建表需要注意什么?

- **使用自增 `id` 作为主键**
- **使用 `innodb` 引擎**
- **使用统一编码 `utf8mb4`**
- **尽量将所有列定义为 `NOT NULL`**

  索引 NULL 列需要额外空间来保存空还是非空，所以要占用更多的空间

  进行比较和计算时对于 NULL 值做特别的处理，可能使索引失效

  可以使用 0，特殊值，空字符串代替 NULL

- **所有表和字段都要添加注释，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**
- **存储业务数据的表，建议不要物理删除，添加字段 `is_deleted` 进行逻辑删除**
- **尽量避免在表中建立预留字段**

  预留字段的命名很难做到见名识义

  预留字段无法确认存储的数据类型

  使用很大的VARCHAR影响表性能

  对预留字段类型修改时，会对全表进行锁定

  修改字段成本远远大于增加一个字段

- **适当进行反范式化设计，便于查询和索引优化**

  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

- **尽量做到冷热数据分离，减小表的宽度**

  MySQL限制每个表最多存储4096列，每行大小不能超过65535个字节

  减少磁盘IO，保证热数据的内存缓存命中率

  更有效的利用缓存，避免使用SELECT * 这种方式读入无用的冷数据

  可以对表进行垂直拆分，将经常一起使用额列放到一个表中

- **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：如果预计三年后的数据量根本达不到这个级别，就不要在创建表时就分库分表

  500w并不是MySQL数据库的限制，MySQL并不会对单表数据量做限制，限制取决于存储设置和文件系统

  可以通过历史数据归档，分库分表等手段来控制

- **避免在数据库中存储图片、文件等二进制数据**

导致物理文件大，影响读取表数据时系统的IO效率

将图片、文件存储到文件服务器中，数据库中仅存储地址信息

如何命名?

- **库名与应用名保持一致**
- **常用业务字段团队内部需要使用统一命名，具体需参考团队规范**
- **表名和字段名，必须使用小写字母或数字，下划线分割，禁止出现数字开头，禁止两个下划线中间只出现数字见名知意，不可超过32字符**

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： `aliyun_admin` ， `rdc_config` ， `level3_name`

反例： `AliyunAdmin` ， `rdcConfig` ， `level_3_name`

- **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

正例： `system_user` ， `repay_order`

反例： `system_users` ， `repay_orders`

- **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）**
- **禁止使用 MySQL 保留字，如 `desc` 、 `range` 、 `match` 、 `delayed` 等（请参考MySQL 官方保留字）**
- **主键索引使用 pk_前缀；唯一索引使用 uk_前缀；普通索引使用 idx_前缀**
- **不同表存储相同数据的列（关联列）的列名和列类型必须完全一致**
- **临时库、表名必须以 `bak_` 为前缀，并以 `_yyyyMMdd` 实时日期为后缀。例如 `tmp_test01_20190409`**
- **备份库、表必须以 `bak_` 为前缀，并以 `_yyyyMMdd` 实时日期为后缀。例如 `bak_test01_20190409`**

如何选择列的类型?

- **字段可以使用多种数据类型时，优先考虑数字类型，其次为日期或二进制类型，最后是字符类型**

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

正例： `level TINYINT UNSIGNED NOT NULL DEFAULT 0 COMMENT '等级'`

反例： `level VARCHAR(2) NOT NULL DEFAULT '' COMMENT '等级'`

- **整数型选择能符合需求的最短列类型，如果为非负数，必须声明 `UNSIGNED`**

正例： `id bigint(20) NOT NULL AUTO_INCREMENT COMMENT '主键'`

反例： `id INT UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键'`

符合需求指可长期满足，不要因为节省长度影响正常业务

扩展：声明整数类型时，可不指定长度，整数类型的长度是固定的

int(3)中的3仅仅代表显示长度，不会限制存储空间

可搭配zerofill关键字进行零补全，但都是用于终端的显示，不影响实际存储的值

- **整数范围参考**

列类型	存储空间	取值范围 signed	取值范围 unsigned
tinyint	1字节	-128~127	0~255
smallint	2字节	-32768~32767	0~65535
mediumint	3字节	-8388608~8388607	0~16777215
int	4字节	-2147483648~-2147483647	0~4294967295
bigint	8字节	-9223372036854775808~9223372036854775807	0~18446744073709551615

- **实数类型使用 `DECIMAL`，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，得到错误的结果

  若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

  decimal 占用空间由定义的宽度决定 每4个字节可以存储9个数字，小数点占用一个

- **禁止使用字符串类型代替日期类型，日期占用空间小，便于查找过滤，有丰富的处理函数**
  - 日期范围参考

列类型	存储空间	格式	范围	备注
datetime	8字节	YYYY-MM-DD HH:MM:SS[.fraction]	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59	与时区无关 datetime(6)指定6位微秒
timestamp	4字节	YYYY-MM-DD HH:MM:SS[.fraction]	1970-01-01~ 2038-01-19	受时区影响 timestamp(6)指定6位微秒
date	3字节	YYYY-MM-DD	1000-01-01 ~ 9999-12-31	
time	3字节	HH:MM:SS[.fraction]	-838:59:59 ~ 838:59:59	存储时间点或持续时间 time(6)指定6位微秒

