07 事务

在执行SQL语句的时候，某些业务要求，一系列操作必须全部执行，而不能仅执行一部分。例如，一个转账操作：

```
-- 从id=1的账户给id=2的账户转账100元
-- 第一步：将id=1的A账户余额减去100
UPDATE accounts SET balance = balance - 100 WHERE id = 1;
-- 第二步：将id=2的B账户余额加上100
UPDATE accounts SET balance = balance + 100 WHERE id = 2;
```

这两条SQL语句必须全部执行，或者，由于某些原因，如果第一条语句成功，第二条语句失败，就必须全部撤销。

这种把多条语句作为一个整体进行操作的功能，被称为数据库*事务*。数据库事务可以确保该事务范围内的所有操作都可以全部成功或者全部失败。如果事务失败，那么效果就和没有执行这些SQL一样，不会对数据库数据有任何改动。

可见，数据库事务具有ACID这4个特性：

- A：Atomic，原子性，将所有SQL作为原子工作单元执行，要么全部执行，要么全部不执行；
- C：Consistent，一致性，事务完成后，所有数据的状态都是一致的，即A账户只要减去了100，B账户则必定加上了100；
- I：Isolation，隔离性，如果有多个事务并发执行，每个事务作出的修改必须与其他事务隔离；
- D：Duration，持久性，即事务完成后，对数据库数据的修改被持久化存储。

对于单条SQL语句，数据库系统自动将其作为一个事务执行，这种事务被称为*隐式事务*。

要手动把多条SQL语句作为一个事务执行，使用`BEGIN`开启一个事务，使用`COMMIT`提交一个事务，这种事务被称为*显式事务*，例如，把上述的转账操作作为一个显式事务：

```
BEGIN;
UPDATE accounts SET balance = balance - 100 WHERE id = 1;
UPDATE accounts SET balance = balance + 100 WHERE id = 2;
COMMIT;
```

很显然多条SQL语句要想作为一个事务执行，就必须使用显式事务。

`COMMIT`是指提交事务，即试图把事务内的所有SQL所做的修改永久保存。如果`COMMIT`语句执行失败了，整个事务也会失败。

有些时候，我们希望主动让事务失败，这时，可以用`ROLLBACK`回滚事务，整个事务会失败：

```
BEGIN;
UPDATE accounts SET balance = balance - 100 WHERE id = 1;
UPDATE accounts SET balance = balance + 100 WHERE id = 2;
ROLLBACK;
```

数据库事务是由数据库系统保证的，我们只需要根据业务逻辑使用它就可以。

隔离级别

对于两个并发执行的事务，如果涉及到操作同一条记录的时候，可能会发生问题。因为并发操作会带来数据的不一致性，包括脏读、不可重复读、幻读等。数据库系统提供了隔离级别来让我们有针对性地选择事务的隔离级别，避免数据不一致的问题。

SQL标准定义了4种隔离级别，分别对应可能出现的数据不一致的情况：

ISOLATION LEVEL	脏读（DIRTY READ）	不可重复读（NON REPEATABLE READ）	幻读（PHANTOM READ）
Read Uncommitted	Yes	Yes	Yes
Read Committed	-	Yes	Yes
Repeatable Read	-	-	Yes
Serializable	-	-	-

我们会依次介绍4种隔离级别的数据一致性问题。

小结

数据库事务具有ACID特性，用来保证多条SQL的全部执行。

Read Uncommitted

Read Uncommitted是隔离级别最低的一种事务级别。在这种隔离级别下，一个事务会读到另一个事务更新后但未提交的数据，如果另一个事务回滚，那么当前事务读到的数据就是脏数据，这就是脏读（Dirty Read）。

我们来看一个例子。

首先，我们准备好`students`表的数据，该表仅一行记录：

```
mysql> select * from students;
+----+-------+
| id | name  |
+----+-------+
|  1 | Alice |
+----+-------+
1 row in set (0.00 sec)
```

