

## 知识点列表

编号	名称	描述	级别
1	约束条件	五种约束条件的概念和使用语法	***
2	视图	视图的概念和定义语法	**
3	索引	索引的概念和定义语法	**
4	序列	序列的概念和定义语法	**

注：   \*\*"理解级别   \*\*\*"掌握级别   \*\*\*\*"应用级别

## 目录

1. 知识点回顾.....	4
1.1. Sql 语句的分类.....	4
1.2. CRUD.....	4
1.3. Oracle 数据库的用户 **.....	4
1.3.1. 用户账户.....	4
1.3.2. Sqlplus 登录命令.....	4
2. 约束条件 constraint.....	5
2.1. 主键( Primary key, 简称 PK ) **.....	5
2.1.1. 列级约束.....	5
2.1.2. 表级约束.....	6
2.2. 非空约束( not null , 简称 NN ) **.....	7
2.3. 唯一约束( Unique , 简称 UK ) **.....	8
2.3.1. 列级约束.....	8
2.3.2. 表级约束.....	10
2.4. 检查约束( Check , 简称 CK ) **.....	11
2.5. 外键( Foreign key, 简称 FK ) **.....	13
2.5.1. 学员表( Student )和专业表( Major )的表结构.....	13
2.5.2. 外键约束.....	16
2.6. 复制表 , 不复制约束条件.....	20
2.7. 建立约束条件的时机 *.....	20
2.7.1. 建表同时建立约束条件.....	20
2.8. 数据字典 *.....	21

3. 数据库的其他对象 .....	22
3.1. 数据库中的主要对象 * .....	22
3.2. 视图 View * .....	23
3.3. 索引 Index * .....	27
3.4. 序列 Sequence ** .....	29

达内IT培训集团

## 1. 知识点回顾

### 1.1. Sql 语句的分类

- 1) Select --重点掌握
- 2) DML : insert / update / delete
- 3) DDL : create / drop / truncate / alter
- 4) TCL : commit / rollback / savepoint
- 5) DCL : grant / revoke

### 1.2. CRUD

CRUD 是指在做计算处理时的增加(Create)、查询(Retrieve)( 重新得到数据 )、更新(Update)和删除>Delete)几个单词的首字母简写。

### 1.3. Oracle 数据库的用户 \*\*

#### 1.3.1. 用户账户

- 1) 当 Oracle 安装完毕，系统会提供 2 个默认账户： sys 和 system  
账户的密码在数据库安装时自定义输入
- 2) 数据库默认安装的账户( 做测试使用 )： scott 账户，密码 tiger
- 3) 中关村校区数据库账户： openlab/账户，密码 open123
- 4) 数据库管理员可以新建若干账户，比如： exam/exam123

#### 1.3.2. Sqlplus 登录命令

如果本地没有 sqlplus, 使用 telnet 登陆到远程服务器，再使用 sqlplus 工具。

如果本地有 sqlplus，登录远程数据库服务器的方式：

假设远程服务器 192.169.0.26 的 Oracle 数据库，数据库名为 tarena

```
C:\>sqlplus openlab/open123@192.168.0.26:1521/tarena

SQL*Plus: Release 11.1.0.7.0 - Production on 星期一 12月 19 18:39:51 2011

Copyright (c) 1982, 2008, Oracle. All rights reserved.

连接到:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options

SQL> █
```

## 2. 约束条件 constraint

### 2.1. 主键( Primary key, 简称 PK ) \*\*

- 1) 主键约束( primary key ) = 不能重复 + 不能为 null
- 2) 主键约束可以用两种方式定义：列级约束和表级约束

#### 2.1.1. 列级约束

##### 【案例 1】主键约束演示( 列级约束条件 )

```
SQL> create table dept_ning1(
    deptno number(2) primary key,           --列级约束条件
    dname varchar2(20),
    location varchar2(40)
);

SQL> insert into dept_ning1 values( 10 , 'developer' , 'beijing' );

SQL> insert into dept_ning1 values( 10 , 'market' , 'shenzhen' );--再执行一遍
-- 如果插入重复编码，会提示错误：
ORA-00001: unique constraint (NINGLJ.SYS_C003994) violated
-- 其中，SYS_C003994 是数据库自定义的主键名
-- 如果约束条件没有显式命名，数据库默认给约束条件命名为 SYS_C*****
```

```

运行 SQL 命令行

SQL> create table dept_ning1(
  2 deptno number(2) primary key,
  3 dname varchar2(20),
  4 location varchar2(40)
  5 );

表已创建。

SQL> insert into dept_ning1
  2 values(10,'developer','beijing');

已创建 1 行。

SQL> insert into dept_ning1
  2 values(10,'market','shenzhen');
insert into dept_ning1
*
第 1 行出现错误:
ORA-00001: 违反唯一约束条件 (SYS.SYS_C003994)

SQL>

```

### 2.1.2. 表级约束

#### 【案例 2】主键约束演示(表级约束条件)

```

SQL> create table dept_ning2(
      deptno number(2) ,
      dname varchar2(20) ,
      location varchar2(40) ,
      constraint dept_ning2_deptno_pk primary key (deptno)
    );

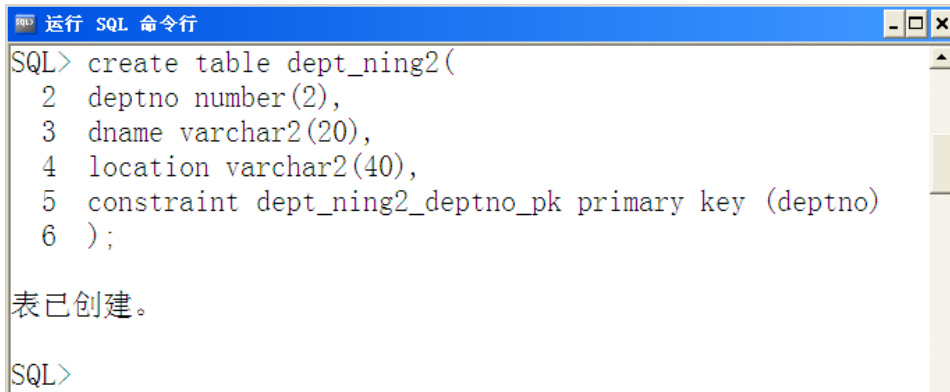
-- 表级约束条件
-- 建议约束命名规则：表名_列名_约束条件的类型
-- 当插入重复编码时，会提示具体的约束条件名字错误，方便定位出错的原因

SQL> insert into dept_ning2 values( 10 , 'developer' , 'beijing' ) ;

--再执行一遍，查看错误提示，将提示具体出错的约束条件名称

```

```
SQL> insert into dept_ning2 values( 10 , 'market' , 'shenzhen' );
```



```
运行 SQL 命令行
SQL> create table dept_ning2(
2  deptno number(2),
3  dname varchar2(20),
4  location varchar2(40),
5  constraint dept_ning2_deptno_pk primary key (deptno)
6 );

表已创建。

SQL>
```

