**MySQL**

摘要:

**关键词：**

# 锁和事务

MySQL的Innodb下的锁和事务。

<https://www.cnblogs.com/wangkaihua/p/10217490.html>

## 锁

读锁：也叫共享锁、S锁，若事务T对数据对象A加上S锁，则事务T可以读A但不能修改A，其他事务只能再对A加S锁，而不能加X锁，直到T释放A上的S 锁。这保证了其他事务可以读A，但在T释放A上的S锁之前不能对A做任何修改。

写锁：又称排他锁、X锁。若事务T对数据对象A加上X锁，事务T可以读A也可以修改A，其他事务不能再对A加任何锁，直到T释放A上的锁。这保证了其他事务在T释放A上的锁之前不能再读取和修改A。

表锁：操作对象是数据表。Mysql大多数锁策略都支持(常见mysql innodb)，是系统开销最低但并发性最低的一个锁策略。事务t对整个表加读锁，则其他事务可读不可写，若加写锁，则其他事务增删改都不行。

行级锁：操作对象是数据表中的一行。是MVCC技术用的比较多的，但在MYISAM用不了，行级锁用mysql的储存引擎实现而不是mysql服务器。但行级锁对系统开销较大，处理高并发较好。

## 本地事务

事务是靠锁策略来实现的。

使用MVCC <https://www.cnblogs.com/chenpingzhao/p/5065316.html>

一致无锁读：select

锁读：select \* from table for update;select \* from table for share\

加锁操作：update，insert，delete

## 分布式事务

本来想弄明白服务层面的事务，不想在网上查资料，查着查着查到数据库层面的分布式事务。不过想来也是这样，事务的一致性最终体现在持久层。

Mysql的XA事务分为外部XA和内部XA

内部的XA:是MySQL自己的机制，怎么使用？

外部的XA: 开源的代理工具：网易的DDB，淘宝的TDDL（Diamond）等等。

<https://www.2cto.com/database/201806/752199.html>

似乎只是对数据库改造成分库分表的多机模式，也就是分布式数据库的解决方案。

# 索引

Mysql中的索引，以Innodb为例，底层使用的是B+树来实现的。

重点讲多字段的符合索引：

参考：<https://www.cnblogs.com/yyjie/p/7486975.html>

1. 全值匹配
2. 匹配最左列
3. 匹配列前缀 like ‘zhou%’
4. 匹配范围
5. 对于整数类型 全部走索引

## 测试案例

使用MySQL world 数据

-- mysql的InnoDB和MyISAM引擎都不支持Hash索引

-- name 常规索引 btree,CountryCode 常规索引 Btree .查询只使用了name索引

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode='NLD' and t.name = 'Groningen' ;

-- name 使用了Like 左侧使用模糊查询不走索引

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode='NLD' and t.name like '%oningen' ;

-- name 使用了Like 右侧使用模糊查询走索引

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode='NLD' and t.name like 'Groning%' ;

-- 测试联合索引

-- Name, CountryCode 联合索引

-- name 走了联合索引

EXPLAIN SELECT \* FROM `city` t WHERE t.name = 'Groningen' ;

-- CountryCode 没有走索引

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode='NLD' ;

-- CountryCode，Name 联合索引

-- name 没有走索引

EXPLAIN SELECT \* FROM `city` t WHERE t.name = 'Groningen' ;

-- CountryCode 走了联合索引

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode='NLD' ;

-- where 后面跟了联合索引的所有字段，和顺序无关，都走索引

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode='NLD' and t.name = 'Groningen' ;

EXPLAIN SELECT \* FROM `city` t WHERE t.name = 'Groningen' and t.CountryCode='NLD';

-- 走索引

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode='NLD' and t.name like '%oningen' ;

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode='NLD' and t.name like 'Gronin%' ;

EXPLAIN SELECT \* FROM `city` t WHERE t.name like '%roningen' and t.CountryCode='NLD';

-- 不走索引

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode like '%LD' and t.name = 'Groningen' ;

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode LIKE'%LD' and t.name like '%oningen' ;

EXPLAIN SELECT \* FROM `city` t WHERE t.name like '%roningen' and t.CountryCode LIKE'%LD';

-- 不走索引

EXPLAIN SELECT \* FROM `city` t WHERE t.CountryCode like'%LD' and t.name like 'Groni%' ;

-- 联合索引总结：第一要保证有索引第一个字段，才可能走索引。第二