

|* 学习目标

- * 能够理解PL/SQL的概述

 - * CUD-->解决网络瓶颈

- * 能够掌握PL/SQL的语法

 - * declare begin end;

 - * 变量：%type , %rowtype

 - * 常量：constant

 - * 运算符：:=,=,||, ..,=>

 - * 条件语句

 - if c then

 - elsif then

 - else

 - end if;

 - * 循环语句

 - * loop

 - ...

 - exit when ...

 - ..

 - end loop;

 - * while c loop

 - end loop;

 - * for m_count in[reverse] 1..5 loop

 - end loop;

 - * 异常

 - EXCEPTION

when ZERO_DEVIDE then

when ohters then

sqlcode,sqlerrm

* 自定义异常

age _exfeption EXCEPTION;

raise

* 能够掌握Oracle的游标

* Cursor

* 隐式 (CUD)

sql , %found,%notfound,%rowcount,%isopen

* 显示(R)

Cursor name is select

open

fetch(循环--->%notfound)

close

* 能够掌握Oracle的存储过程

* create or replace procedure pro_add(in,out)

is

...

begin

....

end;

* begin

pro_add;

pro_add(a,b,c,d,e);

end;

* 回顾

* 表与表之间链接

* 交叉链接 (笛卡尔积)

```
select * from employee,teacher;
```

* 内链接

```
select * from employee e inner join teacher t on e.tea_no=t.tea_no ;
```

```
select * from employee e join teacher t on e.tea_no=t.tea_no;
```

```
select * from employee e , teacher t where e.tea_no=t.tea_no;
```

* 外链接

* 左

```
select * from employee e left join teacher t on e.tea_no=t.tea_no;
```

```
select * from employee e , teacher t where e.tea_no=t.tea_no(+);
```

* 右

```
select * from employee e right join teacher t on e.tea_no=t.tea_no;
```

```
select * from employee e , teacher t where e.tea_no(+)=t.tea_no;
```

* 全

```
select * from employee e full join teacher t on e.tea_no=t.tea_no;
```

* 数据结构：B树，B-树，B+树

* B树：Oracle默认索引存储机制

* B+树; mysql默认索引存储机制、

* 索引

* 提供查询性能，避免全部扫描

```
* create unique|bitmap index index_name on employee(emp_age);
```

```
* create unique|bitmap index index_name on employee(emp_name,emp_age);
```

```
* drop index index_name ;
```

* SQL优化 (少让数据库翻译，不要让索引失效)

* 尽量不要写‘*’

* SQL语句用大写

- * 尽量使用内链接，少用外链接，少用子查询

- * 在链接表过程，where写在on右边

- * 索引相关

- * 尽量不要使用不等于 !=,<>

- * 尽量不要 like %dd%

- * order by 后面字段，尽量加索引

- * 在where 后面条件，判断时候，类型要一致

- * 尽量少用 not

- * 尽量少用 or , 1 or 2 in (1,2)

- * 索引列要设置成not null

- * 尽量少用 is null,is not null

- * 子查询当中，尽量用exists代替in

- * 序列

- * create sequence seq_no

- start with 1

- increment by 1

- maxvalue 5000

- cycle|nocycle

- cache 30

- * seq_no.nextval,seq_no.currval

- * 分区表

- create table employee(

- ...

-)

- partition by range(col_name)(

- partiton P1 values less than ()

- partiton P1 values less than ()

- partiton P1 values less than (maxvalue)

)

```
partition by list(col_name)(  
    partiton P1 values ()  
    partiton P1 values ()  
    partiton P1 values (default)  
)
```