## 2.2. 非空约束( not null , 简称 NN ) \*\*

**注意：**非空约束只能定义在列级

### 【案例 3】非空约束演示

```
--只能定义在列级
SQL> create table student_ning(
      id number(4) primary key,
      name varchar2(10) not null,
      age number(2)
);

SQL> insert into student_ning values(1, 'zhangwei', 20);

SQL> insert into student_ning values(2, 'zhangwei', 19);
--名字可以重复

SQL> insert into student_ning values(3, null, 18);
--报错：提示 name 列不能为 NULL
--这条语句也可以写成：
-- insert into student_ning(id, age) values(3, 18);
```

```

运行 SQL 命令行
SQL> create table student_ning(
  2   id number(4) primary key,
  3   name varchar2(10) not null,
  4   age number(2)
  5 );

表已创建。

SQL> insert into student_ning values(1, 'zhangwei', 20);

已创建 1 行。

SQL> insert into student_ning values(2, 'zhangwei', 19);

已创建 1 行。

SQL> insert into student_ning values(3, null, 18);
insert into student_ning values(3, null, 18)
                                           *
第 1 行出现错误:
ORA-01400: 无法将 NULL 插入 ("SYS"."STUDENT_NING"."NAME")

SQL>

```

#### 【案例 4】给非空约束命名

```

SQL> drop table student_ning; --删除表
SQL> create table student_ning(
      id number(4) primary key,
      name varchar2(10) constraint student_name_nn not null,
      age number(2)
);

```

## 2.3. 唯一约束( Unique , 简称 UK ) \*\*

### 2.3.1. 列级约束

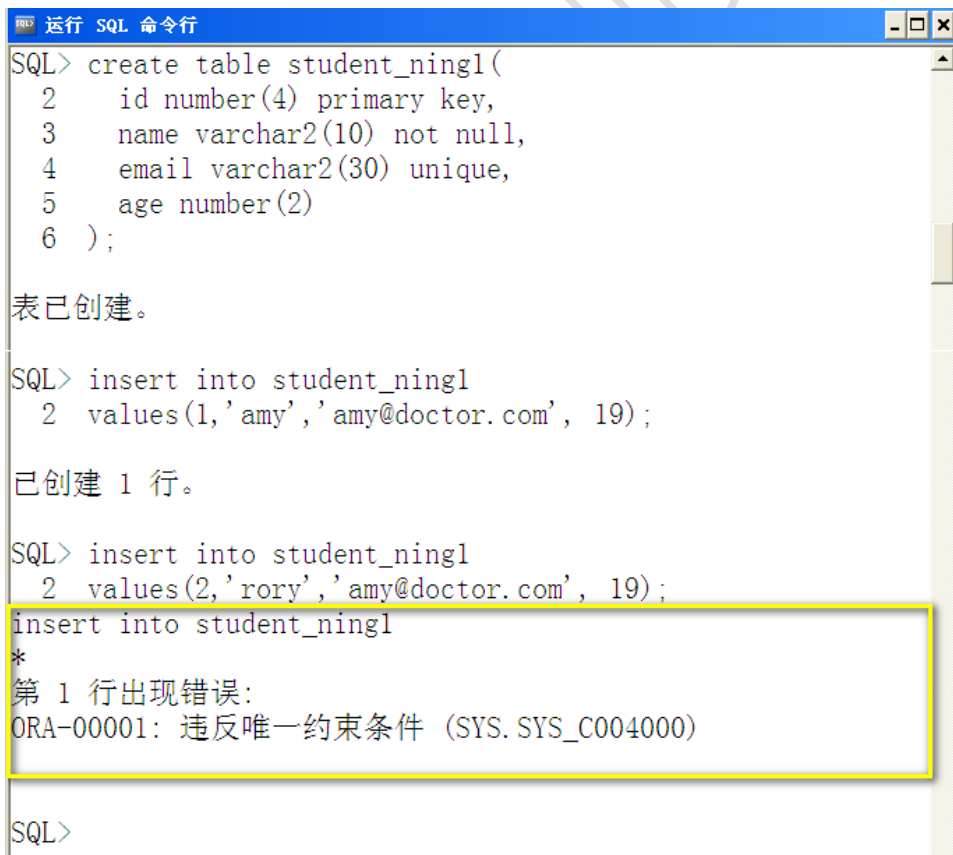


【案例 5】唯一约束演示( 列级约束条件 )

```
SSQL> create table student_ning1(
        id number(4) primary key,
        name varchar2(10) not null,
        email varchar2(30) unique,           --列级约束
        age number(2)
    );

SQL> insert into student_ning1
    values(1, 'amy', 'amy@doctor.com', 19);

SQL> insert into student_ning1
    values(2, 'rory', 'amy@doctor.com', 19);
--会出错误 : ORA-00001: 唯一约束条件被违反
```



```
运行 SQL 命令行
SQL> create table student_ning1(
2   id number(4) primary key,
3   name varchar2(10) not null,
4   email varchar2(30) unique,
5   age number(2)
6 );

表已创建。

SQL> insert into student_ning1
2 values(1,'amy','amy@doctor.com', 19);

已创建 1 行。

SQL> insert into student_ning1
2 values(2,'rory','amy@doctor.com', 19);
insert into student_ning1
*
第 1 行出现错误:
ORA-00001: 违反唯一约束条件 (SYS.SYS_C004000)

SQL>
```

### 2.3.2. 表级约束

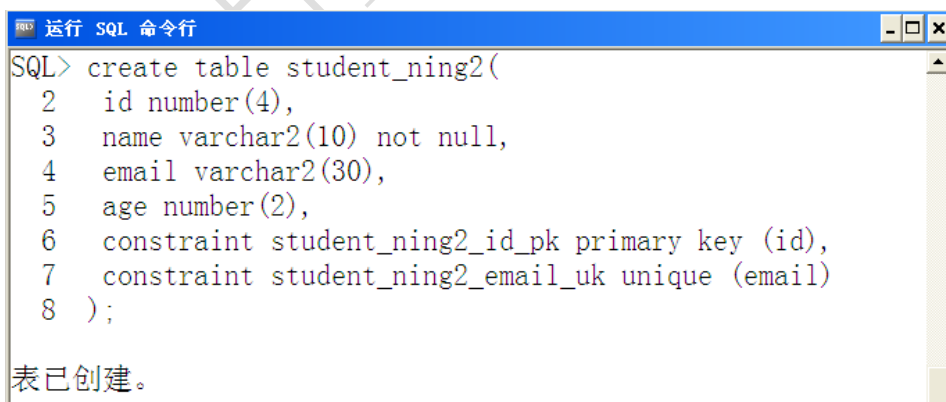
#### 【案例 6】唯一约束演示( 表级约束条件 )

```
SQL> create table student_ning2(
    id number(4),
    name varchar2(10) not null,
    email varchar2(30),
    age number(2),
    constraint student_ning2_id_pk primary key(id),
    constraint student_ning2_email_uk unique(email)
);
-- 主键约束建立在表级
-- 唯一约束建立在表级

SQL> insert into student_ning2 values(1,'amy','amy@doctor.com',19);

SQL> insert into student_ning2 values(2, 'rory', null, 20);
--unique 约束只要求不能重复，可以为 NULL

SQL> update student_ning2 set email = 'amy@doctor.com'
where id = 2;
--不管是 insert 还是 update，email 都不能重复
```



```
运行 SQL 命令行
SQL> create table student_ning2(
2 id number(4),
3 name varchar2(10) not null,
4 email varchar2(30),
5 age number(2),
6 constraint student_ning2_id_pk primary key (id),
7 constraint student_ning2_email_uk unique (email)
8 );
表已创建。
```

```
SQL> insert into student_ning2
  2 values(1,'amy','amy@doctor.com',19);

已创建 1 行。

SQL> insert into student_ning2
  2 values(2, 'rory', null, 20);

已创建 1 行。

SQL> update student_ning2 set email = 'amy@doctor.com'
  2 where id = 2;
update student_ning2 set email = 'amy@doctor.com'
*
第 1 行出现错误:
ORA-00001: 违反唯一约束条件 (SYS.STUDENT_NING2_EMAIL_UK)

SQL>
```

