一、简述MySQL索引及其作用?

是数据库管理系统中一个排序的数据结构，根据不同的存储引擎索引分为Hash索引、B+树索引等。常见的InnoDB存储引擎的默认索引实现为：B+树索引。索引可以协助快速查询、更新数据库表中数据。

二、什么是事务?

事务是一系列的操作，需要要符合ACID特性，即：事务中的操作要么全部成功，要么全部失败。

三、如何保证数据的完整性?（事务的特点）

特点:

```
Atomicity 原子性
Consistency 一致性
Isolation 隔离性
Durability 耐久性
```

1、原子性

原子性：所有语句作为一个单元全部成功执行或全部取消。不能出现中间状态。

初中知识：原子是物质的构成单元之一，具备化学不可分割性
在一个事务工作单元中，所有标准事务语句（DML），要么全成功，要么全回滚。

2、一致性

一致性：如果数据库在事务开始时处于一致状态，则在执行该事务期间将保留一致状态。

事务发生前、中、后都应该保证数据始终一致状态
MySQL的各项功能的设置，都是最终要保证一致性。

例如：
一个转账事务，里面有两条sql语句，一条是张三减少100元，另一个是李四加100元
转账前：
张三：500元
李四：500元
总额：1000元

事务执行完成后，即转账后
张三：400元
李四：600元
两个人总钱数：1000元

ps：前后数据类型也要保持一致

3、隔离性

隔离性：事务之间不相互影响。
mysql支持多事务并发工作的系统。
a工作的时候不能收到其他事物的影响

4、持久性

持久性：事务成功完成后，所做的所有更改都会准确地记录在数据库中。所做的更改不会丢失。

当事务提交（commit命令执行成功后，此次事务操作的所有数据“落盘”），都要永久保存下去。不会因为数据实例发生故障。

四、事务的隔离机制

1、未提交读(Read Uncommitted)

允许脏读，其他事务只要修改了数据，即使未提交，本事务也能看到修改后的数据值。也就是可能读取到其他会话中未提交事务修改的数据。

有可能出现的问题：

脏页读，不可重复读，幻读

2、提交读(Read Committed)

只能读取到已经提交的数据。Oracle等多数数据库默认都是该级别 (不重复读)。

有可能出现的问题：

不可重复读，幻读

3、可重复读(Repeated Read)

可重复读。无论其他事务是否修改并提交了数据，在这个事务中看到的数据值始终不受其他事务影响。

有可能出现的问题：

幻读，但是可以通过其他手段防止幻读出现。

4、串行读(Serializable)

完全串行化的读，每次读都需要获得表级共享锁，读写相互都会阻塞。

串行化事务。以上问题都能规避，但是不利于事务的并发。

五、简述不同隔离基础出现的问题

1、脏读

1)读到别人在内存中未提交的数据。（只begin未commit，但是其他用户看到了）

2)内存缓存原因，直接读脏页数据

2、不可重复读

1)同一个事务窗口，一个事务读到另一个事务修改后并提交的数据（update）。在同一个事务中，对于同一组数据读取到的结果不一致。比如，事务B 在 事务A 提交前读到的结果，和在 事务A 提交后读到的结果可能不同。不可重复读出现的原因就是由于事务并发修改记录而导致的。

2)一个用户两次查看数据之间，另一个用户对这个数据进行了更新。两次查看不一致。

防止不可重复读现象：

利用的就是undo的一致性快照读，mvcc重要功能

3、幻读

1)同一个事物窗口，用户1进行事务更新过程中，用户2完成一个事务更新。影响到用户1的操作。（用户1将id>10的用户信息删除过程中（未commit），用户2添加了一个id>10的用户（已经commit）。当用户1完成删除操作，再次查看，发现了用户2添加的用户（用户1不知道啊）。就出现了幻读。

2)一个用户进行数据更改期间，另一个用户也进行修改并完成。互相干扰。出现数据未完全修改。

3）在一个事务窗口中，更新过程中，出现了别的插入数据的换行。

防止幻读现象：

通过RR，已经可以解决99%以上的幻读。为了更加严谨，加入了GAP(间隙锁)锁，next-lock（行锁加GAP）。

六、锁机制

机制是为了避免，在数据库有并发事务的时候，可能会产生数据的不一致而诞生的的一个机制。锁从类别上分为：

共享锁：

又叫做读锁，当用户要进行数据的读取时，对数据加上共享锁，共享锁可以同时加上多个。

排他锁：

又叫做写锁，当用户要进行数据的写入时，对数据加上排他锁，排他锁只可以加一个，他和其他的排他锁,共享锁都相斥。

七、简述MySQL表中为什么建议添加主键?

