第1章 中级 DBA 面试宝典

1.1 体系结构及基础管理

1.1.1 MySQL 有哪些安装方式？

```
Centos: rpm(yum)、二进制、源码。
```

1.1.2 MySQL 5.6 、 5.7 安装过程有什么区别？

```
初始化：
5.6 : /data/app/mysql/scripts/mysql_install_db
5.7 : mysqld --initialize-insecure / --initialize
```

1.1.3 MySQL 5.7、 8.0 在用户管理有什么区别，请举例说明？

```
8.0 之前：
用户管理：
1. create user oldboy@’10.0.0.%’ identified by ‘123’;
2. grant all on *.* to oldboy@’10.0.0.%’;
或者：
Grant all on *.* to oldguo@’10.0.0.%’ identified by ‘123’;
8.0 以后：
1. 只能先建用户后授权
2. 加密方式：从 native 插件方式改变为了 sha2 加密插件。远程客户端、主从、MHA 架构等。
```

1.1.4 请介绍 MySQL 授权表有哪些，都有什么作用？

```
User         ：用户、密码、加密插件、全库级别的用户权限。
Db           ：单库级别
tables_priv  ：表级别
columns_priv：列级别
procs_priv   ：存储过程
proxies_priv：代理
```

1.1.5 简述你在工作中 MySQL 远程连接的方式？

```
mysql -uroot  -p
mysql -uroot -p -h 10.0.0.51 -P3306
sqlyog
navicat
```

1.1.6 请列举 MySQL 配置文件的读取顺序？

```
/etc/my.cnf /etc/mysql/my.cnf /usr/local/mysql/etc/my.cnf  ~/.my.cnf
[root@db01 ~]# cat .my.cnf
[client]
user=root
password=123
```

1.1.7 请简述 MySQL 配置文件结构？列举至少 5 项基础配置项？

```
标签项：
server: [mysqld] [mysqld_safe] [server]
```

```
clinet: [mysql] [mysqldump] [client]
user=mysql
basedir=/data/app/mysql
datadir=/data/3306/data
server_id=6
log_error=/data/3306/mysql.log
socket=/tmp/mysql.sock
```

1.1.8 请列举 MySQL 的启动和关闭的方式？

启动： sys-v、systemd → support-fils/mysql.server→mysqld_safe→mysqld

关闭：sys-v、systemd → support-fils/mysql.server ， shutdown，mysqladmin shutdown

1.1.9 请简述 MySQL 忘记本地 root 密码的处理方法和其中的原理？

1. 关数据库
2. 启动到安全模式： mysqld_safe --skip-grant-tables --skip-networking &
3. flush privileges; grant \ alter \set password
4. 重启数据库

1.1.10 你们公司使用多实例环境吗？在什么地方用的？

1. 测试环境
2. 分布式架构
3. 备用系统

1.1.11 如何查看数据库的当前连接情况？

```
db01 [(none)]>select * from information_schema.processlist;
db01 [(none)]>show full processlist;
```

1.1.12 简述数据库启动不了，你的排查过程？

（1）看日志 log_error

（2）[root@db01 ~]# mysqld --defaults-file=/etc/my.cnf &

1.1.13 升级至 5.7 、8.0，升级方式上有什么区别？8.0 要注意什么？

区别：升级到 5.7 ，需要执行 mysql_upgrade,升级到 8.0,直接用高版本软件挂低版本库启动。

升级到 8.0 以前，一定要提前备份。

升级时候建议，不要跨多版本。5.6 最新版升级到 5.7,5.7 最新版升级到 8.0

[root@db01 ~]# mysqlsh root:123@10.0.0.51:3306 -e "util.checkForServerUpgrade()"

1.1.14 mysql 查询表中数据中文字体乱码，原因可能是什么？如何修改？

字符集问题。

一般是客户端软件的字符集设定和服务器端不一致。

character_set_server=utf8mb4

Set names utf8mb4;

替换字符集：逻辑导出数据，库 alter，替换备份中的字符集，导入数据。

1.1.15 什么是实例？

公司 = boss+经理+员工+办公室

实例 = mysqld + master thread + 干活的 thread + 内存结构

1.1.16 简述 MySQL 的程序结构？

Mysqld = server 层（连接层+SQL 层） + 存储引擎层

1.1.17 简述一条 Select 语句的执行过程？

连接层：
连接协议
验证
连接线程
SQL 层：
语法、语义、权限
解析，预处理
优化
执行
日志记录
存储引擎层
文件系统数据读写
锁
事务
MVCC
自动故障恢复
ACID

1.1.18 请简述 MySQL 的逻辑结构和宏观物理结构？

逻辑：
库
表
宏观：
Myisam： myi myd frm
InnoDB: ibd 、frm、ibdata、ibtmp、undo、redo

1.1.19 请简述段、区、页的构成？

段：一个表就是一个段（分区表除外）。
区： 簇。一个区连续 64 个页。默认是 1M
页：page。默认是 16KB，连续 4 个 os block。

1.2 SQL 篇

1.2.1 请简述 select 语句的各个子句的执行顺序？

```
select from  where  group by having order by limit
```

1.2.2 请列举 SQL 语句的种类和代表命令？

DDL : create database/table drop database/table alter database/table
DCL : grant reovke
DML : select update delete insert

1.2.3 请简述 SQL_MODE 的作用？ONLY_FULL_GROUP_BY 是干什么用的？

规范 SQL 语句的。日期非 0、除数不为 0..,控制 group + 聚合函数。select list 中的列，要么在 group by ,要么聚合函数操作。

1.2.4 请简述 MySQL utf8 和 utf8mb4 区别？

utf8 ： 最大支持 3 个字节的字符
uft8mb4: 最大支持 4 字节长度字符。比如，emoji 表情字符。

1.2.5 请简述 tinyint int bigint 如何计算的存储位数？

tinyint : 1 字节长度。2^8 =256 , 0-255
int : 4 字节
bingint : 8 字节

1.2.6 请简述 char(10)和 varchar(10)的区别，生产如何选择？并阐述为什么？

char(10) 定长
varchar(10) 变长

1.2.7 请简述 datetime 和 timestamp 区别？

datetime 8 字节
timestamp 4 字节

1.2.8 请简述你们数据库开发过程，选择数据类型的规范是什么？

数字类型
整数
浮点数
字符串类型
char
varchar
enum
时间类型
datetime
timestamp
合适的、简短的、足够的。

1.2.9 请列举你了解的数据库的约束有哪些？他们的特点是什么？

pk
not null
unique
unsigned

1.2.10 请简述你们公司在 Schema 设计过程中有哪些开发规范？

名字： 小写、16 字符以内、业务相关、不要用数字开头、不要是预留字符串
设置字符集、存储引擎显式设置
数据类型：合适的、简短的、足够的
必须要有主键
每个列尽量非空，可以设置默认值
列名要有意义，不要预留字符，不要超过 16 字符
每个列、每个表都有注释

1.2.11 数据库范式有哪些？

第一范式（1NF）：要求数据库表的每一列都是不可分割的原子数据项。

举例说明：

学号	姓名	性别	家庭信息	学校信息
20150001	李白	男	3口人，北京	硕士，研二
20150002	杜甫	男	2口人，南京	本科，大四
20150003	王维	男	4口人，上海	本科，大三
20150004	白居易	男	3口人，河南	硕士，研一
20150005	刘禹锡	男	4口人，上海	本科，大一
20150006	李清照	女	5口人，西安	硕士，研三
20150007	苏轼	男	2口人，南京	大专，大二
20150008	屈原	男	4口人，吉林	博士，博五
20150009	陶渊明	男	1口人，长沙	博士，博三

