**数据库规范**

目录

[MYSQL数据库设计基本原则 3](#_Toc189036162)

[1、核心原则   3](#_Toc1002460821)

[2、字段类原则   3](#_Toc1349807832)

[3、索引类原则 3](#_Toc202985516)

[4、sql类原则 3](#_Toc1373535976)

[5、性能分析工具   4](#_Toc1717427029)

[一、命名规范 5](#_Toc468377076)

[1.1 表命名规范 5](#_Toc1485950077)

[规范 5](#_Toc1275613176)

[示例 5](#_Toc901401031)

[待优化 5](#_Toc1497482079)

[1.2字段命名 6](#_Toc1820442560)

[规范 6](#_Toc978587111)

[示例 6](#_Toc1683805851)

[待优化命名 6](#_Toc185437591)

[字段设计原则 6](#_Toc650820140)

[字段规范描述 7](#_Toc1199878809)

[二、设计规范 7](#_Toc1491697533)

[2.1 范式应用 7](#_Toc1236342053)

[2.2 表设计 8](#_Toc149116399)

[2.3 字段设计 8](#_Toc85901944)

[2.4 类型选择 8](#_Toc644962024)

[日期和时间数据类型 8](#_Toc1526770959)

[整型 9](#_Toc157409910)

[浮点型 9](#_Toc2035987913)

[字符串数据类型 9](#_Toc844422493)

[其它类型 9](#_Toc1636900475)

[三． 索引 10](#_Toc2120765255)

# MYSQL数据库设计基本原则

## 1、核心原则

1. 不在数据库做运算;
2. cpu计算务必移至业务层;
3. 控制列数量(字段少而精,字段数建议在20以内);
4. 平衡范式与冗余(效率优先；往往牺牲范式)
5. 拒绝3B(拒绝大sql语句：big sql、拒绝大事物：big transaction、拒绝大批量：big batch);

## 2、字段类原则

1. 用好数值类型(用合适的字段类型节约空间);
2. 字符转化为数字(能转化的最好转化,同样节约空间、提高查询性能);
3. 避免使用NULL字段(NULL字段很难查询优化、NULL字段的索引需要额外空间、NULL字段的复合索引无效);
4. 少用text类型(尽量使用varchar代替text字段);

## 3、索引类原则

1. 合理使用索引(改善查询,减慢更新,索引一定不是越多越好);
2. 字符字段必须建前缀索引;
3. 不在索引做列运算;
4. innodb主键推荐使用自增列(主键建立聚簇索引,主键不应该被修改,字符串不应该做主键)(理解Innodb的索引保存结构就知道了);
5. 不用外键(由程序保证约束);

4、sql类原则

1. sql语句尽可能简单(一条sql只能在一个cpu运算,大语句拆小语句,减少锁时间,一条大sql可以堵死整个库);
2. 简单的事务;
3. 避免使用trig/func(触发器、函数不用客户端程序取而代之);
4. 不用select \*(消耗cpu,io,内存,带宽,这种程序不具有扩展性);
5. OR改写为IN(or的效率是n级别);
6. OR改写为UNION(mysql的索引合并很弱智);

select id from t where phone = ’159′ or name = ‘john’;

            =>

            select id from t where phone=’159′

            union

            select id from t where name=’jonh’

1. 避免负向%;
2. 慎用count(\*);
3. limit高效分页(limit越大，效率越低);
4. 使用union all替代union(union有去重开销);
5. 少用连接join;
6. 使用group by;
7. 请使用同类型比较;
8. 打散批量更新;

## 5、性能分析工具

show profile;

mysqlsla;

mysqldumpslow;

explain;

show slow log;

show processlist;

# 一、命名规范

采用26个英文字母(区分大小写)和0-9的自然数(经常不需要)加上下划线'\_'组成，命名简洁明确，多个单词用下划线'\_'分隔，一个项目一个数据库，多个项目慎用同一个数据库。

表、列、索引、视图、存储过程、自定义函数，的命名的原则是简、短、精以及建明知意，易懂好记，有意义的英文单词、常用缩写，多个单词组成的，当长度比较大时应使用缩写，特殊情况使用首字母。

