深入理解乐观锁与悲观锁 (http://www.hollischuang.com/archives/934)

2016-01-05 分类：数据库 (http://www.hollischuang.com/archives/category/%e6%95%b0%e6%8d%ae%e5%ba%93)
阅读(18980) 评论(7)

在数据库的锁机制 (http://www.hollischuang.com/archives/909)中介绍过，数据库管理系统（DBMS）中的并发控制的任务是确保在多个事务同时存取数据库中同一数据时不破坏事务的隔离性和统一性以及数据库的统一性。

乐观并发控制(乐观锁)和悲观并发控制（悲观锁）是并发控制主要采用的技术手段。

无论是悲观锁还是乐观锁，都是人们定义出来的概念，可以认为是一种思想。其实不仅仅是关系型数据库系统中有乐观锁和悲观锁的概念，像memcache、hibernate、tair等都有类似的概念。

针对于不同的业务场景，应该选用不同的并发控制方式。所以，不要把乐观并发控制和悲观并发控制狭义的理解为DBMS中的概念，更不要把他们和数据中提供的锁机制（行锁、表锁、排他锁、共享锁）混为一谈。其实，在DBMS中，悲观锁正是利用数据库本身提供的锁机制来实现的。

下面来分别学习一下悲观锁和乐观锁。

悲观锁

> 在关系数据库管理系统里，悲观并发控制（又名“悲观锁”，Pessimistic Concurrency Control，缩写“PCC”）是一种并发控制的方法。它可以阻止一个事务以影响其他用户的方式来修改数据。如果一个事务执行的操作都某行数据应用了锁，那只有当这个事务把锁释放，其他事务才能够执行与该锁冲突的操作。悲观并发控制主要用于数据争用激烈的环境，以及发生并发冲突时使用锁保护数据的成本要低于回滚事务的成本的环境中。

悲观锁，正如其名，它指的是对数据被外界（包括本系统当前的其他事务，以及来自外部系统的事务处理）修改持保守态度(悲观)，因此，在整个数据处理过程中，将数据处于锁定状态。 悲观锁的实现，往往依靠数据库提供的锁机制 （也只有数据库层提供的锁机制才能真正保证数据访问的排他性，否则，即使在本系统中实现了加锁机制，也无法保证外部系统不会修改数据）

在数据库中，悲观锁的流程如下：

在对任意记录进行修改前，先尝试为该记录加上排他锁(http://www.hollischuang.com/archives/923)（exclusive locking）。

如果加锁失败，说明该记录正在被修改，那么当前查询可能要等待或者抛出异常。 具体响应方式由开发者根据实际需要决定。

如果成功加锁，那么就可以对记录做修改，事务完成后就会解锁了。

其间如果有其他对该记录做修改或加排他锁的操作，都会等待我们解锁或直接抛出异常。

MySQL InnoDB中使用悲观锁

要使用悲观锁，我们必须关闭mysql数据库的自动提交属性，因为MySQL默认使用autocommit模式，也就是说，当你执行一个更新操作后，MySQL会立刻将结果进行提交。 `set autocommit=0;`

```
//0.开始事务
begin;/begin work;/start transaction; (三者选一就可以)
//1.查询出商品信息
select status from t_goods where id=1 for update;
//2.根据商品信息生成订单
insert into t_orders (id,goods_id) values (null,1);
//3.修改商品status为2
update t_goods set status=2;
//4.提交事务
commit;/commit work;
```

上面的查询语句中，我们使用了 `select…for update` 的方式，这样就通过开启排他锁(http://www.hollischuang.com/archives/923)的方式实现了悲观锁。此时在t_goods表中，id为1的那条数据就被我们锁定了，其它的事务必须等本次事务提交之后才能执行。这样我们可以保证当前的数据不会被其它事务修改。

上面我们提到，使用 `select…for update` 会把数据给锁住，不过我们需要注意一些锁的级别，MySQL InnoDB默认行级锁(http://www.hollischuang.com/archives/914)。行级锁都是基于索引的，如果一条SQL语句用不到索引是不会使用行级锁的，会使用表级锁把整张表锁住，这点需要注意。

优点与不足

悲观并发控制实际上是“先取锁再访问”的保守策略，为数据处理的安全提供了保证。但是在效率方面，处理加锁的机制会让数据库产生额外的开销，还有增加产生死锁的机会；另外，在只读型事务处理中由于不会产生冲突，也没必要使用锁，这样做只能增加系统负载；还有会降低了并行性，一个事务如果锁定了某行数据，其他事务就必须等待该事务处理完才可以处理那行数

乐观锁

> 在关系数据库管理系统里，乐观并发控制（又名“乐观锁”，Optimistic Concurrency Control，缩写“OCC”）是一种并发控制的方法。它假设多用户并发的事务在处理时不会彼此互相影响，各事务能够在不产生锁的情况下处理各自影响的那部分数据。在提交数据更新之前，每个事务会先检查在该事务读取数据后，有没有其他事务又修改了该数据。如果其他事务有更新的话，正在提交的事务会进行回滚。乐观事务控制最早是由孔祥重（H.T.Kung）教授提出。

乐观锁（ Optimistic Locking ）相对悲观锁而言，乐观锁假设认为数据一般情况下不会造成冲突，所以在数据进行提交更新的时候，才会正式对数据的冲突与否进行检测，如果发现冲突了，则让返回用户错误的信息，让用户决定如何去做。

