数据库事务概括

1. 说明

一组 SQL,一个逻辑工作单位，执行时整体修改或者整体回退。

2.事务相关概念

1）事务的提交和回滚：COMMIT/ROLLBACK

2）事务的开始和结束

开始事务：连接到数据库，执行 DML、DCL、DDL 语句

结束事务： 1. 执行 DDL(例如 CREATE TABLE),DCL(例如 GRANT),系统自动执行 COMMIT 语句

2. 执行 COMMIT/ROLLBACK

3. 退出/断开数据库的连接自动执行 COMMIT 语句

4. 进程意外终止，事务自动 rollback

5. 事务 COMMIT 时会生成一个唯一的系统变化号（SCN）保存到事务表

3）保存点（savepoint）： 可以在事务的任何地方设置保存点，以便 ROLLBACK

4）事务的四个特性 ACID ：

1. Atomicity（原子性）：事务中 sql 语句不可分割，要么都做，要么都不做

2. Consistency(一致性) ： 指事务操作前后，数据库中数据是一致的，数据满足业务规则约束（例如账户金额的转出和转入），与原子性对应。

3. Isolation（隔离性）：多个并发事务可以独立运行，而不能相互干扰，一个事务修改数据未提交前，其他事务看不到它所做的更改。

4. Durability（持久性）：事务提交后，数据的修改是永久的。

5） 死锁：当两个事务相互等待对方释放资源时，就会形成死锁，下面章节详细分析

oracle 事务隔离级别

1 .两个事务并发访问数据库数据时可能存在的问题

1. 幻想读：

事务 T1 读取一条指定 where 条件的语句，返回结果集。此时事务 T2 插入一行新记录并 commit，恰好满足 T1 的 where 条件。然后 T1 使用相同的条件再次查询，结果集中可以看到 T2 插入的记录，这条新纪录就是幻想。

2. 不可重复读取：

事务 T1 读取一行记录，紧接着事务 T2 修改了 T1 刚刚读取的记录并 commit，然后 T1 再次查询，发现与第一次读取的记录不同，这称为不可重复读。

3. 脏读：

事务 T1 更新了一行记录，还未提交所做的修改，这个 T2 读取了更新后的数据，然后 T1 执行回滚操作，取消刚才的修改，所以 T2 所读取的行就无效，也就是脏数据。

2.oracle 事务隔离级别

oracle 支持的隔离级别：（不支持脏读）

```
READ COMMITTED--不允许脏读，允许幻想读和不可重复读
```

```
SERIALIZABLE--以上三种都不允许
```

sql 标准还支持 READ UNCOMMITTED (三种都允许)和 REPEATABLE READ（不允许不可重复读和脏读，只允许幻想读）

以上区别在下面章节事务建立，隔离级别分析中说明

事务相关语句

1. 事务相关语句概括

```
SET TRANSACTION----设置事务属性 SET CONSTRAINT -----设置约束
SAVEPOINT ------------建立存储点
RELEASE SAVEPOINT --释放存储点 ROLLBACK---------------回滚
COMMIT-------------------提交
```

2. 建立事务、隔离级别分析

1）建立事务

```
SET TRANSACTION READ ONLY--事务中不能有任何修改数据库中数据的操作语句，这包括 insert、update、delete、create 语句
SET TRANSACTION READ WRITE--默认设置,该选项表示在事务中可以有访问语句、修改语句
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE--serialzable 可以执行 DML 操作
```

注意：这些语句是互斥的，不能够同时设置两个或者两个以上的选项

2）read only

eg：

```
set transaction read only;
select * from student;
```

结果：

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	女	1	1	100
2	ccc	小李	男	2	2	70
3	hhh	喵喵	女	7	4	60
4	eee	果果	女	5	8	60
5	fff	多多	女	7	2	(null)
6	ggg	钱钱	女	7	3	60
7	bbb	小花	女	1	1	80
8	ddd	小雨	女	5	8	60

执行：

```
update student set name='小丸子' where id='ccc';
```

结果：

```
在行 3 上开始执行命令时出错:update student set name='小丸子' where id='ccc'
错误报告:
SQL 错误: ORA-01456: 不能在 READ ONLY 事务处理中执行插入/删除/更新操作 01456.
00000 -  "may not perform insert/delete/update operation inside a READ ONLY
transaction"*Cause:    A non-DDL insert/delete/update or select for update
operation
         was attempted*Action:   commit (or rollback) transaction, and
re-execute
```

3） read write

eg：

```
set transaction read write;select * from student;
```

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	女	1	1	100
2	ccc	小李	男	2	2	70
3	hhh	喵喵	女	7	4	60
4	eee	果果	女	5	8	60
5	fff	多多	女	7	2	(null)
6	ggg	钱钱	女	7	3	60
7	bbb	小花	女	1	1	80
8	ddd	小雨	女	5	8	60

```
update student set name='小丸子' where id='ccc';select * from student;
```

结果：

```
transaction READ 成功。1 行已更新。
```

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	fff	多多	1	7	2	(null)
7	ggg	钱钱	2	7	3	60
8	hhh	喵喵	2	7	4	60

