

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化计算机语言。SQL 对大小写不敏感且语句后需要加分号。

SQL主要包括以下几个部分:

- 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

```
1 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:

```
1 CREATE TABLE table_name
2 (
3     column_name1    datatype1    [constraint_condition1][,
4     column_name2    datatype2    [constraint_condition2]]...
5 );
```

例子 - ORACLE

```
1 CREATE TABLE PDAHC.T_STUDENTS
2 (
3     ID      VARCHAR2(10 BYTE),
4     NAME    VARCHAR2(10 BYTE),
5     AGE     INTEGER,
6     SEX     CHAR(2 BYTE),
7     BIRTH   DATE
8 );
9
```

例子 - MYSQL

```
1  -- 学生信息表
2  CREATE TABLE T_STUDENTS
3  (
4      stuID  VARCHAR(10) PRIMARY KEY,
5      NAME   VARCHAR(10) NOT NULL,
6      AGE    INT NOT NULL,
7      SEX    CHAR(2) NOT NULL,
8      BIRTH  DATE NOT NULL
9  );
10
11 -- 学生成绩表
12 CREATE TABLE T_RESULTS
13 (
14     stuID  VARCHAR(15),
15     curID  VARCHAR(15),
16     result DOUBLE,
17     PRIMARY KEY(stuID, curID),
18     FOREIGN KEY(stuID) REFERENCES T_STUDENTS(stuID) ON DELETE CASCADE
19 );
20
21 -- 课程信息表
22 CREATE TABLE T_CURRICULUM
```

```

23 (
24     curID          VARCHAR(15) PRIMARY KEY,
25     curNAME        VARCHAR(15),
26     CREDIT          INT,
27     LEARNTIME       INT,
28     TEACHERNAME     VARCHAR(15)
29 );
30
31 -- 教师信息表
32 CREATE TABLE T_TEACHERS
33 (
34     teaID   VARCHAR(15) PRIMARY KEY,
35     teaNAME VARCHAR(15),
36     AGE     INT,
37     SEX     CHAR(2),
38     depID   VARCHAR(15),
39     depNAME VARCHAR(15),
40     PROFESSION VARCHAR(15),
41     SALARY  INT(5),
42     PENSION DOUBLE
43 );
44
45 -- 院系信息表
46 CREATE TABLE T_DEPTS
47 (
48     deptID   VARCHAR(15),
49     deptName VARCHAR(15)
50 );
51

```

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

```

1 //语法
2 ALTER TABLE table_name ADD(column_name datatype [constraint_conditon]);
3
4 //MYSQL 向教师信息表中增加一个地址信息的列
5 mysql> ALTER TABLE T_TEACHERS ADD ADDRESS VARCHAR(50);
6 Query OK, 0 rows affected
7 Records: 0 Duplicates: 0 Warnings: 0

```

删除一列

```
1 //语法
2 ALTER TABLE table_name DROP column_name;
3
4 //MYSQL 从教师信息表中删除一个地址信息的列
5 mysql> ALTER TABLE T_TEACHERS DROP ADDRESS;
6 Query OK, 0 rows affected
7 Records: 0 Duplicates: 0 Warnings: 0
```

修改一列

```
1 //语法
2 ALTER TABLE table_name MODIFY column_name datatype;
3
4 //MYSQL 从教师信息表中删除一个地址信息的列
5 mysql> ALTER TABLE T_TEACHERS DROP ADDRESS;
6 Query OK, 0 rows affected
7 Records: 0 Duplicates: 0 Warnings: 0
```

增加一个约束条件

```
1 //语法
2 ALTER TABLE table_name ADD constraint_type(column_name);
3
4 //MYSQL 对教师信息表中AGE列增加一个CHECK约束 在15到60之间
5 mysql> ALTER TABLE T_TEACHERS ADD CHECK(AGE BETWEEN 15 AND 60);
6 Query OK, 0 rows affected
7 Records: 0 Duplicates: 0 Warnings: 0
```

删除一个约束条件

```
1 //语法
2 ALTER TABLE table_name DROP constraint_type;
3
4 //MYSQL 对教师信息表中删除CHECK约束
5 mysql> ALTER TABLE T_TEACHERS DROP CHECK;
6 Query OK, 0 rows affected
7 Records: 0 Duplicates: 0 Warnings: 0
```

增加一个索引

```
1 //语法
2 ALTER TABLE table_name ADD INDEX index_name(column_name1[,column_name2]...);
3
4 //MYSQL 从教师信息表中增加一个索引
5 mysql> ALTER TABLE T_TEACHERS ADD INDEX i_age(teaID);
6 Query OK, 0 rows affected
7 Records: 0 Duplicates: 0 Warnings: 0
```

删除一个索引

```

1 //语法
2 ALTER TABLE table_name DROP INDEX index_name;
3
4 //MYSQL 从教师信息表中增加一个索引
5 mysql> ALTER TABLE T_TEACHERS ADD INDEX i_age(teaID);
6 Query OK, 0 rows affected
7 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	

```

1 CREATE INDEX 索引名 ON 表名 (列名1 , [列名2]...)
2
3 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 ---- 1、查询表中全部列的记录
2 -- 语法
3 SELECT * FROM 表名或视图名;
4
5 -- MYSQL
6 mysql> SELECT * FROM T_STUDENTS;
7 +-----+-----+-----+-----+-----+
8 | stuID | NAME | AGE | SEX | BIRTH |
9 +-----+-----+-----+-----+-----+
10 | S000001 | 赵大 | 25 | 男 | 1995-04-12 |
11 | S000002 | 钱二 | 21 | 男 | 1999-05-13 |
12 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
13 | S000004 | 李四 | 24 | 男 | 1996-05-23 |
14 | S000005 | 周五 | 20 | 男 | 2000-05-03 |
15 +-----+-----+-----+-----+-----+
16 5 rows in set
17
18
19 ---- 2、查询表中指定列的记录
20 -- 语法
21 SELECT 目标列[,目标列,...] FROM 表名或视图名[,表名或视图名];

```

```

22
23 -- MYSQL
24 mysql> SELECT stuID,NAME FROM T_STUDENTS;
25 +-----+-----+
26 | stuID   | NAME |
27 +-----+-----+
28 | s000001 | 赵大 |
29 | s000002 | 钱二 |
30 | s000003 | 孙三 |
31 | s000004 | 李四 |
32 | s000005 | 周五 |
33 +-----+-----+
34 5 rows in set
35
36 ---- 3、查询表中不重复列的记录
37 -- 语法
38 SELECT DISTINCT
39 目标列[,目标列,...] FROM 表名或视图名[,表名或视图名];
40
41 -- MYSQL
42 mysql> SELECT NAME FROM T_STUDENTS;
43 +-----+
44 | NAME |
45 +-----+
46 | 赵大 |
47 | 钱二 |
48 | 孙三 |
49 | 李四 |
50 | 周五 |
51 | 赵大 |
52 +-----+
53 6 rows in set
54
55 ---- 4、使用列别名查询
56 -- 语法
57 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];
58
59 -- MYSQL
60 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;
61 +-----+-----+-----+-----+-----+
62 | 学号   | 姓名 | 年龄 | 性别 | 出生日期   |
63 +-----+-----+-----+-----+-----+
64 | s000001 | 赵大 | 25 | 男 | 1995-04-12 |
65 | s000002 | 钱二 | 21 | 男 | 1999-05-13 |
66 | s000003 | 孙三 | 24 | 女 | 1996-08-06 |
67 | s000004 | 李四 | 24 | 男 | 1996-05-23 |
68 | s000005 | 周五 | 20 | 男 | 2000-05-03 |
69 | s000006 | 赵大 | 25 | 男 | 1995-04-12 |
70 +-----+-----+-----+-----+-----+
71 6 rows in set
72
73 ---- 5、连接字段查询
74 -- 语法
75 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];
76

```

```

77  -- MYSQL 使用CONCAT函数
78  SELECT CONCAT(CONCAT(stuID,""),CONCAT(NAME,""), CONCAT(SEX,""),
CONCAT(BIRTH,""),AGE) STUDENT_INFO FROM T_STUDENTS;
79  +-----+
80  | STUDENT_INFO |
81  +-----+
82  | S000001,赵大,男,1995-04-12,25 |
83  | S000002,钱二,男,1999-05-13,21 |
84  | S000003,孙三,女,1996-08-06,24 |
85  | S000004,李四,男,1996-05-23,24 |
86  | S000005,周五,男,2000-05-03,20 |
87  | S000006,赵大,男,1995-04-12,25 |
88  +-----+
89  6 rows in set

```

5.2 WHERE条件查询

```

1  ---- 1、比较查询
2  --- 算术比较运算符 > >= = < <= != <> !> !<
3  -- 语法
4  WHERE 字段 算术比较运算符 值
5  -- MYSQL
6  mysql> SELECT * FROM T_STUDENTS WHERE AGE > 22;
7  +-----+
8  | stuID | NAME | AGE | SEX | BIRTH |
9  +-----+
10 | S000001 | 赵大 | 25 | 男 | 1995-04-12 |
11 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
12 | S000004 | 李四 | 24 | 男 | 1996-05-23 |
13 | S000006 | 赵大 | 25 | 男 | 1995-04-12 |
14 +-----+
15 4 rows in set
16
17 --- 范围查询 BETWEEN..AND... IN
18 -- 语法
19 WHERE 字段 BETWEEN 值1 AND 值2
20
21 WHERE 字段 IN (值1,值2,...)
22 -- MYSQL
23 mysql> SELECT * FROM T_STUDENTS WHERE AGE BETWEEN 20 AND 24;
24 +-----+
25 | stuID | NAME | AGE | SEX | BIRTH |
26 +-----+
27 | S000002 | 钱二 | 21 | 男 | 1999-05-13 |
28 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
29 | S000004 | 李四 | 24 | 男 | 1996-05-23 |
30 | S000005 | 周五 | 20 | 男 | 2000-05-03 |
31 +-----+
32 4 rows in set
33
34 mysql> SELECT * FROM T_STUDENTS WHERE AGE BETWEEN 20 AND 24;
35 +-----+
36 | stuID | NAME | AGE | SEX | BIRTH |
37 +-----+
38 | S000002 | 钱二 | 21 | 男 | 1999-05-13 |
39 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
40 | S000004 | 李四 | 24 | 男 | 1996-05-23 |

