

知识点列表

编号	名称	描述	级别
1	DML 操作	掌握 DML 操作的语法	***
2	事务概念	了解事务及事务相关的命令	**
3	DDL 操作	掌握 DDL 操作的语法	**
4	DCL 操作	掌握 DCL 操作的语法	*
5	执行批处理脚本文件	在 sqlPlus 中执行批处理脚本文件	*

注： **"理解级别 ***"掌握级别 ****"应用级别

目录

1. 知识点回顾.....	4
1.1. 内连接.....	4
1.1.1. 等值连接(on 后面的条件是 “=” 等于).....	4
1.1.2. 非等值连接(on 后面的条件不是等值操作).....	4
1.2. 外连接.....	6
1.3. SQL 语句的种类.....	6
1.4. Emp 表、Dept 表、Salgrade 表数据结构.....	6
2. DML 操作**.....	8
2.1. insert(插入数据) **.....	8
2.1.1. 使用方法：插入全部列值时，不用写列名.....	8
2.1.2. 使用方法：写列名.....	10
2.1.3. 复制表 **.....	10
2.1.4. rownum 关键字 **.....	12
2.2. update(更新数据) **.....	12
2.3. delete(删除数据) **.....	13
2.3.1. 删除表中的重复数据.....	14
2.3.2. rename 关键字.....	15
2.3.3. rowid 关键字 删除重复数据(较简单，性能高).....	15
3. Transaction(事务) **.....	18
3.1. 事务的开始和终止(事务边界).....	18
3.2. 事务中的数据状态.....	19

3.2.1. 如果多个会话操作同一张表的数据.....	19
3.2.2. Savepoint.....	23
4. DDL 操作.....	25
4.1. create(建表) **	25
4.2. drop(删除结构和全部的表数据) **	25
4.2.1. 和表相关的数据字典 *	26
4.3. truncate(截取 , 截断) *	28
4.4. alter(修改结构) *	28
4.4.1. add 关键字.....	28
4.4.2. rename 关键字.....	29
4.4.3. modify 关键字.....	29
4.4.4. drop column.....	29
4.5. 执行数据库脚本(script)文件 *.sql **	29
5. 数据控制语言 DCL	31

1. 知识点回顾

1.1. 内连接

内连接包括等值连接和非等值连接。内连接返回满足条件的记录。

内连接语法：t1 join t2 on 条件

1.1.1. 等值连接(on 后面的条件是 “=” 等于)

形如 “emp t1 join dept t2 on t1.deptno = t2.deptno” 的形式为等值连接

1.1.1.1. 自连接(等值连接的一种形式)

自连接是等值连接的一种。表中的列外键关联自己表的主键列。

自连接的数据来源只有一个表，通过使用表的别名虚拟成两个表的方式实现。

形如 “emp t1 join emp t2 on t1.mgr = t2.empno” 的形式为自连接

1.1.2. 非等值连接(on 后面的条件不是等值操作)

非等值连接指在多个表间使用非等号连接，查询在多个表间有非等值关系的数据，非等值连接操作符包括：>、<、<>、>=、<=以及 Between And、like、in 等。

【案例 1】计算员工的薪水等级？

Emp_xxx 表			Salgrade_xxx 表		
编号 empno	姓名 ename	薪水 salary	级别 grade	最低薪 lowsal	最高薪 hisal
1001	zhangwj	10000	1	10001	99999
1002	liucs	8000	2	8001	10000
1003	liyi	9000	3	6001	8000
1004	guofr	5000	4	4001	6000
			5	1	4000

- **步骤 1：准备数据(1 建表 ;2 插入数据 ;3 commit)**

```
SQL> create table salgrade_xxx(
    grade number(2) ,
    lowsal number(7, 2) ,
    hisal number(7, 2)
);

SQL> insert into salgrade_ning values(1, 10001, 99999);
SQL> insert into salgrade_ning values(2, 8001, 10000);
SQL> insert into salgrade_ning values(3, 6001, 8000);
SQL> insert into salgrade_ning values(4, 4001, 6000);
SQL> insert into salgrade_ning values(5, 1, 4000);
SQL> commit;
```

- **步骤 2：计算员工的薪水等级**

```
SQL> select e.empno, e.ename, e.salary, s.grade
    from emp_xxx e join salgrade_xxx s
    on e.salary between s.lowsal and s.hisal;
```

运行 SQL 命令行			
SQL> select e.empno, e.ename, e.salary, s.grade			
2 from emp_xxx e join salgrade_xxx s			
3 on e.salary between s.lowsal and s.hisal;			
EMPNO	ENAME	SALARY	GRADE
1005	张三丰	15000	1
1001	张无忌	10000	2
1003	李翊	9000	2
1002	刘苍松	8000	3
1004	郭芙蓉	5000	4
1006	燕小六	5000	4
1008	黄蓉	5000	4
1010	郭靖	4500	4
1007	陆无双	3000	5
1009	韦小宝	4000	5

已选择10行。

1.2. 外连接

有些时候需要返回那些不满足条件的记录，这种情况无法用内连接查询。

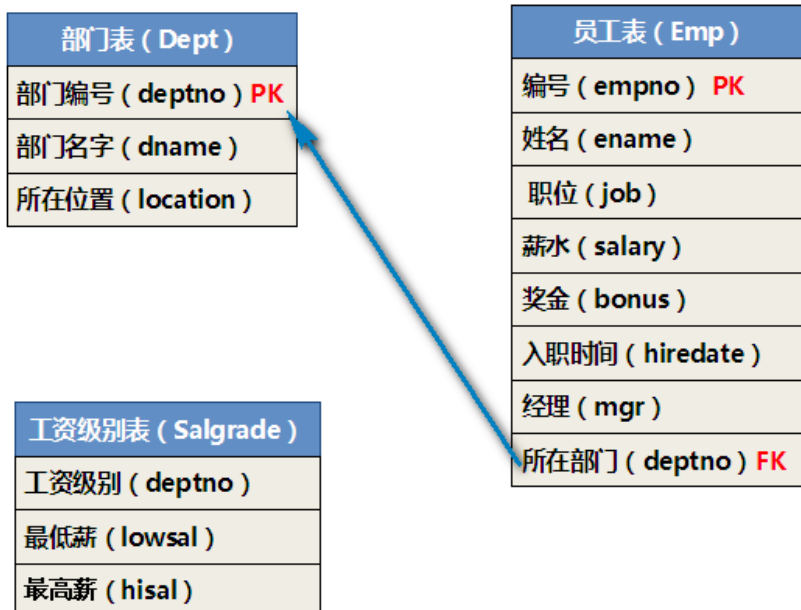
外连接分 3 种：

- ✓ t1 表 **left** outer join t2 表 on 条件
- ✓ t1 表 **right** outer join t2 表 on 条件
- ✓ t1 表 **full** outer join t2 表 on 条件

