大家好，上一小节中我们对MySQL中的索引和存储引擎做了简单的介绍。本小节中，我们接着学习MySQL相关知识点，包括日志模块，锁机制以及事务等重要知识点，希望大家可以有效理解与掌握。

10

（1）MySQL的日志模块binlog和redo log有了解吗?

答：在MySQL的使用中，更新操作也是很频繁的，如果每一次更新操作都根据条件找到对应的记录，然后将记录更新，再写回磁盘，那么**IO成本以及查找记录的成本**都很高。所以，出现了日志模块，即我们的update更新操作是**先写日志，在合适的时间才会去写磁盘**，日志更新完毕就将执行结果返回给了客户端。

MySQL中的日志模块主要有**redo log（重做日志）和binlog（归档日志）。**

redo log：

redo log是**InnoDB引擎特有**的日志模块，redo log是**物理日志**，记录了某个数据页上做了哪些修改。InnoDB的redo log是固定大小的，比如可以配置为一组4个文件，每个文件的大小是1GB，那么redo log总共就可以记录 4GB的操作。从头开始写，写到末尾就又回到开头循环写。

InnoDB的redo log保证了数据库发生异常重启之后，之前提交的记录不会丢失，这个能力称为crash-safe。

binlog：

binlog是**Server层自带**的日志模块，binlog是逻辑日志，记录本次修改的原始逻辑，说白了就是SQL语句。binlog是追加写的形式，可以写多个文件，不会覆盖之前的日志。通过mysqlbinlog可以解析查看binlog日志。binlog日志文件的格式：statement，row，mixed。

- **statement格式**的binlog记录的是完整的SQL语句，优点是日志文件小，性能较好，缺点也很明显，那就是准确性差，遇到SQL语句中有now()等函数会导致不准确
- **row格式**的binlog中记录的是数据行的实际数据的变更，优点就是数据记录准确，缺点就是日志文件较大。
- **mixed格式**的binlog是前面两者的混合模式

业界目前推荐使用的是 row 模式，因为很多情况下对准确性的要求是排在第一位的。

解析：

MySQL的日志模块属于有点深度的知识点，在校招面试中考察不深，我们仅仅对这部分知识有个初步的认识和了解，就足以在面试中留下好印象并且获得加分。

在更新数据库的时候，通过redo log和binlog的**两阶段提交**，可以确保数据库异常崩溃之后数据的正确恢复。

在对数据库误操作之后，可以通过**备份库+binlog**可以将数据库状态恢复到“任意“时刻。

接下来，我们看一个关于日志模块的问题分析：

面试官：为什么MySQL会突然变慢一下?

当内存数据页和磁盘数据页内容不一致的时候，这个内存页就是“脏页”。内存数据写入到磁盘后，内存和磁盘上的数据页的内容就一致了，这个时候的内存页就是”干净页“。

前面我们介绍了更新数据库的时候是**先写日志**，当合适的机会（空闲）出现的时候才会**更新磁盘**。但是**当redo log 写满了，要 flush 脏页**，也就是把内存里的数据写入磁盘，会导致MySQL执行速度突然变慢一瞬间。

10

（2）MySQL事务有哪些特性？（重点掌握）

答：事务是单个逻辑工作单元执行的**一系列操作**，是一个不可分割的工作单位。满足如下的四大特性：

- **原子性（Atomicity）：**事务作为一个整体被执行，要么全部执行，要么全部不执行；
- **一致性（Consistency）：**保证数据库状态从一个一致状态转变为另一个一致状态；
- **隔离性（Isolation）：**多个事务并发执行时，一个事务的执行不应影响其他事务的执行；
- **持久性（Durability）：**一个事务一旦提交，对数据库的修改应该永久保存

解析：

MySQL事务又是一个几乎必考的MySQL知识点，可以拓展的知识点较多，我们一起来接着看如下的考察点。

面试官：“如果不做控制，多个事务并发操作数据库会产生哪些问题吗？“

- **丢失更新：**

两个不同事务同时获得相同数据，然后在各自事务中同时修改了该数据，那么先提交的事务更新会被后提交事务的更新给覆盖掉，这种情况先提交的事务所做的更新就被覆盖，导致数据更新丢失。

