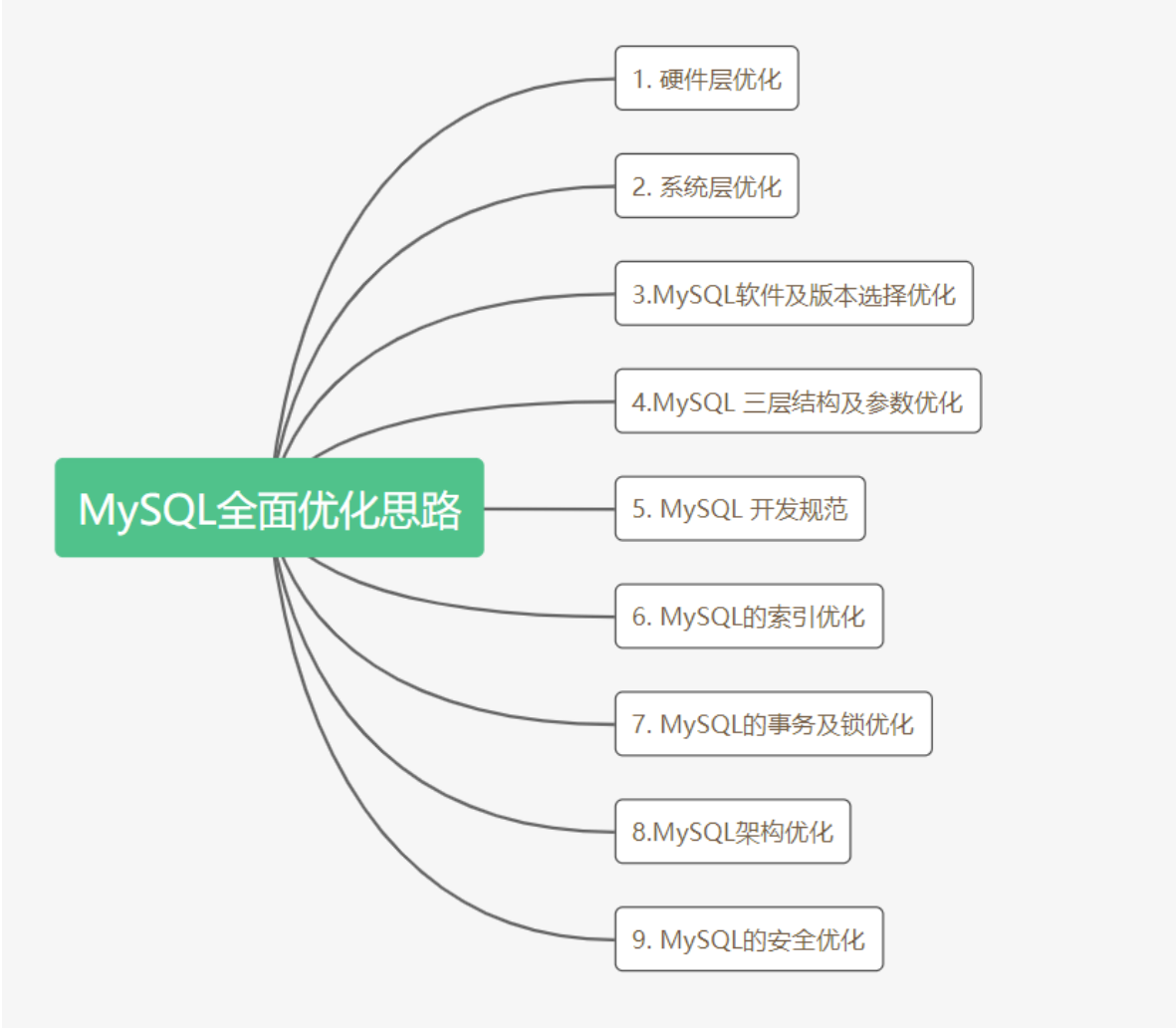
MySQL全面优化

0.优化思路



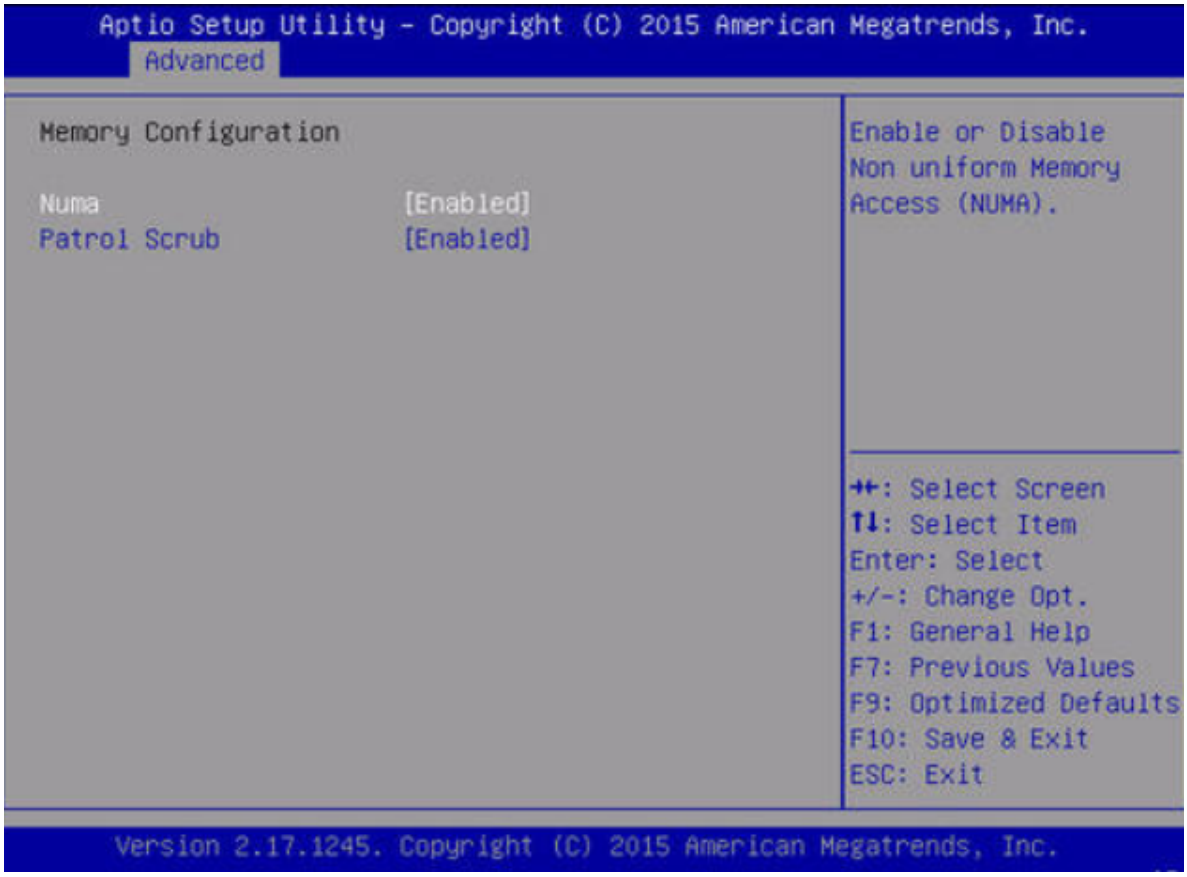
1.硬件层面优化

1.0 硬件选配

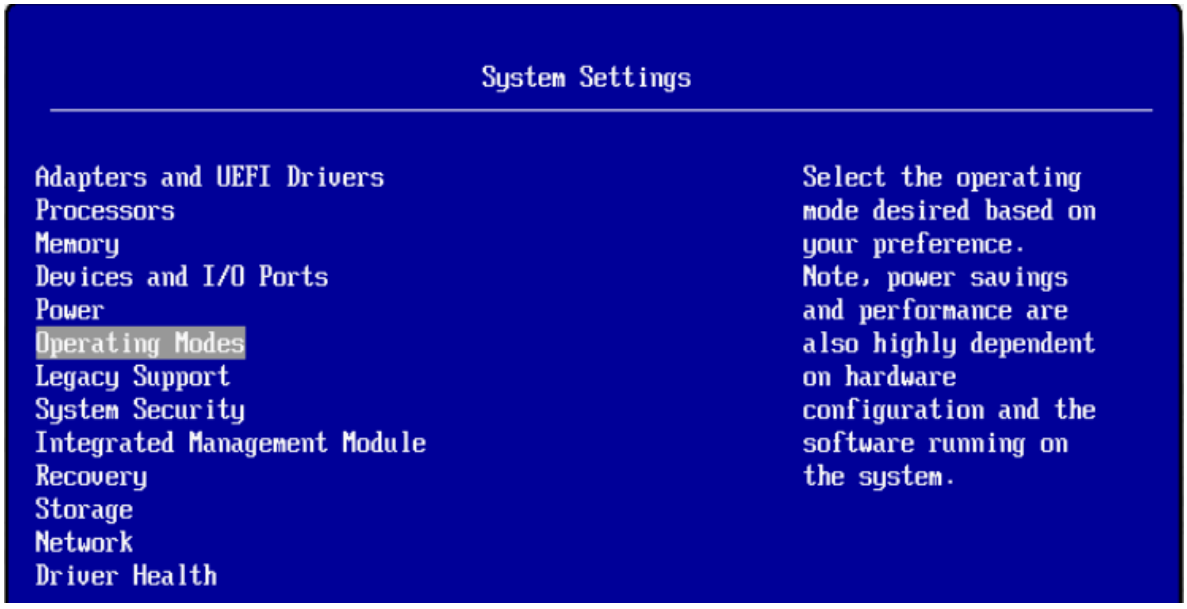
```
DELL、HP、IBM、华为、浪潮。
CPU：I、E
内存：ECC
IO : SAS 、 pci-e SSD 、 Nvme flash
raid卡：Raid10
网卡： 单卡单口

云服务器： ECS 、RDS 、PolarDB、TDSQL
```

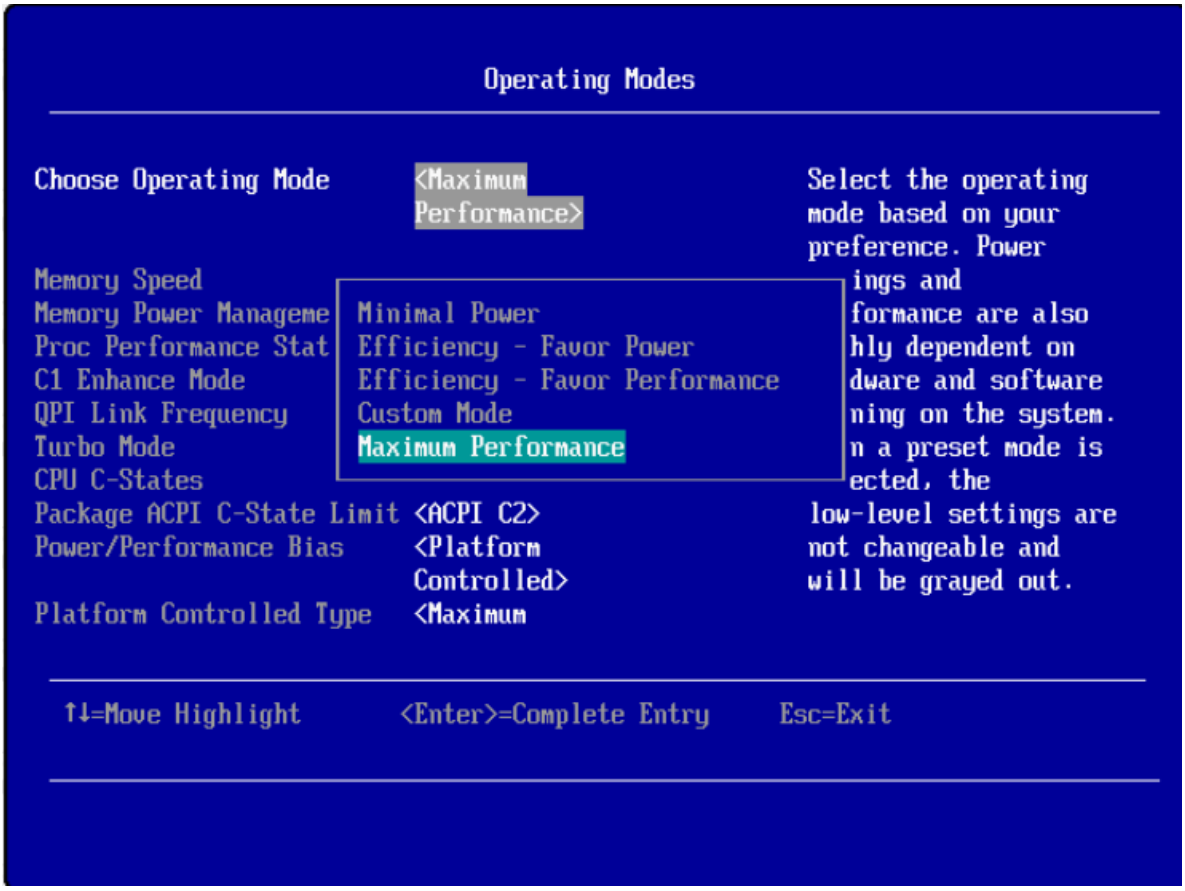
1.1 关闭NUMA

```
a. bios级别:
在bios层面numa关闭时，无论os层面的numa是否打开，都不会影响性能。

# numactl --hardware
available: 1 nodes (0)        #如果是2或多个nodes就说明numa没关掉

b. OS grub级别:
vi /boot/grub/grub.conf
#/* Copyright 2010, Oracle. All rights reserved. */

default=0
timeout=5
hiddenmenu
foreground=000000
background=ffffff
splashimage=(hd0,0)/boot/grub/oracle.xpm.gz

title Trying_COD0_as_HD0
root (hd0,0)
kernel /boot/vmlinuz-2.6.18-128.1.16.0.1.el5 root=LABEL=DBSYS ro
bootarea=dbsys rhgb quiet console=ttyS0,115200n8 console=tty1
crashkernel=128M@16M numa=off
initrd /boot/initrd-2.6.18-128.1.16.0.1.el5.img

在os层numa关闭时,打开bios层的numa会影响性能，QPS会下降15-30%;

c. 数据库级别:

mysql> show variables like '%numa%';
+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| innodb_numa_interleave | OFF   |
+------------------------+-------+

或者:
vi /etc/init.d/mysqld
找到如下行
# Give extra arguments to mysqld with the my.cnf file. This script
# may be overwritten at next upgrade.
$bindir/mysqld_safe --datadir="$datadir" --pid-file="$mysqld_pid_file_path"
$other_args >/dev/null &

wait_for_pid created "$!" "$mysqld_pid_file_path"; return_value=$?
将$bindir/mysqld_safe --datadir="$datadir"这一行修改为:

/usr/bin/numactl --interleave all $bindir/mysqld_safe --datadir="$datadir"
--pid-file="$mysqld_pid_file_path" $other_args >/dev/null &
wait_for_pid created "$!" "$mysqld_pid_file_path"; return_value=$?
```



