马士兵教育
www.mashibing.com

这几道mysql题你搞懂了，金三银四涨薪稳了

授课：连鹏举

面试题：

- 1、mysql原子性和持久性怎么保证
- 2、innodb和myisam区别
- 3、索引分类
- 4、innodb的底层数据结构
- 5、为什么底层使用B+树不用B树

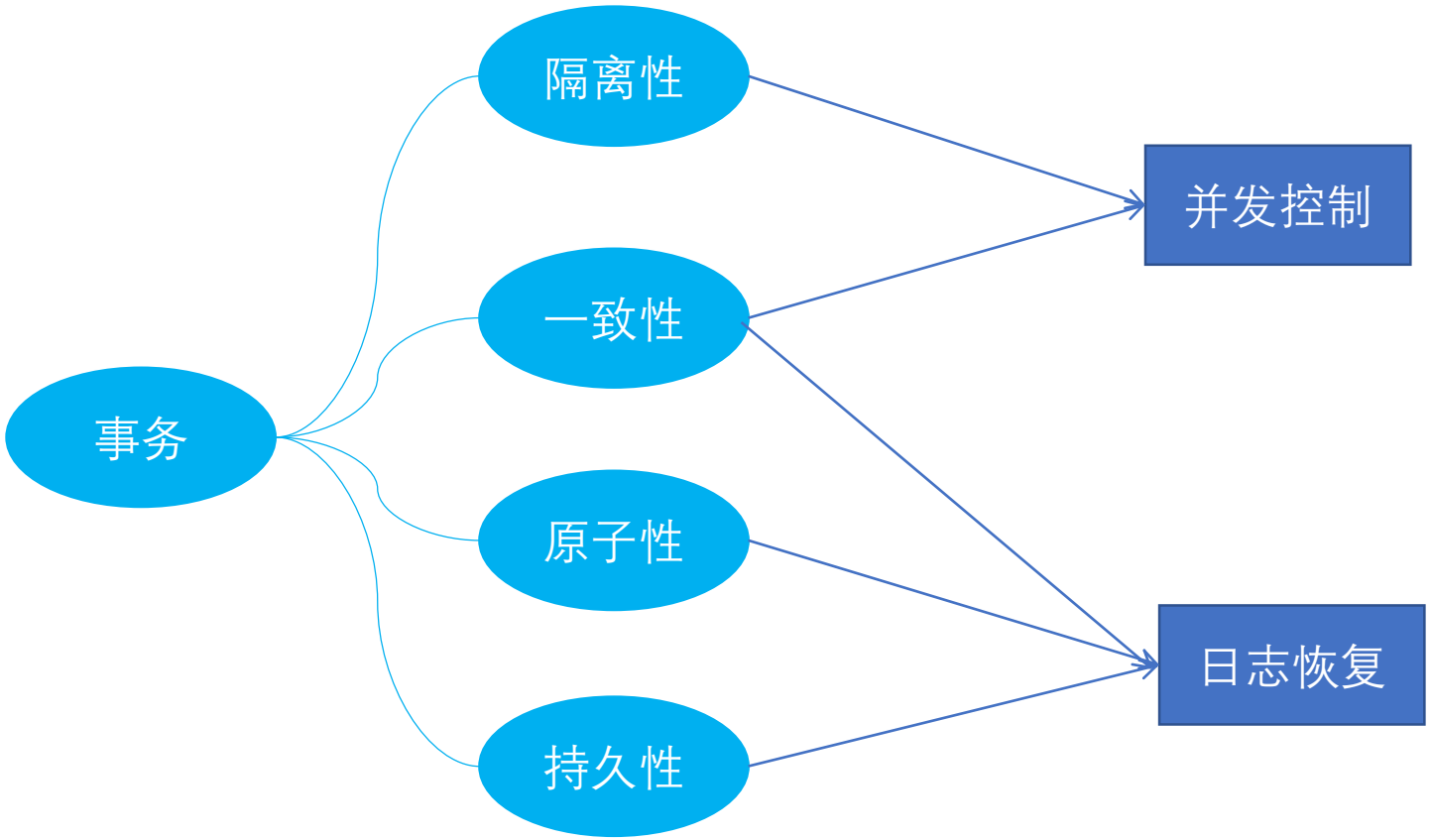
事务的特点（ACID）

- 原子性(Atomicity):事务中的所有操作作为一个整体像原子一样不可分割，要么全部成功,要么全部失败。
- 一致性(Consistency):事务的执行结果必须使数据库从一个一致性状态到另一个一致性状态。一致性状态是指:1.系统的状态满足数据的完整性约束(主码,参照完整性,check约束等) 2.系统的状态反应数据库本应描述的现实世界的真实状态,比如转账前后两个账户的金额总和应该保持不变。
- 隔离性(Isolation):并发执行的事务不会相互影响,其对数据库的影响和它们串行执行时一样。比如多个用户同时往一个账户转账,最后账户的结果应该和他们按先后次序转账的结果一样。
- 持久性(Durability):事务一旦提交,其对数据库的更新就是持久的。任何事务或系统故障都不会导致数据丢失。

