

# 东方国信移动事业部产品研发一部

## 数据库开发手册

### 一、开发规范

#### 1. 表名定义规范

(1) 数据库表的命名以是名词的复数形式，如 cities, categories, friends 等等。这一条不是硬性规定，可以看个人习惯，统一就好。

(2) 如果表名由几个单词组成，则单词间用下划线 (“\_”) 分割，如 subscribed\_pois, poi\_categories 等。

(3) 表名尽量用全名。

(4) 表名限制在 30 个字符内。当表的全名超过 30 字符时，可用缩写来减少表名的长度，如 description-->desc; information-->info; address-->addr 等。

(5) 同类数据库对象要有统一的前缀名

一个比较复杂的系统，数据库表往往很多，要快速定位自己需要的表，不太容易。因此，在同一个模块中的表，需要加上一个统一的前缀。例如，系统管理相关的表可以统一加一个前缀 sys\_，用户表 sys\_user；角色表 sys\_role；组织表 sys\_ornization；角色用户表 sys\_role\_user 等。

#### 2. 字段名定义规范

(1) 数据库字段全部采用英文单词。

(2) 字段长度不宜过长，尽量简洁明了。

字段名限制在 30 个字符内。当字段名超过 30 字符时，可用缩写来减少字段名的长度，如 description-->desc; information-->info; address-->addr 等。

(3) 如果表或者是字段的名称仅有一个单词，那么建议不使用缩写，而是用完整的单词。

(4) 如果表或者字段由多个单词构成，单词之间用 “\_” 隔开，不要用驼峰式命名，例如用户名字段，应为 user\_name，而不是 userName。

(5) 常用的字段例如 name，不宜直接用 name，最好定义为 xx\_name。

防止关联查询的时候，两个表的字段名称一样，不方便处理。可能还需要起别名 as 。

(6) 字段名称尽量避免中文拼音。

(7) 字段尽量避免关键字。

(8) 表中应该避免可为空的列。

虽然表中允许空列，但是，空字段是一种比较特殊的数据类型。数据库在处理的时候，需要进行特殊的处理。如此的话，就会增加数据库处理记录的复杂性。当表中有比较多的空字段时，在同等条件下，数据库处理的性能会降低许多。

解决方法：通过设置默认值的形式，来避免空字段的产生。

(9) 不能为空的字段最好加上默认值。

所有字段在设计时，除以下数据类型 timestamp、datetime 外，必须有默认值。字符型的默认值为一个空字符串''；数值型的默认值为数值 0；布尔型的默认值为数值 0；系统中所有逻辑型中数值 0 表示为“假”；数值 1 表示为“真”。datetime、smalldatetime 类型的字段没有默认值，必须为 NULL。

### 3. 字段数据类型规范

(1) 如果需要时分秒时间记录，建议用 datetime 类型。

(2) 如果需要的更多是日期的查询，建议用 varchar 型，不用 DATE。

(3) 如果需要比时分秒更精确的时间记录，建议用 long 型，用应用生成时间戳，System.currentTimeMillis() 进行存储。

(4) 字符串存储能用 varchar 不要用 text，varchar 存储，搜索性能都高于 text，text 查询是会产生临时磁盘文件，性能差。

(5) 备注字段如果不是特别长，尽量用 varchar 不用 text，在 daoImpl 层处理超出长度部分，进行截取存储。

(6) 数字类型选择，放弃 float 和 double，尽量用 decimal。

(7) 放弃用 BLOB 二进制数据类型，如果涉及大数据存储，进行 DB-索引-文件存储系统模式来处理。

## 4. 索引设计规范

- (1) 根据业务需求设计索引。
- (2) 索引不要过多，影响插入更新性能。
- (3) 一个字段的值范围很小，不要设置索引，索引不生效还浪费插入性能。
- (4) 如果有幂等性需求，设置数据库唯一索引。
- (5) null 值对索引是一大伤害，所以不要让索引的列有 null 值存在。
- (6) 尽量加索引的字段的数据类型小，也就是能用整数不用 varchar 能用短的 varchar 不用长的 varchar，不要在 text 上设置索引。