## 2.4. 检查约束( Check , 简称 CK ) \*\*

### 【案例 7】检查约束演示

```
SQL> create table student_ning3(
      id number(4) primary key,
      name varchar2(10) not null,
      email varchar2(30) unique,
      age number(2) check(age > 10),
      gender char(1) check(gender in('F', 'M') )--'F'代表女生 ;'M'代表男生
    );
```

因为约束条件建立在列级时可读性不好，而且不方便定义约束条件名字，一般建议定义在表级。

### 【案例 8】检查约束演示(建立在表级)

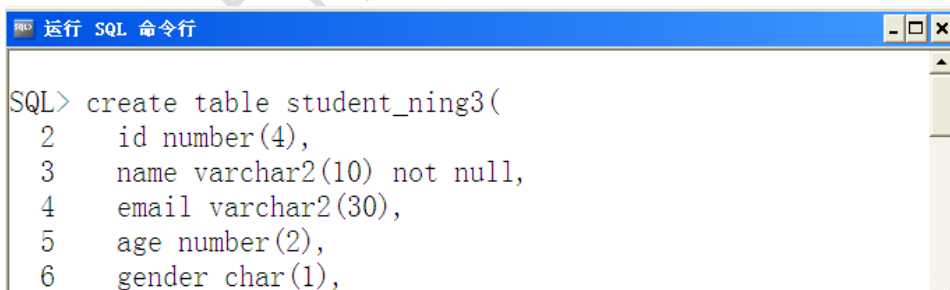
```
SQL> drop table student_ning3;
```

```
SQL> create table student_ning3(
        id number(4),
        name varchar2(10) not null,
        email varchar2(30),
        age number(2),
        gender char(1),          --'F'代表女生 ;'M'代表男生
        constraint student_ning3_id_pk
            primary key(id),
        constraint student_ning3_email_uk
            unique(email),
        constraint student_ning3_age_ck
            check(age > 10),
        constraint student_ning3_gender_ck
            check(gender in('F', 'M', 'f', 'm'))
    );

SQL> insert into student_ning3 values(1, 'amy', null, 19, 'F');

SQL> insert into student_ning3 values(2, 'rory', null, 8, 'M');
--违反 check 约束 age > 10

SQL> insert into student_ning3 values(3, 'doctor', null, 50, 'A');
--违反 check 约束 gender in ('F','M')
```



The screenshot shows a window titled "运行 SQL 命令行" (Run SQL Command Line). The command entered is:
 

```
SQL> create table student_ning3(
2   id number(4),
3   name varchar2(10) not null,
4   email varchar2(30),
5   age number(2),
6   gender char(1),
```

 The text is displayed in a monospaced font with line numbers 2 through 6 on the left margin.

```

7  constraint student_ning3_id_pk
8      primary key (id),
9  constraint student_ning3_email_uk
10     unique (email),
11  constraint student_ning3_age_ck
12     check (age > 10),
13  constraint student_ning3_gender_ck
14     check (gender in ('F', 'M', 'f', 'm'))
15 );

```

表已创建。

```

SQL> insert into student_ning3
2  values(1,'amy',null,19,'F');

```

已创建 1 行。

```

SQL> insert into student_ning3
2  values(2,'rory',null,8,'M');

```

```
insert into student_ning3
```

\*

第 1 行出现错误:

ORA-02290: 违反检查约束条件 (SYS.STUDENT\_NING3\_AGE\_CK)

```

SQL> insert into student_ning3
2  values(3,'doctor',null,50,'A');

```

```
insert into student_ning3
```

\*

第 1 行出现错误:

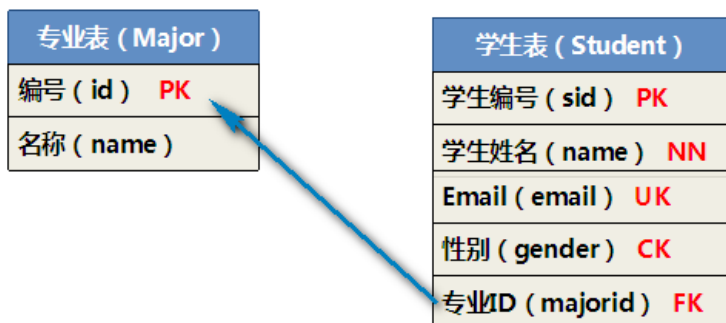
ORA-02290: 违反检查约束条件 (SYS.STUDENT\_NING3\_GENDER\_CK)

```
SQL>
```

## 2.5. 外键( Foreign key, 简称 FK ) \*\*

之前讲的几个约束条件都是用来约束单个表中的列，而外键约束定义在两个表的两个字段上( 或者一个表的两个字段上 )，用于保证相关两个字段的的关系。

### 2.5.1. 学员表( Student )和专业表( Major )的表结构



### 【案例 9】Major 和 Student 建表

```
--专业表( Major )
SQL> create table major_ning(
        id number(2) primary key,
        name char(20)
    );

SQL> insert into major_ning values(1, 'computer');
SQL> insert into major_ning values(2, 'history');
SQL> insert into major_ning values(3, 'music');
SQL> insert into major_ning values(4, 'sing');
SQL> commit ;

--学生表( Student )
SQL> create table student_ning4(
        sid number(3),
        name varchar2(20) not null,
        email varchar2(30),
        gender char(1),
        majorid number(2),
        constraint stu_n4_sid_pk
            primary key(sid),
        constraint stu_n4_email_uk
            unique (email),
        constraint stu_n4_g_ck
            check (gender in ('F','M')),
```

```
constraint stu_n4_mid_fk
foreign key (majorid) references major_ning(id)
);
```