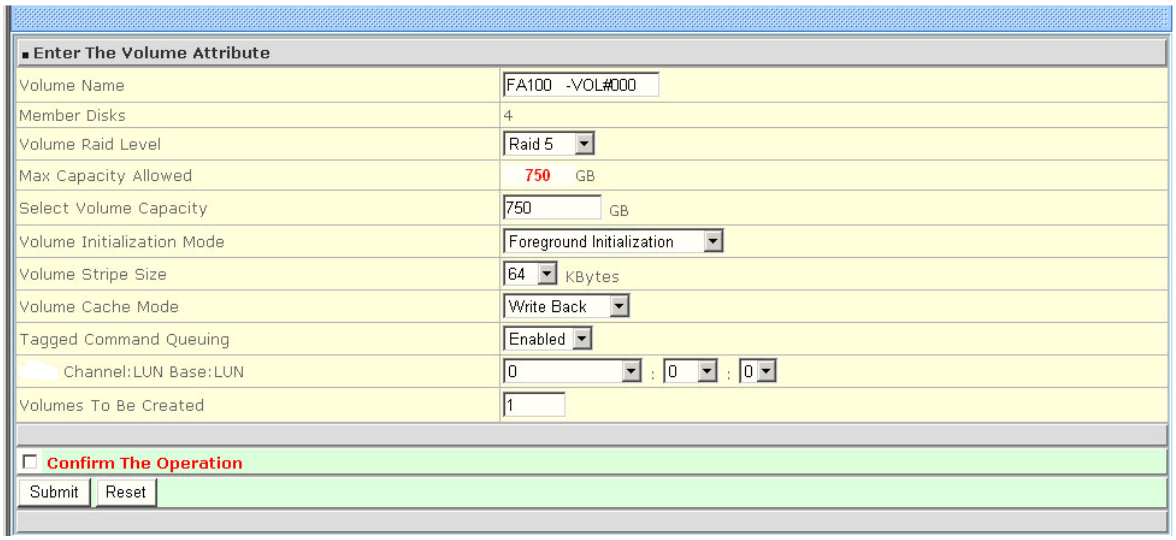
1.2 开启CPU高性能模式





1.3 阵列卡配置建议

```
raid10(推荐)
SSD或者PCI-E或者Flash
强制回写（Force WriteBack）
BBU 电池 ： 如果没电会有较大性能影响、定期充放电，如果UPS、多路电源、发电机。可以关闭。
关闭预读
有可能的话开启Cache(如果UPS、多路电源、发电机。)
```



1.4 关闭THP

```
vi /etc/rc.local
```

```
在文件末尾添加如下指令：
if test -f /sys/kernel/mm/transparent_hugepage/enabled; then
   echo never > /sys/kernel/mm/transparent_hugepage/enabled
fi
if test -f /sys/kernel/mm/transparent_hugepage/defrag; then
   echo never > /sys/kernel/mm/transparent_hugepage/defrag
fi

[root@master ~]# cat /sys/kernel/mm/transparent_hugepage/enabled
always madvise [never]
[root@master ~]# cat  /sys/kernel/mm/transparent_hugepage/defrag
always madvise [never]

```

1.5 网卡绑定

```
bonding技术，业务数据库服务器都要配置bonding继续。建议是主备模式。
交换机一定要堆叠。
```

1.6 存储多路径

```
使用独立存储设备的话，需要配置多路径。
linux 自带 : multipath
厂商提供    :
```

2.系统层面优化

```
a. 更改文件句柄和进程数
内核优化 /etc/sysctl.conf
vm.swappiness <= 5（也可以设置为0）
vm.dirty_ratio <= 20
vm.dirty_background_ratio <= 10
net.ipv4.tcp_max_syn_backlog = 819200
net.core.netdev_max_backlog = 400000
net.core.somaxconn = 4096
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_tw_recycle=0

limits.conf
nofile 63000

b. 防火墙
禁用selinux :  /etc/sysconfig/selinux 更改SELINUX=disabled.
iptables如果不使用可以关闭。可是需要打开MySQL需要的端口号

c. 文件系统优化
推荐使用XFS文件系统
MySQL数据分区独立 ，例如挂载点为: /data
mount参数 defaults, noatime, nodiratime, nobarrier 如/etc/fstab:
/dev/sdb /data                    xfs
defaults,noatime,nodiratime,nobarrier        1 2

