

去哪儿MySQL开发规范

DBATEAM
2014-03-03

- 命名规范
- 基础规范
- 库表设计
- 字段设计
- 索引规范
- SQL设计
- 行为规范

命名规范

- 命名规范
- ✓ 库名、表名、字段名必须使用小写字母，并采用下划线分割
- ✓ 库名、表名、字段名禁止超过32个字符。须见名之意
- ✓ 库名、表名、字段名禁止使用MySQL保留字
- ✓ 临时库、表名必须以tmp为前缀，并以日期为后缀
- ✓ 备份库、表必须以bak为前缀，并以日期为后缀

```
create table TTT (`insert` int(10) not null ...)  
create table abc_1202 ...  
alter table t add index idx_uid_mid_time(uid,mid,time)  
alter table t add index idx_uid(uid,mid,time)  
tmp_test01_0704  
bak_test01_20130704
```

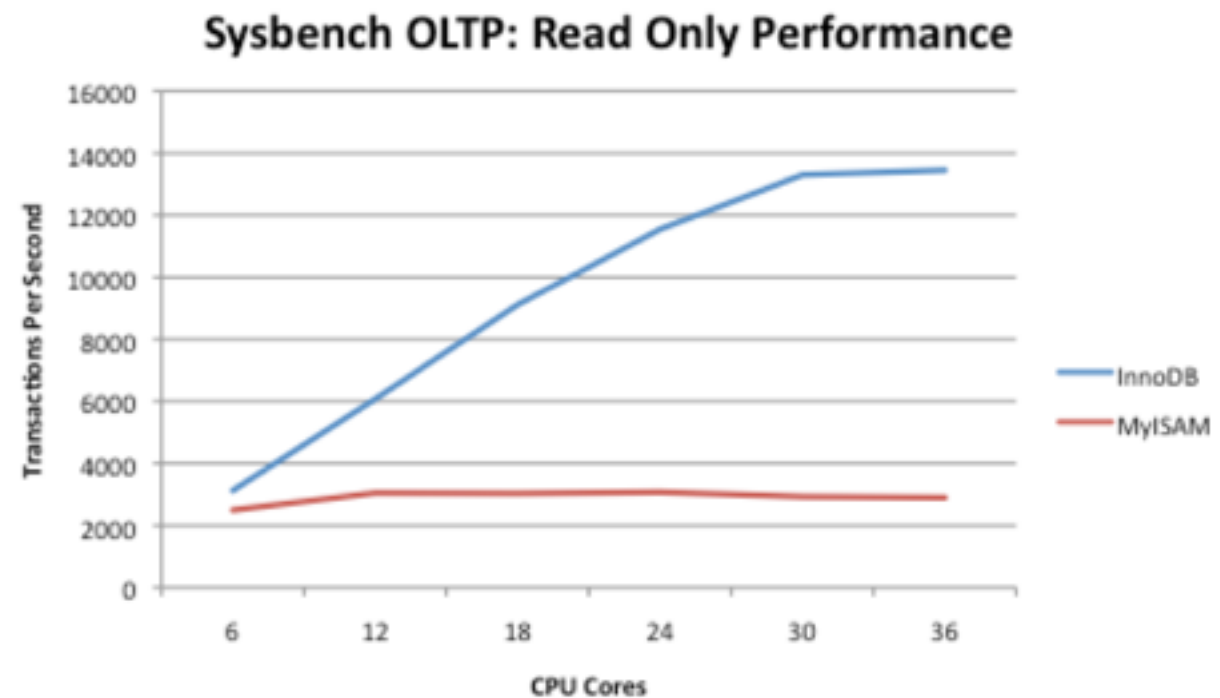
命名规范

- 命名规范
- ✓ 库名、表名、字段名必须使用小写字母，并采用下划线分割
- ✓ 库名、表名、字段名禁止超过32个字符。须见名之意
- ✓ 库名、表名、字段名禁止使用MySQL保留字
- ✓ 临时库、表名必须以tmp为前缀，并以日期为后缀
- ✓ 备份库、表必须以bak为前缀，并以日期为后缀

```
create table TTT (`insert` int(10) not null ...)  
create table abc_1202 ...  
alter table t add index idx_uid_mid_time(uid,mid,time)  
alter table t add index idx_uid(uid,mid,time)  
tmp_test01_0704  
bak_test01_20130704
```