- **Major 表**



```
运行 SQL 命令行
SQL> create table major_ning (
  2 id number(2) primary key,
  3 name char(20)
  4 );

表已创建。

SQL> insert into major_ning values(1, 'computer');

已创建 1 行。

SQL> insert into major_ning values(2, 'history');

已创建 1 行。

SQL> insert into major_ning values(3, 'music');

已创建 1 行。

SQL> insert into major_ning values(4, 'sing');

已创建 1 行。

SQL> commit;
```

- **Student 表**

```

运行 SQL 命令行
SQL> create table student_ning4(
2     sid number(3),
3     name varchar2(20) not null,
4     email varchar2(30),
5     gender char(1),
6     majorid number(2),
7     constraint stu_n4_sid_pk
8         primary key(sid),
9     constraint stu_n4_email_uk
10        unique (email),
11     constraint stu_n4_g_ck
12        check (gender in ('F','M')),
13     constraint stu_n4_mid_fk
14        foreign key (majorid) references major_ning(id)
15 );

表已创建。

SQL>

```

## 2.5.2. 外键约束

### 【案例 10】外键约束演示 1\_insert

```

SQL> insert into student_ning4 values(101, 'amy', null, 'F', 1);

SQL> insert into student_ning4
    Values(102, 'river', 'river@sina.com', 'F', 9);
-- 新增数据，不存在 9 这个专业
-- 提示错误：错误码：ORA-02291
--中文提示：未找到父项关键字
--英文提示：parent key not found
--下例是英文提示：
    ORA-02291: integrity constraint (NINGLJ.STU_N4_MID_FK)
    violated - parent key not found

```



```

运行 SQL 命令行
SQL> insert into student_ning4
  2 values(101,'amy',null,'F',1);

已创建 1 行。

SQL> insert into student_ning4
  2 values(102,'river','river@sina.com','F', 9);
insert into student_ning4
*
第 1 行出现错误:
ORA-02291: 违反完整约束条件 (SYS.STU_N4_MID_FK) - 未找到父项

SQL>

```

#### 【案例 11】外键约束演示 2\_insert null

```

SQL> insert into student_ning4
  Values(102, 'river', 'river@sina.com', 'F', null);
-- 外键列可以插入空值 null

```

```

运行 SQL 命令行
SQL> insert into student_ning4
  2 values(102,'river','river@sina.com','F', null);

已创建 1 行。

SQL>

```

#### 【案例 12】外键约束演示 3\_delete 删除主表中的数据

```

SQL> delete from major_ning where id = 1;
-- 删除专业表( Major )中 id=1 的数据
-- 提示错误: 错误码: ORA-02292
-- 中文提示: 已找到子记录
-- 英文提示: child record found
-- 因为学生表( Student )中某条数据的“专业 ID”
  和专业表( Major )中的“编号”关联

```

```

运行 SQL 命令行
SQL> delete from major_ning where id = 1;
delete from major_ning where id = 1
*
第 1 行出现错误:
ORA-02292: 违反完整约束条件 (SYS.STU_N4_MID_FK) - 已找到子记

SQL>

```

### 【案例 13】外键约束演示 4\_on delete set null

on delete set null 的作用：一旦主表数据被删除，从表关联数据置为 null

```

--删除原约束条件
SQL> alter table student_ning4
      drop constraint stu_n4_mid_fk;
--重建约束条件
SQL>> alter table student_ning4
      add constraint stu_n4_mid_fk
      foreign key (majorid) references major_ning(id) on delete set null;

SQL> insert into student_ning4
      values(104,'rory',null,'M',1); --rory 是 1 专业的学生

SQL> delete from major_ning where id = 1; --删除编码为 1 的专业

SQL> select * from student_ning4; --amy 的专业被设置为 NULL

```

SQL> Telnet 192.168.0.26

```
SQL> alter table student_ning4
2 drop constraint stu_n4_mid_fk;
```

Table altered.

```
SQL> alter table student_ning4
2 add constraint stu_n4_mid_fk
3 foreign key (majorid) references major_ning(id)
4 on delete set null;
```

Table altered.

```
SQL> insert into student_ning4
2 values(104,'rory',null,'M',1);
```

1 row created.

```
SQL> delete from major_ning where id = 1;
```

1 row deleted.

```
SQL> select * from student_ning4;
```

SID	NAME	EMAIL	G	MAJORID
101	amy		F	
104	rory		M	

#### 【案例 14】外键约束演示 4\_on delete cascade

on delete cascade 的作用：级联删除( 主表数据删除，从表相关联数据统统删除 )

--删除原约束条件

```
SQL> alter table student_ning4
drop constraint stu_n4_mid_fk;
```

--重建约束条件

```
SQL>> alter table student_ning4
add constraint stu_n4_mid_fk
foreign key (majorid) references major_ning(id) on delete cascade;
```

```
SQL> insert int student_ning4
      values(104, 'amy', null, 'F',3); -- amy 在 3 部门

SQL> delete from major_ning where id = 3; -- 删除 3 部门

SQL> select * from student_ning4; -- amy 已被删除
```

## 2.6. 复制表，不复制约束条件

**注意：**复制表时，不复制约束条件

create table 表名 as 查询语句

## 2.7. 建立约束条件的时机 \*

### 2.7.1. 建表同时建立约束条件

**【案例 15】建表同时建立约束条件演示，可以建立在表级或列级，可以给名字或由系统自动命名**

```
SQL> create table student_ning5(
      id number(3) ,
      name char(20) not null ,
      email char(50) unique,
      majorid number(2) ,
      constraint stu_id_pk primary key(id) ,
      constraint stu_mid_fk foreign key(majorid) references major(id)
);
```

**【案例 16】在创建完表以后创建约束**

```
SQL> create table student_ning6(
      id number(3),
      name char(20) not null,
      email char(50),
      gender char(1),
      majorid number(2)
);
```

```
SQL> alter table student_ning6
      add constraint stu_n6_id_pk primary key(id);
SQL> alter table student_ning6
      add constraint stu_n6_mid_fk foreign key(majorid)
      references major_ning(id);
SQL> alter table student_ning6
      add constraint stu_n6_email_uk unique(email);
SQL> alter table student_ning6
      add constraint stu_n6_g_ck check(gender in ('M', 'F'));
```

**建议：**在建表之后创建约束条件，结构清晰，可读性好，方便操作约束条件。

项目建立数据库初始环境脚本文件的示例：

#### 【案例 17】Drop 表&&Create 新表的脚本

```
-----脚本文件 begin-----

alter table student drop constraint stu_mid_fk ;    -- 删除参照表约束
drop table student ;                               -- 删除参照表( 从表 )
drop table major ;                                 -- 删除被参照表( 主表 )

create table major(...) ;                          -- 创建被参照表( 主表 )
create table student(...) ;                        -- 创建参照表( 从表 )
alter table student add constraint ....             -- 增加约束

-----脚本文件 end-----
```