```

```
d. 不使用LVM

e. io调度
SAS ： deadline
SSD&PCI-E： noop


```

3. 数据库版本选择

```
1、稳定版：选择开源的社区版的稳定版GA版本。
2、选择mysql数据库GA版本发布后6个月-12个月的GA双数版本，大约在15-20个小版本左右。
3、要选择前后几个月没有大的BUG修复的版本，而不是大量修复BUG的集中版本。
4、要考虑开发人员开发程序使用的版本是否兼容你选的版本。
5、作为内部开发测试数据库环境，跑大概3-6个月的时间。
6、优先企业非核心业务采用新版本的数据库GA版本软件。
7、向DBA高手请教，或者在技术氛围好的群里和大家一起交流，使用真正的高手们用过的好用的GA版本产品。

最终建议： 8.0.20是一个不错的版本选择。向后可以选择双数版。
```

4.数据库三层结构及核心参数优化

4.1 连接层

```
max_connections=单节点不高于3000
max_connect_errors=大一点。
wait_timeout=600
interactive_wait_timeout=600
net_read_timeout
net_write_timeout
max_allowed_packet
```

4.2 Server层

```
sql_safe_updates                =1
slow_query_log                  =ON
slow_query_log_file             =/xxx
long_query_time                 =1
log_queries_not_using_indexes   =ON
log_throttle_queries_not_using_indexes = 10
sort_buffer\join_buffer\read_buffer\read_rnd_buffer，建议不超过8M
tmp_table、heap_table,建议不要超过128M
sql_mode,建议保持默认。
max_execution_time，建议跑批量是设置较大。
lock_wait_timeout，建议设置在60秒以内
lower_case_table_names          =1
thread_cache_size               =64
character_set_server            =utf8或者utf8mb4
log_timestamps                  =SYSTEM
init_connect                    ="set names utf8"
```

```
event_scheduler                 =OFF
secure-file-priv                =/xxx
expire_logs_days                =10
sync_binlog                     =1
log-bin                         =/opt/log/mysql/blog/mysql-bin
log-bin-index                   =/opt/log/mysql/blog/mysql-bin.index
max_binlog_size                 =500M
binlog_format                   =ROW
max_binlog_cache_size           =2G
max_binlog_stmt_cache_size      =2G

```

4.3 存储引擎层

```
transaction-isolation               ="READ-COMMITTED"
innodb_data_home_dir                =/xxx
innodb_log_group_home_dir           =/xxx
innodb_log_file_size                =2048M
innodb_log_files_in_group           =3
innodb_flush_log_at_trx_commit      =2
innodb_flush_method                 =O_DIRECT/fsync
innodb_io_capacity                  =1000
innodb_io_capacity_max              =4000
innodb_buffer_pool_size             =64G
innodb_buffer_pool_instances        =4
innodb_log_buffer_size              =64M
innodb_max_dirty_pages_pct          =85
innodb_lock_wait_timeout            =10
innodb_open_files                   =63000
innodb_page_cleaners                =4
innodb_sort_buffer_size             =64M
innodb_print_all_deadlocks          =1
innodb_rollback_on_timeout          =ON
innodb_deadlock_detect              =ON
```

4.4 复制

```
relay_log                       =/opt/log/mysql/blog/relay
relay_log_index                 =/opt/log/mysql/blog/relay.index
max_relay_log_size              =500M
relay_log_purge                 =ON
relay_log_recovery              =ON
rpl_semi_sync_master_enabled                =ON
rpl_semi_sync_master_timeout                =1000
rpl_semi_sync_master_trace_level            =32
rpl_semi_sync_master_wait_for_slave_count   =1
rpl_semi_sync_master_wait_no_slave          =ON
rpl_semi_sync_master_wait_point             =AFTER_SYNC
rpl_semi_sync_slave_enabled                 =ON
rpl_semi_sync_slave_trace_level             =32
binlog_group_commit_sync_delay              =1
binlog_group_commit_sync_no_delay_count     =1000
gtid_mode                       =ON
```

```
enforce_gtid_consistency        =ON
master_verify_checksum          =ON
sync_master_info                =1
skip-slave-start                =1
#read_only                      =ON
#super_read_only                =ON
log_slave_updates               =ON
server_id                       =2330602
report_host                     =xxxx
report_port                     =3306
slave_parallel_type                         =LOGICAL_CLOCK
slave_parallel_workers                      =4
master_info_repository                      =TABLE
relay_log_info_repository                   =TABLE
```

4.5 其它

```
客户端配置：
[mysql]
no-auto-rehash
pager less
```

5.开发规范

5.1 字段规范

```
1. 每个表建议在30个字段以内。
2. 需要存储emoji字符的，则选择utf8mb4字符集。
3. 机密数据，加密后存储。
4. 整型数据，默认加上UNSIGNED。
5. 存储IPV4地址建议用INT UNSIGNE，查询时再利用INET_ATON()、INET_NTOA()函数转换。
6. 如果遇到BLOB、TEXT大字段单独存储表或者附件形式存储。
7. 选择尽可能小的数据类型，用于节省磁盘和内存空间。
8. 存储浮点数，可以放大倍数存储。
9. 每个表必须有主键，INT/BIGINT并且自增做为主键，分布式架构使用sequence序列生成器保存。
10. 每个列使用not null，或增加默认值。
```

5.2 SQL语句规范

```
### 1. 去掉不必要的括号
如：       ((a AND b) AND c OR (((a AND b) AND (c AND d))))
修改成     (a AND b AND c) OR (a AND b AND c AND d)
### 2.  去掉重叠条件
如：       (a<b AND b=c) AND a=5
修改成     b>5 AND b=c AND a=5
如：       (B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)
修改成     B=5 OR B=6
### 3. 避免使用not in、not exists 、<>、like %%
### 4. 多表连接，小表驱动大表
### 5. 减少临时表应用，优化order by 、group by、uninon、distinct、join等
### 6. 减少语句查询范围，精确查询条件
### 7. 多条件，符合联合索引最左原则
```

```
### 8. 查询条件减少使用函数、拼接字符等条件、条件隐式转换
### 9. union all 替代 union
### 10.减少having子句使用
### 11.如非必须不使用 for update语句
### 12.update和delete，开启安全更新参数
### 13.减少inset  ... select语句应用
### 14.使用load 替代insert录入大数据
### 15.导入大量数据时，可以禁用索引、增大缓冲区、增大redo文件和buffer、关闭autocommit、RC级别可以提高效率
### 16.优化limit，最好业务逻辑中先获取主键ID，再基于ID进行查询
    limit 5000000,10
### 17. DDL执行前要审核
### 18. 多表连接语句执行前要看执行计划

