健康160
Time Is Life
让健康更简单

数据库设计规范

运维中心/DBA – 张伟科

目录

CONTENTS

数据库设计规范

1、目的

为了规范数据库表设计，为了减少数据冗余，数据库更新、插入、删除异常，提高数据库的性能和稳定性，提高工作效率。

咱们公司业务平台数据库运行异常，约有**80%**是由于数据库设计不符合规范导致的。

2、字符规范

采用26个**英文字符**（区分大小写），“**0-9**”这10个自然数和下划线“_”，共63个字符组成，不能出现其他字符（注释除外）。

3、库设计规范

创建数据库时必须显式指定字符集，并且字符集只能是**utf8**或者**utf8mb4**。

创建数据库SQL举例：

(1)、create database db_name1 default character set utf8;

(2)、create database db_name1 default character set utf8mb4;

4、表设计规范

表设计规范1:

错误范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
)ENGINE = INNODB DEFAULT CHARSET = utf8;
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	1	表 'hdbs_user' 需要设置注释.	CREATE TABLE h...	0

正确范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
)ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网
站用户基本信息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	0		CREATE TABLE h...	0

4、表设计规范

表设计规范1小结：

【强制】表中必须用comment说明该表的功能和作用。

从一开始就进行数据字典的维护，避免后续在数据表的使用中，表的作用靠猜。

4、表设计规范

表设计规范2:

错误范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
)ENGINE = INNODB DEFAULT CHARSET = ASCII COMMENT = '网站
用户基本信息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	1	字符集(ascii)不在允许的范围之列!	CREATE TABLE h...	0

正确范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站
用户基本信息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	0		CREATE TABLE h...	0

4、表设计规范

表设计规范2小结：

【强制】创建表时必须显式指定字符集为utf8或utf8mb4。

数据库和表的字符集统一使用utf8，若是有字段需要存储emoji表情之类的，则将表或字段设置成utf8mb4；因为utf8号称万国码，其无需转码、无乱码风险且节省空间，而utf8mb4又向下兼容utf8。

4、表设计规范

表设计规范3:

错误范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
)ENGINE = MEMORY DEFAULT CHARSET = utf8 COMMENT = '网站用户
基本信息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	2	仅支持innodb存储引擎 (表'hdbs_user').	CREATE TABLE h...	0

正确范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
)ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户
基本信息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	0		CREATE TABLE h...	0

4、表设计规范

表设计规范3小结：

【强制】创建表时必须显式指定表存储引擎类型，如无特殊需求，一律为InnoDB。

MySQL 5.5之前默认使用Myisam，MySQL 5.6以后默认的为InnoDB，InnoDB支持事务，支持行级锁，更好的恢复性，高并发下性能也更好。

5、列设计规范

列设计规范1:

错误范例

```
CREATE TABLE hdbs_user (
  id VARCHAR (30) NOT NULL DEFAULT '' COMMENT '主键id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_review_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	1	不正确的表定义！主键必须设置为自增列	CREATE TABLE h...	0

正确范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_review_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	0		CREATE TABLE h...	0

5、列设计规范

列设计规范1小结：

【强制】强制要求主键为id，类型为int或bigint，且为auto_increment，初始值为1，主键必须使用无符号标志unsigned。

Innodb 是一种索引组织表，其数据存储的逻辑顺序和索引顺序是相同的；每张表可以有多个索引，但表的存储顺序只能有一种，Innodb 是按照主键索引的顺序来组织表的，因此不要使用更新频繁的列、UUID、MD5、HASH和字符串列作为主键，这些列无法保证数据的顺序增长，主键建议使用自增ID 值。

5、列设计规范

列设计规范2:

错误范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 ,
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_review_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	1	列 'user_id' 需要设置注释.	CREATE TABLE h...	0

正确范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_review_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	0		CREATE TABLE h...	0

5、列设计规范

列设计规范2小结：

【强制】必须用comment详细说明每个字段的意义。

从一开始就进行数据字典的维护，避免后续在数据表的使用中，字段的作用靠猜。

5、列设计规范

列设计规范3:

错误范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信
息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	1	列'username'需要设置默认值	CREATE TABLE h...	0

正确范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMM
ENT '真实姓名',
 create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信
息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	0		CREATE TABLE h...	0

5、列设计规范

列设计规范3小结：

【强制】字段必须为NOT NULL，且拥有默认值。

数据库所有为NULL 的列需要额外的空间来存储，因此会占用更多的空间；
数据库在进行比较和计算时需要对NULL 值做特别处理。

6、索引设计规范

索引设计规范1:

错误范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uni_user_id (user_id),
  KEY indx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信
息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	1	索引 'uni_user_id' 需要以'uniq_'为前缀(表'hdbs_user'). 索引 'indx_username' 需要以'idx_'为前缀(表'hdbs_user'). 列'username'需要设置默认值	CREATE TABLE h...	0

正确范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (create_time, user_revie
w_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信
息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	0		CREATE TABLE h...	0

6、索引设计规范

索引设计规范1小结：

【强制】普通索引名必须以idx_开头，唯一索引名必须以uniq_开头。

6、索引设计规范

索引设计规范2:

错误范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_cre_tim_id_use_id_use_upd (create
_time,user_review_status,id,user_id,userna
me,update_time)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信
息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	1	索引 'idx_cre_tim_id_use_id_use_upd'指定了太多的字段(表 'hdbs_user') 最多允许 5 个字段.	CREATE TABLE h...	0

正确范例

