九、事务处理、数据伪列

# 事务处理

之前所学的数据更新操作，相比数据查询来说，安全性较差，若数据被误改或误删，则可能造成重要数据丢失。在Oracle中提供了事物处理机制去完善数据更新机制，能有效防止数据丢失。

## 1.1 事务的概念

**（1）事务：**所谓事务(Transaction)是访问并可能更新数据库中各种数据项的一个程序执行单元(unit)(即一系列DML语句)。事务通常由高级数据库操纵语言或编程语言（如SQL，C++或Java）书写的用户程序的执行所引起，并用形如**begin transaction**和**end transaction**语句（或函数调用）来界定。事务由事务开始(**begin transaction**)和事务结束(**end transaction**)之间执行的全体操作组成。

**（2） session:**在Oracle中，事物是针对一个session进行的。在Oracle数据库中，把每一个连接到Oracle上的用户对象都称为一个session,每个session之间彼此相互独立，不会进行相互通信。每一个session都独享自己的事务控制，而不受其他session的干扰。此外，为了保证数据的同步（即一致性），如果某个session进行更新数据库表时还没有提交事务，则其它更新相同数据表的session也无法更新事务，直到前一个session提交之后的session才可以更新，但这种机制有可能造成死锁。例如：在关系数据库中，一个事务可以是一条SQL语句，一组SQL语句或整个程序。

**（3） 事务控制:** 通过将一组相关操作组合为一个要么全部成功、要么全部失败的逻辑工作单元，以简化错误恢复、提高应用程序的可靠性。

## 1.2 特性

　　事务是恢复和并发控制的基本单位。

　　事务应该具有4个属性：原子性、一致性、隔离性、持续性。这四个属性通常称为ACID特性。

### （1）原子性（atomicity）：

一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做。

例如：在银行转账系统包括很多步骤，若其中有一步错误则有可能造成巨大损失，故需要撤销之前转账的所有步骤，使系统恢复到转账前的状态。

### （2）一致性（consistency）：

事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。

### （3）隔离性（isolation）：

一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。 例如：多个用户都在利用转账系统转账，每个用户操作都不会相互干扰。

### （4）持久性（durability）：

持续性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

## 1. 3 数据库事务组成：

* + - * 0~多条DML语句
      * 1条DDL（Data Define Language）语句
      * 1条DCL（Data Control Language）语句

## 1.4 事务的开始和结束

### （1）事务开始

事务开始于第一条可执行语句；

### （2）事务结束

* 遇到commit 或rollback语句
* 遇到DDL 或DCL 语句
* 用户会话结束
* 系统崩溃

### （3）事务的提交和回滚

显式的提交和回滚操作是为了更好地保证数据的一致性。下述情况下事务会被自动提交:

* 执行一个DDL 语句
* 执行一个DCL 语句
* 正常结束会话
* 会话异常终止/系统崩溃时事务会被自动回滚

### （4）提交/回滚前数据状态

* 当前事务中数据状态的改变是可以恢复的；
* 当前事务中的DML操作结果只对当前用户（会话）可见，其它用户（会话）看不到当前事务中数据状态的改变，直到当前事务结束；
* 当前事务中DML语句所涉及到的行被锁定，其他用户（会话）不能对其进行修改操作；

### （5）事务提交后数据状态

* 数据的修改永久生效、不可再撤消
* 数据以前的状态永久性丢失、无法恢复
* 所有的用户（会话）都将看到操作后的结果
* 记录锁被释放，其它用户可对这些进行修改操作
* 事务中的保存点（ Savepoints ）被清除

### （6）事务回滚后数据状态

* 数据的修改被撤消
* 数据恢复到修改前的状态
* 记录锁被释放

## 1.5 Oracle常用事务处理命令：

|  |  |
| --- | --- |
| **ROLLBACK** | 回滚当前事务，撤销当前事务操作，使其回到原点； |
| **COMMIT** | 提交事物，使得数据库中数据真正更新，一旦提交事务，则无法回滚； |
| **SAVEPOINT** **保存点名称** | 设置保存点，设置回退标识，使用回退命令可以回退到此点； |
| **ROLLBACK TO 保存点名称** | 回退至保存点； |

