学习SQL

一、SQL 简介

> SQL,全称为Structured Query Language，结构化查询语言，是用于访问和处理数据库的标准计算机语言。SQL 对大小写不敏感且语句后需要加分号。
>
> SQL主要包括一下几个部分：
>
> - 数据定义语言（DDL）：主要用于创建(CREATE)、修改(ALTER)、删除(DROP)数据库对象。
> - 数据查询语言（DQL）：主要用于查询(SELECT)数据库中的数据、包括FROM、WHERE、GROUPBY、HAVING和WITH五个子句。
> - 数据操作语言（DML）:主要用于插入(INSERT)、修改(UPDATE)、删除(DELETE)数据表中数据。
> - 数据控制语言（DCL）：主要用于授予(GRANT)、回收(REMOVE)用户权限。
> - 事务控制语言（ACL）：主要用于保证数据库中的数据一致性，提交(COMMIT)和回滚(ROLLBACK)

二、数据库管理

2.1 创建数据库

```
CREATE DATABASE NJITDB;
```

2.2 删除数据库

> DROP DATABASE TESTDB;

三、用户管理

3.1 创建用户

> 创建用户XIAOWANG 密码:HAHA
>
> *CREATE USER XIAOWANG IDENTIFIED BY HAHA;*

3.2 删除用户

> 删除XIAOWANG用户
>
> *DROP USER XIAOWANG;*

3.3 授予权限

> 授予XIAOWANG 角色CONNECT和RESOURCE
>
> GRANT SESSION,CONNECT,RESOURCE TO XIAOWANG;
>
> GRANT SELECT ON SCOTT.emp TO XIAOWANG;

3.4 回收权限

> 撤回 用户XIAOWANG的相关权限
>
> *REVOKE SELECT ON SCOTT.EMP FROM XIAOWANG;*
>
> *REVOKE CREATE SESSION , CONNECT , RESOURCE FROM XIAOWANG;*

四、数据表管理

4.1 创建数据表

语法:

```
CREATE TABLE table_name
(
    column_name1    datatype1   [constraint_condition1][,
    column_name2    datatype2   [constraint_condition2]]...
);
```

例子 - ORACLE

```
CREATE TABLE PDAHC.T_STUDENTS
(
  ID      VARCHAR2(10 BYTE),
  NAME    VARCHAR2(10 BYTE),
  AGE     INTEGER,
  SEX     CHAR(2 BYTE),
  BIRTH   DATE
);

```

例子 - MYSQL

```
-- 学生信息表
CREATE TABLE T_STUDENTS
(
  stuID  VARCHAR(10) PRIMARY KEY,
  NAME   VARCHAR(10) NOT NULL,
  AGE    INT NOT NULL,
  SEX    CHAR(2) NOT NULL,
  BIRTH  DATE NOT NULL
);

-- 学生成绩表
CREATE TABLE T_RESULTS
(
    stuID VARCHAR(15),
    curID VARCHAR(15),
    result DOUBLE,
    PRIMARY KEY(stuID, curID),
    FOREIGN KEY(stuID) REFERENCES T_STUDENTS(stuID) ON DELETE CASCADE
);

-- 课程信息表
CREATE TABLE T_CURRICULUM
```

```
(
    curID       VARCHAR(15) PRIMARY KEY,
    curNAME     VARCHAR(15),
    CREDIT      INT,
    LEARNTIME   INT,
    TEACHERNAME VARCHAR(15)
);

-- 教师信息表
CREATE TABLE T_TEACHERS
(
    teaID   VARCHAR(15) PRIMARY KEY,
    teaNAME VARCHAR(15),
    AGE     INT,
    SEX     CHAR(2),
    depID   VARCHAR(15),
    depNAME VARCHAR(15),
    PROFESSION  VARCHAR(15),
    SALARY      INT(5),
    PENSION     DOUBLE
);

-- 院系信息表
CREATE TABLE T_DEPTS
(
    deptID      VARCHAR(15),
    deptName    VARCHAR(15)
);

```

4.1.1 使用约束

Constraint	约束	数量限制	作用
NOT NULL	非空约束	允许多个	保证列值的非空性
UNIQUE	唯一约束	允许多个	保证列值的唯一性，可为空值
PRIMARY KEY	主键约束	只有一个	保证列值的唯一性，禁止空值
FOREIGN KEY	外键约束		保证关联表的完整性
CHECK	检查约束	允许多个	限制列值的取值范围或者取值条件

4.2 修改数据表

增加一列

```
//语法
ALTER TABLE table_name ADD(column_name datatype [constraingt_conditon]);

//MYSQL 向教师信息表中增加一个地址信息的列
mysql> ALTER TABLE T_TEACHERS ADD ADDRESS VARCHAR(50);
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
```

删除一列

```
//语法
ALTER TABLE table_name DROP column_name;

//MYSQL 从教师信息表中删除一个地址信息的列
mysql> ALTER TABLE T_TEACHERS DROP ADDRESS;
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
```

修改一列

```
//语法
ALTER TABLE table_name MODIFY column_name datatype;

//MYSQL 从教师信息表中删除一个地址信息的列
mysql> ALTER TABLE T_TEACHERS DROP ADDRESS;
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
```

增加一个约束条件

```
//语法
ALTER TABLE table_name ADD constraint_type(column_name);

//MYSQL 对教师信息表中AGE列增加一个CHECK约束 在15到60之间
mysql> ALTER TABLE T_TEACHERS ADD CHECK(AGE BETWEEN 15 AND 60);
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
```

删除一个约束条件

```
//语法
ALTER TABLE table_name DROP constraint_type;

//MYSQL 对教师信息表中删除CHECK约束
mysql> ALTER TABLE T_TEACHERS DROP CHECK;
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
```

增加一个索引

```
//语法
ALTER TABLE table_name ADD INDEX index_name(column_name1[,column_name2]...);

//MYSQL 从教师信息表中增加一个索引
mysql> ALTER TABLE T_TEACHERS ADD INDEX i_age(teaID);
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
```

删除一个索引

```
//语法
ALTER TABLE table_name DROP INDEX index_name;

//MYSQL 从教师信息表中增加一个索引
mysql> ALTER TABLE T_TEACHERS ADD INDEX i_age(teaID);
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
```

4.3 删除数据表

SQL语法： DROP TABLE 表名 [constraint_condition]

4.1.2 使用索引

索引	作用	使用
唯一索引	保证列值的唯一性	
主索引	保证列值的唯一性	
单列索引	提高查询效率	定义在单个数据列上的索引，经常以某个列为查询条件
复合索引	提高查询效率	定义在多个数据列上的索引，经常以某些列为查询条件
聚簇索引	提高查询效率	

```
CREATE INDEX 索引名 ON 表名（列名1 , [列名2]...）

DROP  INDEX 索引名 ON 表名（列名1 , [列名2]...）
```

五、数据查询

5.1 基本查询操作

```
---- 1、查询表中全部列的记录
-- 语法
SELECT * FROM 表名或视图名;

-- MYSQL
mysql> SELECT * FROM T_STUDENTS;
+---------+------+-----+-----+------------+
| stuID   | NAME | AGE | SEX | BIRTH      |
+---------+------+-----+-----+------------+
| S000001 | 赵大 |  25 | 男  | 1995-04-12 |
| S000002 | 钱二 |  21 | 男  | 1999-05-13 |
| S000003 | 孙三 |  24 | 女  | 1996-08-06 |
| S000004 | 李四 |  24 | 男  | 1996-05-23 |
| S000005 | 周五 |  20 | 男  | 2000-05-03 |
+---------+------+-----+-----+------------+
5 rows in set


---- 2、查询表中指定列的记录
-- 语法
SELECT 目标列[,目标列,..] FROM 表名或视图名[,表名或视图名];
```

```
-- MYSQL
mysql> SELECT stuID,NAME FROM T_STUDENTS;
+---------+------+
| stuID   | NAME |
+---------+------+
| S000001 | 赵大 |
| S000002 | 钱二 |
| S000003 | 孙三 |
| S000004 | 李四 |
| S000005 | 周五 |
+---------+------+
5 rows in set

---- 3、查询表中不重复列的记录
-- 语法
SELECT DISTINCT
目标列[,目标列,..] FROM 表名或视图名[,表名或视图名];

-- MYSQL
mysql> SELECT NAME FROM T_STUDENTS;
+------+
| NAME |
+------+
| 赵大 |
| 钱二 |
| 孙三 |
| 李四 |
| 周五 |
| 赵大 |
+------+
6 rows in set

---- 4、使用列别名查询
-- 语法
SELECT 目标列 [AS] 列别名 [,目标列 [AS] 列别名 ,..] FROM 表名或视图名[,表名或视图
名];

-- MYSQL
mysql> SELECT stuID AS 学号 , NAME AS 姓名 , AGE AS 年龄 , SEX 性别 , BIRTH 出
生日期 FROM T_STUDENTS;
+---------+------+------+------+------------+
| 学号    | 姓名 | 年龄 | 性别 | 出生日期   |
+---------+------+------+------+------------+
| S000001 | 赵大 |   25 | 男   | 1995-04-12 |
| S000002 | 钱二 |   21 | 男   | 1999-05-13 |
| S000003 | 孙三 |   24 | 女   | 1996-08-06 |
| S000004 | 李四 |   24 | 男   | 1996-05-23 |
| S000005 | 周五 |   20 | 男   | 2000-05-03 |
| S000006 | 赵大 |   25 | 男   | 1995-04-12 |
+---------+------+------+------+------------+
6 rows in set

---- 5、连接字段查询
-- 语法
SELECT 目标列 [AS] 列别名 [,目标列 [AS] 列别名 ,..] FROM 表名或视图名[,表名或视图
名];

```

```
-- MYSQL 使用CONCAT函数
SELECT CONCAT(CONCAT(stuID,","),CONCAT(NAME,","), CONCAT(SEX,","),
CONCAT(BIRTH,","),AGE) STUDENT_INFO FROM T_STUDENTS;
+------------------------------+
| STUDENT_INFO                 |
+------------------------------+
| S000001,赵大,男,1995-04-12,25 |
| S000002,钱二,男,1999-05-13,21 |
| S000003,孙三,女,1996-08-06,24 |
| S000004,李四,男,1996-05-23,24 |
| S000005,周五,男,2000-05-03,20 |
| S000006,赵大,男,1995-04-12,25 |
+------------------------------+
6 rows in set
```