基础规范

- 基础规范
- ✓ 使用INNODB存储引擎
- ✓ 表字符集使用UTF8
- ✓ 所有表都需要添加注释
- ✓ 单表数据量建议控制在5000W以内
- ✓ 不在数据库中存储图片、文件等大数据
- ✓ 禁止在线上做数据库压力测试
- ✓ 禁止从测试、开发环境直连数据库



Feature	InnoDB	MyISAM
ACID Transactions	Yes	No
Configurable ACID Properties	Yes	No
Crash Safe	Yes	No
Foreign Key Support	Yes	No
Row-Level Locking Granularity	Yes	No (Table)
MVCC	Yes	No

库表设计

● 库表设计

- ✓ 禁止使用分区表
- ✓ 拆分大字段和访问频率低的字段，分离冷热数据
- ✓ 用HASH进行散表，表名后缀使用十进制数，下标从0开始
- ✓ 按日期时间分表需符合YYYY[MM][DD][HH]格式
- ✓ 采用合适的分库分表策略。例如千库十表、十库百表等

comment_20120815
comment_20120816
~~comment_120817~~
~~user_39~~
~~user_3A~~
~~user_3B~~
~~user_3C~~

```
CREATE TABLE `blog` (  
  `blog_id` int(11) NOT NULL AUTO_INCREMENT,  
  `author_id` int(11) NOT NULL,  
  `create_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  `rank` int(11) DEFAULT '0',  
  `category` varchar(20) DEFAULT NULL,  
  `abstract` varchar(100) DEFAULT NULL,  
  `tags` varchar(100) NOT NULL DEFAULT '',  
  `body` text,  
  PRIMARY KEY (`blog_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```


字段设计

- 尽可能不使用TEXT、BLOB类型
- 用DECIMAL代替FLOAT和DOUBLE存储精确浮点数
- Simple is good
 - ✓ 将字符转化为数字
 - ✓ 使用TINYINT来代替ENUM类型
- Generosity can be unwise
 - ✓ 存储 “hello” 时VARCHAR(5) VS VARCHAR(200)

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
' '	' '	4 bytes	' '	1 byte
' ab '	' ab '	4 bytes	' ab '	3 bytes
' abcd '	' abcd '	4 bytes	' abcd '	5 bytes
' abcdefgh '	' abcd '	4 bytes	' abcd '	5 bytes

The best strategy is to allocate only as much space as you really need.

字段设计

- 尽可能不使用TEXT、BLOB类型
- 用DECIMAL代替FLOAT和DOUBLE存储精确浮点数
- Simple is good
 - ✓ 将字符转化为数字
 - ✓ 使用TINYINT来代替ENUM类型
- Generosity can be unwise
 - ✓ 存储 “hello” 时VARCHAR(5) VS VARCHAR(200)

ENUM('Mercury', 'Venus', 'Earth')

Value	Index
NULL	NULL
' '	0
'Mercury'	1
'Venus'	2
'Earth'	3

numbers ENUM('0', '1', '2')

