蘑菇租房MYSQL数据库开发规范

目录

[一、数据库版本 2](#_Toc495418741)

[二、硬件环境 3](#_Toc495418742)

[三、设计规范 3](#_Toc495418743)

[1.字段设计 3](#_Toc495418744)

[2.表设计 6](#_Toc495418745)

[四、枚举规范【参考】 8](#_Toc495418746)

[五、SQL脚本提交规范 8](#_Toc495418747)

[六、索引规范 11](#_Toc495418748)

[七、SQL规范 13](#_Toc495418749)

[1.query 13](#_Toc495418750)

[2.insert 14](#_Toc495418751)

[3.update 14](#_Toc495418752)

[4.delete 14](#_Toc495418753)

[5.replace 14](#_Toc495418754)

[6.存储过程 15](#_Toc495418755)

[7.触发器 15](#_Toc495418756)

[8.事件 15](#_Toc495418757)

[9.视图 15](#_Toc495418758)

[八、慢查询规范 15](#_Toc495418759)

[1.慢查询标准【推荐】 15](#_Toc495418760)

[2.慢查询巡检机制【推荐】 15](#_Toc495418761)

[九、安全 16](#_Toc495418762)

[十、数据行为规范 16](#_Toc495418763)

[十一、每个规范的监察机制及奖惩制度 17](#_Toc495418764)

[1.以上规范适用于所有历史SQL 17](#_Toc495418765)

[2.设计规范 17](#_Toc495418766)

[3.枚举规范 17](#_Toc495418767)

[4.SQL脚本提交规范 17](#_Toc495418768)

[5.索引规范 17](#_Toc495418769)

[6.SQL规范 18](#_Toc495418770)

[7.慢查询规范 18](#_Toc495418771)

[8.安全规范 18](#_Toc495418772)

本规范版本：Version-1.2.0

本规范编写负责人：张思杰（清风）

编写时间：2016年8月25日

修改历史

|  |  |  |
| --- | --- | --- |
| 版本编号 | 变更时间 | 修改人 |
| Version-1.0.1 | 2016.08.25 | 清风 |
| Version-1.0.2 | 2016.08.31 | 清风 |
| Version-1.0.3 | 2016.09.01 | 清风 |
| Version-1.0.4 | 2016.09.05 | 清风 |
| Version-1.1.0 | 2016.09.08 | 清风 |
| Version-1.1.1 | 2016.11.24 | 清风 |
| Version-1.1.2 | 2017.02.09 | 清风 |
| Version-1.1.3 | 2017.02.23 | 清风 |
| Version-1.1.4 | 2017.03.14 | 清风 |
| Version-1.1.5 | 2017.03.24 | 清风 |
| Version-1.1.6 | 2017.03.28 | 清风 |
| Version-1.1.7 | 2017.07.06 | 清风 |
| Version-1.1.8 | 2017.08.15 | 清风 |
| Version-1.1.9 | 2017.09.18 | 清风 |
| Version-1.2.0 | 2017.10.10 | 清风 |

# 一、数据库版本

MYSQL 5.6.27

Mariadb10.1.14,10.1.21

GaleraMariadb 10.0.24

# 二、硬件环境

阿里云主机ECS

|  |  |  |  |
| --- | --- | --- | --- |
| CPU | 内存 | 磁盘 | 备注 |
| 32核 | 64G | 1000G | 主SERVER都是SSD |

# 三、设计规范

约定：

【**强制**】代表无论如何必须遵循

【**推荐**】并不强制遵循，但需优先考虑

【**参考**】作为参考，一些补充说明

## 1.字段设计

* + 1. 【**强制**】每个单词不超过6个字母。使用驼峰命名法，如userId、createTime。
    2. 【**强制**】所有表至少需要有这些字段，例如：

`id` BIGINT(20)UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT '主键'

`createTime` DATETIME NOT NULL DEFAULTCURRENT\_TIMESTAMP COMMENT '创建日期'