5.2 WHERE条件查询

```
---- 1、比较查询
--- 算术比较运算符 > >= = < <= != <> !> !<
-- 语法
WHERE 字段 算术比较运算符 值
-- MYSQL
mysql> SELECT * FROM T_STUDENTS WHERE AGE > 22;
+---------+------+-----+-----+------------+
| stuID   | NAME | AGE | SEX | BIRTH      |
+---------+------+-----+-----+------------+
| S000001 | 赵大 |  25 | 男  | 1995-04-12 |
| S000003 | 孙三 |  24 | 女  | 1996-08-06 |
| S000004 | 李四 |  24 | 男  | 1996-05-23 |
| S000006 | 赵大 |  25 | 男  | 1995-04-12 |
+---------+------+-----+-----+------------+
4 rows in set

--- 范围查询 BETWEEN..AND...  IN
-- 语法
WHERE 字段 BETWEEN 值1 AND 值2

WHERE 字段 IN (值1,值2,...)
-- MYSQL
mysql> SELECT * FROM T_STUDENTS WHERE AGE BETWEEN 20 AND 24;
+---------+------+-----+-----+------------+
| stuID   | NAME | AGE | SEX | BIRTH      |
+---------+------+-----+-----+------------+
| S000002 | 钱二 |  21 | 男  | 1999-05-13 |
| S000003 | 孙三 |  24 | 女  | 1996-08-06 |
| S000004 | 李四 |  24 | 男  | 1996-05-23 |
| S000005 | 周五 |  20 | 男  | 2000-05-03 |
+---------+------+-----+-----+------------+
4 rows in set

mysql> SELECT * FROM T_STUDENTS WHERE AGE BETWEEN 20 AND 24;
+---------+------+-----+-----+------------+
| stuID   | NAME | AGE | SEX | BIRTH      |
+---------+------+-----+-----+------------+
| S000002 | 钱二 |  21 | 男  | 1999-05-13 |
| S000003 | 孙三 |  24 | 女  | 1996-08-06 |
| S000004 | 李四 |  24 | 男  | 1996-05-23 |
```

```
| S000005 | 周五 |  20 | 男   | 2000-05-03 |
+---------+------+-----+------+------------+
4 rows in set

---- 2、逻辑查询 AND OR NOT
-- 语法
WHERE 条件1 逻辑运算符 条件2 ...
-- MYSQL
SELECT * FROM T_STUDENTS WHERE BIRTH > '1996-00-00' AND AGE > 21;
+---------+------+-----+------+------------+
| stuID   | NAME | AGE | SEX | BIRTH      |
+---------+------+-----+------+------------+
| S000003 | 孙三 |  24 | 女   | 1996-08-06 |
| S000004 | 李四 |  24 | 男   | 1996-05-23 |
+---------+------+-----+------+------------+

---- 3、模糊查询 LIKE REGEXP
通配符:
    匹配单个字符    : '_'
    匹配0或多个字符  : '%'
-- 语法
WHERE 字段 LIKE 模糊值
-- MYSQL
mysql> SELECT * FROM T_STUDENTS WHERE NAME LIKE '%三';
+---------+------+-----+------+------------+
| stuID   | NAME | AGE | SEX | BIRTH      |
+---------+------+-----+------+------------+
| S000003 | 孙三 |  24 | 女   | 1996-08-06 |
| S000007 | 吴三 |  25 | 男   | 1995-02-22 |
+---------+------+-----+------+------------+
2 rows in set

mysql> SELECT * FROM T_STUDENTS WHERE NAME LIKE '_三';
+---------+------+-----+------+------------+
| stuID   | NAME | AGE | SEX | BIRTH      |
+---------+------+-----+------+------------+
| S000003 | 孙三 |  24 | 女   | 1996-08-06 |
| S000007 | 吴三 |  25 | 男   | 1995-02-22 |
+---------+------+-----+------+------------+
2 rows in set

-- 转移字符 ESCAPE定义转义符
mysql> SELECT * FROM T_STUDENTS WHERE NAME LIKE '%$_%' ESCAPE '$';
+---------+------------+-----+------+------------+
| stuID   | NAME       | AGE | SEX | BIRTH      |
+---------+------------+-----+------+------------+
| S000008 | JERMAN_RAY |  23 | 女   | 1997-05-22 |
+---------+------------+-----+------+------------+
1 row in set

-- MYSQL REGEXP关键字 正则表达式 进行模糊匹配
mysql>  SELECT * FROM T_STUDENTS WHERE NAME REGEXP '[_]+' ;
+---------+------------+-----+------+------------+
| stuID   | NAME       | AGE | SEX | BIRTH      |
+---------+------------+-----+------+------------+
| S000008 | JERMAN_RAY |  23 | 女   | 1997-05-22 |
+---------+------------+-----+------+------------+
1 row in set
```

5.3 排序与分组查询

```
---- 常见的聚合函数：
聚合函数又称为分组函数或者统计函数，主要用于对得到的一组数据进行统计计算。常用的聚合函数由COUNT,MAX,MIN,SUM,AVG
-- MYSQL
mysql> SELECT COUNT(AGE),MAX(AGE),MIN(AGE),AVG(AGE),SUM(AGE) FROM T_STUDENTS;
+------------+----------+----------+----------+----------+
| COUNT(AGE) | MAX(AGE) | MIN(AGE) | AVG(AGE) | SUM(AGE) |
+------------+----------+----------+----------+----------+
|          8 |       25 |       20 | 23.3750  | 187      |
+------------+----------+----------+----------+----------+
1 row in set

---- ORDER BY排序
-- 1、ASC 升序[默认排序方式] DESC 降序  2、必须为最后一个子句
-- 语法
ORDER BY 列1 [,列2,...] [ASC|DESC]
-- MYSQL
mysql> SELECT * FROM T_STUDENTS ORDER BY AGE DESC;
+---------+------------+-----+-----+------------+
| stuID   | NAME       | AGE | SEX | BIRTH      |
+---------+------------+-----+-----+------------+
| S000001 | 赵大       |  25 | 男  | 1995-04-12 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 |
+---------+------------+-----+-----+------------+
8 rows in set

---- GRUOP BY分组
-- 语法
GRUOP BY 列1 [,列2,...] [HAVING 条件表示式]
-- MYSQL
mysql> SELECT SEX ,COUNT(stuID) FROM T_STUDENTS GROUP BY SEX;
+-----+--------------+
| SEX | COUNT(stuID) |
+-----+--------------+
| 女  |            2 |
| 男  |            6 |
+-----+--------------+
2 rows in set

mysql> SELECT AGE ,COUNT(stuID) FROM T_STUDENTS GROUP BY AGE;
+-----+--------------+
| AGE | COUNT(stuID) |
+-----+--------------+
|  20 |            1 |
|  21 |            1 |
|  23 |            1 |
|  24 |            2 |
```

```
|  25 |            3 |
+-----+--------------+
5 rows in set

mysql> SELECT AGE ,COUNT(stuID),SEX,COUNT(stuID) FROM T_STUDENTS GROUP BY
AGE,SEX;
+-----+--------------+-----+--------------+
| AGE | COUNT(stuID) | SEX | COUNT(stuID) |
+-----+--------------+-----+--------------+
|  20 |            1 | 男  |            1 |
|  21 |            1 | 男  |            1 |
|  23 |            1 | 女  |            1 |
|  24 |            1 | 女  |            1 |
|  24 |            1 | 男  |            1 |
|  25 |            3 | 男  |            3 |
+-----+--------------+-----+--------------+
6 rows in set

mysql> SELECT AGE ,COUNT(stuID),SEX,COUNT(stuID) FROM T_STUDENTS GROUP BY
AGE,SEX HAVING AGE >= 23;
+-----+--------------+-----+--------------+
| AGE | COUNT(stuID) | SEX | COUNT(stuID) |
+-----+--------------+-----+--------------+
|  23 |            1 | 女  |            1 |
|  24 |            1 | 女  |            1 |
|  24 |            1 | 男  |            1 |
|  25 |            3 | 男  |            3 |
+-----+--------------+-----+--------------+
4 rows in set

---- 限制结果集函数
-- MYSQL LIMIT offset,n
mysql> SELECT * FROM T_STUDENTS ORDER BY AGE ;

+---------+------------+-----+-----+------------+
| stuID   | NAME       | AGE | SEX | BIRTH      |
+---------+------------+-----+-----+------------+
| S000005 | 周五       |  20 | 男  | 2000-05-03 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 |
+---------+------------+-----+-----+------------+
8 rows in set

mysql> SELECT * FROM T_STUDENTS ORDER BY AGE LIMIT 1,3;

+---------+------------+-----+-----+------------+
| stuID   | NAME       | AGE | SEX | BIRTH      |
+---------+------------+-----+-----+------------+
| S000002 | 钱二       |  21 | 男  | 1999-05-13 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 |
+---------+------------+-----+-----+------------+
3 rows in set
```

```

-- ORACL ROWBUM
SELECT * FROM KHB_USER WHERE ROWNUM <= 5 ORDER BY KHB_USER.USER_ID;
/*
沈晓芳 10012515    111111  我是个勤劳的小蜜蜂    是！！！
苏伟  1033242 111111  我是个勤劳的小蜜蜂    是
q   54424   628142  我是个勤劳的小蜜蜂    是!
许海英 78957   78957   我是个勤劳的小蜜蜂    是!
王丽君 Z013356 222555  我是个勤劳的小蜜蜂    是!\
*/
```

5.4 连接与集合查询

5.4.1 连接查询

1、内连接

```
---- 等值连接
-- 方法一 WHERE子句
SELECT 表1.字段 , 表2.字段 ...
FROM 表1 , 表2
WHERE 表1.字段1 = 表2.字段2
-- MYSQL
mysql> SELECT
T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result FROM
T_STUDENTS ,T_RESULTS WHERE T_STUDENTS.stuID = T_RESULTS.stuID;
+---------+------+---------+--------+
| stuID   | NAME | curID   | result |
+---------+------+---------+--------+
| S000001 | 赵大 | C000001 |     82 |
| S000001 | 赵大 | C000002 |     92 |
| S000001 | 赵大 | C000003 |    100 |
| S000002 | 钱二 | C000001 |     82 |
| S000002 | 钱二 | C000002 |     94 |
| S000002 | 钱二 | C000003 |     46 |
| S000003 | 孙三 | C000001 |     82 |
| S000003 | 孙三 | C000002 |     96 |
| S000003 | 孙三 | C000003 |     46 |
| S000004 | 李四 | C000001 |     89 |
| S000004 | 李四 | C000002 |     64 |
| S000004 | 李四 | C000003 |     58 |
+---------+------+---------+--------+
12 rows in set

-- 方法二 ON子句
SELECT 表1.字段 , 表2.字段 ...
FROM 表1 JOIN 表2
ON 表1.字段1 = 表2.字段2
-- MYSQL
SELECT T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result
FROM T_STUDENTS JOIN T_RESULTS ON T_STUDENTS.stuID = T_RESULTS.stuID;
+---------+------+---------+--------+
| stuID   | NAME | curID   | result |
+---------+------+---------+--------+
| S000001 | 赵大 | C000001 |     82 |
| S000001 | 赵大 | C000002 |     92 |
| S000001 | 赵大 | C000003 |    100 |
```

```
| S000002 | 钱二 | C000001 |     82 |
| S000002 | 钱二 | C000002 |     94 |
| S000002 | 钱二 | C000003 |     46 |
| S000003 | 孙三 | C000001 |     82 |
| S000003 | 孙三 | C000002 |     96 |
| S000003 | 孙三 | C000003 |     46 |
| S000004 | 李四 | C000001 |     89 |
| S000004 | 李四 | C000002 |     64 |
| S000004 | 李四 | C000003 |     58 |
+---------+------+---------+--------+
12 rows in set

-- 方法三 USING子句
SELECT 表1.字段 , 表2.字段 ...
FROM 表1 JOIN 表2
USING (关联字段)
-- MYSQL
SELECT T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result
FROM T_STUDENTS JOIN T_RESULTS USING (stuID);
 SELECT T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result
FROM T_STUDENTS JOIN T_RESULTS USING (stuID);
+---------+------+---------+--------+
| stuID   | NAME | curID   | result |
+---------+------+---------+--------+
| S000001 | 赵大 | C000001 |     82 |
| S000001 | 赵大 | C000002 |     92 |
| S000001 | 赵大 | C000003 |    100 |
| S000002 | 钱二 | C000001 |     82 |
| S000002 | 钱二 | C000002 |     94 |
| S000002 | 钱二 | C000003 |     46 |
| S000003 | 孙三 | C000001 |     82 |
| S000003 | 孙三 | C000002 |     96 |
| S000003 | 孙三 | C000003 |     46 |
| S000004 | 李四 | C000001 |     89 |
| S000004 | 李四 | C000002 |     64 |
| S000004 | 李四 | C000003 |     58 |
+---------+------+---------+--------+
12 rows in set


---- 非等值连接
-- MYSQL
mysql>  SELECT
T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result FROM
T_STUDENTS ,T_RESULTS WHERE T_STUDENTS.stuID = T_RESULTS.stuID AND
T_RESULTS.result > 90;
+---------+------+---------+--------+
| stuID   | NAME | curID   | result |
+---------+------+---------+--------+
| S000001 | 赵大 | C000002 |     92 |
| S000001 | 赵大 | C000003 |    100 |
| S000002 | 钱二 | C000002 |     94 |
| S000003 | 孙三 | C000002 |     96 |
+---------+------+---------+--------+
4 rows in set
```

