WIKIPEDIA

事务隔离

维基百科，自由的百科全书

事务隔离（英语：Transaction Isolation）定义了数据库系统中一个操作的结果在何时以何种方式对其他并发操作可见。隔离是事务ACID（原子性、一致性、隔离性、持久性）四大属性之一。

目录

并发控制

隔离级别

- 可串行化
- 可重复读
- 提交读
- 未提交读

默认隔离级别

读现象举例

- 脏读
- 不可重复读
- 幻影读

隔离级别、读现象和锁

- 隔离级别vs读现象
- 隔离级别vs 锁持续时间

参考文献

相关条目

外部链接

并发控制

并发控制描述了数据库事务隔离以保证数据正确性的机制。为了保证并行事务执行的准确执行，数据库和存储引擎在设计的时候着重强调了并发控制这一点。典型的事务相关机制限制数据的访问顺序（执行调度）以满足可序列化 和可恢复性。限制数据访问意味着降低了执行的性能，并发控制机制就是要保证在满足这些限制的前提下提供尽可能高的性能。经常在不损害正确性的情况下，为了达到更好的性能，可序列化的要求会减低一些，但是为了避免数据一致性的破坏，可恢复性必须保证。

两阶段锁是关系数据库中最常见的提供了可序列化和可恢复性的并发控制机制，为了访问一个数据库对象，事务首先要获得这个对象的锁。对于不同的访问类型（如对对象的读写操作）和锁的类型，如果另外一个事务正持有这个对象的锁，获得锁的过程会被阻塞或者延迟。

隔离级别

在数据库事务的ACID四个属性中，隔离性是一个最常放松的一个。为了获取更高的隔离等级，数据库系统的锁机制或者多版本并发控制机制都会影响并发。 应用软件也需要额外的逻辑来使其正常工作。

很多数据库管理系统定义了不同的“事务隔离等级”来控制锁的程度。在很多数据库系统中，多数的数据库事务都避免高等级的隔离等级（如可串行化）从而减少对系统的锁定开销。程序员需要小心的分析数据库访问部分的代码来保证隔离级别的降低不会造成难以发现的代码bug。相反的，更高的隔离级别会增加死锁发生的几率，同样需要编程过程中去避免。

ANSI/ISO SQL定义的标准隔离级别如下：

可串行化

最高的隔离级别。

在基于锁机制并发控制的DBMS实现可串行化，要求在选定对象上的读锁和写锁保持直到事务结束后才能释放。在SELECT 的查询中使用一个“WHERE”子句来描述一个范围时应该获得一个“范围锁”（range-locks）。这种机制可以避免“幻读”（phantom reads）现象（详见下文）。

当采用不基于锁的并发控制时不用获取锁。但当系统探测到几个并发事务有“写冲突”的时候，只有其中一个是允许提交的。这种机制的详细描述见“快照隔离”

可重复读

在可重复读（REPEATABLE READS）隔离级别中，基于锁机制并发控制的DBMS需要对选定对象的读锁（read locks）和写锁（write locks）一直保持到事务结束，但不要求“范围锁”，因此可能会发生“幻影读”。

提交读

在提交读（READ COMMITTED）级别中，基于锁机制并发控制的DBMS需要对选定对象的写锁一直保持到事务结束，但是读锁在SELECT操作完成后马上释放（因此“不可重复读”现象可能会发生，见下面描述）。和前一种隔离级别一样，也不要求“范围锁”。

未提交读

未提交读（READ UNCOMMITTED）是最低的隔离级别。允许“脏读”（dirty reads），事务可以看到其他事务“尚未提交”的修改。

通过比低一级的隔离级别要求更多的限制，高一级的级别提供更强的隔离性。标准允许事务运行在更强的事务隔离级别上。(如在可重复读隔离级别上执行提交读的事务是没有问题的)

默认隔离级别

不同的DBMS默认隔离级别也不同。大多数据库允许用户设置隔离级别。有些DBMS在执行一个SELECT语句时使用额外的语法来获取锁（如*SELECT ... FOR UPDATE*来获得在访问的数据行上的排他锁）。

读现象举例

ANSI/ISO SQL 92标准涉及三种不同的一个事务读取另外一个事务可能修改的数据的“读现象”。

下面的例子中，两个事务，事务1执行语句1。接着，事务2执行语句2并且提交，最后事务1再执行语句1. 查询使用如下的数据表。

users

id	name	age
1	Joe	20
2	Jill	25

脏读

当一个事务允许读取另外一个事务修改但未提交的数据时，就可能发生脏读。

脏读和不可重复读（non-repeatable reads）类似。事务2没有提交造成事务1的语句1两次执行得到不同的结果集。在未提交读隔离级别唯一禁止的是更新混乱，即早期的更新可能出现在后来更新之前的结果集中。

在我们的例子中，事务2修改了一行，但是没有提交，事务1读了这个没有提交的数据。现在如果事务2回滚了刚才的修改或者做了另外的修改的话，事务1中查到的数据就是不正确的了。

事务 1 **事务 2**

```
/* 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 */
```

在这个例子中，事务2回滚后就没有id是1，age是21的数据行了。

不可重复读

在一次事务中，当一行数据获取两遍得到不同的结果表示发生了“不可重复读”.