- **脏读：**

事务A读取了事务B未提交的数据，由于事务B回滚，导致了事务A的数据不一致，结果事务A出现了脏读；

- **不可重复读：**

一个事务在自己没有更新数据库数据的情况，同一个查询操作执行两次或多次得到的结果数值不同，因为别的事务更新了该数据，并且提交了事务。

- **幻读：**

事务A读的时候读出了N条记录，事务B在事务A执行的过程中增加 了1条，事务A再读的时候就变成了N+1条，这种情况就叫做幻读。

注意：幻读是指一种结构上的改变，比如说条数发生了改变；不可重复读是指读出的数值发生了改变。

面试官：“MySQL数据库事务的隔离级别有哪些？“

为了避免数据库事务操作中的问题，MySQL定义了4个事务隔离级别，不同的隔离级别对事务的处理不同。

- **读未提交(Read Uncommitted)：**

允许脏读取。如果一个事务已经开始写数据，则另外一个数据则不允许同时进行写操作，但允许其他事务读此行数据。

- **读已提交(Read Committed)：**

允许不可重复读取，但不允许脏读取。读取数据的事务允许其他事务继续访问该行数据，但是未提交的写事务将会禁止其他事务访问该行。

- **可重复读(Repeatable Read)：**

禁止不可重复读取和脏读取，但是有时可能出现幻读。读取数据的事务将会禁止写事务(但允许读事务)，写事务则禁止任何其他事务。

- **序列化(Serializable)：**

提供严格的事务隔离。它要求事务序列化执行，事务只能一个接着一个地执行，但不能并发执行。

事务的隔离级别越高，对数据的完整性和一致性保证越佳，但是对并发操作的影响也越大。MySQL事务默认隔离级别是可重复读。

题目总结：

关于事务和隔离级别，是MySQL中的基础知识点，需要大家熟练掌握。在本小节中，我们仅仅是做了一个抛砖引玉的简要阐述，大家如果有基础稍弱感觉很晦涩的点，可以在文章下边评论，我会再次和大家一起交流学习。

（3）MySQL中的锁机制？（重点掌握）

答：MySQL数据库的锁分为**表级锁和行级锁**。从数据库的角度看，行级锁又可以分为**独占锁和共享锁**。

独占锁（排他锁），也称X锁（Exclusive Lock）：

独占锁锁定的资源只允许进行锁定操作的程序使用，其它任何对它的操作均不会被接受。执行数据更新命令，**即INSERT、UPDATE 或DELETE 命令时，MySQL会自动使用独占锁**。但当对象上有其它锁存在时，无法对其加独占锁。独占锁一直到事务结束才能被释放。

在select命令中使用独占锁的SQL语句为：**select … for update;**

共享锁，也叫S锁（Shared Lock）：

共享锁顾名思义，那就是其锁定的资源可以被其它用户读取，但其它用户不能修改。如果在select查询语句中要手动加入共享锁，那么对应的SQL语句为：**select ... lock in share mode**

这里需要注意：

一个事务在一行数据上加入了独占锁，那么其余事务不可以在该数据行上加入任何锁。也就是说加过排他锁的数据行在其他事务种是不能修改数据的，也不能通过**for update和lock in share mode锁**的方式查询数据，但**可以直接通过select ...from...查询数据**，因为普通查询没有任何锁机制。

MySQL中的死锁：

在多线程章节（第8节：Java进阶 - 高效并发编程（上））中，我们介绍了多线程为了争夺资源可能会造成死锁，也就是一种环路等待的现象。MySQL中的死锁主要是**多个事务使用行级锁对某行数据加锁造成的**，上一小节说了MyISAM不支持行级锁，所以MySQL中的死锁主要是在说InnoDB存储引擎的死锁。

那么MySQL的死锁该如何解决呢?

我们可以在业务上和数据库设置上来解决MySQL死锁。分别介绍如下：

业务逻辑上的死锁解决方案：

- 指定锁的获取顺序
- 大事务拆分成各个小事务
- 在同一个事务中，一次锁定尽量多的资源，减少死锁概率
- 给表建立合适的索引以及降低事务的隔离级别等

10

数据库的设置来解决死锁：

- 通过参数 innodb_lock_wait_timeout 根据实际业务场景来**设置超时时间**，InnoDB引擎默认值是50s。
- **发起死锁检测**，发现死锁后，**主动回滚**死锁链条中的某一个事务，让其他事务得以继续执行。将参数 innodb_deadlock_detect 设置为 on，表示开启这个逻辑（默认是开启状态）。