`updateTime` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP COMMENT '更改日期'

`valid` TINYINT(4) NOT NULL default '1' COMMENT '用于标示该记录是否逻辑删除，0:已删除 1:正常有效'

**说明**：ID主键、createTime记录创建时间、updateTime记录变更时间、valid用于标识是否为逻辑删除记录，默认NOT NULL default ‘1’

* + 1. 【**推荐**】业务操作表至少需要有这些字段，例如：

`id`BIGINT(20)UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT ‘主键’

`status` TINYINT (4) NOT NULL COMMENT ’状态用于标示业务状态’

`valid` TINYINT(4) NOT NULLDEFAULT‘1’ COMMENT ’用于标示该记录是否逻辑删除，0:已删除 1:正常有效’

`createTime` DATETIME NOT NULL DEFAULTCURRENT\_TIMESTAMP COMMENT ’创建日期’

`createBy` INT(11) NOT NULL COMMENT ’创建人’

`createByType` INT(11) NOT NULL COMMENT ‘创建人类型(参考字典表组名:userType)’

`updateTime` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP on update CURRENT\_TIMESTAMP COMMENT ‘修改时间’

`updateBy` INT(11) NOT NULL COMMENT ‘修改人’

`updateByType` INT(11) NOT NULL COMMENT ‘修改人类型(参考字典表组名:userType)’

`soDoneCode` INT(11) NOT NULL COMMENT ‘操作流水号’

`remark` VARCHAR(255) NULL COMMENT ‘备注’

* + 1. 【**参考**】字段规范说明

| 字段名 | 字段类型 | 字段其他属性 | 字段说明 | |
| --- | --- | --- | --- | --- |
| createBy | INT(11) | NOT NULL | 创建人 | |
| createTime | DATETIME | NOT NULL,current\_timestamp, | 创建时间 | |
| soDoneCode | INT(11) | NOT NULL, | 操作流水号 | |
| remark | VARCHAR(255) | NULL, | 注释 | |
| status | TINYINT (4) | NOT NULL, | 状态 | |
| updateBy | INT(11) | NOT NULL, | 更新人 | |
| updateTime | DATETIME | NOT NULL, current\_timestamp on update | 更新时间 |

* + 1. 【**强制**】字段的编码如无特别需求不指定字符集，字符集统一在表级别定义
    2. 【**强制**】大字段规范：

禁止BLOB、TEXT类型

**说明**：BLOB与TEXT在MYSQL中优化不如VARCHAR；BLOB无法使用索引，TEXT需要使用全文索引但MYSQL无法很好的支持全文索引

字段为VARCHAR并超过5000长度，如果该字段很少被访问到，且其他字段又是读密集性访问，必须将该字段拆分成单独表存放。

**说明**：VARCHAR为变长字符串，不预先分配磁盘存储空间，但读到内存中分配的却是固定的定义长度；其次MYSQL一个数据页的大小为16K，如果一行数据能占用1/3页，会不同程度的影响性能也浪费空间

* + 1. 【**强制**】VARCHAR超过512长度必须经过DBA评估
    2. 【**强制**】精确到秒的时间采用DATETIME

**说明**：5.6之后DATETIME与TIMESTAMP性能接近，但前者要比后者存储的范围更大。

精确到年使用YEAR

精确到日期使用DATE

禁止用VARCHAR类型存储时间

**说明**：浪费空间，且性能下降

禁止使用INT类型存储时间

**说明**：虽然INT可以减少存储空间，但需要转换

* + 1. 【**推荐**】避免使用NULL字段

**说明**：NULL字段占额外空间，NULL值不走任何索引

* + 1. 【**强制**】字段其他属性值为NOT NULL时须加DEFAULT默认值
    2. 【**强制**】禁止使用ENUM枚举类型，使用TINYINT代替，添加新的ENUM值需要DDL锁表操作
    3. 【**强制**】存储精确浮点数必须使用DECIMAL来代替FLOAT、DOUBLE