事务的特点

- 在事务的四个特点中，一致性是事务的根本追求，而在某些情况下会对事务的一致性造成破坏:
  - 事务的并发执行
  - 事务故障或系统故障
- 数据库系统通过并发控制技术和日志恢复技术来避免这种情况的发生
  - 并发控制技术保证了事务的隔离性,使数据库的一致性状态不会因为并发执行的操作被破坏。
  - 日志恢复技术保证了事务的原子性,使一致性状态不会因事务或系统故障被破坏。同时使已提交的对数据库的修改不会因系统崩溃而丢失,保证了事务的持久性。

事务的特点



事务的实现原理

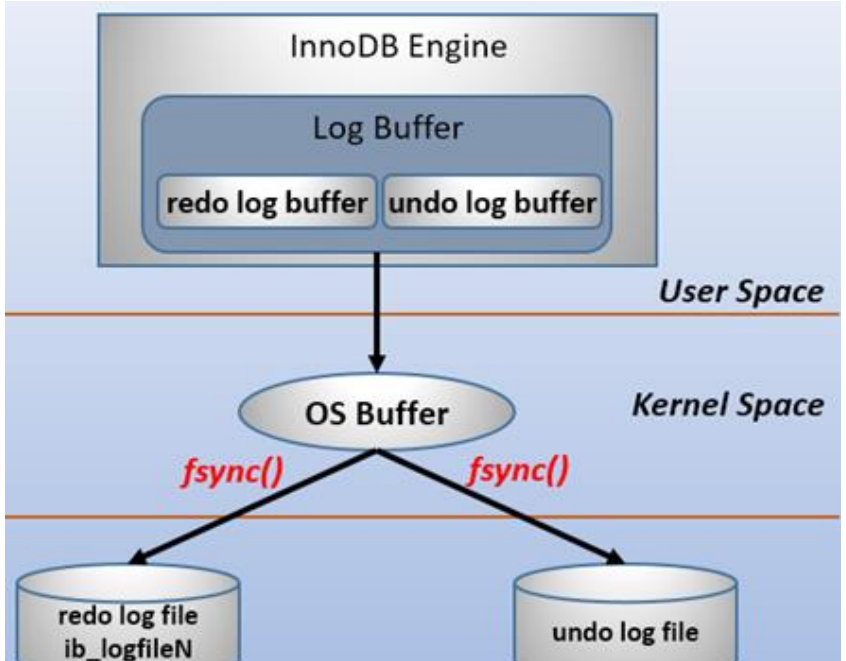
- 事务的原子性是通过 undo log 来实现的
- 事务的持久性是通过 redo log 来实现的
- 事务的隔离性是通过 (读写锁+MVCC)来实现的
- 事务的一致性是通过原子性，持久性，隔离性来实现的！！！