```

```

41 | S000005 | 周五 | 20 | 男 | 2000-05-03 |
42 +-----+-----+-----+-----+-----+
43 4 rows in set
44
45 ---- 2、逻辑查询 AND OR NOT
46 -- 语法
47 WHERE 条件1 逻辑运算符 条件2 ...
48 -- MYSQL
49 SELECT * FROM T_STUDENTS WHERE BIRTH > '1996-00-00' AND AGE > 21;
50 +-----+-----+-----+-----+-----+
51 | stuID | NAME | AGE | SEX | BIRTH |
52 +-----+-----+-----+-----+-----+
53 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
54 | S000004 | 李四 | 24 | 男 | 1996-05-23 |
55 +-----+-----+-----+-----+-----+
56
57 ---- 3、模糊查询 LIKE REGEXP
58 通配符:
59     匹配单个字符 : '_'
60     匹配0或多个字符 : '%'
61 -- 语法
62 WHERE 字段 LIKE 模糊值
63 -- MYSQL
64 mysql> SELECT * FROM T_STUDENTS WHERE NAME LIKE '%三';
65 +-----+-----+-----+-----+-----+
66 | stuID | NAME | AGE | SEX | BIRTH |
67 +-----+-----+-----+-----+-----+
68 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
69 | S000007 | 吴三 | 25 | 男 | 1995-02-22 |
70 +-----+-----+-----+-----+-----+
71 2 rows in set
72
73 mysql> SELECT * FROM T_STUDENTS WHERE NAME LIKE '_三';
74 +-----+-----+-----+-----+-----+
75 | stuID | NAME | AGE | SEX | BIRTH |
76 +-----+-----+-----+-----+-----+
77 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
78 | S000007 | 吴三 | 25 | 男 | 1995-02-22 |
79 +-----+-----+-----+-----+-----+
80 2 rows in set
81
82 -- 转移字符 ESCAPE定义转义符
83 mysql> SELECT * FROM T_STUDENTS WHERE NAME LIKE '%$_%' ESCAPE '$';
84 +-----+-----+-----+-----+-----+
85 | stuID | NAME | AGE | SEX | BIRTH |
86 +-----+-----+-----+-----+-----+
87 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 |
88 +-----+-----+-----+-----+-----+
89 1 row in set
90
91 -- MYSQL REGEXP关键字 正则表达式 进行模糊匹配
92 mysql> SELECT * FROM T_STUDENTS WHERE NAME REGEXP '[_]+' ;
93 +-----+-----+-----+-----+-----+
94 | stuID | NAME | AGE | SEX | BIRTH |
95 +-----+-----+-----+-----+-----+
96 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 |
97 +-----+-----+-----+-----+-----+
98 1 row in set

```


5.3 排序与分组查询

```
1  ---- 常见的聚合函数:
2  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
3  -- MYSQL
4  mysql> SELECT COUNT(AGE),MAX(AGE),MIN(AGE),AVG(AGE),SUM(AGE) FROM
   T_STUDENTS;
5  +-----+-----+-----+-----+-----+
6  | COUNT(AGE) | MAX(AGE) | MIN(AGE) | AVG(AGE) | SUM(AGE) |
7  +-----+-----+-----+-----+-----+
8  |          8 |        25 |        20 | 23.3750 |    187    |
9  +-----+-----+-----+-----+-----+
10 1 row in set
11
12 ---- ORDER BY排序
13 -- 1、ASC 升序[默认排序方式] DESC 降序 2、必须为最后一个子句
14 -- 语法
15 ORDER BY 列1 [,列2,...] [ASC|DESC]
16 -- MYSQL
17 mysql> SELECT * FROM T_STUDENTS ORDER BY AGE DESC;
18 +-----+-----+-----+-----+-----+
19 | stuID | NAME | AGE | SEX | BIRTH |
20 +-----+-----+-----+-----+-----+
21 | S000001 | 赵大 | 25 | 男 | 1995-04-12 |
22 | S000006 | 赵大 | 25 | 男 | 1995-04-12 |
23 | S000007 | 吴三 | 25 | 男 | 1995-02-22 |
24 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
25 | S000004 | 李四 | 24 | 男 | 1996-05-23 |
26 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 |
27 | S000002 | 钱二 | 21 | 男 | 1999-05-13 |
28 | S000005 | 周五 | 20 | 男 | 2000-05-03 |
29 +-----+-----+-----+-----+-----+
30 8 rows in set
31
32 ---- GROUP BY分组
33 -- 语法
34 GROUP BY 列1 [,列2,...] [HAVING 条件表示式]
35 -- MYSQL
36 mysql> SELECT SEX ,COUNT(stuID) FROM T_STUDENTS GROUP BY SEX;
37 +-----+-----+
38 | SEX | COUNT(stuID) |
39 +-----+-----+
40 | 女 | 2 |
41 | 男 | 6 |
42 +-----+-----+
43 2 rows in set
44
45 mysql> SELECT AGE ,COUNT(stuID) FROM T_STUDENTS GROUP BY AGE;
46 +-----+-----+
47 | AGE | COUNT(stuID) |
48 +-----+-----+
49 | 20 | 1 |
50 | 21 | 1 |
51 | 23 | 1 |
52 | 24 | 2 |
```

```

53 | 25 | 3 |
54 +-----+
55 5 rows in set
56
57 mysql> SELECT AGE ,COUNT(stuID),SEX,COUNT(stuID) FROM T_STUDENTS GROUP BY
AGE,SEX;
58 +-----+-----+-----+-----+
59 | AGE | COUNT(stuID) | SEX | COUNT(stuID) |
60 +-----+-----+-----+-----+
61 | 20 | 1 | 男 | 1 |
62 | 21 | 1 | 男 | 1 |
63 | 23 | 1 | 女 | 1 |
64 | 24 | 1 | 女 | 1 |
65 | 24 | 1 | 男 | 1 |
66 | 25 | 3 | 男 | 3 |
67 +-----+-----+-----+-----+
68 6 rows in set
69
70 mysql> SELECT AGE ,COUNT(stuID),SEX,COUNT(stuID) FROM T_STUDENTS GROUP BY
AGE,SEX HAVING AGE >= 23;
71 +-----+-----+-----+-----+
72 | AGE | COUNT(stuID) | SEX | COUNT(stuID) |
73 +-----+-----+-----+-----+
74 | 23 | 1 | 女 | 1 |
75 | 24 | 1 | 女 | 1 |
76 | 24 | 1 | 男 | 1 |
77 | 25 | 3 | 男 | 3 |
78 +-----+-----+-----+-----+
79 4 rows in set
80
81 ---- 限制结果集函数
82 -- MYSQL LIMIT offset,n
83 mysql> SELECT * FROM T_STUDENTS ORDER BY AGE ;
84
85 +-----+-----+-----+-----+-----+
86 | stuID | NAME | AGE | SEX | BIRTH |
87 +-----+-----+-----+-----+-----+
88 | S000005 | 周五 | 20 | 男 | 2000-05-03 |
89 | S000002 | 钱二 | 21 | 男 | 1999-05-13 |
90 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 |
91 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
92 | S000004 | 李四 | 24 | 男 | 1996-05-23 |
93 | S000001 | 赵大 | 25 | 男 | 1995-04-12 |
94 | S000006 | 赵大 | 25 | 男 | 1995-04-12 |
95 | S000007 | 吴三 | 25 | 男 | 1995-02-22 |
96 +-----+-----+-----+-----+-----+
97 8 rows in set
98
99 mysql> SELECT * FROM T_STUDENTS ORDER BY AGE LIMIT 1,3;
100
101 +-----+-----+-----+-----+-----+
102 | stuID | NAME | AGE | SEX | BIRTH |
103 +-----+-----+-----+-----+-----+
104 | S000002 | 钱二 | 21 | 男 | 1999-05-13 |
105 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 |
106 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
107 +-----+-----+-----+-----+-----+
108 3 rows in set

```

```

109
110 -- ORACL ROWBUM
111 SELECT * FROM KHB_USER WHERE ROWNUM <= 5 ORDER BY KHB_USER.USER_ID;
112 /*
113 沈晓芳 10012515    111111  我是个勤劳的小蜜蜂    是!!!
114 苏伟   1033242 111111  我是个勤劳的小蜜蜂    是
115 q     54424   628142  我是个勤劳的小蜜蜂    是!
116 许海英 78957    78957   我是个勤劳的小蜜蜂    是!
117 王丽君 2013356 222555  我是个勤劳的小蜜蜂    是! \
118 */

```

5.4 连接与集合查询

5.4.1 连接查询

1、内连接

```

1  ---- 等值连接
2  -- 方法一 WHERE子句
3  SELECT 表1.字段 , 表2.字段 ...
4  FROM 表1 , 表2
5  WHERE 表1.字段1 = 表2.字段2
6  -- MYSQL
7  mysql> SELECT
8  T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result FROM
9  T_STUDENTS ,T_RESULTS WHERE T_STUDENTS.stuID = T_RESULTS.stuID;
10
11 +-----+-----+-----+-----+
12 | stuID   | NAME  | curID  | result |
13 +-----+-----+-----+-----+
14 | S000001 | 赵大  | C000001 | 82     |
15 | S000001 | 赵大  | C000002 | 92     |
16 | S000001 | 赵大  | C000003 | 100    |
17 | S000002 | 钱二  | C000001 | 82     |
18 | S000002 | 钱二  | C000002 | 94     |
19 | S000002 | 钱二  | C000003 | 46     |
20 | S000003 | 孙三  | C000001 | 82     |
21 | S000003 | 孙三  | C000002 | 96     |
22 | S000003 | 孙三  | C000003 | 46     |
23 | S000004 | 李四  | C000001 | 89     |
24 | S000004 | 李四  | C000002 | 64     |
25 | S000004 | 李四  | C000003 | 58     |
26 +-----+-----+-----+-----+
27 12 rows in set
28
29 -- 方法二 ON子句
30 SELECT 表1.字段 , 表2.字段 ...
31 FROM 表1 JOIN 表2
32 ON 表1.字段1 = 表2.字段2
33 -- MYSQL
34 SELECT T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result
35 FROM T_STUDENTS JOIN T_RESULTS ON T_STUDENTS.stuID = T_RESULTS.stuID;
36
37 +-----+-----+-----+-----+
38 | stuID   | NAME  | curID  | result |
39 +-----+-----+-----+-----+
40 | S000001 | 赵大  | C000001 | 82     |
41 | S000001 | 赵大  | C000002 | 92     |
42 | S000001 | 赵大  | C000003 | 100    |