1.3. SQL 语句的种类

- 1) 数据查询语言 DQL : select
- 2) 数据定义语言 DDL : create / drop / alter / truncate
- 3) 数据操纵语言 DML : insert / update / delete
- 4) 事务控制语言 TCL : commit / rollback / savepoint
- 5) 数据控制语言 DCL : grant / revoke (与用户权限相关，后续讲解)

1.4. Emp 表、Dept 表、Salgrade 表数据结构



- Emp 表

运行 SQL 命令行

SQL> desc emp_xxx;

名称	是否为空? 类型
EMPNO	NUMBER (4)
ENAME	VARCHAR2 (20)
JOB	VARCHAR2 (15)
SALARY	NUMBER (7, 2)
BONUS	NUMBER (7, 2)
HIREDATE	DATE
MGR	NUMBER (4)
DEPTNO	NUMBER (10)

运行 SQL 命令行

SQL> select * from emp_xxx;

EMPNO	ENAME	JOB	SALARY	BONUS	HIREDATE	MGR	DEPTNO
1012	amy				05-12月-11		
1001	张无忌	Manager	10000	2000	12-5月 -10	1005	10
1002	刘苍松	Analyst	8000	1000	01-4月 -11	1001	10
1003	李翊	Analyst	9000	1000	11-4月 -10	1001	10
1004	郭芙蓉	Programmer	5000		01-1月 -11	1001	10
1005	张三丰	President	15000		15-5月 -08		20
1006	燕小六	Manager	5000	400	01-2月 -09	1005	20
1007	陆无双	clerk	3000	500	01-2月 -09	1006	20
1008	黄蓉	Manager	5000	500	01-5月 -09	1005	30
1009	韦小宝	salesman	4000		20-2月 -09	1008	30
1010	郭靖	salesman	4500	500	10-5月 -09	1008	30
1013	余泽成						

已选择12行。

- Dept 表

```
运行 SQL 命令行
SQL> desc dept_xxx;
名称                                是否为空? 类型
-----
DEPTNO                             NUMBER(2)
DNAME                               VARCHAR2(20)
LOCATION                             VARCHAR2(20)

SQL> select * from dept_xxx;

DEPTNO DNAME      LOCATION
-----
10 developer  beijing
20 account   shanghai
30 sales      guangzhou
40 operations tianjin
```

- **Salgrade 表**

```
运行 SQL 命令行
SQL> desc salgrade_xxx;
名称                                是否为空? 类型
-----
GRADE                               NUMBER(2)
LOWSAL                              NUMBER(7, 2)
HISAL                               NUMBER(7, 2)

SQL> select * from salgrade_xxx;

GRADE    LOWSAL    HISAL
-----
1        10001    99999
2         8001    10000
3         6001     8000
4         4001     6000
5          1     4000
```

2. DML 操作**

DML 数据控制语句，包括插入、更新、删除操作。

2.1.insert(插入数据)**

2.1.1. 使用方法：插入全部列值时，不用写列名

【案例 2】insert 演示：不写列名

--必须提供全部的列数据，数据的顺序必须按照表结构

--如果有数据没提供，用 null 表示

SQL> insert into dept_xxx values(66, 'market', null);

SQL> insert into emp_xxx

values(1020, 'rory', 'Programmer', 6000, null, null, sysdate, 10);

The screenshot shows two SQL command windows. The top window displays the contents of the 'dept_xxx' table, which includes departments 10 through 66. The bottom window displays the contents of the 'emp_xxx' table, listing employees from EMPNO 1001 to 1013, with the newly inserted employee 1020 (rory) highlighted at the bottom.

运行 SQL 命令行

DEPTNO	DNAME	LOCATION
10	developer	beijing
20	account	shanghai
30	sales	guangzhou
40	operations	tianjin
66	market	

运行 SQL 命令行

SQL> select * from emp_xxx;

EMPNO	ENAME	JOB	SALARY	BONUS	HIREDATE
1012	amy				05-12月-11
1001	张无忌	Manager	10000	2000	12-5月 -10
1002	刘苍松	Analyst	8000	1000	01-4月 -11
1003	李翊	Analyst	9000	1000	11-4月 -10
1004	郭芙蓉	Programmer	5000		01-1月 -11
1005	张三丰	President	15000		15-5月 -08
1006	燕小六	Manager	5000	400	01-2月 -09
1007	陆无双	clerk	3000	500	01-2月 -09
1008	黄蓉	Manager	5000	500	01-5月 -09
1009	韦小宝	salesman	4000		20-2月 -09
1010	郭靖	salesman	4500	500	10-5月 -09
1013	余泽成				
1020	rory	Programmer	6000		07-12月-11

已选择13行。

2.1.2. 使用方法：写列名

建议采用写列名的方式，即使是插入全部数据，也建议把列名写全

【案例 3】insert 演示：写列名

```
SQL> insert into emp_xxx( empno , salary , ename , hiredate )  
      values( 1020 , 6000 , 'rory' ,  
            to_date( '2011/10/10' , 'yyyy/mm/dd' ) );  
  
-- to_date 函数是 oracle 独有的  
-- 在 oracle 数据库中插入日期数据，一定要用 to_date 处理  
-- 不要采用默认格式，比如：'10-OCT-11'
```

2.1.3. 复制表 **

2.1.3.1. 复制全表

```
create table 表名  
as  
查询语句；
```

2.1.3.2. 只复制结构，不复制数据

【案例 4】复制表：只复制结构，不复制数据

```
SQL> create table salgrade_yyy  
      as  
      select * from salgrade_xxx  
      where 1 <> 1;
```

2.1.3.3. 复制一部分数据(给查询语句加条件)

如果复制表时的查询语句中有表达式或者函数(包括单行函数和组函数)，必须指定新表中的列名
指定方式：给列设置别名 ;或者在新表中设置列名

【案例 5】复制表：复制一部分数据

```
--通过设置别名的方式，指定新表中的列名
SQL> create table emp_yyy
      as
      select empno , ename , salary*12 year_sal  --year_sal 为新表的列名
      from emp_xxx
      where deptno = 10 ;
```

【案例 6】复制表：复制一部分数据

```
--通过在新表中设置列名的方式，指定新表中的列名
SQL> create table emp_count( did , emp_num )  --新表中的列名
      as
      select deptno , count(*)
      from emp_xxx
      group by deptno ;
```