1. 事务回滚示例；

1）先拷贝emp表到myemp表中；

|  |
| --- |
| CREATE TABLE myemp AS (SELECT \* FROM emp); |

2）删除表中所有数据；

|  |
| --- |
| DELETE FROM myemp; |
|  |

**注**：当前的删除，只是在数据库缓存中做了更改，并未提交的数据库；例如：如果

用另一个用户去登录数据库，查询emp表，则会发现表数据并没有变化：

|  |
| --- |
|  |
|  |

3)调用回滚命令，回到事务原点；

|  |
| --- |
| ROLLBACK; |
|  |
|  |

1. 事务提交示例；

1）先拷贝emp表到myemp表中；

|  |
| --- |
| CREATE TABLE myemp AS (SELECT \* FROM emp); |

2）删除表中原工编号为7369的员工；

|  |
| --- |
|  |

3）调用commit命令，提交事物；

|  |
| --- |
|  |

**注：**从上图可以看出，虽然在提交后进行了回滚，但数据还是无法更改；若此时，其他用户访问该表，数据也作了更改；

|  |
| --- |
|  |
|  |

1. 对于同一数据，前一事务不提交，则后面的事务无法更新，示例；

1）先拷贝emp表到myemp表中；

|  |
| --- |
| CREATE TABLE myemp AS (SELECT \* FROM emp); |

2）将表中所有员工工资涨50%；

|  |
| --- |
| UPDATE myemp SET sal=sal\*1.5; |

3）在另一控制台登录，将表中所有员工工资降20%；

|  |
| --- |
| UPDATE myemp SET sal=sal\*0.8; |
|  |
|  |
|  |

从上图可以看出，当之前的用户更新数据表myemp，且没有提交时，另一控制台上的用户进行数据更新操作时，一直处于阻塞等待，等到前一用户执行COMMIT命令后，才进行更新。

**注**：在Oracle中，所有的数据更新，都会受到事务的控制。

1. 保存点示例：

|  |
| --- |
| INSERT INTO dept VALUES(55,'ADV','BEIJING');  INSERT INTO dept VALUES(56,'SEC','SHANGHAI');  **SAVEPOINT P1;**  INSERT INTO dept VALUES(57,'ACC','TIANJIN');  ---  SELECT \* FROM dept;  **ROLLBACK TO P1;**  SELECT \* FROM dept; |

注：若此时使用ROLLBACK命令，则还是会回退到事务起始点；

## 1.6 Sql Plus的自动提交

Sql Plus中执行SQL语句时可以设置是否自动提交，缺省设置为非自动提交。

1）查看设置：

|  |
| --- |
| SHOW AUTOCOMMIT; |

状态：

* AUTOCOMMIT OFF
* AUTOCOMMIT IMMEDIATE

2）改变设置：

|  |
| --- |
| SET AUTOCOMMIT ON;  SET AUTOCOMMIT OFF; |

# 只读事务

只读事务是指只允许执行查询操作，而不允许执行任何其他DML操作的事务，使用只读事务可以确保用户只能取得某时间点的数据；假定机票代售点每天18点开始统计今天的销售情况，这时可以使用只读事务，在设置了只读事务后，尽管其它会话可能会提交新的事务，但是只读事务将不会取得最新数据的变化，从而可以保证取得特定时间点的数据信息；

设置只读事务

|  |
| --- |
| **SET TRANSACTION READ ONLY;** |

在只读事务设置之后，就不会看到设置之后的DML操作；

# 第三节 数据伪列

数据伪列是指不需要用户处理，而由Oracle自行维护的数据列（字段），在Oracle中，有两个数据伪列：ROWNUM,ROWID;

## ROWNUM

ROWNUM是行号，是在数据表中的默认字段，在创建记录时，自动生成行号。

1. 查看emp表的ROWNUM示例；

|  |
| --- |
| SELECT ROWNUM, ename, empno FROM emp; |
|  |

**注**：ROWNUM并不是永久固定的，而是每次动态重新生成的，例如：只查询部门30的员工的行号；

|  |
| --- |
| SELECT ROWNUM, ename,empno FROM emp WHERE deptno=30; |
|  |

使用ROWNUM可以更加方便地查找数据；

1. 查找emp表中的前5条记录；

|  |
| --- |
| SELECT ROWNUM,empno,ename,job,sal,comm,deptno FROM emp WHERE ROWNUM<=5; |
|  |