```

```

38 | S000002 | 钱二 | C000001 | 82 |
39 | S000002 | 钱二 | C000002 | 94 |
40 | S000002 | 钱二 | C000003 | 46 |
41 | S000003 | 孙三 | C000001 | 82 |
42 | S000003 | 孙三 | C000002 | 96 |
43 | S000003 | 孙三 | C000003 | 46 |
44 | S000004 | 李四 | C000001 | 89 |
45 | S000004 | 李四 | C000002 | 64 |
46 | S000004 | 李四 | C000003 | 58 |
47 +-----+-----+-----+-----+
48 12 rows in set
49
50 -- 方法三 USING子句
51 SELECT 表1.字段 , 表2.字段 ...
52 FROM 表1 JOIN 表2
53 USING (关联字段)
54 -- MYSQL
55 SELECT T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result
56 FROM T_STUDENTS JOIN T_RESULTS USING (stuID);
57 SELECT T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result
58 FROM T_STUDENTS JOIN T_RESULTS USING (stuID);
59 +-----+-----+-----+-----+
60 | stuID | NAME | curID | result |
61 +-----+-----+-----+-----+
62 | S000001 | 赵大 | C000001 | 82 |
63 | S000001 | 赵大 | C000002 | 92 |
64 | S000001 | 赵大 | C000003 | 100 |
65 | S000002 | 钱二 | C000001 | 82 |
66 | S000002 | 钱二 | C000002 | 94 |
67 | S000002 | 钱二 | C000003 | 46 |
68 | S000003 | 孙三 | C000001 | 82 |
69 | S000003 | 孙三 | C000002 | 96 |
70 | S000003 | 孙三 | C000003 | 46 |
71 | S000004 | 李四 | C000001 | 89 |
72 | S000004 | 李四 | C000002 | 64 |
73 | S000004 | 李四 | C000003 | 58 |
74 +-----+-----+-----+-----+
75 12 rows in set
76
77 ----- 非等值连接
78 -- MYSQL
79 mysql> SELECT
80 T_RESULTS.stuID,T_STUDENTS.NAME,T_RESULTS.curID,T_RESULTS.result FROM
81 T_STUDENTS ,T_RESULTS WHERE T_STUDENTS.stuID = T_RESULTS.stuID AND
82 T_RESULTS.result > 90;
83 +-----+-----+-----+-----+
84 | stuID | NAME | curID | result |
85 +-----+-----+-----+-----+
86 | S000001 | 赵大 | C000002 | 92 |
87 | S000001 | 赵大 | C000003 | 100 |
88 | S000002 | 钱二 | C000002 | 94 |
89 | S000003 | 孙三 | C000002 | 96 |
90 +-----+-----+-----+-----+
91 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。

```
1  --
2  如果多表连接查询时没有WHERE子句中指定查询条件，那么这种查询就是交叉查询。
3
4  --MYSQL
5  mysql> SELECT * FROM T_STUDENTS,T_RESULTS;
6  +-----+-----+-----+-----+-----+-----+-----+-----+
7  | stuID | NAME | AGE | SEX | BIRTH | stuID | curID |
8  | result |
9  +-----+-----+-----+-----+-----+-----+-----+-----+
10 | s000001 | 赵大 | 25 | 男 | 1995-04-12 | s000001 | C000001 |
11 | 82 |
12 | s000002 | 钱二 | 21 | 男 | 1999-05-13 | s000001 | C000001 |
13 | 82 |
14 | s000003 | 孙三 | 24 | 女 | 1996-08-06 | s000001 | C000001 |
15 | 82 |
16 | s000004 | 李四 | 24 | 男 | 1996-05-23 | s000001 | C000001 |
17 | 82 |
18 | s000005 | 周五 | 20 | 男 | 2000-05-03 | s000001 | C000001 |
19 | 82 |
20 | s000006 | 赵大 | 25 | 男 | 1995-04-12 | s000001 | C000001 |
21 | 82 |
22 | s000007 | 吴三 | 25 | 男 | 1995-02-22 | s000001 | C000001 |
23 | 82 |
24 | s000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 | s000001 | C000001 |
25 | 82 |
26 | s000001 | 赵大 | 25 | 男 | 1995-04-12 | s000001 | C000002 |
27 | 92 |
28 | s000002 | 钱二 | 21 | 男 | 1999-05-13 | s000001 | C000002 |
29 | 92 |
30 | s000003 | 孙三 | 24 | 女 | 1996-08-06 | s000001 | C000002 |
31 | 92 |
32 | s000004 | 李四 | 24 | 男 | 1996-05-23 | s000001 | C000002 |
33 | 92 |
34 | s000005 | 周五 | 20 | 男 | 2000-05-03 | s000001 | C000002 |
35 | 92 |
36 | s000006 | 赵大 | 25 | 男 | 1995-04-12 | s000001 | C000002 |
37 | 92 |
38 | s000007 | 吴三 | 25 | 男 | 1995-02-22 | s000001 | C000002 |
39 | 92 |
40 | s000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 | s000001 | C000002 |
41 | 92 |
42 | s000001 | 赵大 | 25 | 男 | 1995-04-12 | s000001 | C000003 |
43 | 100 |
44 | s000002 | 钱二 | 21 | 男 | 1999-05-13 | s000001 | C000003 |
45 | 100 |
46 | s000003 | 孙三 | 24 | 女 | 1996-08-06 | s000001 | C000003 |
47 | 100 |
48 | s000004 | 李四 | 24 | 男 | 1996-05-23 | s000001 | C000003 |
49 | 100 |
```

29		S000005		周五		20		男		2000-05-03		S000001		C000003	
		100													
30		S000006		赵大		25		男		1995-04-12		S000001		C000003	
		100													
31		S000007		吴三		25		男		1995-02-22		S000001		C000003	
		100													
32		S000008		JERMAN_RAY		23		女		1997-05-22		S000001		C000003	
		100													
33		S000001		赵大		25		男		1995-04-12		S000002		C000001	
		82													
34		S000002		钱二		21		男		1999-05-13		S000002		C000001	
		82													
35		S000003		孙三		24		女		1996-08-06		S000002		C000001	
		82													
36		S000004		李四		24		男		1996-05-23		S000002		C000001	
		82													
37		S000005		周五		20		男		2000-05-03		S000002		C000001	
		82													
38		S000006		赵大		25		男		1995-04-12		S000002		C000001	
		82													
39		S000007		吴三		25		男		1995-02-22		S000002		C000001	
		82													
40		S000008		JERMAN_RAY		23		女		1997-05-22		S000002		C000001	
		82													
41		S000001		赵大		25		男		1995-04-12		S000002		C000002	
		94													
42		S000002		钱二		21		男		1999-05-13		S000002		C000002	
		94													
43		S000003		孙三		24		女		1996-08-06		S000002		C000002	
		94													
44		S000004		李四		24		男		1996-05-23		S000002		C000002	
		94													
45		S000005		周五		20		男		2000-05-03		S000002		C000002	
		94													
46		S000006		赵大		25		男		1995-04-12		S000002		C000002	
		94													
47		S000007		吴三		25		男		1995-02-22		S000002		C000002	
		94													
48		S000008		JERMAN_RAY		23		女		1997-05-22		S000002		C000002	
		94													
49		S000001		赵大		25		男		1995-04-12		S000002		C000003	
		46													
50		S000002		钱二		21		男		1999-05-13		S000002		C000003	
		46													
51		S000003		孙三		24		女		1996-08-06		S000002		C000003	
		46													
52		S000004		李四		24		男		1996-05-23		S000002		C000003	
		46													
53		S000005		周五		20		男		2000-05-03		S000002		C000003	
		46													
54		S000006		赵大		25		男		1995-04-12		S000002		C000003	
		46													
55		S000007		吴三		25		男		1995-02-22		S000002		C000003	
		46													
56		S000008		JERMAN_RAY		23		女		1997-05-22		S000002		C000003	
		46													
57		S000001		赵大		25		男		1995-04-12		S000003		C000001	
		82													

58	S000002 钱二	21 男 1999-05-13 S000003 C000001
	82	
59	S000003 孙三	24 女 1996-08-06 S000003 C000001
	82	
60	S000004 李四	24 男 1996-05-23 S000003 C000001
	82	
61	S000005 周五	20 男 2000-05-03 S000003 C000001
	82	
62	S000006 赵大	25 男 1995-04-12 S000003 C000001
	82	
63	S000007 吴三	25 男 1995-02-22 S000003 C000001
	82	
64	S000008 JERMAN_RAY	23 女 1997-05-22 S000003 C000001
	82	
65	S000001 赵大	25 男 1995-04-12 S000003 C000002
	96	
66	S000002 钱二	21 男 1999-05-13 S000003 C000002
	96	
67	S000003 孙三	24 女 1996-08-06 S000003 C000002
	96	
68	S000004 李四	24 男 1996-05-23 S000003 C000002
	96	
69	S000005 周五	20 男 2000-05-03 S000003 C000002
	96	
70	S000006 赵大	25 男 1995-04-12 S000003 C000002
	96	
71	S000007 吴三	25 男 1995-02-22 S000003 C000002
	96	
72	S000008 JERMAN_RAY	23 女 1997-05-22 S000003 C000002
	96	
73	S000001 赵大	25 男 1995-04-12 S000003 C000003
	46	
74	S000002 钱二	21 男 1999-05-13 S000003 C000003
	46	
75	S000003 孙三	24 女 1996-08-06 S000003 C000003
	46	
76	S000004 李四	24 男 1996-05-23 S000003 C000003
	46	
77	S000005 周五	20 男 2000-05-03 S000003 C000003
	46	
78	S000006 赵大	25 男 1995-04-12 S000003 C000003
	46	
79	S000007 吴三	25 男 1995-02-22 S000003 C000003
	46	
80	S000008 JERMAN_RAY	23 女 1997-05-22 S000003 C000003
	46	
81	S000001 赵大	25 男 1995-04-12 S000004 C000001
	89	
82	S000002 钱二	21 男 1999-05-13 S000004 C000001
	89	
83	S000003 孙三	24 女 1996-08-06 S000004 C000001
	89	
84	S000004 李四	24 男 1996-05-23 S000004 C000001
	89	
85	S000005 周五	20 男 2000-05-03 S000004 C000001
	89	
86	S000006 赵大	25 男 1995-04-12 S000004 C000001
	89	