```
CREATE TABLE hdbs_user (
  id BIGINT (11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键
id',
  user_id BIGINT (11) NOT NULL DEFAULT 0 COMMENT '用户id',
  username VARCHAR (45) NOT NULL DEFAULT '' COMMENT '真实姓名',
  create_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户记录创建的时间',
  update_time datetime NOT NULL DEFAULT '1000-01-
01 00:00:00' COMMENT '用户资料修改的时间',
  user_review_status TINYINT NOT NULL DEFAULT 0 COMMENT '用户资料
审核状态，1:通过，2:审核中，3:未通过，4:还未提交',
  PRIMARY KEY (id),
  UNIQUE KEY uniq_user_id (user_id),
  KEY idx_username (username),
  KEY idx_create_time_user_review_status (
create_time, user_review_status)
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT = '网站用户基本信
息';
```

阶段	错误等级	错误信息	SQL语句	影响行数
审核完成	0		CREATE TABLE h...	0

6、索引设计规范

索引设计规范2小结：

【强制】单个索引指定字段不能超过5个

索引并不是越多越好！索引可以提高效率同样也可以降低效率；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，甚至有些情况下会降低查询效率，因为MySQL优化器在选择如何优化查询时，会根据统计信息，对每一个可以用到的索引来进行评估，以生成出一个最好的执行计划，如果同时有很多个索引都可以用于查询，就会增加MySQL优化器生成执行计划的时间，同样会降低查询性能。

7、SQL规范

【强制】大表常用查询，必须针对过滤条件中使用的字段创建索引，否则会导致全表扫描，数据库负载飙升。

【强制】类型转换不能使用索引：where条件里等号左右字段类型必须一致，否则无法利用索引。

【强制】过滤条件使用like前缀模糊匹配无法使用索引，会导致全表扫描。

【强制】索引在计算列里无法使用：where条件里索引列不要使用函数或表达式，否则无法利用索引。

目录

CONTENTS

经典案例

1、经典案例一

经典案例一：大表常用查询，必须针对过滤条件中使用的字段创建索引，否则会导致全表扫描，数据库负载飙升。

真实案例：见【 TiDB3数据库问题跟进处理群】2022年11月9日16:25业务数据库TiDB3**高负载**问题。

演示案例：查询医生表电话号码为13828836993的医生信息。
说明：表doctor字段phone未创建了索引，如下图：

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	字段	索引类型	索引方法	注释
idx_unit_id_dep_id_phone_	`unit_id`, `dep_id`, `phone`, `doctor_name`	NORMAL	BTREE	
idx_unit_id	`unit_id`	NORMAL	BTREE	

SQL：`select doctor_name,sex,phone from doctor where phone='13828836993';`
正确的方法：对常用的查询SQL过滤条件中使用的字段创建索引，加速查询效率。

1、经典案例一

真实案例截图:

1、经典案例一

演示效果:

```
mysql> explain select doctor_name,sex,phone from doctor where phone='13828836993'; 创建索引前SQL执行计划
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows   | Extra       |
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
|  1 | SIMPLE      | doctor | ALL  | NULL          | NULL | NULL    | NULL | 479790 | Using where |
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
1 row in set (0.00 sec)

mysql> select doctor_name,sex,phone from doctor where phone='13828836993';  创建索引前SQL执行情况
+-------------+------+-------------+
| doctor_name | sex  | phone       |
+-------------+------+-------------+
| 陈小岚      |    0 | 13828836993 |
+-------------+------+-------------+
1 row in set (0.68 sec)

mysql> ALTER TABLE `doctor` ADD INDEX `idx_phone` (`phone`) USING BTREE ; 创建索引
Query OK, 0 rows affected (3.67 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain select doctor_name,sex,phone from doctor where phone='13828836993'; 创建索引后SQL执行计划
+----+-------------+--------+------+---------------+-----------+---------+-------+------+-----------------------+
| id | select_type | table  | type | possible_keys | key       | key_len | ref   | rows | Extra                 |
+----+-------------+--------+------+---------------+-----------+---------+-------+------+-----------------------+
|  1 | SIMPLE      | doctor | ref  | idx_phone     | idx_phone | 63      | const |    1 | Using index condition |
+----+-------------+--------+------+---------------+-----------+---------+-------+------+-----------------------+
1 row in set (0.00 sec)

mysql> select doctor_name,sex,phone from doctor where phone='13828836993'; 创建索引后SQL执行情况
+-------------+------+-------------+
| doctor_name | sex  | phone       |
+-------------+------+-------------+
| 陈小岚      |    0 | 13828836993 |
+-------------+------+-------------+
1 row in set (0.00 sec)
```

2、经典案例二

经典案例二：类型转换不能使用索引，where条件里等号左右字段类型必须一致，否则无法利用索引。

真实案例：见【平台安全可用骨干群】10月11日10:20诊中业务数据库**拥堵**问题

演示案例：查询医生表电话号码为13828836993的医生信息。

说明：表doctor字段phone的类型为varchar，并且对此字段创建了索引。

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	字段	索引类型	索引方法	注释
idx_unit_id_dep_id_phone_	`unit_id`, `dep_id`, `phone`, `doctor_name`	NORMAL	BTREE	
idx_unit_id	`unit_id`	NORMAL	BTREE	
idx_phone	`phone`	NORMAL	BTREE	

phone=13828836993;

SQL2（正确）：select doctor_name,sex,phone from doctor where phone='13828836993';

2、经典案例二

真实案例截图：

2、经典案例二

演示效果:

```
mysql> explain select doctor_name,sex,phone from doctor where phone=13828836993;  SQL1执行计划
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows   | Extra       |
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
|  1 | SIMPLE      | doctor | ALL  | idx_phone     | NULL | NULL    | NULL | 479790 | Using where |
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
1 row in set (0.01 sec)

mysql> explain select doctor_name,sex,phone from doctor where phone='13828836993';  SQL2执行计划
+----+-------------+--------+------+---------------+-----------+---------+-------+------+-----------------------+
| id | select_type | table  | type | possible_keys | key       | key_len | ref   | rows | Extra                 |
+----+-------------+--------+------+---------------+-----------+---------+-------+------+-----------------------+
|  1 | SIMPLE      | doctor | ref  | idx_phone     | idx_phone | 63      | const |    1 | Using index condition |
+----+-------------+--------+------+---------------+-----------+---------+-------+------+-----------------------+
1 row in set (0.00 sec)

mysql> select doctor_name,sex,phone from doctor where phone=13828836993;  SQL1执行情况
+-------------+------+-------------+
| doctor_name | sex  | phone       |
+-------------+------+-------------+
| 陈小岚      |    0 | 13828836993 |
+-------------+------+-------------+
1 row in set, 49793 warnings (0.56 sec)

