



MySQL数据库开发的三十六条军规

石展@赶集

<http://weibo.com/wushizhan>



- 来自一线的实战经验
- 每一军规背后都是血淋淋教训
- 不要华丽，只要实用
- 若有一条让你有所受益，慰矣
- 主要针对数据库开发人员

总是在灾难发生后，才想起容灾的重要性；
总是在吃过亏后，才记得曾经有人提醒过。

一.核心军规(5)

二.字段类军规(6)

三.索引类军规(5)

四.SQL类军规(15)

五.约定类军规(5)

1

核心军规

gan 赶集 尽量不在数据库做运算

- 别让脚趾头想事情
- 那是脑瓜子的职责
- 让数据库多做她擅长的事：
 - ✓ 尽量不在数据库做运算
 - ✓ 复杂运算移到程序端CPU
 - ✓ 尽可能简单应用MySQL



- 举例: md5() / Order by Rand()

gan 赶集 尽量不在数据库做运算

- 别让脚趾头想事情
- 那是脑瓜子的职责
- 让数据库多做她擅长的事：
 - ✓ 尽量不在数据库做运算
 - ✓ 复杂运算移到程序端CPU
 - ✓ 尽可能简单应用MySQL



- 举例: md5() / Order by Rand()

ganji 赶集 控制单表数据量

gan 赶集 平衡范式与冗余

- 平衡是门艺术

- 严格遵循三大范式？
- 效率优先、提升性能
- 没有绝对的对与错
- 适当牺牲范式、加入冗余
- 但会增加代码复杂度



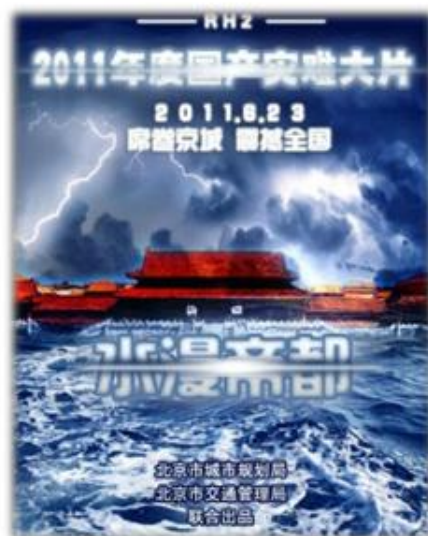
gan 赶集 拒绝3B

- 数据库并发像城市交通
 - 非线性增长
- 拒绝3B
 - 大SQL (**B**IG SQL)
 - 大事务 (**B**IG Transaction)
 - 大批量 (**B**IG Batch)
- 详细解析见后



gan 赶集 核心军规小结

- 尽量不在数据库做运算
- 控制单表数据量
- 保持表身段苗条
- 平衡范式与冗余
- 拒绝3B



gan 赶集

2

字段类军规



gan 赶集 用好数值字段类型

- 三类数值类型：

- ✓ TINYINT(1Byte)
- ✓ SMALLINT(2B)
- ✓ MEDIUMINT(3B)
- ✓ INT(4B)、BIGINT(8B)

BAD CASE:

- INT(1) VS INT(11)
- BIGINT AUTO_INCREMENT
- DECIMAL(18,0)

- ✓ FLOAT(4B)、DOUBLE(8B)
- ✓ DECIMAL(M,D)

gan 赶集 将字符转化为数字

- 数字型VS字符串型索引

- ✓ 更高效
- ✓ 查询更快
- ✓ 占用空间更小

- 举例：用无符号INT存储IP，而非CHAR(15)

- INT UNSIGNED
- INET_ATON()
- INET_NTOA()

ganji 赶集 优先使用ENUM或SET

- 优先使用ENUM或SET
 - 字符串
 - 可能值已知且有限
- 存储
 - ENUM占用1字节，转为数值运算
 - SET视节点定，最多占用8字节
 - 比较时需要加 ' 单引号(即使是数值)
- 举例
 - `sex` enum('F','M') COMMENT '性别'
 - `c1` enum('0','1','2','3') COMMENT '职介审核'

ganji 赶集 避免使用NULL字段

- 避免使用NULL字段
 - 很难进行查询优化
 - NULL列加索引，需要额外空间
 - 含NULL复合索引无效
- 举例
 - ~~`a` char(32) DEFAULT NULL~~
 - ~~`b` int(10) NOT NULL~~
 - `c` int(10) NOT NULL DEFAULT 0