```

87 | S000007 | 吴三 | 25 | 男 | 1995-02-22 | S000004 | C000001 |
89 |
88 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 | S000004 | C000001 |
89 |
89 | S000001 | 赵大 | 25 | 男 | 1995-04-12 | S000004 | C000002 |
64 |
90 | S000002 | 钱二 | 21 | 男 | 1999-05-13 | S000004 | C000002 |
64 |
91 | S000003 | 孙三 | 24 | 女 | 1996-08-06 | S000004 | C000002 |
64 |
92 | S000004 | 李四 | 24 | 男 | 1996-05-23 | S000004 | C000002 |
64 |
93 | S000005 | 周五 | 20 | 男 | 2000-05-03 | S000004 | C000002 |
64 |
94 | S000006 | 赵大 | 25 | 男 | 1995-04-12 | S000004 | C000002 |
64 |
95 | S000007 | 吴三 | 25 | 男 | 1995-02-22 | S000004 | C000002 |
64 |
96 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 | S000004 | C000002 |
64 |
97 | S000001 | 赵大 | 25 | 男 | 1995-04-12 | S000004 | C000003 |
58 |
98 | S000002 | 钱二 | 21 | 男 | 1999-05-13 | S000004 | C000003 |
58 |
99 | S000003 | 孙三 | 24 | 女 | 1996-08-06 | S000004 | C000003 |
58 |
100 | S000004 | 李四 | 24 | 男 | 1996-05-23 | S000004 | C000003 |
58 |
101 | S000005 | 周五 | 20 | 男 | 2000-05-03 | S000004 | C000003 |
58 |
102 | S000006 | 赵大 | 25 | 男 | 1995-04-12 | S000004 | C000003 |
58 |
103 | S000007 | 吴三 | 25 | 男 | 1995-02-22 | S000004 | C000003 |
58 |
104 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 | S000004 | C000003 |
58 |
105 +-----+-----+-----+-----+-----+-----+-----+-----+
---+
106 96 rows in set
107

```

3、自连接

```

1  --
2  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。
3  -- MySQL
4  mysql> SELECT * FROM T_STUDENTS S1 ,T_STUDENTS S2 WHERE S1.SEX
5  = '男' AND S1.AGE > S2.AGE;S2.AGE;
6  +-----+-----+-----+-----+-----+-----+-----+-----+
---+-----+
7  | stuID | NAME | AGE | SEX | BIRTH | stuID | NAME | AGE |
SEX | BIRTH |
8  +-----+-----+-----+-----+-----+-----+-----+-----+
---+-----+
9  | S000001 | 赵大 | 25 | 男 | 1995-04-12 | S000002 | 钱二 | 21 | 男
| 1999-05-13 |

```



```

15      T_STUDENTS S
16 LEFT OUTER JOIN T_RESULTS R
17 ON
18     S.stuID = R.stuID;
19 +-----+-----+-----+-----+
20 | stuID   | NAME       | curID   | result   |
21 +-----+-----+-----+-----+
22 | S000001 | 赵大       | C000001 | 82       |
23 | S000001 | 赵大       | C000002 | 92       |
24 | S000001 | 赵大       | C000003 | 100      |
25 | S000002 | 钱二       | C000001 | 82       |
26 | S000002 | 钱二       | C000002 | 94       |
27 | S000002 | 钱二       | C000003 | 46       |
28 | S000003 | 孙三       | C000001 | 82       |
29 | S000003 | 孙三       | C000002 | 96       |
30 | S000003 | 孙三       | C000003 | 46       |
31 | S000004 | 李四       | C000001 | 89       |
32 | S000004 | 李四       | C000002 | 64       |
33 | S000004 | 李四       | C000003 | 58       |
34 | S000005 | 周五       | NULL    | NULL     |
35 | S000006 | 赵大       | NULL    | NULL     |
36 | S000007 | 吴三       | NULL    | NULL     |
37 | S000008 | JERMAN_RAY | NULL    | NULL     |
38 +-----+-----+-----+-----+
39 16 rows in set
40
41 -- ORACLE
42 SELECT 表1.字段,表2.字段...
43 FROM 表1 LEFT [OUTER] JOIN 表2
44 WHERE 表1.字段(+) = 表2.字段
45
46 SELECT
47     T1.CUST_MAT_NO,
48     T1.PILE_NO,
49     T2.CRANE_NAME
50 FROM
51     KHB_PLATE_PILENO T1,
52     L1T109 T2
53 WHERE
54     T1.CUST_MAT_NO (+) = T2.CUST_MAT_NO
55 AND ROWNUM <= 5;
56
57 CUST_MAT_NO          PILE_NO          CRANE_NAME
58 -----
59 E0346051100          W00292          W003
60 E0346051200          W00292          W003
61 E0346050100          W00292          W003
62 E0345117200          X00101          X004
63 E0345117300          X00191          X004
64
65
66 ---- 右外连接
67 右侧表数据将全部显示出来
68 -- MYSQL
69 SELECT 表1.字段,表2.字段...
70 FROM 表1 RIGHT [OUTER] JOIN 表2
71 ON 表1.字段 = 表2.字段
72

```

```

73 mysql> SELECT
74     S.stuID,
75     S.NAME,
76     R.curID,
77     R.result
78 FROM
79     T_STUDENTS S
80 RIGHT OUTER JOIN T_RESULTS R
81 ON
82     S.stuID = R.stuID;
83 +-----+-----+-----+-----+
84 | stuID | NAME | curID | result |
85 +-----+-----+-----+-----+
86 | S000001 | 赵大 | C000001 | 82 |
87 | S000001 | 赵大 | C000002 | 92 |
88 | S000001 | 赵大 | C000003 | 100 |
89 | S000002 | 钱二 | C000001 | 82 |
90 | S000002 | 钱二 | C000002 | 94 |
91 | S000002 | 钱二 | C000003 | 46 |
92 | S000003 | 孙三 | C000001 | 82 |
93 | S000003 | 孙三 | C000002 | 96 |
94 | S000003 | 孙三 | C000003 | 46 |
95 | S000004 | 李四 | C000001 | 89 |
96 | S000004 | 李四 | C000002 | 64 |
97 | S000004 | 李四 | C000003 | 58 |
98 +-----+-----+-----+-----+
99 12 rows in set
100
101 -- ORACLE
102 SELECT 表1.字段,表2.字段...
103 FROM 表1 LEFT [OUTER] JOIN 表2
104 WHERE 表1.字段 = 表2.字段(+)
105
106 SELECT
107     T1.CUST_MAT_NO,
108     T1.PILE_NO,
109     T2.CRANE_NAME
110 FROM
111     KHB_PLATE_PILENO T1,
112     L1T109 T2
113 WHERE
114     T1.CUST_MAT_NO = T2.CUST_MAT_NO(+)
115 AND ROWNUM <= 5;
116
117 CUST_MAT_NO          PILE_NO      CRANE_NAME
118 -----
119 B0360045100          V00941
120 B0360046200          X00691
121 B0360047200          V00941
122 B0360047300          V00941
123 B0360048100          X00691
124
125 ---- 全外连接
126 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。
127 -- 语法
128 SELECT 表1.字段,表2.字段...
129 FROM 表1 FULL [OUTER] JOIN 表2
130 ON 表1.字段 = 表2.字段

```

5.4.2 集合查询

```
1  ---- 并操作(UNION)
2  -- 语法
3  SELECT 语句1
4  UNION
5  SELECT 语句2
6
7  -- 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
8
9  -- MYSQL
10 SELECT
11     S.stuID,
12     S.NAME
13 FROM
14     T_STUDENTS S
15 UNION
16     SELECT
17         R.curID,
18         R.result
19     FROM
20         T_RESULTS R;
21 +-----+-----+
22 | stuID | NAME      |
23 +-----+-----+
24 | S000001 | 赵大      |
25 | S000002 | 钱二      |
26 | S000003 | 孙三      |
27 | S000004 | 李四      |
28 | S000005 | 周五      |
29 | S000006 | 赵大      |
30 | S000007 | 吴三      |
31 | S000008 | JERMAN_RAY |
32 | C000001 | 82        |
33 | C000002 | 92        |
34 | C000003 | 100       |
35 | C000002 | 94        |
36 | C000003 | 46        |
37 | C000002 | 96        |
38 | C000001 | 89        |
39 | C000002 | 64        |
40 | C000003 | 58        |
41 +-----+-----+
42 17 rows in set
43
44 ---- 交操作(INTERSECT)
45
46 ---- 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
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 |
```

```

26 | S000003 | 孙三 | 24 | 女 | 1996-08-06 |
27 | S000004 | 李四 | 24 | 男 | 1996-05-23 |
28 | S000005 | 周五 | 20 | 男 | 2000-05-03 |
29 | S000006 | 郑六(*) | 25 | 男 | 1995-04-12 |
30 | S000007 | 吴三(*) | 25 | 男 | 1995-02-22 |
31 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 |
32 +-----+-----+-----+-----+-----+
33 8 rows in set
34
35 ---- 2、基于多表查询创建视图
36 CREATE VIEW VIEW_NAME
37 ([别名1[,别名2]...])
38 AS
39 subQuery
40
41 mysql> CREATE VIEW V_STUDENTS_RESULT
42 (v_学号,v_姓名,v_课号,v_成绩)
43 AS
44 SELECT S.stuID,S.NAME,R.curID,R.result
45 FROM T_STUDENTS S,T_RESULTS R
46 WHERE R.result >= 60
47 AND S.stuID = R.stuID;
48
49 mysql> SELECT * FROM V_STUDENTS
50 _RESULT;
51 +-----+-----+-----+-----+
52 | v_学号 | v_姓名 | v_课号 | v_成绩 |
53 +-----+-----+-----+-----+
54 | S000001 | 赵大(*) | C000001 | 82 |
55 | S000001 | 赵大(*) | C000002 | 92 |
56 | S000001 | 赵大(*) | C000003 | 100 |
57 | S000002 | 钱二 | C000001 | 82 |
58 | S000002 | 钱二 | C000002 | 94 |
59 | S000003 | 孙三 | C000001 | 82 |
60 | S000003 | 孙三 | C000002 | 96 |
61 | S000004 | 李四 | C000001 | 89 |
62 | S000004 | 李四 | C000002 | 64 |
63 +-----+-----+-----+-----+
64 9 rows in set
65
66 ---- 3、基于函数、分组数据创建视图
67 CREATE VIEW VIEW_NAME
68 ([别名1[,别名2]...])
69 AS
70 subQuery
71
72 mysql> CREATE VIEW V_TEACHER
73 (v_编号,v_姓名,v_学院,v_薪资)
74 AS
75 SELECT teaID,teaNAME,depNAME,MAX(salary) as salary
76 FROM T_TEACHERS
77 GROUP BY depID
78 HAVING MAX(salary) >= 5000;
79
80 mysql> SELECT * FROM V_TEACHER;
81 +-----+-----+-----+-----+
82 | v_编号 | v_姓名 | v_学院 | v_薪资 |
83 +-----+-----+-----+-----+

```