2、交叉连接

交叉连接返回的结果式一个笛卡尔积，积两个集合相乘的结果，一般情况下，不会使用交叉连接进行数据查询。

```
--
如果多表连接查询时没有WHRER子句中指定查询条件，那么这种查询就是交叉查询。

--MYSQL
mysql> SELECT * FROM T_STUDENTS,T_RESULTS;
+---------+------------+-----+-----+------------+---------+---------+-----
---+
| stuID   | NAME       | AGE | SEX | BIRTH      | stuID   | curID   |
result |
+---------+------------+-----+-----+------------+---------+---------+-----
---+
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000001 | C000001 |
82 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000001 | C000001 |
82 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000001 | C000001 |
82 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000001 | C000001 |
82 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000001 | C000001 |
82 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000001 | C000001 |
82 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000001 | C000001 |
82 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000001 | C000001 |
82 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000001 | C000002 |
92 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000001 | C000002 |
92 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000001 | C000002 |
92 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000001 | C000002 |
92 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000001 | C000002 |
92 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000001 | C000002 |
92 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000001 | C000002 |
92 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000001 | C000002 |
92 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000001 | C000003 |
 100 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000001 | C000003 |
 100 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000001 | C000003 |
 100 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000001 | C000003 |
 100 |
```

```
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000001 | C000003 |
 100 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000001 | C000003 |
 100 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000001 | C000003 |
 100 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000001 | C000003 |
 100 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000002 | C000001 |
82 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000002 | C000001 |
82 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000002 | C000001 |
82 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000002 | C000001 |
82 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000002 | C000001 |
82 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000002 | C000001 |
82 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000002 | C000001 |
82 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000002 | C000001 |
82 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000002 | C000002 |
94 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000002 | C000002 |
94 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000002 | C000002 |
94 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000002 | C000002 |
94 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000002 | C000002 |
94 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000002 | C000002 |
94 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000002 | C000002 |
94 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000002 | C000002 |
94 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000002 | C000003 |
46 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000002 | C000003 |
46 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000002 | C000003 |
46 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000002 | C000003 |
46 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000002 | C000003 |
46 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000002 | C000003 |
46 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000002 | C000003 |
46 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000002 | C000003 |
46 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000003 | C000001 |
82 |
```

```
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000003 | C000001 |
82 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000003 | C000001 |
82 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000003 | C000001 |
82 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000003 | C000001 |
82 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000003 | C000001 |
82 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000003 | C000001 |
82 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000003 | C000001 |
82 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000003 | C000002 |
96 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000003 | C000002 |
96 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000003 | C000002 |
96 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000003 | C000002 |
96 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000003 | C000002 |
96 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000003 | C000002 |
96 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000003 | C000002 |
96 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000003 | C000002 |
96 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000003 | C000003 |
46 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000003 | C000003 |
46 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000003 | C000003 |
46 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000003 | C000003 |
46 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000003 | C000003 |
46 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000003 | C000003 |
46 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000003 | C000003 |
46 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000003 | C000003 |
46 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000004 | C000001 |
89 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000004 | C000001 |
89 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000004 | C000001 |
89 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000004 | C000001 |
89 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000004 | C000001 |
89 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000004 | C000001 |
89 |
```

```
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000004 | C000001 |
89 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000004 | C000001 |
89 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000004 | C000002 |
64 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000004 | C000002 |
64 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000004 | C000002 |
64 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000004 | C000002 |
64 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000004 | C000002 |
64 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000004 | C000002 |
64 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000004 | C000002 |
64 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000004 | C000002 |
64 |
| S000001 | 赵大       |  25 | 男  | 1995-04-12 | S000004 | C000003 |
58 |
| S000002 | 钱二       |  21 | 男  | 1999-05-13 | S000004 | C000003 |
58 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 | S000004 | C000003 |
58 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 | S000004 | C000003 |
58 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 | S000004 | C000003 |
58 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 | S000004 | C000003 |
58 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 | S000004 | C000003 |
58 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 | S000004 | C000003 |
58 |
+---------+------------+-----+-----+------------+---------+---------+-----
---+
96 rows in set

```

3、自连接

```
--
自连接是对同一张数据表中的内容进行连接查询，所以需要为表定义不同的别名。
-- MYSQL
mysql> SELECT * FROM T_STUDENTS S1 ,T_STUDENTS S2 WHERE S1.SEX
 = '男' AND S1.AGE > S2.AGE;S2.AGE;
+---------+------+-----+-----+------------+---------+------------+-----+---
--+------------+
| stuID   | NAME | AGE | SEX | BIRTH      | stuID   | NAME       | AGE |
SEX | BIRTH      |
+---------+------+-----+-----+------------+---------+------------+-----+---
--+------------+
| S000001 | 赵大 |  25 | 男  | 1995-04-12 | S000002 | 钱二       |  21 | 男
| 1999-05-13 |
```

```
| S000004 | 李四 |  24 | 男  | 1996-05-23 | S000002 | 钱二       |  21 | 男
| 1999-05-13 |
| S000006 | 赵大 |  25 | 男  | 1995-04-12 | S000002 | 钱二       |  21 | 男
| 1999-05-13 |
| S000007 | 吴三 |  25 | 男  | 1995-02-22 | S000002 | 钱二       |  21 | 男
| 1999-05-13 |
| S000001 | 赵大 |  25 | 男  | 1995-04-12 | S000003 | 孙三       |  24 | 女
| 1996-08-06 |
| S000006 | 赵大 |  25 | 男  | 1995-04-12 | S000003 | 孙三       |  24 | 女
| 1996-08-06 |
| S000007 | 吴三 |  25 | 男  | 1995-02-22 | S000003 | 孙三       |  24 | 女
| 1996-08-06 |
| S000001 | 赵大 |  25 | 男  | 1995-04-12 | S000004 | 李四       |  24 | 男
| 1996-05-23 |
| S000006 | 赵大 |  25 | 男  | 1995-04-12 | S000004 | 李四       |  24 | 男
| 1996-05-23 |
| S000007 | 吴三 |  25 | 男  | 1995-02-22 | S000004 | 李四       |  24 | 男
| 1996-05-23 |
| S000001 | 赵大 |  25 | 男  | 1995-04-12 | S000005 | 周五       |  20 | 男
| 2000-05-03 |
| S000002 | 钱二 |  21 | 男  | 1999-05-13 | S000005 | 周五       |  20 | 男
| 2000-05-03 |
| S000004 | 李四 |  24 | 男  | 1996-05-23 | S000005 | 周五       |  20 | 男
| 2000-05-03 |
| S000006 | 赵大 |  25 | 男  | 1995-04-12 | S000005 | 周五       |  20 | 男
| 2000-05-03 |
| S000007 | 吴三 |  25 | 男  | 1995-02-22 | S000005 | 周五       |  20 | 男
| 2000-05-03 |
| S000001 | 赵大 |  25 | 男  | 1995-04-12 | S000008 | JERMAN_RAY |  23 | 女
| 1997-05-22 |
| S000004 | 李四 |  24 | 男  | 1996-05-23 | S000008 | JERMAN_RAY |  23 | 女
| 1997-05-22 |
| S000006 | 赵大 |  25 | 男  | 1995-04-12 | S000008 | JERMAN_RAY |  23 | 女
| 1997-05-22 |
| S000007 | 吴三 |  25 | 男  | 1995-02-22 | S000008 | JERMAN_RAY |  23 | 女
| 1997-05-22 |
+---------+------+-----+-----+------------+---------+------------+-----+---
--+------------+
```

4、外连接

外连接查询不仅可以返回满足查询条件的记录，可还以返回不匹配的记录

```
---- 左外连接
左侧表数据将全部显示出来
-- 语法
-- MYSQL
SELECT 表1.字段,表2.字段...
FROM 表1 LEFT [OUTER] JOIN 表2
ON 表1.字段 = 表2.字段

mysql> SELECT
    S.stuID,
    S.NAME,
    R.curID,
    R.result
FROM
```

```
    T_STUDENTS S
LEFT OUTER JOIN T_RESULTS R
ON
    S.stuID = R.stuID;
+---------+------------+---------+--------+
| stuID   | NAME       | curID   | result |
+---------+------------+---------+--------+
| S000001 | 赵大       | C000001 |     82 |
| S000001 | 赵大       | C000002 |     92 |
| S000001 | 赵大       | C000003 |    100 |
| S000002 | 钱二       | C000001 |     82 |
| S000002 | 钱二       | C000002 |     94 |
| S000002 | 钱二       | C000003 |     46 |
| S000003 | 孙三       | C000001 |     82 |
| S000003 | 孙三       | C000002 |     96 |
| S000003 | 孙三       | C000003 |     46 |
| S000004 | 李四       | C000001 |     89 |
| S000004 | 李四       | C000002 |     64 |
| S000004 | 李四       | C000003 |     58 |
| S000005 | 周五       | NULL    | NULL   |
| S000006 | 赵大       | NULL    | NULL   |
| S000007 | 吴三       | NULL    | NULL   |
| S000008 | JERMAN_RAY | NULL    | NULL   |
+---------+------------+---------+--------+
16 rows in set

-- ORACLE
SELECT 表1.字段,表2.字段...
FROM 表1 LEFT [OUTER] JOIN 表2
WHERE 表1.字段(+) = 表2.字段

SELECT
    T1.CUST_MAT_NO,
    T1.PILE_NO,
    T2.CRANE_NAME
FROM
    KHB_PLATE_PILENO T1,
    L1T109 T2
WHERE
    T1.CUST_MAT_NO (+) = T2.CUST_MAT_NO
AND ROWNUM <= 5;

CUST_MAT_NO          PILE_NO    CRANE_NAME
-------------------- ---------- ------------
E0346051100          W00292     W003
E0346051200          W00292     W003
E0346050100          W00292     W003
E0345117200          X00101     X004
E0345117300          X00191     X004


---- 右外连接
右侧表数据将全部显示出来
-- MYSQL
SELECT 表1.字段,表2.字段...
FROM 表1 RIGHT [OUTER] JOIN 表2
ON 表1.字段 = 表2.字段

```

```
mysql> SELECT
    S.stuID,
    S.NAME,
    R.curID,
    R.result
FROM
    T_STUDENTS S
RIGHT OUTER JOIN T_RESULTS R
ON
    S.stuID = R.stuID;
+---------+------+---------+--------+
| stuID   | NAME | curID   | result |
+---------+------+---------+--------+
| S000001 | 赵大 | C000001 |     82 |
| S000001 | 赵大 | C000002 |     92 |
| S000001 | 赵大 | C000003 |    100 |
| S000002 | 钱二 | C000001 |     82 |
| S000002 | 钱二 | C000002 |     94 |
| S000002 | 钱二 | C000003 |     46 |
| S000003 | 孙三 | C000001 |     82 |
| S000003 | 孙三 | C000002 |     96 |
| S000003 | 孙三 | C000003 |     46 |
| S000004 | 李四 | C000001 |     89 |
| S000004 | 李四 | C000002 |     64 |
| S000004 | 李四 | C000003 |     58 |
+---------+------+---------+--------+
12 rows in set

-- ORACLE
SELECT 表1.字段,表2.字段...
FROM 表1 LEFT [OUTER] JOIN 表2
WHERE 表1.字段 = 表2.字段(+)

SELECT
    T1.CUST_MAT_NO,
    T1.PILE_NO,
    T2.CRANE_NAME
FROM
    KHB_PLATE_PILENO T1,
    L1T109 T2
WHERE
    T1.CUST_MAT_NO  = T2.CUST_MAT_NO(+)
AND ROWNUM <= 5;

CUST_MAT_NO          PILE_NO    CRANE_NAME
-------------------- ---------- ------------
B0360045100          V00941
B0360046200          X00691
B0360047200          V00941
B0360047300          V00941
B0360048100          X00691

---- 全外连接
显示左右侧表的全部数据，可看作左外连接和右外连接的合集，不包括重复数据。
-- 语法
SELECT 表1.字段,表2.字段...
FROM 表1 FULL [OUTER] JOIN 表2
ON 表1.字段 = 表2.字段
```

```
131
132
```