1. 查询emp表中6~10条记录；

若按照常规做法，只要在WHERE子句中限定ROWNUM的范围即可，但这样做却查

不出结果；

|  |
| --- |
| SELECT ROWNUM,empno,ename,job,sal,comm,deptno FROM emp WHERE ROWNUM<=10 AND ROWNUM>=6; |
| SELECT ROWNUM,empno,ename,job,sal,comm,deptno FROM emp WHERE ROWNUM BETWEEN 6 AND 10; |
|  |

造成以上现象的原因是，ROWNUM并不是数据表的真实字段，要想实现以上功能，需要结合子查询先查出前10条记录再查出后5条记录；

|  |
| --- |
| SELECT \* FROM (  SELECT ROWNUM r,empno,ename,job,sal,comm,deptno FROM emp  WHERE ROWNUM<=10) tem  WHERE tem.r>=6; |
|  |

### 补：分页显示的查询语句思路

第一页：

当前所在页（currentPage）:1

每页显示记录长度（pageSize）：5

|  |
| --- |
| SELECT \* FROM (  SELECT ROWNUM r,empno,ename,job,sal,comm,deptno FROM emp  WHERE ROWNUM<=5) tem  WHERE tem.r>=1; |

第二页：

当前所在页（currentPage）:2

每页显示记录长度（pageSize）：5

|  |
| --- |
| SELECT \* FROM (  SELECT ROWNUM r,empno,ename,job,sal,comm,deptno FROM emp  WHERE ROWNUM<=10) tem  WHERE tem.r>=6; |

第三页：

当前所在页（currentPage）:3

每页显示记录长度（pageSize）：5

|  |
| --- |
| SELECT \* FROM (  SELECT ROWNUM r,empno,ename,job,sal,comm,deptno FROM emp  WHERE ROWNUM<=15) tem  WHERE tem.r>=11; |

第n页：

当前所在页（currentPage）:cp

每页显示记录长度（pageSize）：ps

|  |
| --- |
| SELECT \* FROM (  SELECT ROWNUM r,empno,ename,job,sal,comm,deptno FROM emp  WHERE ROWNUM<=cp\*ps) tem  WHERE tem.r>=(cp-1)\*ps+1; |

## ROWID

ROWID表示每条记录存储物理地址的编号，每一条记录的ROWID都不会重复，即使是

记录的内容相同；例如：查询dept表中记录的ROWID;

|  |
| --- |
| SELECT ROWID ,deptno,dname FROM dept; |
|  |

ROWID的格式：AAAMgxAAEAAAAAQAAA

|  |  |
| --- | --- |
| 数据对象号 | AAAMgx |
| 相对文件号 | AAE |
| 数据块号 | AAAAAQ |
| 数据行号 | AAA |

1. 利用ROWID删除dept表中的重复记录；
2. 先向dept表中增加几个有字段重复的记录

|  |
| --- |
| INSERT INTO dept(deptno,dname,loc) VALUES(50,'TEST','BEIJING');  INSERT INTO dept(deptno,dname,loc) VALUES(51,'TEST','BEIJING');  INSERT INTO dept(deptno,dname,loc) VALUES(52,'TEST','BEIJING');  INSERT INTO dept(deptno,dname,loc) VALUES(53,'DEVELOP','CHANGSHA');  INSERT INTO dept(deptno,dname,loc) VALUES(54,'DEVELOP','CHANGSHA');  INSERT INTO dept(deptno,dname,loc) VALUES(55,'DEVELOP','CHANGSHA');  INSERT INTO dept(deptno,dname,loc) VALUES(56,'DESIGN','HANGZHOU');  INSERT INTO dept(deptno,dname,loc) VALUES(57,'DESIGN','HANGZHOU');  INSERT INTO dept(deptno,dname,loc) VALUES(58,'DESIGN','HANGZHOU');  INSERT INTO dept(deptno,dname,loc) VALUES(59,'DESIGN','HANGZHOU');  COMMIT; |
|  |

1. 通过分组查询先找到重复数据,然后根据判定记录条数是否大于1来不重复的显示记录，保持最早增加的记录内容；

|  |
| --- |
| SELECT dname,loc,MIN(ROWID) FROM dept  GROUP BY dname,loc  HAVING COUNT(deptno)>1; |
|  |

1. 观察重复记录的ROWID，发现先插入的记录，其ROWID较小，故可以先找出最小的ROWID，然后利用子查询删除不属于最小ROWID的全部元素；

|  |
| --- |
| DELETE FROM dept  WHERE ROWID NOT IN(  SELECT MIN(ROWID)  FROM dept  GROUP BY dname,loc); |
|  |