* PL/SQL概述

* PL/SQL是 Procedure Language & Structured Query Language 的缩写

* PL /SQL是一种高级数据库程序设计语言，该语言专门用于在各种环境下对Oracle数据库进行访问。

* PL /SQL语言集成于数据库服务器中，所以PL/SQL代码可以对数据进行快速高效的处理。

* PL/SQL好处

* 对于客户/服务器环境来说，真正的瓶颈是网络上。

无论网络多快，只要客户端与服务器进行大量的数据交换。

应用运行的效率自然就回受到影响。如果使用PL/SQL进行编程，

将这种具有大量数据处理的应用放在服务器端来执行。

自然就省去了数据在网上的传输时间

* 能够掌握PL/SQL的语法

```
1 * pl/sql的构成  
2 [DECLARE]  
3     --声明变量等;  
4 BEGIN  
5     --程序主要部分，一般用来执行过程语句或SQL语句;  
6 [EXCEPTION]  
7     --异常处理;
```

```

8 END;
9
10 * 运算符
11 =      等于
12      比较运算符
13 <>,!=,~=,^=      不等于
14 <      小于
15 >      大于
16 <=     小于或等于
17 >=     大于或等于
18 +      加号      算术运算符
19 -      减号
20 *      乘号
21 /      除号
22 :=     赋值号      赋值运算符
23 =>     关系号      关系号
24 ..     范围运算符      范围运算符
25 ||     字符连接符      连接运算符
26 is null      是空值      逻辑运算符
27 between and      介于两者之间
28 in        在一系列值中间
29 and       逻辑与
30 or        逻辑或
31 not       取反
32
33 *变量与常量
34 * 变量: variable_name data_type[(size)][:=init_value];
35 * 常量: variable_name CONSTANT data_type[(size)] :=init_value;
36
37 * 数据类型
38 * 常用标准类型: CHAR(CHARATER,NCHAR),VARCHAR2,NUMBER(P,S),DATE,BOOLEAN等
39 * 属性类型: %TYPE与%ROWTYPE
40 * %TYPE:可以用来定义数据变量的类型与已定义的数据变量(表中的列)一致。
41 * %ROWTYPE: 与某一数据库表的结构一致(修改数据库表结构,可以实时保持一致)
42      访问方式声明为rowtype的变量名.字段名.
43
44 * 常量,变量,%TYPE,%ROWTYPE基本使用
45 /*
46      准备工作: 创建员工表, 插入测试数据
47 */

```

```

48 --drop table employee;
49 CREATE TABLE employee
50 (
51     emp_no VARCHAR2(8) PRIMARY KEY NOT NULL,    --工号，主键，非空
52     emp_name VARCHAR2(30) NOT NULL, --姓名,非空
53     emp_id VARCHAR2(18), --身份证号，代表18位整数
54     emp_age NUMBER(3,0) --年龄
55 );
56 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg001','张大','441
57 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg002','张二','441
58 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg003','张三','441
59 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg004','张四','441
60 commit;
61 /*
62 定义常量或变量、赋值使用示例：定义常量c_name直接存放员公司名字，
63 定义变量e_name,e_age分别用来存放工号为lh001的员工（通过查询表）的姓名及年龄，
64 定义变量e_mark存放员工成绩，设定员工成绩为95.5,最后输出该员工的公司、姓名、年龄及成绩
65 */
66 declare
67     c_name constant varchar(20):='亮哥教育';
68     e_name varchar(30);
69     e_age number(3,0);
70     e_mark number(3,1);
71 begin
72     --赋值方式一：使用SELECT INTO给变量赋值
73     select emp_name,emp_age into e_name,e_age from employee
74     where emp_no='lg001';
75     --赋值方式二：使用赋值操作符“:=”给变量赋值
76     e_mark:=95.5;
77     --输出相关信息，DBMS_OUTPUT.PUT_LINE为具有输出功能的函数
78     dbms_output.put_line('公司:' || c_name);
79     dbms_output.put_line('姓名:' || e_name);
80     dbms_output.put_line('年龄:' || e_age);
81     dbms_output.put_line('分数:' || e_mark);
82 end;
83 结果：
84 公司:亮哥教育
85 姓名:张大
86 年龄:19
87 分数:95.5

```

```
88
89
90 --%TYPE的使用示例：在上一示例基础上
91 declare
92     c_name constant varchar2(20):='亮哥教育';
93     e_name employee.emp_name%type;
94     e_age employee.emp_age%type;
95     e_mark number(3,1);
96 begin
97     select emp_name,emp_age into e_name,e_age from employee
98     where emp_no='lg001';
99     e_mark:=95.5;
100     dbms_output.put_line('公司:'||c_name);
101     dbms_output.put_line('姓名:'||e_name);
102     dbms_output.put_line('年龄:'||e_age);
103     dbms_output.put_line('分数:'||e_mark);
104 end;
```

105 结果:

106 公司:亮哥教育

107 姓名:张大

108 年龄:19

109 分数:95.5

110

111 --%ROWTYPE的使用

```
112 declare
113     c_name constant varchar2(20):='亮哥教育';
114     empinfo employee%rowtype;
115     e_mark number(3,1);
116 begin
117     select * into empinfo from employee
118     where emp_no='lg001';
119     e_mark:=95.5;
120     dbms_output.put_line('公司:'||c_name);
121     dbms_output.put_line('姓名:'||empinfo.emp_name);
122     dbms_output.put_line('年龄:'||empinfo.emp_age);
123     dbms_output.put_line('分数:'||e_mark);
124 end;
```

125 结果:

126 公司:亮哥教育

127 姓名:张大


```

128 年龄:19
129 分数:95.5
130
131
132 * 控制语句
133 语法1:
134 if <boolean表达式> then
135     执行语句;
136 end if;
137 语法2:
138 if <boolean表达式 >then
139     执行语句;
140 else
141     执行语句;
142 end if;
143 语法3:
144 if <boolean表达式> then
145     执行语句;
146 elsif <boolean表达式> then
147     执行语句;
148 elsif <boolean表达式> then
149     执行语句;
150 else
151     执行语句;
152 end if;
153 使用示例
154 1. 判断工号为lg002的员工的小孩是否可以上小学（国家规定需要6周岁才可以上小学），如果可以
155 2. 判断工号为lg001的员工的小孩是否可以上小学（国家规定需要6周岁才可以上小学），不论可否
156 3. 判断工号为lg001员工的小孩是否可以上小学（国家规定需要6周岁才可以上小学），超过6岁
157
158 /*
159 准备工作：创建员工表，插入测试数据
160 */
161 --drop table employee;
162 CREATE TABLE employee
163 (
164     emp_no CHAR(8) PRIMARY KEY NOT NULL,    --工号，主键，非空
165     emp_name VARCHAR2(30) NOT NULL, --姓名,非空
166     emp_id VARCHAR2(18), --身份证号，代表18位整数
167     child_age NUMBER(3,0) --年龄

```

```

168 );
169 insert into employee(emp_no,emp_name,emp_id,child_age) values('lg001','张大','4
170 insert into employee(emp_no,emp_name,emp_id,child_age) values('lg002','张二','4
171 insert into employee(emp_no,emp_name,emp_id,child_age) values('lg003','张三','4
172 insert into employee(emp_no,emp_name,emp_id,child_age) values('lg004','张四','4
173 commit;
174
175 /*
176 示例练习:
177 1. 判断工号为lg002的员工的小孩是否可以上小学（国家规定需要6周岁才可以上小学），如果可以
178 2. 判断工号为lg001的员工的小孩是否可以上小学（国家规定需要6周岁才可以上小学），不论可
179 3. 判断工号为lg001员工的小孩是否可以上小学（国家规定需要6周岁才可以上小学），超过6岁
180 根据员工情况输入相关结果。
181 */
182 --练习1
183 declare
184     e_age employee.emp_age%type;
185 begin
186     select emp_age into e_age from employee
187     where emp_no='lg002';
188     if e_age>=6 then
189         dbms_output.put_line('此员工年龄为:'||e_age||'岁,可以上小学');
190     end if;
191 end;
192 结果:
193 此员工孩子年龄为:6岁,可以上小学
194
195 --练习2
196 declare
197     c_age employee.child_age%type;
198 begin
199     select child_age into c_age from employee
200     where emp_no='lg001';
201     if c_age>=6 then
202         dbms_output.put_line('此员工孩子年龄为:'||c_age||'岁,可以上小学');
203     else
204         dbms_output.put_line('此员工孩子年龄为:'||c_age||'岁,不可以上小学');
205     end if;
206 end;
207 结果:

```

```

208 此员工孩子年龄为:5岁,不可以上小学
209
210 --练习3
211 declare
212     c_age employee.child_age%type;
213 begin
214     select child_age into c_age from employee
215     where emp_no='lg003';
216     if c_age>6 then
217         dbms_output.put_line('此员工孩子年龄为:'||c_age||'岁,超龄');
218     elsif c_age=6 then
219         dbms_output.put_line('此员工孩子年龄为:'||c_age||'岁,可以上小学');
220     else
221         dbms_output.put_line('此员工孩子年龄为:'||c_age||'岁,不可以上小学');
222     end if;
223 end;
224 结果:
225 此员工孩子年龄为:7岁,超龄
226
227
228 * 循环控制
229 loop循环
230 语法:
231 loop
232     执行语句;
233     exit when <条件语句
234 end loop;
235 示例:
236 由于某次考试难度太大,需要给整个班级的员工加分,每次加5分,但最高分不能超过90分
237
238 /*
239 准备工作: 创建员工表,插入测试数据
240 */
241 --drop table employee;
242 CREATE TABLE employee
243 (
244     emp_no CHAR(8) PRIMARY KEY NOT NULL,    --工号,主键,非空
245     emp_name VARCHAR2(30) NOT NULL,--姓名,非空
246     emp_id VARCHAR2(18), --身份证号,代表18位字符
247     emp_mark NUMBER(3,1)  --分数