例如，账户表，userId 设置为唯一索引，一个人只能有一个账户；收支流水表的 userId 设置普通索引，查询效率高；create\_time 默认加索引；a, b, c 组合索引，索引构建是用 a + b + c 进行构建的索引，如果是 btree 索引，查询的时候 abc | ab | a 三种查询条件都会走索引，不需要对 a 和 ab 重复创建索引，但是 like '%b' | '%bc' | '%c' 不会走索引，所以，组合索引，一定使用频率高的放在最左边。

## 5. 注释规范

- (1) 每个字段要有注释。
- (2) 每个表名要有注释。
- (3) 字段的取值含义或者范围，枚举值要有注释。

## 二、设计原则

### 1. 原始单据与实体之间的关系

可以是一对一、一对多、多对多的关系。在一般情况下，它们是一对一的关系：即一张原始单据对应且只对应一个实体。在特殊情况下，它们可能是一对多或多对一的关系，即一张原始单证对应多个实体，或多张原始单证对应一个实体。这里的实体可以理解为基本表。明确这种对应关系后，对我们设计录入界面大有好处。

〔例 1〕：一份员工履历资料，在人力资源信息系统中，就对应三个基本表：员工基本情况表、社会关系表、工作简历表。这就是“一张原始单证对应多个实体”的典型例子。

### 2. 主键与外键

一般而言，一个实体不能既无主键又无外键。在 E--R 图中，处于叶子部位的实体，可以定义主键，也可以不定义主键（因为它无子孙），但必须要有外键（因为它有父亲）。

主键与外键的设计，在全局数据库的设计中，占有重要地位。当全局数据库的设计完成以后，有个美国数据库设计专家说：“键，到处都是键，除了键之外，什么也没有”，这就是他的数据库设计经验之谈，也反映了他对信息系统核心（数据模型）的高度抽象思想。因为：主键是实体的高度抽象，主键与外键的配对，表示实体之间的连接。

### 3. 基本表的性质

基本表与中间表、临时表不同，因为它具有如下四个特性：

- （1）原子性。基本表中的字段是不可再分解的。
- （2）原始性。基本表中的记录是原始数据（基础数据）的记录。
- （3）演绎性。由基本表与代码表中的数据，可以派生出所有的输出数据。

(4) 稳定性。基本表的结构是相对稳定的，表中的记录是要长期保存的。

理解基本表的性质后，在设计数据库时，就能将基本表与中间表、临时表区分开来。

#### 4. 范式标准

基本表及其字段之间的关系，应尽量满足第三范式。但是，满足第三范式的数据库设计，往往不是最好的设计。为了提高数据库的运行效率，常常需要降低范式标准：适当增加冗余，达到以空间换时间的目的。

〔例 2〕：有一张存放商品的基本表，如表 1 所示。“金额”这个字段的存在，表明该表的设计不满足第三范式，因为“金额”可以由“单价”乘以“数量”得到，说明“金额”是冗余字段。但是，增加“金额”这个冗余字段，可以提高查询统计的速度，这就是以空间换时间的作法。在 Rose 2002 中，规定列有两种类型：数据列和计算列。“金额”这样的列被称为“计算列”，而“单价”和“数量”这样的列被称为“数据列”。

表 1 商品表的表结构

商品名称	商品型号	单价	数量	金额
电视机	29 吋	2,500	40	100,000

#### 5. 通俗的理解三个范式

通俗的理解三个范式，对于数据库设计大有好处。在数据库设计中，为了更好地应用三个范式，就必须通俗的理解三个范式（通俗地理解是够用的理解，并不是最科学最准确的理解）：