```

84 | T00001 | 教师1 | 学院1 | 7875 |
85 | T00002 | 教师2 | 学院2 | 5400 |
86 +-----+-----+-----+-----+
87 2 rows in set
88
89 ---- 4、创建只读视图
90 CREATE VIEW V_TEACHER2
91 AS
92 SELECT * FROM T_TEACHERS
93 WITH READ ONLY;
94
95 -- 删除视图
96 DROP VIEW VIEW_NAME
97 DROP VIEW V_STUDENTS;

```

六、数据更新

6.1 插入数据 (INSERT)

```

1  ---- 1、插入单行数据记录
2  -- 语法
3  INSERT INTO 表名[(列1, 列2...)] VALUES (列值1,列值2...)
4  -- MYSQL
5  INSERT INTO T_STUDENTS VALUES('S000001','赵大',25,'男','19950412');
6  INSERT INTO T_STUDENTS VALUES('S000002','钱二',21,'男','19990513');
7  INSERT INTO T_STUDENTS VALUES('S000003','孙三',24,'女','19960806');
8  INSERT INTO T_STUDENTS VALUES('S000004','李四',24,'男','19960523');
9  INSERT INTO T_STUDENTS VALUES('S000005','周五',20,'男','20000503');
10 INSERT INTO T_STUDENTS VALUES('S000006','赵大',25,'男','19950412');
11 INSERT INTO T_STUDENTS VALUES('S000007','吴三',25,'男','19950222');
12 INSERT INTO T_STUDENTS VALUES('S000008','JERMAN_RAY',23,'女','19970522');
13
14 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
15 ('S000001', 'C000001', '82');
16 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
17 ('S000001', 'C000002', '92');
18 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
19 ('S000001', 'C000003', '100');
20 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
21 ('S000002', 'C000001', '82');
22 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
23 ('S000002', 'C000002', '94');
24 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
25 ('S000002', 'C000003', '46');
26 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
27 ('S000003', 'C000001', '82');
28 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
29 ('S000003', 'C000002', '96');
30 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
31 ('S000003', 'C000003', '46');
32 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
33 ('S000004', 'C000001', '89');
34 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
35 ('S000004', 'C000002', '64');

```

```

25 INSERT INTO `NJITDB`.`T_RESULTS` (`stuID`, `curID`, `result`) VALUES
   ('S000004', 'C000003', '58');
26
27 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');
28 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');
29 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');
30 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');
31 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');
32 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');
33
34
35 -- 先视图中插入数据记录
36 -- 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)

```

1  -- 修改单行数据记录
2  -- 单条数据满足WHERE后面的查询条件
3  UPDETE 表名 SET 列1 = 值1[,列2 = 值2] WHERE 条件
4  UPDATE T_STUDENTS SET NAME = '郑六' WHERE stuID = 'S000006';
5
6  -- 修改多行数据记录
7  -- 多条数据满足WHERE后面的查询条件
8  UPDETE 表名 SET 列1 = 值1[,列2 = 值2] WHERE 条件
9  UPDATE T_STUDENTS SET NAME = CONCAT(NAME, '(*)') WHERE AGE >= 25
10
11 -- CASE条件表达式修改数据记录
12 UPDATE T_TEACHERS
13 SET salary =
14 CASE
15 WHEN salary <= 4000 THEN salary+salary*0.1
16 WHEN salary > 4000 AND salary <= 6000 THEN salary+salary*0.08
17 WHEN salary > 6000 THEN salary+salary*0.05
18 ELSE salary
19 END;
20
21 -- 修改视图中的数据记录
22 -- 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)

```

1  -- 删除满足条件的记录

```



```

2 DELETE FROM 表名 WHERE 条件
3 DELETE FROM T_STUDENTS WHERE stuID = 'S000001';
4
5 -- 若不加条件，将删除所有数据
6 DELETE FROM 表名
7 DELETE FROM T_STUDENTS;
8
9 -- 定义外键约束的表中删除数据记录
10 -- 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。
11
12 ----- 删除前
13 mysql> SELECT * FROM T_STUDENTS;
14 +-----+-----+-----+-----+-----+
15 | stuID | NAME      | AGE | SEX | BIRTH      |
16 +-----+-----+-----+-----+-----+
17 | S000002 | 钱二      | 21 | 男 | 1999-05-13 |
18 | S000003 | 孙三      | 24 | 女 | 1996-08-06 |
19 | S000004 | 李四      | 24 | 男 | 1996-05-23 |
20 | S000005 | 周五      | 20 | 男 | 2000-05-03 |
21 | S000006 | 赵大      | 25 | 男 | 1995-04-12 |
22 | S000007 | 吴三      | 25 | 男 | 1995-02-22 |
23 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 |
24 +-----+-----+-----+-----+-----+
25 7 rows in set
26
27 mysql> SELECT * FROM T_RESULTS;
28 +-----+-----+-----+
29 | stuID | curID | result |
30 +-----+-----+-----+
31 | S000002 | C000001 | 82 |
32 | S000002 | C000002 | 94 |
33 | S000002 | C000003 | 46 |
34 | S000003 | C000001 | 82 |
35 | S000003 | C000002 | 96 |
36 | S000003 | C000003 | 46 |
37 | S000004 | C000001 | 89 |
38 | S000004 | C000002 | 64 |
39 | S000004 | C000003 | 58 |
40 +-----+-----+-----+
41 9 rows in set
42
43 ----- 删除
44 DELETE FROM T_STUDENTS WHERE stuID = 'S000002';
45
46 ----- 删除后
47 mysql> SELECT * FROM T_STUDENTS;
48 +-----+-----+-----+-----+-----+
49 | stuID | NAME      | AGE | SEX | BIRTH      |
50 +-----+-----+-----+-----+-----+
51 | S000003 | 孙三      | 24 | 女 | 1996-08-06 |
52 | S000004 | 李四      | 24 | 男 | 1996-05-23 |
53 | S000005 | 周五      | 20 | 男 | 2000-05-03 |
54 | S000006 | 赵大      | 25 | 男 | 1995-04-12 |
55 | S000007 | 吴三      | 25 | 男 | 1995-02-22 |
56 | S000008 | JERMAN_RAY | 23 | 女 | 1997-05-22 |
57 +-----+-----+-----+-----+-----+
58 6 rows in set

```

```

59
60 mysql> SELECT * FROM T_RESULTS;
61 +-----+-----+-----+
62 | stuID   | curID   | result |
63 +-----+-----+-----+
64 | S000003 | C000001 | 82     |
65 | S000003 | C000002 | 96     |
66 | S000003 | C000003 | 46     |
67 | S000004 | C000001 | 89     |
68 | S000004 | C000002 | 64     |
69 | S000004 | C000003 | 58     |
70 +-----+-----+-----+
71 6 rows in set
72
73 -- 删除视图中的数据记录
74 -- 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：

```

1 BEGIN
2     SAVEPOINT sp;
3
4     INSERT INTO T_STUDENTS
5         VALUES ('203150548',
6                 '张三',
7                 20,
8                 '男',
9                 to_date('1996-06-03', 'yyyy-mm-dd'));
10 EXCEPTION
11     WHEN TIMEOUT_ON_RESOURCE
12     THEN
13         ROLLBACK TO sp;
14         COMMIT;
15 END;

```

八、权限管理

```

1  -- 授予/回收指定用户操作数据表的权限
2  GRANT/REVOKE 权限[,权限] ON TABLE 表名[,表明] TO 用户[,用户]
3
4  GRANT/REVOKE SELECT,INSERT,DELETE,UPDATE ON TABLE T_STUDENTS TO ZZZ;
5
6  -- 授予/回收指定用户操作数据列的权限
7  GRANT/REVOKE 权限[,权限] ON TABLE 表名[,表明] TO 用户[,用户]
8
9  GRANT/REVOKE UPDATE(NAME,AGE) ON TABLE T_STUDENTS TO ZZZ;
10
11 -- 授予/回收指定用户授权的权限
12 GRANT/REVOKE 权限[,权限] ON TABLE 表名[,表明] TO 用户[,用户] WITH GRANT OPTION
13
14 GRANT/REVOKE SELECT ON TABLE T_STUDENTS TO ZZZ WITH GRANT OPTION;
15
16 -- 授予/回收创建数据表的权限
17 GRANT/REVOKE CREATETAB ON DATABASE DBName TO 用户
18
19 GRANT/REVOKE CREATETAB ON DATABASE NJITDB TO zzz;
20
21 -- 授予/回收所有用户的操作权限
22 GRANT/REVOKE 权限[,权限] ON TABLE 表名[,表明] TO PUBLIC
23
24 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符串	LEFT(string)	/
从指定字符串右侧读取字符串	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Oracle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

```

1  <<名字>> --语句块命名
2  DECLARE
3      /*
4      * 定义部分
5      * 定义常量、变量、游标、数据类型等
6      */
7  BEGIN
8      /*
9      * 执行部分
10     * 处理业务逻辑，实现程序功能
11     */
12  EXCEPTION
13     /*
14     * 异常处理部分

```

```

15      * 处理程序中可能出现的错误
16      */
17  END;
18
19  注意,其中只有执行部分是必需的, 其它都是可选的。
20
21  <<NAME>>
22  DECLARE
23      v_username    VARCHAR (20) := 'zzz';
24      v_userid      VARCHAR (20) := 'z153654';
25      v_password    VARCHAR (20) := '123456';
26  BEGIN
27      UPDATE KHB_USER
28          SET USER_ID = v_userid
29          WHERE USER_NAME = v_username;
30
31  EXCEPTION
32      WHEN NO_DATA_FOUND
33      THEN
34          INSERT INTO KHB_USER (USER_NAME, USER_ID, USER_PASSWORD)
35              VALUES (v_username, v_userid, v_password);
36  END;

```

数据类型