在基于锁的并发控制中“不可重复读”现象发生在当执行SELECT 操作时没有获得读锁或者SELECT操作执行完后马上释放了读锁； 多版本并发控制中当没有要求一个提交冲突(commit conflict)的事务回滚也会发生“不可重复读”现象。

事务 1 **事务 2**

```
/* 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 */
```

在这个例子中，事务2提交成功，因此他对id为1的行的修改就对其他事务可见了。但是事务1在此前已经从这行读到了另外一个“age”的值。在可序列化（SERIALIZABLE）和可重复读的隔离级别，数据库在第二次SELECT请求的时候应该返回事务2更新之前的值。在提交读和未提交读，返回的是更新之后的值，这个现象就是不可重复读。

有两种策略可以避免不可重复读。一个是要求事务2延迟到事务1提交或者回滚之后再执行。这种方式实现了**T1, T2** 的串行化调度。串行化调度可以支持可重复读。

另一种策略是*多版本并发控制*。为了得到更好的并发性能，允许事务2先提交。但因为事务1在事务2之前开始，事务1必须在其开始执行时间点的数据库的快照上面操作。当事务1最终提交时候，数据库会检查其结果是否等价于**T1, T2**串行调度。如果等价，则允许事务1提交，如果不等价，事务1需要回滚并抛出个串行化失败的错误。

使用基于锁的并发控制，在可重复读的隔离级别中，ID=1的行会被锁住，在事务1提交或回滚前一直阻塞语句2的执行。在提交读的级别，语句1第二次执行，age已经被修改了。

在*多版本并发控制*机制下，可序列化(SERIALIZABLE)级别，两次SELECT语句读到的数据都是事务1开始的快照，因此返回同样的数据。但是，如果事务1试图UPDATE这行数据，事务1会被要求回滚并抛出一个串行化失败的错误。

在提交读隔离级别，每个语句读到的是语句执行前的快照，因此读到更新前后不同的值。在这种级别不会有串行化的错误（因为这种级别不要求串行化），事务1也不要求重试。

幻影读

在事务执行过程中，当两个完全相同的查询语句执行得到不同的结果集。这种现象称为“幻影读（phantom read）”

当事务没有获取*范围锁*的情况下执行*SELECT ... WHERE*操作可能会发生“幻影读”。

“幻影读”是*不可重复读*的一种特殊场景：当事务1两次执行*SELECT ... WHERE*检索一定范围内数据的操作中间，事务2在这个表中创建了(如INSERT)了一行新数据，这条新数据正好满足事务1的“WHERE”子句。

事务 1 **事务 2**

```
/* 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;
```

需要指出的是事务1执行了两遍同样的查询语句。如果设了最高的隔离级别，两次会得到同样的结果集，这也正是数据库在可序列化（SERIALIZABLE）隔离级别上需要满足的。但是在较低的隔离级别上，第二次查询可能会得到不同的结果集。

在可序列化隔离级别，查询语句1在age从10到30的记录上加锁，事务2只能阻塞直至事务1提交。在可重复读级别，这个范围不会被锁定，允许记录插入，因此第二次执行语句1的结果中会包括新插入的行。

隔离级别、读现象和锁

隔离级别vs读现象

隔离级别	脏读	不可重复读	幻影读
未提交读	可能发生	可能发生	可能发生
提交读	-	可能发生	可能发生
可重复读	-	-	可能发生
可序列化	-	-	-

可序列化（Serializable）隔离级别不等同于可串行化（Serializable）。可串行化调度是避免以上三种现象的必要条件，但不是充分条件。

“可能发生”表示这个隔离级别会发生对应的现象，“-”表示不会发生。

隔离级别vs 锁持续时间

在基于锁的并发控制中，隔离级别决定了锁的持有时间。**"C"**-表示锁会持续到事务提交。 **"S"** –表示锁持续到当前语句执行完毕。如果锁在语句执行完毕就释放则另外一个事务就可以在这个事务提交前修改锁定的数据，从而造成混乱。

隔离级别I	写操作	读操作	范围操作 (...where...)
未提交读	S	S	S
提交读	C	S	S
可重复读	C	C	S
可序列化	C	C	C

参考文献

相关条目

- 原子性
- 一致性
- 持久性
- 锁
- 乐观并发控制
- 关系数据库
- 快照隔离

外部链接

- Oracle® Database Concepts (http://docs.oracle.com/cd/B12037_01/server.101/b10743/toc.htm), chapter 13 Data Concurrency and Consistency, Preventable Phenomena and Transaction Isolation Levels (http://docs.oracle.com/cd/B12037_01/server.101/b10743/consist.htm#sthref1919)
- Oracle® Database SQL Reference (http://docs.oracle.com/cd/B19306_01/server.102/b14200/toc.htm), chapter 19 SQL Statements: SAVEPOINT to UPDATE (http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_10.htm#i2068385), SET TRANSACTION (http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_10005.htm#i2067247)
- in JDBC: Connection constant fields (http://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html#field_summary), Connection.getTransactionIsolation() (http://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html#getTransactionIsolation()), Connection.setTransactionIsolation(int) (http://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html#setTransactionIsolation(int))
- in Spring Framework: @Transactional (http://static.springsource.org/spring/docs/current/javadoc-api/org/springframework/transaction/annotation/Transactional.html), Isolation (http://static.springsource.org/spring/docs/current/javadoc-api/org/springframework/transaction/annotation/Isolation.html)
