第3章 数据操作

3.1 数据库操作语句

本章将要学习的操作命令总结如表3-1所示。

语　句	描　　述
INSERT	插入新行
UPDATE	修改(更新)已经存在的行
DELETE	删除表中已经存在的行

3.1.1 插入数据

可以使用INSERT命令，向已经存在的表插入数据，语法格式如下：

INSERT INTO 表名 [(字段列表)] {VALUES(表达式1, 表达式2,...)|QUERY语句}；

1．数据插入基本语法

最常见的插入操作可使用以下的语法(该形式一次只能插入一行数据)：

INSERT INTO 表名[(字段列表)] VALUES (表达式列表);

插入字段的值的类型要和字段的类型一一对应。字符串类型的字段值必须用单引号括起来，例如：'CLERK'。字符串类型的字段值超过定义的长度会出错, 最好在插入前进行长度校验。

字段列表如果省略则代表全部字段。

【训练1】 表的部分字段插入练习。

步骤1：将新雇员插入到emp表：

```
INSERT INTO emp(empno,ename,job)
VALUES (1000, '小李', 'CLERK');
```

执行结果为：

```
已创建1行。
```

步骤2：显示插入结果

SELECT * FROM emp WHERE empno=1000;

执行结果：

```
     EMPNO ENAME      JOB       MGR HIREDATE       SAL
COMM     DEPTNO
---------- ---------- --------- ------- ----------------- -----
----- ---------- ----------
      1000 小李       CLERK
```

说明：INSERT 语句的emp表名后的括号中为要插入的字段列表，VALUES后的括号中为要插入的字段值列表。要插入的字段是雇员编号empno、名称ename和职务job。其他没有插入的字段，系统会填写为表的默认值。如果在表的创建时没有说明默认值，则将插入NULL值。在本训练中，其他没有插入的字段值均为空值NULL。

日期类型的字段值也要用单引号括起来，如'10-1月-03'。日期型的数据默认格式为DD-MON-YY，默认的世纪为当前的世纪，默认的时间为午夜12点。如果指定的世纪不是本世纪或时间不是午夜12点，则必须使用TO_DATE系统函数对字符串进行转换。

【训练2】 时间字段的插入练习。

步骤1：将新雇员插入到emp表：

```
INSERT INTO emp(empno,ename,job,hiredate)
VALUES (1001, '小马', 'CLERK', '10-1月-03');
```

执行结果为：

```
已创建 1 行。
```

说明：在本训练中，插入的雇员雇佣时间为2003年1月10日。

注意：时间的默认格式为DD-MON-YY。

如果要插入表的全部字段，则表名后的字段列表可以省略，如下面的训练。

【训练3】 表的全部字段的插入练习。

执行以下的查询：

```
INSERT INTO dept VALUES (50, '培训部','深圳');
```

执行结果：

```
已创建 1 行。
```

说明：此种方式省略了字段名列表，要注意插入数据的顺序必须与表的字段默认顺序保持一致。如果不知道表的字段默认顺序，可以用DESCRIBE命令查看。

【训练4】 插入空值练习。

执行以下的查询：

INSERT INTO emp(empno,ename,job,sal) VALUES(1005,'杨华', 'CLERK',null);

执行结果：

已创建 1 行。

说明：以上训练虽然指定了插入字段sal，但在插入的数值位置指定了NULL值，所以sal的插入值还是NULL。

【练习1】向雇员表插入全部字段的一条记录。

2．复制数据

另一种插入数据(相当于复制)方法的语法格式是：

INSERT INTO 表名(字段列表) SELECT(字段名1, 字段名2, ...) FROM 另外的表名;

该形式一次可以插入多行数据。

【训练5】 通过其他表插入数据的练习。

步骤1：创建一个新表manager：

CREATE TABLE manager AS SELECT empno,ename,sal FROM emp WHERE job='MANAGER';