ganji 赶集 少用并拆分TEXT/BLOB

- TEXT类型处理性能远低于VARCHAR
 - 强制生成硬盘临时表
 - 浪费更多空间
 - VARCHAR(65535) ==> 64K (注意UTF-8)
- 尽量不用TEXT/BLOB数据类型
- 若必须使用则拆分到单独的表
- 举例：

```
CREATE TABLE t1 (  
  id INT NOT NULL AUTO_INCREMENT,  
  data text NOT NULL,  
  PRIMARY KEY (id)  
) ENGINE=InnoDB;
```

ganji 赶集 不在数据库里存图片

post_image_id	created_time	file_bytes
1	2010-08-04 16:46:53	{Binary/Image} 39K
2	2010-08-04 16:46:55	{Binary/Image} 37K
3	2010-08-04 16:46:58	{Binary/Image} 70K
4	2010-08-04 19:18:34	{Binary/Image} 76K
5	2010-08-05 12:06:39	{Binary/Image} 34K
6	2010-08-05 12:06:40	{Binary/Image} 41K
7	2010-08-05 12:09:42	{Binary/Image} 48K
8	2010-08-05 14:21:14	{Binary/Image} 167K
9	2010-08-05 14:21:15	{Binary/Image} 197K
10	2010-08-05 14:21:16	{Binary/Image} 133K
11	2010-08-05 14:28:23	{Binary/Image} 64K
12	2010-08-05 14:33:38	{Binary/Image} 34K
13	2010-08-05 14:33:39	{Binary/Image} 41K
14	2010-08-05 14:36:17	{Binary/Image} 167K



```
(admin@ns)[(none)]> select count(*) from post_image_data;
+-----+
| count(*) |
+-----+
| 383676 |
+-----+
1 row in set (0.00 sec)

(admin@ns)[(none)]> quit
Bye
[work@yz ~]$ ll -h post_image_data*
-rw-rw---- 1 work work 88K Dec 29 00:27 post_image_data.frm
-rw-rw---- 1 work work 23G Jun 23 17:11 post_image_data.MYD
-rw-rw---- 1 work work 7.0M Jun 23 17:11 post_image_data.MVI
```

ganji 赶集 字段类军规小结

- 用好数值字段类型
- 将字符转化为数字
- 优先使用枚举ENUM/SET
- 避免使用NULL字段
- 少用并拆分TEXT/BLOB
- 不在数据库里存图片

ganji 赶集


3

索引类军规

gan 赶集 谨慎合理添加索引

- 谨慎合理添加索引
 - 改善查询
 - 减慢更新
 - 索引不是越多越好
- 能不加的索引尽量不加
 - 综合评估数据密度和数据分布
 - 最好不超过字段数20%
- 结合核心SQL优先考虑覆盖索引
- 举例
 - 不要给“性别”列创建索引

gan 赶集 字符字段必须建前缀索引

- 区分度
 - 单字母区分度：26
 - 4字母区分度： $26*26*26*26=456,976$
 - 5字母区分度： $26*26*26*26*26=11,881,376$ 
 - 6字母区分度： $26*26*26*26*26*26=308,915,776$
- 字符字段必须建前缀索引

```
`pinyin` varchar(100) DEFAULT NULL COMMENT '小区拼音',  
KEY `idx_pinyin` (`pinyin`(8)),  
) ENGINE=InnoDB
```

gan 赶集 不在索引列做运算

- 不在索引列进行数学运算或函数运算
 - 无法使用索引
 - 导致全表扫描
- 举例

```
(admin@ms)[beijing]] select * from work_log where id +1 = 6630913 \G  
1 row in set (51.62 sec)
```

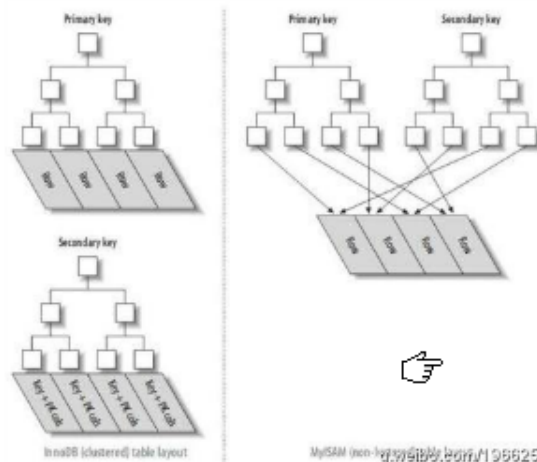
```
(admin@ms)[beijing]] select * from work_log where id = 6630913 -1 \G  
1 row in set (0.00 sec)
```

