生活旅行部
MYSQL 数据库规范

文档更改记录

版本号	修改日期	修改人	更改内容
1.0	2018 年 1 月 19	伍振华	初稿

文档审核记录

审核日期	审核人	备注

1. 建表规约

第1条 【强制】数据库字符集选择UTF8

第2条 【强制】建议使用InnoDB存储引擎

5.5 以后的默认引擎，支持事务，行级锁，更好的恢复性，高并发下性能更好，对多核，大内存，ssd 等硬件支持更好。

第3条 【强制】库名、表名、字段名不允许使用大写字母，使用“_”分割

大小写共用容易引起混乱，为了规范，不允许使用大写：

1) MySQL 有配置参数 lower_case_table_names，不可动态更改，linux 系统默认为 0，即库表名以实际情况存储，大小写敏感。如果是 1，以小写存储，大小写不敏感。如果是 2，以实际情况存储，但以小写比较。
2) 如果大小写混合用，可能存在 abc,Abc,ABC 等多个表共存，容易导致混乱。
3) 字段名显示区分大小写，但实际使用不区分，即不可以建立两个名字一样但大小写不一样的字段。

禁止出现数字开头，禁止两个下划线中间只出现数字。

正例：getter_admin，task_config，level3_name 反例：GetterAdmin，taskConfig，level_3_name。

第4条 【强制】库名、表名、字段名见名知意,建议使用名词而不是动词

1) 便于理解。
2) 库表是一种客观存在的事物，一种对象，所以建议使用名词。
3) 表名不使用复数名词：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。

第5条 【强制】库名定义为“ltyf_系统名缩写”

第6条 【强制】命名禁用保留字

如 desc 、 range 、 match 、 delayed 等 ， 请 参 考 MySQL 官 方 保 留 字（https://dev.mysql.com/doc/refman/5.7/en/keywords.html）。

第7条 【强制】表命名详细规则

1) 表名格式：系统简称_基本表名。
2) 表名长度不能超过 30 个字符，表名中含有单词全部采用单数形式。
3) 表名含有的单词用完整的单词,如果导致基本表名长度超过 26 个字符，则从最后一个单词开始，依次向前采用该单词的缩写。(如果没有约定的缩写，则采用该单词前 4 个字母来表示)。

第8条 【强制】使用整数类型（一般是TINYINT UNSIGNED）存储枚举类型

ENUM，有下面三个问题，不建议使用。

1) 添加新的值要做 DDL;
2) 默认值问题(将一个非法值插入 ENUM【也就是说，允许的值列之外的字符串】，将插入空字符串以作为特殊错误值);
3) 索引值问题（插入数字实际是插入索引对应的值）。

第9条 【强制】使用decimal存储小数类型，禁止使用float和double

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。

因为标准的计算机浮点数，在内部是用二进制表示的，但在将一个十进制数转换为二进制浮点数时，可能造成误差，原因是不是所有的数（比如一个带小数的十进制）都能转换成有限长度的二进制数。

另外，对于有固定小数位精度的小数类型（比如金额，精确到分，以元表示时），可以考虑使用整数表示（存储时乘以固定的倍数，在实际使用时再除掉）。

第10条 【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

比如UUID列，建议使用CHAR(36)。

第11条 【强制】表必备字段：create_time，update_time

create_time：表示记录创建时间，TIMESTAMP 类型，要求 NOT NULL DEFAULT CURRENT_TIMESTAMP，即自动设置为记录插入时间。

update_time：表示记录最后更新时间，TIMESTAMP 类型，要求 NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP，即自动设置为最后更新时间。

该功能要求 MYSQL 版本不低于 5.6，因为 5.6 之前一个表不支持多余一个的自动更新 TIMESTAMP 列。

第12条 【强制】整形定义中不添加长度，比如使用INT，而不是INT(4)