```

1  -- 1、标量类型
2  ---- 数字类型
3      NUMBER
4      BINARY_INTEGER
5      PLS_INTEGER
6      BINARY_FLOAT
7      BINARY_DOUBLE
8
9  ---- 字符类型
10     CHAR
11     VARCHAR2
12     NCHAR
13     NVARCHAR2
14     LONG
15     LONGRAW
16     RAW
17
18  ---- 布尔类型
19
20  ---- 日期时间类型
21     DATE
22     TIMESTAMP
23     INTERVAL
24
25  -----定义和使用标量类型: BEGIN-----
26  -----
27  -- 定义语法规则
28  variable_name [CONSTANT] datatype [NOT NULL] [:= value[DEFAULT]]
29
30  -- 例子
31  DECLARE
32      v_username    KHB_USER.USER_NAME%TYPE;

```



```

32     v_userid      KHB_USER.USER_ID%TYPE   ;
33     v_password    KHB_USER.USER_PASSWORD%TYPE ;
34 BEGIN
35     SELECT KHB_USER.USER_NAME,KHB_USER.USER_ID,KHB_USER.USER_PASSWORD
36     INTO v_username,v_userid,v_password
37     FROM KHB_USER
38     WHERE KHB_USER.USER_ID = '10027107';
39
40     DBMS_OUTPUT.PUT_LINE('用户姓名: '|| v_username);
41     DBMS_OUTPUT.PUT_LINE('用户工号: '|| v_userid);
42     DBMS_OUTPUT.PUT_LINE('用户密码: '|| v_password);
43 END;
44 -----定义和使用标量类型: END-----
45
46 -- 2、复合类型
47     记录
48     表
49     嵌套表
50     变长数组
51 -----定义和使用复合类型: BEGIN-----
52
53 -- 语法规则 定义记录类型和记录变量
54     TYPE type_name IS RECORD(
55         record_declaration[,
56         record_declaration
57         ...]
58     );
59
60     record_name type_name;
61
62 -- 例子 定义和使用记录
63 DECLARE
64     TYPE t_KHB_USER IS RECORD(
65         v_username      KHB_USER.USER_NAME%TYPE,      -- 用户姓名
66         v_userid        KHB_USER.USER_ID%TYPE ,        -- 用户工号
67         v_password      KHB_USER.USER_PASSWORD%TYPE    -- 用户密码
68     );
69
70     user_record t_KHB_USER;          -- 定义记录变量
71 BEGIN
72     SELECT KHB_USER.USER_NAME,KHB_USER.USER_ID,KHB_USER.USER_PASSWORD
73     INTO user_record
74     FROM KHB_USER
75     WHERE KHB_USER.USER_ID = '10027107';
76
77     DBMS_OUTPUT.PUT_LINE('用户姓名: '|| user_record.v_username);
78     DBMS_OUTPUT.PUT_LINE('用户工号: '|| user_record.v_userid);
79     DBMS_OUTPUT.PUT_LINE('用户密码: '|| user_record.v_password);
80 END;
81 -----
82
83 -- 语法规则 定义表类型和表变量
84     TYPE type_name IS TABLE OF element_type
85     [NOT NULL] INDEX BY key_type;
86

```

```

87     table_name type_name;
88
89 -- 例子 定义和使用表
90 DECLARE
91     TYPE t_KHB_USER IS TABLE OF KHB_USER.USER_NAME%TYPE
92         INDEX BY BINARY_INTEGER;
93
94     user_table    t_KHB_USER;                                --
定义表变量
95 BEGIN
96     SELECT KHB_USER.USER_NAME
97         INTO user_table(1)
98         FROM KHB_USER
99         WHERE KHB_USER.USER_ID = '10027107';
100
101     DBMS_OUTPUT.PUT_LINE ('用户姓名: ' || user_table (1));
102 END;
103
104 -----
105 -----
106 -- 语法规则 定义嵌套表类型和嵌套表变量
107     TYPE type_name IS TABLE OF element_type [NOT NULL];
108
109     table_name type_name;
110
111 -- 例子 定义和使用嵌套表
112 DECLARE
113     TYPE t_KHB_USER IS TABLE OF KHB_USER.USER_NAME%TYPE;
114
115     user_table    t_KHB_USER := t_KHB_USER ('aaa', 'bbb', 'ccc');    --
定义嵌套表变量
116 BEGIN
117     DBMS_OUTPUT.PUT_LINE (user_table (1));
118     DBMS_OUTPUT.PUT_LINE (user_table (2));
119     DBMS_OUTPUT.PUT_LINE (user_table (3));
120 END;
121
122 -----
123 -----
124 -- 语法规则 定义变长数组类型和变长数组变量
125     TYPE type_name IS VARRAY(max_size) OF element_data [NOT NULL];
126
127     array_name type_name;
128
129 -- 例子 定义和使用变长数组
130 DECLARE
131     TYPE t_KHB_USER IS VARRAY(20) OF KHB_USER.USER_NAME%TYPE;
132
133     user_varray    t_KHB_USER := t_KHB_USER ('aaa', 'bbb', 'ccc');    -- 定义
变长数组变量
134 BEGIN
135     DBMS_OUTPUT.PUT_LINE (user_varray (1));
136     DBMS_OUTPUT.PUT_LINE (user_varray (2));
137     DBMS_OUTPUT.PUT_LINE (user_varray (3));
138 END;

```

```

139 -----定义和使用复合类型： END-----
140 -----
141 -- 3、引用类型
142     REF
143     REF CURSOR
144
145 -- 4、LOB类型
146     BFILE
147     BLOB
148     CLOB
149     NCLOB
150
151 -- 5、子类型
152 -----定义和使用子类型： BEGIN-----
153 -----
154 -- 语法规则 定义子类型和子类型变量
155     SUBTYPE subtype_name IS original_type [(constraint)][NOT NULL];
156
157     variable_name subtype_name;
158
159 -- 例子 定义和使用子类型
160     DECLARE
161         SUBTYPE t_sum IS NUMBER;
162         v_total t_sum(4);
163     BEGIN
164         v_total :=1000;
165         DBMS_OUTPUT.PUT_LINE (v_total);
166     END;
167 -----定义和使用子类型： END-----
168 -----

```

10.3 控制结构

分支结构

```

1  -- IF语句
2  ----- 语法规则 允许嵌套
3  IF expression1 THEN
4      statement1;
5  ELSIF expression2 THEN
6      statement2;
7  ELSE
8      statement3;
9  END IF;
10
11
12 -- CASE语句
13 ----- 语法规则
14 CASE selector
15     WHEN value1 THEN
16         statement1;
17     WHEN value2 THEN
18         statement2;
19     ...
20     WHEN valueN THEN

```

```

21         statementN;
22     [ELSE statementN+1;]
23 END CASE;
24

```

循环结构

```

1  -- LOOP语句
2  ---- 语法规则
3  LOOP
4      statement;
5      EXIT [WHEN condition]
6  END LOOP;
7
8  ---- 例子
9  DECLARE
10     v_index    NUMBER := 1;
11     v_count    NUMBER := 0;
12 BEGIN
13     SELECT COUNT (KHB_USER.USER_NAME)
14         INTO v_count
15         FROM KHB_USER
16         WHERE KHB_USER.USER_PASSWORD = '123456';
17
18     LOOP
19         DBMS_OUTPUT.PUT_LINE (v_index);
20         v_index := v_index + 1;
21         EXIT WHEN v_index > v_count;
22     END LOOP;
23 END;
24
25 -- WHILE - LOOP语句
26 ---- 语法规则
27 WHILE condition LOOP
28     statement;
29     ...
30 END LOOP;
31
32 ---- 例子
33 DECLARE
34     v_index    NUMBER := 1;
35     v_count    NUMBER := 0;
36 BEGIN
37     SELECT COUNT (KHB_USER.USER_NAME)
38         INTO v_count
39         FROM KHB_USER
40         WHERE KHB_USER.USER_PASSWORD = '123456';
41
42     WHILE v_index <= v_count LOOP
43         DBMS_OUTPUT.PUT_LINE (v_index);
44         v_index := v_index + 1;
45     END LOOP;
46 END;
47
48 -- FOR - LOOP语句
49 ---- 语法规则
50 FOR counter IN [REVERSE] lower_bound..higher_bound LOOP

```

```

51     statement;
52     ...
53 END LOOP;
54
55 ---- 例子
56 DECLARE
57     v_index    NUMBER := 1;
58     v_count    NUMBER := 0;
59 BEGIN
60     SELECT COUNT (KHB_USER.USER_NAME)
61     INTO v_count
62     FROM KHB_USER
63     WHERE KHB_USER.USER_PASSWORD = '123456';
64
65     FOR v_index IN 1..v_count LOOP
66         DBMS_OUTPUT.PUT_LINE (v_index);
67     END LOOP;
68 END;
69

```

顺序结构

```

1  -- GOTO语句
2  ---- 语法规则 能不用就不用
3  GOTO label_name;
4
5  DECLARE
6      v_index    NUMBER := 1;
7      v_count    NUMBER := 0;
8  BEGIN
9      GOTO print;
10
11     SELECT COUNT (KHB_USER.USER_NAME)
12     INTO v_count
13     FROM KHB_USER
14     WHERE KHB_USER.USER_PASSWORD = '123456';
15
16
17     << print >>
18     DBMS_OUTPUT.PUT_LINE(v_count);
19 END;
20
21 -- NULL语句
22 DECLARE
23     v_index    NUMBER := 1;
24     v_count    NUMBER := 0;
25 BEGIN
26     GOTO print;
27
28     SELECT COUNT (KHB_USER.USER_NAME)
29     INTO v_count
30     FROM KHB_USER
31     WHERE KHB_USER.USER_PASSWORD = '123456';
32
33
34     << print >>
35     NULL;

```

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  -- 1、声明游标
2  CURSOR cursor_name IS select_statement;
3
4  -- 2、打开游标OPEN
5  OPEN cursor_name;
6
7  -- 3、提取结果FETCH
8  FETCH cursor_name [BULK COLLECT] INTO variable1 [,variable2...]
9
10 -- variable 可以是标量、记录和集合
11 -- 使用标量变量读取游标结果集中的数据
12 -- 使用记录标量读取游标结果集中的数据
13 -- 使用集合变量读取游标结果集中的数据
14
15 -- 4、关闭游标CLOSE
16 CLOSE cursor_name
17
18
19 -- 例子
20 DECLARE
21     TYPE t_tablename IS TABLE OF KHB_USER%ROWTYPE;
22
23     user_table  t_tablename;
24
25     CURSOR cursor_user
26     IS
27         SELECT *
28         FROM KHB_USER
29         WHERE KHB_USER.USER_PASSWORD = '123456';
30 BEGIN
31     OPEN cursor_user;
32
33     IF cursor_user%ISOPEN
-- 1、声明游标
-- 2、打开游标