我们在来看一个关于行级锁的问题吧~

面试官：“行级锁什么时候会锁住整个表？“

InnoDB行级锁是通过锁索引记录实现的，如果更新的列没建索引是会锁住整个表的。

悲观锁与乐观锁：

从程序员的角度看，数据库中的锁又可以分为**悲观锁和乐观锁。**

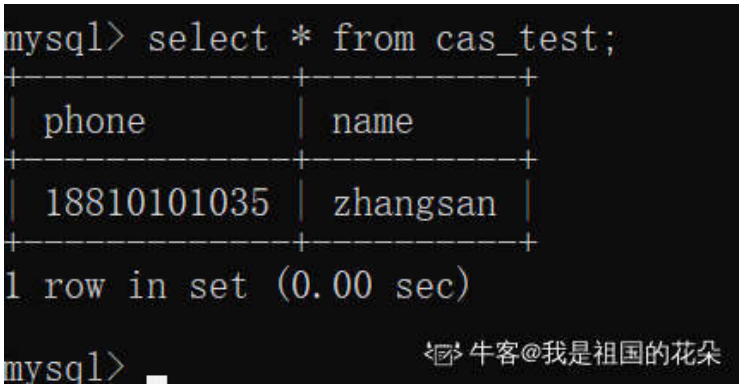
悲观锁：利用数据库的锁机制实现，在整个数据处理过程中都加入了锁，以保持排他性。

乐观锁：乐观锁可以利用CAS实现，在**操作数据**的时候进行一个比较，按照**当前事务中的数据和数据库表中的该数据是否一致**来决定是否要执行本次操作。

我们来举例说下乐观锁的CAS实现，有如下的数据库表和数据：

```
create table cas_test(
phone varchar(32) primary key,
name varchar(32)
)engine=InnoDB;

// 插入数据
insert into cas_test values("18810101035","zhangsan");
```



当我们将数据更新后，会比较该行数据与数据库表中的该行数据是否一致，以此来决定是否要更新数据。

面试官：“乐观锁的ABA问题有了解吗？如何解决？“

ABA问题是指在当前事务读取该行数据时是A，经过别的事务修改成B，但是在当前事务要更新数据的时候，该行数据又被别的事务修改为A，事实上数据行是发生过改变的，存在并发问题。

ABA问题可以通过**基于数据版本（Version）记录机制**来解决。也就是为数据增加一个版本标识。读取出数据时，将此版本号一同读出，之后更新时，对此版本号加一。根据当前事务的数据版本号和数据库中数据的版本号对比来决定是否更新数据。

与给当前数据增加一个数据版本类似，我们也可以增加基于时间戳机制来解决ABA问题，通过时间戳来记录当前数据行变化。

基于数据版本Version机制，表结构设计如下所示：

```
mysql> create table cas_test_version(
    -> phone varchar(32) primary key,
    -> name varchar(32),
    -> version varchar(32)
    -> )engine=InnoDB;
Query OK, 0 rows affected (0.01 sec)

mysql> desc cas_test_version;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| phone   | varchar(32) | NO   | PRI | NULL    |       |
| name    | varchar(32) | YES  |     | NULL    |       |
| version | varchar(32) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```


基于时间戳机制，表结构设计如下所示：

```
mysql> create table cas_test_timestamp(
    -> phone varchar(32) primary key,
    -> name varchar(32),
    -> timestamp varchar(128)
    -> )engine=InnoDB;
Query OK, 0 rows affected (0.01 sec)

mysql> desc cas_test_timestamp;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| phone     | varchar(32)  | NO   | PRI | NULL    |       |
| name      | varchar(32)  | YES  |     | NULL    |       |
| timestamp | varchar(128) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.02 sec)