整数定义的长度不像字符串类型，不影响实际存储的数据长度，比如 INT(4)也可以存储12345，为了避免误解，定义时不指定长度。

第13条 【强制】varchar长度尽可能小，且不大于2000

VARCHAR(N)，N 尽可能小，因为 MySQL 在进行排序和创建临时表一类的内存操作时，会使用 N 的长度申请内存。

N 大于 2000 时，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。

第14条 【强制】数据库 - 视图命名

1) 视图名用系统简称_V_开头，视图名长度不能超过 30 个字符。
2) 视图由几个表产生就用下划线（_）连接几个表的名，如果表过多可以将表名适当简化，但一定要列出所有表名。

第15条 【强制】禁止使用VARBINARY、BLOB存储图片、文件

第16条 【强制】禁止使用分区表

MySQL 的分区表实际性能不是很好，且管理维护成本较高。

第17条 【推荐】临时库、临时表名必须以 tmp 为前缀并以日期为后缀

第18条 【推荐】备份库、备份表名必须以 bak 为前缀并以日期为后缀

第19条 【推荐】字段使用注释说明含义

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。

第20条 【推荐】适当使用冗余字段

冗余字段可以提高查询性能，但必须考虑数据一致。冗余字段应遵循：

1) 不是频繁修改的字段。

2) 不是 varchar 超长字段，更不能是 text 字段。

正例：用户姓名使用频率高，字段长度短，基本一成不变，可在相关联的表中冗余存储，避免关联查询。

第21条 【推荐】单表行数超过1000万行或者单表容量超过2GB，才推荐进行分库分表

如果预计三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。

第22条 【推荐】使用INT UNSIGNED存储IPV4

使用 INT UNSIGNED 而不是 char(15)来存储 ipv4 地址，可以节省空间，读写时 INET_ATON 和 INET_NTOA 函数进行转换。

第23条 【推荐】存储时间（精确到秒）建议使用TIMESTAMP类型

一般优先使用 TIMESTAMP 而不是 DATETIME，因为：

1) TIMESTAMP 使用 4 字节，而 DATETIME 需要 8 个字节。
2) TIMESTAMP 存储时间范围是 1970-01-01 08:00:01 到 2038-01-19 11:14:07，一般可以满足需求，否则请使用 DATETIME。
3) TIMESTAMP 可以自动更新（CURRENT_TIMESTAMP），不过从 5.6 版本开始，DATETIME 也支持了。

第24条 【推荐】建议字段定义为NOT NULL

1) 如果 null 字段被索引，需要额外的 1 字节；
2) null 使索引，索引统计，值的比较变得更复杂；
3) 可用 0，''代替；
4) 如果是索引字段，一定要定义为 not null。

第25条 【推荐】使用合适的存储长度

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。

1) 使用 UNSIGNED 存储非负整数，可以避免误存负数，且扩大了表示范围。
2) 可以预估存储整数的上下限时，使用可以满足存储要求的最短整数类型。tinyint,smallint,int,bigint4 种整数类型分别占用 1,2,4,8 个字节。

比如，比如存储范围是 0-200，那么请使用 TINYINT UNSIGNED。

第26条 【推荐】TIMESTAMP类型使用default值并由程序定义值

在 MySQL 5.6.6 之前，TIMESTAMP 列如果没有明确声明 NULL 属性，默认为 NOT NULL。（而其他数据类型，如果没有显示声明为 NOT NULL，则允许 NULL 值。）设置 TIMESTAMP 的列值为 NULL，会自动存储为当前 timestamp。建议由程序定义数值。

CREATE TABLE 语句中，第 1 个 TIMESTAMP 列可以用下面的任何一种方式声明：

1) 如果定义时 DEFAULT CURRENT_TIMESTAMP 和 ON UPDATE CURRENT_TIMESTAMP 子句都有，列值为默认使用当前的时间戳，并且自动更新。

2) 如果不使用 DEFAULT 或 ON UPDATE 子句，那么它等同于 DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP。

3) 如果只有 DEFAULT CURRENT_TIMESTAMP 子句，而没有 ON UPDATE 子句，列值默认

为当前时间戳但不自动更新。

4) 如果没用 DEFAULT 子句，但有 ON UPDATE CURRENT_TIMESTAMP 子句，列默认为 0 并自动更新。

5) 如果有一个常量值 DEFAULT，该列会有一个默认值，而且不会自动初始化为当前时间戳。如果该列还有一个 ON UPDATE CURRENT_TIMESTAMP 子句，这个时间戳会自动更新，否则该列有一个默认的常量但不会自动更新。