执行结果：

表已创建。

步骤2：从emp表拷贝数据到manager：

```
INSERT INTO manager
SELECT          empno, ename, sal
FROM   emp
WHEREjob = 'CLERK';
```

执行结果：

已创建 1 行。

步骤3：查询结果：

```
SELECT    * FROM MANAGER;
```

结果为：

```
 EMPNO          ENAME          SAL
---------------- -------------- ------------------
 7566     JONES            2975
 7698     BLAKE            2850
 7782     CLARK            2450
          1000      小李
```

说明：CREATE命令用来根据已经存在的表创建新表。步骤1根据emp表创建一个新表manager，该表只有3个字段empno,ename和sal，创建的同时将emp表中职务为manager的雇员复制到其中。步骤2从emp表中把职务为clerk的雇员插入到manager表中。

3．使用序列

使用INSERT语句时，可以通过序列来填写某些数值型或字符型的列。序列是一个要预先定义的有序的数值序列，应该先建立一个序列，然后在插入语句中使用，序列将在以后章节中介绍。

【训练6】 插入数据中使用序列的练习。

步骤1：创建从2000起始，增量为1 的序列abc：

CREATE SEQUENCE abc INCREMENT BY 1 START WITH 2000

MAXVALUE 99999 CYCLE NOCACHE;

执行结果：

序列已创建。

步骤2：在INSERT 语句使用序列，序列的名称为abc：

```
INSERT INTO manager VALUES(abc.nextval,'小王',2500);
```

执行结果：

```
已创建 1 行。
```

```
INSERT INTO manager VALUES(abc.nextval,'小赵',2800);
```

执行结果：

```
已创建 1 行。
```

步骤3：使用SELECT语句观察结果：

SELECT empno,ename,sal FROM emp;

执行结果：

```
   EMPNO ENAME          SAL
---------- ---------- ----------
      7566 JONES            2975
      7698 BLAKE            2850
      7782 CLARK            2450
      2000 小王             2500
      2001 小赵        2800
```

说明：步骤1创建序列，步骤2在插入时使用序列来填充雇员编号，使用abc.nextval可获得序列中的下一个值。

后边两个记录的雇员编号来自序列，并且是递增的。

3.1.2 修改数据

修改数据的语句UPDATE对表中指定字段的数据进行修改，一般需要通过添加WHERE条件来限定要进行修改的行，如果不添加WHERE条件，将对所有的行进行修改。

(1) 修改数据的语句UPDATE的基本语法如下：

UPDATE 表名 SET 字段名1=表达式1, 字段名2=表达式2, ... WHERE 条件;

【训练1】 修改小李(编号为1000)的工资为3000。

执行以下的查询：

```
UPDATE      emp
SET     sal = 3000
WHERE       empno = 1000;
```

执行结果：

```
已更新 1 行。
```

说明：该操作将编号为1000的雇员的工资改为3000。

【训练2】 将小李(编号为1000)的雇佣日期改成当前系统日期，部门编号改为50。

执行以下的查询：

```
UPDATE          emp
SET     hiredate=sysdate, deptno=50
WHERE           empno = 1000;
```

执行结果：

已更新 1 行。

说明：该操作同时修改编号为1000的雇员的雇佣日期和部门编号两个字段的值。

如果修改的值没有赋值或定义，将把原来字段的内容清为NULL。若修改值的长度超过定义的长度，则会出错。

注意：本例中不能省略WHERE条件，否则将会修改表的所有行。

【练习1】将SCOTT的职务改为MANAGER，工资改为4000。

【训练3】 为所有雇员增加100元工资。

执行以下的查询：

```
UPDATE  emp
SET  sal =sal+100;
```

执行结果：

已更新18行。

说明：若没有WHERE条件，将修改表的所有行。sal=sal+100的含义是：对于每条记录，取出原来sal字段的工资，加100后再赋给sal字段。

【练习2】将emp表的部门10的雇员工资增加10%。

(2) UPDATE语句的另外一种用法：

UPDATE 表名 SET(字段名1, 字段名2, ...)=SELECT (字段名1, 字段名2, ...) FROM 另外的表名WHERE条件;