在上面的表中，"家庭信息"和"学校信息"列均不满足原子性的要求，故不满足第一范式，调整如下：

学号	姓名	性别	家庭人口	户籍	学历	所在年级
20150001	李白	男	3口人	北京	硕士	研二
20150002	杜甫	男	2口人	南京	本科	大四
20150003	王维	男	4口人	上海	本科	大三
20150004	白居易	男	3口人	河南	硕士	研一
20150005	刘禹锡	男	4口人	上海	本科	大一
20150006	李清照	女	5口人	西安	硕士	研三
20150007	苏轼	男	2口人	南京	大专	大二
20150008	屈原	男	4口人	吉林	博士	博五
20150009	陶渊明	男	1口人	长沙	博士	博三

可见，调整后的每一列都是不可再分的，因此满足第一范式（1NF）；

第二范式（2NF）：在 1NF 的基础上，非码属性必须完全依赖于候选码（在 1NF 基础上消除非主属性对主码的部分函数依赖）
第二范式需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。

举例说明：

订单号	产品号	产品数量	产品折扣	产品价格	订单金额	订单时间
2008003	205	100	0.9	8.9	2870	20080103
2008003	206	200	0.8	9.9	2870	20080103
2008005	207	200	0.75	10	2000	20080203
2008006	207	400	0.85	12	4800	20080206
2008007	207	1000	0.88	14	14000	20080209
2008008	210	240	0.95	8	12255	20100423
2008008	211	300	0.75	8	12255	20100423
2008008	212	350	0.8	15.9	12255	20100423

在上图所示的情况中，同一个订单中可能包含不同的产品，因此主键必须是"订单号"和"产品号"联合组成，

但可以发现，产品数量、产品折扣、产品价格与"订单号"和"产品号"都相关，但是订单金额和订单时间仅与"订单号"相关，与"产品号"无关，

这样就不满足第二范式的要求，调整如下，需分成两个表：

订单号	产品号	产品数量	产品折扣	产品价格
2008003	205	100	0.9	8.9
2008003	206	200	0.8	9.9
2008005	207	200	0.75	10
2008006	207	400	0.85	12
2008007	207	1000	0.88	14
2008008	210	240	0.95	8
2008008	211	300	0.75	8
2008008	212	350	0.8	15.9

订单号	订单金额	订单时间
2008003	2870	20080103
2008003	2870	20080103
2008005	2000	20080203
2008006	4800	20080206
2008007	14000	20080209
2008008	12255	20100423
2008008	12255	20100423
2008008	12255	20100423

第三范式（3NF）：在 2NF 基础上，任何非主属性不依赖于其它非主属性（在 2NF 基础上消除传递依赖）
第三范式需要确保数据表中的每一列数据都和主键直接相关，而不能间接相关。

举例说明：

学号	姓名	性别	家庭人口	班主任姓名	班主任性别	班主任年龄
20150001	李白	男	3口人	陈洁	女	35
20150002	杜甫	男	2口人	陈洁	女	35
20150003	王维	男	4口人	陈洁	女	35
20150004	白居易	男	3口人	李丽	女	32
20150005	刘禹锡	男	4口人	李丽	女	32
20150006	李清照	女	5口人	王安	男	29
20150007	苏轼	男	2口人	南林	男	34
20150008	屈原	男	4口人	南林	男	34
20150009	陶渊明	男	1口人	王安	男	29

上表中，所有属性都完全依赖于学号，所以满足第二范式，但是"班主任性别"和"班主任年龄"直接依赖的是"班主任姓名"，

而不是主键"学号"，所以需做如下调整：

学号	姓名	性别	家庭人口	班主任姓名
20150001	李白	男	3口人	陈洁
20150002	杜甫	男	2口人	陈洁
20150003	王维	男	4口人	陈洁
20150004	白居易	男	3口人	李丽
20150005	刘禹锡	男	4口人	李丽
20150006	李清照	女	5口人	王安
20150007	苏轼	男	2口人	南林
20150008	屈原	男	4口人	南林
20150009	陶渊明	男	1口人	王安

班主任姓名	班主任性别	班主任年龄
陈洁	女	35
陈洁	女	35
陈洁	女	35
李丽	女	32
李丽	女	32
王安	男	29
南林	男	34
南林	男	34
王安	男	29

这样以来，就满足了第三范式的要求。

ps:如果把上表中的班主任姓名改成班主任教工号可能更确切，更符合实际情况，不过只要能理解就行。

1.2.12 你们公司 Online DDL 如何处理的？

一般 DDL 操作最好都采用 pt-osc 或 gh-ost 这样的工具来实施，并且实施之前务必要先检查当前目标表上是否有事务或大查询未结束，避免严重的 MDL 锁等待
除了 8.0 以上版本，除了追加式新增列（追加式）、表改名、新增虚拟列这三种支持 INSTANT 的操作可以直接跑 DDL，其余的都统统

采用 pt-osc/gh-ost 工具，相对更不容易出状。

执行 ALTER TABLE DDL 时，不要节外生枝指定 ALGORITHM=?, LOCK=?选项，因为 MySQL 会自行判断该采用哪种方式。本来可以 INPLACE 的，可能不小心给指定成 COPY 就悲剧了

1.2.13 5.6、5.7、8.0 在 Online DDL 的改变

ALTER TABLE 时 ALGORITHM 可以指定的几种方式：

COPY ，是指 DDL 时，会生成（临时）新表，将原表数据逐行拷贝到新表中，在此期间会阻塞 DML

INPLACE，无需拷贝全表数据到新表，但可能还是需要 IN-PLACE 方式（原地，无需生成新的临时表）重建整表。这种情况下，在 DDL 的初始准备和最后结束两个阶段时通常需要加排他 MDL 锁（metadata lock，元数据锁），除此外，DDL 期间不会阻塞 DML

INSTANT，只需修改数据字典中的元数据，无需拷贝数据也无需重建整表，同样，也无需加排他 MDL 锁，原表数据也不受影响。整个 DDL 过程几乎是瞬间完成的，也不会阻塞 DML。这个新特性是 8.0.12 引入的，再次感谢腾讯互娱 DBA 团队的贡献

1.2.14 简述 pt-osc 或者 gh-ost 第三方工具在处理 DDL 时的原理？

1. 查看是否有从节点
2. 查看是否有外键
3. 创建新表
4. 修改新表结构
5. 创建触发器，保证拷贝过程中的数据同步
6. 拷贝表数据
7. rename 表
8. 删除旧表
9. 删除触发器