主键是数据库确保数据行在整张表唯一性的保障，即使数据库中表没有主键，也建议添加一个自增长的ID列作为主键，设定了主键之后，在后续的删改查的时候可能更加快速以及确保操作数据范围安全。

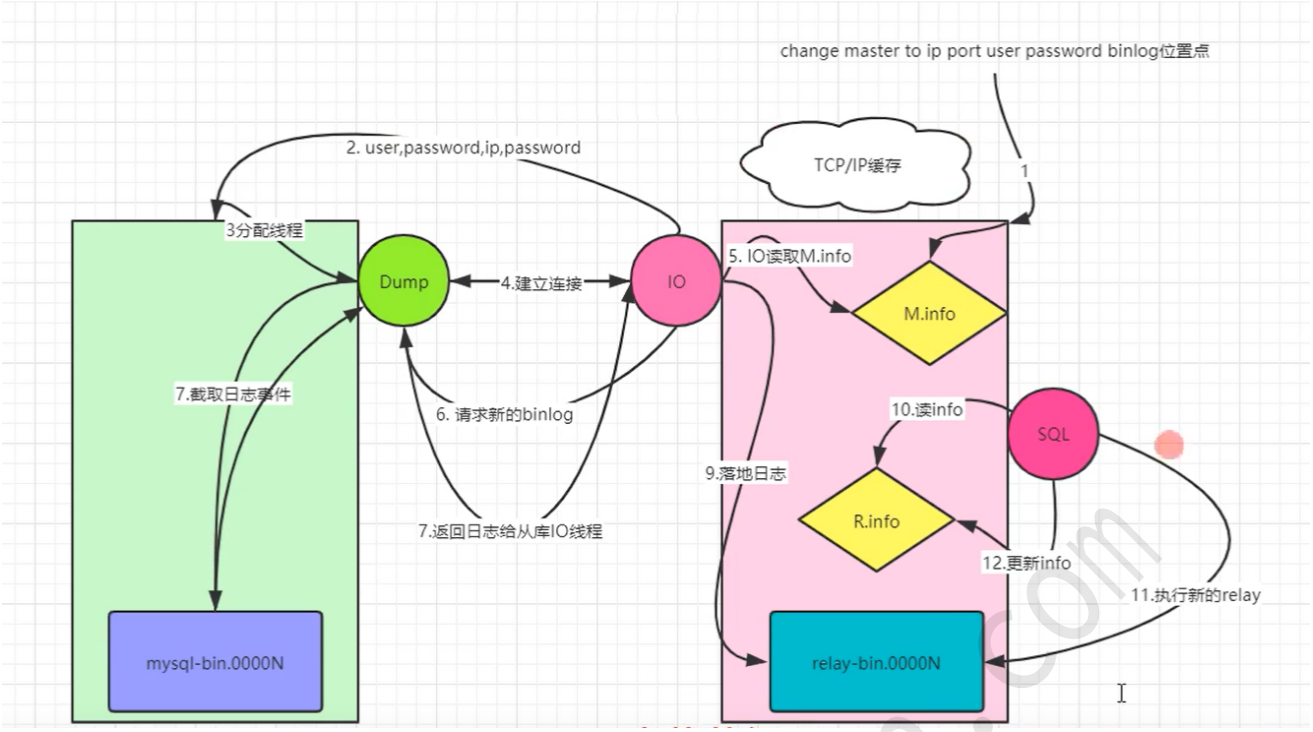
八、简述MySQL所支持的存储引擎?

MySQL支持多种存储引擎，常见的有InnoDB、MyISAM、Memory、Archive等。通常使用InnoDB引擎都是最合适的，InnoDB也是MySQL的默认存储引擎。

九、简述MySQL InnoDB引擎和MyISAM引擎的差异?

InnoDB支持事物，而MyISAM不支持事物。
InnoDB支持行级锁，而MyISAM支持表级锁。
InnoDB支持MVCC，而MyISAM不支持。
InnoDB支持外键，而MyISAM不支持。
InnoDB不支持全文索引，而MyISAM支持。

十、主从复制的过程



1.slave：change master to 时，ip port user password binlog position写入到master.info进行记录
2.slave： start slave 时，从库会启动IO线程和SQL线程
3.IO_T，读取master.info信息，获取主库port、ip、user、password信息连接主库
4. 主库会生成一个准备binlog DUMP线程，来响应从库
5. IO_T根据master.info记录的binlog文件名和position号，请求主库DUMP最新日志
6. DUMP线程检查主库的binlog日志，如果有新的，TP(传送)给从库的IO_T
7. IO_T将收到的日志存储到了TCP/IP 缓存，立即返回ACK给主库 ，主库工作完成
8.IO_T将缓存中的数据，存储到relay-log日志文件,更新master.info文件中binlog文件名和postion，IO_T工作完成
9.SQL_T读取relay-log.info文件，获取到上次执行到的relay-log的位置，作为起点，回放relay-log
10.SQL_T回放完成之后，会更新relay-log.info文件。
11. relay-log会有自动清理的功能。参数：relay_log_purge=on,定期删除应用过的relay-log
细节：
12.主库一旦有新的日志生成，会发送“信号”给binlog dump ，IO线程再请求