【训练4】 根据其他表修改数据。

执行以下的查询：

```
UPDATE  manager
SET (ename, sal) =(SELECT ename,sal FROM emp WHERE   empno = 7788)
WHERE   empno = 1000;
```

执行结果：

已更新 1 行。

说明：该操作将manager表中编号为1000的记录的雇员名字和工资修改成为emp表的编号为7788的雇员的名字和工资。

3.1.3 删除数据

删除数据的基本语法如下：

DELETE FROM表名 WHERE 条件;

要从表中删除满足条件的记录，WHERE条件一般不能省略，如果省略就会删除表的全部数据。

【训练1】 删除雇员编号为1000的新插入的雇员。

步骤1：删除编号为1000的雇员：

```
DELETE FROM emp WHERE empno=1000;
```

结果为：

```
已删除 1 行。
```

步骤2：显示删除结果：

```
SELECT * FROM emp WHERE empno=1000;
```

结果为：

```
未选定行。
```

说明：本例删除雇员编号为1000的雇员，它在WHERE中指定删除的记录。删除记录并不能释放Oracle中被占用的数据块表空间，它只是把那些被删除的数据块标成unused。

如果确实要删除一个大表里的全部记录，可以用TRUNCATE 命令，它可以释放占用的数据块表空间，语法为：

TRUNCATE TABLE 表名;

【训练2】 彻底删除manager表的内容。

执行以下的命令：

```
TRUNCATE TABLE manager;
```

执行结果：

```
表已截掉。
```

说明：此命令和不带WHERE条件的DELETE语句功能类似，不同的是，DELETE命令进行的删除可以撤销，但此命令进行的删除不可撤销。

注意：TRUNCATE TABLE命令用来删除表的全部数据而不是删除表，表依旧存在。

3.2 数据库事务

3.2.1 数据库事务的概念

事务是由相关操作构成的一个完整的操作单元。两次连续成功的COMMIT或ROLLBACK之间的操作，称为一个事务。在一个事务内，数据的修改一起提交或撤销，如果发生故障或系统错误，整个事务也会自动撤销。

比如，我们去银行转账，操作可以分为下面两个环节：

(1) 从第一个账户划出款项。

(2) 将款项存入第二个账户。

在这个过程中，两个环节是关联的。第一个账户划出款项必须保证正确的存入第二个账户，如果第二个环节没有完成，整个的过程都应该取消，否则就会发生丢失款项的问题。整个交易过程，可以看作是一个事物，成功则全部成功，失败则需要全部撤消，这样可以避免当操作的中间环节出现问题时，产生数据不一致的问题。

数据库事务是一个逻辑上的划分，有的时候并不是很明显，它可以是一个操作步骤，也可以是多个操作步骤。

我们可以这样理解数据库事物：对数据库所做的一系列修改，在修改过程中，暂时不写入数据库，而是缓存起来，用户在自己的终端可以预览变化，直到全部修改完成，并经过检查确认无误后，一次性提交并写入数据库，在提交之前，必要的话所做的修改都可以取消。提交之后，就不能撤销，提交成功后其他用户才可以通过查询浏览数据的变化。

以事务的方式对数据库进行访问，有如下的优点：

* 把逻辑相关的操作分成了一个组。
* 在数据永久改变前，可以预览数据变化。
* 能够保证数据的读一致性。

3.2.2 数据库事务的应用

数据库事务处理可分为隐式和显式两种。显式事务操作通过命令实现，隐式事务由系统自动完成提交或撤销(回退)工作，无需用户的干预。

隐式提交的情况包括：当用户正常退出SQL*Plus或执行CREATE、DROP、GRANT、REVOKE等命令时会发生事务的自动提交。

还有一种情况，如果把系统的环境变量AUTOCOMMIT设置为ON(默认状态为OFF)，则每当执行一条INSERT、DELETE或UPDATE命令对数据进行修改后，就会马上自动提交。设置命令格式如下：