✓ 此脚本文件可以直接在数据库 sqlplus 中运行，用于初始化数据环境。

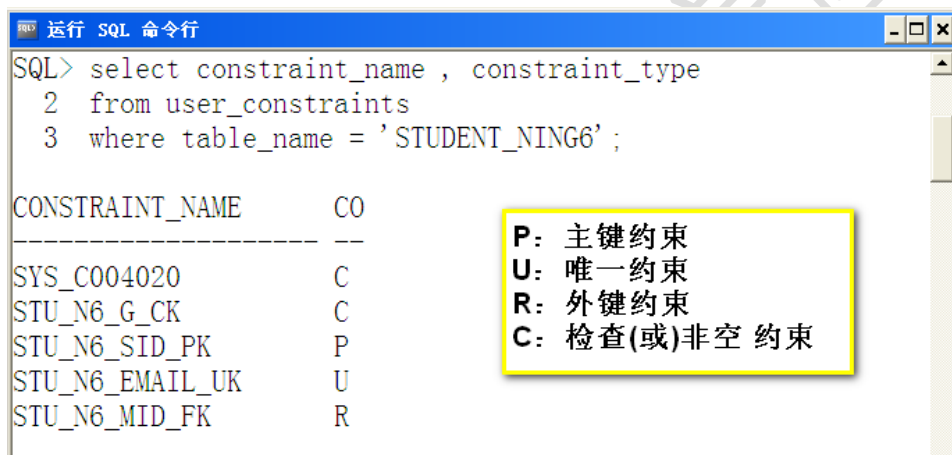
## 2.8. 数据字典 \*

- |                     |                                |
|---------------------|--------------------------------|
| 1) user_tables      | 用户所有的数据表                       |
| 2) user_constraints | 用户所有的约束条件                      |
| 3) user_objects     | 用户所有的对象( 表、视图、索引等 )            |
| 4) all_tables       | 用户能访问的数据表<br>包括自己的和别的用户允许自己访问的 |

- 5) all\_constraints                      用户能访问的约束条件
- 6) all\_objects                        用户能访问的对象( 表、视图、索引等 )
- 7) 数据字典的格式如 :  
     User\_XXX: 用户自己的对象  
     All\_XXX : 用户能访问的对象  
     Dba\_XXX : 数据库所有的对象

### 【案例 18】查询 student\_ning6 表的约束条件名和类型

```
SQL> select constraint_name , constraint_type
      from user_constraints
      where table_name = 'STUDENT_NING6';
```



CONSTRAINT_NAME	CO
SYS_C004020	C
STU_N6_G_CK	C
STU_N6_SID_PK	P
STU_N6_EMAIL_UK	U
STU_N6_MID_FK	R

**P: 主键约束**  
**U: 唯一约束**  
**R: 外键约束**  
**C: 检查(或)非空 约束**

## 3. 数据库的其他对象

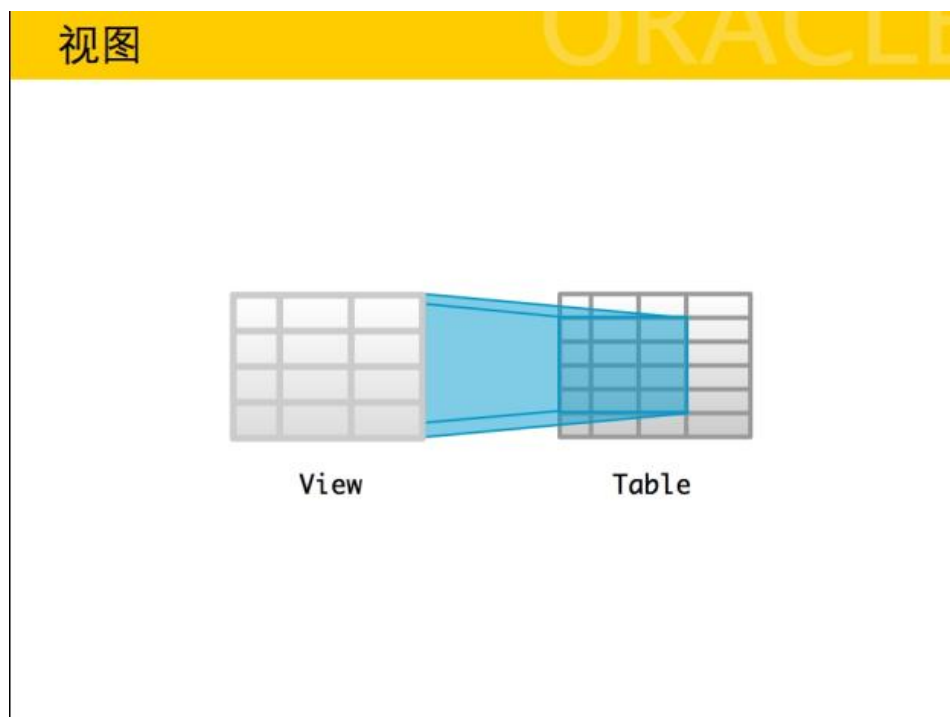
### 3.1. 数据库中的主要对象 \*

- 1) 表 Table( 掌握 )
  - ✓ 表是数据库存储的基本单元，在关系型数据库中，表是一个二维结构，由行( Row )和列( Record )组成
- 2) 视图 View( 掌握 )
  - ✓ 视图也被称为虚表( 虚拟的表 )，视图对应于一条 Select 语句，这条查询语句得到的结果集 被赋予一个名字，即视图的名字，此后可以像操作表一样操作这个视图。
  - ✓ 使用视图的主要目的是简化复杂查询。
- 3) 索引 Index( 掌握 )

- ✓ 索引是用来在数据库中加速表查询的数据库对象，通过快速路径访问方式快速定位数据，可有效较少磁盘 I/O 操作，提高访问性能。
- 4) 序列 Sequence( 掌握 )
  - ✓ 序列( Sequence )是一种用来生成唯一数字值的数据库对象。序列的值由 Oracle 程序按递增或递减顺序自动生成，通常用来自动生成表的主键值，是一种高效率获得唯一键值的途径。
- 5) 存储过程 Procedure( 了解 )
  - ✓ 过程( Procedure )是一种 PL/SQL 存储程序单元，主要用于在数据库中完成特定的操作或者任务，如果在程序中经常需要执行某个操作，可以基于这些操作建立一个过程，用于简化客户端的开发和维护，以及提高执行性能。
- 6) 函数 Function( 了解 )
  - ✓ PL/SQL 中的函数被用来执行复杂的计算，并返回计算结果。
- 7) 包 Package( 了解 )
  - ✓ 包是一种比较特殊的 PL/SQL 程序，它并不是一个 PL/SQL 存储程序块，而是用于将相关的存储过程和函数组织起来，组成 PL/SQL 存储程序组。
  - ✓ 包由两个独立部分组成：包头和包体。
- 8) 触发器 Trigger( 了解 )
  - ✓ PL/SQL 程序中的触发器的结构类似于函数和过程，与函数和过程不同，触发器是在事件发生时隐式地运行的。
  - ✓ 相当于 Java 语言中的事件监听器
- 9) 同义词 Synonym( 了解 )
  - ✓ 同义词是数据库对象的别名，目的是简化 SQL 查询，以及隐藏数据库对象的信息。