```

6.索引优化

```
1. 非唯一索引按照“i_字段名称_字段名称[_字段名]”进行命名。
2. 唯一索引按照“u_字段名称_字段名称[_字段名]”进行命名。
3. 索引名称使用小写。
4. 索引中的字段数不超过5个。
5. 唯一键由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。
6. 没有唯一键或者唯一键不符合5中的条件时，使用自增id作为主键。
7. 唯一键不和主键重复。
8. 索引选择度高的列作为联合索引最左条件
9. ORDER BY，GROUP BY，DISTINCT的字段需要添加在索引的后面。
10. 单张表的索引数量控制在5个以内，若单张表多个字段在查询需求上都要单独用到索引，需要经过DBA评估。查询性能问题无法解决的，应从产品设计上进行重构。
11. 使用EXPLAIN判断SQL语句是否合理使用索引，尽量避免extra列出现：Using File Sort，Using Temporary。
12. UPDATE、DELETE语句需要根据WHERE条件添加索引。
13. 对长度大于50的VARCHAR字段建立索引时，按需求恰当的使用前缀索引，或使用其他方法。
14. 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 all_url(ID INT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
url VARCHAR(255) NOT NULL DEFAULT 0,
url_crc32 INT UNSIGNED NOT NULL DEFAULT 0,
index idx_url(url_crc32));
15. 合理创建联合索引（避免冗余），(a,b,c) 相当于 (a) 、(a,b) 、(a,b,c)。
16. 合理利用覆盖索引，减少回表。
17. 减少冗余索引和使用率较低的索引
```

7.锁优化

7.1 latch 闩锁

a. 介绍

```
    Latch用于管理对共享内存资源的并发访问，例如，操作缓冲池汇总的LRU列表，删除、添加、移动LRU列表中的元素，为了保证一致性，必须有锁的介入，这就是latch锁。
```

b.latch 和 lock的区别

	lock	latch
对象	SQL操作流程	线程
保护	数据库对象（库、表、行、索引、表空间、数据页等）	所有共享内存数据结构
生命周期	整个操作周期	临界资源
锁定模式	MDL、table、Record、Gap、Nextlock、意向	rw-latch、mutex

c.查看latch争用的类型

```
mysql> show engine innodb mutex;
+--------+-----------------------------+----------+
| Type   | Name                        | Status   |
+--------+-----------------------------+----------+
| InnoDB | rwlock: dict0dict.cc:2687   | waits=1  |
| InnoDB | rwlock: dict0dict.cc:1184   | waits=13 |
| InnoDB | rwlock: log0log.cc:844      | waits=35 |
| InnoDB | sum rwlock: buf0buf.cc:1457 | waits=4  |
+--------+-----------------------------+----------+

可以在latch争用较为严重情况下，定位到源码的位置点，从而获得到底什么原因导致争用。

也可以在此时通过以下工具分析堆栈信息:
pstack -p `pidof mysqld` >/tmp/aa.txt
pt-pmp /tmp/aa.txt|more
```

d.什么时候发生争用

```
1）a 访问x内存链表
2）b 排队等待x解锁 ，占了cpu，但是cpu发现你在等待，所以cpu将b踢出
3）访问锁链的时间，就是找数据的时间。
4）b知道很a快所以，b不去排队，这时去spin也就是空转cpu，然后再去看一下内存数据结构，a是否已解锁
5）b转了一圈后，在b spin的时间段的时间中，c进来了，连续多次的spin后，产生了os waits
6）操作系统将b从cpu中踢出

latch争用的表面现象：latch争用会表现为cpu繁忙，IO很闲，没有做实际的事情。

```

e.如何监控是否latach争用较为严重

```
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 13
OS WAIT ARRAY INFO: signal count 13
RW-shared spins 0, rounds 0, OS waits 0
RW-excl spins 2, rounds 60, OS waits 2
RW-sx spins 2, rounds 60, OS waits 2
Spin rounds per wait: 0.00 RW-shared, 30.00 RW-excl, 30.00 RW-sx

rounds:    意思是每次询问旋转的次数
os waits：表示sleep，当突然增长比较快的时候，说明latch争用比较严重
```

```
rw-shared spin 的次数
rw-excl   spin的次数
```

f. latch争用发生的原因

```
1、内存访问太频繁（不停地找）
2、list链太长（链上挂10000个快，被持有的几率太大）

```

g.如何降低latch争用

```
如果出现latch争用比较严重
1.优化大sql，降低对内存读的数量--效果比较明显
2.增加instances的数量
```

7.2 全局锁 Global Read lock

a. 介绍

```
全局读锁。
加锁方法： FTWRL，flush tables with read lock.
解锁方法： unlock tables;
出现场景：
    mysqldump  --master-data
    xtrabackup（8.0之前早期版本）等备份时。
属于类型： MDL（matedatalock）层面锁
影响情况： 加锁期间，阻塞所有事务写入，阻塞所有已有事务commit。
MDL，等待时间受 lock_wait_timeout=31536000
```

b. 检测方法

```
UPDATE performance_schema.setup_instruments
SET ENABLED = 'YES', TIMED = 'YES'
WHERE NAME = 'wait/lock/metadata/sql/mdl';

mysql> select * from performance_schema.metadata_locks;

mysql> select OBJECT_SCHEMA ,OBJECT_NAME
,LOCK_TYPE,LOCK_DURATION,LOCK_STATUS ,OWNER_THREAD_ID,OWNER_EVENT_ID from
performance_schema.metadata_locks;

mysql> show processlist;
mysql> select * from sys.schema_table_lock_waits;

```

c. 一个经典故障：5.7 xtrabackup/mysqldump备份时数据库出现hang状态，所有查询都不能进行

```
session1: 模拟一个大的查询或事务
mysql> select *,sleep(100)  from city where id<10 limit 1 ;

```

```
session2: 模拟备份时的FTWRL
mysql> flush tables with read lock;
-- 此时发现命令被阻塞

session3: 发起正常查询，发现被阻塞
mysql> select * from world.city where id=1;

结论： 备份时，一定要选择业务不繁忙期间，否则有可能会阻塞正常业务。


案例2：
5.7 innobackupex备份全库，进程死了，mysql里就是全库读锁，后边insert 全阻塞了