mysql> select doctor_name,sex,phone from doctor where phone='13828836993';  SQL2执行情况
+-------------+------+-------------+
| doctor_name | sex  | phone       |
+-------------+------+-------------+
| 陈小岚      |    0 | 13828836993 |
+-------------+------+-------------+
1 row in set (0.00 sec)
```

3、经典案例三

经典案例三：过滤条件使用like前缀模糊匹配无法使用索引，会导致全表扫描。

真实案例：见【数据库相关事宜沟通群】2021年8月4日15:17医院基础资料业务**慢查询**SQL问题。

演示案例：查询医生表电话号码包含28837的医生信息。

说明：表doctor字段phone的类型为varchar，并且对此字段创建了索引。

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	字段	索引类型	索引方法	注释
idx_unit_id_dep_id_phone_	`unit_id`, `dep_id`, `phone`, `doctor_name`	NORMAL	BTREE	
idx_unit_id	`unit_id`	NORMAL	BTREE	
idx_phone	`phone`	NORMAL	BTREE	

SQL：`select doctor_name,sex,phone from doctor where phone like '%28837%';`

正确的方法：可以考虑使用ES等服务，提供全文检索功能。

3、经典案例三

真实案例截图：

数据库相关事宜沟通群

2021年8月4日 15:17

2021-8-4 15:17:19

以下慢查询SQL无法使用索引（doctor_name），因为过滤条件使用了前缀模糊匹配条件，会导致全表扫描，烦请改造SQL。

```
SELECT
    id id,
    doctor_name doctorName,
    doc_spell docSpell,
    sex sex,
    image image,
    card card,
    city_id cityId,
    edu_level eduLevel,
```

回到最新位置

数据库相关事宜沟通群

```
AND doctor_name LIKE CONCAT(
    CONCAT('%', '郑发德'),
    '%'
)
LIMIT 0,
 10;
```

```
EXPLAIN SELECT
  id id,
  doctor_name doctorName,
  doc_spell docSpell,
  sex sex,
  image image,
  card card,
  city_id cityId,
  edu_level eduLevel,
  zcid zcid,
  aca_zc acazc,
  union_id unionId,
  phone phone,
  mobile mobile,
  email email,
  url url,
  expert expert,
  is_expert isExpert,
  is_ask isAsk,
  is_vip isVip,
```

信息 | 结果1 | 概况 | 状态

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	platform_	ALL	(Null)	(Null)	(Null)	(Null)	662728	Using whe

回到最新位置

3、经典案例三

演示效果:

```
mysql> explain select doctor_name,sex,phone from doctor where phone like '%28837%';
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows   | Extra       |
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
|  1 | SIMPLE      | doctor | ALL  | NULL          | NULL | NULL    | NULL | 479790 | Using where |
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
1 row in set (0.00 sec)

mysql> select doctor_name,sex,phone from doctor where phone like '%28837%';
+-------------+------+-------------+
| doctor_name | sex  | phone       |
+-------------+------+-------------+
| 李明静      |    1 | 13728837210 |
| 吴治建      |    0 | 13508288372 |
+-------------+------+-------------+
2 rows in set (0.65 sec)
```

SQL执行计划

SQL执行情况

4、经典案例四

经典案例四：索引在计算列里无法使用，where条件里索引列不要使用函数或表达式，否则无法利用索引。

真实案例：见【数据库相关事宜沟通群】2021年7月19日16:16活动业务出现的**慢查询**SQL问题

演示案例：查询医生表电话号码以1858585开头的医生信息。

说明：表doctor字段phone的类型为varchar，并且对此字段创建了索引。

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	字段	索引类型	索引方法	注释
idx_unit_id_dep_id_phone_	`unit_id`, `dep_id`, `phone`, `doctor_name`	NORMAL	BTREE	
idx_unit_id	`unit_id`	NORMAL	BTREE	
idx_phone	`phone`	NORMAL	BTREE	

SQL1（错误）：`select doctor_name,sex,phone from doctor where substr(phone,1 ,7)='1858585';`

SQL2（正确）：`select doctor_name,sex,phone from doctor where phone like '1858585%';`

4、经典案例四

真实案例截图：

4、经典案例四

演示效果:

```
mysql> explain select doctor_name,sex,phone from doctor where substr(phone,1 ,7)='1858585'; SQL1执行计划
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows   | Extra       |
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
|  1 | SIMPLE      | doctor | ALL  | NULL          | NULL | NULL    | NULL | 479790 | Using where |
+----+-------------+--------+------+---------------+------+---------+------+--------+-------------+
1 row in set (0.00 sec)

mysql> explain select doctor_name,sex,phone from doctor where phone like '1858585%'; SQL2执行计划
+----+-------------+--------+-------+---------------+-----------+---------+------+------+-----------------------+
| id | select_type | table  | type  | possible_keys | key       | key_len | ref  | rows | Extra                 |
+----+-------------+--------+-------+---------------+-----------+---------+------+------+-----------------------+
|  1 | SIMPLE      | doctor | range | idx_phone     | idx_phone | 63      | NULL |    1 | Using index condition |
+----+-------------+--------+-------+---------------+-----------+---------+------+------+-----------------------+
1 row in set (0.02 sec)

mysql> select doctor_name,sex,phone from doctor where substr(phone,1 ,7)='1858585'; SQL1执行情况
+-------------+------+-------------+
| doctor_name | sex  | phone       |
+-------------+------+-------------+
| 许靖        |    0 | 18585855127 |
+-------------+------+-------------+
1 row in set (1.17 sec)