金额格式标准：DECIMAL(10,2)。

**说明**：FLOAT表示单精度浮点，DOUBLE表示双精度浮点，以上两种都存在精度损失现象，很可能值在比较时得不到正确的结果，而DECIMAL是定点类型不存在上述问题

* + 1. 【**参考**】INT整数遵守适用原则

整数字段名称区分：TINYINT< SMALLINT < MEDIUMINT < INT < BIGINT

占用空间：TINYINT:1字节 SMALLINT:2字节MEDIUINT:3字节INT:4字节BIGINT:8字节

长度(UNSIGNED)：TINYINT:0-255 SMALLINT:0-65535 MEDIUINT:0-16777215 INT:0-4294967295 BIGINT:0-18446744073709551615

* + 1. 【**推荐**】字段允许适当冗余，但必须考虑数据一致性。

**说明**：提高查询性能，避免不必要的关联查询

冗余字段应遵循：

1. 不是频繁修改的字段
2. 不是VARCHAR或TEXT超长字段
   * 1. 【**强制**】禁用保留字，如desc、range、match、select、delayed等，请参考MySQL官方保留字<https://dev.mysql.com/doc/refman/5.7/en/keywords.html>

## 2.表设计

* + 1. 【**强制**】建新表前须预估表容量、读取量、主要查询语句（三原则），事先分析可否建立冗余字段从而有效减少关联量。
    2. 【**强制**】表名以模块名（4位字母）加下划线开始，后续单词不适用下划线。每个单词不超过4个字母。如acct\_busirec。
       - 1. 对于房态等表，需要分历史表和当前表，历史表在尾部添加\_his。比如flat\_room的历史数据放入flat\_room\_his中，而当前的数据放在当前表中。注意是当前数据，而不是当前有效数据，比如某房源正在装修，这对租房而言是无效的，但是也要放在当前表中。
         2. 表名需根据服务模块命名与区分，比如：acct\_账务类，user\_用户表，bill\_账单类
    3. 【**强制**】每张表新建必须有自增长主键
       1. CREATE TABLE `table\_create\_template` (
       2. `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'ID',
       3. PRIMARY KEY(`id`)
       4. ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='建表样式表';
    4. 【**强制**】\_his后缀的表必须遵循如下要求：
       - 1. 此表只允许insert
         2. 此表禁止select，update
         3. 此表必须有lastUpdateTime字段

**说明**：DBA需要根据lastUpdateTime字段对历史数据进行归档维护

lastUpdateTime DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP on update CURRENT\_TIMESTAMP COMMENT '最后更新时间'

* + 1. 【**强制**】建表默认编码：utf8mb4

**说明**：编码只在表级别指定，禁止在字段中指定编码，极个别特殊情况请与DBA联系，由DBA负责评估

* + 1. 【**强制**】所有业务表全部使用innodb存储引擎

**说明**：我们的数据库采用的是集群模式，目前该模式下只兼容innodb存储引擎

* + 1. 【**推荐**】单表不得超过1000万行，单表容量不得高于20GB，行平均长度不得超过16K。
    2. 【**强制**】单库中的表量不得超过500个
    3. 【**强制**】单表字段数不得超过50个
    4. 【**强制**】表记录为日志类型，并且表容量预估超过500万，表必须采用分区，分区命名方式如下：
       - 1. 分区键为时间字段
         2. 分区键不能更新
         3. 查询条件需带上分区键，比如createtime>= '2017-06-01 00:00:00' and createtime< '2017-07-01'
         4. 分区命名方式为P\_20170101
         5. 每月一个分区

CREATE TABLE `t\_part` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT '主键',

...