5.4.2 集合查询

```
---- 并操作(UNION)
-- 语法
SELECT 语句1
UNION
SELECT 语句2

-- 注意 两个SELECT语句查询结果集的列必须相同，并且数据类型要一致

-- MYSQL
SELECT
    S.stuID,
    S.NAME
FROM
    T_STUDENTS S
UNION
    SELECT
        R.curID,
        R.result
    FROM
        T_RESULTS R;
+---------+------------+
| stuID   | NAME       |
+---------+------------+
| S000001 | 赵大       |
| S000002 | 钱二       |
| S000003 | 孙三       |
| S000004 | 李四       |
| S000005 | 周五       |
| S000006 | 赵大       |
| S000007 | 吴三       |
| S000008 | JERMAN_RAY |
| C000001 | 82         |
| C000002 | 92         |
| C000003 | 100        |
| C000002 | 94         |
| C000003 | 46         |
| C000002 | 96         |
| C000001 | 89         |
| C000002 | 64         |
| C000003 | 58         |
+---------+------------+
17 rows in set

---- 交操作(INTERSECT)

---- 差操作(MINUS)
```

5.5 子查询

5.5.1 单行子查询

```
1
```

5.5.2 多行子查询

```
1
```

5.5.3 多列子查询

```
1
```

5.5.4 使用ANY和ALL运算符的子查询

```
1
```

5.5.5 相关子查询

```
1
```

5.5.6 多重子查询

```
1
```

5.6 视图

```
1  视图可以看作式从一张或者多张数据表(或视图)中导出的表，本身不存储数据，是张虚拟的表。如果数
   据表中的数据发生变化，则与之相关的视图中的数据也会随之变化。
2
3  -- 视图的作用
4  1、 提高数据访问的安全性
5  2、 简化复杂查询操作
6
7  -- 创建视图
8  ---- 1、基于单表查询创建视图
9  CREATE VIEW VIEW_NAME
10 ([别名1[,别名2]...])
11 AS
12 subQuery
13
14 mysql> CREATE VIEW V_STUDENTS
15 (v_学号,v_姓名,v_年龄,v_性别,v_出生日期)
16 AS
17 SELECT * FROM T_STUDENTS;
18 Query OK, 0 rows affected
19
20 mysql> SELECT * FROM V_STUDENTS;
21 +---------+------------+--------+--------+------------+
22 | v_学号  | v_姓名     | v_年龄 | v_性别 | v_出生日期 |
23 +---------+------------+--------+--------+------------+
24 | S000001 | 赵大(*)    |     25 | 男     | 1995-04-12 |
25 | S000002 | 钱二       |     21 | 男     | 1999-05-13 |
```

```
| S000003 | 孙三        |     24 | 女     | 1996-08-06 |
| S000004 | 李四        |     24 | 男     | 1996-05-23 |
| S000005 | 周五        |     20 | 男     | 2000-05-03 |
| S000006 | 郑六(*)     |     25 | 男     | 1995-04-12 |
| S000007 | 吴三(*)     |     25 | 男     | 1995-02-22 |
| S000008 | JERMAN_RAY |     23 | 女     | 1997-05-22 |
+---------+------------+--------+--------+------------+
8 rows in set

---- 2、基于多表查询创建视图
CREATE VIEW VIEW_NAME
([别名1[,别名2]...])
AS
subQuery

mysql> CREATE VIEW V_STUDENTS_RESULT
(v_学号,v_姓名,v_课号,v_成绩)
AS
SELECT S.stuID,S.NAME,R.curID,R.result
FROM T_STUDENTS S,T_RESULTS R
WHERE R.result >= 60
AND S.stuID = R.stuID;

mysql> SELECT * FROM V_STUDENTS
_RESULT;
+---------+---------+---------+--------+
| v_学号  | v_姓名  | v_课号  | v_成绩 |
+---------+---------+---------+--------+
| S000001 | 赵大(*) | C000001 |     82 |
| S000001 | 赵大(*) | C000002 |     92 |
| S000001 | 赵大(*) | C000003 |    100 |
| S000002 | 钱二    | C000001 |     82 |
| S000002 | 钱二    | C000002 |     94 |
| S000003 | 孙三    | C000001 |     82 |
| S000003 | 孙三    | C000002 |     96 |
| S000004 | 李四    | C000001 |     89 |
| S000004 | 李四    | C000002 |     64 |
+---------+---------+---------+--------+
9 rows in set

---- 3、基于函数、分组数据创建视图
CREATE VIEW VIEW_NAME
([别名1[,别名2]...])
AS
subQuery

mysql>  CREATE VIEW V_TEACHER
(v_编号,v_姓名,v_学院,v_薪资)
AS
SELECT teaID,teaNAME,depNAME,MAX(salary) as salary
FROM T_TEACHERS
GROUP BY depID
HAVING MAX(salary) >= 5000;

mysql> SELECT * FROM  V_TEACHER;
+--------+--------+--------+--------+
| v_编号 | v_姓名 | v_学院 | v_薪资 |
+--------+--------+--------+--------+
```

```
| T00001 | 教师1  | 学院1  |   7875 |
| T00002 | 教师2  | 学院2  |   5400 |
+--------+--------+--------+--------+
2 rows in set

---- 4、创建只读视图
CREATE VIEW V_TEACHER2
AS
SELECT * FROM T_TEACHERS
WITH READ ONLY;

-- 删除视图
DROP VIEW VIEW_NAME
DROP VIEW V_STUDENTS;
```

六、数据更新

6.1 插入数据 (INSERT)

```
---- 1、插入单行数据记录
-- 语法
INSERT INTO 表名[(列1，列2...)] VALUES (列值1,列值2...)
-- MYSQL
INSERT INTO T_STUDENTS VALUES('S000001','赵大',25,'男','19950412');
INSERT INTO T_STUDENTS VALUES('S000002','钱二',21,'男','19990513');
INSERT INTO T_STUDENTS VALUES('S000003','孙三',24,'女','19960806');
INSERT INTO T_STUDENTS VALUES('S000004','李四',24,'男','19960523');
INSERT INTO T_STUDENTS VALUES('S000005','周五',20,'男','20000503');
INSERT INTO T_STUDENTS VALUES('S000006','赵大',25,'男','19950412');
INSERT INTO T_STUDENTS VALUES('S000007','吴三',25,'男','19950222');
INSERT INTO T_STUDENTS VALUES('S000008','JERMAN_RAY',23,'女','19970522');

INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000001', 'C000001', '82');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000001', 'C000002', '92');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000001', 'C000003', '100');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000002', 'C000001', '82');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000002', 'C000002', '94');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000002', 'C000003', '46');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000003', 'C000001', '82');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000003', 'C000002', '96');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000003', 'C000003', '46');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000004', 'C000001', '89');
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000004', 'C000002', '64');
```

```sql
INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES ('S000004', 'C000003', '58');

INSERT INTO `NJITDB`.`T_TEACHERS` (`teaID`, `teaNAME`, `AGE`, `SEX`, `depID`, `depNAME`, `PROFESSION`, `SALARY`, `PENSION`) VALUES ('T00001', '教师1', '28', '女', 'D00001', '学院1', '普通职员', '7500', '0');
INSERT INTO `NJITDB`.`T_TEACHERS` (`teaID`, `teaNAME`, `AGE`, `SEX`, `depID`, `depNAME`, `PROFESSION`, `SALARY`, `PENSION`) VALUES ('T00002', '教师2', '22', '女', 'D00002', '学院2', '普通职员', '3500', '0');
INSERT INTO `NJITDB`.`T_TEACHERS` (`teaID`, `teaNAME`, `AGE`, `SEX`, `depID`, `depNAME`, `PROFESSION`, `SALARY`, `PENSION`) VALUES ('T00003', '教师3', '23', '女', 'D00002', '学院2', '普通职员', '4500', '0');
INSERT INTO `NJITDB`.`T_TEACHERS` (`teaID`, `teaNAME`, `AGE`, `SEX`, `depID`, `depNAME`, `PROFESSION`, `SALARY`, `PENSION`) VALUES ('T00004', '教师4', '25', '男', 'D00001', '学院1', '普通职员', '6500', '0');
INSERT INTO `NJITDB`.`T_TEACHERS` (`teaID`, `teaNAME`, `AGE`, `SEX`, `depID`, `depNAME`, `PROFESSION`, `SALARY`, `PENSION`) VALUES ('T00005', '教师5', '25', '女', 'D00002', '学院2', '普通职员', '5000', '0');
INSERT INTO `NJITDB`.`T_TEACHERS` (`teaID`, `teaNAME`, `AGE`, `SEX`, `depID`, `depNAME`, `PROFESSION`, `SALARY`, `PENSION`) VALUES ('T00006', '教师6', '26', '男', 'D00001', '学院1', '普通职员', '5000', '0');


-- 先视图中插入数据记录
-- 操作方式是一样的，只是对视图中的数据插入操作，最终会转化为数据表的插入操作
```

6.2 修改数据 (UPDATE)

```sql
-- 修改单行数据记录
-- 单条数据满足WHERE后面的查询条件
UPDETE 表名 SET 列1 = 值1[,列2 = 值2] WHERE 条件
UPDATE T_STUDENTS SET NAME = '郑六' WHERE stuID = 'S000006';

-- 修改多行数据记录
-- 多条数据满足WHERE后面的查询条件
UPDETE 表名 SET 列1 = 值1[,列2 = 值2] WHERE 条件
UPDATE T_STUDENTS SET NAME = CONCAT(NAME,'(*)') WHERE AGE >= 25

-- CASE条件表达式修改数据记录
UPDATE T_TEACHERS
SET salary =
CASE
WHEN salary <= 4000 THEN salary+salary*0.1
WHEN salary > 4000 AND salary <= 6000 THEN salary+salary*0.08
WHEN salary > 6000 THEN salary+salary*0.05
ELSE salary
END;

-- 修改视图中的数据记录
-- 操作方式是一样的，只是对视图中的数据修改操作，最终会转化为数据表的修改操作
```

