# Oracle第二天

## 1.回顾

## 2.多表查询

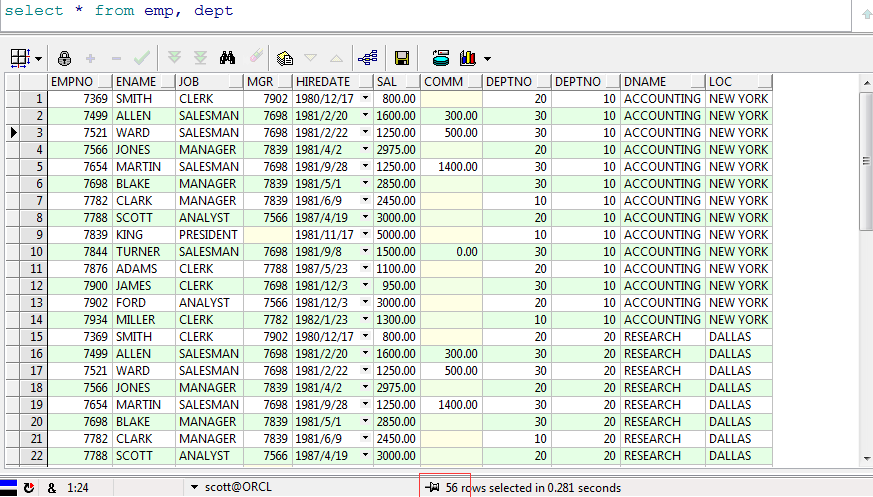
### 1.多表连接基本查询

使用一张以上的表做查询就是多表查询

语法： SELECT {DISTINCT} \*|列名.. FROM 表名 别名，表名1 别名

{WHERE 限制条件 ORDER BY 排序字段 ASC|DESC...}

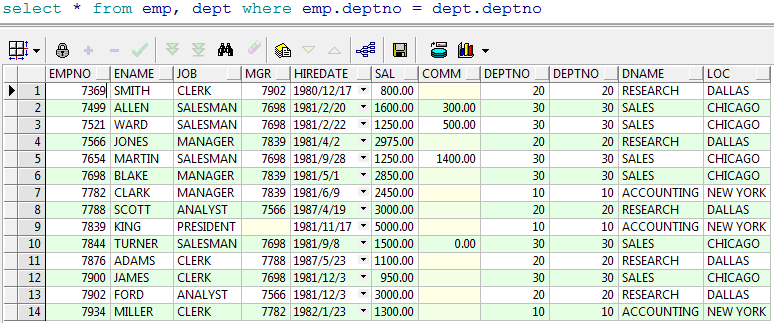
范例：查询员工表和部门表



我们发现产生的记录数是56条，我们还会发现emp表是14条，dept表是4条，56正是emp表和dept表的记录数的乘积，我们称其为笛卡尔积。

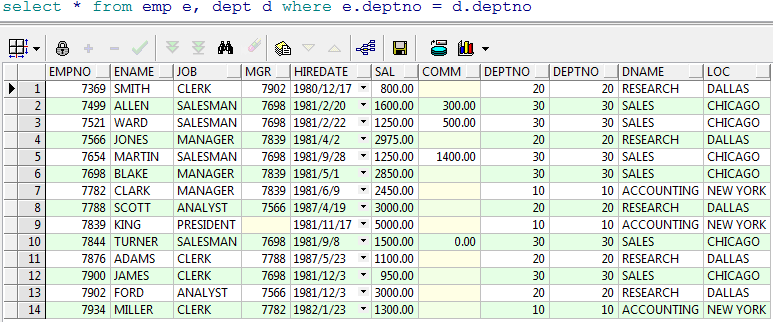
如果多张表进行一起查询而且每张表的数据很大的话笛卡尔积就会变得非常大，对性能造成影响，想要去掉笛卡尔积我们需要关联查询。

在两张表中我们发现有一个共同的字段是depno，depno就是两张表的关联的字段，我们可以使用这个字段来做限制条件，两张表的关联查询字段一般是其中一张表的主键，另一张表的外键。