1.2.15 请简述 drop table 、truncate table、delete from table 的区别？

都能删除表数据。

drop ： 表结构+表空间。

truncate ： 表数据，物理删除表段，清空所有数据页。立即释放磁盘空间。

delete ： 表数据，逻辑删除，打上删除标记。不立即释放磁盘空间。

1.2.16 请简述如何利用 update 替换 delete 语句实现伪删除

1. 添加状态列
2. delete 命令替换为 update
3. select 语句添加状态条件

1.2.17 如果要你规划一个 10 亿的大表，你有什么好的方案？

索引应用、存储问题。

1、 分区表
2、 分布式架构
3、 归档表

1.2.18 如果这张 10 亿单表已经存在了，想要删除 1000W 数据如何处理？

pt-archiver

1.2.19 生产中使用过分区表吗？你们使用的是什么分表策略？分区表有什么优势和劣势

range: id range 、 时间 range

list

1.2.20 请简述 group by 语句的执行原理

分组列值进行排序、去重复、聚合函数其他列聚合。

1.2.21 Where 和 having 语句的区别

where group by having

1.2.22 生产中进行数据库资产统计，都统统计什么？如何统计？

information_schema.tables columns 。

1.2.23 请介绍你常用的聚合函数及其作用

avg sum min max count group_concat

1.2.24 简述多表连接的方式

笛卡尔
内连接
外连接

1.3 索引篇

1.3.1 请列举 MySQL 索引的类型

Btree
Rtree
Fulltext
HASH

1.3.2 MySQL 为什么选择 Btree 查找算法

Balancer Tree,快速锁定范围。扫描次数是一样的。

1.3.3 MySQL 索引算法演变: B-tree、B+Tree、B*Tree 的区别？

是否有双向指针。

1.3.4 MySQL 聚簇索引如何构建 B+Tree

叶子节点：聚簇索引组织表，按照聚簇索引顺序，物理有序的存储的是表的数据行
枝节点： 保存下层叶子节点 ID 的范围+指针
根节点： 保存下层枝节点 ID 范围+指针

1.3.5 MySQL 辅助索引如何构建 B+Tree

叶子节点：ID+辅助索引列排序后生成
枝节点： 保存下层叶子节点辅助索引键值的范围+指针
根节点： 保存下层枝节点辅助索引键值范围+指针

1.3.6 什么是回表查询?如何减少回表？

通过辅助索引查找到 ID，回到聚簇索引查询。
随机 IO，IOPS，IO 增多。
联合索引（覆盖）、MRR。

1.3.7 索引树高度影响因素有哪些？

数据行。分库分表、归档表。
数据类型。
辅助索引键值长度。

主键不要使用 uuid

1.3.8 如何获取执行计划？如何理解执行计划的输出？

desc /explain

type : All 、index、range、ref、eq_ref、const(system)

key_len: 判断联合索引应用

extra : filesort temp BNL SNL BKA MRR ICP where index

1.3.9 联合索引应用有哪些原则？那些情况下不能覆盖到联合索引？

遵循最左原则。a,b,c abc ab a . 没有最左列条件。查询条件中不等值查询。

1.3.10 索引有哪些自优化能力？

ICP

MRR

CHANGE BUFFER

AHI

1.3.11 MySQL 8.0 在索引优化中添加的新特性有哪些？

invisible index

逆序索引 order a asc,b desc index(a asc ,b DESC)

1.3.12 请简述优化器算法：ICP、MRR 是如何实现的？

ICP a= b != c=

MRR : 减少回表。随机 IO 减少。同一个数据页读取次数。

1.3.13 请简述在多变连接查询中的多种算法：SNL、BNL、BKA？

画图说明。

1.3.14 请说明索引应用有哪些规范？

建索引原则

(1) 必须要有主键,如果没有可以做为主键条件的列,创建无关列

(2) 经常做为 where 条件列 order by group by join on, distinct 的条件(业务:产品功能+用户行为)

(3) 最好使用唯一值多的列作为索引,如果索引列重复值较多,可以考虑使用联合索引

(4) 列值长度较长的索引列,我们建议使用前缀索引.

(5) 降低索引条目,一方面不要创建没用索引,不常使用的索引清理,percona toolkit(xxxxx)

(6) 索引维护要避开业务繁忙期

不走索引的情况

(1) 没有查询条件，或者查询条件没有建立索引

(2) 查询结果集是原表中的大部分数据，应该是 15-25%以上

(3) 索引本身失效，统计信息不真实

(4) 查询条件使用函数在索引列上，或者对索引列进行运算，运算包括(+，-，*，/，！等)

(5) 隐式转换导致索引失效.这一点应当引起重视.也是开发中经常会犯的错误.

(6) <> ，not in 不走索引（辅助索引）

(7) like "%_" 百分号在最前面不走

1.4 存储引擎篇

1.4.1 MySQL 支持的存储引擎类型有哪些？

Show engines;

1.4.2 InnoDB 存储引擎的核心特性有哪些？InnoDB 和 MyISAM 的区别？

InnoDB：
事务
MVCC
行锁
热备
CSR
AHI
CHANGE BUFFER
DW

1.4.3 TokuDB 等存储引擎相较于 InnoDB 有什么优势？在什么场景应用？

TokuDB 的优点：1、高压缩比 2、高 insert 性能 3、增删字段秒级。
TokuDB 的缺点：1、cpu usr 态消耗高 2、响应时间变长。

1.4.4 MySQL 的碎片是如何产生的？你是如何处理的?

大批量的 delete。 Alter table t1 engine=innodb; 第一次整理效果很明显。逻辑导出、删除表、导入。

1.4.5 简述 InnoDB 物理存储结构？

表空间---》 段 、 区、页
日志文件： redo 、undo
临时表： ibtmp1

1.4.6 请介绍 InnoDB 的表空间类型和作用？

独立表空间： ibd 数据行和索引
共享表空间：系统数据
普通表空间：一般不用
临时表空间：undo+临时表
Undo 表空间：回滚日志

1.4.7 InnoDB 的共享表空间在不同版本有什么变化？

5.5 ： DW、DD、tmp、undo、CB、用户数据
5.6 ： DW、DD、tmp、undo、CB
5.7 ： DW、DD、undo、CB
8.0 ： 8.0.19（DW、CB）、8.0.20（CB）

1.4.8 简述表空间迁移的过程？

```
Target:
Create table t1;
Alter table discard tablespace;
Source:
Lock table t1 read;
scp  ibd IP:/dir
Target :
Chown  -R mysql.mysql /dir
Alter table t1 import tablespace;
Source:
Unlock tables;
```

1.4.9 共享表空间如何扩容？

```
Innodb_data_file_path=xx
```

1.4.10 如何独立 UnDO 表空间？

```
innodb_undo_directory
innodb_undo_tablespaces
innodb_undo_log_truncate
innodb_purge_rseg_truncate_frequency
```

1.4.11 什么是事务的 ACID？ACID 是如何保证的？

原子性 ：undo，redo

一致性 ： CR（redo、undo）、DW、MVCC、锁。

隔离性 ：隔离级别、MVCC、锁

持久性 ：redo

1.4.12 简述一个事务的完整工作流程？

一、Server 层阶段：

1、连接器： 提供连接协议、验证、提供连接会话线程。

2、分析器：分析器会对该语句分析，判断是否语法有错误、判断执行权限

3、解析器：根据统计信息，解析预处理。

4、优化器：代价优化、选择索引，生成执行计划。

5、执行器：根据优化器生成的执行计划，执行 SQL，调用插件存储引擎做后续处理。

二、InnoDB 引擎层阶段：

1、事务执行阶段：判断 BP 缓存，如果存在，根据 BTREE 查找数据。不存在则将需要的数据页加载到 BP

通过 B+Tree 读取到磁盘的索引页加载到 BP 缓冲池中 。

如何加载到 BP 缓冲池中：

首先 通过 space id 和 page no 哈希计算之后把 索引页加载到指定的 buffer pool instance 中

判断 free list 是否有空闲页可用(Innodb_buffer_pool_pages_free、 Innodb_buffer_pool_wait_free)，