```
mysql> INSERT INTO t (numbers) VALUES (2), ('2'), ('3');  
mysql> SELECT * FROM t;
```

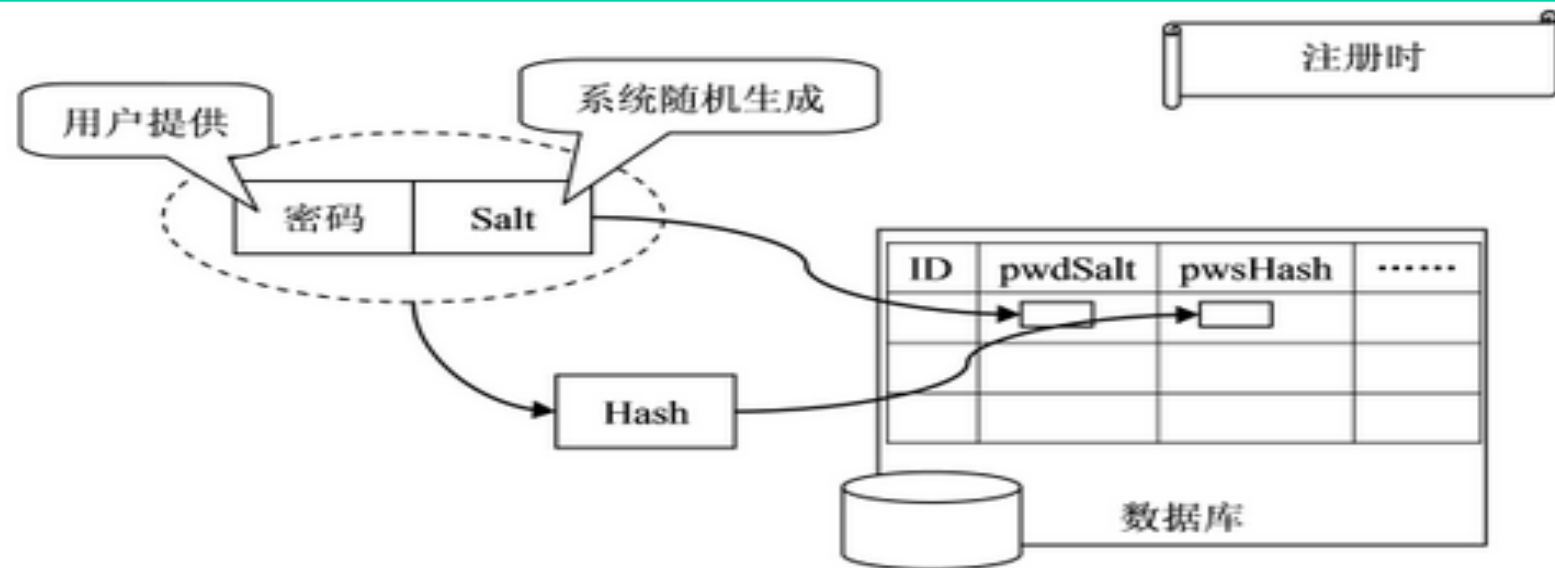
```
+-----+  
| numbers |  
+-----+  
| 1       |  
| 2       |  
| 2       |  
+-----+
```

- ❑ 存储index，而不是字符串
- ❑ 枚举值改变会导致DDL

字段设计

- Avoid null if possible
 - ✓ 所有字段均定义为NOT NULL
- Smaller is usually better
 - ✓ 使用UNSIGNED存储非负整数
 - ✓ INT类型固定占用4字节存储
 - ✓ 使用timestamp存储时间
 - ✓ 使用INT UNSIGNED存储IPV4
 - ✓ 使用VARBINARY存储大小写敏感的变长字符串
 - ✓ 禁止在数据库中存储明文密码

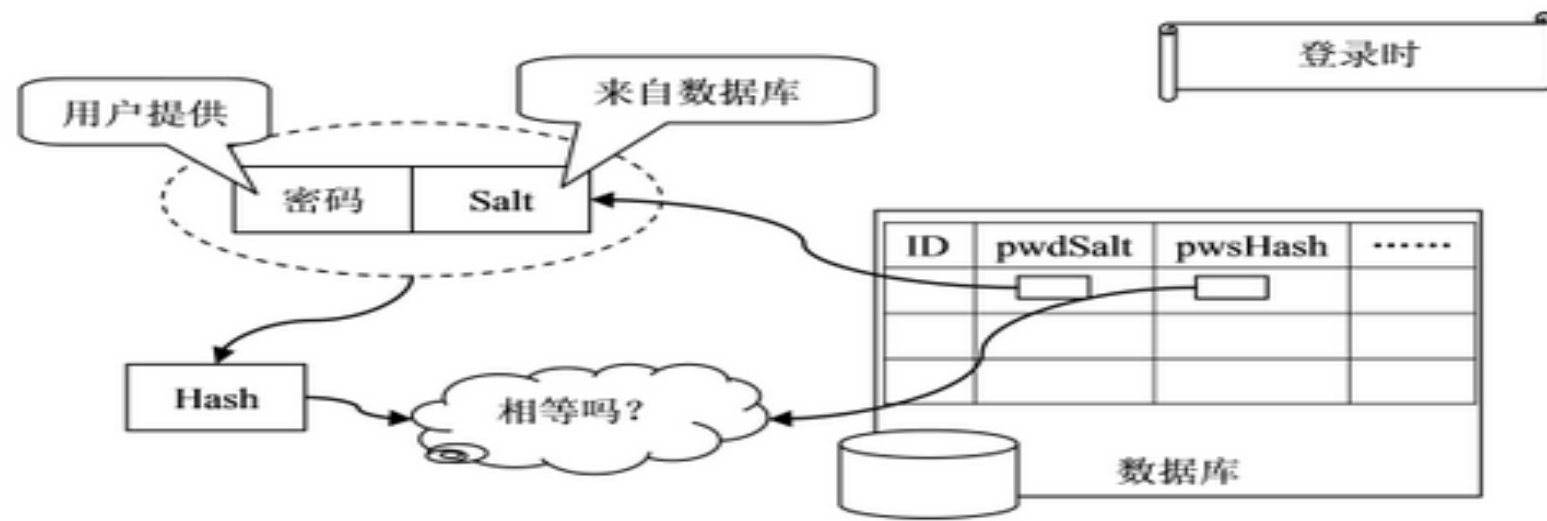
thunder.huang		*7382FCB230650
yinggang.zhao		*9259A57894430
yanwei.zhou		*13AB54B0442CC
dbcenter		*BF8126E8BB5AA



字段设计

- Avoid null if possible
 - ✓ 所有字段均定义为NOT NULL