然后，分别开启两个MySQL客户端连接，按顺序依次执行事务A和事务B：

时刻	事务A	事务B
1	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
2	BEGIN;	BEGIN;
3	UPDATE students SET name = 'Bob' WHERE id = 1;	
4		SELECT * FROM students WHERE id = 1;
5	ROLLBACK;	
6		SELECT * FROM students WHERE id = 1;
7		COMMIT;

当事务A执行完第3步时，它更新了`id=1`的记录，但并未提交，而事务B在第4步读取到的数据就是未提交的数据。

随后，事务A在第5步进行了回滚，事务B再次读取`id=1`的记录，发现和上一次读取到的数据不一致，这就是脏读。

可见，在Read Uncommitted隔离级别下，一个事务可能读取到另一个事务更新但未提交的数据，这个数据有可能是脏数据。

Read Committed

在Read Committed隔离级别下，一个事务可能会遇到不可重复读（Non Repeatable Read）的问题。

不可重复读是指，在一个事务内，多次读同一数据，在这个事务还没有结束时，如果另一个事务恰好修改了这个数据，那么，在第一个事务中，两次读取的数据就可能不一致。

我们仍然先准备好`students`表的数据：

```
mysql> select * from students;
+----+-------+
| id | name  |
+----+-------+
|  1 | Alice |
+----+-------+
1 row in set (0.00 sec)
```

然后，分别开启两个MySQL客户端连接，按顺序依次执行事务A和事务B：

时刻	事务A	事务B
1	SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
2	BEGIN;	BEGIN;
3		SELECT * FROM students WHERE id = 1;
4	UPDATE students SET name = 'Bob' WHERE id = 1;	
5	COMMIT;	
6		SELECT * FROM students WHERE id = 1;
7		COMMIT;

当事务B第一次执行第3步的查询时，得到的结果是`Alice`，随后，由于事务A在第4步更新了这条记录并提交，所以，事务B在第6步再次执行同样的查询时，得到的结果就变成了`Bob`，因此，在Read Committed隔离级别下，事务不可重复读同一条记录，因为很可能读到的结果不一致。

Repeatable Read

在Repeatable Read隔离级别下，一个事务可能会遇到幻读（Phantom Read）的问题。

幻读是指，在一个事务中，第一次查询某条记录，发现没有，但是，当试图更新这条不存在的记录时，竟然能成功，并且，再次读取同一条记录，它就神奇地出现了。

我们仍然先准备好`students`表的数据：

```
mysql> select * from students;
+----+-------+
| id | name  |
+----+-------+
|  1 | Alice |
+----+-------+
1 row in set (0.00 sec)
```

然后，分别开启两个MySQL客户端连接，按顺序依次执行事务A和事务B：

时刻	事务A	事务B
1	SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;	SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
2	BEGIN;	BEGIN;
3		SELECT * FROM students WHERE id = 99;
4	INSERT INTO students (id, name) VALUES (99, 'Bob');	
5	COMMIT;	
6		SELECT * FROM students WHERE id = 99;
7		UPDATE students SET name = 'Alice' WHERE id = 99;
8		SELECT * FROM students WHERE id = 99;
9		COMMIT;

事务B在第3步第一次读取`id=99`的记录时，读到的记录为空，说明不存在`id=99`的记录。随后，事务A在第4步插入了一条`id=99`的记录并提交。事务B在第6步再次读取`id=99`的记录时，读到的记录仍然为空，但是，事务B在第7步试图更新这条不存在的记录时，竟然成功了，并且，事务B在第8步再次读取`id=99`的记录时，记录出现了。

可见，幻读就是没有读到的记录，以为不存在，但其实是可以更新成功的，并且，更新成功后，再次读取，就出现了。

Serializable

Serializable是最严格的隔离级别。在Serializable隔离级别下，所有事务按照次序依次执行，因此，脏读、不可重复读、幻读都不会出现。

虽然Serializable隔离级别下的事务具有最高的安全性，但是，由于事务是串行执行，所以效率会大大下降，应用程序的性能会急剧降低。如果没有特别重要的情景，一般都不会使用Serializable隔离级别。

默认隔离级别

如果没有指定隔离级别，数据库就会使用默认的隔离级别。在MySQL中，如果使用InnoDB，默认的隔离级别是Repeatable Read。