- **使用 `TINYINT` 代替 `ENUM`**

  ENUM本身是字符串类型但内部是由整数存储的，最多可以存储65535种不同的枚举值

  修改ENUM值需要使用ALTER语句，会导致锁表

  ORDER BY操作效率低，需要先转换为字符串才排序

  不要使用数值作为ENUM的枚举值，产生混淆

- **使用 `INT UNSIGNED` 存储 IPV4**

> 将字符串（15字节）转化为用数字（4字节）存储，节省空间
>
> 在存入和读取时使用函数进行转换

```
SELECT INET_ATON('255.255.255.255');
SELECT INET_NTOA(4294967295);
```

- **使用 `VARCHAR` 时 选择能符合需求的最小长度**

> VARCHAR(N) 中的 N 表示字符数不是字节数，不预先分配存储空间
>
> 符合需求指可长期满足，不要因为节省长度影响正常业务
>
> 长度不要超过 5000，如果存储长度大于此值，建议定义字段类型为 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，或将数据存入OSS

- **使用 VARBINARY 存储大小写敏感的变长字符串，VARBINARY 默认区分大小写，没有字符集概念，速度快**

如何建立索引?

建立索引的建议

1. 索引列的选择

- SELECT、UPDATE、DELETE语句的WHERE从句中的列建立索引，多个同时出现的列建立联合索引提高性能
- 包含在ORDER BY、GROUP BY、DISTINCT中的列建立索引，多个同时出现的列建立联合索引提高性能
- 多表JOIN的关联列上建立索引

2. 索引列顺序的选择

- 联合索引中，索引按照从左到右的顺序来使用的
- 把区分度最高的列放在联合索引的最左侧，区分度最高的就是主键和唯一索引的列
- 区分度相差不大时，选择字段长度小的放在左侧
- 区分度相差不大，且字段长度相差也不大时，将使用更频繁的列放在左侧

3. 避免进入如下误区

- 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。

索引建立规范

- **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
>
> 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

- **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**

> 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

- **每张表索引不要超过5个**

  索引并不是越多越好，索引可以提高效率也可以降低效率

  索引可以增加查询效率，但同样也会降低插入和更新的效率

  索引过多，也会增加查询优化器选择查询计划的时间，导致查询效率的降低

  禁止给表中每一列都建立单独的索引

- **避免建立冗余索引和重复索引**

  重复索引如：primary key(id)、index(id)、unique index(id)

  MySQL中的主键自动建立非空唯一索引

  冗余索引如：index(a,b,c)、index(a,b)、index(a)

- **对于频繁查询优先考虑使用覆盖索引**

  频繁查询如查询商品库存

  覆盖索引指包含了所有查询字段的索引

  不仅仅是WHERE从句GROUP BY从句中的列，也包含SELECT查询的列组合

  避免InnoDB表进行索引的二次查询

  可以把随机IP变为顺序IO加快查询效率

开发需要注意什么?

- **避免数据类型的隐式转换，会导致索引失效**

  反例：

  ```
  SELECT name FROM users WHERE id='111';
  ```

- **禁止使用 `SELECT *`**

  消耗更多的CPU和IP以及网络带宽资源

  无法使用覆盖索引

  可以减少表结构变更带来的影响

- **使用 `IN` 代替 `OR`**

  in的值不要超过500个

  in 操作可以有效的利用索引

- **不使用反向查询，如 `NOT IN` 和 `NOT LIKE`**

- **禁止使用 `ORDER BY RAND()` 进行随机排序**

  会把表中所有符合条件的数据装载到内存中进行排序

  消耗大量的CPU和IO及内存资源

  推荐在程序中获取一个随机值，然后根据随机值从数据库获取数据

- **禁止在 `WHERE` 从句中对列进行函数转换和计算**

  反例：

  ```
  ...WHERE DATE(gmt_create)='20180101';
  ```

  正例：

  ```
  ...WHERE gmt_create>='20180101' AND gmt_create<'20180102';
  ```

- **禁止 SQL 中存放业务逻辑**
- **使用用 `UNION ALL` 代替 `UNION`**

  UNION ALL 不需要对结果集再进行行排序

- **禁止在数据库中存储明文密码**
- **避免使用子查询，可以把子查询优化为 `JOIN` 操作**

  子查询的结果集无法使用索引

  子查询会产生临时表操作，如果子查询数据量大则严重影响效率

  消耗过多的CPU和IO资源

- **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

  每join一个表会多占用一部分内存(join_buffer_size控制)

  会产生临时表操作，影响查询效率

  MySQL最多允许关联61个表，建议不超过5个

- **超过 100w 行的批量写操作，要分批多次进行操作**

  大批量写操作可能会造成严重的主从延迟

  binlog日志为row格式时会产生大量的日志

  避免产生大事务操作，导致阻塞

- **减少与数据库的交互次数**

数据库更适合处理批量操作

合并多个相同操作到一起，可以提高处理效率

- **合理拆分复杂的大 SQL 为多个小 SQL**

MySQL中一个SQL只能只用一个CPU计算

SQL拆分后可以通过并行执行来提高处理效率

- **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。

- **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**
- **禁止在线上数据库做压力测试**
- **禁止从开发环境，测试环境直接连接生产环境数据库**

操作线上数据时需要注意什么?

- **禁止使用不含有字段列表的 `INSERT` 语句**
- **`UPDATE` 少量数据时，先使用 `SELECT` 将需要更改的数据查出，并在 `UPDATE` 语句的 `WHERE` 条件中添加主键限制**
- **`UPDATE` 大量数据时，先对数据进行备份**
- **禁止单条 SQL 语句同时更新多个表**
- **修改字段时，将多个 `ALERT` 语句合并为一个执行**
- **若需大批量插入或更新，执行语句很多，建议每隔三四百行添加 `SELECT SLEEP(1);` 语句**