### 3.2. 视图 View \*

- 1) 视图的使用和表相同
- 2) 视图的好处：简化查询 ;隐藏数据表的列
- 3) 视图不包含任何数据。是基表数据的投影



### 【案例 19】创建视图, 视图的定义是一个数据表的子集

```
SQL> create view v_emp_ning
      as
      select empno, ename, job from emp_ning
      where deptno = 20 ;

      --视图的使用方式和表相同
SQL> desc v_emp_ning
SQL> select * from v_emp_ning;
      --视图的好处：简化查询 ;隐藏数据表的列
```

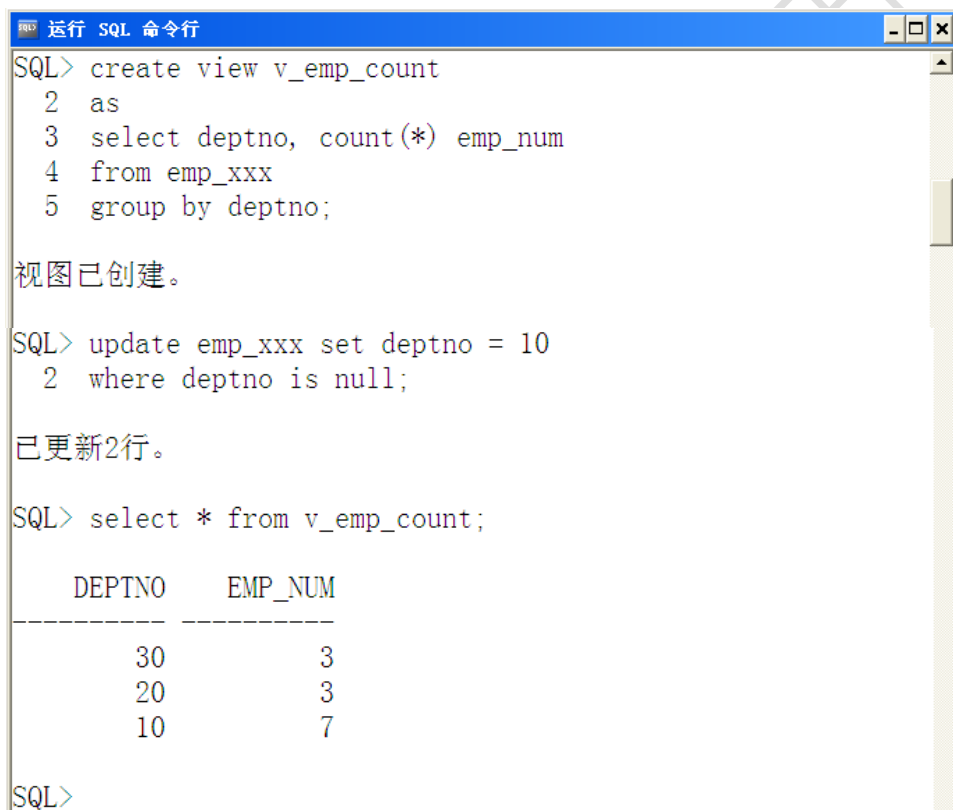
### 【案例 20】创建视图 , 视图的定义是一个复杂查询语句

```
SQL> create view v_emp_count
      as
      select deptno, count(*) emp_num
      from emp_ning
      group by deptno ;
```



```
SQL> update emp_ning set deptno = 10
      where deptno is null ;
--修改基表数据

SQL> select * from v_emp_count ;
--视图查询到的是修改后的数据
--视图不包含任何数据，是基表数据的投影
```



```
运行 SQL 命令行
SQL> create view v_emp_count
2 as
3 select deptno, count(*) emp_num
4 from emp_xxx
5 group by deptno;

视图已创建。

SQL> update emp_xxx set deptno = 10
2 where deptno is null;

已更新2行。

SQL> select * from v_emp_count;

  DEPTNO    EMP_NUM
-----
      30         3
      20         3
      10         7

SQL>
```

**【案例 21】** 视图可以使用 **create or replace** 来创建或覆盖，并可查询视图的定义。

```
SQL> create or replace view v_emp_count
      as
      select deptno, count(*) emp_num ,
      sum(salary) sum_s ,
      avg(nvl(salary,0)) avg_s ,
      max(salary) max_s ,
```

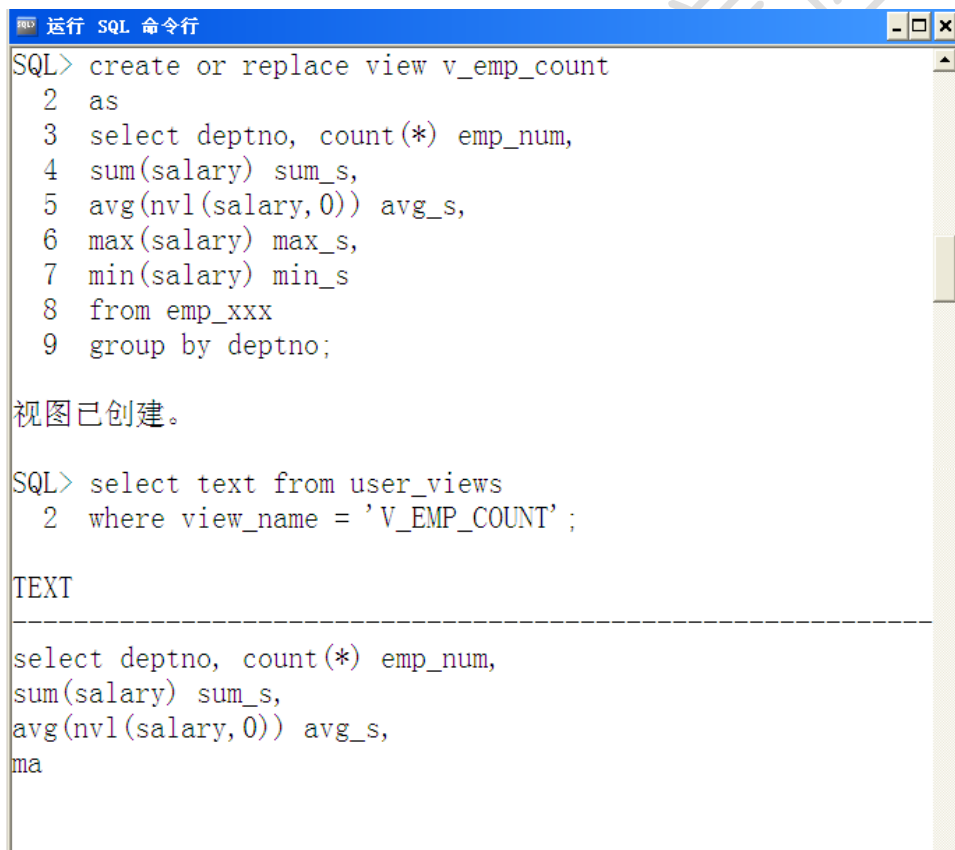
```
min(salary) min_s
from emp_ning
group by deptno ;
```