```

```

248 );
249 insert into employee(emp_no,emp_name,emp_id,emp_mark) values('lg001','张大','44
250 insert into employee(emp_no,emp_name,emp_id,emp_mark) values('lg002','张二','44
251 insert into employee(emp_no,emp_name,emp_id,emp_mark) values('lg003','张三','44
252 insert into employee(emp_no,emp_name,emp_id,emp_mark) values('lg004','张四','44
253 commit;
254 select * from employee;
255
256 /*
257 由于某次考试难度太大，需要给整个班级的员工加分，每次加5分，但最高分不能超过90分
258 */
259 declare
260 e_mark employee.emp_mark%type;
261 begin
262 loop
263     select max(emp_mark) into e_mark from employee;
264     exit when (e_mark+5)>90;
265     update employee set emp_mark=emp_mark+5;
266 end loop;
267 commit;
268 end;
269
270
271 while循环
272 语法:
273 while <布尔表达式> loop
274     执行语句;
275 end loop;
276
277 使用示例
278 继续使用上面的表，及完成：“由于某次考试难度太大，需要给整个班级的员工加分，每次加5分，
279 declare
280 e_mark employee.emp_mark%type:=0;
281 begin
282     select max(emp_mark) into e_mark from employee;
283     while (e_mark+5)<=90 loop
284         update employee set emp_mark=emp_mark+5;
285         select max(emp_mark) into e_mark from employee;
286     end loop;
287     commit;

```

```

288 end;
289
290 for循环
291 语法:
292 for 循环计数器 in [REVERSE] 下限..上限 loop
293     执行语句;
294 end loop;
295 说明: ..两点表示范围, 1..4表示时将从1到4进行循环, 小(例如 1)写前边, REVERSE表示反转
296 使用示例
297 declare
298     m_count number(3,0);
299 begin
300     for m_count in 1..4 loop
301         dbms_output.put_line(m_count);
302     end loop;
303 end;
304 结果
305 1
306 2
307 3
308 4
309 declare
310     m_count number(3,0);
311 begin
312     for m_count in reverse 1..4 loop
313         dbms_output.put_line(m_count);
314     end loop;
315 end;
316 结果:
317 4
318 3
319 2
320 1
321
322 继续使用上面的表, 及完成: “由于某次考试难度太大, 需要给整个班级的员工加分, 每次加5分,
323
324 declare
325     m_timer number(3,0);
326     m_count number(3,0);
327     e_mark employee.emp_mark%type;

```

```

328 begin
329     select max(emp_mark) into e_mark from employee;
330     --dbms_output.put_line(e_mark);
331     -- e_mark:=51;
332     --m_timer:=(90-e_mark)/5;--假如e_mark:=51,m_timer=8,加了8次5之后等于91, 超过90,
333     -- floor 向下取整数的函数（不会四舍五入）;
334     m_timer:=floor((90-e_mark)/5);
335     for m_count in 1..m_timer loop
336         update employee set emp_mark=emp_mark+5;
337     end loop;
338     dbms_output.put_line(m_count);
339 end;
340
341 *顺序控制
342 用于按指定顺序执行的语句。主要是null语句
343 null语句：是一个可执行语句，相当于一个占位符或不执行操作的空语句。主要用来提高程序语句
344 --广东或香港等有些地方不喜欢4的数字，在使用数字时跳过4
345 declare
346     m_count number(3,0):=0;
347 begin
348     while m_count<10 loop
349         m_count:=m_count+1;
350         if m_count=4 then
351             null;--去掉null会的报错
352         else
353             dbms_output.put_line('m_count='||m_count);
354         end if;
355     end loop;
356 end;
357 结果：
358 m_count=1
359 m_count=2
360 m_count=3
361 m_count=5
362 m_count=6
363 m_count=7
364 m_count=8
365 m_count=9
366 m_count=10
367