没有则淘汰脏页或者 lru list 的 Old 页

把数据页 copy 到 free list 中，然后加载到 lru list 的 old 区的 midpoint（头部）

2、申请记录的锁。根据不同的隔离级别，对索引和记录添加指定的排他锁。如果别的事务有冲突锁，则等待。死锁出现则回滚事务。

3、写 undo，将修改前的数据值记录，生成事务编号，回滚指针。

4、 修改数据行，并记录 DB_ROLL_PTR、 DB_TRX_ID 、DB_ROW_ID，指向 UNDO 回滚日志，用于回滚数据和实现 MVCC 的多版本。

5、写 redo log buffer：

判断 redo log buffer 是否够用，redo log buffer 不够用就等待，体现在状态值 Innodb_log_waits 上。

在 BP 缓冲池 的 Lru list 中 old 区的 midpont 中对该数据页的行记录的字段值做更新操作。

并把修改之后的字段值写入到 redo log buffer 中

并给 LSN 加上当前 redo log 写入的长度(写入长度为 100 的 redo log，LSN 就会加上 100)

（因为 redo group commit 的原因，这次事务所产生的 redo log buffer 可能会跟随其它事务一同 flush 并且 sync 到磁盘上）

字段值在 BP 缓冲池更新成功以后，对应的数据页变为脏页。

server 写 binlog cache

按照 event 的格式,记录到 binlog_cache 中

写 change buffer：

把这条 sql， 需要在辅助索引上做的修改，写入到 change buffer page，等到下次有其他 sql 需要读取该二级索引时，再去与二级索引做 merge。

等待 commit 或者 rollback

如果执行 commit：

（1）事务的 COMMIT 分为 prepare 阶段与 commit 阶段

事务的 COMMIT 操作，在存储引擎层与 server 层之间采用的是内部 XA；

两阶段提交协议，保证两个事务的一致性,这里主要保证 redo log 和 binlog 的原子性；

（2）redo log prepare:

写入 redo log 处于 prepare 状态 并且写入事务的 xid；

将 redo log buffer 刷新到 redo log 磁盘文件中，用于崩溃恢复； #刷盘的方式由 innodb_flush_log_at_trx_commit 决定

（3）binlog write&fsync: 执行器把 binlog cache 里的完整事务和 redo log prepare 中的 XID 写入到 binlog 中

dump 线程会从 binlog_cache 里把 event 主动发送给 slave 的 I/O 线程，同时执行 fsync 刷盘(大事务的话这步非常耗时)，并清空 binlog cache。

事务中写 binlog 的部分日志：

200311 11:06:54 server id 123306 end_log_pos 439 CRC32 0x1c809de0 Xid = 614

COMMIT/*!*/;

binlog 刷盘的方式由 sync_binlog 决定；binlog 写入完成，事务就算是成功。

事务执行过程中，先把日志写到 binlog cache，事务提交的时候，再把 binlog cache 写到 binlog file 中。当 sync_binlog 为 1 时，当 binlog 落盘以后才会通知 dump thread 进行主从复制

（4）redo log commit: commit 阶段，由于之前该事务产生的 redo log 已经 sync 到磁盘了。所以这步只是在 redo log 里标记 commit，说明事务提交成功。

（5）事务提交成功，释放行记录持有的排他锁；

（6）当 binlog 和 redo log 都已经落盘以后，如果触发了 checkpoint 操作：

先把该脏页复制到分练习 doublewrite buffer 里，doublewrite buffer 里的刷新到共享表空间（ibdata），然后才是把脏页写入到磁盘中.

如果执行 rollback:

根据当前 DB_ROLL_PTR、 DB_TRX_ID 获取 UNDO 回滚日志，进行事务回滚。

1.4.13 什么是 MVCC？如何保证的？

多版本并发控制。

RC、RR 保证的机制。通过不同的 read-view 保证的，一致性非锁定读。

undo 快照，隐藏字段 DB_ROLL_PTR、 DB_TRX_ID 、DB_ROW_ID。

1.4.14 请简述 CSR 自动故障恢复的过程？

前滚： redo

回滚： undo

DW ： 原数据页完整性

1.4.15 什么是 LSN？作用是什么？

查看 LSN:

```
    show engine innodb status
Log sequence number 2687274848548    #当前系统最大的 LSN 号
    Log flushed up to 2687274848516      #当前已经写入 redo 日志文件的 LSN
    Pages flushed up to 2687273963960    #已经将更改写入脏页的 lsn 号
    Last checkpoint at 2687273963960     #就是系统最后一次刷新 buffer pool 脏中页数据到磁盘的 checkpoint
```

以上 4 个 LSN 是递减的，即： LSN1>=LSN2>=LSN3>=LSN4.

每个数据页有 2 个 LSN：重做日志有 LSN，checkpoint 有 LSN。

LSN（log sequence number）日志序列号，5.6.3 之后占用 8 字节，LSN 主要用于发生 crash 时对数据进行 recovery，LSN 是一个一直递增的整型数字，表示事务写入到日志的字节总量。

LSN 不仅只存在于重做日志中，在每个数据页头部也会有对应的 LSN 号，该 LSN 记录当前页最后一次修改的 LSN 号，用于在 recovery 时对比重做日志 LSN 号决定是否对该页进行恢复数据。

checkpoint 也是有 LSN 号记录的，LSN 号串联起一个事务开始到恢复的过程。

1.4.16 什么是 WAL 机制？

redo 日志优先落盘。达到数据持久化的目的。

1.4.17 简述 Checkpoint 的功能？

作用：

1、缩短数据库的恢复时间；

2、缓冲池不够用时，将脏页刷新到磁盘；

3、重做日志不可用时，刷新脏页。

刷新条件：

1、MasterThread Checkpoint： 异步刷新，每秒或每 10 秒从缓冲池脏页列表刷新一定比例的页回磁盘。异步刷新，即此时 InnoDB 存储引擎可以进行其他操作，用户查询线程不会受阻。

2、 FLUSH_LRU_LIST Checkpoint：

InnoDB 存储引擎需要保证 LRU 列表中差不多有 N 个空闲页可供使用。

mysql5.6 之后，也就是 Innodb1.2.x 开始，这个检查放在了单独的线程（Page Cleaner）中进行。

设置参数：innodb_lru_scan_dept：控制 LRU 列表中可用页的数量，该值默认 1024

3、Async/Sync Flush Checkpoint：

指重做日志不可用的情况，需要强制刷新页回磁盘，此时的页时脏页列表选取的。

这种情况是保证重做日志的可用性，说白了就是，重做日志中可以循环覆盖的部分空间太少了，换种说法，就是极短时间内产生了大量的 redo log。

计算公式：

写入日志的 LSN:redo_lsn

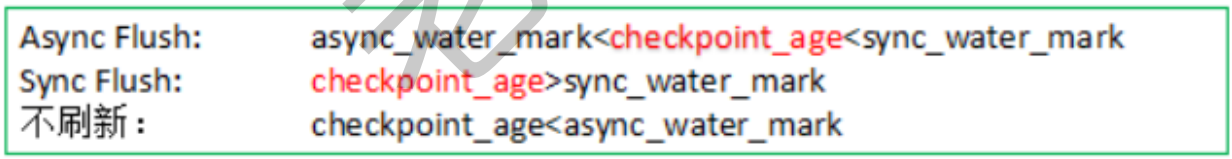
刷新回磁盘的最新页 LSN:checkpoint_lsn

定义:

checkpoint_age = redo_lsn - checkpoint_lsn

async_water_mark = 75% * total_redo_file_size

sync_water_mark = 90% * total_redo_file_size



4、 Dirty Page too much Checkpoint

即脏页太多，强制 checkpoint.保证缓冲池有足够可用的页。

参数设置：innodb_max_dirty_pages_pct = 75

1.4.18 简述 MySQL 的隔离级别？什么是脏读、不可重复读、幻读？

RU ： 脏读 、不可重复、幻读

RC ： 不可重复、幻读

RR ： 幻读 ， GAP 、next-lock

SR ：

1.4.19 简述 MySQL 中锁的种类及作用？

```
latch: rw-lock、 mutex
schema ： system lock、FTWRL
table ： MDL、table lock
record ： 行锁 、 gap、nextlock
```