## 1.1 表命名规范

### 规范

1. 采用26个英文字母(区分大小写)和0-9的自然数(经常不需要)加上下划线'\_'组成，命名简洁明确，多个单词用下划线'\_'分隔
2. 全部小写命名，禁止出现大写
3. 禁止使用数据库关键字，如：name，time ，datetime，password等
4. 表名称不应该取得太长（一般不超过三个英文单词）
5. 表的名称一般使用名词或者动宾短语
6. 用单数形式表示名称，例如，使用 employee，而不是 employees
7. 明细表的名称为：主表的名称+字符dtl（detail缩写）
8. 例如：采购定单的名称为：po\_order，则采购定单的明细表为：po\_orderdtl
9. 表必须填写描述信息（使用SQL语句建表时）

### 示例

1. 系统模块开头：sys\_（system）
2. 模块\_+功能点  示例：alllive\_log   alllive\_category
3. 功能点  示例：live   message
4. 通用表  示例：all\_user
5. 正确使用：landing\_task、landing\_task\_cancel
6. 禁止出现：\_landing\_task

### 待优化

①冗余：

错误示例：yy\_alllive\_video\_recomment    yy\_alllive\_open\_close\_log

说明：去除项目名，简化表名长度，去”yy\_”

②相同类别表命名存在差异，管理性差

错误示例：yy\_all\_live\_category    yy\_alllive\_comment\_user

说明：去除项目名，统一命名规则，均为”yy\_alllive\_”开头即可

③命名格式存在差异

错误示例：yy\_showfriend    yy\_user\_getpoints    yy\_live\_program\_get

说明：去除项目名，统一命名规则，动宾短语分离且动宾逻辑顺序统一

## 1.2字段命名

### 规范

（1）采用26个英文字母(区分大小写)和0-9的自然数(经常不需要)。

（2）全部小写命名，禁止出现大写

（3）字段必须填写描述信息

（4）禁止使用数据库关键字，如：name，time ，datetime password 等

（5）字段名称一般采用名词或动宾短语

（6）采用字段的名称必须是易于理解，一般不超过三个英文单词

（7）在命名表的列时，不要重复表的名称

例如，在名employe的表中避免使用名为employee\_lastname的字段

（8）不要在列的名称中包含数据类型

（9）字段命名使用完整名称，禁止缩写

### 示例

1. 单个名词  示例：userid    username    sex
2. 动宾短语  示例：isfriend   isgood
3. 符合名次简称 示例：增值税：vat
4. 拼音首字母 示例：增值税代开缴税金额：vatdkjsje
5. 符合名词简称+单个名词 示例：增值税税率：vatrate
6. 符合名词简称+拼音 示例：增值税附加税税率：vatfjssl

### 待优化命名

①大小写规则不统一

错误示例：user\_id    houseID

说明：使用统一规则，修改为”user\_id”，”house\_id”

②加下划线规则不统一

错误示例：user\_name    user\_id    is\_friend    is\_good

说明：严禁使用下划线进行分类，降低可读性，方便查看，修改为”username”，”userid”，”isfriend”，”isgood”。

③字段表示不明确

错误示例：user\_id person\_id

说明：使用完整名称，降低可读性，修改为”userid”，”personid”

### 字段设计原则

（1）所有字段在设计时，除以下数据类型timestamp、image、datetime、smalldatetime、uniqueidentifier、binary、sql\_variant、binary 、varbinary外，必须有默认值，字符型的默认值为一个空字符值串’’，数值型的默认值为数值0，逻辑型的默认值为数值0

（2）系统中所有逻辑型中数值0表示为“假”，数值1表示为“真”，datetime、smalldatetime类型的字段没有默认值，必须为NULL

（3）用尽量少的存储空间来存储一个字段的数据

使用int就不要使用varchar、char，

用varchar(16)就不要使varchar(256)

IP地址使用int类型

固定长度的类型最好使用char，例如：邮编(postcode)

能使用tinyint就不要使用smallint，int

最好给每个字段一个默认值，最好不能为null

（4）用合适的字段类型节约空间

字符转化为数字(能转化的最好转化，同样节约空间、提高查询性能)

避免使用NULL字段(NULL字段很难查询优化、NULL字段的索引需要额外空间、NULL字段的复合索引无效)

少用text类型(尽量使用varchar代替text字段)