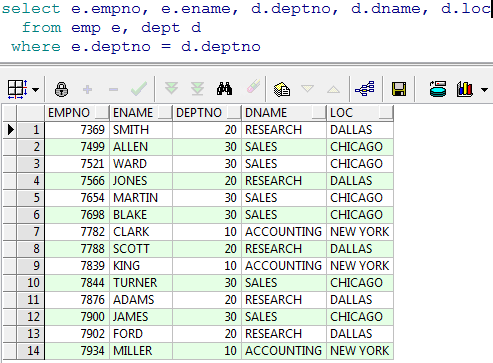


关联之后我们发现数据条数是14条，不在是56条。

多表查询我们可以为每一张表起一个别名

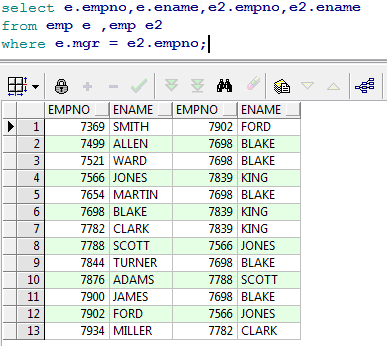


范例：查询出雇员的编号，姓名，部门的编号和名称，地址



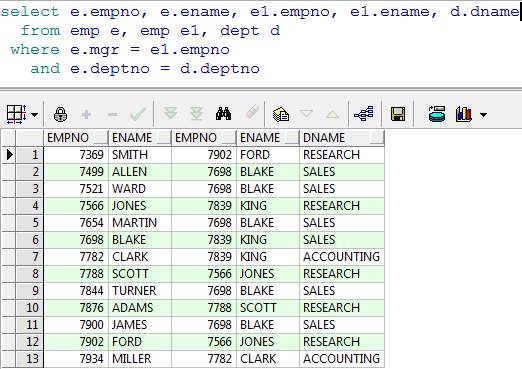
范例：查询出每个员工的上级领导

分析：emp表中的mgr字段是当前雇员的上级领导的编号，所以该字段对emp表产生了自身关联，可以使用mgr字段和empno来关联



范例:在上一个例子的基础上查询该员工的部门名称

分析：只要在上一个例子基础上再加一张表的关联，使用deptno来做关联字段即可



范例：查询出每个员工编号，姓名，部门名称，工资等级和他的上级领导的姓名，工资等级

select e.empno,

e.ename,

decode(s.grade,

1,'一级',

2,'二级',

3,'三级',

4,'四级',

5,'五级') grade,

d.dname,

e1.empno,

e1.ename,

decode(s1.grade,

1,'一级',

2,'二级',

3,'三级',

4,'四级',

5,'五级') grade

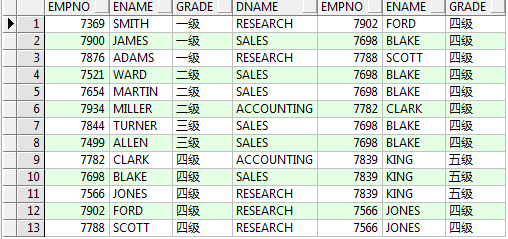
from emp e, emp e1, dept d, salgrade s, salgrade s1

where e.mgr = e1.empno

and e.deptno = d.deptno

and e.sal between s.losal and s.hisal

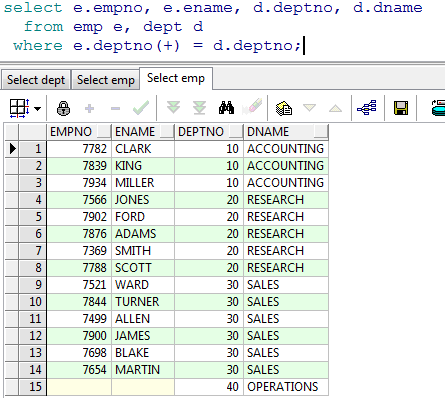
and e1.sal between s1.losal and s1.hisal



