**衣淘淘**

**数据库设计**

**陈开一**

二○一九年一月

# 一.命名规范

## 1.1 表名

表名统一小写，采用字母和下划线组成，不同含义单词采用下划线分割。

例：

账户表account；

用户清单表users\_list；

衣物评论clothes\_comment等。

## 1.2 属性名

属性名采用java变量命名规范，首词的首字母小写，从第二个单词含义开始首字母大写。

例：

用户表的userID；

用户注册时间registerDate；

用户最后登录时间lastLoginDate等。

## 1.3 触发器和约束

触发器命名规范类似表名，采用字母和下划线组成，统一小写。格式为表名+下划线+触发器标志trg+触发器类型和操作类型（由于存储过程太费时，因此废弃）。

例：

用户表主键自增触发器：user\_trg\_aftinsert;

用户表修改信息触发器：user\_trg\_aftupdate;

插入检测触发器：<tablename>\_trg\_befinsert；等

约束命名与触发器类似，采用字母和下划线组成，统一小写。格式为表名+属性名+类型。

例：

用户昵称唯一性约束：user\_Name\_unique；

用户性别检查约束：user\_sex\_check；等

## 1.4 视图

视图命名规范类似表名，采用字母和下划线组成，统一小写。格式为逻辑表名+下划线+view。

例：

用户租赁记录视图：user\_order\_view;

商家商品库存视图：business\_store\_view;等

## 1.5 索引

索引能够提高查询效率，因此比较重要。索引命名由表名，和属性名组成，中间采用下划线分开（MySql会自动为主键建立索引）。

例：

商品价格索引：clothes\_set\_price;

# 二.数据库优化（1.0）

## 2.1 建立索引

为了避免全表扫描，最好为可能大量用到where以及order by的列建立索引，从而提高查找效率。例如用户需要经常搜索商品租赁价格，因此商品的price属性会经常被访问，所以最好为price添加索引。语句如下：

CREATE INDEX clothes\_set\_price ON clothes\_set(price ASC/DES);

但是索引并不是越多越好，应当考虑最小化索引字段，因为每次UPDATE或者INSERT都会重建索引，引起UPDATE和INSERT语句低效率。一般情况下，一个表的索引最多不能超过5-6个。

## 2.2 避免”NULL”值

避免对表中的NULL值进行WHERE语句判断，否则MySql搜索引擎会对全表进行扫描。例如:

SELECT clothesID WHERE price IS NULL;

这样即使我们创建了index，搜索引起也会放弃，从而进行全表查询。当数据量达到百万、千万级别的时候将会非常耗时。而避免NULL值的方法有多种：

①尽量对可能多的属性设置”NOT NULL”约束避免NULL值。（描述，评论之类的除外）；

②能够使用INT类型数据的尽量使用INT，因为INT类型可以用0代替VARCHAR的NULL。

## 2.3 避免！=和<>

与2.2相同，如果对WHERE语句使用!=或者<>的话，搜索引擎会自动放弃已经建立好的（如果有）的索引，进行全表查询。

## 2.4 采用UNION代替OR

我们WHERE语句中使用了OR，当OR两段的属性一个有索引而另一个无索引的情况下，搜索引擎会自动放弃已经建立好的字段的索引进行全表查询，例：

SELECT clothesID FROM clothes\_set WHERE price = 100 OR address = WHU；

上面的查询语句中，显然price应该建立索引而address是没必要建立索引的，但是MySql会对全表进行扫面，采用UNION替换成如下:

SELECT clothesID FROM clothes\_set WHERE price = 100

UNION ALL

SELECT clothesID FROM clothes\_set WHERE address = WHU;

## 2.5 采用BETWEEN替换IN和NOT IN

WHERE语句中的IN和NOT IN同样会覆盖索引引起全表扫描。比如：

SELECT lastLoginDate FROM users WHERE userID IN(1,2,3)；

类似上面这种连续数值的语句，尽量使用BETWEEN来查找，BETWEEN语句首先会去查找是否有索引，如果没有则全表扫描:

SELECT lastLoginDate FROM users WHERE userID BETWEEN 1 AND 3;

## 2.6 强制索引

动态查询的时候，如果WHERE语句中后面出现参数，MySql只有在运行的时候才会解析局部变量， 但是默认搜索引擎是需要在编译阶段就已经确定，不能推迟。例如PreparedStatement中，我们定义如下查询语句：

SELECT password FROM accout WHERE username = ?；

后面的？只有在运行的时候通过setString方法才能获取参数，即使我们建立了索引，还是没用。但是我们可以强制其使用索引index，如下:

