# 数据库开发设计规范

# 数据库设计规范

KB小秘书2019-07-166545浏览量

**简介：**

基于阿里数据库设计规范扩展而来

基于阿里数据库设计规范扩展而来

### 设计规范

1.【推荐】字段允许适当冗余，以提高查询性能，但必须考虑数据一致。冗余字段应遵循:

* 不是频繁修改的字段。
* 不是 varchar 超长字段，更不能是 text 字段。

正例:商品类目名称使用频率高，字段长度短，名称基本一成不变，可在相关联的表中冗余存 储类目名称，避免关联查询。

2.【推荐】单表行数超过 500 万行或者单表容量超过 2GB，才推荐进行分库分表。 说明:如果预计2年后的数据量根本达不到这个级别，请不要在创建表时就分库分表。

3.【推荐】id必须是主键，每个表必须有主键，且保持增长趋势的， 小型系统可以依赖于 MySQL 的自增主键，大型系统或者需要分库分表时才使用内置的 ID 生成器

4.【强制】id类型没有特殊要求，必须使用bigint unsigned，禁止使用int，即使现在的数据量很小。id如果是数字类型的话，必须是8个字节。参见最后例子

* 方便对接外部系统，还有可能产生很多废数据
* 避免废弃数据对系统id的影响
* 未来分库分表，自动生成id，一般也是8个字节

5.【推荐】字段尽量设置为 NOT NULL， 为字段提供默认值。 如字符型的默认值为一个空字符值串’’;数值型默认值为数值 0;逻辑型的默认值为数值 0;

6.【推荐】每个字段和表必须提供清晰的注释

7.【推荐】时间统一格式:‘YYYY-MM-DD HH:MM:SS’

8.【强制】更新数据表记录时，必须同时更新记录对应的 gmt\_modified 字段值为当前时间,

### 命名规范

1.【强制】表达是与否概念的字段，必须使用 is\_xxx 的方式命名，数据类型是 unsigned tinyint ( 1表示是，0表示否)。  
说明:任何字段如果为非负数，必须是 unsigned。  
正例:表达逻辑删除的字段名 is\_deleted，1 表示删除，0 表示未删除。  
2.【强制】表名、字段名必须使用小写字母或数字，禁止出现数字开头，禁止两个下划线中间只 出现数字。数据库字段名的修改代价很大，因为无法进行预发布，所以字段名称需要慎重考虑。 说明:MySQL 在 Windows 下不区分大小写，但在 Linux 下默认是区分大小写。因此，数据库 名、表名、字段名，都不允许出现任何大写字母，避免节外生枝。 正例:health\_user，rdc\_config，level3\_name 反例:HealthUser，rdcConfig，level\_3\_name  
3.【强制】表名不使用复数名词。 说明:表名应该仅仅表示表里面的实体内容，不应该表示实体数量，对应于 DO 类名也是单数 形式，符合表达习惯。  
4.【强制】禁用保留字，如 desc、range、match、delayed 等，请参考 MySQL 官方保留字。  
5.【强制】主键索引名为 pk\_字段名;唯一索引名为 uk\_字段名;普通索引名则为 idx\_字段名。  
说明:pk\_ 即 primary key;uk\_ 即 unique key;idx\_ 即 index 的简称。  
6.【强制】小数类型为 decimal，禁止使用 float 和 double。  
说明:float 和 double 在存储的时候，存在精度损失的问题，很可能在值的比较时，得到不 正确的结果。如果存储的数据范围超过 decimal 的范围，建议将数据拆成整数和小数分开存储。  
7.【强制】如果存储的字符串长度几乎相等，使用 char 定长字符串类型。  
8.【强制】varchar 是可变长字符串，不预先分配存储空间，长度不要超过 5000，如果存储长 度大于此值，定义字段类型为 text，独立出来一张表，用主键来对应，避免影响其它字段索 引效率。  
9.【强制】表必备三字段:id, is\_delete,gmt\_create, gmt\_modified。 说明:其中id必为主键，类型为unsigned bigint、单表时自增、步长为1。gmt\_create, gmt\_modified 的类型均为 date\_time 类型，前者现在时表示主动创建，后者过去分词表示被 动更新。  
10.【强制】所有命名必须使用全名，有默认约定的除外，如果超过 30 个字符，使用缩写，请尽量名字易懂简短，如 description --> desc;information --> info;address --> addr 等  
11.【推荐】表的命名最好是加上“业务名称\_表的作用”。 正例:health\_user / trade\_config  
12.【推荐】库名与应用名称尽量一致。如health  
13.【推荐】如果修改字段含义或对字段表示的状态追加时，需要及时更新字段注释  
14.【推荐】所有时间字段，都以 gmt\_开始，后面加上动词的过去式，最后不要加上 time 单词，例如 gmt\_create