6.3 删除数据 (DELETE)

```sql
-- 删除满足条件的记录
```

```
DELETE FROM 表名 WHERE 条件
DELETE FROM T_STUDENTS WHERE stuID = 'S000001';

-- 若不加条件，将删除所有数据
DELETE FROM 表名
DELETE FROM T_STUDENTS;

-- 定义外键约束的表中删除数据记录
-- 外键约束定义的删除方式是CASCADE时，先将主表数据删除，从表也就跟着删除了，若先删除从表数据，主表数据还在。

----- 删除前
mysql> SELECT * FROM T_STUDENTS;
+---------+------------+-----+-----+------------+
| stuID   | NAME       | AGE | SEX | BIRTH      |
+---------+------------+-----+-----+------------+
| S000002 | 钱二       |  21 | 男  | 1999-05-13 |
| S000003 | 孙三       |  24 | 女  | 1996-08-06 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 |
+---------+------------+-----+-----+------------+
7 rows in set

mysql> SELECT * FROM T_RESULTS;
+---------+---------+--------+
| stuID   | curID   | result |
+---------+---------+--------+
| S000002 | C000001 |     82 |
| S000002 | C000002 |     94 |
| S000002 | C000003 |     46 |
| S000003 | C000001 |     82 |
| S000003 | C000002 |     96 |
| S000003 | C000003 |     46 |
| S000004 | C000001 |     89 |
| S000004 | C000002 |     64 |
| S000004 | C000003 |     58 |
+---------+---------+--------+
9 rows in set

----- 删除
DELETE FROM T_STUDENTS WHERE stuID = 'S000002';

----- 删除后
mysql> SELECT * FROM T_STUDENTS;
+---------+------------+-----+-----+------------+
| stuID   | NAME       | AGE | SEX | BIRTH      |
+---------+------------+-----+-----+------------+
| S000003 | 孙三       |  24 | 女  | 1996-08-06 |
| S000004 | 李四       |  24 | 男  | 1996-05-23 |
| S000005 | 周五       |  20 | 男  | 2000-05-03 |
| S000006 | 赵大       |  25 | 男  | 1995-04-12 |
| S000007 | 吴三       |  25 | 男  | 1995-02-22 |
| S000008 | JERMAN_RAY |  23 | 女  | 1997-05-22 |
+---------+------------+-----+-----+------------+
6 rows in set
```

```
mysql> SELECT * FROM T_RESULTS;
+---------+---------+--------+
| stuID   | curID   | result |
+---------+---------+--------+
| S000003 | C000001 |     82 |
| S000003 | C000002 |     96 |
| S000003 | C000003 |     46 |
| S000004 | C000001 |     89 |
| S000004 | C000002 |     64 |
| S000004 | C000003 |     58 |
+---------+---------+--------+
6 rows in set

-- 删除视图中的数据记录
-- 操作方式是一样的，只是对视图中的数据删除操作，最终会转化为数据表的删除操作
```

七、事务控制

7.1 概念

事务是指单个逻辑工作单元执行的操作集合。

事务的满足ACID属性：

- 原子性(Atomicity)：事务中所有执行的操作要么全部成功、要么全部不执行，回滚到之前的状态
- 一致性(Consistency)：事务操作执行完成之后，数据库中的数据必须保证合法一致的状态。
- 隔离性(Isolation)：事务看到的数据库中的数据要么是事务修改之前的状态，要么是之后的状态。
- 持久性(Durability)：事务一旦执行完成，数据库中的数据将会永久保存下来。

7.2 简单案例

ORACLE：

```
BEGIN
    SAVEPOINT sp;

    INSERT INTO T_STUDENTS
         VALUES ('203150548',
                 '张三',
                 20,
                 '男',
                 to_date('1996-06-03','yyyy-mm-dd'));
EXCEPTION
    WHEN TIMEOUT_ON_RESOURCE
    THEN
        ROLLBACK TO sp;
        COMMIT;
END;
```

八、权限管理

```
-- 授予/回收指定用户操作数据表的权限
GRANT/REVOKE 权限[,权限] ON TABLE 表名[,表明] TO 用户[,用户]

GRANT/REVOKE SELECT,INSERT,DELETE,UPDATE ON TABLE T_STUDENTS TO ZZZ;

-- 授予/回收指定用户操作数据列的权限
GRANT/REVOKE 权限[,权限] ON TABLE 表名[,表明] TO 用户[,用户]

GRANT/REVOKE UPDATE(NAME,AGE) ON TABLE T_STUDENTS TO ZZZ;

-- 授予/回收指定用户授权的权限
GRANT/REVOKE 权限[,权限] ON TABLE 表名[,表明] TO 用户[,用户] WITH GRANT OPTION

GRANT/REVOKE SELECT ON TABLE T_STUDENTS TO ZZZ WITH GRANT OPTION;

-- 授予/回收创建数据表的权限
GRANT/REVOKE CREATETAB ON DATABASE DBName TO 用户

GRANT/REVOKE CREATETAB ON DATABASE NJITDB  TO zzz;

-- 授予/回收所有用户的操作权限
GRANT/REVOKE 权限[,权限] ON TABLE 表名[,表明] TO PUBLIC

GRANT/REVOKE SELECT,INSERT,DELETE,UPDATE ON TABLE T_STUDENTS TO PUBLIC;
```

九、参考

1、SQL类型

2、SQL函数

- 聚合函数(统计函数)

函数功能	MySQL函数	Oracle函数
求平均值	AVG()	AVG()
计算行数	COUNT()	COUNT()
求最大值	MAX()	MAX()
求最小值	MIN()	MIN()
求总和	SUM()	SUM()

- 字符函数

函数功能	MySQL函数	Oracle函数
取得字符的ASCII码	ASCII(string)	ASCII(string)
字符串连接	CONCAT(string1,string2,...)	CONCAT(string1,string2)
取值指定子串在字符串中的位置	INSTR(string,substring)	INSTR(string,substring[,n[,m]])
将字符串全部转换成大写	UPPER(string)	UPPER(string)
将字符串全部转换成小写	LOWER(string)_	LOWER(string)_
去除字符串左侧空格或字符	LTRIM(string[,set])	LTRIM(string[,set])
去除字符串右侧空格或字符	RTRIM(string[,set])	RTRIM(string[,set])
替换指定的字串	REPLACE(string)	REPLACE(string)
字符串逆序	REVERSE(string)	REVERSE(string)
左侧填充空格或字符	LPAD(string,length,padstring)	LPAD(string,length,padstring)
右侧填充空格或字符	RPAD(string,length,padstring)	RPAD(string,length,padstring)
截取字符串	SUBSTRING(string FROM start [FOR length])	SUBSTRING(string,start [, length)
从指定字符串左侧读取字串	LEFT(string)	/
从指定字符串右侧读取字串	RIGHT(string)	/
计算字符串长度	LENGTH(string)	LENGTH(string)
比较两个字符串	STRCMP(string1,string2)	/

- 数字函数

函数功能	MySQL函数	Oracle函数
绝对值	ABS(n)	ABS(n)
反余弦值	ACOS(n)	ACOS(n)
发正弦值	ASIN(n)	ASIN(n)
余弦值	COS(n)	COS(n)
余切值	COT(n)	COT(n)
正弦值	SIN(n)	SIN(n)
正切值	TAN(n)	TAN(n)
取大于等于指定数的最小整数	CEIL(n)	CEIL(n)
取小于等于指定值的最大整数	FLOOR(n)	FLOOR(n)
弧度转换成角度	DEGREES(n)	/
角度转换成弧度	RADIANS(n)	/
以e为底的幂运算	EXP(n)	EXP(n)
求自然对数	LOG(n)	LN(n)
以10为底对数	LOG10(n)	LOG(10,n)
求余数	MOD(n,m)	MOD(n,m)
求π值	PI()	/
以其它任意数为底的幂运算	POWER(a,b)	POWER(a,b)
求平方	POWER(m,n)	POWER(m,n)
求四舍五入值	ROUND(n [,m])	ROUND(n [,m])
求平方根	SQRT()	SQRT()
保留指定值的小数位数	TRUNCATE(n[,m])	TRUNC(n[,m])
求集合中最大值	GREATEST(n1,n2,n3...)	GREATEST(n1,n2,n3...)
求集合中最小值	LEAST(n1,n2,n3...)	LEAST(n1,n2,n3...)

- 日期函数

函数功能	MySQL函数	Oracle函数
获取当前系统的日期和时间	NOW()或者SYSDATE()	SYSDATE
对日期进行加减运算	DATE_ADD(date,INTERVAL expression type) eg.SELECT DATE_ADD(SYSDATE(),INTERVAL 1 DAY) FROM DUAL;	

- 类型转换函数

函数功能	MySQL函数	Oracle函数
字符转换函数	CAST(expression AS datatype[length])	TO_CHAR(n[,fmt[.'nlsparams']])
日期转换函数	DATE_FORMAT(date,fmt)	TO_Date(string[,fmt[.'nlsparams']])
数值转换函数	/	TO_NUMBER(char, [,fmt[.'nlsparams']])

- 空值处理函数

函数功能	MySQL函数	Oracle函数
判断表达式是否为NULL	ISNULL(expression)	/
if exp1 不为 NULL return exp1 else return exp2	/	NVL(exp1,exp2)
if exp1 不为 NULL return exp2 else return exp3		NVL2(exp1,exp2,exp3)
找到第一个不为NULL的表达式值	COALESCE(exp1,exp2,exp3...)	COALESCE(exp1,exp2,exp3...)

- 分支函数

函数功能	MySQL函数	Oracle函数
分支控制	IF(exp,result1,result2)	DECODE(exp,search1,result1[,search2,result2]...[deafault_result])
多分支控制	CASE WHEN condition1 THEN result1 WHEN condition2 THEN result2 ... WHEN conditionn THEN resultn [ELSE default_value] END	CASE WHEN condition1 THEN result1 WHEN condition2 THEN result2 ... WHEN conditionn THEN resultn [ELSE default_value] END

十、PL/SQL - ORACLE

10.1 简介

PL/SL（Procedural Language/SQL）是Orcale对标准数据SQL语言的扩展，增加了许多程序设计语言中的过程结构，主要包括以下几个方面:

- 变量
- 数据类型
- 结构控制语句
- 过程与函数
- 异常处理

10.2 基础

基本语句块结构

```
<<名字>> --语句块命名
DECLARE
    /*
    * 定义部分
    * 定义常量、变量、游标、数据类型等
    */
BEGIN
    /*
    * 执行部分
    * 处理业务逻辑，实现程序功能
    */
EXCEPTION
    /*
    * 异常处理部分
```

```
    * 处理程序中可能出现的错误
    */
END;

注意,其中只有执行部分是必需的，其它都是可选的。

<<NAME>>
DECLARE
    v_username    VARCHAR (20) := 'ZZZ';
    v_userid      VARCHAR (20) := 'z153654';
    v_password    VARCHAR (20) := '123456';
BEGIN
    UPDATE KHB_USER
       SET USER_ID = v_userid
     WHERE USER_NAME = v_username;

EXCEPTION
    WHEN NO_DATA_FOUND
    THEN
        INSERT INTO KHB_USER (USER_NAME, USER_ID, USER_PASSWORD)
             VALUES (v_username, v_userid, v_password);
END;
```

数据类型

```
-- 1、标量类型
---- 数字类型
    NUMBER
    BINARY_INTEGER
    PLS_INTEGER
    BINARY_FLOAT
    BINARY_DOUBLE

---- 字符类型
    CHAR
    VARCHAR2
    NCHAR
    NVARCHAR2
    LONG
    LONGRAW
    RAW

---- 布尔类型

---- 日期时间类型
    DATE
    TIMESTAMP
    NTERVAL

-----------------------------------定义和使用标量类型：BEGIN------------------
-----------
-- 定义语法规则
variable_name [CONSTANT] datatype [NOT NULL] [:= value[DEFAULT]]

-- 例子
DECLARE
    v_username    KHB_USER.USER_NAME%TYPE;
```

```
    v_userid     KHB_USER.USER_ID%TYPE  ;
    v_password   KHB_USER.USER_PASSWORD%TYPE  ;
BEGIN
    SELECT KHB_USER.USER_NAME,KHB_USER.USER_ID,KHB_USER.USER_PASSWORD
    INTO v_username,v_userid,v_password
    FROM KHB_USER
    WHERE KHB_USER.USER_ID = '10027107';

    DBMS_OUTPUT.PUT_LINE('用户姓名：'|| v_username);
    DBMS_OUTPUT.PUT_LINE('用户工号：'|| v_userid);
    DBMS_OUTPUT.PUT_LINE('用户密码：'|| v_password);
END;
-----------------------------------定义和使用标量类型：END-------------------
------------

-- 2、复合类型
    记录
    表
    嵌套表
    变长数组
-----------------------------------定义和使用复合类型：BEGIN------------------
----------
-- 语法规则 定义记录类型和记录变量
    TYPE type_name IS RECORD(
        record_declaration[,
        record_declaration
        ...]
    );

    record_name type_name;

-- 例子 定义和使用记录
DECLARE
    TYPE t_KHB_USER IS RECORD(
    v_username   KHB_USER.USER_NAME%TYPE,        -- 用户姓名
    v_userid     KHB_USER.USER_ID%TYPE  ,        -- 用户工号
    v_password   KHB_USER.USER_PASSWORD%TYPE     -- 用户密码
    );

    user_record t_KHB_USER;                  -- 定义记录变量
BEGIN
    SELECT KHB_USER.USER_NAME,KHB_USER.USER_ID,KHB_USER.USER_PASSWORD
    INTO user_record
    FROM KHB_USER
    WHERE KHB_USER.USER_ID = '10027107';

    DBMS_OUTPUT.PUT_LINE('用户姓名：'|| user_record.v_username);
    DBMS_OUTPUT.PUT_LINE('用户工号：'|| user_record.v_userid);
    DBMS_OUTPUT.PUT_LINE('用户密码：'|| user_record.v_password);
END;

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

-- 语法规则 定义表类型和表变量
    TYPE type_name IS TABLE OF element_type
    [NOT NULL] INDEX BY key_type;

