

Mysql中“select ... for update”排他锁(转)

原帖地址

<https://blog.csdn.net/claram/article/details/54023216>

Mysql InnoDB 排他锁

用法： select ... for update;


例如： select * from goods where id = 1 for update;

排他锁的申请前提：没有线程对该结果集中的任何行数据使用排他锁或共享锁，否则申请会阻塞。


for update仅适用于InnoDB，且必须在事务块(BEGIN/COMMIT)中才能生效。在进行事务操作时，通过“for update”语句，MySQL会对查询结果集中每行数据都添加排他锁，其他线程对该记录的更新与删除操作都会阻塞。排他锁包含行锁、表锁。

场景分析


假设有一张商品表 goods，它包含 id，商品名称，库存量三个字段，表结构如下：




```
1 CREATE TABLE `goods` (  
2   `id` int(11) NOT NULL AUTO_INCREMENT,  
3   `name` varchar(100) DEFAULT NULL,  
4   `stock` int(11) DEFAULT NULL,  
5   PRIMARY KEY (`id`),  
6   UNIQUE KEY `idx_name` (`name`) USING HASH  
7 ) ENGINE=InnoDB
```



插入如下数据：



```
1 INSERT INTO `goods` VALUES ('1', 'prod11', '1000');  
2 INSERT INTO `goods` VALUES ('2', 'prod12', '1000');  
3 INSERT INTO `goods` VALUES ('3', 'prod13', '1000');  
4 INSERT INTO `goods` VALUES ('4', 'prod14', '1000');  
5 INSERT INTO `goods` VALUES ('5', 'prod15', '1000');  
6 INSERT INTO `goods` VALUES ('6', 'prod16', '1000');  
7 INSERT INTO `goods` VALUES ('7', 'prod17', '1000');  
8 INSERT INTO `goods` VALUES ('8', 'prod18', '1000');  
9 INSERT INTO `goods` VALUES ('9', 'prod19', '1000');
```




一、数据一致性

假设有A、B两个用户同时各购买一件 id=1 的商品，用户A获取到的库存量为 1000，用户B获取到的库存量也为 1000，用户A完成购买后修改该商品的库存量为 999，用户B完成购买后修改该商品的库存量为 999，此时库存量数据产生了一致。


有两种解决方案：

悲观锁方案：每次获取商品时，对该商品加排他锁。也就是在用户A获取获取 id=1 的商品信息时对该行记录加锁，期间其他用户阻塞等待访问该记录。悲观锁适合写入频繁的场景。




```
1 begin;  
2 select * from goods where id = 1 for update;  
3 update goods set stock = stock - 1 where id = 1;  
4 commit;
```

乐观锁方案：每次获取商品时，不对该商品加锁。在更新数据的时候需要比较程序中的库存量与数据库中的库存量是否相等，如果相等则进行更新，反之程序重新获取库存量，再次进行比较，直到两个库存量的数值相等才进行数据更新。乐观锁适合读取频繁的场景。



```
1 #不加锁获取 id=1 的商品对象  
2 select * from goods where id = 1  
3  
4 begin;  
5 #更新 stock 值，这里需要注意 where 条件 “stock = cur_stock”，只有程序中获取到的库存量与数据库中的库存量相等才执行更新  
6 update goods set stock = stock - 1 where id = 1 and stock = cur_stock;  
7 commit;
```




如果我们需要设计一个商城系统，该选择以上的哪种方案呢？

查询商品的频率比下单支付的频次高，基于以上我可能会优先考虑第二种方案（当然还有其他的方案，这里只考虑以上两种方案）。


二、行锁与表锁

1、只根据主键进行查询，并且查询到数据，主键字段产生行锁。




```
1 begin;  
2 select * from goods where id = 1 for update;  
3 commit;
```

2、只根据主键进行查询，没有查询到数据，不产生锁。




```
1 begin;  
2 select * from goods where id = 1 for update;  
3 commit;
```

3、根据主键、非主键含索引 (name) 进行查询，并且查询到数据，主键字段产生行锁，name字段产生行锁。



```
1 begin;  
2 select * from goods where id = 1 and name='prod11' for update;  
3 commit;
```

4、根据主键、非主键含索引 (name) 进行查询，没有查询到数据，不产生锁。



```
1 begin;  
2 select * from goods where id = 1 and name='prod12' for update;  
3 commit;
```

公告

昵称： 九号云
园龄： 1年8个月
粉丝： 1
关注： 0
+加关注

	<	2019年8月						>
日	一	二	三	四	五	六		
28	29	30	31	1	2	3		
4	5	6	7	8	9	10		
11	12	13	14	15	16	17		
18	19	20	21	22	23	24		
25	26	27	28	29	30	31		
1	2	3	4	5	6	7		

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

跨域解析第三方网站json内容(1)

随笔档案

2019年8月(1)
2019年3月(6)
2019年2月(1)
2019年1月(4)
2018年12月(2)
2018年10月(1)
2018年6月(3)
2018年5月(3)
2018年3月(1)
2017年12月(7)

最新评论

1. Re:HashMap(常用)方法个人理解

可以的

--c++天下第一
2. Re:HashMap(常用)方法个人理解

学到了

--本尘
3. Re:idea从零搭建简单的springboot+Mybatis

代码地址

--九号云
4. Re:关于监听微服务功能

@ dgdyq如果一个微服务挂了之后，他会跑出异常，会组织下一个微服务的监听，如要监听多个微服务需要写多个相同的方法。而且如果是微服务发送请求，运营平台进行监听的话，在运营平台只用对比map中各个ke...