- Smaller is usually better
 - ✓ 使用UNSIGNED存储非负整数
 - ✓ INT类型固定占用4字节存储
 - ✓ 使用timestamp存储时间
 - ✓ 使用INT UNSIGNED存储IPV4
 - ✓ 使用VARBINARY存储大小写敏感的变长字符串
 - ✓ 禁止在数据库中存储明文密码

thunder.huang		*7382FCB230650
yinggang.zhao		*9259A57894430
yanwei.zhou		*13AB54B0442CC
dbcenter		*BF8126E8BB5AA



索引规范

● 索引的用途

- ✓ 去重
- ✓ 加速定位
- ✓ 避免排序
- ✓ 覆盖索引

● 索引数量控制

- ✓ 单张表中索引数量不超过5个
- ✓ 单个索引中的字段数不超过5个
- ✓ 对字符串使用前缀索引，前缀索引长度不超过8个字符
- ✓ 建议优先考虑前缀索引，必要时可添加伪列并建立索引

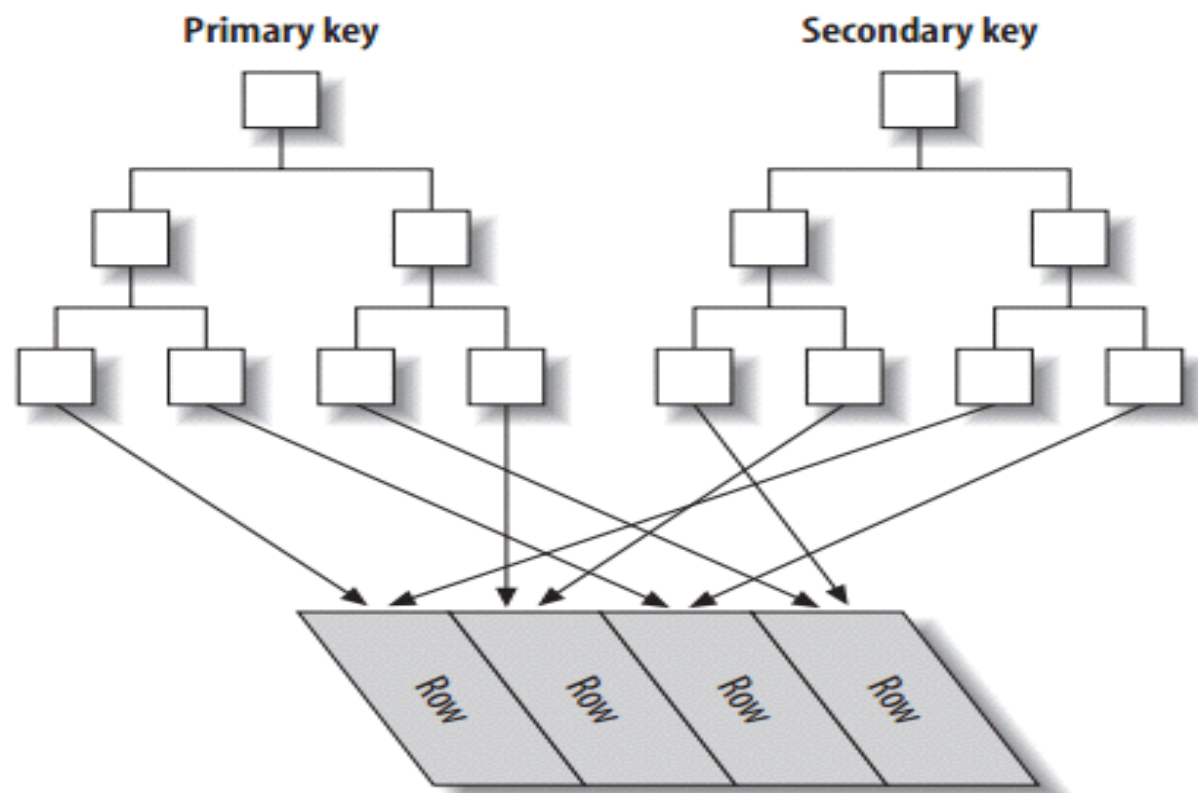
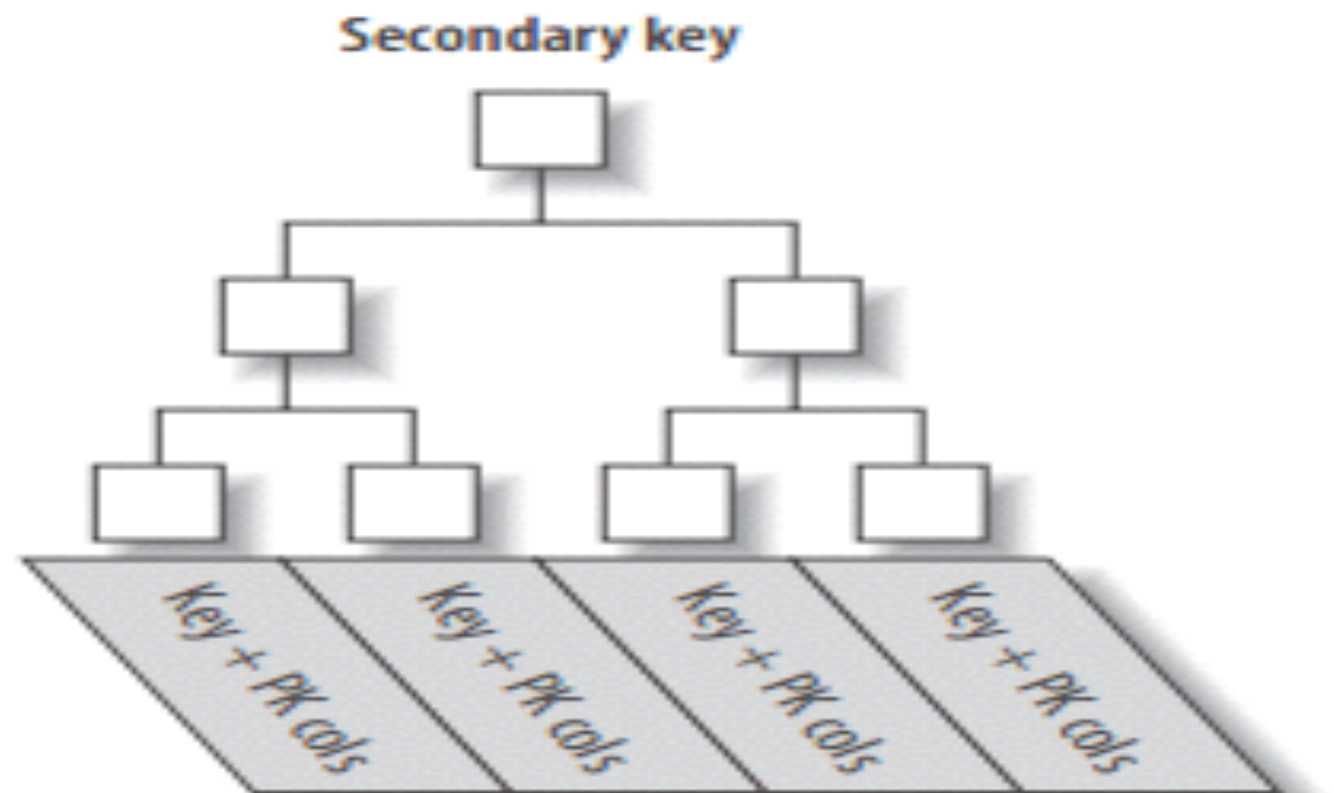
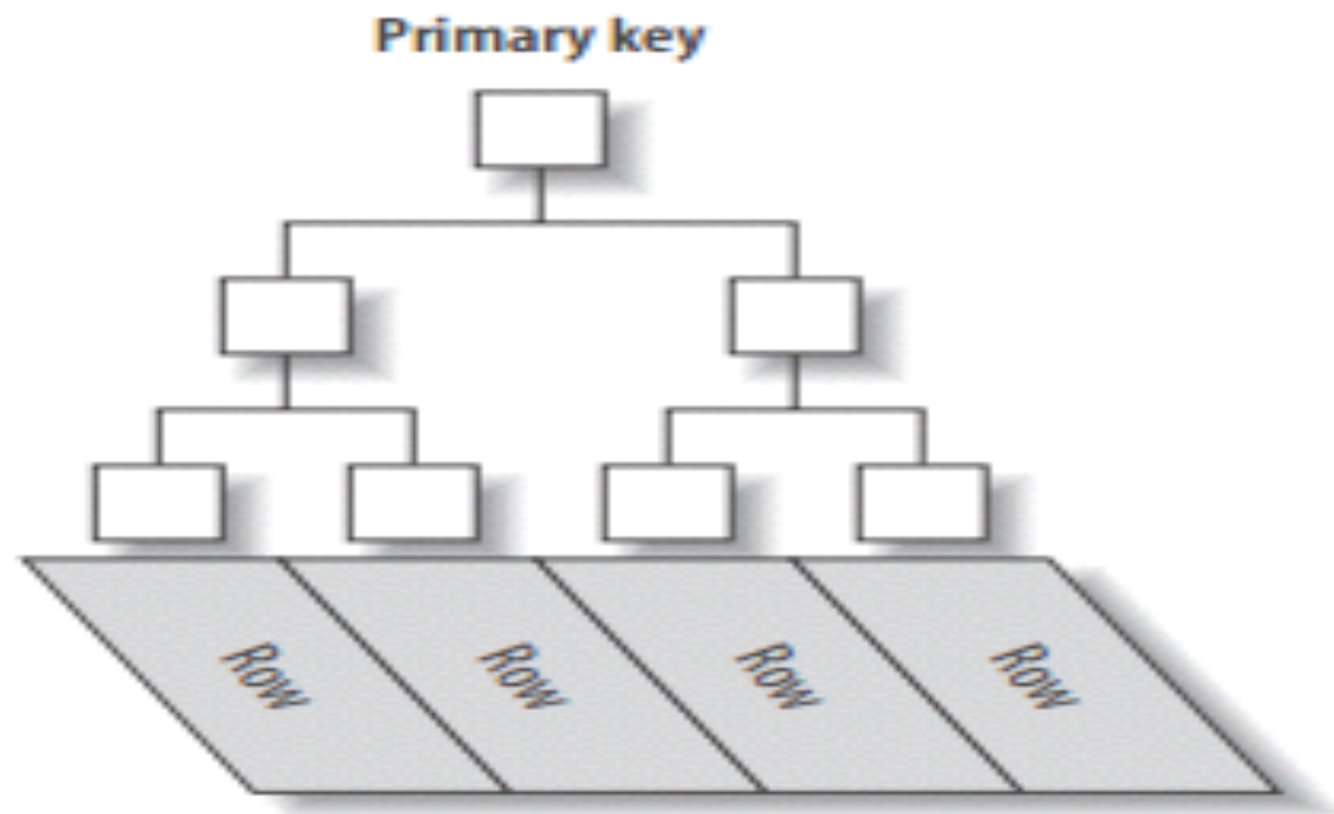
```
mysql> SELECT COUNT(DISTINCT LEFT(city, 3))/COUNT(*) AS sel3,  
-> COUNT(DISTINCT LEFT(city, 4))/COUNT(*) AS sel4,  