```

```
    table_name type_name;

-- 例子 定义和使用表
DECLARE
    TYPE t_KHB_USER IS TABLE OF KHB_USER.USER_NAME%TYPE
        INDEX BY BINARY_INTEGER;

    user_table   t_KHB_USER;                                        -- 定义表变量
BEGIN
    SELECT KHB_USER.USER_NAME
      INTO user_table(1)
      FROM KHB_USER
     WHERE KHB_USER.USER_ID = '10027107';

    DBMS_OUTPUT.PUT_LINE ('用户姓名：' || user_table (1));
END;

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

-- 语法规则 定义嵌套表类型和嵌套表变量
    TYPE type_name IS TABLE OF element_type [NOT NULL];

    table_name type_name;

-- 例子 定义和使用嵌套表
DECLARE
    TYPE t_KHB_USER IS TABLE OF KHB_USER.USER_NAME%TYPE;

    user_table   t_KHB_USER := t_KHB_USER ('aaa', 'bbb', 'ccc');      -- 定义嵌套表变量
BEGIN
    DBMS_OUTPUT.PUT_LINE (user_table (1));
    DBMS_OUTPUT.PUT_LINE (user_table (2));
    DBMS_OUTPUT.PUT_LINE (user_table (3));
END;

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

-- 语法规则 定义变长数组类型和变长数组变量
    TYPE type_name IS VARRAY(max_size) OF element_data [NOT NULL];

    array_name type_name;

-- 例子 定义和使用变长数组
DECLARE
    TYPE t_KHB_USER IS VARRAY(20) OF KHB_USER.USER_NAME%TYPE;

    user_varray   t_KHB_USER := t_KHB_USER ('aaa', 'bbb', 'ccc');   -- 定义变长数组变量
BEGIN
    DBMS_OUTPUT.PUT_LINE (user_varray (1));
    DBMS_OUTPUT.PUT_LINE (user_varray (2));
    DBMS_OUTPUT.PUT_LINE (user_varray (3));
END;
```

```
----------------------------------定义和使用复合类型: END-------------------
----------

-- 3、引用类型
    REF
    REF CURSOR

-- 4、LOB类型
    BFILE
    BLOB
    CLOB
    NCLOB

-- 5、子类型
----------------------------------定义和使用子类型：BEGIN-------------------
---------
-- 语法规则 定义子类型和子类型变量
    SUBTYPE subtype_name IS original_type [(constraint)][NOT NULL];

    variable_name subtype_name;

-- 例子 定义和使用子类型
    DECLARE
        SUBTYPE t_sum IS NUMBER;
        v_total t_sum(4);
    BEGIN
        v_total :=1000;
        DBMS_OUTPUT.PUT_LINE (v_total);
    END;

----------------------------------定义和使用子类型: END--------------------
---------
```

10.3 控制结构

分支结构

```
-- IF语句
---- 语法规则  允许嵌套
IF expression1 THEN
    statement1;
ELSIF expression2 THEN
    statement2;
ELSE
    statement3;
END IF;


-- CASE语句
---- 语法规则
CASE selector
    WHEN value1 THEN
        statement1;
    WHEN value2 THEN
        statement2;
        ...
    WHEN valueN THEN
```

```
        statementN;
    [ELSE statementN+1;]
END CASE;

```

循环结构

```
--  LOOP语句
---- 语法规则
LOOP
    statement;
    EXIT [WHEN condition]
END LOOP;

---- 例子
DECLARE
    v_index   NUMBER := 1;
    v_count   NUMBER := 0;
BEGIN
    SELECT COUNT (KHB_USER.USER_NAME)
      INTO v_count
      FROM KHB_USER
     WHERE KHB_USER.USER_PASSWORD = '123456';

    LOOP
        DBMS_OUTPUT.PUT_LINE (v_index);
        v_index := v_index + 1;
        EXIT WHEN v_index > v_count;
    END LOOP;
END;

--  WHILE - LOOP语句
---- 语法规则
WHILE condition LOOP
    statement;
    ...
END LOOP;

---- 例子
DECLARE
    v_index   NUMBER := 1;
    v_count   NUMBER := 0;
BEGIN
    SELECT COUNT (KHB_USER.USER_NAME)
      INTO v_count
      FROM KHB_USER
     WHERE KHB_USER.USER_PASSWORD = '123456';

    WHILE v_index <= v_count LOOP
        DBMS_OUTPUT.PUT_LINE (v_index);
        v_index := v_index + 1;
    END LOOP;
END;

--  FOR - LOOP语句
---- 语法规则
FOR counter IN [REVERSE] lower_bound..higher_bound LOOP
```

```
    statement;
    ...
END LOOP;

---- 例子
DECLARE
    v_index   NUMBER := 1;
    v_count   NUMBER := 0;
BEGIN
    SELECT COUNT (KHB_USER.USER_NAME)
      INTO v_count
      FROM KHB_USER
     WHERE KHB_USER.USER_PASSWORD = '123456';

    FOR v_index IN 1..v_count LOOP
        DBMS_OUTPUT.PUT_LINE (v_index);
    END LOOP;
END;

```

顺序结构

```
-- GOTO语句
---- 语法规则  能不用就不用
GOTO label_name;

DECLARE
    v_index   NUMBER := 1;
    v_count   NUMBER := 0;
BEGIN
    GOTO print;

    SELECT COUNT (KHB_USER.USER_NAME)
      INTO v_count
      FROM KHB_USER
     WHERE KHB_USER.USER_PASSWORD = '123456';


    << print >>
    DBMS_OUTPUT.PUT_LINE (v_count);
END;

-- NULL语句
DECLARE
    v_index   NUMBER := 1;
    v_count   NUMBER := 0;
BEGIN
    GOTO print;

    SELECT COUNT (KHB_USER.USER_NAME)
      INTO v_count
      FROM KHB_USER
     WHERE KHB_USER.USER_PASSWORD = '123456';


    << print >>
    NULL;
```

```
END;
```

10.4 游标

在Oracle中，使用SQL语句进行增删改查操作时，数据库管理系统会在内存为其分配一个区域，这个区域是一段上下文的缓冲区，游标就是指向该区域中数据的指针。

显示游标

属性

属性	说明
%FOUND	提取数据是否成功
%NOTFOUND	提取数据是否失败
%ISOPEN	判断游标是否打开
%ROWCOUNT	游标结果集中的行数

使用

```
-- 1、声明游标
CURSOR cursor_name IS select_statement;

-- 2、打开游标OPEN
OPEN cursor_name;

-- 3、提取结果FETCH
FETCH cursor_name [BULK COLLECT] INTO variable1 [,variable2...]

-- variable 可以是标量、记录和集合
-- 使用标量变量读取游标结果集中的数据
-- 使用记录标量读取游标结果集中的数据
-- 使用集合变量读取游标结果集中的数据

-- 4、关闭游标CLOSE
CLOSE cursor_name


-- 例子
DECLARE
    TYPE t_tablename IS TABLE OF KHB_USER%ROWTYPE;

    user_table   t_tablename;

    CURSOR cursor_user
    IS
        SELECT *
          FROM KHB_USER
         WHERE KHB_USER.USER_PASSWORD = '123456';           -- 1、声明游标
BEGIN
    OPEN cursor_user;                                       -- 2、打开游标
OPEN

    IF cursor_user%ISOPEN
```

```
    THEN
        DBMS_OUTPUT.PUT_LINE ('打开游标成功');

        FETCH cursor_user BULK COLLECT INTO user_table;      -- 3、提取结果FETCH

        IF cursor_user%FOUND
        THEN
            DBMS_OUTPUT.PUT_LINE (
                '提取数据成功,行数：' || cursor_user%ROWCOUNT);
        ELSIF cursor_user%NOTFOUND
        THEN
            DBMS_OUTPUT.PUT_LINE ('提取数据失败');
        END IF;

        FOR i IN 1 .. cursor_user%ROWCOUNT                  -- user_table.COUNT
        LOOP
            DBMS_OUTPUT.PUT_LINE (
                    '用户名：'
                || user_table (i).USER_NAME
                || ' 用户工号：'
                || user_table (i).USER_ID
                || ' 用户密码：'
                || user_table (i).USER_PASSWORD);
        END LOOP;

        CLOSE cursor_user;                                  -- 4、关闭游标CLOSE
    ELSE
        DBMS_OUTPUT.PUT_LINE ('打开游标失败');
    END IF;
END;
```

游标变量

使用

```
-- 1、声明游标标量
TYPE type_name IS REF CURSOR [RETURN return_type];
v_cursor_name type_name;

注意 return_type必须是记录类型，可以显示声明，可以用%ROWTYPE隐式声明

-- 2、打开游标标量OPEN
OPEN v_cursor_name FOR select_statement;

-- 3、提取结果FETCH
FETCH v_cursor_name [BULK COLLECT] INTO variable1 [,variable2...]

-- variable 可以是标量、记录和集合
-- 使用标量变量读取游标结果集中的数据
-- 使用记录标量读取游标结果集中的数据
-- 使用集合变量读取游标结果集中的数据

-- 4、关闭游标标量CLOSE
CLOSE v_cursor_name
```

```
20
21  -- 例子
22  DECLARE
23      TYPE t_tablename IS TABLE OF KHB_USER%ROWTYPE;
24
25      user_table      t_tablename;
26
27      TYPE t_cursorname IS REF CURSOR RETURN KHB_USER%ROWTYPE;
28
29      v_cursor_user   t_cursorname;
30  BEGIN
31      OPEN v_cursor_user FOR SELECT * FROM KHB_USER WHERE
    KHB_USER.USER_PASSWORD = '123456';
32
33      IF v_cursor_user%ISOPEN
34      THEN
35          DBMS_OUTPUT.PUT_LINE ('打开游标成功');
36
37          FETCH v_cursor_user BULK COLLECT INTO user_table;
38
39          IF v_cursor_user%FOUND
40          THEN
41              DBMS_OUTPUT.PUT_LINE (
42                  '提取数据成功,行数: ' || v_cursor_user%ROWCOUNT);
43          ELSIF v_cursor_user%NOTFOUND
44          THEN
45              DBMS_OUTPUT.PUT_LINE ('提取数据失败');
46          END IF;
47
48          FOR i IN 1 .. v_cursor_user%ROWCOUNT                --
    user_table.COUNT
49          LOOP
50              DBMS_OUTPUT.PUT_LINE (
51                     '用户名: '
52                  || user_table (i).USER_NAME
53                  || ' 用户工号: '
54                  || user_table (i).USER_ID
55                  || ' 用户密码: '
56                  || user_table (i).USER_PASSWORD);
57          END LOOP;
58
59          CLOSE v_cursor_user;
60      ELSE
61          DBMS_OUTPUT.PUT_LINE ('打开游标失败');
62      END IF;
63  END;
```

