**内部资料 注意保密**

**Mysql数据库设计规范**

**V1.0.0**

2016.9.18

**应用研发二部**



**卓望数码技术（深圳）有限公司 2000,2016**

**All Rights Reserved**

版本历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 作者 | 参与者 | 起止日期 | 备注 |
| 1.0.0 | 徐勇 |  | 2016/9/18 | 起草 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[前言 3](#_Toc462048833)

[第一章 命名约定 3](#_Toc462048834)

[1.1 命名规则 3](#_Toc462048835)

[1.2 存储引擎 4](#_Toc462048836)

[1.3 字符集 4](#_Toc462048837)

[第二章 设计约定 4](#_Toc462048838)

[2.1 表设计约定 4](#_Toc462048839)

[2.2 索引设计约定 6](#_Toc462048840)

[第三章 Sql约定 7](#_Toc462048841)

[3.1 基本规则 7](#_Toc462048842)

[3.2 建议事项 8](#_Toc462048843)

[第四章 性能优化 8](#_Toc462048844)

[4.1 基本规则 8](#_Toc462048845)

[4.2 其他规则 10](#_Toc462048846)

# 前言

本文将Mysql数据库设计与开发时在其命名规范、注解、性能调优等方面进行统一的规范和要求，将在web程序、业务程序、通讯接口等程序中出现的Mysql对象进行统一规范，以便于开发阶段的代码编写及日后维护工作，增强可读性。

内容主要包括：

* Mysql对象命名规则；
* Mysql设计时约定与规范；
* Mysql的sql编写规范；
* Mysql的性能优化。

# 命名约定

所有对象名应该遵循以下原则：可读性原则、表意性原则

## 命名规则

采用26 个英文字母(区分大小写)和0-9 这十个自然数,加上下划线'\_'组成,共63 个字符.不能出现其他字符(注释除外).

注意事项:

1) 以上命名都不得超过30 个字符的系统限制.变量名的长度限制为29(不

包括标识字符@).

2) 数据对象、变量的命名都采用英文字符,禁止使用中文命名.绝对不要在对

象名的字符之间留空格.

3) 小心保留词,要保证你的字段名没有和保留词、数据库系统或者常用访问

方法冲突

5) 保持字段名和类型的一致性,在命名字段并为其指定数据类型的时候一

定要保证一致性.假如数据类型在一个表里是整数,那在另一个表里可就别变成字

符型了.

## 存储引擎

默认Innodb

## 字符集

Database Server字符集统一默认UTF-8，table和column从server继承，如果有特殊字符或者生僻字，需要在创建数据库申请时特别说明字符集设置为UTF-8的超集utf-8mb4.

# 设计约定

## 表设计约定

1. 主键

主键命名规则为 pk\_[TableName], 主键归属确定的表，其意义与表名相关。

如主键字段不能满足业务需求，另建unique约束业务字段。

2）外键

尽量少用到主外键关联

3）表名、字段命名规则

* Table:同一模块使用相同的表前缀
* Column：XXX\_XXX，中间以下划线隔开
* 表名、字段名都用小写

1. 字段冗余

非严格遵守3NF，通过业务字段冗余来减少表关联

5）字段类型长度选择

* 主键字段：

gint(20),int(11) 根据预估数据量选择（大表id列考虑unsigned）

* 字符串：

a) varchar:存放可变长度字段，由1-2个字节记录，字符串长度小于等于255，只需要一个字节记录

b) char:存放定长字段，对于经常修改的字段比较有优势，存放非常短的列，例如:’Y’,’N’性能较好

c) 尽可能使用定长char类型，如姓名，身份证号码，如需要变长varchar，尽可能根据实际情况限制长度，如description等

* 整型：

a) 枚举类型建议使用tinyint(4)

b) tinyint(4),smallint(6),int(11),bigint(20)

c) （）内的为默认值，不建议修改

* 日期：

a) datetime 占8个字节，支持 '1000-01-01 00:00:00' 到 '9999-12-31 23:59:59'，默认值非current\_timestamp()下，建议设置为'1970-01-01 08:00:01'。

b) timestamp占4个字节，支持'1970-01-01 00:00:01'到 '2038-01-19 03:14:07'UTC，建表时该字段不加任何属性时默认加非空约束，default值为current\_timestamp，默认值非current\_timestamp()下，建议设置成'1970-01-01 08:00:01'。

c) date类型,默认值非current\_date()下，建议默认值为'1970-01-01’。

d) 一张表只能有一个timestamp列用于标识自动更新，长度不允许自定义，非自动更新的列建议统一使用datetime。

* 大字段(text/blob)