### 2.外连接（左右连接）

1. 右连接

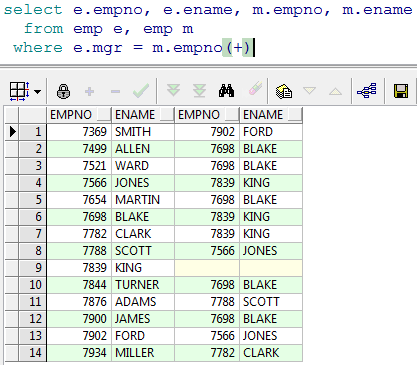
当我们在做基本连接查询的时候，查询出所有的部门下的员工，我们发现编号为40的部门下没有员工，但是要求把该部门也展示出来，我们发现上面的基本查询是办不到的



使用(+)表示左连接或者右连接，当(+)在左边表的关联条件字段上时是左连接，如果是在右边表的关联条件字段上就是右连接。

范例：查询出所有员工的上级领导

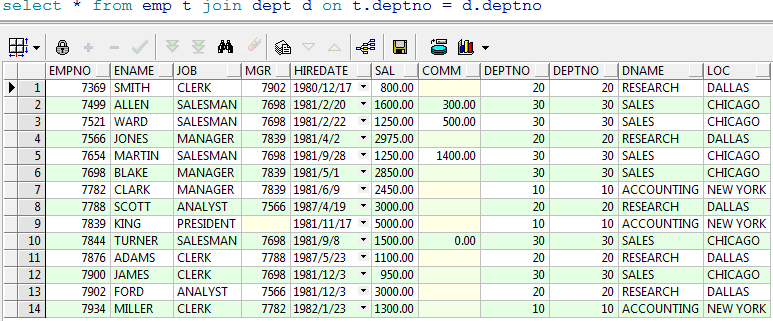
分析：我们发现使用我们以前的做法发现KING的上级领导没有被展示，我们需要使用左右连接把他查询出来



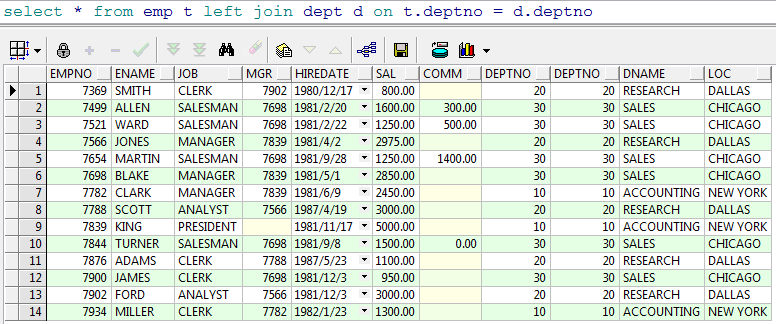
### 3.sql1999对SQL的支持

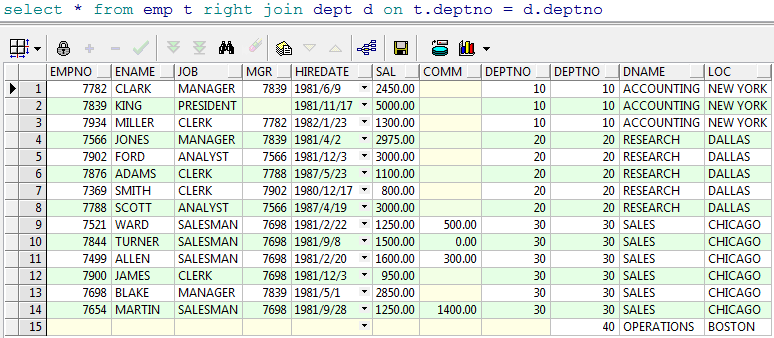
#### 1.ON子句，自己编写连接条件（重要）

On相当于where



#### 2.左连接和右连接LEFT JOIN和RIGHT JOIN（重要）



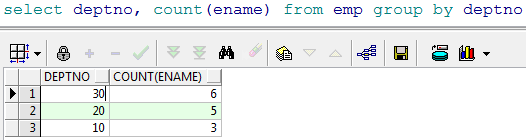


## 3. 分组统计

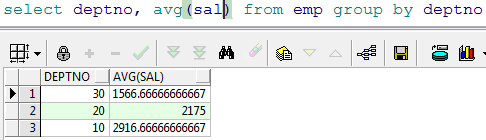
分组统计需要使用GROUP BY来分组

语法：语法：SELECT \* |列名 FROM 表名 {WEHRE 查询条件} {GROUP BY 分组字段} ORDER BY 列名1 ASC|DESC，列名2...ASC|DESC