BAD: select * from table WHERE to_days(current_date) - to_days(date_col) <= 10

GOOD: select * from table WHERE date_col >= DATE_SUB('2011-10-22', INTERVAL 10 DAY);

gan 赶集 自增列或全局ID做INNODB主键

- 对主键建立聚簇索引
- 二级索引存储主键值
- 主键不应更新修改
- 按自增顺序插入值
- 忌用字符串做主键
- 聚簇索引分裂



- 推荐用独立于业务的AUTO_INCREMENT列或全局ID生成器做代理主键
- 若不指定主键，InnoDB会用唯一且非空值索引代替

gan 赶集 尽量不用外键

- 线上OLTP系统（线下系统另论）
 - 外键可节省开发量
 - 有额外开销
 - 逐行操作
 - 可‘到达’其它表，意味着锁
 - 高并发时容易死锁
- 由程序保证约束

gan 赶集 索引类军规小结

- 谨慎合理添加索引
- 字符字段必须建前缀索引
- 不在索引列做运算
- 自增列或全局ID做INNODB主键
- 尽量不用外键

4

SQL类军规

gan 赶集 SQL语句尽可能简单

- 大SQL **VS** 多个简单SQL
 - 传统设计思想
 - BUT MySQL NOT
 - 一条SQL只能在一个CPU运算
 - 5000+ QPS的高并发中，1秒大SQL意味着？
 - **可能一条大SQL就把整个数据库堵死**
- 拒绝大SQL，拆解成多条简单SQL
 - 简单SQL缓存命中率更高
 - 减少锁表时间，特别是MyISAM
 - 用上多CPU

ganji 赶集 保持事务(连接)短小

- 保持事务/DB连接短小精悍
 - 事务/连接使用原则：即开即用，用完即关
 - 与事务无关操作放到事务外面，减少锁资源的占用
 - 不破坏一致性前提下，使用多个短事务代替长事务
- 举例
 - 发贴时的图片上传等待
 - 大量的sleep连接

ganji 赶集 尽可能避免使用SP/TRIG/FUNC

- 线上OLTP系统（线下库另论）
 - 尽可能少用存储过程
 - 尽可能少用触发器
 - 减少使用MySQL函数对结果进行处理
- 由客户端程序负责

ganji 赶集 尽量不用 SELECT *

- 用SELECT * 时
 - 更多消耗CPU、内存、IO、网络带宽
 - 先向数据库请求所有列，然后丢掉不需要列？
- 尽量不用SELECT * ，只取需要数据列
 - 更安全的设计：减少表变化带来的影响
 - 为使用covering index提供可能性
 - Select/JOIN减少硬盘临时表生成，特别是有TEXT/BLOB时

- 举例

```
SELECT * FROM tag WHERE id = 999184
```

→

```
SELECT keyword FROM tag WHERE id = 999184
```



ganji 赶集 改写OR为IN()

- 同一字段，将or改写为in()
 - OR效率： $O(n)$
 - IN 效率： $O(\log n)$
 - 当n很大时，OR会慢很多
- 注意控制IN的个数，建议n小于200

- 举例

```
Select * from opp WHERE phone= '12347856' or  
phone= '42242233' \G
```

→

```
Select * from opp WHERE phone in ('12347856', '42242233')
```

ganji 赶集 改写OR为UNION

- 不同字段，将or改为union
 - 减少对不同字段进行 "or" 查询
 - Merge index往往很弱智
 - 如果有足够信心：set global optimizer_switch='index_merge=off';
- 举例
Select * from opp WHERE phone='010-88886666' or
cellPhone='13800138000';
→
Select * from opp WHERE phone='010-88886666'
union
Select * from opp WHERE cellPhone='13800138000';

ganji 赶集 避免负向查询和% 前缀模糊查询

- 避免负向查询
 - NOT、!=、<>、!<、!>、NOT EXISTS、NOT IN、NOT LIKE等
- 避免 % 前缀模糊查询
 - B+ Tree
 - 使用不了索引
 - 导致全表扫描
- 举例
MySQL> select * from post WHERE title like '北京%';
298 rows in set (0.01 sec)
MySQL> select * from post WHERE title like '%北京%';
572 rows in set (3.27 sec)

gan 赶集 COUNT(*)的几个例子

- 几个有趣的例子：

- COUNT(COL) VS COUNT(*)
- COUNT(*) VS COUNT(1)
- COUNT(1) VS COUNT(0) VS COUNT(100)

```
`id` int(10) NOT NULL  
AUTO_INCREMENT  
COMMENT '公司的id',  
`sale_id` int(10)  
unsigned DEFAULT NULL,
```