2.1.3.4. 创建一个空表，并同时向表中插入多条记录

【案例 7】创建一个同 emp 表结构相同的空表，将部门号为 10 的员工信息放入该表

情景描述

如果有一张表 emp 的数据量为一百万条，此时需要建立 1 张测试表只放入少量测试数据(如 100 条)，执行步骤如下所示：

● 第 1 步 创建一个空表

```
SQL> create table emp_bak1  --emp_bak1 表结构与 emp_xxx 表结构完全相同
      as
      select * from emp_xxx
      where 1 = 0 ;
```

● 第 2 步 插入少量测试数据

```
SQL> insert into emp_bak1
      ( select * from emp_xxx where deptno = 10 ) ;
```

【案例 8】把表中的数据换为部门 20 和 30 的员工记录

```
SQL> delete from emp_bak1 ;  -- 删除全表的数据( from 可以省略 )
```

```
SQL> insert into emp_bak1
      ( select * from emp_xxx where deptno in (20, 30) );
```

2.1.4. rownum 关键字 **

rownum 是 Oracle 数据库提供的，代表行号。

【案例 9】向新表中插入指定记录数的数据，比如前 8 条

```
SQL> delete from emp_bak1 ;
SQL> insert into emp_bak1
      ( select * from emp_xxx where rownum <= 8 ); -- rownum 行号
```

注意：复制表的时候，不复制约束条件

2.2. update(更新数据) **

语法结构：

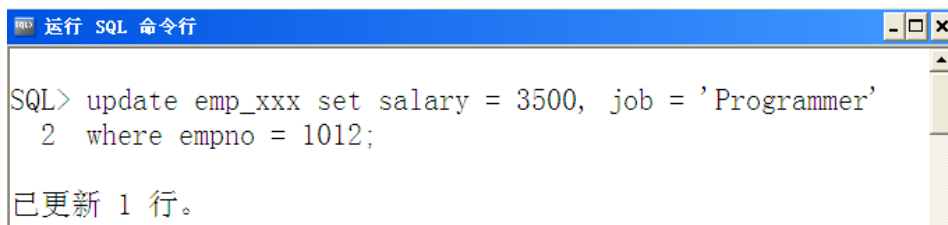
```
update 表名 set 列名 = 新的列值,
              列名 = 新的列值.
...
where 条件;
```

注意：

- ✓ 更新(update)数据表时，注意条件，如果不加条件，修改的是全部表记录
- ✓ rollback 回退，commit 确认

【案例 10】将员工号为 1012 的员工薪水改为 3500，职位改为 Programmer

```
SQL> update emp_xxx
      set salary = 3500 , job = 'Programmer'
      where empno = 1012 ;
```



```
运行 SQL 命令行
SQL> update emp_xxx set salary = 3500, job = 'Programmer'
      2  where empno = 1012;

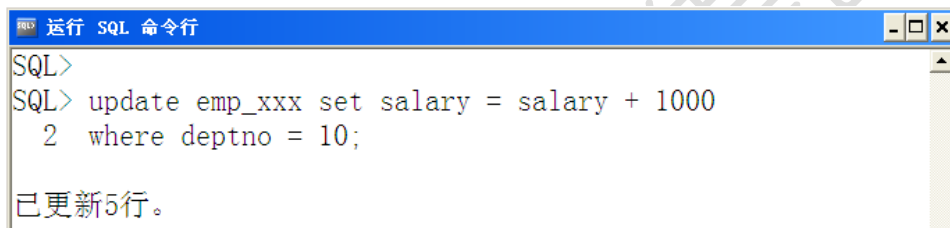
已更新 1 行。
```

```
SQL> select empno , ename , salary , job
      2  from emp_xxx
      3  where empno=1012;
```

EMPNO	ENAME	SALARY	JOB
1012	amy	3500	Programmer

【案例 11】部门 10 的员工薪水+1000

```
SQL> update emp_xxx set salary = salary + 1000
      where deptno = 10 ;
```



```
运行 SQL 命令行
SQL>
SQL> update emp_xxx set salary = salary + 1000
      2  where deptno = 10;

已更新5行。
```

2.3.delete(删除数据)**

语法结构：

delete [from] 表名 where 条件；

注意：

- ✓ 如果删除语句中不加 where 条件，将删掉表中的全部记录
- ✓ rollback 回退，commit 确认
- ✓ drop table 会删除表结构和数据；truncate 删除表数据，保留表结构。Drop 和 truncate 都**不可以**回退。delete 仅删除数据，可以回退

【案例 12】delete 演示

```
SQL> delete emp_bak1 where empno = 1002 ;
SQL> delete emp_bak1 where deptno = 10 ;
SQL> delete emp_bak1 ;                --删除 emp_bak1 表中所有数据
SQL> commit ;
```

2.3.1. 删除表中的重复数据

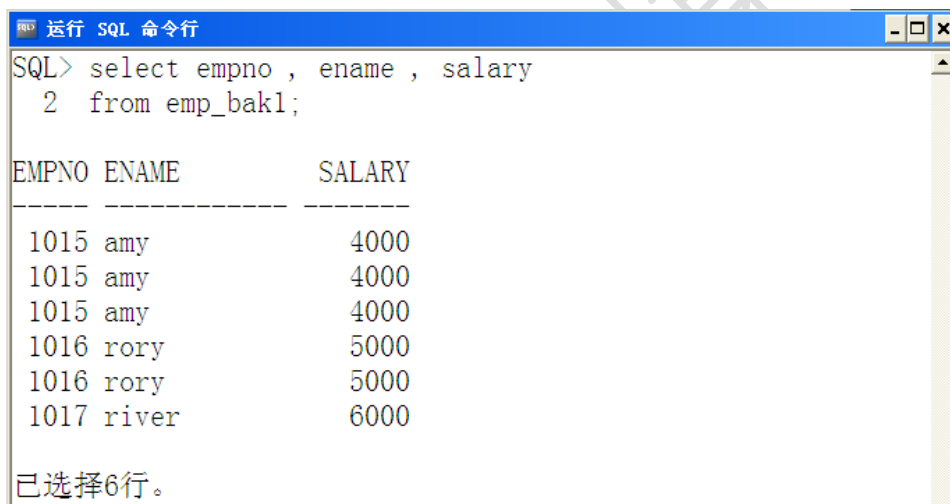
【案例 13】创建表 emp_bak2，只存放不重复的记录

- 步骤 1：数据准备，人为制造一个有重复记录的表

```
--如下语句执行 3 遍，插入 3 条重复数据
SQL> insert into emp_bak1(empno , ename , salary )
      values( 1015 , 'amy' , 4000 );

--如下语句执行 2 遍，插入 2 条重复数据
SQL> insert into emp_bak1(empno , ename , salary)
      values( 1016 , 'rory' , 5000 );

--如下语句执行 1 遍，插入 1 条数据
SQL> insert into emp_bak1(empno , ename , salary )
      values( 1017 , 'river' , 6000 );
```



```
SQL> select empno , ename , salary
2  from emp_bak1;
```