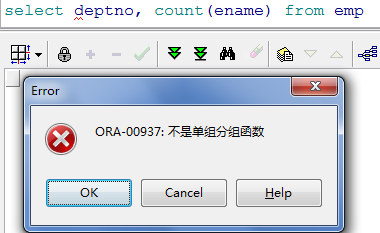
范例：查询每个部门的人数



范例：查询出每个部门的平均工资



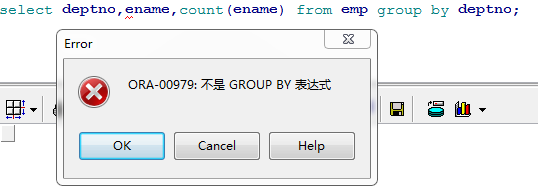
如果我们想查询出来部门编号，和部门下的人数



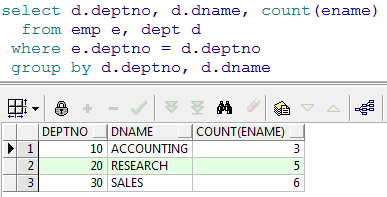
我们发现报了一个ORA-00937的错误

**注意：**

1. **如果使用分组函数，SQL只可以把GOURP BY分组条件字段和分组函数查询出来，不能有其他字段。**
2. **如果使用分组函数，不使用GROUP BY 只可以查询出来分组函数的值**

