数据库设计时应当注意的几个问题

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，即一张原始单据对应多个实体，或多张原始单据对应一个实体。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“一张原始单据对应多个实体”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. 通俗地理解三个范式

通俗地理解三个范式，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，就必须通俗地理解三个范式(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。请读者想想：这是为什么？

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城。
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，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，则要对该表进行水平分割。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。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。

15. 设计索引前的必要准备

决定设计数据库索引前，务必先对业务上涉及到的所有查询进行整理并优化，并针对整理结果对表建立索引与复合索引。建立索引的原则是：合理利用复合索引的最左前缀规则，在保证尽可能多的覆盖查询筛选条件的情况下，建立尽可能少的索引。

在业务层面就应当避免可能遍历整个表的查询，将最大可能的返回结果控制在可控的数量内。例如：让所有的筛选都必须先选定一个特定的时间，并根据实际情况对这个时间的最大跨度做限制，然后只在这个时间段内筛选数据，这使得数据库总遍历条数和查询时间变得可控。