EMPNO	ENAME	SALARY
1015	amy	4000
1015	amy	4000
1015	amy	4000
1016	rory	5000
1016	rory	5000
1017	river	6000

已选择6行。

- 步骤 2：创建表 emp_bak2，只存放不重复的记录，利用 distinct 关键字

```
SQL> create table emp_bak2
      as
      select distinct empno , ename , salary ,
                     hiredate , job , bonus , deptno , mgr
      from emp_bak1 ;
```

```

运行 SQL 命令行
SQL>
SQL> create table emp_bak2
2 as
3 select distinct empno, ename, salary,
4                hiredate, job, bonus, deptno, mgr
5 from emp_bak1;

表已创建。

SQL> select empno , ename , salary from emp_bak2;

EMPNO  ENAME          SALARY
-----
1015 amy              4000
1017 river            6000
1016 rory              5000

```

2.3.2. rename 关键字

【案例 14】将表改名

```

SQL> drop table emp_bak1 ;          -- 删除表 emp_bak1
SQL> rename emp_bak2 to emp_bak1 ; -- 把 emp_bak2 改名为 emp_bak1

```

2.3.3. rowid 关键字 删除重复数据(较简单 , 性能高)

- 1) rowid 是 Oracle 数据库的伪列 , 可以看作是一条数据在数据库中的物理位置
- 2) rowid 是 Oracle 数据库独有的

注意 : 每一条记录的 rowid 在数据库中都是唯一的

【案例 15】rowid 演示

```

运行 SQL 命令行

SQL> select ename , rowid from emp_xxx;

ENAME          ROWID
-----
amy            AAADVFAABAAAI1aAAA
张无忌        AAADVFAABAAAI1aAAB
刘苍松        AAADVFAABAAAI1aAAC
李翊          AAADVFAABAAAI1aAAD
郭芙蓉        AAADVFAABAAAI1aAAE
张三丰        AAADVFAABAAAI1aAAF
燕小六        AAADVFAABAAAI1aAAG
陆无双        AAADVFAABAAAI1aAAH
黄蓉          AAADVFAABAAAI1aAAI
韦小宝        AAADVFAABAAAI1aAAJ
郭靖          AAADVFAABAAAI1aAAK
余泽成        AAADVFAABAAAI1aAAL
rory          AAADVFAABAAAI1aAAM

已选择13行。

```

【案例 16】删除表中重复记录

● 步骤 1：数据准备

```

SQL> delete from emp_bak1 ;
--如下语句执行 3 遍，插入 3 条重复数据
SQL> insert into emp_bak1( empno , ename , salary )
values( 1015 , 'amy' , 4000 ) ;
--如下语句执行 2 遍，插入 2 条重复数据
SQL> insert into emp_bak1( empno , ename , salary)
values( 1016 , 'rory' , 5000 ) ;
--如下语句执行 1 遍，插入 1 条数据
SQL> insert into emp_bak1( empno , ename , salary )
values( 1017 , 'river' , 6000 ) ;

```



```
运行 SQL 命令行
SQL> select empno , ename , salary
      2  from emp_bak1;

EMPNO  ENAME          SALARY
-----  -
1015  amy              4000
1015  amy              4000
1015  amy              4000
1016  rory             5000
1016  rory             5000
1017  river            6000

已选择6行。
```

步骤 2：删除重复数据

```
SQL> delete from emp_bak1
      where rowid not in ( select max(rowid) from emp_bak1
                          group by empno , ename , salary ) ;

--子查询：查询出 empno , ename , salary 相同的 rowid 最大的记录
--主查询：删除 rowid 不在子查询之列的重复数据
```

```
运行 SQL 命令行
SQL> select empno , ename , rowid from emp_bak1;

EMPNO  ENAME          ROWID
-----  -
1015  amy              AAADViAABAAAKXCAAA
1015  amy              AAADViAABAAAKXCAAB
1015  amy              AAADViAABAAAKXCAAC
1016  rory             AAADViAABAAAKXCAAD
1016  rory             AAADViAABAAAKXCAAE
1017  river            AAADViAABAAAKXCAAF

已选择6行。
```

```
SQL> select max(rowid) from emp_bak1
2 group by empno, ename, salary;
```

MAX (ROWID)

```
-----
AAADViAABAAAKXCAAC
AAADViAABAAAKXCAAE
AAADViAABAAAKXCAAF
```

运行 SQL 命令行

```
SQL> delete from emp_bak1
2 where rowid not in (select max(rowid) from emp_bak1
3                      group by empno, ename, salary);
```

已删除3行。

```
SQL> select empno , ename , salary from emp_bak1;
```

EMPNO	ENAME	SALARY
1015	amy	4000
1016	rory	5000
1017	river	6000

3. Transaction(事务)**

- 1) 事务是一组 DML 操作的逻辑单元，用来保证数据的一致性。
- 2) 在一个事务内，组成事务的这组 DML 操作，或者一起成功提交，或者一起被撤销。
- 3) 事务控制语言 TCL(Transaction Control Language)
 - ✓ commit 事务提交 将所有的数据改动提交
 - ✓ rollback 事务回滚 回退到事务之初，数据的状态和事务开始之前完全一致
 - ✓ savepoint 事务保存点(较不常用)

3.1. 事务的开始和终止(事务边界)

- 1) 事务开始
事务开始于上一个事务的终止或者第一条 DML 语句

2) 事务终止

- ✓ 事务终止于 commit/rollback 显式操作(即控制台输入 commit/rollback)
- ✓ 如果连接关闭，事务(Transaction)将隐式提交
- ✓ DDL 操作(比如 create)，事务将隐式提交
- ✓ 如果出现异常，事务将隐式回滚。

