深入分析事务的隔离级别 (http://www.hollischuang.com/archives/943)

2016-01-06 分类：数据库 (http://www.hollischuang.com/archives/category/%e6%95%b0%e6%8d%ae%e5%ba%93)
阅读(13335) 评论(7)

本站采用[知识共享署名-非商业性使用-相同方式共享 许可协议 (http://creativecommons.org/licenses/by-nc-sa/3.0/deed.zh)]进行许可，转载请在 正文明显处 注明原文地址

> 本文详细介绍四种事务隔离级别，并通过举例的方式说明不同的级别能解决什么样的读现象。并且介绍了在关系型数据库中不同的隔离级别的实现原理。

在DBMS中，事务 (http://www.hollischuang.com/archives/898)保证了一个操作序列可以全部都执行或者全部都不执行（原子性），从一个状态转变到另外一个状态（一致性）。由于事务满足久性。所以一旦事务被提交之后，数据就能够被持久化下来，又因为事务是满足隔离性的，所以，当多个事务同时处理同一个数据的时候，多个事务直接是互不影响的，所以，在多个事务并发操作的过程中，如果控制不好隔离级别，就有可能产生脏读 (http://www.hollischuang.com/archives/900)、不可重复读 (http://www.hollischuang.com/archives/900)或者幻读 (http://www.hollischuang.com/archives/900)等读现象。

在数据库事务的ACID (http://www.hollischuang.com/archives/898)四个属性中，隔离性是一个最常放松的一个。可以在数据操作过程中利用数据库的锁机制或者多版本并发控制机制获取更高的隔离等级。但是，随着数据库隔离级别的提高，数据的并发能力也会有所下降。所以，如何在并发性和隔离性之间做一个很好的权衡就成了一个至关重要的问题。

在软件开发中，几乎每类这样的问题都会有多种最佳实践来供我们参考，很多DBMS定义了多个不同的“事务隔离等级”来控制锁 (http://www.hollischuang.com/archives/909)的程度和并发能力。

ANSI/ISO SQL定义的标准隔离级别有四种，从高到底依次为：可序列化(Serializable)、可重复读(Repeatable reads)、提交读(Read committed)、未提交读(Read uncommitted)。

下面将依次介绍这四种事务隔离级别的概念、用法以及解决了哪些问题（读现象）

未提交读(Read uncommitted)

未提交读(READ UNCOMMITTED)是最低的隔离级别。通过名字我们就可以知道，在这种事务隔离级别下，一个事务可以读到另外一个事务未提交的数据。

未提交读的数据库锁情况（实现原理）

事务在读数据的时候并未对数据加锁。

务在修改数据的时候只对数据增加行级(http://www.hollischuang.com/archives/914)共享锁(http://www.hollischuang.com/archives/923)。

现象：

事务1读取某行记录时，事务2也能对这行记录进行读取、更新（因为事务一并未对数据增加任何锁）

当事务2对该记录进行更新时，事务1再次读取该记录，能读到事务2对该记录的修改版本（因为事务二只增加了共享读锁，事务一可以再增加共享读锁读取数据），即使该修改尚未被提交。

事务1更新某行记录时，事务2不能对这行记录做更新，直到事务1结束。（因为事务一对数据增加了共享读锁，事务二不能增加排他写锁(http://www.hollischuang.com/archives/923)进行数据的修改）

举例

下面还是借用我在数据库的读现象浅析 (http://www.hollischuang.com/archives/900)一文中举的例子来说明在未提交读的隔离级别中两个事务之间的隔离情况。

事务一	事务二
/* Query 1 */ SELECT age FROM users WHERE id = 1; /* will read 20 */	
	/* Query 2 */ UPDATE users SET age = 21 WHERE id = 1; /* No commit here */

/* Query 1 */ SELECT age FROM users WHERE id = 1; /* will read 21 */	
	ROLLBACK; /* lock-based DIRTY READ */