1.4.20 简述 double write 的作用？

```
保证数据页的完整性。
```

1.4.21 简述 AHI 的作用？

```
索引的索引。快速从 BP 中找到索引页。
```

1.4.22 简述 Change buffer 作用？

```
例如： insert 数据，更新辅助索引，不是实时更新，而是更新到 changebuffer 中，用的时候 merge
```

1.4.23 如何监控 MySQL 的锁状态？

```
show status like 'innodb_rows_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;
```

1.4.24 请列举 InnoDB 核心参数有哪些？ 并介绍其作用？

```
innodb_buffer_pool_size
innodb_log_buffer_size
innodb_flush_log_at_trx_commit
innodb_flush_method
```

1.5 日志管理篇

1.5.1 错误日志功能，如何配置错误日志，如何查看？

```
log_error=
[error]
```

1.5.2 如何配置二进制日志？

```
server_id
log_bin
binlog_format
sync_binlog
gtid_mode
enforce_gtid_consistency
log_slave_updates
expire_logs_days
```

1.5.3 二进制日志如何查看事件？

```
show binlog events in
```

1.5.4 二进制日志有哪些格式？有什么区别？你们公司用什么格式？

```
DML
SBR : statement
RBR : ROW
MBR
```

1.5.5 什么是双一标准？

```
redo   : innodb_flush_log_at_trx_commit
binlog : sync_binlog
```

1.5.6 什么是 2PC？

```
prepare: redo sync prepare
binlog : commit &sync
commit : redo commit
```

1.5.7 你是如何截取二进制日志的？传统、GTID 有什么不同点？

```
mysqlbinlog start-position stop-position  .include-gtids exclude-gtids  ,skip-gtids
```

1.5.8 如何配置慢日志？

```
slow_log
slow_log_file
log_queries_not_using_indexes
long_query_time
```

1.5.9 慢日志如何处理？慢日志在什么时候使用？

```
mysqldumpslow
pt-query-digest
```

1.6 备份恢复篇

1.6.1 你在备份这块都做过什么具体工作？

```
1、 备份策略
2、 备份检查
3、 恢复演练
4、 故障快速恢复
5、 迁移
```

1.6.2 你们公司的备份策略是什么样子？

```
1、 工具：逻辑、物理
2、 周期：每天、每周
3、 类型：full 、inc
```

1.6.3 你们公司使用什么工具备份。

```
mdp
```

```
xbk
mydumper
主从
延时从
```

1.6.4 请介绍 mysqldump 核心参数：--master-data --single-transaction 功能

--maste-data=2

1.自动锁表（非 InnoDB），global read lock

2. 记录 binlog 位置

--single-transaction

对 InnoDB 表开启快照备份。

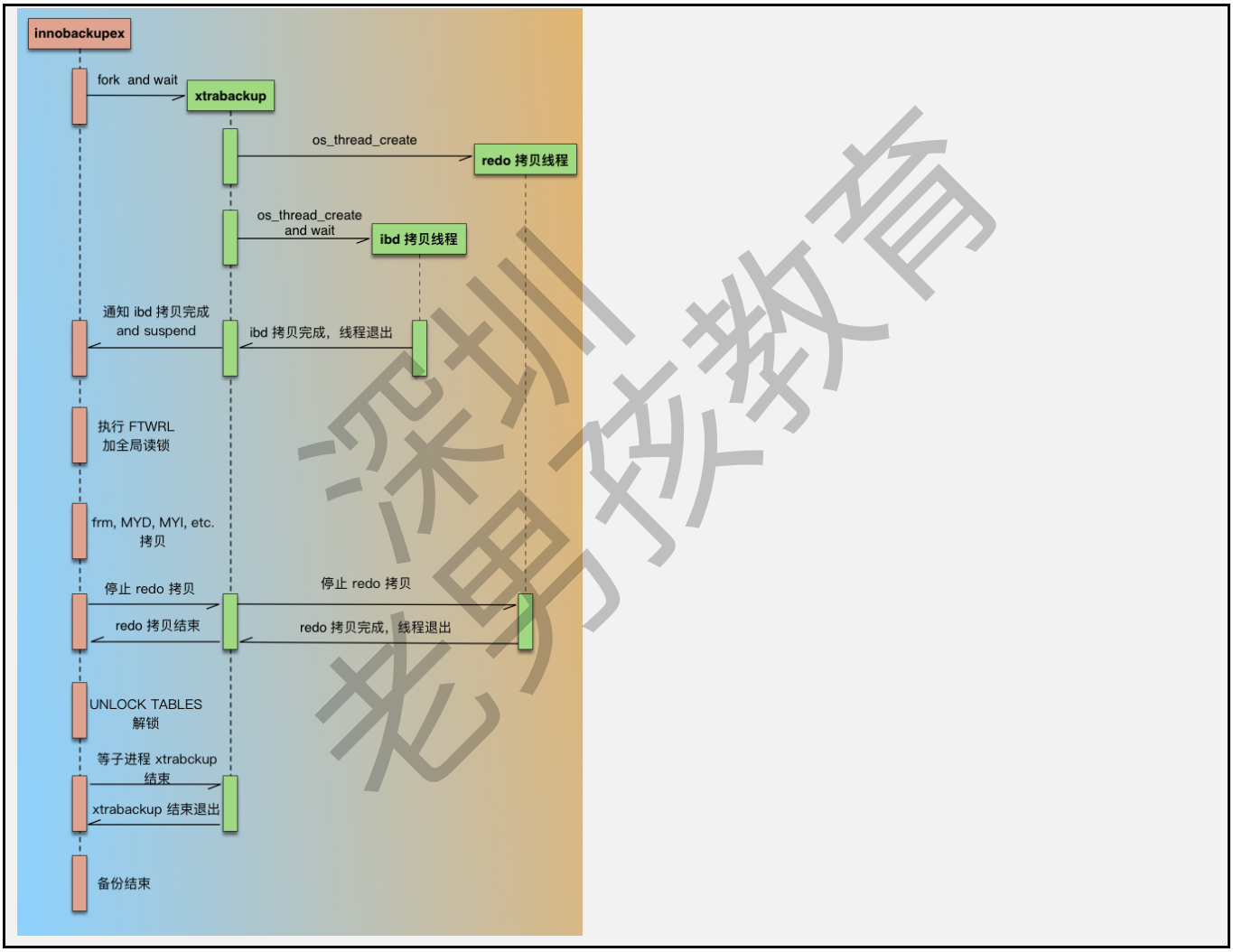
1.6.5 请简述 mysqludmp 备份原理

1、 提取 InnoDB 数据快照。转换为 SQL，存储到 sql 文件中

2、 非 InnoDB 表，全局锁表，转换为 SQL，转存到 SQL 文件

3、 备份方式 create db create table insert

1.6.6 请介绍 xtrabackup 工具的备份原理



1. innobackupex 在启动后，会先 fork 一个进程，启动 xtrabackup 进程，然后就等待 xtrabackup 备份完 ibd 数据文件；