--查询视图的定义

```
SQL> select text from user_views
      where view_name = 'V_EMP_COUNT' ;
```

--如果视图对应的 sql 语句显示不全

```
SQL> set long 1000
```



The screenshot shows a SQL command window titled "运行 SQL 命令行". The user enters the following commands:

```
SQL> create or replace view v_emp_count
2 as
3 select deptno, count(*) emp_num,
4 sum(salary) sum_s,
5 avg(nvl(salary,0)) avg_s,
6 max(salary) max_s,
7 min(salary) min_s
8 from emp_xxx
9 group by deptno;
```

The response is: 视图已创建。

```
SQL> select text from user_views
      2 where view_name = 'V_EMP_COUNT' ;
```

The response is: TEXT

---

```
select deptno, count(*) emp_num,
sum(salary) sum_s,
avg(nvl(salary,0)) avg_s,
ma
```

```
SQL> set long 1000
SQL> select text from user_views
2 where view_name = 'V_EMP_COUNT' ;
```

TEXT

```
-----
select deptno, count(*) emp_num,
sum(salary) sum_s,
avg(nvl(salary,0)) avg_s,
max(salary) max_s,
min(salary) min_s
from emp_xxx
group by deptno
```

```
SQL>
```

### 3.3. 索引 Index \*

index : 用来提高查询效率的机制

- ✓ 全表扫描方式( Full Table Scan ) : 查询效率极低
- ✓ 索引查询 : 比全表扫描快
- ✓ 索引的结构 : 数据+地址( 如 : 张三+Room203 )
- ✓ **注意** : 对于数据变更频繁(DML 操作频繁)的表 , 索引会影响查询性能
- ✓ 自动创建索引 :  
如果数据表有 PK/Unique 两种约束 , 索引自动创建 , 除此以外 , 索引必须手动创建
- ✓ 自定义索引语法 :  
create index 索引名 on 表名(列名) ;

#### 【案例 22】表的主键和唯一约束条件 , 会自动创建索引

```
SQL> create table student_ning7(
    id number(4),
    name char(20),
    email char(40),
    constraint stu_n7_id_pk primary key(id),
    constraint stu_n7_email_uk unique(email)
);
--注意查询时表名大写
SQL> select constraint_name
```

```

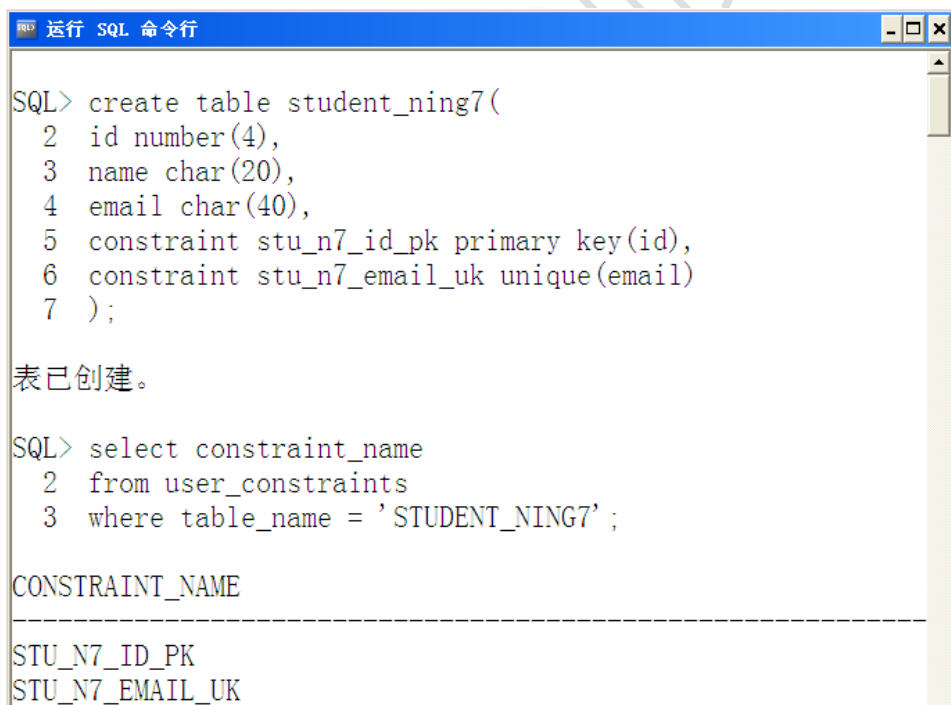
from user_constraints
where table_name = 'STUDENT_NING7';

-- 查询 student_ning7 表上的索引
-- 和主键/唯一约束条件同名，数据库自动创建的索引
SQL> select index_name from user_indexes
      where table_name = 'STUDENT_NING7';

--凡是有约束条件的字段(如 id 或 email)上的查询，会使用索引
SQL> select * from student_ning7 where id = 1001;

--这种查询用不到索引：全表扫描
SQL> select * from student_ning7
      where name = 'zhangsan';

```



```

运行 SQL 命令行

SQL> create table student_ning7(
  2  id number(4),
  3  name char(20),
  4  email char(40),
  5  constraint stu_n7_id_pk primary key(id),
  6  constraint stu_n7_email_uk unique(email)
  7 );

表已创建。

SQL> select constraint_name
  2  from user_constraints
  3  where table_name = 'STUDENT_NING7';

CONSTRAINT_NAME
-----
STU_N7_ID_PK
STU_N7_EMAIL_UK

```

```
SQL> select index_name from user_indexes
2  where table_name = 'STUDENT_NING7';
```

```
INDEX_NAME
```

```
-----
```

```
STU_N7_ID_PK
```

```
STU_N7_EMAIL_UK
```

```
SQL>
```

**【案例 23】**如果经常在名字上做查询，名字字段没有唯一约束，这时可以创建基于名字字段的索引，索引名自定义。

```
SQL> create index idx_stu7_name
      on student_ning7(name);
```