事务一共查询了两次，在两次查询的过程中，事务二对数据进行了修改，并未提交（commit）。但是事务一的第二次查询查到了事务二的修改结果。在数据库的读现象浅析中我们介绍过，这种现象我们称之为脏读 (http://www.hollischuang.com/archives/900)。

所以，**未提交读会导致脏读 (http://www.hollischuang.com/archives/900)**

提交读(Read committed)

提交读(READ COMMITTED)也可以翻译成读已提交，通过名字也可以分析出，在一个事务修改数据过程中，如果事务还没提交，其他事务不能读该数据。

提交读的数据库锁情况

> 事务对当前被读取的数据加 行级共享锁（当读到时才加锁），一旦读完该行，立即释放该行级共享锁；
>
> 事务在更新某数据的瞬间（就是发生更新的瞬间），必须先对其加 行级排他锁，直到事务结束才释放。

现象：

事务1在读取某行记录的整个过程中，事务2都可以对该行记录进行读取（因为事务一对该行记录增加行级共享锁的情况下，事务二同样可以对该数据增加共享锁来读数据。）。

事务1读取某行的一瞬间，事务2不能修改该行数据，但是，只要事务1读取完改行数据，事务2就可以对该行数据进行修改。（事务一在读取的一瞬间会对数据增加共享锁，任何其他事务都不能对该行数据增加排他锁。但是事务一只要读完该行数据，就会释放行级共享锁，一旦锁释放，事务二就可以对数据增加排他锁并修改数据）

事务1更新某行记录时，事务2不能对这行记录做更新，直到事务1结束。（事务一在更新数据的时候，会对该行数据增加排他锁，知道事务结束才会释放锁，所以，在事务二没有提交之前，事务一都能不对数据增加共享锁进行数据的读取。**所以，提交读可以解决 脏读 的现象**）

举例

事务一	事务二
/* Query 1 */ SELECT * FROM users WHERE id = 1;	
	/* Query 2 */ UPDATE users SET age = 21 WHERE id = 1; COMMIT; /* in multiversion concurrency control, or lock-based READ COMMITTED */
/* Query 1 */ SELECT * FROM users WHERE id = 1; COMMIT; /*lock-based REPEATABLE READ */	

在提交读隔离级别中，在事务二提交之前，事务一不能读取数据。只有在事务二提交之后，事务一才能读数据。

但是从上面的例子中我们也看到，事务一两次读取的结果并不一致，**所以提交读不能解决不可重复读的读现象 (http://www.hollischuang.com/archives/900)。**

简而言之，提交读这种隔离级别保证了读到的任何数据都是提交的数据，避免了脏读(dirty reads)。但是不保证事务重新读的时候能读到相同的数据，因为在每次数据读完之后其他事务可以修改刚才读到的数据。

可重复读(Repeatable reads)

可重复读(REPEATABLE READS),由于提交读隔离级别会产生不可重复读的读现象。所以，比提交读更高一个级别的隔离级别就可以解决不可重复读的问题。这种隔离级别就叫可重复读（这名字起的是不是很任性！！）

可重复读的数据库锁情况

> 事务在读取某数据的瞬间（就是开始读取的瞬间），必须先对其加 行级共享锁，直到事务结束才释放；
>
> 事务在更新某数据的瞬间（就是发生更新的瞬间），必须先对其加 行级排他锁，直到事务结束才释放。

现象

> 事务1在读取某行记录的整个过程中，事务2都可以对该行记录进行读取（因为事务一对该行记录增加行级共享锁的情况下，事务二同样可以对该数据增加共享锁来读数据。）。
>
> 事务1在读取某行记录的整个过程中，事务2都不能修改该行数据（事务一在读取的整个过程会对数据增加共享锁，直到事务提交才会释放锁，所以整个过程中，任何其他事务都不能对该行数据增加排他锁。**所以，可重复读能够解决 不可重复读 的读现象**）
>
> 事务1更新某行记录时，事务2不能对这行记录做更新，直到事务1结束。（事务一在更新数据的时候，会对该行数据增加排他锁，知道事务结束才会释放锁，所以，在事务二没有提交之前，事务一都能不对数据增加共享锁进行数据的读取。**所以，提交读可以解决 脏读 的现象**）