3.2. 事务中的数据状态

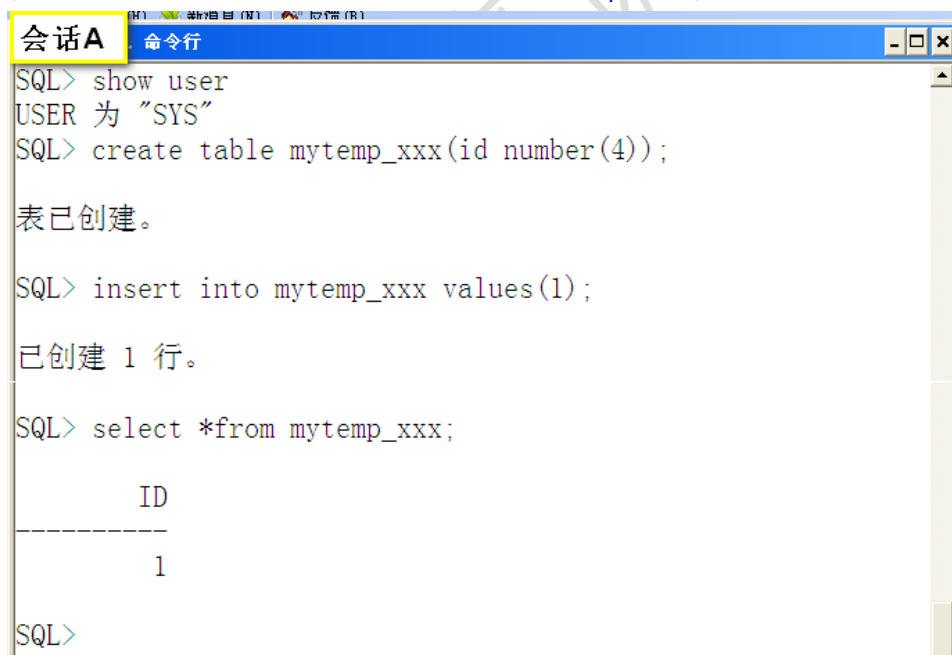
3.2.1. 如果多个会话操作同一张表的数据

当用户与服务器建立连接成功后，服务器端 Oracle 将与客户端建立一个会话(Session)。客户端与 Oracle 的交互都是在这个会话环境中进行的。

【案例 17】Transaction 演示

● 步骤 1

开启一个会话 A，创建表并插入 1 条数据(注意：不提交)
(注意练习时使用同一个用户在两个窗口中登录，比如 openlab)



```

SQL> show user
USER 为 "SYS"
SQL> create table mytemp_xxx(id number(4));
表已创建。
SQL> insert into mytemp_xxx values(1);
已创建 1 行。
SQL> select *from mytemp_xxx;
      ID
-----
      1
SQL>
    
```

● 步骤 2

开启第 2 个会话 B，在会话 A 进行 commit 之前，会话 B 只能查看表结构，查看不到数据

```
SQL> select *from mytemp_xxx;
```

ID
1

```
SQL>
```

会话B 命令行

```
SQL> show user
USER 为 "SYS"
SQL> desc mytemp_xxx;
```

名称	是否
ID	

```
SQL> select * from mytemp_xxx;
```

未选定行

```
SQL>
```

步骤 3

会话 A 中进行 commit 操作后 ;会话 B 中就可以查看数据了

会话A 命令行

```
SQL> show user
USER 为 "SYS"
SQL> create table mytemp_xxx(id number(4));
```

表已创建。

```
SQL> insert into mytemp_xxx values(1);
```

已创建 1 行。

```
SQL> select *from mytemp_xxx;
```

ID
1

```
SQL> commit
```

提交完成。

```
SQL>
```

会话B 命令行

```
SQL> show user
USER 为 "SYS"
SQL> select * from mytemp_xxx;
```

ID
1

```
SQL>
```

- 步骤 4

会话 A 进行 update 操作(没有 commit), 会话 B 看到的仍然是原先的数据

会话A 命令行

```
SQL> update mytemp_xxx set id=1001;

已更新 1 行。

SQL>
```

会话B 命令行

```
SQL> select * from mytemp_xxx;

      ID
-----
      1

SQL>
```

- 步骤 5

会话 A 提交(commit)后, 会话 B 看到被改变的结果

会话A 命令行

```
SQL> update mytemp_xxx set id=1001;

已更新 1 行。

SQL> commit;

提交完成。

SQL>
```

会话B 命令行

```
SQL> select * from mytemp_xxx;

      ID
-----
      1

SQL> /

      ID
-----
     1001

SQL>
```

- 步骤 6

会话 A 进行 update 操作(没有 commit), 会话 B 进行 delete 操作时被挂起, 因为试图操作相同的数据。

会话A L 命令行

```
SQL> update mytemp_xxx set id=1234;
```

已更新 1 行。

SQL>

会话B L 命令行

```
SQL> delete from mytemp_xxx;
```

- **步骤 7**

会话 A 提交(commit), 会话 B 结束阻塞状态, 开始执行

会话A L 命令行

```
SQL> update mytemp_xxx set id=1234;
```

已更新 1 行。

```
SQL> commit;
```

提交完成。

SQL>

会话B L 命令行

```
SQL> delete from mytemp_xxx;
```

已删除 1 行。

SQL>

- **步骤 8**

会话 A 更新后进行回滚操作(rollback)