结论：允许读写

4）isolation level read committed（可幻读和重复读）

1.建立两个事务如下：

事务 1：

```
set transaction  read write;select * from student;
```

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	fff	多多	1	7	2	(null)
7	ggg	钱钱	2	7	3	60
8	hhh	喵喵	2	7	4	60

事务 2：

```
set transaction isolation level read committed;select * from student;
```

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	fff	多多	1	7	2	(null)
7	ggg	钱钱	2	7	3	60
8	hhh	喵喵	2	7	4	60

2. 在事务 1 中修改某行数据并 commit

```
update student set score=88 where id='fff';commit;
```

3. 在事务 2 中查询结果如下：

```
select * from student;
```

结果：

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	fff	多多	1	7	2	88
7	ggg	钱钱	2	7	3	60
8	hhh	喵喵	2	7	4	60

结论：事务 2 的隔离级别为 isolation level read committed，支持不可重复读

4. 在事务 1 中插入一行数据，并提交

```
insert into student values('iii','小梦','1',3,4,10);
```

5. 在事务 2 中查看：

```
select * from student;
```

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	iii	小梦	1	3	4	10
7	fff	多多	1	7	2	66
8	ggg	钱钱	2	7	3	60
9	hhh	喵喵	2	7	4	60

结论：事务 2 隔离级别为 isolation level read committed, 允许幻想读

5） isolation level serializable

1.建立两个事务如下：

事务 1：

```
set transaction  read write;select * from student;
```

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	fff	多多	1	7	2	(null)
7	ggg	钱钱	2	7	3	60
8	hhh	喵喵	2	7	4	60

事务 2：

```
set transaction isolation level serializable;select * from student;
```

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	fff	多多	1	7	2	(null)
7	ggg	钱钱	2	7	3	60
8	hhh	喵喵	2	7	4	60

2. 在事务 1 中修改某行数据并 commit

```
update student set score=88 where id='fff';commit;
```

3. 在事务 2 中查询结果如下：

```
select * from student;
```

结果：

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	fff	多多	1	7	2	(null)
7	ggg	钱钱	2	7	3	60
8	hhh	喵喵	2	7	4	60

结论：事务 2 的隔离级别为 isolation level serializable，不支持不可重复读

4. 在事务 1 中插入一行数据，并提交

```
insert into student values('iii','小梦','1',3,4,10);
```

5. 在事务 2 中查看：

```
select * from student;
```

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	fff	多多	1	7	2	(null)
7	ggg	钱钱	2	7	3	60
8	hhh	喵喵	2	7	4	60

结论：事务 2 的隔离级别为 isolation level serializable，不支持幻想读

ORACLE 锁机制

1. 概括

1）说明

锁是一种机制，多个事务同时访问一个数据库对象时，该机制可以实现对并发的控制

2） oracle 中锁的类别

1.DDL 锁： oracle 自动的施加和释放

2.DML 锁：事务开始时施加，使用 Commit 后者 Rollback 被释放、

3.内部锁： 由 oracle 自己管理以保护内部数据库结构

3）oracle 锁的粒度

1. 行级锁（TX）：阻止该行上的 DML 操作，直到 Commit 或者 Rollback
2. 表级锁（TM）：
3. 数据库级锁：eg： 将数据库锁定为只读模式 alter database open read only;

eg：将数据库设置为限制模式（导入导出数据库时使用）：alter system enable restricted session;

2.锁的模式

1)概括

锁模式	锁描述	解释	SQL操作
0	none		
1	NULL	空	Select
2	RS(Row-S)	行级共享锁，其他对象只能查询这些数据行	Select for update、Lock for update、Lock row share
3	RX(Row-X)	行级排它锁，在提交前不允许做DML操作	Insert、Update、Delete、Lock row share
4	S(Share)	共享锁	Create index、Lock share
5	SRX(S/Row-X)	共享行级排它锁	Lock share row exclusive
6	X(Exclusive)	排它锁	Alter table、Drop able、Drop index、Truncate table 、Lock exclusive

说明：

1. 数字越大，级别越高

2）eg：

```
lock table student in row share mode;
```

```
lock table student in row exclusive mode;  --用于行的修改
lock table student in share mode;   --阻止其他 DML 操作
lock table student in share row exclusive mode;   --阻止其他事务操作
lock table student in exclusive mode;  --独立访问使用
```

3.锁查看

1）概括

```
SELECT * FROM V$SESSION;--查看会话和锁的信息
SELECT * FROM V$SESSION_WAIT;--查看等待的会话信息
SELECT * FROM V$LOCK;--系统中所有锁
SELECT * FROM V$LOCKED_OBJECT;--系统中 DML 锁
```

2)eg:

```
set transaction read write;insert into student values ('jjj','小欣
','1',3,4,90);SELECT * FROM V$LOCked_object;
```