```

7.3 Table lock

a. 介绍

```
表锁。
加锁方式：
    lock table read. 所有会话只读。属于MDL锁。
    lock table write.当前持有会话可以RW，其他会被阻塞。属于MDL锁
    select for update ;
    select for share  ;
解锁方式：
unlock tables;
```

b. 检测方式

```
[mysqld]
performance-schema-instrument='wait/lock/metadata/sql/mdl=ON'
mysql> select * from performance_schema.metadata_locks;

```

7.4 MDL锁

a. 介绍

```
Matedata lock .元数据锁。
作用范围：global 、commit、tablespace、schema、table等
默认timeout时间： lock_wait_timeout
mysql> select @@lock_wait_timeout;
+----------------------+
| @@lock_wait_timeout |
+----------------------+
|            31536000 |
+----------------------+
```

b. 监控

```
[mysqld]
performance-schema-instrument='wait/lock/metadata/sql/mdl=ON'
mysql> select * from performance_schema.metadata_locks;

找到
OWNER_THREAD_ID: 62
mysql>  select * from threads where thread_id='62'\G

PROCESSLIST_ID: 21
kill 21;


```

7.5 autoinc_lock

```
自增锁。
通过参数： innodb_autoinc_lock_mode=0,1,2设定
0： 表锁，每次插入都请求表锁，效率低。
1： mutex方式，预计插入多少行，预申请自增序列。如果出现load或者insert select方式会退化为0.
2： 强制使用mutex方式。并发插入可以更高效。

作用：
The default innodb_autoinc_lock_mode setting is now 2 (interleaved). Interleaved lock mode permits the execution of multi-row inserts in parallel, which improves concurrency and scalability.
```

7.6 row lock

a.介绍

```
record lock 、gap、next lock
都是基于索引加锁,与事务隔离级别有关。
```

b. 监控及分析

```
show status like 'innodb_row_lock%'
select * from information_schema.innodb_trx;
select * from sys.innodb_lock_waits;
select * from performance_schema.threads;
select * from performance_schema.events_statements_current;
select * from performance_schema.events_statements_history;
```

c. 优化方向

```
1. 优化索引
2. 减少事务的更新范围
3. RC
4. 拆分语句：
例如：   update t1 set num=num+10 where k1 <100;  k1 是辅助索引,record lock gap next
       改为:
       select id from t1 where  k1 <100; ---> id: 20,30,50
       update t1 set num=num+10   where id in (20,30,50);
```

7.7 死锁

a.介绍

```
dead lock 多个并发事务之间发生交叉资源依赖时，会出现。
```

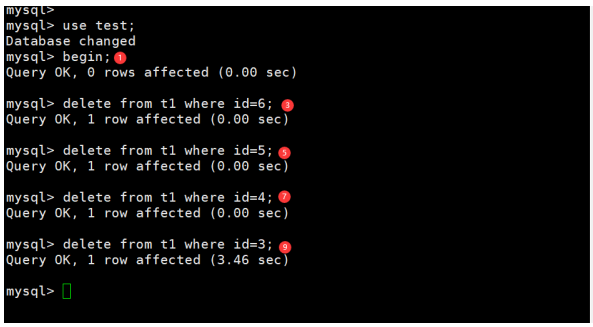
b. 监控及分析

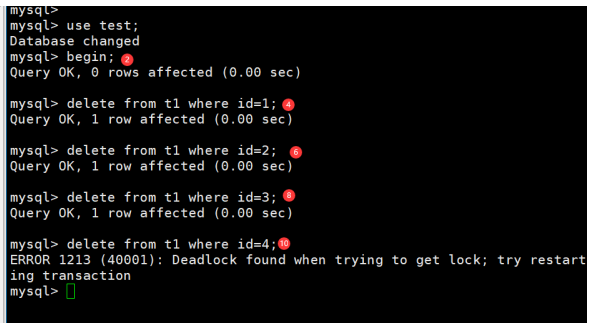
```
show engine innodb status \G
innodb_print_all_deadlocks =1
```

c. 经典死锁案例延时及解析

```
mysql> select * from t1;
+----+------+
| id | k1   |
+----+------+
|  1 | a    |
|  2 | b    |
|  3 | c    |
|  4 | d    |
|  5 | e    |
|  6 | f    |
+----+------+
6 rows in set (0.00 sec)

mysql>
```

```
mysql> desc t1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int         | NO   | PRI | NULL    |       |
| k1    | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```





8.架构优化

```
高可用架构：
    MHA+ProxySQL+GTID
    MGR\InnoDB Cluster
    PXC
读写分离：
    ProxySQL、MySQL-router
NoSQL：
    Redis+sentinel,Redis Cluster
    MongoDB RS/MongoDB SHARDING Cluster
    ES
```

9.安全优化