```

```

34     THEN
35         DBMS_OUTPUT.PUT_LINE ('打开游标成功');
36
37         FETCH cursor_user BULK COLLECT INTO user_table;      -- 3、提取结果
38     FETCH
39
40     IF cursor_user%FOUND
41     THEN
42         DBMS_OUTPUT.PUT_LINE (
43             '提取数据成功,行数: ' || cursor_user%ROWCOUNT);
44     ELSIF cursor_user%NOTFOUND
45     THEN
46         DBMS_OUTPUT.PUT_LINE ('提取数据失败');
47     END IF;
48
49     FOR i IN 1 .. cursor_user%ROWCOUNT      --
50     user_table.COUNT
51     LOOP
52         DBMS_OUTPUT.PUT_LINE (
53             '用户名: '
54             || user_table (i).USER_NAME
55             || ' 用户工号: '
56             || user_table (i).USER_ID
57             || ' 用户密码: '
58             || user_table (i).USER_PASSWORD);
59     END LOOP;
60
61     CLOSE cursor_user;      -- 4、关闭游标
62 CLOSE
63 ELSE
64     DBMS_OUTPUT.PUT_LINE ('打开游标失败');
65 END IF;
66 END;

```

游标变量

使用

```

1  -- 1、声明游标标量
2  TYPE type_name IS REF CURSOR [RETURN return_type];
3  v_cursor_name type_name;
4
5  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
6
7  -- 2、打开游标标量OPEN
8  OPEN v_cursor_name FOR select_statement;
9
10 -- 3、提取结果FETCH
11 FETCH v_cursor_name [BULK COLLECT] INTO variable1 [,variable2...]
12
13 -- variable 可以是标量、记录和集合
14 -- 使用标量变量读取游标结果集中的数据
15 -- 使用记录标量读取游标结果集中的数据
16 -- 使用集合变量读取游标结果集中的数据
17
18 -- 4、关闭游标标量CLOSE
19 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         ELSEIF 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，这种错误是由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  -- 1、声明异常
2
3  -- 2、抛出异常
4
5  -- 3、捕获处理异常
6  EXCEPTION
7      WHEN exception_name THEN
8          statements1;
9      WHEN exception_name THEN
10         statement2;
11     [WHEN OTHERS THEN
12         statements;]
13
14  -- 例子
15  DECLARE
16      e_illegalValue EXCEPTION;                -- 1、声明自定义异常
17      v_count NUMBER;
18  BEGIN
19      SELECT count(*)
20      INTO v_count
21      FROM KHB_USER
22      WHERE KHB_USER.USER_PASSWORD IN ('123456','111111') ;
23
24      IF v_count > 0 THEN
25          RAISE e_illegalValue;                -- 2、抛出自定义异常
26      END IF;
27
28  EXCEPTION                                    -- 3、异常捕获和处理
29      WHEN NO_DATA_FOUND THEN
30          DBMS_OUTPUT.PUT_LINE('查询数据不存在');
31      WHEN e_illegalValue THEN
32          DBMS_OUTPUT.PUT_LINE(v_count || '人的密码太简单');
33      WHEN OTHERS THEN
34          DBMS_OUTPUT.PUT_LINE ('发生其它错误 错误编号: ' || SQLCODE || ' 错误信
35  息: ' || SQLERRM);
36  END;
```

异常处理机制

```
1  -- 1、声明部分中异常处理机制
2
3  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
4  EXCEPTION部分异常处理语句所捕获。
5
6  -- 2、执行部分中异常处理机制
7
8  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。
9
10 -- 3、异常处理部分中异常处理机制
11
12 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  -- 1、对捕获的异常进行处理
2
3  -- 2、确保没有未处理的异常
4
5  -- 3、标记异常发生的位置
6
7  -- 4、控制异常处理程序
8
```

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

```
1  -- 语法规则
2  CREATE [OR REPLACE] PROCEDURE procedure_name
3  [(argument1[model] datatype1 [,argument2[model] datatype2] ...)]
4  {IS | AS}
5  [local declarations]
6  BEGIN
7      executable statements
8  [EXCEPTION
9      exception handlers]
10 END [procedure_name];
11
12
13
14 -- 例子
15 CREATE OR REPLACE PROCEDURE procedure_insert_user (
16     user_name      IN      KHB_USER.USER_NAME%TYPE,
17     user_id        IN      KHB_USER.USER_ID%TYPE,
18     user_password  IN      KHB_USER.USER_PASSWORD%TYPE DEFAULT '123123',
19     user_count     OUT NUMBER)
20 AS
21     v_count        NUMBER;
22     e_illegalValue EXCEPTION;
23 BEGIN
24     IF LENGTH (user_password) < 6 OR LENGTH (user_password) > 10
25     THEN
26         RAISE_APPLICATION_ERROR(-20150, 'the length of password is
invalid');
27     END IF;
28
29     INSERT INTO KHB_USER (USER_NAME, USER_ID, USER_PASSWORD)
30         VALUES (user_name, user_id, user_password);
31
32     SELECT COUNT (DISTINCT KHB_USER.USER_NAME) INTO user_count FROM
KHB_USER;
33
34     SELECT COUNT (*)
35         INTO v_count
36         FROM KHB_USER
37         WHERE KHB_USER.USER_PASSWORD IN ('123456', '111111');
38
```

```

39     IF v_count > 0
40     THEN
41         RAISE e_illegalValue;
42     END IF;
43 EXCEPTION
44     WHEN NO_DATA_FOUND
45     THEN
46         DBMS_OUTPUT.PUT_LINE ('未找到数据');
47     WHEN e_illegalValue
48     THEN
49         DBMS_OUTPUT.PUT_LINE (v_count || '人的密码太简单');
50     WHEN OTHERS
51     THEN
52         DBMS_OUTPUT.PUT_LINE (
53             '发生其它错误 错误编号: '
54             || SQLCODE
55             || ' 错误信息: '
56             || SQLERRM);
57 END procedure_insert_user;

```

调用

```

1 DECLARE
2     v_count NUMBER;
3 BEGIN
4     procedure_insert_user('zzz', 'z1523515', '123123',v_count);
5     DBMS_OUTPUT.PUT_LINE('员工数: ' || v_count);
6 EXCEPTION
7     WHEN OTHERS THEN
8         DBMS_OUTPUT.PUT_LINE(SQLERRM);
9 END;

```

删除

```

1 -- 语法规则
2 DROP procedure_name;
3
4 -- 例子
5 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

```

1 -- 语法规则
2 CREATE [OR REPLACE] FUNCTION function_name
3 [(argument1[model] datatype1 [,argument2[model] datatype2] ...)]
4 RETURN return_datatype
5 {IS | AS}
6 [local declarations]
7 BEGIN
8     executable statements

```

```

9  [EXCEPTION
10      exception handlers]
11  END [function_name];
12
13  -- 例子
14  CREATE OR REPLACE FUNCTION f_insert_user (
15      user_name      IN      KHB_USER.USER_NAME%TYPE,
16      user_id        IN      KHB_USER.USER_ID%TYPE,
17      user_password  IN      KHB_USER.USER_PASSWORD%TYPE DEFAULT '123123',
18      user_count     OUT NUMBER)
19  RETURN KHB_USER%ROWTYPE
20  AS
21      v_count        NUMBER;
22      e_illegalValue EXCEPTION;
23      t_user_record  KHB_USER%ROWTYPE;
24  BEGIN
25      IF LENGTH (user_password) < 6 OR LENGTH (user_password) > 10
26      THEN
27          RAISE_APPLICATION_ERROR(-20150, 'the length of password is
invalid');
28      END IF;
29
30      INSERT INTO KHB_USER (USER_NAME, USER_ID, USER_PASSWORD)
31          VALUES (user_name, user_id, user_password);
32
33      SELECT COUNT (DISTINCT KHB_USER.USER_NAME)
34      INTO user_count
35      FROM KHB_USER;
36
37      SELECT *
38      INTO t_user_record
39      FROM KHB_USER
40      WHERE KHB_USER.USER_ID = '78957';
41
42      RETURN t_user_record;
43
44      SELECT COUNT (*)
45      INTO v_count
46      FROM KHB_USER
47      WHERE KHB_USER.USER_PASSWORD IN ('123456', '111111');
48
49      IF v_count > 0
50      THEN
51          RAISE e_illegalValue;
52      END IF;
53
54  EXCEPTION
55      WHEN NO_DATA_FOUND
56      THEN
57          DBMS_OUTPUT.PUT_LINE ('未找到数据');
58      WHEN e_illegalValue
59      THEN
60          DBMS_OUTPUT.PUT_LINE (v_count || '人的密码太简单');
61      WHEN OTHERS
62      THEN
63          DBMS_OUTPUT.PUT_LINE (
64              '发生其它错误 错误编号: '
65              || SQLCODE

```

```

66         || ' 错误信息: '
67         || SQLERRM);
68 END f_insert_user;
69

```

调用

```

1 DECLARE
2     v_count NUMBER;
3     t_user_record KHB_USER%ROWTYPE;
4 BEGIN
5     t_user_record := f_insert_user('zzz', 'z1523515', '123123',v_count);
6     DBMS_OUTPUT.PUT_LINE('员工数: ' || v_count);
7     DBMS_OUTPUT.PUT_LINE(t_user_record.USER_NAME||' ' ||
t_user_record.USER_ID || ' ' || t_user_record.USER_PASSWORD);
8 EXCEPTION
9     WHEN OTHERS THEN
10         DBMS_OUTPUT.PUT_LINE(SQLERRM);
11 END;

```

删除

