# 数据库设计

简单来说，数据库设计就是根据业务系统的具体需要，结合我们所选用的DBMS（数据库管理系统），为这个业务系统构造出最优的数据存储模型，并建立好数据库中的表结构及表之间的关联关系的过程。使之能有效的对应用系统中的数据进行存储，并可以高效的对已经存储的数据进行访问。

**为什么需要数据库设计？**

|  |  |
| --- | --- |
| 优良的设计 | 糟糕的设计 |
| 减少数据冗余 | 存在大量数据冗余 |
| 避免数据维护异常 | 存在数据插入，更新，删除异常 |
| 节约存储空间 | 浪费大量存储空间 |
| 高效的访问 | 访问数据低效 |

# 需求分析

## 步骤



**需求分析：**

1. 数据是什么
2. 数据有哪些属性
3. 数据和属性各自的特点有哪些

**逻辑设计：**使用ER图对数据库进行逻辑建模

**物理设计：**根据数据库自身的特点把逻辑设计转换为物理设计

**维护优化：**

1. 新的需求进行建表
2. 索引优化
3. 大表拆分

## 重要性

为什么需要需求分析？

1. 了解系统中所需要存储的数据
2. 了解数据的存储特点
3. 了解数据的生命周期

注：了解需要存储的数据类型，比如有的数据是有时效性的，可以采用过期清理和归档的方式处理；还有一些数据增长很快，数据量很大，但是不是核心数据，可以采用分库分表的方式存储；日志一般不适合存储在数据库中，如果要存储在数据库中需要先设计好归档方案。

需要搞清楚这样的一些关系：

1. 实体及实体之间的关系（1对1,1对多,多对多）
2. 实体所包含的属性有什么？
3. 哪些属性或属性的组合可以唯一标识一个实体

注：了解实体的特性和增长量，对于增长快的可以采用归档和分库分表。

## 实例

以一个小型的电商网站为例，在这个电商网站的系统中包括了几个核心模块：用户模块，商品模块，订单模块，购物车模块，供应商模块。

**用户模块**

1. 用于记录注册用户信息
2. 包括属性：用户名、密码、电话、邮箱、身份证号、地址、姓名、昵称
3. 可选唯一标识属性：用户名、身份证、电话
4. 存储特点：随系统上线时间逐渐增加，需要永久存储

注：对于用户是不能做数据删除和归档的，因此需要永久保存。

**商品模块**

1. 用于记录网站中所销售的商品信息
2. 包括属性：商品编码、商品名称、商品描述、商品品类、供应商名称、重量、有效期、价格
3. 可选唯一标识属性：（商品名称，供应商名称）、（商品编码）
4. 存储特点：对于下线商品可以归档存储

注：商品是越来越多的，因此不可以删除，但是有些商品可以下线，可以做归档（但是不是删除），需要永久存储的。

**订单模块**

1. 用于用户订购商品的信息
2. 包括属性：订单号、用户姓名、用户电话、收货地址、商品编号、商品名称、数量、价格、订单状态、支付状态、订单类型
3. 可选唯一标识属性：（订单号）
4. 存储特点：永久存储（分库、分表存储）

注：订单数量大，需要永久存储。

**购物车模块**

1. 用户保存用户购物时选择的商品
2. 包括属性：用户名、商品编号、商品名称、商品价格、商品描述、商品分类、加入时间、商品数量
3. 可选唯一标识：（用户名、商品编号、加入时间）、（购物车编号）
4. 存储特点：不用永久存储（设置归档、清理规则）

供应商模块

1. 用户保存所销售商品的供应商信息
2. 包括属性：供应商标号、供应商名称、联系人、电话、营业执照号、地址、法人
3. 可选唯一标识：（供应商编号）、（营业执照号）
4. 存储特点：永久存储