```
1、 使用普通nologin用户管理MySQL
2、 合理授权用户、密码复杂度及最小权限、系统表保证只有管理员用户可访问。
3、 删除数据库匿名用户
4、 锁定非活动用户
5、 MySQL尽量不暴露互联网,需要暴露互联网用户需要设置明确白名单、替换MySQL默认端口号、使用ssl连接
6、 优化业务代码，防止SQL注入。
```

10.常用工具介绍

一、 PT（percona-toolkits）工具的应用:

1. pt工具安装
   [root@master ~]# yum install -y percona-toolkit-3.1.0-2.el7.x86_64.rpm

2. 常用工具使用介绍
   2.1 pt-archiver 归档表
   场景：
   面试题： 亿级的大表，delete批量删除100w左右数据。
   面试题： 定期按照时间范围，进行归档表。

重要参数

--limit 100　　每次取100行数据用pt-archive处理
--txn-size 100　设置100行为一个事务提交一次，
--where 'id<3000'　设置操作条件
--progress 5000　每处理5000行输出一次处理信息
--statistics　　输出执行过程及最后的操作统计。（只要不加上--quiet，默认情况下pt- archive都会输出执行过程的）
--charset=UTF8　指定字符集为UTF8—这个最后加上不然可能出现乱码。
--bulk-delete　　批量删除source上的旧数据(例如每次1000行的批量删除操作)

注意: 需要归档表中至少有一个索引,做好是where条件列有索引

使用案例:
1.归档到数据库
db01 [test]>create table test1 like t100w;
pt-archiver --source h=10.0.0.51,D=test,t=t100w,u=oldguo,p=123 --dest h=10.0.0.51,D=test,t=test1,u=oldguo,p=123 --where 'id<10000' --no-check-charset --no-delete --limit=1000 --commit-each --progress 1000 --statistics

2.只清理数据
pt-archiver --source h=10.0.0.51,D=test,t=t100w,u=oldguo,p=123 --where 'id<10000' --purge --limit=1 --no-check-charset

3.只把数据导出到外部文件，但是不删除源表里的数据
pt-archiver --source h=10.0.0.51,D=world,t=city,u=root,p=123 --where '1=1' --no-check-charset --no-delete --file="/tmp/archiver.dat"

2.2 pt-osc

场景:
　　修改表结构、索引创建删除
　　不能加快速度，但能减少业务影响（锁）。

面试题：
pt-osc工作流程:
1、检查更改表是否有主键或唯一索引，是否有触发器
2、检查修改表的表结构，创建一个临时表，在新表上执行ALTER TABLE语句
create table bak like t1;
alter table bak add telnum char(11) not null;
3、在源表上创建三个触发器分别对于INSERT UPDATE DELETE操作
create trigger
a
b
c
4、从源表拷贝数据到临时表，在拷贝过程中，对源表的更新操作会写入到新建表中
insert into bak select * from t1
5、将临时表和源表rename（需要元数据修改锁，需要短时间锁表）
6、删除源表和触发器，完成表结构的修改。

pt-osc工具限制
1、源表必须有主键或唯一索引，如果没有工具将停止工作
2、如果线上的复制环境过滤器操作过于复杂，工具将无法工作
3、如果开启复制延迟检查，但主从延迟时，工具将暂停数据拷贝工作

4、如果开启主服务器负载检查，但主服务器负载较高时，工具将暂停操作
5、当表使用外键时，如果未使用--alter-foreign-keys-method参数，工具将无法执行
6、只支持Innodb存储引擎表，且要求服务器上有该表1倍以上的空闲空间。

pt-osc之alter语句限制
1、不需要包含alter table关键字，可以包含多个修改操作，使用逗号分开，如"drop clolumn c1, add column c2 int"
2、不支持rename语句来对表进行重命名操作
3、不支持对索引进行重命名操作
4、如果删除外键，需要对外键名加下划线，如删除外键fk_uid, 修改语句为"DROP FOREIGN KEY _fk_uid"

pt-osc之命令模板

--execute表示执行

--dry-run表示只进行模拟测试

表名只能使用参数t来设置，没有长参数