-> COUNT(DISTINCT LEFT(city, 5))/COUNT(*) AS sel5,  
-> COUNT(DISTINCT LEFT(city, 6))/COUNT(*) AS sel6,  
-> COUNT(DISTINCT LEFT(city, 7))/COUNT(*) AS sel7  
-> FROM sakila.city_demo;
```

sel3	sel4	sel5	sel6	sel7
0.0239	0.0293	0.0305	0.0309	0.0310

- ✓ Ø单字母区分度: 26
- ✓ Ø4字母区分度: $26*26*26*26=456,976$
- ✓ Ø5字母区分度: $26*26*26*26*26=11,881,376$
- ✓ Ø6字母区分度: $26*26*26*26*26*26=308,915,776$

索引规范

- 主键准则
- ✓ 表必须有主键
- ✓ 不使用更新频繁的列
- ✓ 尽量不选择字符串列
- ✓ 不使用UUID MD5 HASH
- ✓ 默认使用非空的唯一键
- ✓ 建议选择自增或发号器



索引规范

- 重要的SQL必须被索引
 - ✓UPDATE、DELETE语句的WHERE条件列
 - ✓ORDER BY、GROUP BY、DISTINCT的字段
 - ✓多表JOIN的字段
- 区分度最大的字段放在前面
- 核心SQL优先考虑覆盖索引
- 避免冗余和重复索引
- 索引不是越多越好
 - ✓综合评估数据密度和分布
 - ✓考虑查询和更新比例