原则上不允许这种字段，尽可能的拆分成小字段，如果特别需要，而又读写频繁，另外建一张表

6）索引和查询字段不允许为空

所有字段尽可能使用not null：

* 整型默认值可为0
* 字符串默认值可为''
* 时间默认值可为current\_timestamp或 '1970-01-01 08:00:01'
* date类型默认值可为 current\_date()或者'1970-01-01'。

7）注释

建表包含表注释和字段注释，尤其字段是枚举类型需要说明每一种含义

## 索引设计约定

1）索引命名约定

* 主键：pk\_columnName (或者让数据库自动命名);
* 唯一键：uk\_columnName;
* 普通索引：ix\_columnName;
* 组合索引：ix\_column1\_column2\_Column3;

2）聚簇索引约定

自增长，auto\_increment

3）非聚集索引(二级索引)约定

* 查询条件经常用到a,b,c列，建立复合索引（a,b,c）
* 查询条件经常使用到b,c列，建立复合索引（b,c）
* 查询条件还要使用到c列，单独在c上建立索引
* 以此类推
* 对超过20的不定长字符串，建议创建前缀索引add index(XXXX(20))
* 不建议在每个字段上都加索引

4）索引原则

主键选取：应选择唯一、长度小、查询多的字段做为主键，优先选择数值型字段（或者自增字段）；

关联字段：有关联关系的字段，需要在关联字段上建索引，并确保类型一致；

筛选字段：经常作为查询条件的筛选字段（where条件），一般需要考虑建索引；

排序分组字段：查询时经常作为排序（order by）或者分组（group by）的字段，应考虑创建索引；

组合索引：频繁的查询进程使用某些固定的列，可以考虑为这些列创建一个组合索引（组合列不要超过三个）；

筛选条件较好的字段（如：日期）应放在组合索引的最前面；

索引并非越多越好，它会带来数据更新性能消耗和磁盘空间占用，原则上建议一个表的索引数量不要超过八个；

筛选力度小的列，不建议创建索引（如：状态status字段，性别字段等）。

组合索引，如果包含范围查询的和等值查询的情况，可根据值的可选择性高的等值查询的条件放前面，可选择性差的放第二以此类推，最后放范围查询条件的列。

# Sql约定

## 基本规则

* 不要使用"SELECT \*"，必须使用列的列表；
* 不要使用全文检索（Full Text Search）；
* 尽量少的用到游标、触发器、存储过程等；
* 查询时尽量不要有太多的表关联；
* 不要在没有匹配索引的表上进行for update这类的操作，会锁定整个表；
* 未经过DBA同意，禁止在程序端大批量更新或者删除数据，因为这样操作很可能造成复制的大量阻塞和延时；
* 一条SQL语句中包含多个对象时，引用对象的列必须用"对象名.列名"或"对象别名.列名"的方式；
* 新建表必须有主键；
* 脚本中禁止出现 drop 和 truncate 语句，如果需要删除表，文字说明就可以;
* 禁止在已有数据的表上删除列，这个操作无法回滚：alter table xxx drop column xxx；
* 为方便DA处理元数据，每个列必须添加说明；
* 不要大批量的查询数据结果，如果需要返回大量数据，请使用分页的方式处理；、

## 建议事项

* 原则上不建议使用BLOB和TEXT类型的字段，如要使用，请将这两种类型的字段单独放到关联表中做处理；
* 建议将复杂的计算和逻辑操作放到程序端处理，而不是使用SQL语句处理，因为程序端方便扩容，数据库端扩容能力有限；
* 数据结构设计时，可以适当设计冗余字段，减少查询的复杂度，提高查询的性能；
* 预期产生大数据量的表，提前考虑分表分库的方案；
* 多行数据需要做处理时，建议批量处理，而不是一条条来处理；
* 建议在每条查询语句后面加上Limit关键字，控制返回的数据量，防止不可控的返回大量的数据；
* 规划时，表数据量不要超过千万级别，表大小不要超过50G，否则应提前考虑策略（可以和DBA协商方案，使用历史表、表数据拆分等）