- 示例

```
mysql> select count(*),count(0),  
-> count(1),count(100),count(id  
-> count(sale_id),count(null)  
-> from user_company\G
```

- 结论

- ✓ COUNT(*)=count(1)
- ✓ COUNT(0)=count(1)
- ✓ COUNT(1)=count(100)
- ✓ COUNT(*)!=count(col)
- ✓ WHY?

```
***** 1. row  
count(*): 4091458  
count(0): 4091458  
count(1): 4091458  
count(100): 4091458  
count(id): 4091458  
count(sale_id): 403643  
count(null): 0
```

gan 赶集 减少COUNT(*)

- MyISAM VS INNODB

- ✓ 不带 WHERE COUNT()
- ✓ 带 WHERE COUNT()

- COUNT(*)的资源开销大，尽量不用少用

- 计数统计

- ✓ 实时统计：用memcache，双向更新，凌晨跑基准
- ✓ 非实时统计：尽量用单独统计表，定期重算

ganji 赶集 LIMIT 高效分页

- 传统分页：
 - `Select * from table limit 10000,10;`
- LIMIT 原理：
 - Limit 10000,10
 - 偏移量越大则越慢
- 推荐分页：
 - `Select * from table WHERE id >= 23423 limit 11;`
#10+1 (每页10条)
 - `select * from table WHERE id >= 23434 limit 11;`

ganji 赶集 LIMIT 的高效分页

- 分页方式二：
 - `Select * from table WHERE id >= (select id from table limit 10000,1) limit 10;`
- 分页方式三：
 - `SELECT * FROM table INNER JOIN (SELECT id FROM table LIMIT 10000,10) USING (id) ;`
- 分页方式四：
 - 程序取ID : `select id from table limit 10000,10;`
 - `Select * from table WHERE id in (123,456...);`
- 可能需按场景分析并重组索引

gan 赶集 LIMIT的高效分页

- 示例：

MySQL> select sql_no_cache * from post limit 10,10;
10 row in set (0.01 sec)

MySQL> select sql_no_cache * from post limit 20000,10;
10 row in set (0.13 sec)

MySQL> select sql_no_cache * from post limit 80000,10;
10 rows in set (0.58 sec)

MySQL> select sql_no_cache id from post limit 80000,10;
10 rows in set (0.02 sec)

MySQL> select sql_no_cache * from post WHERE id >= 323423 limit 10;
10 rows in set (0.01 sec)

MySQL> select * from post WHERE id >= (select sql_no_cache id from post limit 80000,1) limit 10 ;
10 rows in set (0.02 sec)

gan 赶集 用UNION ALL 而非 UNION

- 若无需对结果进行去重，则用UNION ALL
 - UNION有去重开销

- 举例

MySQL> SELECT * FROM detail20091128 UNION ALL
SELECT * FROM detail20110427 UNION ALL
SELECT * FROM detail20110426 UNION ALL
SELECT * FROM detail20110425 UNION ALL
SELECT * FROM detail20110424 UNION ALL
SELECT * FROM detail20110423;

赶集 分解联接保证高并发

- 高并发DB不建议进行两个表以上的JOIN

- 适当分解联接保证高并发

- ✓ 可缓存大量早期数据
- ✓ 使用了多个MyISAM表
- ✓ 对大表的小ID IN()
- ✓ 联接引用同一个表多次

- 举例：

```
MySQL> Select * from tag JOIN tag_post on tag_post.tag_id=tag.id  
JOIN post on tag_post.post_id=post.id WHERE tag.tag= '二手玩具' ;
```

→

```
MySQL> Select * from tag WHERE tag= '二手玩具' ;
```

```
MySQL> Select * from tag_post WHERE tag_id=1321;
```

```
MySQL> Select * from post WHERE post.id in (123,456,314,141)
```