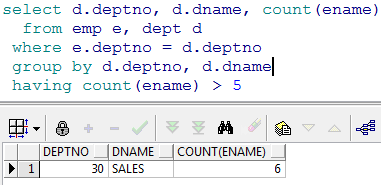
****

范例：按部门分组，查询出部门名称和部门的员工数量

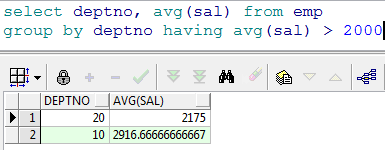


范例：查询出部门人数大于5人的部门

分析：需要给count(ename)加条件，此时在本查询中不能使用where，可以使用HAVING



范例：查询出部门平均工资大于2000的部门



注意：现在可以做学员练习题了



## 4. 子查询

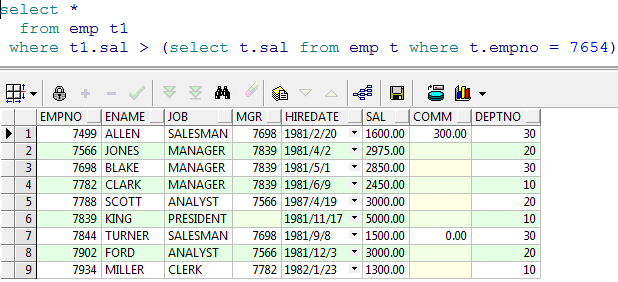
### 1.子查询

在一个查询的内部还包括另一个查询，则此查询称为子查询。

Sql的任何位置都可以加入子查询。

范例：查询比7654工资高的雇员

分析：查询出7654员工的工资是多少，把它作为条件



所有的子查询必须在“（）”中编写

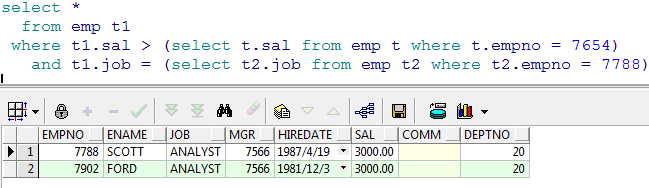
子查询在操作中有三类：

单列子查询：返回的结果是一列的一个内容

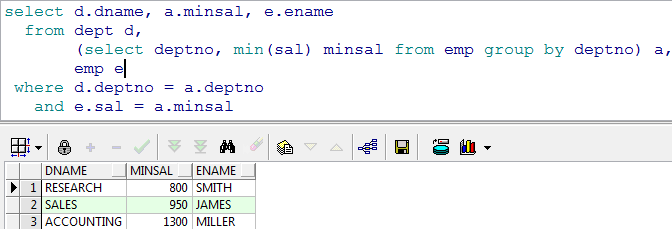
单行子查询：返回多个列，有可能是一个完整的记录

多行子查询：返回多条记录

范例：查询出比雇员7654的工资高，同时从事和7788的工作一样的员工

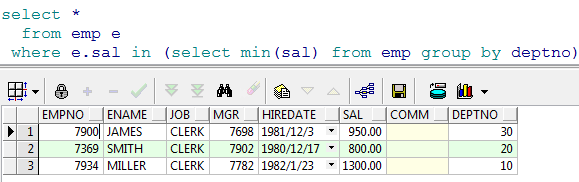


范例：要求查询每个部门的最低工资和最低工资的雇员和部门名称



在返回多条记录的子查询可以把它的结果集当做一张表，给起个别名， 如图中的a。

范例：查询出每个部门最低工资的员工

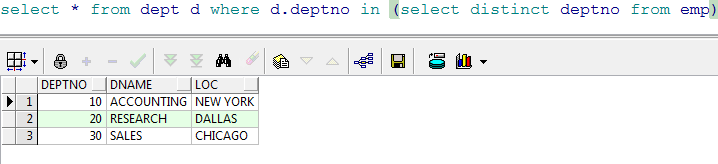


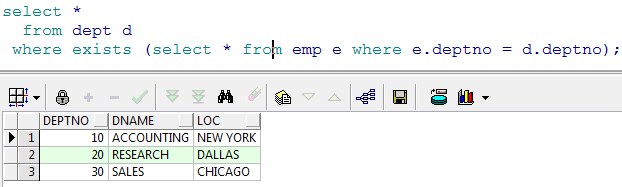
### 2. exists和not exists关键字

exists （sql 返回结果集为真）

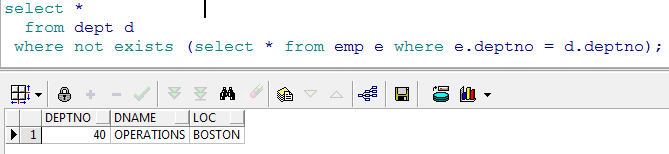
not exists (sql 不返回结果集为真）

范例：查询出有员工的部门有哪些？





范例：查询出没有员工的部门



### 3.Insert中加入子查询

语法：

Insert into 表名(列1,列2，列3,….)

Select 列1,列2，列3,….

From 表名

Where条件

注意:

1. 此时不要写values
2. 插入的列名与查询的列名要一致

### 4.Update与子查询

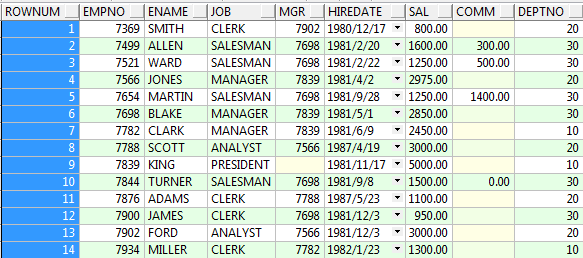
### 5.Delete与子查询

## 5. Rownum与分页查询

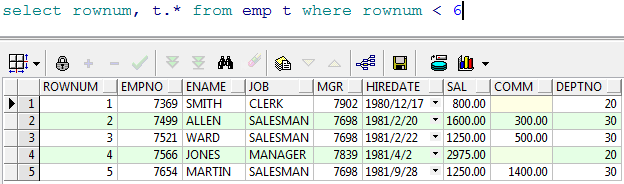
ROWNUM:表示行号，实际上此是一个列,但是这个列是一个伪列,此列可以在每张表中出现。

范例：查询emp表带有rownum列

select rownum, t.\* from emp t



我们可以根据rownum来取结果集的前几行，比如前5行



但是我们不能取到中间几行，因为rownum不支持大于号，只支持小于号，如果想 实现我们的需求怎么办呢？答案是使用子查询，也正是oracle分页的做法。

第一种写法：

select \*

from (select rownum rm, a.\* from (select \* from emp) a where rownum < 11) b where b.rm > 5