### 类型规范

1.表示状态字段(0-255)的使用 TINYINT UNSINGED，禁止使用枚举 类型，注释必须清晰地说明每个枚举的含义，以及是否多选等

2.表示boolean类型的都使用TINYINT(1),因为mysql本身是没有boolean类型的，在自动生成代码的时候，DO对象的字段就是boolean类型，例如 is\_delete;其余所有时候都使用TINYINT(4)

TINYINT(4),这个括号里面的数值并不是表示使用多大空间存储，而是最大显示宽度，并且只有字段指定zerofill时有用，没有zerofill，(m)就是无用的,例如id BIGINT ZEROFILL NOT NULL,所以建表时就使用默认就好了，不需要加括号了，除非有特殊需求，例如TINYINT(1)代表boolean类型。

TINYINT(1)，TINYINT(4)都是存储一个字节，并不会因为括号里的数字改变。例如TINYINT(4)存储22则会显示0022，因为最大宽度为4，达不到的情况下用0来补充。

3.【参考】合适的字符存储长度，不但节约数据库表空间、节约索引存储，更重要的是提升检索速度。

| 类型 | 字节 | 表示范围 |
| --- | --- | --- |
| tinyint | 1 | 无符号值: 0～255;有符号值： -128~127 |
| smallint | 2 | 无符号值: 0～65536;有符号值： -32768~32767 |
| mediumint | 3 | 无符号值: 0～16777215;有符号值： -8388608~8388607 |
| int | 4 | 无符号值: 0~4294967295;有符号值： -2147483648~2147483647 |
| bigint | 8 | 无符号值: 0~((2³²×²)-1);有符号值： -(2³²×²)/2 ~ (2³²×²)/2-1 |

4.非负的数字类型字段，都添加上 UNSINGED, 如可以使用 INT UNSINGED 字段存 IPV4

5.时间字段使用时间日期类型，不要使用字符串类型存储，日期使用DATE类型，年使用YEAR类型，日期时间使用DATETIME

6.字符串VARCHAR(N), 其中 N表示字符个数，请尽量减少 N 的大小，参考：code VARCHAR(32)；name VARCHAR(32)；memo VARCHAR(512)；

7.Blob 和 Text 类型所存储的数据量大，删除和修改操作容易在数 据表里产生大量的碎片，避免使用 Blob 或 Text 类型

### 索引规范

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。

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。

2.【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。

即使双表 join 也要注意表索引、SQL 性能。

3.【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。

4.【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。

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。

5.【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。

6.【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:主键索引、唯一索引、普通索引，而覆盖索引是一种查询的效果，用explain的结果，extra列会出现:using index。

7.【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

8.【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:

* 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。
* ref 指的是使用普通的索引(normal index)。
* 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。

9.【推荐】建组合索引的时候，区分度最高的在最左边。  
正例:如果 where a=? and b=? ，a 列的几乎接近于唯一值，那么只需要单建 idx\_a 索引即 可。

说明:存在非等号和等号混合判断条件时，在建索引时，请把等号条件的列前置。如:where a>? and b=? 那么即使 a 的区分度更高，也必须把 b 放在索引的最前列。

10【推荐】防止因字段类型不同造成的隐式转换，导致索引失效。

11.【参考】创建索引时避免有如下极端误解

* 宁滥勿缺。认为一个查询就需要建一个索引。
* 宁缺勿滥。认为索引会消耗空间、严重拖慢更新和新增速度。
* 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。

**12. 总结**

• 索引占磁盘空间，不要重复的索引，尽量短 • 只给常用的查询条件加索引 • 过滤性高的列建索引，取值范围固定的列不建索引 • 唯一的记录添加唯一索引 • 频繁更新的列不要建索引 • 不要对索引列运算 • 同样过滤效果下，保持索引长度最小 • 合理利用组合索引，注意索引字段先后顺序 • 多列组合索引，过滤性高的字段最前 • order by 字段建立索引，避免 filesort • 组合索引，不同的排序顺序不能使用索引 • <>!=无法使用索引