第 2 个 TIMESTAMP 列，如果没有声明为 NULL 或者 DEFAULT 子句，默认自动分配’0000-00-00 00:00:00’。插入行时没有指明改列的值，该列默认分配’0000-00-00 00:00:00’，且没有警告。

要关闭警告，需要加入下面的参数：

explicit_defaults_for_timestamp=1

第27条 【参考】拆分大字段和访问频率低的字段，分离冷热数据

2. 索引规约

为什么 MySQL 的性能依赖于索引？

MySQL 的查询速度依赖良好的索引设计，因此索引对于高性能至关重要。合理的索引会加快查询速度（包括 UPDATE 和 DELETE 的速度，MySQL 会将包含该行的 page 加载到内存中，然后进行 UPDATE 或者 DELETE 操作），不合理的索引会降低速度。

MySQL 索引查找类似于新华字典的拼音和部首查找，当拼音和部首索引不存在时，只能通过一页一页的翻页来查找。当 MySQL 查询不能使用索引时，MySQL 会进行全表扫描，会消耗大量的 IO。

第1条 【强制】索引命名规则

1) 索引命名使用全大写。
2) 包含多列的索引，每列都要在命名中提现，使用下划线(_)分隔。
3) 索引名长度不超过 30 字符。
4) 主键索引 PK_开头；唯一索引 UK_开头；普通索引 IDX_开头

第2条 【强制】单张表的索引数量控制在5个以内

InnoDB 的 secondary index 使用 b+tree 来存储，因此在 UPDATE、DELETE、INSERT 的时候需要对 b+tree 进行调整，过多的索引会减慢更新的速度。

第3条 【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。

第4条 【强制】表必须有主键

1) 有唯一键且由 3 个以下字段组成，并且字段都是整形时，使用唯一键作为主键。
2) 如果不满足条件 1，则使用自增（或者通过发号器获取）id 作为主键。

第5条 【强制】超过三个表禁止join

随着表数量的增加，join 的效率会急剧下降。

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。

第6条 【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。

第7条 【推荐】在varchar字段上建立索引时，建议指定索引长度

全字段索引可能降低效率，没必要对全字段建立索引，可以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。

第8条 【推荐】合理创建联合索引（避免冗余）

说明：(a,b,c) 相当于 (a) 、(a,b) 、(a,b,c)。

第9条 【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一种效果，用 explain 的结果，extra 列会出现：using index。

InnoDB 存储引擎中，secondary index（非主键索引）中没有直接存储行地址，存储主键值。如果用户需要查询 secondary index 中所不包含的数据列时，需要先通过 secondary index 查找到主键值，然后再通过主键查询到其他数据列，因此需要查询两次。

覆盖索引的概念就是查询可以通过在一个索引中完成，覆盖索引效率会比较高，主键查询是天然的覆盖索引。

合理的创建索引以及合理的使用查询语句，当使用到覆盖索引时可以获得性能提升。

比如 SELECT email,uid FROM user_email WHERE uid=xx，如果 uid 不是主键，适当时候可以将索引添加为 index(uid,email)，以获得性能提升。

第10条 【推荐】SQL性能优化的目标：至少要达到 range 级别

使用 EXPLAIN 判断 SQL 语句是否合理使用索引，至少要达到 range 级别，要求是 ref 级别，如果可以是 const 最好。尽量避免 extra 列出现：Using File Sort，Using Temporary。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，和全表扫描类似。

第11条 【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。

第12条 【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。

第13条 【参考】过长VARCHAR字段索引优化方法

如果 VARCHAR 索引长度过长时，效率会明细变差，如果较短索引的区分度不够（见第 7 条），那么可以添加 crc32 或者 MD5 Hash 字段，对该字段建立索引。

如下面的表增加一列 url_crc32，然后对 url_crc32 建立索引，减少索引字段的长度，提高效率。

```
CREATE TABLE url(
        ……
        url VARCHAR(255) NOT NULL DEFAULT 0,
        url_crc32 INT UNSIGNED NOT NULL DEFAULT 0,
        ……
        index idx_url(url_crc32)
)
```