运行 SQL 命令行

```
SQL> create index idx_stu7_name
2  on student_ning7(name);
```

索引已创建。

```
SQL>
```

### 3.4. 序列 Sequence \*\*

序列的特性：产生连续的不同的数字值用来作为数据表的主键。

- ✓ 序列是数据库中的独立对象
- ✓ 表可以用序列产生的值作为主键，也可以不用
- ✓ 序列可以为一个或多个表产生主键，也可以不用

**建议：**一个序列为一个表产生主键

- ✓ 序列这种对象在 Oracle、db2 等数据库中有，在 mysql、sql server 中没有。

#### 【案例 24】创建序列

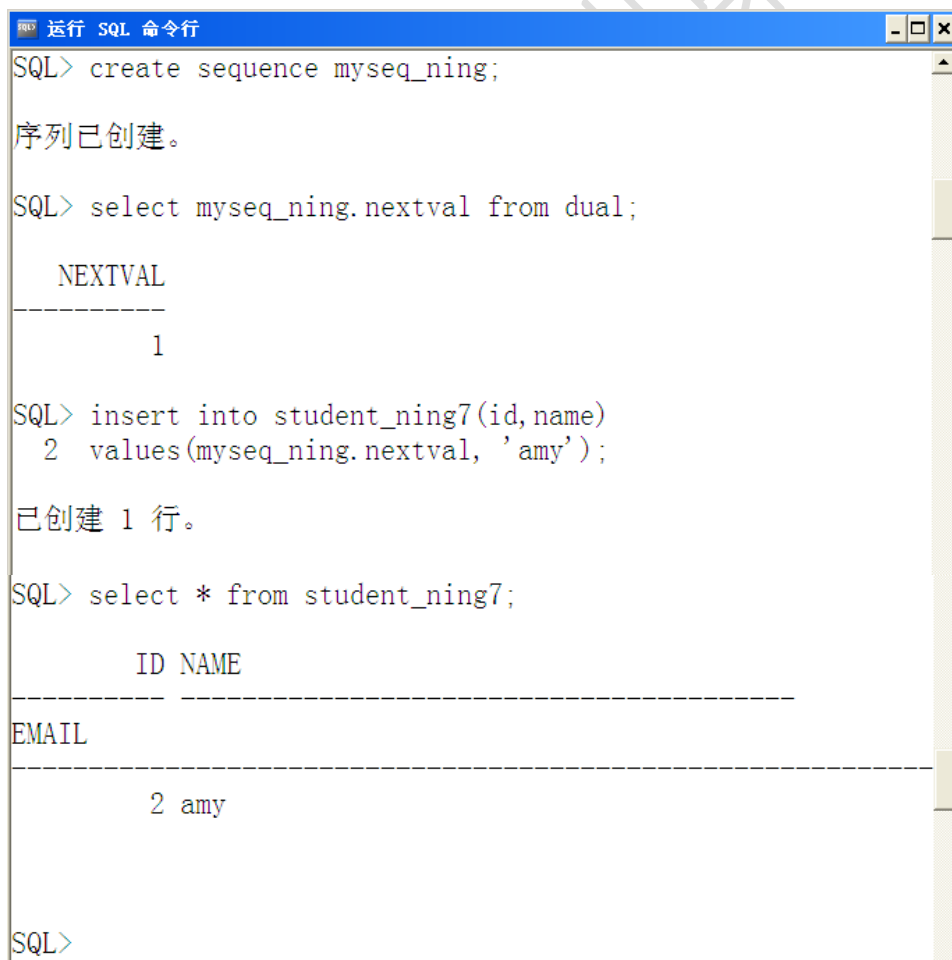
```
-- 产生从 1 开始的数字值，步进是 1
SQL> create sequence myseq_ning;

-- 查看序列产生的值
SQL> select myseq_ning.nextval from dual;

-- 使用序列产生的值作为表的主键值
SQL> insert into student_ning7(id,name)
      values(myseq_ning.nextval, 'amy');

SQL> select * from student_ning7;

-- 显示结果为 2 amy
-- 注意：每调用一次 myseq_ning.nextval 就会获得 1 个递增的数
```



运行 SQL 命令行

```
SQL> create sequence myseq_ning;

序列已创建。

SQL> select myseq_ning.nextval from dual;

      NEXTVAL
-----
           1

SQL> insert into student_ning7(id,name)
      2  values(myseq_ning.nextval, 'amy');

已创建 1 行。

SQL> select * from student_ning7;

      ID NAME
-----
      2 amy

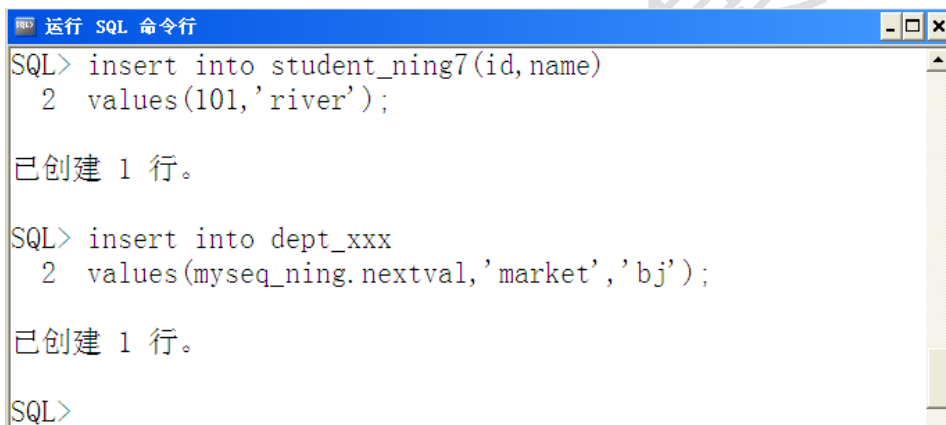
EMAIL

SQL>
```

**【案例 25】同一序列可以为一个或多个表产生主键( 序列和表是独立的对象 )**

```
-- student_ning7 和序列 myseq_ning 的关系 :
-- 是数据库中的独立对象

-- 表可以用序列产生的值作为主键 , 也可以不用
-- 序列可以为一个或多个表产生主键 , 也可以不用
SQL> insert into student_ning7(id,name) values(100,'river') ;
SQL> insert into dept_ning values(myseq_ning.nextval,'market','bj') ;
-- 建议 : 一个序列只为一个表产生主键
```



```
运行 SQL 命令行
SQL> insert into student_ning7(id, name)
2 values(101,'river');

已创建 1 行。

SQL> insert into dept_xxx
2 values(myseq_ning.nextval,'market','bj');

已创建 1 行。

SQL>
```

**【案例 26】建立序列 , 主键值从 1000 开始 , 步进是 2**

```
SQL> create sequence myseq_ning1
      start with 1000
      increment by 2 ;

SQL> insert into student_ning7
      values(myseq_ning1.nextval , 'song' , null) ;
```

```

运行 SQL 命令行
SQL> create sequence myseq_ning1
  2 start with 1000
  3 increment by 2;

序列已创建。

SQL> insert into student_ning7
  2 values(myseq_ning1.nextval, 'song', null);

已创建 1 行。

SQL> select *from student_ning7;

      ID NAME
-----
EMAIL
-----
      2 amy

      100 river
  
```

**【案例 27】删除序列( 对曾经产生过的数据没有任何影响 )**

```

SQL> drop sequence myseq_ning1 ;
SQL> select * from student_ning7;
  
```

```

运行 SQL 命令行
SQL> drop sequence myseq_ning1;

序列已删除。

SQL>
SQL>
  
```

**学习建议**

3) **Oracle 入门简单，深入较难**

4) **学习过程：**

SQL --> PL/SQL( 过程/函数 ) --> Oracle 体系结构 --> 数据库调优



### 考试认证

- 1) **java 方向的认证考试：**  
SCJP(Sun Certificated Java Programmer) , 现在叫 OCJP
- 2) **Oracle 数据库方向的认证考试：**  
OCP(Oracle Certified Professional)

**【Oracle 课程结束】**