十一、高可用方案

MySQL主从复制：Mysql内建的复制功能是构建大型，高性能应用程序的基础。将Mysql的数据分布在多个节点（slaves）之上，复制过程中一个服务器充当主服务器，而一个或多个其它服务器充当从服务器。主服务器将更新写入二进制日志文件，并维护文件的一个索引以跟踪日志循环。这些日志可以记录发送到从服务器的更新。
MySQL双主：参考MySQL主从复制。
MySQL双主多从：参考MySQL主从复制。
MySQL复制+Keepalived高可用：MySQL自身的复制，对外基于Keepalived技术，暴露一个VIP，从而实现高可用。
Heartbeat + MySQL 实现MySQL的高可用：通过Heartbeat的心跳检测和资源接管、集群中服务的监测、失效切换等功能，结合MySQL来实现高可用性。

十二、mysql优化

1、硬件优化（物理机数据库）

1.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 bonding + 交换机堆叠
云服务器： ECS 、RDS 、TDSQL、PolarxDB
```

2.关闭NUMA

1）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
```

2）OS grub级别:

```
vi /boot/grub2/grub.cfg
#/* 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_C0D0_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%;
```

3）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"
$other_args >/dev/null &
wait_for_pid created "$!" "$mysqld_pid_file_path"; return_value=$?
```

3.磁盘阵列卡建议

```
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。)
```

2、系统优化

1.内核优化

```
内核优化 /etc/sysctl.conf
vm.swappiness = 5
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
```

2.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
```

3.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
```

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

1.连接层

```
max_connections=1000          #*****
max_connect_errors=999999
wait_timeout=600              #*****
interactive_wait_timeout=3600
net_read_timeout   = 120
net_write_timeout  = 120
max_allowed_packet= 32M       #*****
```

2.server层

```
sql_safe_updates                =1                # *****
slow_query_log                  =ON
slow_query_log_file             =/data/3307/slow.log    # *****
long_query_time                 =1                  # *****
log_queries_not_using_indexes   =ON                 # *****
log_throttle_queries_not_using_indexes = 10         # *****
sort_buffer                     = 1M
join_buffer                     = 1M
read_buffer                     = 1M
read_rnd_buffer                 = 1M
tmp_table                       = 16M
heap_table                      = 16M
max_execution_time              = 28800
lock_wait_timeout               = 60                # *****
lower_case_table_names          =1                  # *****
thread_cache_size               =64
log_timestamps                  =SYSTEM             # *****
init_connect                    ="set names utf8"   # *****
event_scheduler                 =OFF
secure-file-priv                =/tmp               # *****
binlog_expire_logs_seconds      =2592000            # *****
sync_binlog                     =1                  # *****
log-bin                         =/data/3307/mysql-bin
log-bin-index                   =/data/3307/mysql-bin.index
max_binlog_size                 =500M
binlog_format                   =ROW
```

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            # *****
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.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_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

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
```

5.其他

```
客户端配置：
[mysql]
no-auto-rehash
```

5、开发规范

1.字段规范

1. 每个表建议在30个字段以内(了解三大范式)。
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bigINT 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。

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、union、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     limit 10 , 200
### 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

```
mysql> select * from sys.schema_unused_indexes;
```

```
mysql> select * from sys.schema_redundant_indexes\G
```

7、锁优化

1.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
```

2.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\G

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;

场景： 业务反馈所有写入做不了.
mysql> show processlist;
mysql> select * from performance_schema.metadata_locks\G
mysql> select * from sys.schema_table_lock_waits;


沟通后决定怎么处理?
kill ?
```

3.一个经典故障：5.7

```
xtrabackup/mysqldump备份时数据库出现hang状态，所有修改查询都不能进行
```
```
session1: 模拟一个大的查询或事务
mysql> select id,sleep(100)  from city where id<100  for update ;

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

session3: 发起查询，发现被阻塞
mysql> select * from world.city where id=1 for update;

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。
```

4.row lock wait

1）介绍

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

2）行锁监控及分析

```
# 查询锁等待详细信息
select * from sys.innodb_lock_waits;   ----> blocking_pid(锁源的连接线程)

# 通过连接线程找SQL线程
select * from performance_schema.threads;

# 通过SQL线程找到 SQL语句
select * from performance_schema.events_statements_history;
```

3）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);
```

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。
```

十三、简述MySQL常见备份方式和工具?

1、MySQL自带

mysqldump：mysqldump支持基于innodb的热备份，使用mysqldump完全备份+二进制日志可以实现基于时间点的恢复，通常适合备份数据比较小的场景。

系统层面

tar备份：可以使用tar之类的系统命令对整个数据库目录进行打包备份。

lvm快照备份：可基于文件系统的LVM制作快照，进行对整个数据库目录所在的逻辑卷备份。

2、第三方备份工具

可使用其他第三方工具进行备份，如xtrabackup工具，该工具支持innodb的物理热备份，支持完全备份、增量备份，而且速度非常快，支持innodb存储引起的数据在不同数据库之间迁移，支持复制模式下的从机备份恢复备份恢复。