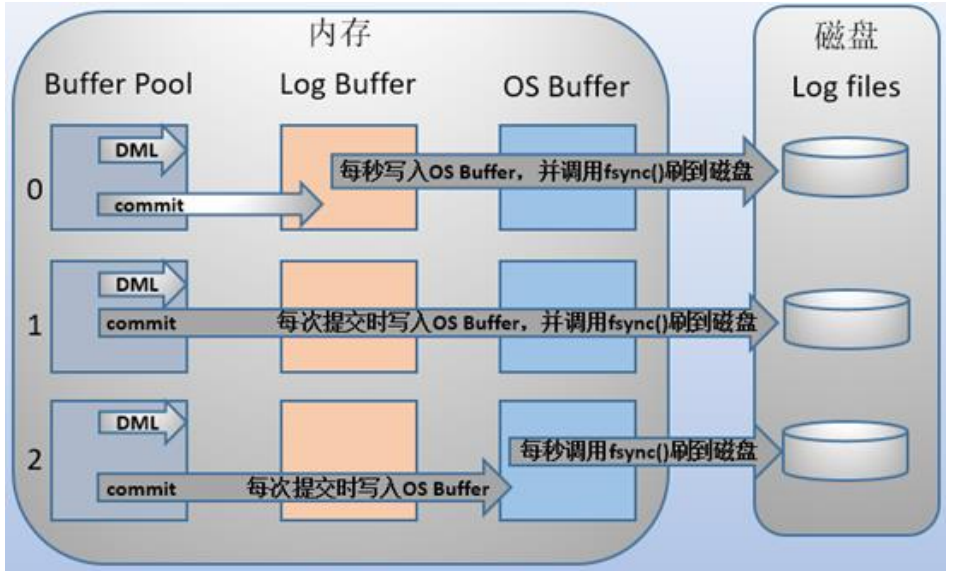
原子性实现原理：Undo log

- Undo Log是为了实现事务的原子性，在MySQL数据库InnoDB存储引擎中，还用Undo Log来实现多版本并发控制(简称：MVCC)
- 在操作任何数据之前，首先将数据备份到一个地方（这个存储数据备份的地方称为Undo Log）。然后进行数据的修改。如果出现了错误或者用户执行了ROLLBACK语句，系统可以利用Undo Log中的备份将数据恢复到事务开始之前的状态
- 注意：undo log是逻辑日志，可以理解为：
  - 当delete一条记录时，undo log中会记录一条对应的insert记录
  - 当insert一条记录时，undo log中会记录一条对应的delete记录
  - 当update一条记录时，它记录一条对应相反的update记录

持久性实现原理：Redo log

- 和Undo Log相反，Redo Log记录的是新数据的备份。在事务提交前，只要将Redo Log持久化即可，不需要将数据持久化。当系统崩溃时，虽然数据没有持久化，但是Redo Log已经持久化。系统可以根据Redo Log的内容，将所有数据恢复到最新的状态





Mysql的隔离级别

- 事务具有隔离性,理论上来说事务之间的执行不应该相互产生影响,其对数据库的影响应该和它们串行执行时一样。
- 然而完全的隔离性会导致系统并发性能很低,降低对资源的利用率,因而实际上对隔离性的要求会有所放宽,这也会一定程度造成对数据库一致性要求降低
- SQL标准为事务定义了不同的隔离级别,从低到高依次是
  - 读未提交(READ UNCOMMITTED)：对事务处理的读取没有任何限制，不推荐
  - 读已提交(READ COMMITTED):
  - 可重复读(REPEATABLE READ)
  - 串行化(SERIALIZABLE)

Mysql的隔离级别

- 不同的隔离级别可能导致不同的并发异常，如下图：

事务的隔离级别	脏读	不可重复读	幻读
读未提交(READ UNCOMMITTED)	√	√	√
读已提交(READ COMMITTED)		√	√
可重复读(REPEATABLE READ)			√
串行化(SERIALIZABLE)			

隔离性实现原理：锁

- 在mysql中，锁可以分为两类：
  - 共享锁：共享锁定是将对象数据变为**只读**形式，不能进行更新，所以也成为读取锁定；
  - 排他锁：排他锁定是当执行INSERT/UPDATE/DELETE的时候，其它事务**不能读取**该数据，因此也成为写入锁定。
- 锁的粒度：锁定对象的大小是锁的粒度
  - 记录
  - 表

隔离性实现原理：锁

- 基于锁的并发控制流程
  - 事务根据自己对数据项进行的操作类型申请相应的锁(读申请共享锁,写申请排他锁)
  - 申请锁的请求被发送给锁管理器。锁管理器根据当前数据项是否已经有锁以及申请的和持有的锁是否冲突决定是否为该请求授予锁。
  - 若锁被授予,则申请锁的事务可以继续执行;若被拒绝,则申请锁的事务将进行等待,直到锁被其他事务释放。
- 可能出现的问题:
  - 死锁:多个事务持有锁并互相循环等待其他事务的锁导致所有事务都无法继续执行
- 扩展：除了锁可以实现并发控制之外，还有其他策略:
  - 基于时间戳的并发控制
  - 基于有效性检查的并发控制
  - 基于快照隔离的并发控制

故障及故障恢复

- 事务的执行流程如下:
  - 系统会为每个事务开辟一个私有工作区
  - 事务读操作将从磁盘中拷贝数据项到工作区中,在执行写操作前所有的更新都作用于工作区中的拷贝.
  - 事务的写操作将把数据输出到内存的缓冲区中,等到合适的时间再由缓冲区管理器将数据写入到磁盘
- 由于数据库存在立即修改和延迟修改,所以在事务执行过程中可能存在以下情况:
  - 在事务提交前出现故障,但是事务对数据库的部分修改已经写入磁盘数据库中。这导致了事务的原子性被破坏。
  - 在系统崩溃前事务已经提交,但数据还在内存缓冲区中,没有写入磁盘。系统恢复时将丢失此次已提交的修改。这是对事务持久性的破坏。

故障及故障恢复

- 撤销事务undo:将事务更新的所有数据项恢复为日志中的旧值
- 重做事务redo:将事务更新的所有数据项恢复为日志中的新值。
- 事务正常回滚/因事务故障中止将进行redo
- 系统从崩溃中恢复时将先进行redo再进行undo。