SET AUTOCOMMIT ON/OFF

隐式回退的情况包括：当异常结束SQL*Plus或系统故障发生时，会发生事务的自动回退。

显式事务处理的数据库事务操作语句有3条，如表3-2所示。

表3-2 事务控制语句

语　句	描　　述
COMMIT	数据库事务提交，将变化写入数据库
ROLLBACK	数据库事务回退，撤销对数据的修改
SAVEPOINT	创建保存点，用于事务的阶段回退

COMMIT操作把多个步骤对数据库的修改，一次性地永久写入数据库，代表数据库事务的成功执行。ROLLBACK操作在发生问题时，把对数据库已经作出的修改撤消，回退到修改前的状态。在操作过程中，一旦发生问题，如果还没有提交操作，则随时可以使用ROLLBACK来撤消前面的操作。SAVEPOINT则用于在事务中间建立一些保存点，ROLLBACK可以使操作回退到这些点上边，而不必撤销全部的操作。一旦COMMIT完成，就不能用ROLLBACK来取消已经提交的操作。一旦ROLLBACK完成，被撤消的操作要重做，必须重新执行相关操作语句。

如何开始一个新的事务呢？一般情况下，开始一个会话(即连接数据库)，执行第一条SQL语句将开始一个新的事务，或执行COMMIT提交或ROLLBACK撤销事务，也标志新的事务的开始。另外，执行DDL(如CREATE)或DCL命令也将自动提交前一个事务而开始一个新的事务。

数据在修改的时候会对记录进行锁定，其他会话不能对锁定的记录进行修改或加锁，只有当前会话提交或撤销后，记录的锁定才会释放。详细内容见下一节。

我们通过以下的训练来为雇员SCOTT增加工资，SCOTT的雇员号为7788。

【训练1】 学习使用COMMIT和ROLLBACK。

步骤1：执行以下命令，提交尚未提交的操作：

```
COMMIT;
```

执行结果：

```
提交完成。
```

显示SCOTT的现有工资：

```
SELECT ename,sal FROM emp WHERE empno=7788;
```

执行结果：

```
ENAME          SAL
---------- ----------
SCOTT          3000
```

步骤2：修改雇员SCOTT的工资：

```
UPDATE emp SET sal=sal+100 WHERE empno=7788;
```

执行结果：

```
已更新1行。
```

显示修改后的SCOTT的工资：

```
SELECT ename,sal FROM emp WHERE empno=7788;
```

执行结果：

```
ENAME           SAL
--------- ----------------------
SCOTT           3100
```

步骤3：假定修改操作后发现增加的工资应该为1000而不是100，为了取消刚做的操作，可以执行以下命令：

ROLLBACK;

执行结果：

回退已完成。

显示回退后SCOTT的工资恢复为3000：

SELECT ename,sal FROM emp WHERE empno=7788;

执行结果：

ENAME	SAL
SCOTT	3000

步骤4：重新修改雇员SCOTT的工资，工资在原有基础上增加1000：

```
UPDATE emp SET sal=sal+1000 WHERE empno=7788;
```

执行结果：

```
已更新 1 行。
```

显示修改后SCOTT的工资：

```
SELECT ename,sal FROM emp WHERE empno=7788;
```

执行结果：

```
ENAME          SAL
---------- ----------
SCOTT          4000
```

步骤5：经查看修改结果正确，提交所做的修改：

COMMIT;