举例

事务一	事务二
/* Query 1 */ SELECT * FROM users WHERE id = 1; COMMIT;	
	/* Query 2 */ UPDATE users SET age = 21 WHERE id = 1; COMMIT; /* in multiversion concurrency control, or lock-based READ COMMITTED */

在上面的例子中，只有在事务一提交之后，事务二才能更改该行数据。所以，只要在事务一从开始到结束的这段时间内，无论他读取该行数据多少次，结果都是一样的。

从上面的例子中我们可以得到结论：可重复读隔离级别可以解决不可重复读的读现象。**但是可重复读这种隔离级别中，还有另外一种读现象他解决不了，那就是幻读(http://www.hollischuang.com/archives/900)。**看下面的例子：

事务一	事务二
/* Query 1 */ SELECT * FROM users WHERE age BETWEEN 10 AND 30;	
	/* Query 2 */ INSERT INTO users VALUES (3, 'Bob', 27); COMMIT;
/* Query 1 */ SELECT * FROM users WHERE age BETWEEN 10 AND 30;	

上面的两个事务执行情况及现象如下：

1.事务一的第一次查询条件是 `age BETWEEN 10 AND 30;` 如果这是有十条记录符合条件。这时，他会给符合条件的这十条记录增加行级共享锁。任何其他事务无法更改这十条记录。

2.事务二执行一条sql语句，语句的内容是向表中插入一条数据。因为此时没有任何事务对表增加表级锁(http://www.hollischuang.com/archives/914)，所以，该操作可以顺利执行。

3.事务一再次执行 `SELECT * FROM users WHERE age BETWEEN 10 AND 30;` 时，结果返回的记录变成了十一条，比刚刚增加了一条，增加的这条正是事务二刚刚插入的那条。

所以，事务一的两次范围查询结果并不相同。这也就是我们提到的幻读。

可序列化(Serializable)

可序列化(Serializable)是最高的隔离级别，前面提到的所有的隔离级别都无法解决的幻读，在可序列化的隔离级别中可以解决。

我们说过，产生幻读的原因是事务一在进行范围查询的时候没有增加范围锁(range-locks：给SELECT的查询中使用一个“WHERE”子句描述范围加锁），所以导致幻读。

可序列化的数据库锁情况

> 事务在读取数据时，必须先对其加 表级共享锁 ，直到事务结束才释放；
>
> 事务在更新数据时，必须先对其加 表级排他锁 ，直到事务结束才释放。

现象

> 事务1正在读取A表中的记录时，则事务2也能读取A表，但不能对A表做更新、新增、删除，直到事务1结束。(因为事务一对表增加了表级共享锁，其他事务只能增加共享锁读取数据，不能进行其他任何操作）
>
> 事务1正在更新A表中的记录时，则事务2不能读取A表的任意记录，更不可能对A表做更新、新增、删除，直到事务1结束。（事务一对表增加了表级排他锁，其他事务不能对表增加共享锁或排他锁，也就无法进行任何操作）

虽然可序列化解决了脏读、不可重复读、幻读等读现象。但是序列化事务会产生以下效果：

1.无法读取其它事务已修改但未提交的记录。

查看 8

2.在当前事务完成之前，其它事务不能修改目前事务已读取的记录。

3.在当前事务完成之前，其它事务所插入的新记录，其索引键值不能在当前事务的任何语句所读取的索引键范围中。

四种事务隔离级别从隔离程度上越来越高，但同时在并发性上也就越来越低。之所以有这么几种隔离级别，就是为了方便开发人员在开发过程中根据业务需要选择最合适的隔离级别。

参考资料

维基百科-事务隔离 (https://zh.wikipedia.org/wiki/%E4%BA%8B%E5%8B%99%E9%9A%94%E9%9B%A2)

数据库隔离级别 及 其实现原理 (http://my.oschina.net/HuQingmiao/blog/518101)

【公告】版权声明 (http://www.hollischuang.com/转载说明)

(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)

分享到： 更多 (3)

相关推荐

- 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)

- 深入理解乐观锁与悲观锁 (http://www.hollischuang.com/archives/934)
- 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)
- 彻底理解数据库事务 (http://www.hollischuang.com/archives/898)

登录

来说两句吧...