```
pt-online-schema-change \
--host="127.0.0.1" \
--port=3358 \
--user="root" \
--password="root@root" \
--charset="utf8" \
--max-lag=10 \
--check-salve-lag='xxx.xxx.xxx.xxx' \
--recursion-method="hosts" \
--check-interval=2 \
--database="testdb1" \
  t="tb001" \
--alter="add column c4 int" \
--execute
```

例子:
pt-online-schema-change --user=oldguo --password=123 --host=10.0.0.51 --alter "add column state int not null default 1" D=test,t=t100w --print --execute
pt-online-schema-change --user=oldguo --password=123 --host=10.0.0.51 --alter "add index idx(num)" D=test,t=t100w --print --execute

2.3 pt-table-checksum
场景:　校验主从数据一致性

2.3.1 创建数据库
Create database pt CHARACTER SET utf8;

创建用户checksum并授权
GRANT ALL ON . TO 'checksum'@'10.0.0.%' IDENTIFIED BY 'checksum';
flush privileges;

2.3.2 参数:
--[no]check-replication-filters：是否检查复制的过滤器，默认是yes，建议启用不检查模式。
--databases | -d：指定需要被检查的数据库，多个库之间可以用逗号分隔。
--[no]check-binlog-format：是否检查binlog文件的格式，默认值yes。建议开启不检查。因为在默认的row格式下会出错。
--replicate`：把checksum的信息写入到指定表中。
--replicate-check-only：只显示不同步信息

pt-table-checksum --nocheck-replication-filters --no-check-binlog-format --replicate=pt.checksums --create-replicate-table --databases=test --tables=t1
h=10.0.0.51,u=checksum,p=checksum,P=3306

#!/bin/bash
date >> /root/db/checksum.log
pt-table-checksum --nocheck-binlog-format --nocheck-plan --nocheck-replication-filters --replicate=pt.checksums --set-vars innodb_lock_wait_timeout=120 --databases test --tables t1 -u'checksum' -p'checksum' -h'10.0.0.51' >> /tmp/checksum.log
date >> /root/db/checksum.log

2.4 pt-table-sync
主要参数介绍
--replicate： 指定通过pt-table-checksum得到的表.
--databases : 指定执行同步的数据库。
--tables： 指定执行同步的表，多个用逗号隔开。
--sync-to-master： 指定一个DSN，即从的IP，他会通过show processlist或show slave status 去自动的找主。
h=： 服务器地址，命令里有2个ip，第一次出现的是Master的地址，第2次是Slave的地址。
u=： 帐号。
p=： 密码。
--print： 打印，但不执行命令。
--execute： 执行命令。

pt-table-sync --replicate=pt.checksums --databases test --tables t1 h=10.0.0.51,u=checksum,p=checksum,P=3306 h=10.0.0.52,u=checksum,p=checksum,P=3306 --print

pt-table-sync --replicate=pt.checksums --databases test --tables t1 h=10.0.0.51,u=checksum,p=checksum,P=3306 h=10.0.0.52,u=checksum,p=checksum,P=3306 --execute

2.5 pt-duplicate-key-checker
作用：检查数据库重复索引
pt-duplicate-key-checker --database=test h='10.0.0.51' --user=oldguo --password=123

2.6 pt-kill 语句
场景： 无法正常kill的连接。

常用参数说明
--daemonize 放在后台以守护进程的形式运行;
--interval 多久运行一次，单位可以是s,m,h，d等默认是s –不加这个默认是5秒
--victims 默认是oldest,只杀最古老的查询。这是防止被查杀是不是真的长时间运行的查询，他们只是长期等待 这种种匹配按时间查询，杀死一个时间最高值。
--all 杀掉所有满足的线程
--kill-query 只杀掉连接执行的语句，但是线程不会被终止
--print 打印满足条件的语句
--busy-time 批次查询已运行的时间超过这个时间的线程;

--idle-time 杀掉sleep 空闲了多少时间的连接线程，必须在--match-command sleep时才有效—也就是匹配使用 -- –match-command 匹配相关的语句。
----ignore-command 忽略相关的匹配。 这两个搭配使用一定是ignore-commandd在前 match-command在后，
--match-db cdelzone 匹配哪个库
command有：Query、Sleep、Binlog Dump、Connect、Delayed insert、Execute、Fetch、Init DB、Kill、Prepare、Processlist、Quit、Reset stmt、Table Dump

例子：

杀掉空闲链接sleep 5秒的 SQL 并把日志放到/home/pt-kill.log文件中

/usr/bin/pt-kill --user=用户名 --password=密码 --match-command Sleep --idle-time 5 --victim all --interval 5 --kill --daemonize -S /tmp/mysql.sock --pid=/tmp/ptkill.pid --print --log=/tmp/pt-kill.log &

查询SELECT 超过1分钟

/usr/bin/pt-kill --user=用户名 --password=密码 --busy-time 60 --match-info "SELECT|select" --victim all --interval 5 --kill --daemonize -S -S /tmp/mysql.sock --pid=/tmp/ptkill.pid --print --log=/tmp/pt-kill.log &

Kill掉 select IFNULl.*语句开头的SQL

pt-kill --user=用户名 --password=密码 --victims all --busy-time=0 --match-info="select IFNULl.*" --interval 1 -S /tmp/mysqld.sock --kill --daemonize --pid=/tmp/ptkill.pid --print --log=/tmp/pt-kill.log &

kill掉state Locked

/usr/bin/pt-kill --user=用户名 --password=密码 --victims all --match-state='Locked' --victim all --interval 5 --kill --daemonize -S /tmp/mysqld.sock --pid=/tmp/ptkill.pid --print --log=/tmp/pt-kill.log &

kill掉 a库，web为10.0.0.11的链接

pt-kill --user=用户名 --password=密码 --victims all --match-db='a' --match-host='10.0.0.11' --kill --daemonize --interval 10 -S /tmp/mysqld.sock --pid=/tmp/ptkill.pid --print-log=/tmp/pt-kill.log &

指定哪个用户kill

pt-kill   --user=用户名 --password=密码 --victims all --match-user='root' --kill --daemonize --interval 10 -S /home/zb/data/my6006/socket/mysqld.sock --pid=/tmp/ptkill.pid --print --log=/home/pt-kill.log &

kill掉 command query | Execute

pt-kill --user=用户名 --password=密码 --victims all --match-command= "query|Execute" --interval 5 --kill --daemonize -S /tmp/mysqld.sock --pid=/tmp/ptkill.pid --print --log=/home/pt-kill.log &

7. 显示主从结构：pt-slave-find
   [root@db01 tmp]# pt-slave-find -h10.0.0.51 -P3306 -uchecksum -pchecksum
   10.0.0.51
   Version   5.7.28-log
   Server ID   51

Uptime 27:57 (started 2020-05-15T13:24:15)
Replication Is not a slave, has 1 slaves connected, is not read_only
Filters
Binary logging ROW
Slave status
Slave mode STRICT
Auto-increment increment 1, offset 1
InnoDB version 5.7.28
+- 10.0.0.52
Version 5.7.28-log
Server ID 52
Uptime 28:18 (started 2020-05-15T13:23:54)
Replication Is a slave, has 0 slaves connected, is not read_only
Filters
Binary logging ROW
Slave status 0 seconds behind, running, no errors
Slave mode STRICT
Auto-increment increment 1, offset 1
InnoDB version 5.7.28
[root@db01 tmp]#

8. 监控主从延时

pt-heartbeat

主库:
pt-heartbeat --user=oldguo --ask-pass --host=10.0.0.51 --port=3306 --create-table -D test --interval=1 --update --replace --daemonize

从库:
pt-heartbeat --user=oldguo --ask-pass --host=10.0.0.52 --port=3306 -D test --table=heartbeat --monitor

9. pt-show-grants

作用: 用户和权限信息迁移。

pt-show-grants -h10.0.0.51 -P3306 -uchecksum -pchecksum

-- Grants dumped by pt-show-grants
-- Dumped from server 10.0.0.51 via TCP/IP, MySQL 5.7.28-log at 2020-05-15 17:11:06
-- Grants for 'checksum'@'10.0.0.%'
CREATE USER IF NOT EXISTS 'checksum'@'10.0.0.%';
ALTER USER 'checksum'@'10.0.0.%' IDENTIFIED WITH 'mysql_native_password' AS '*E5E390AF1BDF241B51D9C0DBBEA262CC9407A2DF' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK;
GRANT ALL PRIVILEGES ON . TO 'checksum'@'10.0.0.%';
-- Grants for 'mysql.session'@'localhost'
CREATE USER IF NOT EXISTS 'mysql.session'@'localhost';
ALTER USER 'mysql.session'@'localhost' IDENTIFIED WITH 'mysql_native_password' AS

'THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT LOCK;
GRANT SELECT ON `mysql`.`user` TO 'mysql.session'@'localhost';
GRANT SELECT ON `performance_schema`. TO 'mysql.session'@'localhost';
GRANT SUPER ON . TO 'mysql.session'@'localhost';
-- Grants for 'mysql.sys'@'localhost'
CREATE USER IF NOT EXISTS 'mysql.sys'@'localhost';
ALTER USER 'mysql.sys'@'localhost' IDENTIFIED WITH 'mysql_native_password' AS *'THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT LOCK;*
GRANT SELECT ON `sys`.`sys_config` TO 'mysql.sys'@'localhost';
GRANT TRIGGER ON `sys`. TO 'mysql.sys'@'localhost';
GRANT USAGE ON . TO 'mysql.sys'@'localhost';
-- Grants for 'repl'@'10.0.0.%'
CREATE USER IF NOT EXISTS 'repl'@'10.0.0.%';
ALTER USER 'repl'@'10.0.0.%' IDENTIFIED WITH 'mysql_native_password' AS '*23AE809DDACAF96AF0FD78ED04B6A265E05AA257' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK;
GRANT REPLICATION SLAVE ON . TO 'repl'@'10.0.0.%';
-- Grants for 'root'@'10.0.0.%'
CREATE USER IF NOT EXISTS 'root'@'10.0.0.%';
ALTER USER 'root'@'10.0.0.%' IDENTIFIED WITH 'mysql_native_password' AS '*23AE809DDACAF96AF0FD78ED04B6A265E05AA257' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK;
GRANT ALL PRIVILEGES ON . TO 'root'@'10.0.0.%';
-- Grants for 'root'@'localhost'
CREATE USER IF NOT EXISTS 'root'@'localhost';
ALTER USER 'root'@'localhost' IDENTIFIED WITH 'mysql_native_password' AS '*23AE809DDACAF96AF0FD78ED04B6A265E05AA257' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK;
GRANT ALL PRIVILEGES ON . TO 'root'@'localhost' WITH GRANT OPTION;
GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION;

11.IS、PS、SYS用法介绍