.htm)*R<U，F>∈1NF，如果对于R的每个函数依赖X→Y，若Y不属于X，则X必含有候选码，那么R∈BCNF。*

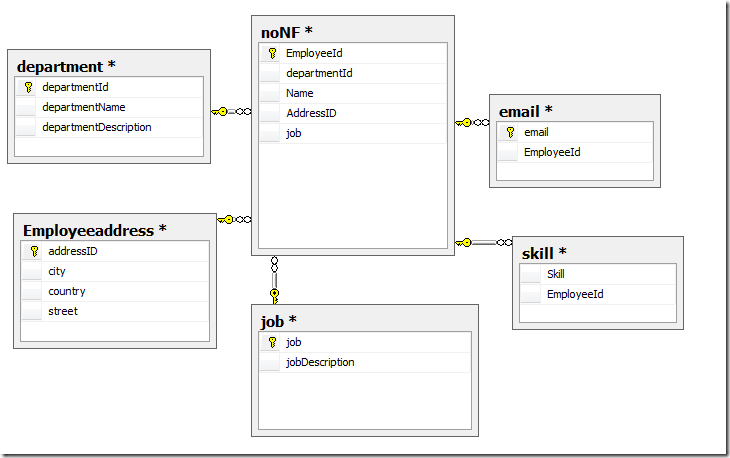
简单的说，bc范式是在第三范式的基础上的一种特殊情况，既每个表中只有一个候选键（在一个数据库中每行的值都不相同，则可称为候选键），在上面第三范式的noNf表中可以看出，每一个员工的email都是唯一的（难道两个人用同一个email??）则，此表不符合bc范式，对其进行bc范式化后的关系图为:

[](http://images.cnblogs.com/cnblogs_com/CareySon/WindowsLiveWriter/ebfdc5eb7fff_14F19/bcnf_2.png)

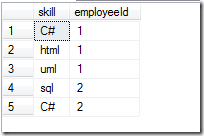
**对表进行第四范式(4NF)**

[*关系模式*](http://baike.baidu.com/view/68347.htm)*R<U，F>∈1NF，如果对于R的每个非平凡多值依赖X→→Y（Y ⊆ X），X都含有候选码，则R∈4NF。*

简单的说，第四范式是消除表中的多值依赖，也就是说可以减少维护数据一致性的工作。对于上面bc范式化的表中，对于员工的skill，两个可能的值是”C#,sql,javascript”和“C#，UML,Ruby”,可以看出，这个数据库属性存在多个值，这就可能造成数据库内容不一致的问题，比如第一个值写的是”C#”,而第二个值写的是”C#.net”,解决办法是将多值属性放入一个新表，则第四范式化后的关系图如下：

[](http://images.cnblogs.com/cnblogs_com/CareySon/WindowsLiveWriter/ebfdc5eb7fff_14F19/4nf_2.png)

而对于skill表则可能的值为:

[](http://images.cnblogs.com/cnblogs_com/CareySon/WindowsLiveWriter/ebfdc5eb7fff_14F19/4nfdemo_2.png)

**总结**

     上面对于数据库范式进行分解的过程中不难看出，应用的范式登记越高，则表越多。表多会带来很多问题：

1 查询时要连接多个表，增加了查询的复杂度

2 查询时需要连接多个表，降低了数据库查询性能

而现在的情况，磁盘空间成本基本可以忽略不计，所以数据冗余所造成的问题也并不是应用数据库范式的理由。

因此，并不是应用的范式越高越好，要看实际情况而定。第三范式已经很大程度上减少了数据冗余，并且减少了造成插入异常，更新异常，和删除异常了。我个人观点认为，大多数情况应用到第三范式已经足够，在一定情况下第二范式也是可以的。

由于本人对数据库研究还处于初级阶段，所以上述如有不当之处，还望高手不吝指教…