mysql> select doctor_name,sex,phone from doctor where phone like '1858585%';  SQL2执行情况
+-------------+------+-------------+
| doctor_name | sex  | phone       |
+-------------+------+-------------+
| 许靖        |    0 | 18585855127 |
+-------------+------+-------------+
1 row in set (0.00 sec)
```

5、经典案例五

真实案例：诊中业务支付表**主键**使用**varchar**类型（如下图），导致变更表结构时数据库**卡死**，数据库的读写性能也越来越低。

字段 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	类型	长度	小数点	不是 null	键	注释
id	varchar	255		☑	1	id
unit_id	varchar	40		☐		医院Id
yuyue_id	varchar	40		☐		预约Id
phone	varchar	50		☐		手机号
healthy_no	varchar	300		☐		健康卡号
card_no	varchar	40		☐		诊疗卡号
social_no	varchar	40		☐		社保卡号
order_no	varchar	40		☐		订单号
his_pay_no	varchar	300		☐		his订单号
pres_no	varchar	50		☐		处方编号

真实案例：见2021年11月9日诊中业务数据库故障问题

5、经典案例五

真实案例截图：

事件时间	故障时长	故障描述	发现者	故障分类	故障分析	数据库类型	是否解决	责任人
2021-11-09	13min	诊中数据库堵塞时长约13分钟左右	监控	运维	使用pt-schema-online-change变更诊中业务数据库大表结构时，因业务流量出现抖动，导致数据库堵塞时长约13分钟左右	mysql	是	张伟科 杨刚

发件人： zhangwk@91160.com
发送时间： 2021-11-26 16:27
收件人：
抄送：
主题： 【重要紧急】诊中业务数据库deps优化方案

Dear All:

为了诊中业务数据库能长期保持健壮、稳定、高效的运行，也便于后续变更表结构更加方便高效，梳理出以下紧急重要的事项需要开发同学配合优化处理：

1、为以下大表增加自增主键id字段：

（1）、dep_pay_info（记录数：311485414）

（2）、dep_pay_info_detail（记录数：25850358）

（3）、dep_pay_record（记录数：23027010）

备注：**如果不增加的话：后续数据库的读写性能会越来越低，甚至长期堵塞，造成业务不可用；后续将没办法变更表结构，因为变更表结构将会导致数据库卡死。**

附件1

数据库设计规范详细版，详见：
https://confluence.rd.91160.com/pages/editpage.action?pageId=22447174

91160-平台研发 / ... / 基础规范

数据库设计规范

Edit Watch Share ...

Created by 欧阳斌, last modified by 张伟科 less than a minute ago

第1章 文档目的

本文档为了规范数据库表设计，为了减少数据冗余，数据库更新、插入、删除异常。提高数据库的性能和稳定性，提高工作效率，

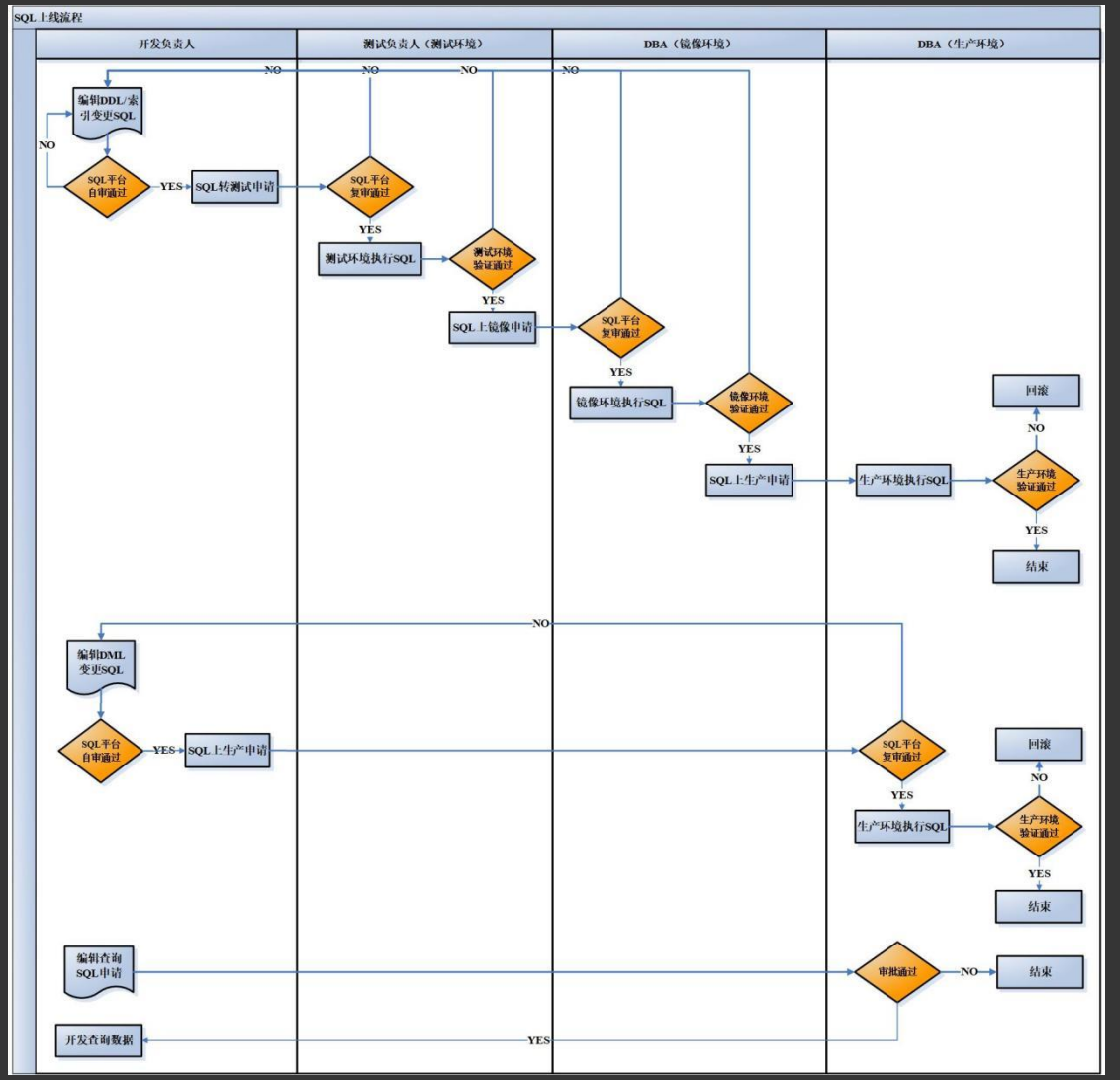
附件2

1、数据库变更流程：生产环境必须走**邮件审批**（开发由**开发负责人审批**；测试由**黄礼/张亮审批**）流程

2、SQL审核平台地址：

本地环境：**http://yearninglocal.91160.com/**

生产环境：**http://yearning.91160.com/**



目录

CONTENTS

Q&A

THANKS

健康160 Time Is Life
让健康更简单