第二种写法：

select \*

from (select rownum **r** ,**emp.**\* from **emp**) **b**

where **b.r** >5 and **b.r** <11

## 6.视图

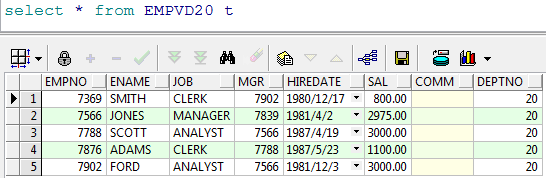
视图就是封装了一条复杂查询的语句。

语法1.：CREATE VIEW 视图名称 AS 子查询

范例：建立一个视图，此视图包括了20部门的全部员工信息

create view empvd20 as select \* from emp t where t.deptno = 20

视图创建完毕就可以使用视图来查询，查询出来的都是20部门的员工

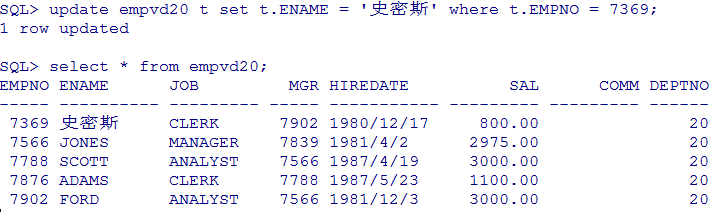


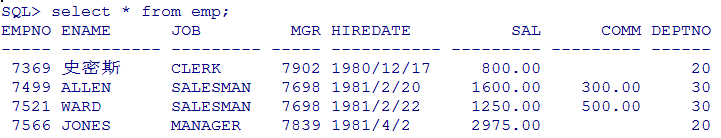
语法2：CREATE OR REPLACE VIEW 视图名称 AS 子查询

如果视图已经存在我们可以使用语法2来创建视图，这样已有的视图会被覆盖。

create or replace view empvd20 as select \* from emp t where t.deptno = 20

那么视图可以修改吗？



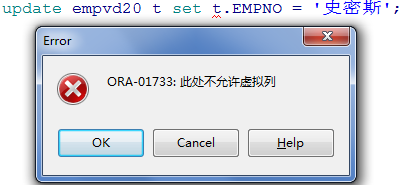


我们尝试着修改视图但是发现是视图所查询的表的字段值被修改了。所以我们一般不会去修改视图。

我们可以设置视图为只读。

语法3：CREATE OR REPLACE VIEW 视图名称 AS 子查询 WITH READ ONLY

create or replace view empvd20 as select \* from emp t where t.deptno = 20 with read only



## 7.索引

索引是用于加速数据存取的数据对象。合理的使用索引可以大大降低i/o 次数,从而

提高数据访问性能。索引有很多种我们主要介绍常用的几种:

为什么添加了索引之后，会加快查询速度呢？

**图书馆**：如果杂乱地放书的话检索起来就非常困难,所以将书分类，然后再建一个箱子，箱

子里面放卡片，卡片里面可以按类查询,按书名查或者类别查,这样的话速度会快很多很多，

这个就有点像索引。索引的好处就是提高你找到书的速度，但是正是因为你建了索引，就应该有人专门来维护索引，维护索引是要有时间精力的开销的，也就是说索引是不能乱建的，所以建索引有个原则：如果有一个字段如果不经常查询，就不要去建索引。现在把书变成我们的表，把卡片变成我们的索引，就知道为什么索引会快，为什么会有开销。

创建索引的语法：

创建索引：

#### 1． 单列索引

单列索引是基于单个列所建立的索引，比如:

CREATE index 索引名 on 表名(列名)

#### 2． 复合索引

复合索引是基于两个列或多个列的索引。在同一张表上可以有多个索引，但是

要求列的组合必须不同,比如：

Create index emp\_idx1 on emp(ename,job);

Create index emp\_idx1 on emp(job,ename);

范例：给person表的name建立索引

create index pname\_index on person(name);

范例：给person表创建一个name和gender的索引

create index pname\_gender\_index on person(name, gender);

#### 3. 索引重建

1.索引重建的必要性

2.索引重建的语法

索引的使用原则：

* 在大表上建立索引才有意义
* 在where子句后面或者是连接条件上的字段建立索引(经常使用的列才建索引)
* 索引的层次不要超过4层

原理参考：二叉树 BTree B+树

参考网址：<http://blog.csdn.net/kennyrose/article/details/7532032>

## 8.PL/SQL

**procedure** [prə'siːdʒə] n. 程序，手续；步骤

**language** ['læŋgwɪdʒ] n. 语言；语言文字；表达能力

**declare** [dɪ'kleə] vt. 宣布，声明；断言，宣称vi. 声明，宣布

**begin** [bɪ'gɪn] vt. 开始

什么是PL/SQL?