**模块间关系**



# 逻辑设计

## ER图

关系：一个关系对应通常所说的一张表

元组：表中的一行即一个元组

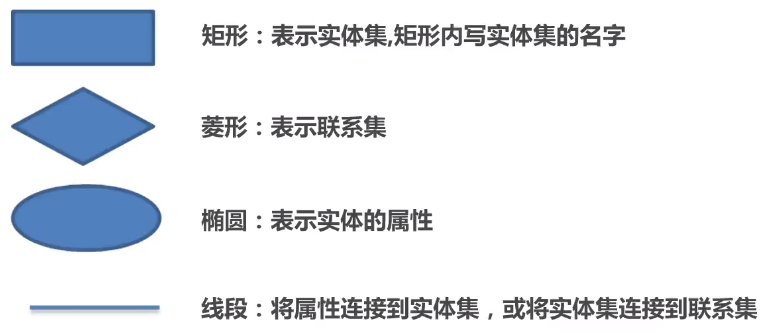
属性：表中的一列即一个属性，每一个属性都有一个名称，称为属性名

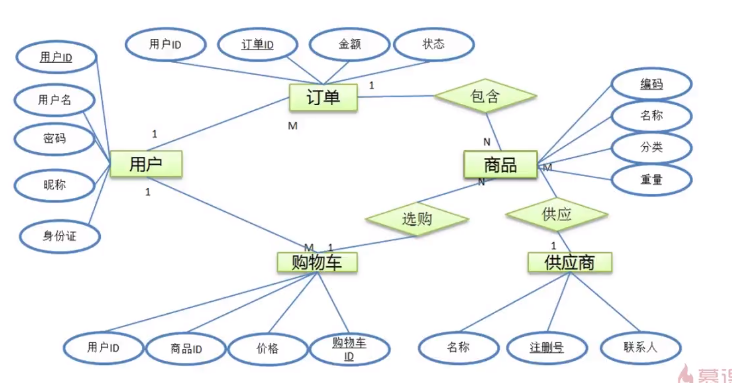
候选码：表中的某个属性组，它可以唯一确定一个元组

主码：一个关系有多个候选码，选定其中一个为主码

域：属性的取值范围

分量：元组中的一个属性值





## 设计范式



常见数据库设计范式包括：第一范式，第二范式，第三范式及BC范式，当然还有第四及第五范式。

### 操作异常

插入异常：如果某实体随着另一个实体的存在而存在，即缺少某个实体时无法表示这个实体，那么这个表就存在插入异常。

更新异常：如果更新表所对应的某个实体实例的单独属性时，需要将多行更新，那么就说这个表存在更新异常。

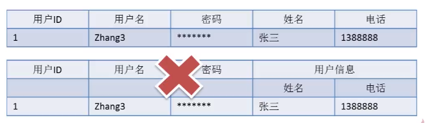
删除异常：如果删除表的某一行来反映某实体实例，失效时导致另一个不同实体实例信息丢失，那么这个表就存在删除异常。

### 数据冗余

数据冗余是指相同的数据再多个地方存在，或者说表中的某个列可以由其他列计算得到，这样就说表中存在着数据冗余。

## 第一范式(1NF)

数据库中的所有字段都是单一属性，不可再分的。这个单一属性是由基本的数据类型所构成的，如整数，浮点数，字符串等。换句话说，第一范式要求数据库中的表都是二维表。



## 第二范式(2NF)

数据库的表中不存在非关键字段对任意候选关键字段的部分函数依赖。

部分函数依赖是指存在着组合关键字中的某一关键字决定非关键字的情况。

换言之，所有单关键字段的表都符合第二范式。



由于供应商和商品之间是多对多的关系，所以只有使用商品名称和供应商名称才可以唯一标识出一件商品。也就是商品名称和供应商名称是一组组合关键字，上述表中存在以下的部分函数依赖关系：