```
CREATE TABLE test (  
ID INT NOT NULL  
PRIMARY KEY,  
UNIQUE(ID),  
INDEX(ID)  
);
```

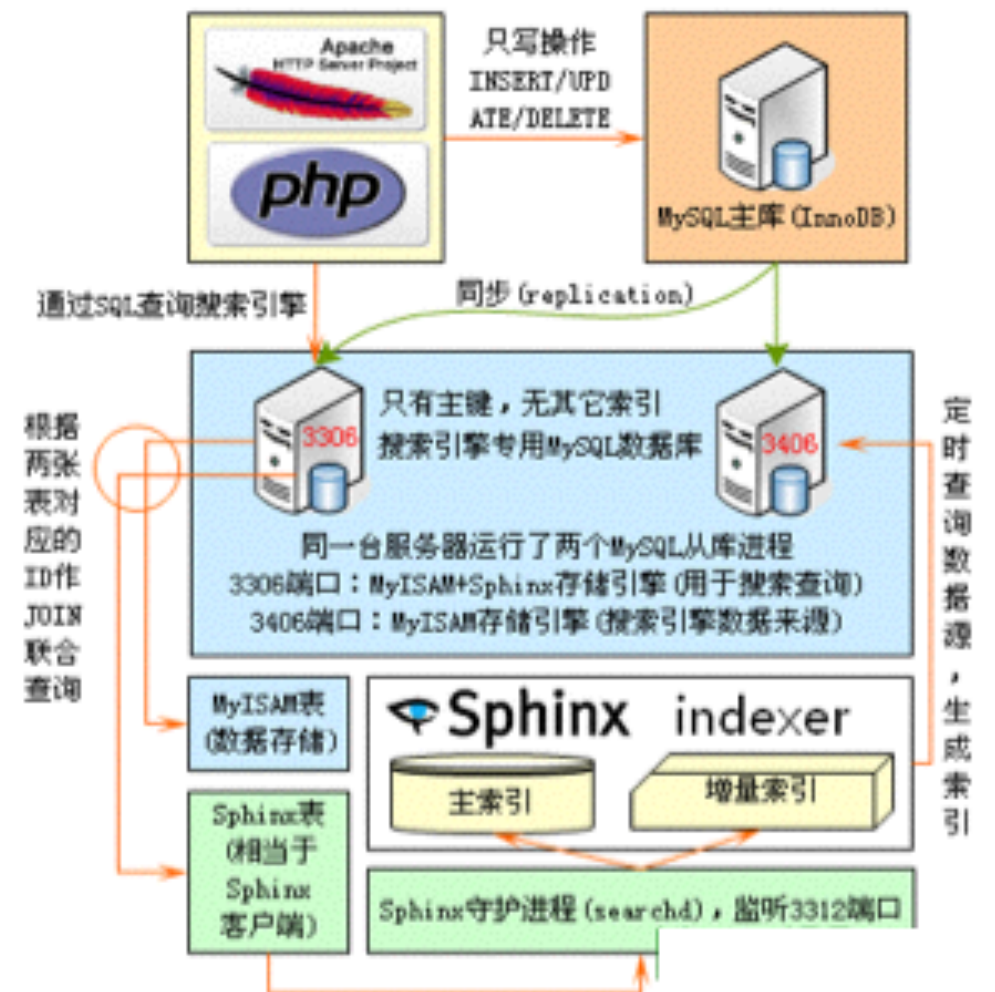
idx_a_b_c(a,b,c)	select a,b,c from test where a=123
idx_a(a)	select * from test where a=123
idx_a_b(a,b)	select a,b,c from test where d=456

索引是一把双刃剑：降低插入和更新速度，占用磁盘空间

索引规范

- 索引禁忌
 - ✓不在低基数列上建立索引，例如“性别”
 - ✓不在索引列进行数学运算和函数运算
- 尽量不使用外键
 - ✓外键用来保护参照完整性，可在业务端实现
 - ✓对父表和子表的操作会相互影响，降低可用性
 - ✓INNODB本身对online DDL的限制
- 不使用%前导的查询，如like “%ab”
- 不使用负向查询，如not in/like
 - ✓无法使用索引，导致全表扫描
 - ✓全表扫描导致buffer pool利用率降低

PHP+MySQL+Sphinx 搜索引擎架构图



我要找: 女朋友 年龄: 22 到 28 岁 地区: 北京 ☒ 有照片

身高: 不限 到 不限 厘米 学历: 不限 ☐ 以上

住房: 不限 月薪: 不限 ☐ 以上

婚史: 不限 购车: 不限

尺有所短，寸有所长，换一种工具试试！

●使用预编译语句

✓只传参数，比传递SQL语句更高效

✓一次解析，多次使用

✓降低SQL注入概率

●避免隐式转换

✓会导致索引失效

●充分利用前缀索引

✓必须是最左前缀

✓不可能同时用到两个范围条件

字段: `remark` varchar(50) NOT NULL
SQL1: SELECT `id`, `gift_code` FROM gift
WHERE
`deal_id` = 640 AND remark=115127;
1 row in set (0.14 sec)

SQL2: SELECT `id`, `gift_code` FROM
pool_gift WHERE
`deal_id` = 640 AND remark='115129';
1 row in set (0.005 sec)

idx_coll_col2_col3(coll,col2,col3)

select * from tbl_name where coll=val1 and clo2=val2;

~~select * from tbl_name where col2=val2 and col3=val3;~~

~~select * from tbl_name where coll=100 and col2 between 1000 and 2000 and col3 > 3000~~

- 避免使用存储过程、触发器、UDF、events等
 - ✓让数据库做最擅长的事
 - ✓降低业务耦合度，为scale out、sharding留有余地
 - ✓避开BUG
- 避免使用大表的JOIN
 - ✓MySQL最擅长的是单表的主键/二级索引查询
 - ✓JOIN消耗较多内存，产生临时表
- 避免在数据库中进行数学运算
 - ✓MySQL不擅长数学运算和逻辑判断
 - ✓无法使用索引