2. xtrabackup 在备份 InnoDB 相关数据时，是有 2 种线程的，1 种是 redo 拷贝线程，负责拷贝 redo 文件，1 种是 ibd 拷贝线程，负责拷贝 ibd 文件；redo 拷贝线程只有一个，在 ibd 拷贝线程之前启动，在 ibd 线程结束后结束。xtrabackup 进程开始执行后，先启动 redo 拷贝线程,从最新的 checkpoint 点开始顺序拷贝 redo 日志;然后再启动 ibd 数据拷贝线程,在 xtrabackup 拷贝 ibd 过程中，innobackupex 进程一直处于等待状态（等待文件被创建）。

3. xtrabackup 拷贝完成 idb 后，通知 innobackupex（通过创建文件），同时自己进入等待（redo 线程仍然继续拷贝）；

4. innobackupex 收到 xtrabackup 通知后，执行 FLUSH TABLES WITH READ LOCK (FTWRL)，取得一致性位点，然后开始备份非 InnoDB 文件（包括 frm、MYD、MYI、CSV、opt、par 等）。拷贝非 InnoDB 文件过程中，因为数据库处于全局只读状态，如果在业务的主库

备份的话，要特别小心，非 InnoDB 表（主要是 MyISAM）比较多的话整库只读时间就会比较长，这个影响一定要评估到。

5. 当 innobackupex 拷贝完所有非 InnoDB 表文件后，通知 xtrabackup（通过删文件），同时自己进入等待（等待另一个文件被创建）；

6. xtrabackup 收到 innobackupex 备份完非 InnoDB 通知后，就停止 redo 拷贝线程，然后通知 innobackupex redo log 拷贝完成（通过创建文件）；

7. innobackupex 收到 redo 备份完成通知后，就开始解锁，执行 UNLOCK TABLES；

8. 最后 innobackupex 和 xtrabackup 进程各自完成收尾工作，如资源的释放、写备份元数据信息等，innobackupex 等待 xtrabackup 子进程结束后退出。

增量备份

PXB 是支持增量备份的，但是只能对 InnoDB 做增量，InnoDB 每个 page 有个 LSN 号，LSN 是全局递增的，page 被更改时会记录当前的 LSN 号，page 中的 LSN 越大，说明当前 page 越新（最近被更新）。每次备份会记录当前备份到的 LSN（xtrabackup_checkpoints 文件中），增量备份就是只拷贝 LSN 大于上次备份的 page，比上次备份小的跳过，每个 ibd 文件最终备份出来的是增量 delta 文件。MyISAM 是没有增量的机制的，每次增量备份都是全部拷贝的。

增量备份过程和全量备份一样，只是在 ibd 文件拷贝上有不同。

恢复过程

如果看恢复备份集的日志，会发现和 mysqld 启动时非常相似，其实备份集的恢复就是类似 mysqld crash 后，做一次 crash recover。恢复的目的是把备份集中的数据恢复到一个一致性位点，所谓一致就是指原数据库某一时间点各引擎数据的状态，比如 MyISAM 中的数据对应的是 15:00 时间点的，InnoDB 中的数据对应的是 15:20 的，这种状态的数据就是不一致的。PXB 备份集对应的一致点，就是备份时 FTWRL 的时间点，恢复出来的数据，就对应原数据库 FTWRL 时的状态。

因为备份时 FTWRL 后，数据库是处于只读的，非 InnoDB 数据是在持有全局读锁情况下拷贝的，所以非 InnoDB 数据本身就对应 FTWRL 时间点；InnoDB 的 ibd 文件拷贝是在 FTWRL 前做的，拷贝出来的不同 ibd 文件最后更新时间点是不一样的，这种状态的 ibd 文件是不能直接用的，但是 redo log 是从备份开始一直持续拷贝的，最后的 redo 日志点是在持有 FTWRL 后取得的，所以最终通过 redo 应用后的 ibd 数据时间点也是和 FTWRL 一致的。

所以恢复过程只涉及 InnoDB 文件的恢复，非 InnoDB 数据是不动的。备份恢复完成后，就可以把数据文件拷贝到对应的目录，然后通过 mysqld 来启动了。

1.6.7 同时晚上 23：00 开始备份，23:30 分结束。Mdp 和 xbk 两个工具理论上能将数据恢复至几点？

mdp: 23:00

xbk: 23:30

1.6.8 Xtrabackup 的增量备份是如何实现的？

1. full checkpoint : LSN

2. inc1 --incremental-basedir=xx 检查 lastlsn,备份变化的数据页。

1.6.9 Xtrabackup 增量备份恢复要注意什么？

1、 prepare

2、 merging inc to basefull

3、 除了最后一次 inc redo-only,其他所有备份

1.6.10 Mysqldump 和 xtrabackup 备份如何实现基于时间点的恢复？

--master-data=2,结合 binlog

xtrabackup_binlog_info,结合 binlog

1.6.11 请列举你理解的数据库数据损坏场景，然后针对性的提出最佳的恢复方案（假设具有多种类型备份）？

物理损坏： 主从、高可用

逻辑损坏： 备份、binlog2sql, myflashback、延时从

1.6.12 如何实现分库分表备份数据库中的表？

mydumper

information_schem.table concat 拼接备份语句

shell

1.7 主从复制篇

1.7.1 如何为运行了 2 年的数据库，构建一个从库。请说明步骤？是否需要停主库？

1、 主库开 binlog，建立复制用户

2、 server_id server_uuid

3、 初始化新从节点

4、 备份主库数据，恢复到从库。

5、 开启主从 change master to start slave

1.7.2 简述主从复制原理？

1、 线程： dump IO SQL

2、 文件： binlog relaylog master.info relay-log.info

3、 过程

（a） change master ,start slave; IO、SQL

（b） 连接主库，dump

（c） 主库事务，binlog 更新，dump 实时监控 binlog 变化，通知 IO

（d） IO 根据 master.info ,请求日志

（e） dump 发送日志。

（f） io 接收日志，记录到 relaylog 中，更新 master.info

（g） SQL 读取 relay-log.info，回放新的 relaylog

（h） 更新 relay-log.info

（i） relaylog purge 线程自动清理 relaylog。

1.7.3 如何监控主从复制，请说明监控要点？

show slave status \G

线程状态、报错信息、主从延时情况。

1.7.4 请简述你遇到过的主从复制故障？如何监控、分析、规避、处理？

5.1.1 连接主库: connecting

#可能原因

连接信息有误。

网络故障。

防火墙。

最大连接数上线。

排查方法:

[root@db01 data]# mysql -urepl -p123 -h10.0.0.51 -P 3307

```
# 处理方法：
mysql -S /data/3308/mysql.sock -e "stop slave;reset slave all;"
CHANGE MASTER TO
MASTER_HOST='10.0.0.51',
MASTER_USER='repl',
MASTER_PASSWORD='123',
MASTER_PORT=3307,
MASTER_LOG_FILE='mysql-bin.000004',
MASTER_LOG_POS=674,
MASTER_CONNECT_RETRY=10;
start slave;
```

5.1.2 请求日志：NO

主库日志损坏。

日志起点写错。

server_id 重复

```
# 排查方法
show slave status \G
Master_Log_File: mysql-bin.000004
Read_Master_Log_Pos: 674
Last_IO_Error: xxxx
```