（商品名称）🡪（价格，描述，重量，商品有效期）

（供应商名称）🡪（供应商电话）



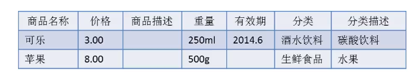
存在的问题：

1. 插入异常
2. 删除异常
3. 更新异常
4. 数据冗余



## 第三范式(3NF)

第三范式是在第二范式的基础上定义的，如果数据表中不存在非关键字段，对任意候选关键字段的传递函数依赖则符合第三范式。



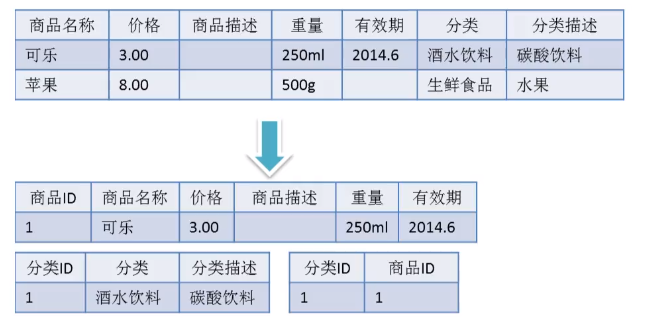
存在以下传递函数依赖关系：

（商品名称）🡪（分类）🡪（分类描述）

也就是说存在非关键字段“分类描述”，对关键字段“商品名称”的传递函数依赖。

存在问题：

（分类，分类描述）对于每一个商品都会进行记录，所以存在着数据冗余。同时也还存在数据的插入，更新及删除异常。



## BC范式

在第三范式的基础上，数据库表中如果不存在任何字段对任一候选主键字段的传递函数依赖则符合BC范式。

换言之，如果复合关键字，则复合关键字之间也不能存在函数依赖关系。



假定：供应商联系人只能受雇于一家供应商，每家供应商可以供应多个商品，则存在如下决定关系：

（供应商，商品ID）🡪（联系人，商品数量）

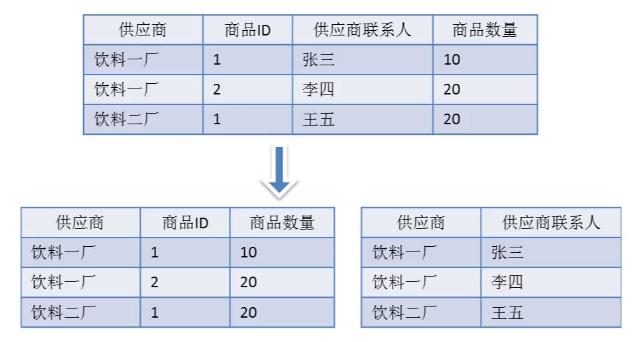
（联系人，商品ID）🡪（供应商，商品数量）

存在下列关系因此不符合BCNF要求：

（供应商）🡪（供应商联系人）

（供应商联系人）🡪（供应商）

并且存在数据操作异常及数据冗余



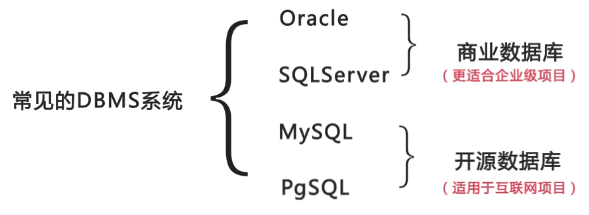
# 物理设计

## 概述

物理设计需要：

1. 选择合适的数据库管理系统
2. 定义数据库、表及字段的命名规范
3. 根据所选的DBMS系统选择合适的字段类型
4. 反范式化设计

## 选型



## MySQL存储引擎



## 表/字段命名规则

1. 可读性原则

使用大写和小写来格式化的库对象名称以获得良好的可读性。

例如：使用CustAddress而不是custaddress来提高可读性（这里要注意有些DBMS系统对表名大小写敏感）。

1. 表意性原则

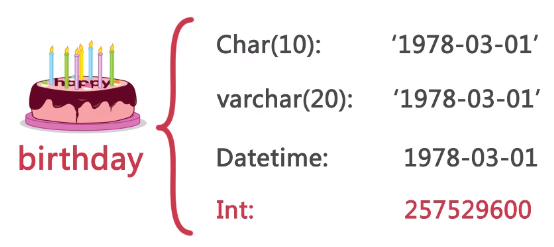
对象的名字应该能够描述它所标识的对象。例如，对于表，表的名称应该能够体现表中存储的数据内容，对于存储过程，存储过程名称应该能够体现存储过程的功能。

1. 长名原则

尽可能少使用或者不使用缩写，适用于数据库名之外的任一对象。

## 字段类型选择

生日的数据存储方式：



### 原则

列的数据类型一方面影响数据存储空间的开销，另一方面也会影响数据查询性能。当一个列可以选择多种数据类型时，应该优先考虑数字类型，其次是日期或二进制类型，最后是字符类型。对于相同级别的数据类型，应该优先选择占用空间小的数据类型。

以上选择原则主要是从下面两个角度考虑：

1. 在对数据进行比较（查询条件、JOIN条件及排序）操作时：同样的数据，字符往往比数字处理慢。
2. 在数据库中，数据处理以页为单位，列的长度越小，利于性能提升。



### char与varchar

选择的原则：

1. 如果列中要存储的数据长度差不多是一致的，则应该考虑用char，否则应该考虑使用varchar；
2. 如果列中的最大数据长度小于50Byte，则一般也考虑用char；