`createTime` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',

PRIMARY KEY (`id`,createTime)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='分区表示例'

PARTITION BY range COLUMNS(createTime)

(

PARTITION P\_20170701 VALUES LESS THAN ('2017-07-01'),

PARTITION P\_20170801 VALUES LESS THAN ('2017-08-01'),

PARTITION P\_20170901 VALUES LESS THAN ('2017-09-01'),

PARTITION P\_20171001 VALUES LESS THAN ('2017-10-01'),

PARTITION P\_20171101 VALUES LESS THAN ('2017-11-01'),

PARTITION P\_20171201 VALUES LESS THAN ('2017-12-01'),

PARTITION P\_20180101 VALUES LESS THAN ('2018-01-01')

);

**说明**：分区采用物理文件分割方式存放于存储媒介上，每一个分片都是一个单独的物理文件，必须避开跨物理文件查询的情况，跨的物理文件越多性能越差，同时也背离了分区的意义

# 四、枚举注释规范【参考】

|  |  |  |
| --- | --- | --- |
| 表名 | 枚举字段名 | 枚举注释COMMENT |
| cntr\_contract\_clause | createByType | 创建人类型(参考字典表组名:userType) |
| bill\_cbfoperlog | fundChannel | 支付提供商类型(3001:微信 2001:支付宝) |

规定使用码表存放枚举值，新枚举值必须存于码表中

# 五、SQL脚本提交规范

* + 1. 【**强制**】表和字段必须写明注释，并且默认引擎ENGINE=InnoDB,默认编码DEFAULT CHARSET=utf8mb4，例如：
       1. CREATE TABLE `table\_create\_template` (
       2. `id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT 'ID',
       3. `comment` varchar(11) NOT NULL COMMENT '请别忘记注释我',
       4. PRIMARY KEY (`id`)
       5. ) **ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='建表样式表';**
    2. 【**强制**】提交sql需要在"/\* \*/"中写明提交原因，并且连接的数据库名称使用'use mogoroomdb;'

/\* Database name `mogoroomdb` 及SQL提交原因\*/

usemogoroomdb;

**此处是你的SQL**

* + 1. 【**推荐**】DML、DDL补充的脚本需要新建同名脚本带序列后缀，测试人员可以及时、有效补充执行SQL变更部分内容，例如：

JG03.BS\_app菜单树配置 #如果该脚本有新增，或修改，请新建如下SQL脚本

JG03.BS\_app菜单树配置**\_2** #请注意，此时后缀序号为”**\_2**”

* + 1. 【**推荐**】必须写明执行顺序，在最开始前缀部分标明，如上例中”JG03.”前缀部分，说明本批的极光组脚本中需第三顺序执行
    2. 【**强制**】禁止使用set autocommit = 0开始事务，统一使用begin/commit开始关闭事务

**说明**：MYSQL默认情况下是开启自动提交事务的，如果使用set autocommit =0后会话会永久关闭自动提交功能，即使后面commit了也不再开启自动提交。

* + 1. 【**强制**】提交的SQL脚本文件必须是UTF-8编码，否则会造成SQL内容乱码
    2. 【**强制**】创建新表时，必须根据表设计要求预估表容量、读取量、主要查询语句（三原则），并将三原则以注释形式写入脚本中，否则不予审核通过

例如：

/\*创建flat\_room表

1. 预估三个月表容量将达到50W条记录
2. 每次读取量1000条以内。
3. 主要查询SQL：

SELECT id,flatsid,roomPrice FROM flat\_room where renterId = 1000001;

SELECT fr.id,fr.flatsid,ff.communityId FROM flat\_roomfr left join flat\_flatsff on fr.flatsid=ff.id where fr.renterId=1000001;

需要预先添加的索引字段flatsid,roomNum

\*/

CREATE TABLE flat\_room(

`id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT 'ID',

`flatsId` int(11) NOT NULL DEFAULT '0' COMMENT '公寓ID',

`styleCode` varchar(16) DEFAULT NULL COMMENT '风格引用-字典表(style)',

……

PRIMARY KEY (`id`),

UNIQUE KEY `roomNum` (`roomNum`),

KEY `IDX\_FLATS` (`flatsId`)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='房间信息'