5.2 SQL 线程故障： 回放中继日志

5.2.1 中继日志损坏

```
# 1. 从库 停 SQL 线程
stop slave  sql_thread ;
# 2. 主库发生新的操作
create database test1;
# 3. 从库删除 relaylog
rm -rf /data/3308/data/db01-relay-bin.00000*
# 4. 启动 SQL 线程
start  slave  sql_thread ;
```

修复：

```
1. cat /data/3308/data/relay-log.info   ----> binlog 位置点
2. 重构
stop slave ;
reset slave all;
CHANGE MASTER TO
MASTER_HOST='10.0.0.51',
MASTER_USER='repl',
MASTER_PASSWORD='123',
MASTER_PORT=3307,
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=154,
```

```
MASTER_CONNECT_RETRY=10;
start slave;
5.2.2 日志回放失败（执行不了 SQL）
# 1. 修改的对象不存在
# 2. 创建的对象已存在
# 3. 约束冲突
# 4. 主从配置不同
# 5. SQL_MODE 不兼容
# 6. 主从版本差异

方法 0：
从库逆反操作。

方法一：
stop slave;
set global sql_slave_skip_counter = 1;
#将同步指针向下移动一个，如果多次不同步，可以重复操作。
start slave;

方法二：
/etc/my.cnf
slave-skip-errors = 1032,1062,1007
常见错误代码:
1007:对象已存在
1032:无法执行 DML
1062:主键冲突,或约束冲突
但是，以上操作有时是有风险的，最安全的做法就是重新构建主从。把握一个原则,一切以主库为主.

方法三： PT 工具
pt-table-checksum
pt-table-sync

方法四：从库只读

mysql> select @@read_only;
mysql> select @@super_read_only;
```

1.7.5 请简述哪些情况会导致主从延时？如何监控、分析、规避、处理？

```
6. 主从复制延时
主库做了操作，从库很久才回放
6.1 主库方面
6.1.1 提供 binlog
binlog 日志文件落地不及时。sync_binlog=1
```

6.1.2 传输 binlog

claassic 模式（无 GTID），dump 线程传输日志是串行的。主库可以并行多个事务。

大事务、并发事务量大。都会导致较高延时。

5.6 版本加入了 GTID 功能，在传输时就可以并行传输日志了。

5.7 版本即使没开 GTID，会自动生成 Anonymous_Gtid。

6.2 从库方面

6.2.1 relay 落地

6.2.2 SQL 回放

单一 SQL 线程，只能串行回放 relaylog。主库可以并发事务，并行传输日志，回放时是串行的。

如果 大事务，并发事务量大。都会导致较高回放延时。

5.6 版本 GTID 模式下，可以开启多个 SQL 线程。但是，5.6 多 SQL 回放时，只能针对不同 database 并行回放。

5.7 版本中 GTID 模式下，可以开启多个 SQL 线程，真正实现了并性回放（MTS）

6.3 外部因素

网络慢

主从配置相差大

1.7.6 请介绍你了解过的主从复制模型？

1 主多从 、双主、级联、延时从库、过滤、半同步

1.7.7 传统的主从复制是异步还是同步？会不会出现主从不一致？出现有什么好的方法解决或预防？

异步。有可能出现不一致。PT 工具。

1.7.8 延时从库的实现原理？主要可以解决什么问题？如何使用延时从库？

SQL 线程延时。逻辑损坏。人工模拟 SQL 线程工作。回放 relay 到故障之前。

1.7.9 过滤复制实现原理？你们公司过滤复制架构如何设计的？

主库： 记录 binlog 从库： 回放。

1.7.10 半同步复制实现原理？5.6 版本和 5.7 版本半同步有什么区别？

after_commit

after_sync

1.7.11 请简述 MGR 的工作原理？Paxos 协议工作原理？

事务提交之前，进行投票,如果半数以上通过，提交事务。参考吃饭的例子。表决、投票、执行。

1.8 架构篇

1.8.1 请介绍一下你们公司的数据库架构？

高可用： MHA、PXC、MGC、MGR

读写分离：ProxySQL 、Maxscale、mysql-router、mycat

分布式架构： Mycat 、sharding-jdbc

1.8.2 请介绍你熟悉的高可用解决方案？

MHA 架构： Manager node

1.8.3 请简述 MHA 的搭建过程？

1. 1 主 2 从独立节点、GTID

2. 互信

3. 软连接

4. 建用户
5. 软件安装（perl\mananger\node）
6. 启动检查： ssh repl
7. 配置文件
8. 启动
9. vip\binlogserver\sendreport

1.8.4 请介绍 MHA 重要的脚本及作用？

Manager 工具包主要包括以下几个工具：

masterha_manger	启动 MHA
masterha_check_ssh	检查 MHA 的 SSH 配置状况
masterha_check_repl	检查 MySQL 复制状况
masterha_master_monitor	检测 master 是否宕机
masterha_check_status	检测当前 MHA 运行状态
masterha_master_switch	控制故障转移（自动或者手动）
masterha_conf_host	添加或删除配置的 server 信息

Node 工具包主要包括以下几个工具：

这些工具通常由 MHA Manager 的脚本触发，无需人为操作

save_binary_logs	保存和复制 master 的二进制日志
apply_diff_relay_logs	识别差异的中继日志事件并将其差异的事件应用于其他的
purge_relay_logs	清除中继日志（不会阻塞 SQL 线程）

1.8.5 请简述 MHA 高可用架构 FAILOVER 原理？

1、 masterha_master_monitor 监控主节点，4 次机会
2、 选主
（1） 日志量
（2） candidate_master=1
（3） 标签顺序
3、 数据补偿
ssh
主从身份切换
vip
binlog_server
sendreport

1.8.6 请介绍你在维护公司 MHA 架构时主要做了哪些工作？

1、 定期的切换。
2、 故障监控处理
3、 主从优化，降低延时。

1.8.7 是否使用过分布式数据库架构？请说明你时如何设计的？

Mycat 架构描述。水平拆分。分片策略

1.8.8 简介 PXC 高可用集群的工作原理

PXC 原理：
PXC 最常使用以下 4 个端口号：

3306-数据库对外服务的端口号。
4444-请求 SST 的端口（SST 是指数据库一个备份全量文件的传输。）
4567-组成员之间进行沟通的一个端口号
4568-用于传输 IST（相对于 SST 来说的一个增量）
PXC 的工作流程：
首先客户端先发起一个事务，该事务先在本地执行，执行完成之后就要发起对事务的提交操作了。在提交之前需要将产生的复制写集广播出去，然后获取到一个全局的事务 ID 号，一并传送到另一个节点上面。通过合并数据之后，发现没有冲突数据，执行 apply_cd 和 commit_cb 动作，否则就需要取消此次事务的操作。而当前 server 节点通过验证之后，执行提交操作，并返回 OK，如果验证没通过，则执行回滚。当然在生产中至少要有 3 个节点的集群环境，如果其中一个节点没有验证通过，出现了数据冲突，那么此时采取的方式就是讲出现不一致的节点踢出集群环境，而且它自己会执行 shutdown 命令，自动关机。

1.9 优化篇

1.9.1 你对 MySQL 做过哪些优化？

1、 主机、存储、网络、OS
2、 实例： 参数
3、 应用： SQL、索引、锁
4、 架构： 高可用、读写分离、分布式

1.9.2 数据库 CPU 爆满，你有什么排查思路？

1、 top -Hp 线程
2、 sys us wait
3、 show prcocsslist
4、 lock
5、 thread SQL 、index

1.9.3 你平常如何监控锁状态的？

```
show status like 'innodb_rows_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;
```