10.5 异常处理

在PL/SQL程序中，一般会遇到两种错误：一种是编译错误，这种错误市由PL/SQL引擎来检测，另一种是运行时错误，这种错误需要通过PL/SQL中的异常处理机制进行捕获和处理。

PL/SQL中的异常有两种类型：预定义异常和自定义异常。

预定义异常

自定义异常

```
-- 1、声明异常

-- 2、抛出异常

-- 3、捕获处理异常
EXCEPTION
    WHEN exception_name THEN
        statements1;
    WHEN exception_name THEN
        statement2;
    [WHEN OTHERS THEN
         statements;]

-- 例子
DECLARE
    e_illegalvalue EXCEPTION;                    -- 1、声明自定义异常
    v_count NUMBER;
BEGIN
    SELECT count(*)
    INTO v_count
    FROM KHB_USER
    WHERE KHB_USER.USER_PASSWORD IN ('123456','111111') ;

    IF v_count > 0 THEN
        RAISE e_illegalvalue;                    -- 2、抛出自定义异常
    END IF;

EXCEPTION                                        -- 3、异常捕获和处理
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('查询数据不存在');
    WHEN e_illegalvalue THEN
        DBMS_OUTPUT.PUT_LINE(v_count||'人的密码太简单');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE ('发生其它错误 错误编号：'||SQLCODE||' 错误信息：'||SQLERRM);
END;
```

异常处理机制

```
-- 1、声明部分中异常处理机制

在一个嵌套的PL/SQL语句块中，如果内层的声明部分在赋值时产生了异常，那么该异常会被外层的EXCEPTION部分异常处理语句所捕获。

-- 2、执行部分中异常处理机制

先在当前语句块中寻找对应的异常处理程序，如果没有就向外层寻找。

-- 3、异常处理部分中异常处理机制

如果异常处理部分产生异常，会抛到外层语句块中进行异常处理。

```

异常处理原则

```
-- 1、对捕获的异常进行处理

-- 2、确保没有未处理的异常

-- 3、标记异常发生的位置

-- 4、控制异常处理程序

```

10.6 存储过程

存储过程可以被看作是经过编译的，永久保存在数据中的一组SQL语句，可以提高程序的重用性和扩展性，为程序提供模块化的功能。(相当于void函数)

创建

```
-- 语法规则
CREATE [OR REPLACE] PROCEDURE procedure_name
[(argument1[model] datatype1 [,argument2[model] datatype2] ...)]
{IS | AS}
[local declarations]
BEGIN
    excutable statements
[EXCEPTION
    exception handlers]
END [procedure_name];



-- 例子
CREATE OR REPLACE PROCEDURE procedure_insert_user (
    user_name       IN     KHB_USER.USER_NAME%TYPE,
    user_id         IN     KHB_USER.USER_ID%TYPE,
    user_password   IN     KHB_USER.USER_PASSWORD%TYPE DEFAULT '123123',
    user_count         OUT NUMBER)
AS
    v_count          NUMBER;
    e_illegalValue   EXCEPTION;
BEGIN
    IF LENGTH (user_password) < 6 OR LENGTH (user_password) > 10
    THEN
        RAISE_APPLICATION_ERROR(-20150, 'the length of password is
invalid');
    END IF;

    INSERT INTO KHB_USER (USER_NAME, USER_ID, USER_PASSWORD)
         VALUES (user_name, user_id, user_password);

    SELECT COUNT (DISTINCT KHB_USER.USER_NAME) INTO user_count FROM
KHB_USER;

    SELECT COUNT (*)
      INTO v_count
      FROM KHB_USER
     WHERE KHB_USER.USER_PASSWORD IN ('123456', '111111');

```

```
    IF v_count > 0
    THEN
        RAISE e_illegalValue;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND
    THEN
        DBMS_OUTPUT.PUT_LINE ('未找到数据');
    WHEN e_illegalValue
    THEN
        DBMS_OUTPUT.PUT_LINE (v_count || '人的密码太简单');
    WHEN OTHERS
    THEN
        DBMS_OUTPUT.PUT_LINE (
                '发生其它错误 错误编号：'
            || SQLCODE
            || ' 错误信息：'
            || SQLERRM);
END procedure_insert_user;
```

调用

```
DECLARE
    v_count NUMBER;
BEGIN
    procedure_insert_user('zzz', 'z1523515', '123123',v_count);
    DBMS_OUTPUT.PUT_LINE('员工数：' || v_count);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
```

删除

```
-- 语法规则
DROP procedure_name;

-- 例子
DROP procedure_insert_user;
```

10.7 函数

函数是一个能够返回计算结果值得子程序，不仅可以提高程序得重用性和扩展性，还利于对程序得维护和管理。

创建

```
-- 语法规则
CREATE [OR REPLACE] FUNCTION function_name
[(argument1[model] datatype1 [,argument2[model] datatype2] ...)]
RETURN return_datatype
{IS | AS}
[local declarations]
BEGIN
    excutable statements
```

```
[EXCEPTION
    exception handlers]
END [function_name];

-- 例子
CREATE OR REPLACE FUNCTION f_insert_user (
    user_name       IN      KHB_USER.USER_NAME%TYPE,
    user_id         IN      KHB_USER.USER_ID%TYPE,
    user_password   IN      KHB_USER.USER_PASSWORD%TYPE DEFAULT '123123',
    user_count          OUT NUMBER)
RETURN  KHB_USER%ROWTYPE
AS
    v_count          NUMBER;
    e_illegalValue   EXCEPTION;
    t_user_record    KHB_USER%ROWTYPE;
BEGIN
    IF LENGTH (user_password) < 6 OR LENGTH (user_password) > 10
    THEN
        RAISE_APPLICATION_ERROR(-20150, 'the length of password is
invalid');
    END IF;

    INSERT INTO KHB_USER (USER_NAME, USER_ID, USER_PASSWORD)
         VALUES (user_name, user_id, user_password);

    SELECT COUNT (DISTINCT KHB_USER.USER_NAME)
    INTO user_count
    FROM KHB_USER;

    SELECT *
    INTO t_user_record
    FROM KHB_USER
    WHERE KHB_USER.USER_ID = '78957';

    RETURN t_user_record;

    SELECT COUNT (*)
      INTO v_count
      FROM KHB_USER
     WHERE KHB_USER.USER_PASSWORD IN ('123456', '111111');

    IF v_count > 0
    THEN
        RAISE e_illegalValue;
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND
    THEN
        DBMS_OUTPUT.PUT_LINE ('未找到数据');
    WHEN e_illegalValue
    THEN
        DBMS_OUTPUT.PUT_LINE (v_count || '人的密码太简单');
    WHEN OTHERS
    THEN
        DBMS_OUTPUT.PUT_LINE (
               '发生其它错误 错误编号：'
            || SQLCODE
```

```
            || ' 错误信息：'
            || SQLERRM);
END f_insert_user;

```

调用

```
DECLARE
    v_count NUMBER;
    t_user_record KHB_USER%ROWTYPE;
BEGIN
    t_user_record := f_insert_user('zzz', 'z1523515', '123123',v_count);
    DBMS_OUTPUT.PUT_LINE('员工数：' || v_count);
    DBMS_OUTPUT.PUT_LINE(t_user_record.USER_NAME||' '||
t_user_record.USER_ID || ' ' || t_user_record.USER_PASSWORD);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
```

删除

```
-- 语法规则
DROP function_name;

-- 例子
DROP f_insert_user;
```

10.8 包

包是一个模式对象，通过一定得逻辑组合，可以将相关的类型，常量，变量，存储过程，函数和异常组合在一起。**相当于类**。包由两部分组成：包说明和包体，两个部分是相互独立的，**包体不是必需的**。包说明和包体是相互分离的数据字典对象，在程序运行时，包说明必须要首先编译成功，包体才能通过编译。如果编译不成功，那么包体也不能被成功编译。

在PL/SQL中，存储过程和函数都可以在包中重载。

创建

```
-- 创建包说明
-- 语法规则
CREATE [OR REPLACE] PACKAGE package_name
{IS | AS}
    [variable_declaration...]           -- 声明变量
    [cursor_declaration...]             -- 声明游标
    [exception_declaration...]          -- 声明异常
    [object_declaration...]             -- 声明对象
    [function_declaration...]           -- 声明函数
    [procedure_declaration...]          -- 声明过程
END [package_name];

-- 例子
CREATE OR REPLACE PACKAGE khb_user_package
AS
e_illegalValue  EXCEPTION;
```

```
17
18  PROCEDURE p_ins_user(
19      user_name       IN      KHB_USER.USER_NAME%TYPE,
20      user_id         IN      KHB_USER.USER_ID%TYPE,
21      user_password   IN      KHB_USER.USER_PASSWORD%TYPE DEFAULT '123123'
22  );
23
24  PROCEDURE p_del_user(
25      user_name       IN      KHB_USER.USER_NAME%TYPE
26  );
27
28  PROCEDURE p_del_user(
29      user_name       IN      KHB_USER.USER_NAME%TYPE,
30      user_id         IN      KHB_USER.USER_ID%TYPE
31  );
32
33  FUNCTION f_sel_user(
34      user_id         IN      KHB_USER.USER_ID%TYPE
35  )
36  RETURN KHB_USER%ROWTYPE;
37
38  END khb_user_package;
39
40
41  -- 创建包体
42  CREATE [OR REPLACE] PACKAGE BODY package_name
43  {IS | AS}
44      -- 公有变量、常量、游标、自定义数据类型、存储过程或者函数的实现
45      -- 定义私有的变量、常量、游标、自定义数据类型、存储过程和函数
46  END [package_name];
47
48  -- 例子
49  CREATE OR REPLACE PACKAGE BODY khb_user_package
50  IS
51      /*****************************
    [p_ins_user]*********************************/
52      PROCEDURE p_ins_user (
53          user_name       IN KHB_USER.USER_NAME%TYPE,
54          user_id         IN KHB_USER.USER_ID%TYPE,
55          user_password   IN KHB_USER.USER_PASSWORD%TYPE DEFAULT '123123')
56      AS
57      BEGIN
58          IF LENGTH (user_password) < 6 OR LENGTH (user_password) > 10
59          THEN
60              RAISE e_illegalValue;
61          END IF;
62
63          INSERT INTO KHB_USER (USER_NAME, USER_ID, USER_PASSWORD)
64               VALUES (user_name, user_id, user_password);
65      EXCEPTION
66          WHEN e_illegalValue
67          THEN
68              DBMS_OUTPUT.PUT_LINE ('数据不合法');
69              ROLLBACK;
70          WHEN OTHERS
71          THEN
72              DBMS_OUTPUT.PUT_LINE ('其他错误 ' || SQLERRM);
73              ROLLBACK;
```

```
    END p_ins_user;

    /*****************************
[p_del_user]***********************************/
    PROCEDURE p_del_user (user_name IN KHB_USER.USER_NAME%TYPE)
    AS
    BEGIN
        DELETE FROM KHB_USER
              WHERE KHB_USER.USER_NAME = user_name;
    EXCEPTION
        WHEN NO_DATA_FOUND
        THEN
            DBMS_OUTPUT.PUT_LINE ('未找到数据');
            ROLLBACK;
        WHEN OTHERS
        THEN
            DBMS_OUTPUT.PUT_LINE ('其他错误 ' || SQLERRM);
            ROLLBACK;
    END p_del_user;

    /*****************************
[p_del_user]***********************************/
    PROCEDURE p_del_user (user_name   IN KHB_USER.USER_NAME%TYPE,
                          user_id     IN KHB_USER.USER_ID%TYPE)
    AS
    BEGIN
        DELETE FROM KHB_USER
              WHERE     KHB_USER.USER_NAME = user_name
                    AND KHB_USER.USER_ID = user_id;
    EXCEPTION
        WHEN NO_DATA_FOUND
        THEN
            DBMS_OUTPUT.PUT_LINE ('未找到数据');
            ROLLBACK;
        WHEN OTHERS
        THEN
            DBMS_OUTPUT.PUT_LINE ('其他错误 ' || SQLERRM);
            ROLLBACK;
    END p_del_user;

    /*****************************
[f_sel_user]***********************************/
    FUNCTION f_sel_user (user_id IN KHB_USER.USER_ID%TYPE)
        RETURN KHB_USER%ROWTYPE
    AS
        t_user_record   KHB_USER%ROWTYPE;
    BEGIN
        SELECT *
          INTO t_user_record
          FROM KHB_USER
         WHERE KHB_USER.USER_ID = user_id;

        RETURN t_user_record;
    EXCEPTION
        WHEN NO_DATA_FOUND
        THEN
            DBMS_OUTPUT.PUT_LINE ('未找到数据');
            ROLLBACK;
```

```
        WHEN OTHERS
        THEN
            DBMS_OUTPUT.PUT_LINE ('其他错误 ' || SQLERRM);
            ROLLBACK;
    END f_sel_user;
END khb_user_package;