SQL | 提取的所有行：1，用时 0.023 秒

	XIDUSN	XIDSLOT	XIDSQN	OBJECT_ID	SESSION_ID	ORACLE_USERNAME	OS_USER_NAME	PROC
1	2	1	48459	119851	1180	HNZC	LJ	7792

分析：1. locked_mode:3 ,因为我们执行的是 insert， 因而是行级排他锁 row exclusive mode

死锁

1. 说明

1.当两个用户希望持有对方的资源时就会发生死锁。即两个用户互相等待对方释放资源，oracle 认定为产生了死锁，在这种情况下，将以牺牲一个用户作为代价，另一个用户继续执行，牺牲的用户的事务将回滚。

随机

2. lORA-00060 的错误并记录在数据库的日志文件 alertSID.log 中。同时在 user_dump_dest 下产生了一个跟踪文件，详细描述死锁的相关信息。

2.死锁产生条件

1. Mutual exclusion（互斥）：资源不能被共享，只能由一个进程使用。
2. Hold and wait（请求并保持）：已经得到资源的进程可以再次申请新的资源。
3. No pre-emption（不可剥夺）：已经分配的资源不能从相应的进程中被强制地剥夺。
4. Circular wait（循环等待条件）：系统中若干进程组成环路，该环路中每个进程都在等待相邻进程正占用的资源。

3.死锁模拟

student table 如下

	ID	NAME	GENDER	CLASSNO	DEPTNO	SCORE
1	aaa	小月	1	1	1	100
2	bbb	小花	1	1	1	80
3	ccc	小丸子	2	2	2	70
4	ddd	小雨	1	5	8	60
5	eee	果果	1	5	8	60
6	iii	小梦	1	3	4	10
7	fff	多多	1	7	2	66
8	ggg	钱钱	2	7	3	60
9	hhh	喵喵	2	7	4	60

1. 开两个进程（此处使用 sqldeveloper 模拟）建立两个事务

事务 1：

```
set transaction read write;
```

事务 2：

```
set transaction read write;
```

2. t1 时刻 事务 1 和事务 2 中分别执行如下语句

事务 1：

```
update student set classno=8 where id='iii';
```

事务 2：

```
update student set score=score+10 where id='jjj';
```

结果如下：

```
1 行已更新
```

1 行已更新

3. t2 时刻事务 1 和事务 2 中分别执行如下语句

事务 1：

```
update student set score=score+20 where id='jjj';
```

事务 2：

```
update student set classno=9 where id='iii';
```

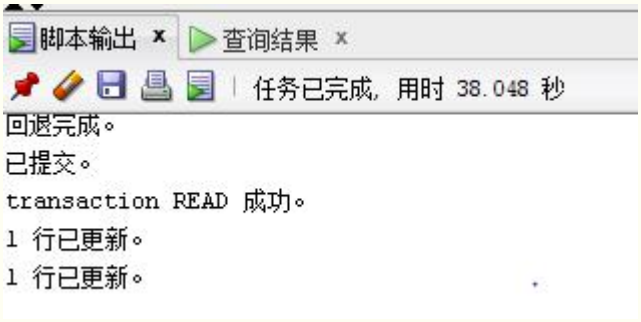
结果如下：

错误报告：

```
SQL 错误: ORA-00060: 等待资源时检测到死锁 00060. 00000 - "deadlock detected
while waiting for resource"*Cause:    Transactions deadlocked one another
while waiting for resources.*Action:   Look at the trace file to see the
transactions and resources
          involved. Retry if necessary.
```

注意：ScriptRunner 处红色的叉，出于等待状态

4. commit 事务 2，事务 1 结果如下：



注意：用时 38.048s， 事务 2commit 前等待的时间

4.解决死锁冲突

1）执行 commit 或者 rollback 结束事务

2）终止会话

还是借用 3 中死锁的例子

在等待资源时执行，查找阻塞会话

```
select sid,serial#,username from v$session where sid in (select
blocking_session from v$session);
```

执行：

```
alter system kill session '423,896';
```

结果：

在行 10 上开始执行命令时出错:alter system kill session '423,896'

错误报告：

SQL 错误：ORA-00027：无法终止当前会话 00027. 00000 - "cannot kill current session"*Cause: Attempted to use ALTER SYSTEM KILL SESSION to kill the current

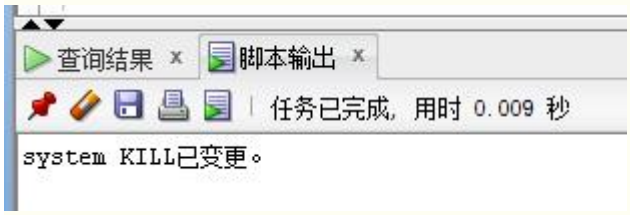
session.*Action: None.

解决方法：另起一个 session，关闭当前 session（If it is necessary to kill the current session, do so from another session.）

另起线程执行上面的语句：

```
alter system kill session '423,896';
```

结果如下：（session kill 成功，死锁解除，事务 1 更新成功）



5. 事务和死锁预防总结

1. 避免应用不运行长事务。
2. 经常提交以避免长时间锁定行。
3. 避免使用 LOCK 命令锁定表。
4. 在非高峰期间执行 DDL 操作，在非高峰期间执行长时间运行的查询或事务。

另外需注意，需监测系统中死锁，调查为什么这些锁正被保持，频率；当死锁发生通过回滚事务 rollback 或者终止会话来解决它。