MySQL

MYSQL索引

MYSQL索引的数据结构，B+Tree，哈希

MYSQL引擎，MyISAM，InnoDB

https://www.cnblogs.com/jiawen010/p/11805241.html

InnoDB中，表数据文件本身就是按B+Tree组织的一个索引结构，聚簇索引就是按照每张表的主键构造一颗B+树，同时叶子节点中存放的就是整张表的行记录数据，也将聚集索引的叶子节点称为数据页。这个特性决定了索引组织表中数据也是索引的一部分；

我们日常工作中，根据实际情况自行添加的索引都是辅助索引，辅助索引就是一个为了需找主键索引的二级索引，现在找到主键索引再通过主键索引找数据；

MyISAM与InnoDB的对比：

1. **是否支持行级锁** : MyISAM 只有表级锁(table-level locking)，而InnoDB 支持行级锁(row-level locking)和表级锁,默认为行级锁。
2. InnoDB使用的是**聚簇索引**，MyISM使用的是**非聚簇索引**
3. **是否支持事务和崩溃后的安全恢复：** **MyISAM** 强调的是性能，每次查询具有原子性,其执行速度比InnoDB类型更快，但是不提供事务支持。但是**InnoDB** 提供事务支持事务，外部键等高级数据库功能。 具有事务(commit)、回滚(rollback)和崩溃修复能力(crash recovery capabilities)的事务安全(transaction-safe (ACID compliant))型表。
4. **是否支持外键：** MyISAM不支持，而InnoDB支持。
5. **是否支持MVCC** ： 仅 InnoDB 支持。应对高并发事务, MVCC比单纯的加锁更高效;MVCC只在 `READ COMMITTED` 和 `REPEATABLE READ` 两个隔离级别下工作;MVCC可以使用 乐观(optimistic)锁和 悲观(pessimistic)锁来实现;各数据库中MVCC实现并不统一。

MySQL，binlog日志有哪几种，如何利用binlog同步es或者从库

MySQL binlog日志有三种格式，分别为Statement,MiXED,以及ROW

- **Statement**：每一条会修改数据的sql都会记录在binlog中
- **Row:**不记录sql语句上下文相关信息，仅保存哪条记录被修改
- **Mixedlevel:** 是以上两种level的混合使用，一般的语句修改使用statment格式保存binlog，statement无法完成主从复制的操作，则采用row格式保存binlog

数据库崩溃时事务的恢复机制（REDO日志和UNDO日志）

原子性和高性能持久化

Undo Log记录事务前数据，Redo Log刚好相反，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

索引什么场景失效

- 未使用索引列作为查询条件
- 类型转换，列类型是字符串，查询条件未加引号
- 不符合前置匹配原则，like操作通配符在前；联合索引ABC采用BC检索
- 在查询条件中使用OR，除非每列都加索引，才会生效
- 对索引列进行函数运算

MySQL隔离级别，RR可以解决幻读问题吗? 为什么

> mysql 的幻读并非什么读取两次返回结果集不同，而是事务在插入事先检测不存在的记录时，惊奇的发现这些数据已经存在了，之前的检测读获取到的数据如同鬼影一般
>
> 不可重复读侧重表达 读-读，幻读则是说 读-写，用写来证实读的是鬼影

mysql在rr隔离级别下一定程度上解决了幻读问题。 由于innodb引擎下存在当前读和快照读的概念。

在当前读的情况下，mysql通过配置可以采用记录锁(Record Lock)+间隙锁(Gap Lock)，让其他插入或删除事务死锁，达到解决幻读问题。select ... lock in share mode, select ... for update

在快照读（普通读）的情况下，mysql如果不更新插入记录，那么由于是读取的旧版本记录，对于其他事务插入数据不可见，从而达到了解决幻读问题。但是如果当前事务更新记录(包括不可见的)，会去读取最新版本内容，从而看到了其他事务插入的数据，产生幻读