赶集 GROUP BY 去除排序

- GROUP BY 实现

- ✓ 分组
- ✓ 自动排序

- 无需排序：Order by NULL

- 特定排序：Group by DESC/ASC

- 举例

```
MySQL> select phone,count(*) from post group by phone limit 1 ;  
1 row in set (2.19 sec)
```

```
MySQL> select phone,count(*) from post group by phone order by null limit 1;  
1 row in set (2.02 sec)
```

ganji 赶集 同数据类型的列值比较

- 原则：数字对数字，字符对字符
- 数值列与字符类型比较
 - 同时转换为双精度
 - 进行比对
- 字符列与数值类型比较
 - 字符列整列转数值
 - 不会使用索引查询

ganji 赶集 同数据类型的列值比较

- 举例：字符列与数值类型比较

字段：`remark` varchar(50) NOT NULL COMMENT '备注，默认为空'，

```
MySQL>SELECT `id`, `gift_code` FROM gift WHERE  
`deal_id` = 640 AND remark=115127;  
1 row in set (0.14 sec)
```

```
MySQL>SELECT `id`, `gift_code` FROM pool_gift WHERE  
`deal_id` = 640 AND remark='115127';  
1 row in set (0.005 sec)
```

ganji 赶集 Load data 导数据

- 批量数据快导入：
 - 成批装载比单行装载更快，不需要每次刷新缓存
 - 无索引时装载比索引装载更快
 - Insert values , values , values 减少索引刷新
 - Load data比insert快约20倍
- 尽量不用 INSERT ... SELECT
 - 延迟
 - 同步出错

ganji 赶集 打散大批量更新

- 大批量更新凌晨操作，避开高峰
- 凌晨不限制
- 白天上限默认为100条/秒（特殊再议）
- 举例：

```
update post set tag=1 WHERE id in (1,2,3);
sleep 0.01;
update post set tag=1 WHERE id in (4,5,6);
sleep 0.01;
.....
```

SHOW PROFILE

MySQLsla

MySQLdumpslow

EXPLAIN

Show Slow Log

Show Processlist

SHOW QUERY_RESPONSE_TIME(Percona)

gan 赶集 SQL类军规小结

- SQL语句尽可能简单
- 保持事务(连接)短小
- 尽可能避免使用SP/TRIG/FUNC
- 尽量不用 SELECT *
- 改写OR语句
- 避免负向查询和% 前缀模糊查询
- 减少COUNT(*)
- LIMIT的高效分页
- 用UNION ALL 而非 UNION
- 分解联接保证高并发
- GROUP BY 去除排序
- 同数据类型的列值比较
- Load data导数据
- 打散大批量更新
- Know Every SQL !

约定类军规

- 构建数据库的生态环境
 - 开发无线上库操作权限
- 原则：线上连线上，线下连线下
 - 实时数据用real库
 - 模拟环境用sim库
 - 测试用qa库
 - 开发用dev库

- 案例：

```
| 90228839 | 600 | 90228839 | 15114928222' | zip  
city='5', realname='', mobiles='15114928222', zip  
| 90231464 | 600 | 90231464 | 15114928222' |  
Extent1.. id,  
Extent1.. partner_id,  
Extent1.. app_id,  
Extent1.. app_code,  
R |  
| 90231817 | 600 | 90231817 | 15114928222' | I
```

```
1090 rows in set (0.00 sec)
```

gan 赶集 禁止未经DBA确认的子查询

- MySQL子查询
 - 大部分情况优化较差
 - 特别WHERE中使用IN id的子查询
 - 一般可用JOIN改写

- 举例:

~~MySQL> select * from table1 where id in (select id from table2);~~

~~MySQL> insert into table1 (select * from table2);~~
~~//可能导致复制异常~~

gan 赶集 永远不在程序端显式加锁

- 永远不在程序端对数据库显式加锁

- 外部锁对数据库不可控
- 高并发时是灾难
- 极难调试和排查

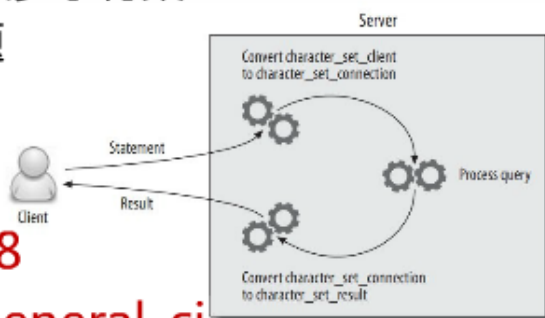


- 并发扣款等一致性问题

- 采用事务
- 相对值修改
- Commit前二次校验冲突

gan 赶集 统一字符集为UTF8

- 字符集：
 - MySQL 4.1 以前只有latin1
 - 为多语言支持增加多字符集
 - 也带来了N多问题
 - 保持简单



- 统一字符集：UTF8
- 校对规则：utf8_general_ci
- 乱码：SET NAMES UTF8

gan 赶集 统一命名规范

- 库表等名称统一用小写
 - Linux VS Windows
 - MySQL库表大小写敏感
 - 字段名的大小写不敏感
- 索引命名默认为“idx_字段名”
- 库名用缩写，尽量在2~7个字母
 - DataSharing ==> ds
- 注意避免用保留字命名
-