评论

8 人参与, 8 条评论

最新评论

三折其肱 2017年3月3日 22:38

小神蛋 [北京市网友] 1

未提交读时， 事务二在修改数据时对数据加了共享锁，所以事务一可以加共享锁，所以可以读，但是看你文章中写道共享锁是不能修改数据，只能读取数据的，这是怎么回事

三折其肱 2

ru

我来解决你的问题：
1.你需要明白前置条件，共享读锁的意思是：同时有多个线程（事务）去读数据，如果此时有排他写锁对这行数据加锁修改，等同的效果就是没有加任何类型的锁，因为你对数据既可以读，也可以写。
2.排他写锁与共享读锁是完全互斥的，没有任何交集，在共享读的时候不能被写操作影响，要不共同读的线程获取的数据都不一样，在排他写数据的时候，其实就是同步加锁，同一时刻只能由一个线程(事务)做操作，其他线程(事务)的任何写与读操作都是不允许的。

回复

光涯 [北京市网友] 2017年9月18日 11:16

> 光涯 [湖南省网友] 1
>
> 可重复读时候，两个事务里面都有读和写的操作，有条记录A，事务1读完后，事务2读，事务1和事务2里面都有对A数据写的操作的话，后续是怎么的处理逻辑。事务1和2都对A记录加了共享锁，直至事务结束，很迷惑，求解答！谢谢

事务1和2都有共享锁，直至事务结束，那事务1和2就一直在相互等待对方释放共享锁的状态，一直等待着吗，那不是死循环了吗。求解答，谢谢！

回复

光涯 [湖南省网友] 2017年9月18日 11:13

可重复读时候，两个事务里面都有读和写的操作，有条记录A，事务1读完后，事务2读，事务1和事务2里面都有对A数据写的操作的话，后续是怎么的处理逻辑。事务1和2都对A记录加了共享锁，直至事务结束，很迷惑，求解答！谢谢

回复

三折其肱 2017年3月3日 22:27

> 小神蛋 [北京市网友] 1
>
> 未提交读时， 事务二在修改数据时对数据加了共享锁，所以事务一可以加共享锁，所以可以读，但是看你文章中写道共享锁是不能修改数据，只能读取数据的，这是怎么回事

ru

回复

小神蛋 [北京市网友] 2017年1月12日 22:26

未提交读时， 事务二在修改数据时对数据加了共享锁，所以事务一可以加共享锁，所以可以读，但是看你文章中写道共享锁是不能修改数据，只能读取数据的，这是怎么回事

回复

貌恭神定 [北京市网友] 2016年8月8日 22:56

浅显易懂，niubility

回复

歌笙逝 [浙江省网友] 2016年8月3日 23:10

请教个问题，在未提交读的级别时，当事务1在更新数据时，对数据进行了行级共享锁，其他事务由于也在读取这行数据，因此也都对这个数据增加了行级共享锁，那么，当事务1结束的时候，会释放行级共享锁，其它事务的行级共享锁该何时释放呢？

回复

益群网 2016年1月7日 12:46

【益群网：逆向网赚，坐等收钱】

【系统优势：】
优势1：静态分红，每日签到就有钱，每日最高一百元
优势2：十级提成，逆向网赚，什么不干，照样有钱赚
优势3：百万资源，永久更新，可一键转存到自己网盘

回复

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)

© 2018
HollisChuang's Blog (http://www.hollischuang.com)