### SQL规范

1.【强制】不要使用 count(列名)或 count(常量)来替代 count()，count()是 SQL92 定义的 标准统计行数的语法，跟数据库无关，跟 NULL 和非 NULL 无关。

count(\*)会统计值为 NULL 的行，而 count(列名)不会统计此列为 NULL 值的行。

2.【强制】count(distinct col) 计算该列除 NULL 之外的不重复行数，

count(distinct col1, col2) 如果其中一列全为NULL，那么即使另一列有不同的值，也返回为0。

3.【强制】当某一列col的值全是 NULL 时，count(col)的返回结果为 0，但 sum(col)的返回结果为 NULL，因此使用 sum()时需注意 NPE 问题。

正例:可以使用如下方式来避免sum的NPE问题:SELECT IF(ISNULL(SUM(g)),0,SUM(g)) FROM table;

4.【强制】使用 ISNULL()来判断是否为 NULL 值。 说明:NULL 与任何值的直接比较都为 NULL。

* NULL<>NULL的返回结果是NULL，而不是false。
* NULL=NULL的返回结果是NULL，而不是true。
* NULL<>1的返回结果是NULL，而不是true。

5.【强制】 在代码中写分页查询逻辑时，若 count 为 0 应直接返回，避免执行后面的分页语句。

6.【强制】不得使用外键与级联，一切外键概念必须在应用层解决。 说明:以学生和成绩的关系为例，学生表中的 student\_id 是主键，那么成绩表中的 student\_id 则为外键。如果更新学生表中的 student\_id，同时触发成绩表中的 student\_id 更新，即为 级联更新。外键与级联更新适用于单机低并发，不适合分布式、高并发集群;级联更新是强阻 塞，存在数据库更新风暴的风险;外键影响数据库的插入速度。

7.【强制】禁止使用存储过程，存储过程难以调试和扩展，更没有移植性。

8.【强制】数据订正时，删除和修改记录时，要先 select，避免出现误删除，确认无误才能执行更新语句。

9.【推荐】in操作能避免则避免，若实在避免不了，需要仔细评估 in 后边的集合元素数量，控  
制在 1000 个之内。

10.【参考】如果有全球化需要，所有的字符存储与表示，均以 utf-8 编码，注意字符统计函数 的区别。

说明:

SELECT LENGTH(“轻松工作”); 返回为12  
SELECT CHARACTER\_LENGTH(“轻松工作”); 返回为4 如果需要存储表情，那么选择 utfmb4 来进行存储，注意它与 utf-8 编码的区别。

11.【参考】TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少，但 TRUNCATE 无事务且不触发trigger，有可能造成事故，故不建议在开发代码中使用此语句。 说明:TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句相同。

12.【推荐】不要写一个大而全的数据更新接口。传入为 POJO 类，不管是不是自己的目标更新字 段，都进行 update table set c1=value1,c2=value2,c3=value3; 这是不对的。执行 SQL 时，不要更新无改动的字段，一是易出错;二是效率低;三是增加 binlog 存储。

13.总结

• 能够快速缩小结果集的 WHERE 条件写在前面，如果有恒量条 件，也尽量放在前面 ，例如 where

1

=

1

• 避免使用 GROUP BY、DISTINCT 等语句的使用，避免联表查 询和子查询 • 能够使用索引的字段尽量进行有效的合理排列 • 针对索引字段使用 >, >=, =, <, <=, IF NULL 和 BETWEEN 将会 使用索引，如果对某个索引字段进行 LIKE 查询，使用 LIKE ‘%abc%’ 不能使用索引，使用 LIKE ‘abc%’ 将能够使用索引 • 如果在 SQL 里使用了 MySQL部分自带函数，索引将失效• 避免直接使用 select \*,只取需要的字段，增加使用覆盖索引使用的可能 • 对于大数据量的查询，尽量避免在 SQL 语句中使用 order by 字句 • 连表查询的情况下，要确保关联条件的数据类型一致，避免嵌套子查询 • 对于连续的数值，使用 between 代替

in

• where 语句中尽量不要使用 CASE 条件 • 当只要一行数据时使用 LIMIT

1

### 例子

CREATE TABLE

`health\_package`

(

`id`

bigint unsigned NOT NULL AUTO\_INCREMENT COMMENT

'序号'

,

`package\_id`

int unsigned NOT NULL COMMENT

'套系 id'

,

`module\_id`

int unsigned NOT NULL COMMENT

'模块 id'

,

`is\_delete`

tinyint unsigned NOT NULL DEFAULT

0

COMMENT

'是否删除，0-未删除，1-删除，默认为0'

,

`gmt\_create`

datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT

'Create time, common column by DB rules'

,

`gmt\_modified`

datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP COMMENT

'Modified time,common column by DB rules '

, PRIMARY KEY (

`id`

)) COMMENT=

'This table stores module and package of health for ...'

;

索引

存储

数据库

SQL

关系型数据库