```

1 -- 语法规则
2 DROP function_name;
3
4 -- 例子
5 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

```

1 -- 创建包说明
2 -- 语法规则
3 CREATE [OR REPLACE] PACKAGE package_name
4 {IS | AS}
5     [variable_declaration...]          -- 声明变量
6     [cursor_declaration...]            -- 声明游标
7     [exception_declaration...]         -- 声明异常
8     [object_declaration...]            -- 声明对象
9     [function_declaration...]          -- 声明函数
10    [procedure_declaration...]          -- 声明过程
11 END [package_name];
12
13 -- 例子
14 CREATE OR REPLACE PACKAGE khb_user_package
15 AS
16     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     /*****
52 [p_ins_user]*****/
53     PROCEDURE p_ins_user (
54         user_name      IN KHB_USER.USER_NAME%TYPE,
55         user_id        IN KHB_USER.USER_ID%TYPE,
56         user_password  IN KHB_USER.USER_PASSWORD%TYPE DEFAULT '123123')
57     AS
58     BEGIN
59         IF LENGTH (user_password) < 6 OR LENGTH (user_password) > 10
60         THEN
61             RAISE e_illegalValue;
62         END IF;
63
64         INSERT INTO KHB_USER (USER_NAME, USER_ID, USER_PASSWORD)
65             VALUES (user_name, user_id, user_password);
66     EXCEPTION
67         WHEN e_illegalValue
68         THEN
69             DBMS_OUTPUT.PUT_LINE ('数据不合法');
70             ROLLBACK;
71         WHEN OTHERS
72         THEN
73             DBMS_OUTPUT.PUT_LINE ('其他错误 ' || SQLERRM);
74             ROLLBACK;

```

```

74     END p_ins_user;
75
76     /*****
[p_del_user]*****/
77     PROCEDURE p_del_user (user_name IN KHB_USER.USER_NAME%TYPE)
78     AS
79     BEGIN
80         DELETE FROM KHB_USER
81             WHERE KHB_USER.USER_NAME = user_name;
82     EXCEPTION
83         WHEN NO_DATA_FOUND
84         THEN
85             DBMS_OUTPUT.PUT_LINE ('未找到数据');
86             ROLLBACK;
87         WHEN OTHERS
88         THEN
89             DBMS_OUTPUT.PUT_LINE ('其他错误 ' || SQLERRM);
90             ROLLBACK;
91     END p_del_user;
92
93     /*****
[p_del_user]*****/
94     PROCEDURE p_del_user (user_name IN KHB_USER.USER_NAME%TYPE,
95                          user_id   IN KHB_USER.USER_ID%TYPE)
96     AS
97     BEGIN
98         DELETE FROM KHB_USER
99             WHERE KHB_USER.USER_NAME = user_name
100             AND KHB_USER.USER_ID = user_id;
101     EXCEPTION
102         WHEN NO_DATA_FOUND
103         THEN
104             DBMS_OUTPUT.PUT_LINE ('未找到数据');
105             ROLLBACK;
106         WHEN OTHERS
107         THEN
108             DBMS_OUTPUT.PUT_LINE ('其他错误 ' || SQLERRM);
109             ROLLBACK;
110     END p_del_user;
111
112     /*****
[f_sel_user]*****/
113     FUNCTION f_sel_user (user_id IN KHB_USER.USER_ID%TYPE)
114     RETURN KHB_USER%ROWTYPE
115     AS
116         t_user_record KHB_USER%ROWTYPE;
117     BEGIN
118         SELECT *
119             INTO t_user_record
120             FROM KHB_USER
121             WHERE KHB_USER.USER_ID = user_id;
122
123         RETURN t_user_record;
124     EXCEPTION
125         WHEN NO_DATA_FOUND
126         THEN
127             DBMS_OUTPUT.PUT_LINE ('未找到数据');
128             ROLLBACK;

```

```

129         WHEN OTHERS
130         THEN
131             DBMS_OUTPUT.PUT_LINE ('其他错误 ' || SQLERRM);
132         ROLLBACK;
133     END f_sel_user;
134 END khb_user_package;
135

```

调用

```

1  DECLARE
2      t_user_record KHB_USER%ROWTYPE;
3
4  BEGIN
5      khb_user_package.p_ins_user('zzz', '123', '123123');
6      khb_user_package.p_ins_user('zzz', '123123', '123123');
7      khb_user_package.p_ins_user('aaa', '1234', '123123');
8
9      khb_user_package.p_del_user('aaa');
10     khb_user_package.p_del_user('zzz', '123123');
11
12     t_user_record := khb_user_package.f_sel_user('10078880');
13     DBMS_OUTPUT.PUT_LINE('姓名: ' || t_user_record.USER_NAME);
14     DBMS_OUTPUT.PUT_LINE('工号: ' || t_user_record.USER_ID);
15     DBMS_OUTPUT.PUT_LINE('密码: ' || t_user_record.USER_PASSWORD);
16
17 END;

```

删除

```

1  -- 删除包体
2  DROP PACKAGE BODY packagebody_name;
3
4  -- 删除包说明和包体
5  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。


```

26     END IF;
27
28     SELECT TO_CHAR (SYSDATE, 'DY') INTO v_data FROM DUAL;
29
30     IF v_data IN ('SAT', 'SUN')
31     THEN
32         RAISE_APPLICATION_ERROR (-20011,
33                                     '不能在周末时间更新KHB_USER表');
34     END IF;
35 END trig_khb_user_editdata;
36
37 -- 2、创建行触发器
38 ---- 执行DML语句时，每作用一行就会被触发一次的触发器。
39 ----- 语法规则
40 CREATE OR REPLACE TRIGGER trigger_name
41 { BEFORE|AFTER} INSERT {OR DELETE OR UPDATE}
42 ON table_name [REFERENCING OLD AS old | NEW AS new]
43 FOR EACH ROW
44 [WHEN (logical_expression)]
45 {DECLARE
46     declaration_statement;}
47 BEGIN
48     execution_statements;
49 END [trigger_name];
50
51 ---- 例子
52 CREATE OR REPLACE TRIGGER trig_khb_user_backup
53 AFTER INSERT OR UPDATE OR DELETE
54 ON KHB_USER
55 FOR EACH ROW
56 DECLARE
57     v_count    NUMBER;
58 BEGIN
59     CASE
60     WHEN INSERTING
61     THEN
62         INSERT INTO KHB_USER_BK
63             VALUES (:new.USER_NAME,
64                     :new.USER_ID,
65                     :new.USER_PASSWORD,
66                     :new.QUES,
67                     :new.ANS);
68
69         SELECT COUNT(*) INTO v_count FROM KHB_USER_BK;
70         DBMS_OUTPUT.PUT_LINE (v_count);
71     WHEN UPDATING
72     THEN
73         UPDATE KHB_USER_BK
74             SET USER_NAME = :new.USER_NAME,
75                 USER_ID = :new.USER_ID,
76                 USER_PASSWORD = :new.USER_PASSWORD,
77                 QUES = :new.QUES,
78                 ANS = :new.ANS
79             WHERE USER_ID = :old.USER_ID;
80     WHEN DELETING
81     THEN
82         UPDATE KHB_USER_BK
83             SET USER_NAME = USER_NAME || '(离职)';

```

```

84         WHERE USER_ID = :old.USER_ID;
85     END CASE;
86 END trig_khb_user_backup ;
87
88 ---- 测试
89 INSERT INTO KHB_USER VALUES('AAA','123','1234123','AAA','AAA');
90
91 UPDATE KHB_USER
92 SET USER_PASSWORD = '147258369'
93 WHERE USER_NAME = 'AAA';
94
95 DELETE FROM KHB_USER
96 WHERE USER_NAME = 'AAA';

```

创建DDL触发器

```

1  ----- 语法规则
2  CREATE OR REPLACE TRIGGER trigger_name
3  { BEFORE|AFTER} DLL
4  ON table_name
5  [WHEN (logical_expression)]
6  {DECLARE
7      declaration_statement;}
8  BEGIN
9      execution_statements;
10 END [trigger_name];
11
12 ---- 例子
13 CREATE OR REPLACE TRIGGER trigger_drop
14     BEFORE DROP
15     ON PDAHC.SCHEMA
16 BEGIN
17     RAISE_APPLICATION_ERROR (-20022, '无效的删除操作');
18 END trigger_drop;

```

创建INSTEAD OF触发器

```

1  -- 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
   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。
2  -- INSTEAD OF触发器 只是适用于视图
3
4  ----- 语法规则
5  CREATE OR REPLACE TRIGGER trigger_name
6  INSTEAD OF INSERT [OR DELETE OR UPDATE]
7  ON view_name
8  [WHEN (logical_expression)]
9  {DECLARE
10     declaration_statement;}
11 BEGIN
12     execution_statements;
13 END [trigger_name];
14
15 ---- 例子
16

```

创建事件触发器

```
1  -- 事件触发器包括系统事件触发器和用户事件触发器。
2  --      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
3  --      用户事件包括用户登录和注销、DDL事件等
4
5  -- 1、创建系统事件触发器
6  ---- 语法规则
7  CREATE OR REPLACE TRIGGER trigger_name
8  {BEFORE OR AFTER} database_event
9  ON [database | schema]
10 {DECLARE
11     declaration_statement;}
12 BEGIN
13     execution_statements;
14 END [trigger_name];
15 ---- 例子
16
17 -----数据库启动
18 CREATE OR REPLACE TRIGGER trig_startup
19     AFTER STARTUP
20     ON DATABASE
21 BEGIN
22     INSERT INTO T_LOG
23         VALUES (ORA_SYSEVENT, SYSDATE, ORA_LOGIN_USER);
24 END;
25 -----数据库关闭
26 CREATE OR REPLACE TRIGGER trig_shutdown
27     BEFORE SHUTDOWN
28     ON DATABASE
29 BEGIN
30     INSERT INTO T_LOG
31         VALUES (ORA_SYSEVENT, SYSDATE, ORA_LOGIN_USER);
32 END;
33
34
35 -- 2、创建用户事件触发器
36 ---- 语法规则
37 CREATE OR REPLACE TRIGGER trigger_name
38 {BEFORE OR AFTER} database_event
39 ON [database | schema]
40 {DECLARE
41     declaration_statement;}
42 BEGIN
43     execution_statements;
44 END [trigger_name];
45
46 ---- 例子
47
48 -----用户登录
49 CREATE OR REPLACE TRIGGER trig_logon
50     AFTER LOGON
51     ON DATABASE
52 BEGIN
53     INSERT INTO T_LOG
54         VALUES (ORA_SYSEVENT, SYSDATE, ORA_LOGIN_USER);
55 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