第一范式：1NF 是对属性的原子性约束，要求属性具有原子性，不可再分解；

第二范式：2NF 是对记录的惟一性约束，要求记录有惟一标识，即实体的惟一性；

第三范式：3NF 是对字段冗余性的约束，即任何字段不能由其他字段派生出来，它要求字段没有冗余。

没有冗余的数据库设计可以做到。但是，没有冗余的数据库未必是最好的数据库，有时为了提高运行效率，就必须降低范式标准，适当保留冗余数据。具体

做法是：在概念数据模型设计时遵守第三范式，降低范式标准的工作放到物理数据模型设计时考虑。降低范式就是增加字段，允许冗余。

## 6. 要善于识别与正确处理多对多的关系

若两个实体之间存在多对多的关系，则应消除这种关系。消除的办法是，在两者之间增加第三个实体。这样，原来一个多对多的关系，现在变为两个一对多的关系。要将原来两个实体的属性合理地分配到三个实体中去。这里的第三个实体，实质上是一个较复杂的关系，它对应一张基本表。一般来讲，数据库设计工具不能识别多对多的关系，但能处理多对多的关系。

〔例 3〕：在“图书馆信息系统”中，“图书”是一个实体，“读者”也是一个实体。这两个实体之间的关系，是一个典型的多对多关系：一本图书在不同时间可以被多个读者借阅，一个读者又可以借多本图书。为此，要在二者之间增加第三个实体，该实体取名为“借还书”，它的属性为：借还时间、借还标志（0 表示借书，1 表示还书），另外，它还应该有两个外键（“图书”的主键，“读者”的主键），使它能与“图书”和“读者”连接。

## 7. 主键 PK 的取值方法

PK 是供程序员使用的表间连接工具，可以是一无物理意义的数字串，由程序自动加 1 来实现。也可以是有物理意义的字段名或字段名的组合。不过前者比后者好。当 PK 是字段名的组合时，建议字段的个数不要太多，多了不但索引占用空间大，而且速度也慢。

## 8. 正确认识数据冗余

主键与外键在多表中的重复出现，不属于数据冗余，这个概念必须清楚，事实上有许多人还不清楚。非键字段的重复出现，才是数据冗余！而且是一种低级冗余，即重复性的冗余。高级冗余不是字段的重复出现，而是字段的派生出现。

〔例 4〕：商品中的“单价、数量、金额”三个字段，“金额”就是由“单价”乘以“数量”派生出来的，它就是冗余，而且是一种高级冗余。冗余的目的是为了提高处理速度。只有低级冗余才会增加数据的不一致性，因为同一数据，可能

从不同时间、地点、角色上多次录入。因此，我们提倡高级冗余（派生性冗余），反对低级冗余（重复性冗余）。

## 9. E--R 图没有标准答案

信息系统的 E--R 图没有标准答案，因为它的设计与画法不是惟一的，只要它覆盖了系统需求的业务范围和功能内容，就是可行的。反之要修改 E--R 图。尽管它没有惟一的标准答案，并不意味着可以随意设计。好的 E--R 图的标准是：结构清晰、关联简洁、实体个数适中、属性分配合理、没有低级冗余。

## 10. 视图技术在数据库设计中很有用

与基本表、代码表、中间表不同，视图是一种虚表，它依赖数据源的实表而存在。视图是供程序员使用数据库的一个窗口，是基表数据综合的一种形式，是数据处理的一种方法，是用户数据保密的一种手段。为了进行复杂处理、提高运算速度和节省存储空间，视图的定义深度一般不得超过三层。若三层视图仍不够用，则应在视图上定义临时表，在临时表上再定义视图。这样反复交迭定义，视图的深度就不受限制了。

对于某些与国家政治、经济、技术、军事和安全利益有关的信息系统，视图的作用更加重要。这些系统的基本表完成物理设计之后，立即在基本表上建立第一层视图，这层视图的个数和结构，与基本表的个数和结构是完全相同。并且规定，所有的程序员，一律只准在视图上操作。只有数据库管理员，带着多个人员共同掌握的“安全钥匙”，才能直接在基本表上操作。

## 11. 中间表、报表和临时表

中间表是存放统计数据的表，它是为数据仓库、输出报表或查询结果而设计的，有时它没有主键与外键（数据仓库除外）。临时表是程序员个人设计的，存放临时记录，为个人所用。基表和中间表由 DBA 维护，临时表由程序员自己用程序自动维护。