注意避免用保留字命名

- 举例: ~~Select * from return;~~

Select * from return ;

ADD	ALL	ALTER	GOTO	GRANT	GROUP	PURGE	RAID0	RANGE
ANALYZE	AND	AS	HAVING	HIGH_PRIORITY	HOUR_MICROSECOND	READ	READS	REAL
ASC	ASENSITIVE	BEFORE	HOUR_MINUTE	HOUR_SECOND	IF	REFERENCES	REGEXP	RELEASE
BETWEEN	BIGINT	BINARY	IGNORE	IN	INDEX	RENAME	REPEAT	REPLACE
BLOB	BOTH	BY	INFILE	INNER	INOUT	REQUIRE	RESTRICT	RETURN
CALL	CASCADE	CASE	INSENSITIVE	INSERT	INT	REVOKE	RIGHT	RLIKE
CHANGE	CHAR	CHARACTER	INT1	INT2	INT3	SCHEMA	SCHEMAS	SECOND_MICROSECOND
CHECK	COLLATE	COLUMN	INT4	INT8	INTEGER	SELECT	SENSITIVE	SEPARATOR
CONDITION	CONNECTION	CONSTRAINT	INTERVAL	INTO	IS	SET	SHOW	SMALLINT
CONTINUE	CONVERT	CREATE	ITERATE	JOIN	KEY	SPATIAL	SPECIFIC	SQL
CROSS	CURRENT_DATE	CURRENT_TIME	KEYS	KILL	LABEL	SQL_EXCEPTION	SQLSTATE	SQLWARNING
CURRENT_TIMESTAMP	CURRENT_USER	CURSOR	LEADING	LEAVE	LEFT	SQL_BIG_RESULT	SQL_CALC_FOUND_ROWS	SQL_SMALL_RESULT
DATABASE	DATABASES	DAY_HOUR	LIKE	LIMIT	LINEAR	SSL	STARTING	STRAIGHT_JOIN
DAY_MICROSECOND	DAY_MINUTE	DAY_SECOND	LINES	LOAD	LOCALTIME	TABLE	TERMINATED	THEN
DEC	DECIMAL	DECLARE	LOCALTIMESTAMP	LOCK	LONG	TINYBLOB	TINYINT	TINYTEXT
DEFAULT	DELAYED	DELETE	LONGBLOB	LONGTEXT	LOOP	TO	TRAILING	TRIGGER
DESC	DESCRIBE	DETERMINISTIC	LOW_PRIORITY	MATCH	MEDIUMBLOB	TRUE	UNDO	UNION
DISTINCT	DISTINCTROW	DIV	MEDIUMINT	MEDIUMTEXT	MIDDLEINT	UNIQUE	UNLOCK	UNSIGNED
DOUBLE	DROP	DUAL	MINUTE_MICROSECOND	MINUTE_SECOND	MOD	UPDATE	USAGE	USE
EACH	ELSE	ELSEIF	MODIFIES	NATURAL	NOT	USING	UTC_DATE	UTC_TIME
ENCLOSED	ESCAPED	EXISTS	NO_WRITE_TO_BINLOG	NULL	NUMERIC	UTC_TIMESTAMP	VALUES	VARBINARY
EXIT	EXPLAIN	FALSE	ON	OPTIMIZE	OPTION	VARCHAR	VARCHARACTER	VARYING
FETCH	FLOAT	FLOAT4	OPTIONALLY	OR	ORDER	WHEN	WHERE	WHILE
FLOAT8	FOR	FORCE	OUT	OUTER	OUTFILE	WITH	WRITE	X509
FOREIGN	FROM	FULLTEXT	PRECISION	PRIMARY	PROCEDURE	XOR	YEAR_MONTH	ZEROFILL



约定类军规小结

- 隔离线上线下的
- 禁止未经DBA确认的子查询上线
- 永远不在程序端显式加锁
- 统一字符集为UTF8
- 统一命名规范