基础知识储备

- 局部性原理

现在，进而讨论在作业信息不全部装入主存的情况下能否保证作业的正确运行？回答是肯定的，早在 1968 年 P.Denning 就研究了程序执行时的**局部性（principle of locality）**原理，对程序局部性原理进行研究还有 Knuth（分析一组学生的 Fortran 程序）、Tanenbaum（分析操作系统的过程）、Huck（分析通用科学计算的程序），发现程序和数据的访问都有聚集成群的倾向，在一个时间段内，仅使用其中一小部分（称**空间局部性**），或者最近访问过的程序代码和数据，很快又被访问的可能性很大（称**时间局部性**）。这只要对程序的执行进行分析就可以发现以下一些情况：

- 磁盘预读(预读的长度一般为页（page）的整数倍)
  - 页是存储器的逻辑块，操作系统往往将主存和磁盘存储区分割为连续的大小相等的块，每个存储块称为一页（在许多操作系统中，页大小通常为4k），主存和磁盘以页为单位交换数据。

索引是什么？

- 索引是帮助 MySQL 高效获取数据的数据结构
- 索引存储在文件系统中
- 索引的文件存储形式与存储引擎有关
- 索引文件的结构
  - hash
  - 二叉树
  - B树
  - B+树

索引的分类

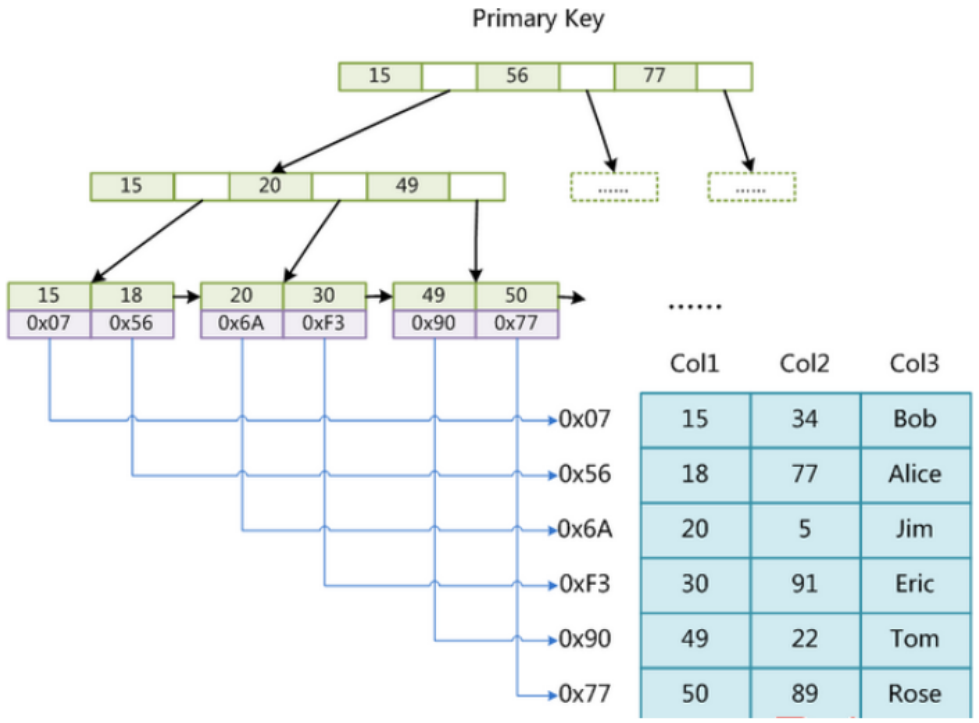
- mysql索引的五种类型：主键索引、唯一索引、普通索引和全文索引、组合索引。通过给字段添加索引可以提高数据的读取速度，提高项目的并发能力和抗压能力。
- 主键索引
  - 主键是一种唯一性索引，但它必须指定为PRIMARY KEY，每个表只能有一个主键。
- 唯一索引
  - 索引列的所有值都只能出现一次，即必须唯一，值可以为空。
- 普通索引
  - 基本的索引类型，值可以为空，没有唯一性的限制。
- 全文索引
  - 全文索引的索引类型为FULLTEXT。全文索引可以在varchar、char、text类型的列上创建
- 组合索引
  - 多列值组成一个索引，专门用于组合搜索

mysql的存储引擎

	MyISAM	InnoDB
索引类型	非聚簇索引	聚簇索引
支持事务	否	是
支持表锁	是	是
支持行锁	否	是
支持外键	否	是
支持全文索引	是	是（5.6后支持）
适合操作类型	大量select	大量insert、delete、update

mysql索引机制

- mysql-MyISAM



mysql索引机制

- mysql-InnoDB