* + 1. 【**强制**】表结构变更，当表文件大小超过5G或者记录数超过1千万，需经过架构总监或一级主管审批，执行时间放在晚上业务低峰期；
    2. DDL脚本提交相关
       1. 【**强制**】所有DDL类SQL脚本必须放在程序发布前执行
       2. 【**强制**】脚本内禁止使用DROP DATABASE、DROP TABLE、TRUNCATE TABLE操作，包括DROP TABLE IF EXISTS语法
    3. DML脚本提交相关
       1. UPDATE

若需要通过SQL脚本方式运行的，须由符合以下标准：

* + - * 1. 【**强制**】数据库执行计划(explain)，单条SQL遍历记录条数(rows)不超过5万条
        2. 【**强制**】单个脚本SQL更新记录总数不超过2000条
        3. 【**强制**】IN里的ID记录数不超过200条
        4. 【**强制**】禁止UPDATE…SELECT…和UPDATE…SELECT…JOIN语法
        5. 【**强制**】上线需有回退SQL
      1. INSERT
         1. 【**强制**】单个脚本插入记录不超过2000条
         2. 【**强制**】禁止INSERT INTO …SELECT…和INSERT INTO …SELECT… JOIN语法
         3. 【**强制**】上线需有回退SQL
      2. DELETE

【**强制**】上线脚本若有物理删除操作的，必须单独通知DBA，并由DBA进行上线审核、上线备份

* + - 1. JOB规范
         1. 【**强制**】研发自行开发的JOB，运行时写并发量每秒不超过200条，如有特殊情况需DBA评估
         2. 【**强制**】所有SQL必须按主键拆成单条
         3. 【**强制**】必须与运维确认JOB运行时间点，合理避开其他JOB运行高峰
         4. 【**强制**】每天凌晨0点至2点之间禁止运行业务性定时器或脚本
      2. 【**强制**】DML脚本中存在全局变量的，必须使用begin/commit开始关闭事务，确保修改数据一致性
      3. 【**强制**】禁止使用REPLACE INTO语法，因需要DELETE权限，是一种变相的DELETE操作

# 六、索引规范

* 1. 【**强制**】每张表必须有一个主键索引，并使用自增长ID，禁止使用UUID

**说明**：符合INNODB高性能原理，自增的主键在物理排序上也是有序的，可以有效提高查询效率，避免随机读取IO

* 1. 【**强制**】联合索引中的字段数不能超过3个，并且区分度最高的在最左边，例如：

where a > 100 and b=3

**说明**：这个组合索引需要把b放在a前面，alter table TAB add indexindexname(b,a);

* 1. 【**强制**】建立索引时优先考虑添加维度型字段，例如：根据房东维度landlordid

wherelandlordid=10000000 and communityid = 3376

**说明**：一张表的索引资源肯定是最紧张的，不可能每个字段或大部分字段上都加索引，这个时候在设计表的时候就要考虑建表维度应该使用哪些字段，这类字段也适合作为索引，根据这些索引查询时结果集会大幅减小

* 1. 【**强制**】唯一索引值不能和主键重复

**说明**：从功能上讲，主键本身就是一种不含NULL的唯一索引

* 1. 【**强制**】业务上具有唯一特征的字段，即使是多个字段组合，也必须建成唯一索引

**说明**：不要以为唯一索引影响了insert速度，这个速度损耗可以忽略，但提高查找速度是明显的；另外，即使在应用层做了非常完善的校验控制，只要没有唯一索引，根据墨菲定律，必然会产生脏数据

* 1. 【**强制**】每张表的二级索引不得超过7个

|  |  |
| --- | --- |
| 字段数 | 可加个数 |
| 10< | 2 |
| 10-20 | 3 |
| 21-30 | 4 |
| 31-40 | 6 |
| 41-50 | 7 |