PL/SQL（Procedure Language/SQL）

PLSQL是Oracle对sql语言的过程化扩展，指在SQL命令语言中增加了过程处理语句（如分支、循环等），使SQL语言具有过程处理能力。把SQL语言的数据操纵能力与过程语言的数据处理能力结合起来，使得PLSQL面向过程但比过程语言简单、高效、灵活和实用。

范例1：为职工涨工资，每人涨10％的工资。

update emp set sal=sal\*1.1

范例2：例2: 按职工的职称长工资,总裁涨1000元,经理涨800元，其他人员涨400元。

这样的需求我们就无法使用一条SQL来实现，需要借助其他程序来帮助完成，也可以使用pl/sql。

### 1.pl/sql程序语法

程序语法：

declare

说明部分 （变量说明，游标申明，例外说明 〕

begin

语句序列 （DML语句〕…

exception

例外处理语句

End;

### 2.常量和变量定义

在程序的声明阶段可以来定义常量和变量。

* 变量的基本类型就是ORACLE中的建表时字段的变量如char, varchar2, date, number, boolean, long

定义语法：varl char(15);

Psal number(9,2);

说明变量名、数据类型和长度后用分号结束说明语句。

常量定义：married constant boolean:=true

* 引用变量

Myname emp.ename%type;

引用型变量，即my\_name的类型与emp表中ename列的类型一样

在sql中使用into来赋值

declare emprec emp.ename%type;

begin

select t.ename into emprec from emp t where t.empno = 7369;

dbms\_output.put\_line(emprec);

end;

将查询到的语句赋值给emprec

* 记录型变量

Emprec emp%rowtype 代表一行

记录变量分量的引用

emp\_rec.ename:='ADAMS';

declare

p emp%rowtype;

begin

select \* into p from emp t where t.empno = 7369;

dbms\_output.put\_line(p.ename || ' ' || p.sal);

end;

**注意**  := 赋值符号等价于java中的=号;

= 逻辑等,判断两个值是否相等,等价于java中的==号

### 3. if分支

语法1：

**IF 条件 THEN**

**语句1;**

**语句2;**

**END IF;**

语法2：

**IF 条件 THEN**

**语句序列1；**

**ELSE**

**语句序列 2；**

**END IF；**

语法3：

**IF 条件 THEN 语句;**

**ELSIF 条件 THEN 语句;**

**ELSIF 条件 THEN 语句;**

**…**

**ELSE 语句;**

**END IF;**

范例1：如果从控制台输入1则输出我是1

accept num prompt ‘请输入一个数字’;

declare

pnum number := &num;

begin

if pnum = 1 then

dbms\_output.put\_line('我是1');

end if;

end;

&代表地址符,意思是指向num的值

接收的是字符串类型,此处会发生类型转换

范例3:判断人的不同年龄段18岁以下是未成年人，18岁以上40以下是成年人，40以上是老年人

accept num prompt ‘请输入一个年龄’;

declare

mynum number := &num;

begin

if mynum < 18 then

dbms\_output.put\_line('未成年人');

elsif mynum >= 18 and mynum < 40 then

dbms\_output.put\_line('中年人');

elsif mynum >= 40 then

dbms\_output.put\_line('老年人');

end if;

end;

### 4.LOOP **[luːp]**循环语句

其中语法2比较常用

语法1：

**WHILE 条件 LOOP**

**.. .**

**total : = total + salary;**

**END LOOP;**

语法2：

**Loop**

**EXIT when 条件;**

**……**

**End loop**

语法3：

**FOR I IN 1 . . 3 LOOP**

**语句序列 ;**

**END LOOP ;**

范例:使用语法1输出1到10的数字

declare

step number := 1;

begin

while step <= 10 loop

dbms\_output.put\_line(step);

step := step + 1;

end loop;

end;