```
md5()/order by rand()  
select ... where to_days(current_date) - to_days(date_col) <= 10  
select ... where date_col >= date_sub(current_date, interval 10 day)  
select ... where date_col >= date_sub('2013-08-17', interval 10 day)  
select ... where date_col >= '2013-08-07'
```

数据库是有状态的服务，调整代码部署更灵活、简单、高效！

- 减少与数据库的交互次数
 - ✓INSERT ... ON DUPLICATE KEY UPDATE
 - ✓REPLACE INTO、INSERT IGNORE、INSERT INTO VALUES(),(),()
 - ✓UPDATE ... WHERE ID IN(10,20,50,...)
- 合理的使用分页
 - ✓限制分页展示的页数
 - ✓只能点击上一页、下一页
 - ✓采用延迟关联
- 拒绝大SQL，拆分成小SQL
 - ✓充分利用QUERY CACHE
 - ✓充分利用多核CPU

```
select * from profiles where sex='M' order by rating limit 10;  
select * from profiles where sex='M' order by rating limit 100000,10;  
select * from profiles inner join (select <pk> from pfiles where x.sex='M'  
order by rating limit 100000,10) as x using (<pk>)
```

- 使用in代替or，in的值不超过1000个
- 禁止使用order by rand()
- 使用EXPLAIN诊断，避免生成临时表
- 用union all而不是union
- 程序应有捕获SQL异常的处理机制
- 禁止单条SQL语句同时更新多个表
- 不使用select *
- ✓消耗CPU和IO、消耗网络带宽
- ✓无法使用覆盖索引
- ✓减少表结构变更带来的影响
- ✓因为大，select/join 可能生成临时表