1.9.4 你平常都调整过什么参数？每个参数调整的依据是什么？

内存
连接
IO
双一
刷写策略

1.9.5 你是如何定位 SQL 语句问题，如何进行优化的？

```
show processlist  -→ explain -→ lock
slow ---》 explain -→ lock
```

1.9.6 简述索引优化的规范？如何建索引？哪些情况不会走索引？

建索引原则
(1) 必须要有主键,如果没有可以做为主键条件的列,创建无关列

(2) 经常做为 where 条件列 order by group by join on, distinct 的条件(业务:产品功能+用户行为)
(3) 最好使用唯一值多的列作为索引,如果索引列重复值较多,可以考虑使用联合索引
(4) 列值长度较长的索引列,我们建议使用前缀索引.
(5) 降低索引条目,一方面不要创建没用索引,不常使用的索引清理,percona toolkit(xxxxx)
(6) 索引维护要避开业务繁忙期
不走索引的情况
没有查询条件，或者查询条件没有建立索引
查询结果集是原表中的大部分数据，应该是 15-25%以上
索引本身失效，统计信息不真实
查询条件使用函数在索引列上，或者对索引列进行运算，运算包括(+，-，*，/，！等)
隐式转换导致索引失效.这一点应当引起重视.也是开发中经常会犯的错误.
<>，not in 不走索引（辅助索引）
like "%_" 百分号在最前面不走

1.9.7 请介绍你使用过的 PT 工具

pt-osc pt-archiver pt-query-digest pt-主从 pt-kill

1.9.8 请介绍你们公司如何监控 MySQL？都监控哪些项目？

zabbix
内存、事务、线程、QPS、TPS、锁、等待、参数的评估指标。

1.10 非技术篇

1.10.1 你觉得 DBA 应该具备哪些职业素养？

人品、严谨、细心、善于底层学习、善于分享、承担压力、知识面广、差异化发展。

1.10.2 你最擅长的是 MySQL 哪部分？

结合咱们课程体系： 架构设计、优化、处理问题、底层原理、、

1.10.3 喜欢 MySQL 吗？平常的学习渠道有哪些？

热衷。数据库大会、论坛、网上的论坛博客、书籍。

1.10.4 你觉得 DBA 需要掌握哪些知识？

结合咱们课程体系。

1.10.5 简述你对于 RDBMS NoSQL NewSQL 的看法？

三代数据库架构。

1.10.6 你在之前公司每天都干些什么活？

1、 巡检
2、 备份
3、 配合开发，代码审计、审核、改写、设计
4、 监控、优化
5、 架构选型、调研、测试
6、 应急处理

1.10.7 你上家公司的架构什么样的？ 数据量多大？ QPS 、TPS 多少？

高可用+读写分离
高可用+分布式

TB 以上 ： 20000-50000 ， 800-1500

1.11 故障篇

1.11.1 安装部署时遇到过什么故障？

版本不兼容
配置文件
目录没创建
权限
环境

1.11.2 数据库版本升级时遇到过什么故障？

5.5 ---》 5.7，5.6 再 5.7

1.11.3 数据库连接不上，可能的原因时什么，如何排查？

网路、用户、端口、防火墙、连接数。mysql 命令模拟登陆。

1.11.4 连接数设置不生效，最多为 214，是什么原因？

limits.cnf

1.11.5 http 409 错误,达到连接数上限。可能是什么原因导致？

连接数上限。

1.11.6 在做 DDL 操作时，数据库 hang 住，原因是什么？

MDL。

1.11.7 一条 SQL 语句，昨天执行很快，突然变慢可能是什么原因？

统计信息不真实，索引失效。

1.11.8 Ibdata1 共享表空间文件损坏，导致数据库无法启动，备份也失效，有什么好的解决思路？

表空间迁移。

1.11.9 基于 binlog+gtid 方式截取的日志无法正常恢复，是什么原因导致？怎么解决？

gtid 幂等性。--skip-gtids

1.11.10 使用 mysqldump 方式构建主从，添加了-- set-gtid-purged =off 导致主从构建失败？

备份不会有 set gtid_purge=

1.11.11 由于宕机，导致主从数据不一致，如何解决？

pt 工具

1.11.12 Zabbix 监控 2000+台主机，监控显示缓慢，每隔三四个月重新搭建，存储空间经常被占满

mysql 5.5 共享表空间。 percona 5.7 + tokudb。

1.11.13 有规律的一段时间，会产生性能低谷

raid 定期的充放电。自动巡查功能。

1.11.14 过度条带化导致的性能问题

LVM 条带化+ raid5 导致 IOPS 过高

1.11.15 MySQL 连接长时间(7200 和 1200 秒)无法释放

keepalive 、 lvs 。 pt-kill

1.11.16 开 QC ， 导致性能降低。 QPS ， TPS 降低。

QC 在分区表中是不起作用的。

1.12 NoSQL 篇

1.12.1 请简述 redis 安全模式？

bindIp
requirepass

1.12.2 请简述 redis 持久化方式？

rdb 快照。
aof binlog。

1.12.3 请简述 redis 数据类型及应用场景？

string 计数器
hash 数据库缓存
list 朋友圈
set 共同好友、二度好友
sortset 排行榜

1.12.4 请简述 redis 事务和 MySQL 的区别？

redis 队列实现的事务
mysql 事务日志

1.12.5 请简述 redis 主从原理？

1. 副本库通过 slaveof 10.0.0.51 6379 命令,连接主库,并发送 SYNC 给主库
2. 主库收到 SYNC,会立即触发 BGSAVE,后台保存 RDB,发送给副本库
3. 副本库接收后会应用 RDB 快照
4. 主库会陆续将中间产生的新的操作,保存并发送给副本库
5. 到此,我们主复制集就正常工作了
6. 再此以后,主库只要发生新的操作,都会以命令传播的形式自动发送给副本库.
7. 所有复制相关信息,从 info 信息中都可以查到.即使重启任何节点,他的主从关系依然都在.
8. 如果发生主从关系断开时,从库数据没有任何损坏,在下次重连之后,从库发送 PSYNC 给主库
9. 主库只会将从库缺失部分的数据同步给从库应用,达到快速恢复主从的目的

1.12.6 请简述 redis sentinel 高可用实现原理？

监控
选主
切换
自动加入集群

1.12.7 请简述 redis cluster 分布式集群实现原理？

1、redis 会有多组分片构成（3 组）
2、redis cluster 使用固定个数的 slot 存储数据（一共 16384slot）
3、每组分片分得 1/3 slot 个数（0-5500 5501-11000 11001-16383）
4、基于 CRC16(key) % 16384 ====》值 （槽位号）。

1.12.8 请介绍 MongoDB 用户管理和 MySQL 的区别？

验证库。use test 登录 验证库。role。

1.12.9 请介绍 MongoDB 复制集技术的工作原理？

3 节点。raft 协议。自动监控、选主、切换。

1.12.10 请介绍 MongoDB Sharding Cluster 架构原理？

mongos

configserver

mongod

1.12.11 请介绍 MongoDB 如何实现自动分片的？

Mongos balancer.

1.12.12 请介绍 MongoDB 的 oplog 作用是什么？

binlog。

复制集。

备份基于时间点回复。