* 1. 【**强制**】使用到ORDER BY，GROUP BY，DISTINCT时字段必须是索引字段

**说明**：索引存放时都是有序的，这是天然的排序方法了；查询条件必须遵循最小化原则，避免全表扫描，如果可以使用到覆盖索引，查询速度会有数量级提高

* 1. 【**强制**】对长度过长的VARCHAR字段建立索引时，可以选择以下方案之一

1. 可以新建一个int列做索引，在JAVA中使用HASH转换成数字然后存入该列
2. 可以选择字段的前n位做前缀索引，n < 16, 并且前n位的选择性较高，重复值较少，下面列子中n为6

alter table tablename1 add key (column1(6));

* 1. 【**强制**】禁止左模糊或全模糊查询，如果需要请走搜索引擎来解决，like后面的百分号通配符不能在匹配值之前，如：

**错误语句**：select id,accountnum from flat\_flats where accountnum like "%01dominic%" #不走索引

**正确语句**：select id,accountnum from flat\_flats where accountnum like "01dominic" #走索引

**说明**：索引文件具有B-TREE的最左前缀匹配特性，如果左边的值未确定，那么无法使用此索引，必定全表扫描

* 1. 【**推荐**】表关联尽量使用主键条件关联，两表关联时主表推荐使用主键关联其他表
  2. 【**强制**】在WHERE条件后面做日期范围过滤，请不要使用函数转换，建议先用select获取想要的时间后存入变量

**错误语句**：DATE\_FORMAT(bill\_salebill.dueDate,'%Y-%m-%d')>=DATE\_FORMAT('2016-05-01','%Y-%m-%d') #杜绝这类语句，因为不会走索引造成全表扫描。

**正确语句**：bill\_salebill.dueDate between "2016-05-01 00:00:00" and "2016-05-01 23:59:59"

* 1. 【**参考**】关于单表查询索引合并
     1. 单表查询用到多个索引时，MYSQL5.5之后可以使用到索引合并，多个走索引的条件可以用OR作为条件关联，会使用到index\_merge优化手段中的Using Union算法。但是多表关联时不会使用到这个优化算法的。
  2. 【**推荐**】合理使用覆盖索引，避免回表
  3. 【**强制**】建立索引或删除索引必须由DBA评审
  4. 【**参考**】不等于运算符
     1. MYSQL5.6版本会对不等于运算符做查询优化，在利用主键查询时，条件中有!=或者<>运算符时仍然会用到索引，利用二级索引查询时如果查询只输出索引字段（覆盖索引）也会走到索引，除此之外数据库会进行全表扫描。
  5. 【**强制**】WHERE条件中不能使用任何表达式

建议使用select字段获取需要的值后计算存入相应变量。

例：

**错误语句**：select fr.id,fr.showpirce from flat\_roomfr where fr.showprice between 0 and fr.showprice + 1 #不走索引

**正确的方法**：先通过select获取变量值spvalueadd1，再将变量值放入主查询sql中（以下SQL并非JAVA示例，具体请根据JAVA示例自行调整）。

* 1. 【**强制**】防止因字段类型不同造成隐式转换，导致索引失效

例如：phone是varchar字段类型

**错误语句**：select id,name,phone,idcard,address from user\_renter where phone=135246123456;#全表扫描

**正确语句**：select id,name,phone,idcard,address from user\_renter where phone=’135246123456’#走索引

* 1. 【**推荐**】更新非常频繁的表，updateTime如果是CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMPTE特性，**不建议加索引**