### 字段规范描述

1. 尽量遵守第三范式的标准（3NF）

     表内的每一个值只能被表达一次

     表内的每一行都应当被唯一的标示

     表内不应该存储依赖于其他键的非键信息

1. 如果字段事实上是与其它表的关键字相关联而未设计为外键引用，需建索引
2. 如果字段与其它表的字段相关联，需建索引
3. 如果字段需做模糊查询之外的条件查询，需建索引
4. 除了主关键字允许建立簇索引外，其它字段所建索引必须为非簇索引

# 二、设计规范

## 2.1 范式应用

1. 第一范式(1NF)：字段值具有原子性,不能再分(所有关系型数据库系统都满足第一范式);

 例如：姓名字段,其中姓和名是一个整体,如果区分姓和名那么必须设立两个独立字段;

1. 第二范式(2NF)：一个表必须有主键,即每行数据都能被唯一的区分;

            备注：必须先满足第一范式;

1. 第三范式(3NF)：一个表中不能包涵其他相关表中非关键字段的信息,即数据表不能有沉余字段;

            备注：必须先满足第二范式;

往往我们在设计表中不能遵守第三范式,因为合理的沉余字段将会给我们减少join的查询;

## 2.2 表设计

1. 通常情况下每张表必须有主键,主键可以是自增唯一序列，也可以是其他表的主键，在本表合并主外键，对于一对多情况，主键必须自增
2. 主键自增时命名为id，为外键时采用外间表+id名称，候选键等不作要求），对于非主键，具有唯一性质的，必须加，唯一索引
3. 每张表，每个字段都必须有正确的中文描述说明，以便快速生成数据字典
4. 业务数据表必要时冗余基础信息表的核心字段，如：部门编码、部门名称，需要成对出现在业务数据表中，对于冗余字段在设计维护时，只能删（逻辑）/增，无法修改。

## 2.3 字段设计

1. 对于字符类型（VERCHAR）字段，确定数据值仅是数字、字母并无其它特殊字符的情况下使用VERCHAR字段，其它必须使用NVARCHAR类型，避免使用VARCHAR类型，了解NVARCHAR类型后，你会更喜欢它。
2. 表字段的长度要统一，尤其基础数据在业务数据中的引用，如：部门编码、部门名称，在任何业务数据表中的长度务必保持一致

存储过程、自定义函数、视图，以及调度，必须要有详细的注释说明，同时在对应项目中做维护记录。

1. 统一入口，统一出口原则：相同意义的数据存储、提取，使用同一过程和方法
2. 不要盲目的加入其他索引，根据需要统一加索引。

## 2.4 类型选择

### 日期和时间数据类型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 长度 | 名称 | 格式 | 应用 |
| date | 3字节 | 日期 | 2014-09-18 | 年月日/年月 |
| time | 3字节 | 时间 | 08:12:12 |  |
| dateTime | 8字节 | 日期-时间 | 2014-09-18  08:12:12 |  |
| timeStamp | 4字节 | 日期-时间 （自动修改） | 2014-09-18  08:12:12 |  |
| year | 4字节 | 年份 | 1999 |  |

### 整型

|  |  |  |
| --- | --- | --- |
| 数据类型 | 长度 | 范围 |
| tinyint | 1字节 | （-128~127） |
| smallint | 2字节 | （-32768~32767） |
| mediumint | 3字节 | （-8388608~8388607） |
| int | 4字节 | （-2147483648~2147483647） |
| bigint | 8字节 | （+-9.22\*10的18次方） |

### 浮点型

|  |  |
| --- | --- |
| MySQL数据类型 | 含义 |
| float(m, d) | 4字节，单精度浮点型，m总个数，d小数位 |
| double(m, d) | 8字节，双精度浮点型，m总个数，d小数位 |
| decimal(m, d) | decimal是存储为字符串的浮点数 |

### 字符串数据类型

|  |  |
| --- | --- |
| MySQL数据类型 | 含义 |
| char(n) | 固定长度，最多255个字符 |
| varchar(n) | 可变长度，最多65535个字符 |
| tinytext | 可变长度，最多255个字符 |
| text | 可变长度，最多65535个字符 |
| mediumtext | 可变长度，最多2的24次方-1个字符 |
| longtext | 可变长度，最多2的32次方-1个字符 |