第14条 【参考】创建索引时应避免的极端误解

1） 宁滥勿缺：误认为一个查询就需要建一个索引。

2） 宁缺勿滥：误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：误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。

3. SQL 规约

第1条 【强制】使用count(*)

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。

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

第2条 【强制】查询、插入时使用显式的字段名称

SELECT、INSERT 语句必须使用指明字段名称，不使用 SELECT *，不使用 INSERT INTO table()。说明：

4） 增加查询分析器解析成本；

5） 增加很多不必要的消耗（cpu、io、内存、网络带宽）；

6） 增加了使用覆盖索引的可能性；

7） 表结构修改时容易引起前后端不一致。

第3条 【强制】使用prepared statement

可以提供性能并且避免 SQL 注入。

第4条 【强制】UPDATE、DELETE语句不使用LIMIT

1） 可能导致主从数据不一致

2） 会记录到错误日志，导致日志占用大量空间

第5条 【强制】count(distinct col) 计算该列除NULL之外的不重复行数

注意 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。

第6条 【强制】使用ISNULL()来判断是否为NULL值

注意：NULL 与任何值的直接比较都为 NULL。 说明：

1） NULL<>NULL 的返回结果是 NULL，而不是 false。

2） NULL=NULL 的返回结果是 NULL，而不是 true。

3） NULL<>1 的返回结果是 NULL，而不是 true。

第7条 【强制】结果数为0的分页逻辑

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。

第8条 【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：（概念解释）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，则为级联更新。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。

第9条 【强制】sum(col)的空指针问题

当某一列的值全是 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;

第10条 【强制】WHERE条件中必须使用合适的类型

比如 id 类型是 string，那么不允许 where id=1 应该使用 where id='1'。说明：避免 MySQL 进行隐式类型转化，MySQL 进行隐式类型转化之后，可能会将索引字段类型转化成=号右边值的类型，导致使用不到索引。

第11条 【强制】当有主从表时，要先删除从表记录，再删除主表记录

第12条 【强制】select、update、delete必须有where条件

第13条 【推荐】Order By排序时，优先使用主键列，索引列

第14条 【推荐】多表关联查询优化

多表联合查询，优先使用 Where 条件，再作表关联，并且被关联的字段尽量添加索引。

第15条 【推荐】减少使用in，控制in后边的集合大小

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。

第16条 【推荐】避免使用存储过程、触发器、自定义函数

1） 容易将业务逻辑和 DB 耦合在一起，难以调试和扩展，移植性差；

2） MySQL 的存储过程、触发器、函数中存在一定的 bug。

第17条 【推荐】避免WHERE条件中的非等值条件

非等值条件（IN、BETWEEN、<、<=、>、>=）会导致后面的条件使用不了索引。

第18条 【推荐】利用延迟关联或者子查询等优化超多分页

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

假如有类似下面分页语句：

```
SELECT * FROM table ORDER BY TIME DESC LIMIT 10000,10;
```

这种分页方式会导致大量的 io，因为 MySQL 使用的是提前读取策略。

推荐分页方式：

SELECT * FROM table WHERE TIME<last_TIME ORDER BY TIME DESC LIMIT 10.

SELECT * FROM table inner JOIN(SELECT id FROM table ORDER BY TIME LIMIT 10000,10) as t USING(id)

第19条 【推荐】使用合理的SQL语句减少与数据库的交互次数

比如使用下面的语句来减少和 db 的交互次数：

4) INSERT ... ON DUPLICATE KEY UPDATE

5) REPLACE

6) INSERT IGNORE

第20条 【参考】避免使用复杂的SQL

避免使用复杂的 SQL，将大的 SQL 拆分成多条简单 SQL 分步执行。说明：简单的 SQL 容易使用到 MySQL 的 query cache；减少锁表时间特别是 MyISAM；可以使用多核 cpu。

4. 操作规约

第1条 【强制】禁止在线上做数据库压力测试

第2条 【强制】禁止从测试、开发环境直连线上数据库