--九号云
5. Re:关于监听微服务功能

为什么不是由运营平台发请求，微服务回应？在运营平台上可以各种设置

--dgdyq

阅读排行榜

1. HashMap(常用)方法个人理解(19444)
2. （转）详解URLConnection(1174)
3. NGINX配置之二: nginx location proxy_pass 后面的url 加与不加/的区别.(689)
4. Mysql中“select ... for update”排他锁(转)(471)
5. 易宝支付(369)

评论排行榜

1. HashMap(常用)方法个人理解(2)
2. 关于监听微服务功能(2)
3. idea从零搭建简单的springboot+Mybatis(1)

推荐排行榜

1. HashMap(常用)方法个人理解(5)
2. Mysql中“select ... for update”排他锁(转)(1)

5、根据主键、非主键不含索引（name）进行查询，并且查询到数据，如果其他线程按主键字段进行再次查询，则主键字段产生行锁，如果其他线程按非主键不含索引字段进行查询，则非主键不含索引字段产生表锁，如果其他线程按非主键含索引字段进行查询，则非主键含索引字段产生行锁，如果索引值是枚举类型，mysql也会进行表锁，这段话有点拗口，大家仔细理解一下。

```
1 begin;
2 select * from goods where id = 1 and name='prod11' for update;
3 commit;
```

6、根据主键、非主键不含索引（name）进行查询，没有查询到数据，不产生锁。

```
1 begin;
2 select * from goods where id = 1 and name='prod12' for update;
3 commit;
```

7、根据非主键含索引（name）进行查询，并且查询到数据，name字段产生行锁。

```
1 begin;
2 select * from goods where name='prod11' for update;
3 commit;
```

8、根据非主键含索引（name）进行查询，没有查询到数据，不产生锁。

```
1 begin;
2 select * from goods where name='prod11' for update;
3 commit;
```

9、根据非主键不含索引（name）进行查询，并且查询到数据，name字段产生表锁。

```
1 begin;
2 select * from goods where name='prod11' for update;
3 commit;
```

10、根据非主键不含索引（name）进行查询，没有查询到数据，name字段产生表锁。

```
1 begin;
2 select * from goods where name='prod11' for update;
3 commit;
```

11、只根据主键进行查询，查询条件为不等于，并且查询到数据，主键字段产生表锁。

```
1 begin;
2 select * from goods where id <> 1 for update;
3 commit;
```

12、只根据主键进行查询，查询条件为不等于，没有查询到数据，主键字段产生表锁。

```
1 begin;
2 select * from goods where id <> 1 for update;
3 commit;
```

13、只根据主键进行查询，查询条件为 like，并且查询到数据，主键字段产生表锁。

```
1 begin;
2 select * from goods where id like '1' for update;
3 commit;
```

14、只根据主键进行查询，查询条件为 like，没有查询到数据，主键字段产生表锁。

```
1 begin;
2 select * from goods where id like '1' for update;
3 commit;
```

测试环境

数据库版本：5.1.48-community

数据库引擎：InnoDB Supports transactions, row-level locking, and foreign keys

数据库隔离策略：REPEATABLE-READ（系统、会话）

总结

- 1、InnoDB行锁是通过给索引上的索引项加锁来实现的，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。
- 2、由于MySQL的行锁是针对索引加的锁，不是针对记录加的锁，所以虽然是访问不同行的记录，但是如果是使用相同的索引键，是会出现锁冲突的。应用设计的时候要注意这一点。
- 3、当表有多个索引的时候，不同的事务可以使用不同的索引锁定不同的行，另外，不论是使用主键索引、唯一索引或普通索引，InnoDB都会使用行锁来对数据加锁。
- 4、即便在条件中使用了索引字段，但是否使用索引来检索数据是由MySQL通过判断不同执行计划的代价来决定的，如果MySQL认为全表扫描效率更高，比如对一些很小的表，它就不会使用索引，这种情况下InnoDB将使用表锁，而不是行锁。因此，在分析锁冲突时，别忘了检查SQL的执行计划，以确认是否真正使用了索引。
- 5、检索值的数据类型与索引字段不同，虽然MySQL能够进行数据类型转换，但却不会使用索引，从而导致InnoDB使用表锁。通过用explain检查两条SQL的执行计划，我们可以清楚地看到了这一点。

好文要顶

关注我

收藏该文

九号云

关注 - 0

粉丝 - 1

+加关注

1

推荐

0

反对

« 上一篇：[HashMap\(常用\)方法个人理解](#)
» 下一篇：[2018-5-17](#)

posted @ 2018-05-04 13:46 九号云 阅读(471) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。

- 【推荐】超50万C++/C#源码：大型实时仿真组态图形源码
- 【推荐】华为云·云创校园套餐9元起，小天鹅音箱等你来拿
- 【推荐】零基础轻松玩转云上产品，获赠礼加返百元大礼
- 【推荐】ALIYUN90% | 免认证 9秒注册阿里云 即开即用

相关博文：

- MySQL中select * for update锁表的范围
- MySQL中select * for update锁表的问题
- Mysql_select for update锁详解
- MySQL中select * for update锁表的范围
- MySQL中select * for update锁表的范围