会话A

SQL 命令行

```
SQL> select *from mytemp_xxx;
```

ID
1234

```
SQL> update mytemp_xxx set id=5678;
```

已更新 1 行。

```
SQL> rollback;
```

回退已完成。

```
SQL> select *from mytemp_xxx;
```

ID
1234

```
SQL>
```

会话B

SQL 命令行

```
SQL> select * from mytemp_xxx;
```

ID
1234

```
SQL>
```

● 结论

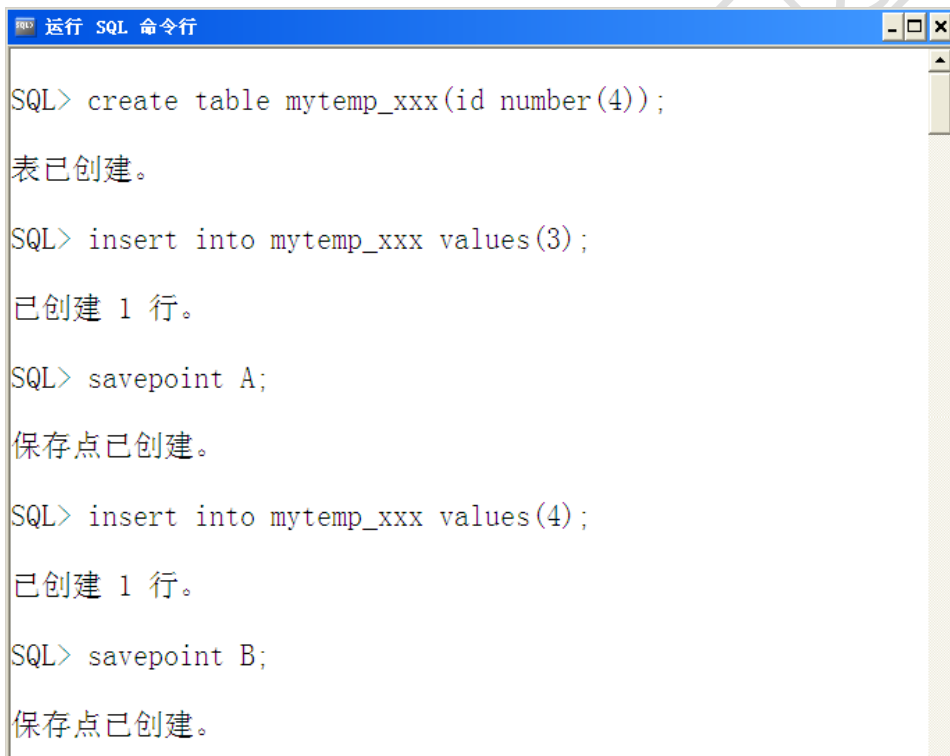
- 1) 事务内部的数据改变只有在自己的会话中能够看到
- 2) 事务会对操作的数据加锁，不允许其它事务操作
- 3) 如果提交(commit)后，数据的改变被确认，则
 - ✓ 所有的会话都能看到被改变的结果；
 - ✓ 数据上的锁被释放；
 - ✓ 保存数据的临时空间被释放
- 4) 如果回滚(rollback)，则
 - ✓ 数据的改变被取消；
 - ✓ 数据上的锁被释放；
 - ✓ 临时空间被释放

3.2.2. Savepoint

设置保存点，可以回滚(rollback)到指定的保存点。

【案例 18】savepoint 演示

```
SQL> create table mytemp_xxx( id number(4) );
      --事务起点
SQL> insert into mytemp_xxx values(3);
SQL> savepoint A;    -- 设置保存点，名为 A
SQL> insert into mytemp_xxx values(4);
SQL> savepoint B;    -- 设置保存点，名为 B
SQL> insert into mytemp_xxx values(5);
SQL> rollback to A;  -- 回滚到保存点 A，注意：A 之后的保存点全部被取消
SQL> select * from mytemp_xxx;  --3 被插入数据库，4、5 没有被插入
```



运行 SQL 命令行

```
SQL> create table mytemp_xxx(id number(4));
表已创建。

SQL> insert into mytemp_xxx values(3);
已创建 1 行。

SQL> savepoint A;
保存点已创建。

SQL> insert into mytemp_xxx values(4);
已创建 1 行。

SQL> savepoint B;
保存点已创建。
```



```
SQL> insert into mytemp_xxx values(5);
```

已创建 1 行。

```
SQL> rollback to A;
```

回退已完成。

```
SQL> select * from mytemp_xxx;
```

ID
3

```
SQL>
```

4. DDL 操作

数据定义语言 DDL : create / drop / alter / truncate

4.1. create(建表)**

建表的两种方式：

1) 第 1 种，自定义表的列和数据类型

```
create table 表名(
    列名 列的数据类型,
    ....
);
```

2) 第 2 种，由一个现存的表复制新表

```
create table 表名
as
查询语句；
```

4.2. drop(删除结构和全部的表数据)**

1) drop 语法结构：drop table 表名；

4.2.1. 和表相关的数据字典 *

- 1) user_tables
字段 [table_name](#) 表名
- 2) user_objects
字段 [created](#) 表的创建时间

【案例 19】找出 11 年 12 月 07 日后创建的表，删除过时的表。

注意日期格式，下例是英文环境下，如果是中文环境，需要写成：'7-12 月-11' 这种形式。也可以使用 to_date 函数。

```
--找出所有 11 年 12 月 7 日后创建的表
SQL> select a.table_name, b.created
      from user_tables a join user_objects b
      on a.table_name = b.object_name
      where b.created > '7-DEC-11';
```

TABLE_NAME	CREATED
USERS_DS	16-DEC-11
STOCK	16-DEC-11
PAYINFO	16-DEC-11

```
SQL> drop table users_ds;
```

```
SQL> select a.table_name, b.created
      2  from user_tables a join user_objects b
      3  on a.table_name = b.object_name
      4  where b.created > '07-DEC-11';
```

TABLE_NAME	CREATED
DEPT_XXX	14-DEC-11
USERS_DS	16-DEC-11
QUESTIONS_DS	16-DEC-11
QUESTIONS_MARKET	16-DEC-11
MERCHANDISE	16-DEC-11
STOCK	16-DEC-11
PAYINFO	16-DEC-11

【案例 20】计算五个月之前创建的数据表的个数

```
SQL> select count(a.table_name)
      from user_tables a join user_objects b
      on a.table_name = b.object_name
      where b.created < add_months(sysdate, -5);
```

```
SQL> select count(a.table_name)
      from user_tables a join user_objects b
      on a.table_name = b.object_name
      where b.created < add_months(sysdate, -5);
```