执行结果：

提交完成。

说明：在执行COMMIT后，工资的修改被永久写入数据库。本训练的第1步，先使用COMMIT命令提交原来的操作，同时标志一个新的事务的开始。

注意：在事务执行过程中，随时可以预览数据的变化。

对于比较大的事务，可以使用SAVEPOINT命令在事务中间划分一些断点，用来作为回退点。

【训练2】 学习使用SAVEPOINT命令。

步骤1：插入一个雇员：

```
INSERT INTO emp(empno, ename, job)
VALUES (3000, '小马','STUDENT');
```

执行结果：

已创建 1 行。

步骤2：插入保存点，检查点的名称为PA：

```
SAVEPOINT pa;
```

执行结果：

保存点已创建。

步骤3：插入另一个雇员：

```
INSERT INTO emp(empno, ename, job)
VALUES (3001, '小黄','STUDENT');
```

执行结果：

已创建 1 行。

步骤4：回退到保存点PA，则后插入的小黄被取消，而小马仍然保留。

```
ROLLBACK TO    pa;
```

执行结果：

回退已完成。

步骤5: 提交所做的修改：

COMMIT;

执行结果：

提交完成。

说明：第4步的回退，将回退到保存点PA，即第3步被撤销。所以最后的COMMIT只提交了对小马的插入。请自行检查插入的雇员。

【练习1】对emp表进行修改，然后退出SQL*Plus，重新启动SQL*Plus，检查所做的修改是否生效。

在Oracle数据库中，有一个叫回滚段的特殊的存储区域。在提交一个事物之前，如果用户进行了数据的修改，在所谓的回滚段中将保存变化前的数据。有了回滚段才能在必要时使用ROLLBACK命令或自动地进行数据撤销。在提交事物之前，用户自己可以看到修改的数据，但因为修改还没有最终提交，其他用户看到的应该是原来的数据，也就是回滚段中的数据，这时用户自己看到的数据和其他用户看到的数据是不同的，只有提交发生后，变化的数据才会被写入数据库，此时用户自己看到的数据和其他用户看到的数据才是一致的，这叫做数据的读一致性。

【训练3】 观察数据的读一致性。

步骤1：显示刚插入的雇员小马:

```
SELECT empno,ename FROM emp WHERE empno=3000;
```

执行结果：

```
EMPNO ENAME
---------- ----------
      3000 小马
```

步骤2：删除雇员小马:

```
DELETE FROM emp WHERE empno=3000;
```

执行结果：

```
已删除 1 行。
```

步骤3：再次显示该雇员，显示结果为该雇员不存在：

SELECT empno,ename FROM emp WHERE empno=3000;

执行结果：

未选定行

步骤4：另外启动第2个SQL*Plus，并以SCOTT身份连接。执行以下命令，结果为该记录依旧存在。

SELECT empno,ename FROM emp WHERE empno=3000;

执行结果：

EMPNO ENAME

---------- ----------

3000 小马

步骤5：在第1个SQL*Plus中提交删除：

COMMIT;

执行结果：

提交完成。

步骤6：在第2个SQL*Plus中再次显示该雇员，显示结果与步骤3的结果一致:

SELECT empno,ename FROM emp WHERE empno=3000;

执行结果：

未选定行

说明：在以上训练中，当第1个SQL*Plus会话删除小马后，第2个SQL*Plus会话仍然可以看到该雇员，直到第1个SQL*Plus会话提交该删除操作后，两个会话看到的才是一致的数据。

3.3 表的锁定

3.3.1 锁的概念

锁出现在数据共享的场合，用来保证数据的一致性。当多个会话同时修改一个表时，需要对数据进行相应的锁定。

锁有“只读锁”、“排它锁”，“共享排它锁”等多种类型，而且每种类型又有“行级锁”(一次锁住一条记录)，“页级锁”(一次锁住一页，即数据库中存储记录的最小可分配单元)，“表级锁”(锁住整个表)。