SELECT password FROM accout WITH(INDEX(name)) WHERE username = ?；

其中name是username字段的索引名称，详见命名规范1.5。此处直接体现了2.1优化的重要性。

## 2.7 避免表达式

WHERE语句中尽量避免表达式，因为出现表达式搜索引擎在编译的时候无法去查找索引从而引起全表查询，通过事先计算表达式来消除。

## 2.8 避免SELECT… INTO语句

SELECT…INTO语句不会产生任何结果集，反而会频繁消耗时间，特别是创建临时表的时候，比如：

SELECT username, clothesID INTO tem\_shopping\_cart WHERE…；

应该为CREATE TABLE tem\_shopping\_car(…)；

## 2.9 UPDATE局部

需要更新信息的时候，UPDATE语句更新的字段越少越好。否则会带来极大内存消耗，产生大量数据库日志。

## 2.10 跨表多时采用分页连接

当需要跨多个表，并且表的数据量超过100条时，尽量避免直接进行外连接。采用分页方法替代，分页方法即SELECT嵌套。即：

SELECT userID FROM(SELECT(…))…;

## 2.11 数据类型

MySql4.1以后支持变长类型VARCHAR，能使用VARCHAR就使用VARCHAR替代CHAR，因为CHAR固定长度，不管该字段的值达没达到最大长度，都会占用指定的空间，但是变长VARCHAR就是根据具体的内容分配，极大节省了空间。

能用INT、FLOAT或者DOUBLE数值类型的，就用数值类型，因为数值类型存储方式简单。而字符型对判等敏感，从而直接降低WHERE，JOIN等语句性能，另一方面同样内容，字符型的存储空间比数值型大很多。

## 2.12 避免游标

游标会带来一定安全隐患，并且很容易出错，游标的效率也很低。如果游标指向的数值超过了一万行，就应该改写。

## 2.13 事务处理

当完成一个事务需要多个步骤的时候，不一定每个步骤都会成功执行，特别是当用户众多，并发量大的时候，如果处理不当轻则产生死锁，重则损坏数据。因此要么所有的步骤全部成功，但如果一个失败，则全部放弃。采用BEGIN…COMMIT语句进行事务处理：

BEGIN

transaction start

COMMIT;

## 2.14 锁定表维护数据

事务是维护数据库完整性的一个非常好的方法，但却因为它的独占性，有时会影响数据库的性能，尤其是在很大的应用系统中。由于在事务执行的过程中，数据库将会被锁定，因此其它的用户请求只能暂时等待直到该事务结束。如果一个数据库系统只有少数几个用户来使用，事务造成的影响不会成为一个太大的问题；但假设有成千上万的用户同时访问一个数据库系统，就会带来比较严重的延迟。有些情况下我们可以通过锁定表的方法来获得更好的性能。例如:

LOCK TABLE clothes\_set WRITE INSERT < clothes > INTO clothes\_set;

UPDATE clothes\_set SET stroe=11…UNLOCKTABLES;

在UNLOCKTABLES执行前，其他用户无法访问该表。

# 三.数据库实体（1.0）

## 3.1 用户列表users\_list

users\_list

(

userID(PK),

username(FK) REFERENCES users(username)

)

## 3.2 用户表users

users

(

username(PK),

vipID(FK) REFERENCES vip(vipID)

sex,

age,

realName,

identity,

phone,

email,

registerDate,

lastLoginDate,

lastLoginDevice,

lastLoginLocation,

attentionBusiness

latelyTrack

)

## 3.3 账号信息表account

account

(

username(PK),

type, //common users or DBM

password //extremely important

)

## 3.4会员表vip

vip

(

vipID(PK),

vipStartTime,

vipEndTime,

vipRestTime,

vipScore,

vipDiscount,

vipLevel,

auto Renew,

)

## 3.5 商家列表business\_list

商家需要手动联系我们注册。

business\_list

(

businessID(PK),

username(FK) REFERENCES business(username)

)

## 3.6 商家business

business

(

username(PK), //社会统一信用代码

identity,

email,

phone

registerDate,

lastLoginDate,

lastLoginLocation,

address

)

## 3.7 商品goods

goods

(

goodsID(PK),

businessID (FK), REFERENCES business\_list(businessID),

name,

sex,

brand,

code,

style,

thick,

scenarios

)

## 3.8 服装clothes

clothes\_set

(

clothesID(PK),

goodsID(FK) REFERENCES goods(goodsID),

color,

size,

stock,

price,

deposit

)

## 3.9 订单order

order

(

orderID(PK),

clothesID(FK),REFERENCES clothes(clothesID),

userID(FK), REFERENCES users\_list(userID),

businessID (FK), REFERENCES business\_list(businessID),

number,

unitPrice,

totalPrice,

rentalDate,

returnDate,

state

)

## 3.10 订单评论order\_comment

order\_comment

(

commentID(PK),

orderID(FK), REFERENCES order(orderID),

commentDate,

commentContent

)

## 3.11 店铺评论business\_comment

business\_comment

(

commentID(PK),

userID(FK), REFERENCES users\_list(userID),

commentDate,

commentContent

)

## 3.12 可选临时表购物车shopping\_cart

shopping\_cart

(

cartID(PK),

userID(FK), REFERENCES users\_list(userID),

clothesID(FK), REFERENCES clothes(clothesID),

number,

totalPrice,

)

# 四.修正

暂无