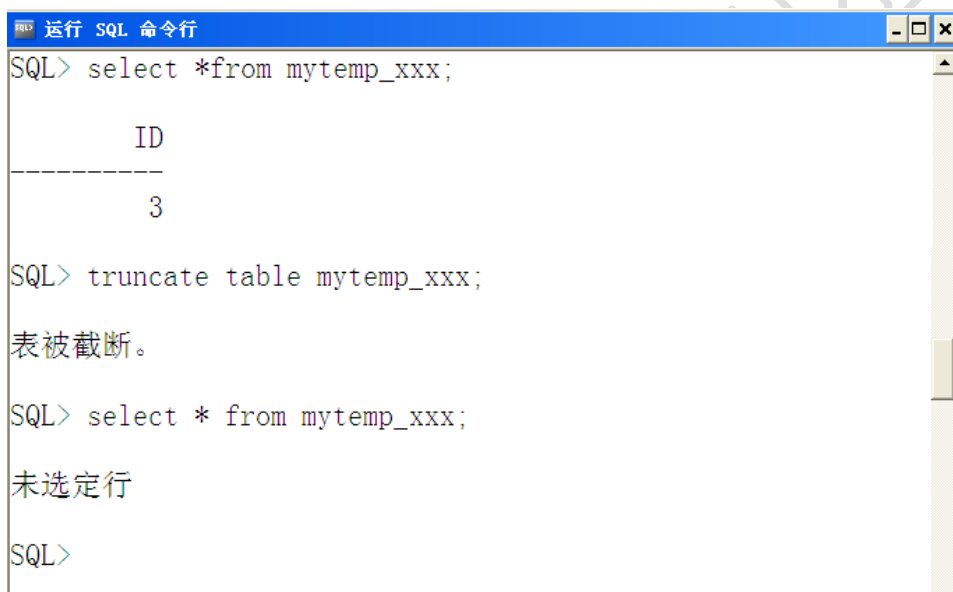
```
COUNT(A.TABLE_NAME)
```

21

4.3. truncate(截取 , 截断) *

- 1) truncate 保留表结构，删除表中所有数据
- 2) truncate 操作不需提交(commit)，没有回退(rollback)的机会
- 3) 语法结构： **truncate table 表名 ;**
- 4) truncate 与 delete 的区别：
 - ✓ truncate 在功能上等同于：delete + commit
 - ✓ delete 操作将删除数据存储到临时空间中，不直接删除，可以回退。
truncate 操作直接删除，不占用临时空间，不能回退。

【案例 21】truncate 演示



```

运行 SQL 命令行
SQL> select *from mytemp_xxx;

      ID
-----
      3

SQL> truncate table mytemp_xxx;

表被截断。

SQL> select * from mytemp_xxx;

未选定行

SQL>
    
```

4.4. alter(修改结构) *

4.4.1. add 关键字

【案例 22】增加列(只能增加在最后一列)

```

SQL> create table mytemp_xxx(id number(4));
SQL> alter table mytemp_xxx add(name char(10));
SQL> alter table mytemp_xxx add(password char(4));
SQL> desc mytemp_xxx
    
```

4.4.2. rename 关键字

【案例 23】修改列名 password 为 pwd

```
SQL> alter table mytemp_xxx rename column password to pwd;  
SQL> desc mytemp_xxx
```

4.4.3. modify 关键字

【案例 24】修改列的数据类型为 pwd char(8)

```
SQL> alter table mytemp_xxx modify (pwd char(8));
```

4.4.4. drop column

【案例 25】删除列 pwd

```
SQL> alter table mytemp_xxx drop column pwd;
```

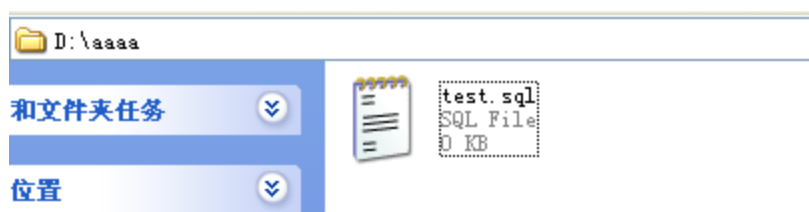
注意：对加列、减列、修改列的操作谨慎进行，不建议操作。

4.5. 执行数据库脚本(script)文件*.sql **

【案例 26】Windows 操作系统：

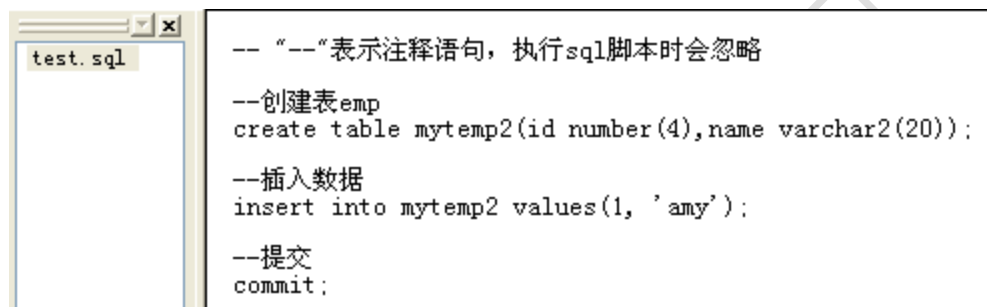
- 步骤 1

在指定路径新建一个.sql 为后缀名的文件



● 步骤 2

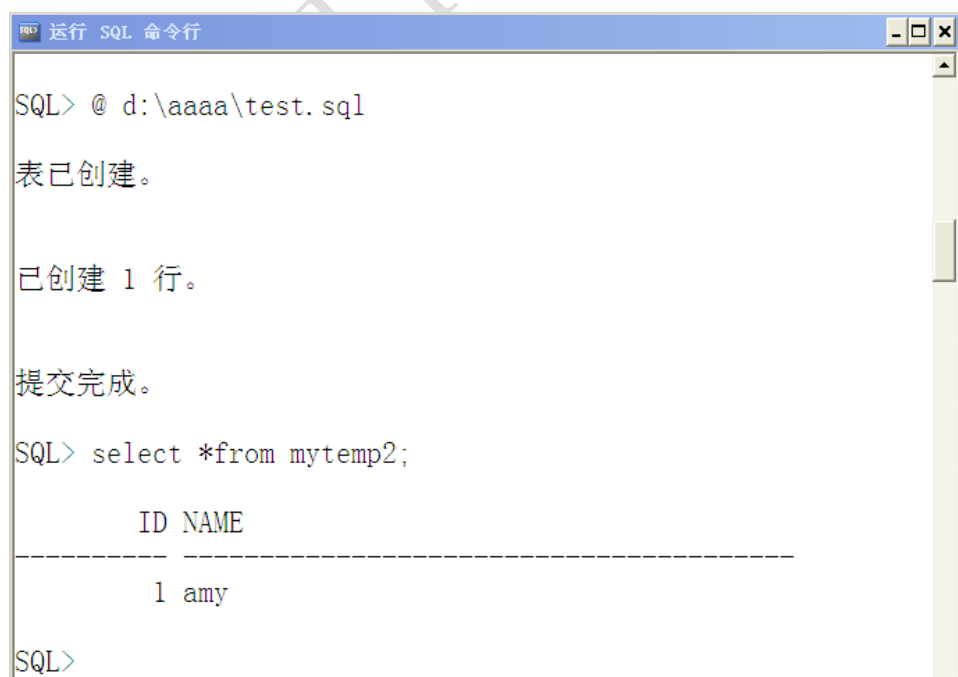
编辑



● 步骤 3

在服务器端运行执行.sql 脚本(脚本文件和 sqlplus 在同一台机器上)

```
SQL> @ d:\aaaa\test.sql
```



【案例 27】Linux 操作系统

1) 步骤 1：创建一个 ning.sql 文本文件：

%提示符下输入 vi 文件名, 在编辑区输入内容如下：

```
sunv210% vi ning.sql
```

```
Telnet 192.168.0.26
create table temp_ning(id number primary key,
name char(20));
insert into temp_ning values(1, 'peter');
commit;
```