若为“行级排它锁”，则除被锁住的行外，该表中其他行均可被其他的用户进行修改(Update)或删除(delete)。若为“表级排它锁”，则所有其他用户只能对该表进行查询(select)操作，而无法对其中的任何记录进行修改或删除。当程序对所做的修改进行提交(commit)或回滚(rollback)后，锁住的资源便会得到释放，从而允许其他用户进行操作。

有时，由于程序的原因，锁住资源后长时间未对其工作进行提交；或是由于用户的原因，调出需要修改的数据后，未及时修改并提交，而是放置于一旁；或是由于客户服务器方式中客户端出现“死机”，而服务器端却并未检测到，从而造成锁定的资源未被及时释放，影响到其他用户的操作。

如果两个事务，分别锁定一部分数据，而都在等待对方释放锁才能完成事务操作，这种情况下就会发生死锁。

3.3.2 隐式锁和显式锁

在Oracle数据库中，修改数据操作时需要一个隐式的独占锁，以锁定修改的行，直到修改被提交或撤销为止。如果一个会话锁定了数据，那么第二个会话要想对数据进行修改，只能等到第一个会话对修改使用COMMIT命令进行提交或使用ROLLBACK命令进行回滚撤销后，才开始执行。因此应养成一个良好的习惯：执行修改操作后，要尽早地提交或撤销，以免影响其他会话对数据的修改。

【训练1】 对emp表的SCOTT雇员记录进行修改，测试隐式锁。

步骤1：启动第一个SQL*Plus，以SCOTT账户登录数据库(第一个会话)，修改SCOTT记录，隐式加锁。

UPDATE emp SET sal=3500 where empno=7788;

执行结果：

已更新 1 行。

步骤2：启动第二个SQL*Plus，以SCOTT账户登录数据库(第二个会话)，进行记录修改操作。

UPDATE emp SET sal=4000 where empno=7788;

执行结果，没有任何输出(处于等待解锁状态)。

步骤3：对第一个会话进行解锁操作：

COMMIT;

步骤4：查看第二个会话，此时有输出结果：

已更新 1 行。

步骤5：提交第二个会话，防止长时间锁定。

说明：两个会话对同一表的同一条记录进行修改。步骤1修改SCOTT工资为3500，没有提交或回滚之前，SCOTT记录处于加锁状态。步骤2的第二个会话对SCOTT进行修改处于等待状态。

步骤3解锁之后(即第一个会话对SCOTT的修改已经完成)，第二个会话挂起的修改此时可以执行。最后结果为第二个会话的修改结果，即SCOTT的工资修改为4000。读者可以使用查询语句检查。

以上是隐式加锁，用户也可以使用如下两种方式主动锁定行或表，防止其他会话对数据的修改。表3-3是对行或表进行锁定的语句。

表3-3 表的显式锁定操作语句

语　句	描　述
SELECT FOR UPDATE	锁定表行，防止其他会话对行的修改
LOCK TABLE	锁定表，防止其他会话对表的修改

3.3.3 锁定行

【训练1】 对emp表的部门10的雇员记录加显式锁，并测试。

步骤1：对部门10加显式锁：

SELECT empno,ename,job,sal FROM emp WHERE deptno=10 FOR UPDATE;

结果为：

```
EMPNO ENAME      JOB              SAL
---------- ---------- ---------- ----------
7782  CLARK      MANAGER     2450
7839  KING       PRESIDENT   5000
7934  MILLER     CLERK       1300
```

步骤2：启动第二个SQL*Plus(第二个会话)，以SCOTT账户登录数据库，对部门10的雇员CLARK进行修改操作。

UPDATE emp SET sal=sal+100 where empno=7782;

执行结果：

没有任何输出(处于等待解锁状态)。

步骤3：在第一个会话进行解锁操作：

COMMIT;

步骤4：查看第二个会话，有输出结果：

已更新 1 行。

说明：步骤1对选定的部门10的雇员加锁，之后其他会话不能对部门10的雇员数据进行修改或删除。如果此时要进行修改或删除，则会处于等待状态。使用COMMIT语句进行解锁之后，如果有挂起的修改或删除操作，则等待的操作此时可以执行。