```

调用

```
DECLARE
    t_user_record KHB_USER%ROWTYPE;

BEGIN
    khb_user_package.p_ins_user('zzz','123','123123');
    khb_user_package.p_ins_user('zzz','123123','123123');
    khb_user_package.p_ins_user('aaa','1234','123123');

    khb_user_package.p_del_user('aaa');
    khb_user_package.p_del_user('zzz','123123');

    t_user_record := khb_user_package.f_sel_user('10078880');
    DBMS_OUTPUT.PUT_LINE('姓名：'||t_user_record.USER_NAME);
    DBMS_OUTPUT.PUT_LINE('工号：'||t_user_record.USER_ID);
    DBMS_OUTPUT.PUT_LINE('密码：'||t_user_record.USER_PASSWORD);

END;
```

删除

```
-- 删除包体
DROP PACKAGE BODY packagebody_name;

-- 删除包说明和包体
DROP PACKAGE package_name;
```

系统包

DBMS_ALERT

该包用于生成并发送报警信息，当特定的数据库值发生变化时，将报警信息传递给应用程序。

序号	子程序	说明
1	过程 SIGNAL	
2	过程 WAITANY和WAITONE	
3	过程 REGISTER	
4	过程 REMOVE和REMOVEALL	
5	过程 SET_DEFAULTS	

DBMS_OUTPUT

该包用于输入和输出信息，方便程序的测试和调试。

序号	子程序	说明
1	过程 ENABLE和DISABLE	
2	过程 PUT和PUT_LINE	
3	过程 GET_LINE和GET_LINES	

DBMS_PIPE

该包用于在同一例程的不同会话之间之间进行管道通信，管道是异步的，一旦相关管道发送一条消息。就不能取消它。

序号	子程序	说明
1	函数 CREATE_PIPE	
2	过程 PACK_MESSAGE	
3	函数 SEND_MESSAGE	
4	函数 RECEIVE_MESSAGE	
5	过程 UNPACK_MESSAGE	
6	函数 NEXT_ITEM_TYP	
7	函数 REMOVE_PIPE	
8	过程 PURGE	

DBMS_JOB

该包用于安排和管理PL/SQL语句块，是Oracle数据库可以在指定的时间执行特定的任务。

序号	子程序	说明
1	过程 SUNMIT	
2	过程 RUN	
3	过程 CHANGE	
4	过程 REMOVE	

DBMS_LOB

该包用于处理LOB数据类型的数据。

序号	子程序	说明
1	过程 APPEND	
2	函数 COMPARE	
3	函数 GETLENGT	
4	过程 COPY	

10.9 触发器

触发器存储在数据库中，它是一种特殊的被隐式执行的存储过程。当有触发事件发生时，Oracle数据就激发触发器，隐式执行触发器相应的代码。

触发器类型：

- DML触发器(数据操作语言触发器)
- DDL触发器(数据定义语言触发器)
- INSTEAD OF 触发器
- 复合触发器
- 事件触发器

触发器用途：

- 增强数据库的数据安全
- 实现复杂数据完整性
- 实现参照完整性
- 实现数据审计

创建DML触发器

```
-- 1、创建语句触发器（表触发器）
----- 执行DML语句时，无论改DML语句会影响多少行数据记录，语句触发器只会被激发一次。
----- 语法规则
CREATE OR REPLACE TRIGGER trigger_name
{ BEFORE|AFTER} INSERT {OR DELETE OR UPDATE}
ON table_name
{DECLARE
    declaration_statement;}
BEGIN
    execution_statements;
END [trigger_name];

---- 例子
CREATE OR REPLACE TRIGGER trig_khb_user_editdata
    BEFORE INSERT OR UPDATE OR DELETE
    ON KHB_USER
DECLARE
    v_data   NUMBER;
BEGIN
    SELECT TO_CHAR (SYSDATE, 'HH24') INTO v_data FROM DUAL;

    IF v_data < 8 OR v_data > 17
    THEN
        RAISE_APPLICATION_ERROR (-20010,
                                 '不能在下班时间更新KHB_USER表');
```

```
    END IF;

    SELECT TO_CHAR (SYSDATE, 'DY') INTO v_data FROM DUAL;

    IF v_data IN ('SAT', 'SUN')
    THEN
        RAISE_APPLICATION_ERROR (-20011,
                                 '不能在周末时间更新KHB_USER表');
    END IF;
END trig_khb_user_editdata;

-- 2、创建行触发器
---- 执行DML语句时，每作用一行就会被触发一次的触发器。
----- 语法规则
CREATE OR REPLACE TRIGGER trigger_name
{ BEFORE|AFTER} INSERT {OR DELETE OR UPDATE}
ON table_name [REFERENCING OLD AS old | NEW AS new]
FOR EACH ROW
[WHEN (logocal_expression)]
{DECLARE
    declaration_statement;}
BEGIN
    execution_statements;
END [trigger_name];

---- 例子
CREATE OR REPLACE TRIGGER trig_khb_user_backup
    AFTER INSERT OR UPDATE OR DELETE
    ON KHB_USER
    FOR EACH ROW
DECLARE
    v_count   NUMBER;
BEGIN
    CASE
        WHEN INSERTING
        THEN
            INSERT INTO KHB_USER_BK
                 VALUES (:new.USER_NAME,
                         :new.USER_ID,
                         :new.USER_PASSWORD,
                         :new.QUES,
                         :new.ANS);

            SELECT COUNT(*) INTO v_count FROM KHB_USER_BK;
            DBMS_OUTPUT.PUT_LINE (v_count);
        WHEN UPDATING
        THEN
            UPDATE KHB_USER_BK
               SET USER_NAME = :new.USER_NAME,
                   USER_ID = :new.USER_ID,
                   USER_PASSWORD = :new.USER_PASSWORD,
                   QUES = :new.QUES,
                   ANS = :new.ANS
             WHERE USER_ID = :old.USER_ID;
        WHEN DELETING
        THEN
            UPDATE KHB_USER_BK
               SET USER_NAME = USER_NAME || '(离职)'
```

```
            WHERE USER_ID = :old.USER_ID;
    END CASE;
END trig_khb_user_backup ;

---- 测试
INSERT INTO KHB_USER VALUES('AAA','123','1234123','AAA','AAA');

UPDATE KHB_USER
SET USER_PASSWORD = '147258369'
WHERE USER_NAME = 'AAA';

DELETE FROM KHB_USER
WHERE USER_NAME = 'AAA';
```

创建DDL触发器

```
----- 语法规则
CREATE OR REPLACE TRIGGER trigger_name
{ BEFORE|AFTER} DLL
ON table_name
[WHEN (logocal_expression)]
{DECLARE
    declaration_statement;}
BEGIN
    execution_statements;
END [trigger_name];

---- 例子
CREATE OR REPLACE TRIGGER trigger_drop
    BEFORE DROP
    ON PDAHC.SCHEMA
BEGIN
    RAISE_APPLICATION_ERROR (-20022, '无效的删除操作');
END trigger_drop;
```

创建INSTEAD OF触发器

```
-- 在视图中执行DML操作时需要一定的限制，简单视图可以执行INSERT,UPDATE和DELETE操作，但是在基于分组函数、DISTINCT、连接查询和集合查询的复杂视图中，不能执行INSERT,UPDATE和DELETE操作。为了能在复杂视图中也可以执行INSERT,UPDATE和DELETE操作，就需要创建INSTEAD OF触发器。
-- INSTEAD OF触发器 只是适用于视图

---- 语法规则
CREATE OR REPLACE TRIGGER trigger_name
INSTEAD OF INSERT [OR DELETE OR UPDATE]
ON view_name
[WHEN (logocal_expression)]
{DECLARE
    declaration_statement;}
BEGIN
    execution_statements;
END [trigger_name];

---- 例子

```

创建事件触发器

```
-- 事件触发器包括系统事件触发器和用户事件触发器。
--           系统事件包括数据库的启动和登录、数据库的关闭和退出、服务器错误等
--           用户事件包括用户登录和注销、DDL事件等

-- 1、创建系统事件触发器
---- 语法规则
CREATE OR REPLACE TRIGGER trigger_name
{BEFORE OR AFTER} database_event
ON [database | schema]
{DECLARE
    declaration_statement;}
BEGIN
    execution_statements;
END [trigger_name];
---- 例子

-------数据库启动
CREATE OR REPLACE TRIGGER trig_startup
    AFTER STARTUP
    ON DATABASE
BEGIN
    INSERT INTO T_LOG
         VALUES (ORA_SYSEVENT, SYSDATE, ORA_LOGIN_USER);
END;
-------数据库关闭
CREATE OR REPLACE TRIGGER trig_shutdown
    BEFORE SHUTDOWN
    ON DATABASE
BEGIN
    INSERT INTO T_LOG
         VALUES (ORA_SYSEVENT, SYSDATE, ORA_LOGIN_USER);
END;


-- 2、创建用户事件触发器
---- 语法规则
CREATE OR REPLACE TRIGGER trigger_name
{BEFORE OR AFTER} database_event
ON [database | schema]
{DECLARE
    declaration_statement;}
BEGIN
    execution_statements;
END [trigger_name];

---- 例子

-------用户登录
CREATE OR REPLACE TRIGGER trig_logon
    AFTER LOGON
    ON DATABASE
BEGIN
    INSERT INTO T_LOG
         VALUES (ORA_SYSEVENT, SYSDATE, ORA_LOGIN_USER);
END;
```

```
56
57  -------用户注销
58  CREATE OR REPLACE TRIGGER trig_logout
59      BEFORE LOGOFF
60      ON DATABASE
61  BEGIN
62      INSERT INTO T_LOG
63          VALUES (ORA_SYSEVENT, SYSDATE, ORA_LOGIN_USER);
64  END;
```

管理触发器

```
1   -- 禁用触发器
2   ALTER TRIGGER trigger_name DISABLE;
3   ALTER TRIGGER trigger_name DISABLE ALL TRIGGERS;
4
5   -- 激发触发器
6   ALTER TRIGGER trigger_name ENABLE;
7   ALTER TRIGGER trigger_name ENABLE ALL TRIGGERS;
8
9   -- 重新编译触发器
10  ALTER TRIGGER trigger_name COMPILE;
11
12  -- 删除触发器
13  DROP TRIGGER trigger_name;
```

问题集锦