mysql>
```


（4）有SQL优化或者MySQL故障排查经历吗?

答：一般情况下，我们遇到一个SQL异常的时候，比如说执行时间超时等，可以通过**explain**查看当前SQL语句的执行情况。explain +SQL语句可以查看当前的SQL语句使用的索引以及其扫描了多少行数据。也可以使用下边的语句来查看数据表的一些信息：

- **show create table TableXX；**查看当前表TableXX的建表语句
- **show index from TableXX；**查看当前表TableXX上的索引

查看了数据表的信息，一般情况下我们可以通过建立索引来提高查询速度，或者修改SQL语句，利用索引下推或者最左前缀原则等来加快查询速度。

解析：

这个题目的考察，主要是针对有实习经验或者有正式工作经验的同学。对于其余同学，我们可以对explain命令的输出有一个基本的了解，相信在面试中也会占得先机，拔得头筹。有能力和信心的同学可以自行学习explain命令的输出内容等知识点。

（5）MySQL建表的约束条件有哪些?

答：约束条件是我们建表的时候对数据库表做的一个限制条件。MySQL建表时候一般有如下的五个约束条件：

- **主键约束**（Primay Key Coustraint） 唯一性，非空性
- **唯一约束** （Unique Counstraint）唯一性，可以空，但只能有一个
- **检查约束** (Check Counstraint) 对该列数据的范围、格式的限制
- **默认约束** (Default Counstraint) 该数据的默认值
- **外键约束** (Foreign Key Counstraint) 需要建立两表间的关系并引用主表的列

（6）MySQL其余知识点?

限于文章篇幅，我们一起MySQL中的重要的大块知识点进行了一个交流与学习。还有一个比较细小或者进阶的重要知识点。这里我们做一个简单的罗列吧。**常见的MySQL知识点如下：**

- limit分页查询使用方式
- 主键自增长设置方式：auto_increment
- SQL语句的优化有哪些?
- order by是怎么工作的?
- MVCC多版本并发控制以及undo log（回滚日志）。（加分项）
- 分布式事务，两阶段，三阶段提交协议等。（加分项）

总结：

上一小节和本小节中，我们对面试中常见的MySQL考察点进行了较为全面的介绍。服务端开发岗位对于数据库的考察是一个必考点，希望大家可以有效理解与掌握。当然，除了文章中的内容，其余MySQL知识点一样可以在评论区交流哦~

限于作者水平，文章中难免会有不妥之处。大家在学习过程中遇到我没有表达清楚或者表述有误的地方，欢迎随时在文章下边指出，我会及时关注，随时改正。另外，大家有任何话题都可以在下边留言，我们一起交流探讨。

讨论 评论

柳杰201905011049420 1#

Mysql事务隔离级别不懂

发表于 2020-01-09 11:12:53 赞(0) 回复(1)

10

Hobo丶 ： 哥，这是数据库的基础。。。

2020-04-19 05:02:22 赞(2) 回复(0)

请输入你的观点

回复

刘畅201904211606816 2#

打卡

发表于 2020-01-10 21:20:47 赞(1) 回复(0)

柳杰201905011049420 3#

?

发表于 2020-01-13 19:09:28 赞(0) 回复(0)

牧水s 4#

打卡

发表于 2020-01-20 17:05:28 赞(0) 回复(0)

四月hope 5#

MVCC多版本并发控制以及undo log（回滚日志）

希望更新下这个，网上太坑人了

发表于 2020-04-14 16:56:00 赞(0) 回复(4)

我是祖国的花朵 作者 ： 你好。1. 回滚日志（undo log）：在 MySQL 中，实际上每条记录在更新的时候都会同时记录一条回滚操作。记录上的最新值，通过回滚操作，都可以得到前一个状态的值。

2020-05-16 21:40:54 赞(0) 回复(0)

我是祖国的花朵 作者 ： MVCC的实现方式：1. InnoDB实现MVCC，多个版本的数据可以共存，主要是依靠数据的隐藏列(也可以称之为标记位)和undo log。 2. 其中数据的隐藏列包括了该行数据的版本号、删除时间、指向undo log的指针等等； 3. 当读取数据时，MySQL可以通过隐藏列判断是否需要回滚并找到回滚需要的undo log，从而实现MVCC；

2020-05-16 21:41:25 赞(0) 回复(0)

我是祖国的花朵 作者： 1. 事务是如何实现的MVCC呢？ 1. 每个事务都有一个事务ID，叫做transaction id(严格递增) 2. 事务在启动时，找到已提交的最大事务ID记为up_limit_id。 3. 事务在更新一条语句时，比如id=1改为了id=2。会把id=1和该行之前的row trx_id写到undo log里，并且在数据页上把id的值改为2,并且把修改这条语句的transaction id记在该行行头。

2020-05-16 21:41:55 赞(0) 回复(0)

我是祖国的花朵 作者： 可以这么理解，希望对你有帮助～

2020-05-16 21:42:16 赞(0) 回复(0)

请输入你的观点

回复

DGQ_00 6#

InnoDB行级锁是通过锁索引记录实现的，如果更新的列没建索引是会锁住整个表的。

这句话应该这样理解：只有通过索引条件检索数据，InnoDB才使用行级锁，否则，InnoDB将使用表锁；也就是说后面的where条件中没有根据索引来检索数据的就会升级为表锁！！！（实验验证过）

发表于 2020-04-14 18:58:12 赞(0) 回复(0)

aud 7#

MySQL建表的约束条件那里可以回答三范式的内容吗

发表于 2020-05-03 09:28:08 赞(0) 回复(0)

Shawn_Liu 8#

05/11 Mark

发表于 2020-05-12 06:50:23 赞(0) 回复(0)

赚多多 9#

你好，请问next key lock怎么理解

发表于 2020-05-16 20:45:38 赞(0) 回复(1)

我是祖国的花朵 作者： 你好，间隙锁的锁定规则是通过在指向数据记录的第一个索引键之前和最后一个索引键之后的空域空间上标记锁定信息而实现的。MySQL通过可重复读的隔离级别+间隙锁

解决了幻读的问题。

2020-05-16 21:38:53 赞(0) 回复(0)

请输入你的观点

10

回复

凌客 10#

什么情况下会用到表级锁

发表于 2020-05-20 19:36:53 赞(0) 回复(0)