相对于悲观锁，在对数据库进行处理的时候，乐观锁并不会使用数据库提供的锁机制。一般的实现乐观锁的方式就是记录数据版本。

> 数据版本,为数据增加的一个版本标识。当读取数据时，将版本标识的值一同读出，数据每更新一次，同时对版本标识进行更新。当我们提交更新的时候，判断数据库表对应记录的当前版本信息与第一次取出来的版本标识进行比对，如果数据库表当前版本号与第一次取出来的版本标识值相等，则予以更新，否则认为是过期数据。

实现数据版本有两种方式，第一种是使用版本号，第二种是使用时间戳。

使用版本号实现乐观锁

使用版本号时，可以在数据初始化时指定一个版本号，每次对数据的更新操作都对版本号执行+1操作。并判断当前版本号是不是该数据的最新的版本号。

```
1.查询出商品信息
select (status,status,version) from t_goods where id=#{id}
2.根据商品信息生成订单
3.修改商品status为2
update t_goods
set status=2,version=version+1
where id=#{id} and version=#{version};
```

优点与不足

乐观并发控制相信事务之间的数据竞争(data race)的概率是比较小的，因此尽可能直接做下去，直到提交的时候才去锁定，所以不会产生任何锁和死锁。但如果直接简单这么做，还是有可能会遇到不可预期的结果，例如两个事务都读取了数据库的某一行，经过修改以后写回数据库，这时就遇到了问题。

参考资料

维基百科-乐观并发控制 (https://zh.wikipedia.org/wiki/%E4%B9%90%E8%A7%82%E5%B9%B6%E5%8F%91%E6%8E%A7%E5%88%B6)

维基百科-悲观并发控制 (https://zh.wikipedia.org/wiki/%E6%82%B2%E8%A7%82%E5%B9%B6%E5%8F%91%E6%8E%A7%E5%88%B6)

mysql悲观锁总结和实践 (http://chenzhou123520.iteye.com/blog/1860954)

mysql乐观锁总结和实践 (http://chenzhou123520.iteye.com/blog/1863407)

乐观锁与悲观锁 (http://www.digpage.com/lock.html)

(全文完)

欢迎关注HollisChuang微信公众账号

标签： 事务 (http://www.hollischuang.com/archives/tag/%e4%ba%8b%e5%8a%a1)

锁 (http://www.hollischuang.com/archives/tag/%e9%94%81)

相关推荐

- MySQL中的读锁和写锁 (http://www.hollischuang.com/archives/1728)

- Java中的事务——JDBC事务和JTA事务 (http://www.hollischuang.com/archives/1658)
- Spring的事务管理机制 (http://www.hollischuang.com/archives/1489)
- 深入分析事务的隔离级别 (http://www.hollischuang.com/archives/943)
- MySQL中的共享锁与排他锁 (http://www.hollischuang.com/archives/923)
- MySQL中的行级锁,表级锁,页级锁 (http://www.hollischuang.com/archives/914)
- 数据库的锁机制 (http://www.hollischuang.com/archives/909)
- 数据库的读现象浅析 (http://www.hollischuang.com/archives/900)

登录

来说两句吧...

评论 **11** 人参与, **7** 条评论

最新评论

Ever [上海市网友] 2017年8月17日 14:59

最后依据 是 有点 问题吧，update version 本身就存在行锁，回写 数据库 是 不会存在 问题的。

回复

chen_h_hui [浙江省杭州市网友] 2017年3月30日 17:44

不会出现aba问题，因为有版本号，即使改回原值，也可以知道到底有没有修改过。

回复

visitor555495129 [江苏省苏州市网友] 2016年12月30日 10:03

> weixin500044310 1
>
> 最后一句我也没理解到, 当两个事务并发时,事务一提交后,如果事务二可以看见事务一提交的version,那么就可以看见事务一修改后的所有数据,这不会有问题吧,如果事务二看不见事务一提交的version,那么事务二因为version字段值不是最新的会更新失败. version是一致赠加的,aba问题不会遇到吧,除非version值空间很小, 乐观锁问题还是有点疑惑???

乐观锁不能防范脏数据

回复

weixin500044310 2016年8月28日 15:34

最后一句我也没理解到,
当两个事务并发时,事务一提交后,如果事务二可以看见事务一提交的version,那么就可以看见事务一修改后的所有数据,这不会有问题吧,如果事务二看不见事务一提交的version,那么事务二因为version字段值不是最新的会更新失败. version是一致赠加的,aba问题不会遇到吧,除非version值空间很小,乐观锁问题还是有点疑惑???

回复

来自HollisChuangwap版

熊初墨啊 [浙江省宁波市网友] 2016年1月7日 21:28

> 面向对象的旅者 1
>
> 最后一句没理解遇到什么问题，请赐教

是指aba的问题，修改了数据库值再改回来你就不知道了，起不到锁的效果

回复 1

来自HollisChuangwap版

HollisChuang 管理员 2016年1月7日 9:16

> 面向对象的旅者 1
>
> 最后一句没理解遇到什么问题，请赐教

多事务并发，如果做不好事务隔离，就有可能导致脏读、不可重复读、幻读等问题。
http://www.hollischuang.com/archives/900

回复

面向对象的旅者 2016年1月7日 4:56

最后一句没理解遇到什么问题，请赐教

回复

来自HollisChuangwap版

HollisChuang正在使用畅言 (http://changyan.kuaizhan.com/)

HollisChuang's Blog

联系我 (http://mail.qq.com/cgi-bin/qm_share?t=qm_mailme&email=-JSTkJCVj5_UiZ2Sm7yNjdKfk5E)

关于我 (/sample-page)