并在命令模式下输入：x 保存退出。

2) 查看文件路径：

```
sunv210% pwd ning.sql
/user/openlab
sunv210%
```

3) 在 sqlplus 中执行此文件：

```
SQL> @ /user/openlab/ning.sql

Table created.

1 row created.

Commit complete.

SQL>
```

5. 数据控制语言 DCL

【案例 28】权限演示

● 步骤 1

假设数据库中有 2 个用户，openlab 和 ninglj



```

SQL> show user
USER is "NINGLJ"
SQL>

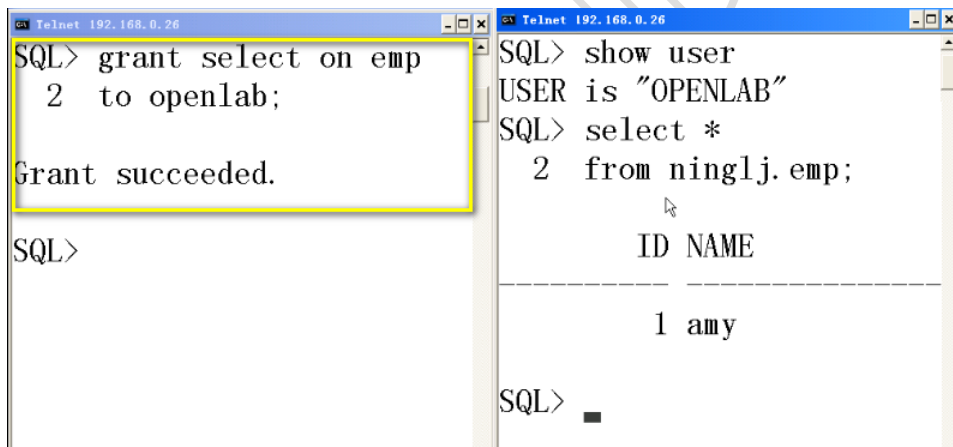
SQL> show user
USER is "OPENLAB"
SQL>
    
```

● 步骤 2

假设现在的用户是 ninglj

ninglj 将查看 emp 表的权限赋予 openlab，openlab 只能看不能改

```
SQL> grant select on emp to openlab;
```



```

SQL> grant select on emp
  2 to openlab;

Grant succeeded.

SQL>

SQL> show user
USER is "OPENLAB"
SQL> select *
  2 from ninglj.emp;

      ID NAME
-----
      1 amy

SQL>
    
```

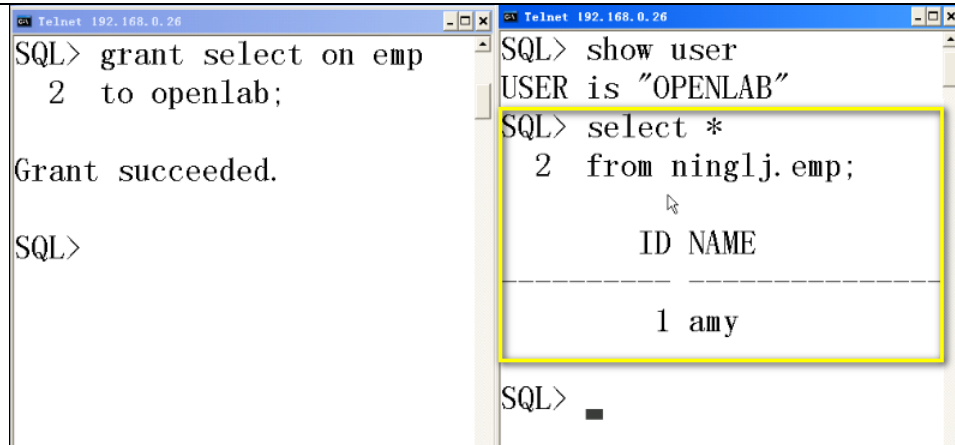
● 步骤 3

假设现在的用户是 openlab

openlab 可以用 ninglj.emp 的方式查询 ninglj 账户的表 emp

```

SQL> select * from ninglj.emp ; --查询到 ninglj 账户下的 emp 表
SQL> select * from emp ;      --查询自己账户下的 emp 表
    
```

```

SQL> grant select on emp
  2  to openlab;

Grant succeeded.

SQL>
  
```

```

SQL> show user
USER is "OPENLAB"
SQL> select *
  2  from ninglj.emp;

      ID NAME
-----
      1 amy
  
```

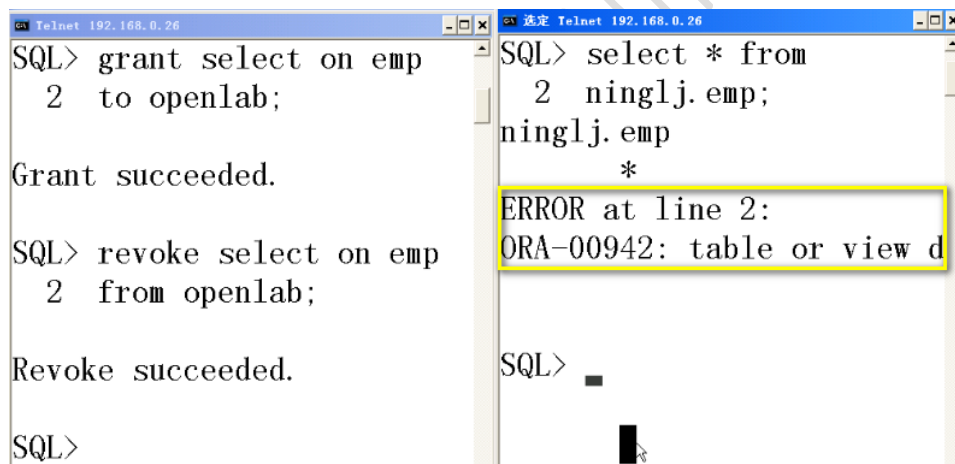
```

SQL>
  
```

● 步骤 4

ninglj 用户将 select 权限回收 ;openlab 账户再次访问时出错

```
SQL> revoke select on emp from openlab ;
```



```

SQL> grant select on emp
  2  to openlab;

Grant succeeded.

SQL> revoke select on emp
  2  from openlab;

Revoke succeeded.

SQL>
  
```

```

SQL> select * from
  2  ninglj.emp;
ninglj.emp
      *
ERROR at line 2:
ORA-00942: table or view does not exist
  
```

```

SQL>
  
```