3.3.4 锁定表

LOCK语句用于对整张表进行锁定。语法如下：

LOCK TABLE 表名 IN {SHARE|EXCLUSIVE} MODE

对表的锁定可以是共享(SHARE)或独占(EXCLUSIVE)模式。共享模式下，其他会话可以加共享锁，但不能加独占锁。在独占模式下，其他会话不能加共享或独占锁。

【训练1】 对emp表添加独占锁。

步骤1：对emp表加独占锁：

LOCK TABLE emp IN EXCLUSIVE MODE;

结果为：

表已锁定。

步骤2：对表进行解锁操作：

COMMIT;

说明：当使用LOCK语句显式锁定一张表时，死锁的概率就会增加。同样地，使用COMMIT或ROLLBACK命令可以释放锁。

注意：必须没有其他会话对该表的任何记录加锁，此操作才能成功。

【练习1】通过两个会话以共享方式锁定dept表，然后分别释放。

3.4 阶段训练

【训练1】 以数据库事务方式将SCOTT从emp表转入manager表，再将SCOTT的工资改成和emp表的KING的工资一样。

步骤1：复制emp表的SCOTT到manager表：

INSERT INTO manager SELECT empno,ename,sal FROM emp WHERE empno=7788;

执行结果：

已创建 1 行。

步骤2：删除emp表的SCOTT：

DELETE FROM emp WHERE empno=7788;

执行结果：

已删除 1 行。

步骤3：修改SCOTT的工资：

UPDATE manager SET sal=(SELECT sal FROM emp WHERE empno=7839) WHERE empno=7788;

执行结果：

已更新 1 行。

步骤4：提交：

COMMIT;

执行结果：

提交完成。

步骤5：查询：

SELECT * FROM manager WHERE empno=7788;

EMPNO ENAME SAL

---------- ---------- ----------

7788 SCOTT 5100

执行结果：

已选择 1 行。

SELECT * FROM emp WHERE empno=7788;

执行结果：

未选定行

说明：该训练中，SCOTT的雇员编号为7788，KING的雇员编号为7839。步骤1先将SCOTT复制到manager表；步骤2删除原来的SCOTT记录；步骤3修改SCOTT的工资为KING的工资；步骤4进行一次性提交；通过步骤5的查询可以看到SCOTT已经移动到了manager 表，其工资修改为5100。

3.5 练习

1. 参照本章的emp表，以下正确的插入语句是：

A. INSERT INTO emp VALUES (1000, '小李', 1500);

B. INSERT INTO emp(ename,empno,sal) VALUES (1000, '小李', 1500);

C. INSERT INTO emp(empno,ename,job) VALUES ('小李',1000,1500);

D. INSERT INTO emp(ename,empno,sal) VALUES ('小李',1000,1500);

2．删除emp表的全部数据，但不提交，以下正确的语句是：

A. DELETE * FROM EMP

B. DELETE FROM EMP

C. TRUNCATE TABLE EMP

D. DELETE TABLE EMP

3．以下不需要进行提交或回退的操作是：

A. 显式的锁定一张表

B. 使用UPDATE修改表的记录

C. 使用DELETE删除表的记录

D. 使用SELECT查询表的记录

4. 当一个用户修改了表的数据，那么

A. 第二个用户立即能够看到数据的变化

B. 第二个用户必须执行ROLLBACK命令后才能看到数据的变化

C. 第二个用户必须执行COMMIT命令后才能看到数据的变化

D. 第二个用户因为会话不同，暂时不能看到数据的变化

5. 对于ROLLBACK命令，以下准确的说法是：

A. 撤销刚刚进行的数据修改操作

B. 撤销本次登录以来所有的数据修改

C. 撤销到上次执行提交或回退操作的点

D. 撤销上一个COMMIT命令