### 其它类型

1. **Enum**  
   enum(“member1″, “member2″, … “member65535″)  
   enum数据类型就是定义了一种枚举，最多包含65535个不同的成员。当定义了一个enum的列时，该列的值限制为列定义中声明的值。如果列声明包含NULL属性，则NULL将被认为是一个有效值，并且是默认值。如果声明了NOT NULL，则列表的第一个成员是默认值。
2. **Set**  
   set(“member”, “member2″, … “member64″)  
   set数据类型为指定一组预定义值中的零个或多个值提供了一种方法，这组值最多包括64个成员。值的选择限制为列定义中声明的值。

# 索引

-- 参考《阿里巴巴开发手册1.3版》

1. **【强制】业务上具有唯一特性的字段，即使是多个字段的组合，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，必然有脏数据产生。

1. **【强制】超过三个表禁止join。需要join的字段，数据类型必须绝对一致；多表关联查询时，保证被关联的字段需要有索引。**

说明：即使双表join也要注意表索引、SQL性能。

1. **【强制】在varchar字段上建立索引时，必须指定索引长度，没必要对全字段建立索引，根据实际文本区分度决定索引长度即可。**

说明：索引的长度与区分度是一对矛盾体，一般对字符串类型数据，长度为20的索引，区分度会高达90%以上，可以使用count(distinct left(列名, 索引长度))/count(\*)的区分度来确定。

1. **【强制】页面搜索严禁左模糊或者全模糊，如果需要请走搜索引擎来解决。**

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。

1. **【推荐】如果有order by的场景，请注意利用索引的有序性。order by 最后的字段是组合索引的一部分，并且放在索引组合顺序的最后，避免出现file\_sort的情况，影响查询性能。**

正例：where a=? and b=? order by c; 索引：a\_b\_c

反例：索引中有范围查找，那么索引有序性无法利用，如：WHERE a>10 ORDER BY b; 索引a\_b无法排序。

1. **【推荐】利用覆盖索引来进行查询操作，避免回表。**

说明：如果一本书需要知道第11章是什么标题，会翻开第11章对应的那一页吗？目录浏览一下就好，这个目录就是起到覆盖索引的作用。

正例：能够建立索引的种类分为主键索引、唯一索引、普通索引三种，而覆盖索引只是一种查询的一种效果，用explain的结果，extra列会出现：using index。

1. **【推荐】利用延迟关联或者子查询优化超多分页场景。**

说明：MySQL并不是跳过offset行，而是取offset+N行，然后返回放弃前offset行，返回N行，那当offset特别大的时候，效率就非常的低下，要么控制返回的总页数，要么对超过特定阈值的页数进行SQL改写。

正例：先快速定位需要获取的id段，然后再关联： SELECT a.\* FROM 表1 a, (select id from 表1 where 条件 LIMIT 100000,20 ) b where a.id=b.id

1. **【推荐】 SQL性能优化的目标：至少要达到 range 级别，要求是ref级别，如果可以是consts最好。**

说明：

1. consts 单表中最多只有一个匹配行（主键或者唯一索引），在优化阶段即可读取到数据。
2. ref 指的是使用普通的索引（normal index）。

3）range 对索引进行范围检索。 反例：explain表的结果，type=index，索引物理文件全扫描，速度非常慢，这个index级别比较range还低，与全表扫描是小巫见大巫。

1. **【推荐】建组合索引的时候，区分度最高的在最左边。**

说明：存在非等号和等号混合时，在建索引时，请把等号条件的列前置。如：where c>? and d=? 那么即使c的区分度更高，也必须把d放在索引的最前列，即索引idx\_d\_c。

正例：如果where a=? and b=? ，如果a列的几乎接近于唯一值，那么只需要单建idx\_a索引即可。

1. **【推荐】防止因字段类型不同造成的隐式转换，导致索引失效。**
2. **【参考】创建索引时避免有如下极端误解**

1）宁滥勿缺。认为一个查询就需要建一个索引。

2）宁缺勿滥。认为索引会消耗空间、严重拖慢更新和新增速度。

3）抵制惟一索引。认为业务的惟一性一律需要在应用层通过“先查后插”方式解决。