面经预热Day16（数据库专题）

1、说一说你了解的MVCC机制

MVCC（Multi-Version Concurrency Control）多版本并发控制，用于管理多个事务同时访问和修改数据库的数据，而不会导致数据不一致或冲突。MVCC的核心思想是每个事务在数据库中看到的数据版本是事务开始时的一个快照，而不是实际的最新版本，这使得多个事务可以并发执行，而不会互相干扰。

MySQL的事务有ACID四大特性，其中的隔离性可以通过锁和MVCC来实现，MVCC适合在一些锁性能较差的情况下使用，提高效率。

如何实现

每一个UndoLog日志中都有一个roll_pointer（回滚指针）用于指向上一个版本的Undo Log。这样对于每一条记录就会构成一个版本链，用于记录所有的修改，每一次进行新的修改后，新的Undo Log会放在版本链的头部。

在我们进行查询的时候应该查询哪个版本呢？这时候就可以通过ReadView来实现。

在事务SELECT查询数据时，就会构造一个ReadView，它包含了版本链的统计信息

1. m_ids 当前活跃的所有事务id（所有未提交的事务）
2. min_trx_id 版本链尾的id
3. max_trx_id 下一个将要分配的事务id（版本链头事务id+1）
4. creator_trx_id 创建这个ReadView的事务的id

查询规则：

该版本是否为当前事务创建 (读取自己修改的数据)，如果是就返回，否则进入下一个判断

该版本的事务id是否小于min_trx_id (在ReadView创建之前，数据已经提交)，可以直接访问

该版本的事务id是否大于max_trx_id (在ReadView创建后，该版本才开启)，不能被访问

该版本事务id在[min_trx_id,max_trx_id]之间，则判断当前版本事务id是否在m_ids中，如果不在，说明事务已经提交可以访问，否则不能访问。

2、索引失效的场景有哪些?

- OR 条件： 当查询中使用多个 OR 条件时，如果这些条件不涉及同一列，索引可能无法有效使用。数据库可能会选择全表扫描而不是使用多个索引。
- 对列进行类型转换： 如果在查询中对列进行类型转换，例如将字符列转换为数字或日期，索引可能会失效。
- 使用通配符前缀搜索： 在使用通配符前缀（如LIKE 'prefix%'）进行搜索时，大多数索引无法使用，因为索引通常是按照列的完整值进行排序的。
- 不等号条件： 当查询中包含不等号条件（如>,<,>=,<=）时，索引可能会失效。通常情况下，索引只能用于等值比较。
- 表连接中的列类型不匹配： 如果在连接操作中涉及的两个表的列类型不匹配，索引可能会失效。例如，一个表的列是整数，另一个表的列是字符，连接时可能会导致索引失效。

3、MySQL的执行引擎有哪些?

主要有MyISAM、InnoDB、Memery等引擎

- InnoDB 引擎提供了对事务ACID的支持，还提供了行级锁和外键的约束
- MyISAM 引擎不支持事务，也不支持行级锁和外键约束
- Memery 就是将数据放在内存中，数据处理速度很快，但是安全性不高

4、MySQL日志文件有哪几种

- Undo Log是Inn哦DB存储引擎层生成的日志，实现了事务中的原子性，主要用于事务回滚和MVCC
- Redo Log是物理日志，记录了某个数据页做了什么修改，每当执行一个事务就会产生一条或者多条物理日志
- binlog（归档日志）是Server层生成的日志，主要用于数据备份和主从复制
- relay log 中继日志，用于主从复制场景下，slave通过io线程拷贝master的bin log后本地生成的日志

5、MySQL有哪些锁？作用是什么？

1. 全局锁

   全局锁主要应用于做全库逻辑备份，这样在备份数据库期间，不会因为数据或表结构的更新，而出现备份文件的数据与预期的不一样，加上全局锁，意味着整个数据库都是只读状态

2. 表级锁

   - 元数据锁（MDL）：对数据库表进行操作时，会自动给这个表加上元数据锁，为了保证当用户对表执行CURD操作时，其他线程对这个表结构做了变更，元数据锁在事务提交后才会释放
   - 意向锁：对某些记录加上共享锁之前，需要先在表级别加上一个意向共享锁，对某些纪录加上独占锁之前，需要先在表级别加上一个意向独占锁。普通的 select 是不会加行级锁的，普通的 select 语句是利用 MVCC 实现一致性读，是无锁的。
   - AUTO-INC 锁: 表里的主键通常都会设置成自增的，之后可以在插入数据时，可以不指定主键的值，数据库会自动给主键赋值递增的值通过 AUTO-INC 锁实现的。在插入数据时，会加一个表级别的 AUTO-INC 锁，然后为被 AUTO_INCREMENT 修饰的字段赋值递增的值，等插入语句执行完成后，才会把 AUTO-INC 锁释放掉。其他事务的如果要向该表插入语句都会被阻塞，从而保证插入数据时字段的值是连续递增的。

3. 行锁

   - 记录锁: 锁住的是一条记录，记录锁分为排他锁和共享锁。

- 间隙锁: 只存在于可重复读隔离级别，目的是为了解决可重复读隔离级别下幻读的现象,间隙锁之间是兼容的，两个事务可以同时持有包含共同间隙范围的间隙锁，并不存在互斥关系。
- Next-Key Lock：Next-Key Lock临键锁，是Record Lock + Gap Lock的组合，锁定一个范围，并且锁定记录本身。next-key lock既能保护该记录，又能阻止其他事务将新纪录插入到被保护记录前面的间隙中。
- 插入意向锁:一个事务在插入一条记录的时候，需要判断插入位置是否已被其他事务加了间隙锁 (next-key lock 也包含间隙锁) 。如果有的话，插入操作就会发生阻塞，直到拥有间隙锁的那个事务提交为止，在此期间会生成一个插入意向锁，表明有事务想在某个区间插入新记录，但是现在处于等待状态。