```
select * from opp where phone='12345678' or phone='234234234'  
select * from opp where phone in ('12345678','234234234')  
select * from app where phone='010-88886666' or cellphone='18618111111'  
select * from opp where phone='010-88886666'  
union all  
select * from opp where cellphone=' 18618111111'
```


行为规范

- 批量导入、导出数据必须提前通知DBA协助观察
- 禁止在线上从库执行后台管理和统计类查询
- 禁止有super权限的应用程序账号存在
- 产品出现非数据库导致的故障时及时通知DBA协助排查
- 推广活动或上线新功能必须提前通知DBA进行流量评估
- 数据库数据丢失，及时联系DBA进行恢复
- 对单表的多次alter操作必须合并为一次操作
- 不在MySQL数据库中存放业务逻辑
- 重大项目的数据库方案选型和设计必须提前通知DBA参与
- 对特别重要的库表，提前与DBA沟通确定维护和备份优先级
- 不在业务高峰期批量更新、查询数据库
- 提交线上建表改表需求，必须详细注明所有相关SQL语句

良好的线上环境需要大家共同努力!

- 什么时候用CHAR，什么时候用VARCHAR?
- 为什么DELETE、UPDATE语句的WHERE条件列要加索引?
- 为什么INNODB表执行count(*)没有MyISAM快?

猛击[开发规范完整版](#)，提出你的建议！
需要其他培训或支持？请mail我们：dbateam@qunar.com