## 12. 完整性约束表现在三个方面

域的完整性：用 Check 来实现约束，在数据库设计工具中，对字段的取值范围进行定义时，有一个 Check 按钮，通过它定义字段的值域。

参照完整性：用 PK、FK、表级触发器来实现。

用户定义完整性：它是一些业务规则，用存储过程和触发器来实现。

## 13. 防止数据库设计打补丁的方法是“三少原则”

(1) 一个数据库中表的个数越少越好。只有表的个数少了，才能说明系统的 E—R 图少而精，去掉了重复的多余的实体，形成了对客观世界的高度抽象，进行了系统的数据集成，防止了打补丁式的设计；

(2) 一个表中组合主键的字段个数越少越好。因为主键的作用，一是建主键索引，二是做为子表的外键，所以组合主键的字段个数少了，不仅节省了运行时间，而且节省了索引存储空间；

(3) 一个表中的字段个数越少越好。只有字段的个数少了，才能说明在系统中不存在数据重复，且很少有数据冗余，更重要的是督促读者学会“列变行”，这样就防止了将子表中的字段拉入到主表中去，在主表中留下许多空余的字段。所谓“列变行”，就是将主表中的一部分内容拉出去，另外单独建一个子表。这个方法很简单，有的人就是不习惯、不采纳、不执行。

数据库设计的实用原则是：在数据冗余和处理速度之间找到合适的平衡点。“三少”是一个整体概念，综合观点，不能孤立某一个原则。该原则是相对的，不是绝对的。“三多”原则肯定是错误的。试想：若覆盖系统同样的功能，一百个实体（共一千个属性）的 E—R 图，肯定比二百个实体（共二千个属性）的 E—R 图，要好得多。

提倡“三少”原则，是叫读者学会利用数据库设计技术进行系统的数据集成。数据集成的步骤是将文件系统集成为应用数据库，将应用数据库集成为主题数据库，将主题数据库集成为全局综合数据库。集成的程度越高，数据共享性就越强，信息孤岛现象就越少，整个企业信息系统的全局 E—R 图中实体的个数、主键的个数、属性的个数就会越少。

提倡“三少”原则的目的，是防止读者利用打补丁技术，不断地对数据库进



行增删改，使企业数据库变成了随意设计数据库表的“垃圾堆”，或数据库表的“大杂院”，最后造成数据库中的基本表、代码表、中间表、临时表杂乱无章，不计其数（即动态创建表而增加表数量），导致企事业单位的信息系统无法维护而瘫痪。

“三多”原则任何人都可以做到，该原则是“打补丁方法”设计数据库的歪理学说。“三少”原则是少而精的原则，它要求有较高的数据库设计技巧与艺术，不是任何人都能做到的，因为该原则是杜绝用“打补丁方法”设计数据库的理论依据。

#### 14. 提高数据库运行效率的办法

在给定的系统硬件和系统软件条件下，提高数据库系统的运行效率的办法是：

（1）在数据库物理设计时，降低范式，增加冗余，少用触发器，多用存储过程。

（2）当计算非常复杂、而且记录条数非常巨大时（例如一千万条），复杂计算要先在数据库外面，以文件系统方式用 C++ 语言计算处理完成之后，最后才入库追加到表中去。这是电信计费系统设计的经验。

（3）发现某个表的记录太多，例如超过一千万条，则要对表进行水平分割。水平分割的做法是，以该表主键 PK 的某个值为界线，将该表的记录水平分割为两个表（即可以表维护，表行数过大手动分割为两个，建个两表 union 的视图对程序透明）。若发现某个表的字段太多，例如超过八十个，则垂直分割该表，将原来的一个表分解为两个表。

（4）对数据库管理系统 DBMS 进行系统优化，即优化各种系统参数，如缓冲区个数。

（5）在使用面向数据的 SQL 语言进行程序设计时，尽量采取优化算法。

总之，要提高数据库的运行效率，必须从数据库系统级优化、数据库设计级优化、程序实现级优化，这三个层次上同时下功夫。

上述十四个技巧，是许多人在大量的数据库分析与设计实践中，逐步总结出来的，要消化理解，实事求是，灵活掌握。并逐步做到：在应用中发展，在发展中应用。