**说明**：频繁更新的表使用CURRENT\_TIMESTAMP ONUPDATE CURRENT\_TIMESTAMPTE数据库当前数据自动变更时间特性，会导致该字段索引失效或部分失效。

# 七、SQL规范

## 1.query

* + 1. **join**
       1. 【**强制**】OLTP非报表业务，禁止join
       2. 【**强制**】OLTP报表业务，不超过3个join
       3. 【**强制**】OLAP报表业务join不限，规范如下：
          1. WHERE条件和ON条件，ON优先
          2. 表关联尽量使用主键条件关联，两表关联时主表推荐使用主键关联其他表
          3. 复杂关联查询中禁止携带超过1000条或大于16M结果集的子查询
          4. 优先使用inner join关联
       4. 【**强制**】OLTP所有业务，禁止一切跨库关联查询
    2. **simple query**
       1. 【**强制**】禁止select \* from tab语法，显式指明字段。
       2. 【**强制**】主条件必须是索引字段，禁止全表扫描
       3. 【**强制**】禁止嵌套子查询
       4. 【**强制**】单表查询多个条件只有全是索引字段时可以使用OR连接符
       5. 【**强制**】严格遵循上述第六项索引规范
       6. 【**强制**】多条件时用括号明确区分条件and和or的执行顺序
    3. **semi-join(半连接)**

1. 【**强制**】IN连接的子查询结果集不得超过200条ID记录

## 2.insert

* + 1. 【**强制**】禁止insert into select \* from tab语法。
    2. 【**强制**】多条插入时使用insert into table(id,name)values(1,’test1’),(2,’test2’),(3,’test3’);标准语法

## 3.update

* + 1. 【**强制**】where必须带条件
    2. 【**强制**】where条件必须包含索引字段
    3. 【**强制**】update必须先用select查询到数据后再做更新操作，条件中禁止使用JOIN关联

**说明**：避免出现误修改，避免更新因关联导致的各种非必要锁表，最简单的UPDATE语句可以有效减小二进制日志量，所以必须禁止JOIN关联

* + 1. 【**强制**】update禁止全字段更新

**说明**：若只需更新一个字段，那就只更新这一个字段，如果全字段都更新也会造成二进制日志量激增，并且容易引发应用程序隐患

## 4.delete

* + 1. 【**强制**】禁止（DBA除外）
    2. 【**强制**】所有应用只允许逻辑删除操作，即使用valid字段表示一行记录是否有效

## 5.replace

* + 1. 【**强制**】禁止

## 6.存储过程

* + 1. 【**强制**】除运维性质存储过程外全部禁止使用

**说明**：存储过程难以调试和扩展，更没有移植性

## 7.触发器

* + 1. 【**强制**】禁止

## 8.事件

* + 1. 【**强制**】禁止

## 9.视图

* + 1. 【**强制**】禁止

# 八、慢查询规范

## 1.慢查询标准【推荐】

* + 1. 第一阶段：面向真实用户的功能SQL不得超过500毫秒
    2. 第二阶段：面向真实用户的功能SQL不得超过200毫秒
    3. 第三阶段：面向真实用户的功能SQL不得超过100毫秒

## 2.慢查询巡检机制【推荐】

* + 1. 每两周进行一次慢查询巡检，公布TOP100慢SQL细则
    2. 慢查询日志定期归档

# 九、安全

* 1. 【**强制**】账号安全
     1. 所有业务需使用独立账号访问数据库
     2. 除DBA之外所有人员不得使用ROOT权限
     3. 业务中使用到的相关账号在配置文件中必须全部加密
  2. 【**强制**】环境安全
     1. 生产环境与测试、开发环境必须全部物理隔离，使用不同网段
     2. 数据库必须使用随机生成的用户名与强密码
     3. 数据库密码保存必须以加密方式呈现，使用第三方软件管理
  3. 数据备份安全
     1. 【**强制**】数据需异地备份
     2. 【**推荐**】备份需实现逻辑备份与物理备份共行机制
     3. 【**推荐**】备份需实现增量机制，现在已使用innobackup工具实现每周一次全备、每小时增备策略