# 性能优化

## 基本规则

1. 尽量避免相同语句由于书写格式的不同，而导致多次语法分析；
2. 尽量使用共享的SQL语句，也就是说，在SQL中尽量采用绑定变量的方式，而不是常量；
3. 尽量不使用“SELECT \*”这样的语句，即使需要查询表中的所有行，也需列出所有的字段名；
4. 尽量避免4个以上表的链表操作，例如：a = b AND b = c AND c = d，如果业务上需要，可以考虑通过中间表的方式进行变通；
5. 大量的排序操作影响系统性能，所以尽量减少ORDER BY和GROUP BY排序操作。如必须使用排序操作，请遵循如下规则：
6. 排序尽量建立在有索引的列上。
7. 如结果集不需唯一，使用UNION ALL代替UNION。
8. 系统可能选择基于规则的优化器，所以将结果集返回数据量小的表作为驱动表（from后边最后一个表）。

驱动表的选择和很多的因素有关系，不仅仅是表的顺序，这点仅做参考，不过养成这个习惯有助于以后进行SQL的优化。

1. 索引的使用
2. 尽量避免对索引列进行计算。
3. 尽量注意比较值与索引列数据类型的一致性，避免使用数据库的类型自动转换功能。
4. 对于复合索引，SQL语句必须使用主索引列。
5. 索引中，尽量避免使用NULL。
6. 对于索引的比较，尽量避免使用NOT=（!=）。
7. 查询列和排序列与索引列次序保持一致。
8. 查询的WHERE过滤原则，应使过滤记录数最多的条件放在最前面。
9. 使用%TYPE、%ROWTYPE方式声明变量，使变量声明的类型与表中的保持同步
10. 在IF/ELSE查询中，使用DECODE
11. 在 SQL 中使用 WHERE 子句过滤数据，而不是在程序中到处使用它进行过滤
12. 执行动态SQL，建议用EXECUTE IMMEDIATE SQL子句；
13. 尽量避免使用UNION，若需要排重，建议使用FROM子句把查询结果UNION ALL起来后，再通过GROUP BY排重，如：

SELECT id

FROM (SELECT id

FROM a

UNION ALL

SELECT id FROM b)

GROUP BY id;

1. 尽量使用外关联代替NOT IN，NOT EXISTS操作。如：

SELECT id, name

FROM a

WHERE id

NOT IN ( SELECT id FROM b); --此写法不建议使用

SELECT a.id, a.name

FROM a, b

WHERE a.id = b.id(+)

AND b.id IS NULL; --建议使用此写法

## 其他规则

* 代码中需要空位时，统一采用空格键输入，不允许用TAB键产生空位，因为不同的编辑器对TAB的空位格数设置不一致，会导致使用TAB键产生空位的代码格式混乱；
* 当一个SQL 语句中涉及到多个表时，始终使用别名来限定字段名，这使其它人阅读起来更方便，避免了含议模糊的引用，其中能够别名中清晰地判断出表名。
* 当在SQL语句中连接多个表时, 请使用表的别名并把别名前缀于每个COLUMN上。这样一来,就可以减少解析的时间并减少那些由COLUMN歧义引起的语法错误；
* 需要UPDATE的字段，不应设计为分区条件字段；
* 频繁更新的表上，不得建立refresh fast on commit类型的物化视图；
* 建立sequence，必须要设置cache，建议不小于100；
* 视图中不应该出现ORDER BY排序；视图的基础数据尽量从表中获取，尽量不要嵌套视图；
* 尽量使用DECODE来简化SQL访问数据库的次数；
* 减少控制语句的判断次数，比如在ELSE（IF…ELSE）语句中，尽量将尽快能检测到结果的判断提前；
* 存储过程，必须有异常捕获代码；
* 存储过程中严禁使用GOTO语句进行跳转；
* 有循环更新的存储过程，必须进行批量提交，且必须进行事物控制；
* 存储过程中如果有更新，必须在异常捕获代码中做回退操作；
* 如无必要，不要在数据库底层使用触发器，尽量在应用层控制；
* 在可以进行批处理的地方，尽量采用批处理的方式，而不是所有的地方都采用游标的方式，采用游标的方式确实可以完成一些大数据量的工作，但对数据库的资源消耗比批处理要大；
* 一个语句中含有多个UNION ALL，并且执行时间过长的情况下，可以考虑中间表的方式进行过渡；
* 对分区表的操作，最好能定位到其中的某个或某几个分区上；
* 在OLTP的处理中，SQL影响的行数一般都很少，对表一般对需进行索引的range或unique扫描，如果发现对表进行全表扫描或对索引进行了全索引扫描（index full scan），这种执行计划一般都需要调整，需联系熟悉SQL优化的人员进行调整。