注：当然，如果这个列很少用，则基于节省空间和减少I/O的考虑，还是可以选择用varchar。

1. 一般不宜定义大于50Byte的char类型列。

### decimal与float

选择的原则：

1. decimal用于存储精确数据，而float只能用于存储非精确数据，故精确数据只能选择使用decimal类型；
2. 由于float的存储空间开销一般比decimal小，故非精确数据优先选择使用float类型。

注：精确到7位小数只需要4个字节，而精确到15位小数只需要8字节。

### 时间类型

1. 使用int来存储时间字段的优缺点：

优点：字段长度比datetime小

缺点：使用不方便，要进行函数转换

限制：只能存储到2038-1-19 11:14:07即2^32为2147483648

1. 需要存储的时间粒度：年月日小时分秒周

## 注意事项

### 选择主键

1. 区分业务主键和数据库主键

业务主键用于标识业务数据，进行表与表之间的关联；

数据库主键为了优化数据存储（Innodb会生成6个字节的隐含主键）

1. 根据数据库的类型，考虑主键是否要顺序增长

有些数据库是按照主键的顺序逻辑存储的

1. 主键的字段类型所占空间要尽可能小

对于使用聚集索引方式存储的表，每个索引后都会附加主键信息

### 避免使用外键约束

1. 降低数据导入的效率
2. 增加维护成本
3. 虽然不建议使用外键约束，但是相关联的列上一定要建立索引

### 避免使用触发器

1. 降低数据导入的效率；
2. 可能会出现意想不到的数据异常；
3. 使业务六级变得复杂。

### 预留字段

1. 无法准确的知道预留字段的类型；
2. 无法准确的知道预留字段中所存储的内容；
3. 后期维护预留字段所要的成本。

## 反范式化表设计

# 维护优化

## 概述

维护和优化需要：

1. 维护数据字典
2. 维护索引
3. 维护表结构
4. 在适当的时候对表进行水平拆分或者垂直拆分

## 数据字典

维护数据字典的方法：

1. 使用第三方工具对数据字典进行维护
2. 利用数据库本身的备注字段来维护数据字典，以MySQL为例：

CREATE TABLE customer(

cust\_id INT AUTO\_INCREMENT NOT NULL COMMENT ‘自增ID’,

cust\_name VARCHAR(10) NOT NULL COMMENT ‘客户姓名’,

PRIMARY KEY(cust\_id)

)COMMENT ‘客户表’;

注：这是比较通用的方法。

1. 导出数据字典

SELECT

a.table\_name,b.TABLE\_COMMENT,a.COLUMN\_NAME,

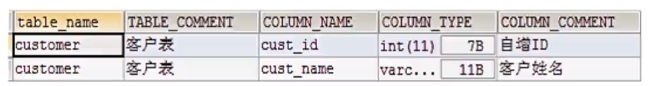
a.COLUMN\_TYPE,a.COLUMN\_COMMENT

FROM information\_schema.COLUMNS a

JOIN information\_schema.TABLES b

ON a.table\_schema=b.table\_schema AND a.table\_name=b.table\_name

WHERE a.table\_name=’customer’；



## 维护索引

如何选择合适的列建立索引？

1. 出现在WHERE从句，GROUP BY从句，ORDER BY从句中的列
2. 可选择性高的列要放在索引的前面
3. 索引中不要包括太长的数据类型

注意：

1. 索引并不是越多越好，过多索引不但会降低写效率而且会降低读的效率
2. 定期维护索引碎片
3. 在SQL语句中不要使用强制索引关键字

## 维护表结构

1. 使用在线变更表结构的工具

MySQL5.5之前可以使用pt-online-schema-change

MySQL5.6之后本身支持在线表结构的变更

1. 同时对数据字典进行维护
2. 控制表的宽度和大小

## 适合的操作

1. 批量操作 VS 逐条操作

注：数据库适合批量操作，而逐条操作适用于存储过程。

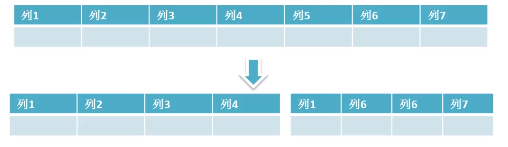
1. 禁止使用select \*这样的查询
2. 控制使用用户自定义函数

注：函数会对索引有影响，使用索引时该列的索引失效。

1. 不要使用数据库中的全文索引

## 垂直和水平拆分

为了控制表的宽度可以进行表的垂直拆分：



注：

1. 经常一起查询的列放在一起
2. Text，blob等大字段拆分出到附加表中

为了控制表的大小可以进行表的水平拆分：