# 十、数据行为规范

* 1. 【**强制**】批量导入、导出数据须提前通知DBA，导出生产环境数据这类操作必须经过技术部一级主管批准，由DBA协助完成
  2. 【**强制**】有推广活动或上线新功能须提前通知DBA及上级领导，并需测试部门测试评估压力及性能情况
  3. 【**强制**】不准使用SUPER权限连接数据库
  4. 【**强制**】单表多次ALTER操作必须合入到一次操作中完成
  5. 【**强制**】单表新增字段或修改字段类型长度必须提前通知DBA，由DBA评估影响并安排操作时间段
  6. 【**强制**】重要数据删除或修改，如需备份必须提前通知到DBA，需要在确认邮件中明确写明需备份相关数据
  7. 【**强制**】禁止所有业务在事务中使用显式加锁，容易增加死锁或阻塞几率，造成大量事务回滚
  8. 【**强制**】SQL禁止在程序中拼接，防止DML失误操作
  9. 【**强制**】所有his后缀的表禁止程序业务侧查询、修改、删除，如：flat\_room\_his
  10. 【**强制**】事务中的DML操作遵循最小化颗粒度原则，在一个事务中单表只对一条相关记录进行DML操作并且使用主键条件检索记录，避免产生死锁而造成的事务回滚。

# 十一、每个规范的监察机制及奖惩制度

## 1.以上规范适用于所有历史SQL

* + 1. 在规范出台之前，违反本规范事项者，须限期整改
    2. 在规范出台之后，违反规范事项者，根据本项规定对违反者进行相应惩处。
    3. 须有监控措施，实现事前、事后的有效监控。
    4. 本规范须在整个研发中心进行宣导

## 2.设计规范

* + 1. 对所有建表DDL类SQL进行自动审核
    2. 表设计阶段，必须符合设计规范，表结构由架构师、业务经理或开发、DBA审核其合理性。
    3. 审核通过后，如果由于表结构引起的事故由架构师与DBA共同承担责任。
    4. 审核失败后，业务开发在未通知架构师或DBA的情况下仍然强行提交相关DDL引起的事故由业务开发人员承担主要责任。

## 3.枚举规范

* + 1. 所有DDL中包含的枚举使用码表保存。

## 4.SQL脚本提交规范

* + 1. 建议针对初犯者以教育为主，惩罚为辅。
    2. 若经反复宣导，该研发人员连续2次仍未按SQL脚本提交规范提交者进行相应惩罚。

## 5.索引规范

* + 1. 研发人员开发查询代码时应主动确认主条件字段是否有建立索引，如果没有，需要和DBA确认，并由DBA评估是否需要建立索引。
    2. 研发人员开发的SQL若是在没有和DBA确认的情况下依旧上线，应负主要责任。
    3. DBA定期对表索引做巡检，因不合理索引引起的事故应由DBA负主要责任。

## 6.SQL规范

* + 1. 由架构组设计编写SQL检测程序，用于监控代码中不符合规范的SQL
    2. 代码中所有SQL都要写明编写作者、修改时补充修改作者，否则将**严惩**不贷。
    3. 研发人员开发的任何SQL代码需在104测试环境或正式环境运行自检（目前DBA只有一个，精力有限，无法对所有SQL逐一审核），研发人员根据慢查询标准向DBA提出支持要求，未经测试直接上线或明知慢SQL未通知DBA者应承担主要责任。
    4. 建立SQL虚拟专家小组，在遇到棘手优化困难时可寻求专家组配合支持，提高优化程度。
    5. 简单SQL优化失误，DBA应负主要责任。

## 7.慢查询规范

* + 1. 使用数据库慢查询监控慢SQL
    2. DBA每周公布一次慢查询TOP100细则，各研发部组根据功能认领慢SQL
    3. 在认领后需根据慢查询标准限时完成整改，否则因其引起的事故需承担主要责任
    4. DBA有义务提出合理建议及优化方案，否则因其引起的事故需承担相应责任
    5. 慢查询发现、整改、测试、发布需根据正式流程操作，个别紧急整改需获得操作预授权。

## 8.安全规范

* + 1. 备份数据后需有自动校验机制保证备份成功，如果失败需通过邮件等方式进行报警，否则因备份丢失数据的事故由DBA承担相应责任。
    2. 生产环境数据因与测试环境交错而导致的事故由DBA、运维承担相应责任。