```

```

368
369 * 异常处理
370 * 异常语法
371 Exception
372     when 异常名 then
373         执行处理语句;
374         --使用others确保不会漏过任何异常
375     when others then
376         执行处理语句;
377
378 经常配套使用的函数:
379     SQLCODE函数: 返回错误代码,
380     SQLERRM函数: 返回错误信息
381
382 例如输出异常信息: DBMS_OUTPUT.PUT_LINE('其它异常, 代码号: '||SQLCODE||', 异常描述:
383 * 预定义异常
384 预定义异常指PL/SQL 程序违反 Oracle 规则或超越系统限制时隐式引发(由oracle自动引发)。
385 常见的预定义异常:
386 ZERO_DIVIDE:以零作为除数时出现
387 DUP_VAL_ON_INDEX: 试图将重复的值存储在使用唯一索引的数据库列中
388 INVALID_NUMBER: 试图将一个非有效的字符串转换成数字
389 TOO_MANY_ROWS : 在执行SELECT INTO语句后返回多行时出现
390 VALUE_ERROR:变量的值超出变量大小
391 CURSOR_ALREADY_OPEN:试图打开已经打开的游标
392 ACCESS_INTO_NULL: 试图给一个没有初始化的对象赋值
393 LOGIN_DENIED : 使用无效的用户名和口令登录Oracle
394 NO_DATA_FOUND : 语句无法返回请求的数据
395
396
397 使用示例
398 --预定义异常
399 declare
400     s_result number(3,0);
401 begin
402     s_result:=10/0;
403     dbms_output.put_line('没有异常');
404 Exception
405     when zero_divide then
406         dbms_output.put_line('除数不能为 0 , 请核查');
407     when others then

```

```

408      --SQLCODE函数返回错误代码，SQLERRM函数返回错误信息
409      dbms_output.put_line('其他异常代码号:'||sqlcode||',异常信息:'||sqlerrm);
410 end;
411 结果:
412 除数不能为0，请核查
413
414 * 自定义异常
415 自定义异常：程序在运行过程中，编程人员根据业务等情况，认为非正常情况，可以自定义异常。
416 第一、定义相关异常；在声明部分定义相关异常，格式：<自定义异常名称> EXCEPTION;
417 第二、抛出异常；在出现异常部分抛出异常，格式：RAISE <异常名称>;
418 第三、处理异常；在异常处理部分对异常进行处理，格式：when <自定义异常名称> then ...，
419
420 使用示例，完成以下功能：
421 根据业务的需要，提取员工的信息时，需要校验年龄不能为负数，
422 如果为负数，请自定义异常age_exception,并将其抛出，处理。
423 例如以如下的测试数据，取出身份证号：'441521199909092113'的员工，判断其年龄，并进行异
424 /*
425 创建员工表，插入测试数据
426 */
427 --drop table employee;
428 CREATE TABLE employee
429 (
430     emp_no CHAR(8) PRIMARY KEY NOT NULL,    --工号，主键，非空
431     emp_name VARCHAR2(30) NOT NULL, --姓名,非空
432     emp_id VARCHAR2(18), --身份证号，代表18位整数
433     emp_age NUMBER(3,0) --年龄
434 );
435 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg001','张大','441
436 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg002','张二','441
437 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg003','张三','441
438 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg04','张四','4415
439 commit;
440
441 declare
442     c_name constant varchar2(20):='亮哥教育';
443     e_name employee.emp_name%type;
444     e_age employee.emp_age%type;
445     e_mark number(3,1);
446     age_exception EXCEPTION;
447 begin

```



```

448 select emp_name,emp_age into e_name,e_age from employee
449 where emp_id='441521199909092111';
450 --年龄小于0岁为异常情况，使用raise抛出异常
451 if e_age < 0 then
452     raise age_exception;
453 end if;
454 e_mark:=95.5;
455 dbms_output.put_line('公司:'||c_name);
456 dbms_output.put_line('姓名:'||e_name);
457 dbms_output.put_line('年龄:'||e_age);
458 dbms_output.put_line('分数:'||e_mark);
459 EXCEPTION
460     when age_exception then
461         -- dbms_output.put_line('异常描述：年龄不能为负数！');
462         RAISE_APPLICATION_ERROR(-20001,'年龄不能为负数！');
463     when TOO_MANY_ROWS then
464         dbms_output.put_line('异常描述'||Sqlerrm||',原因：身份证号码出现重复');
465     when others then
466         dbms_output.put_line('其他异常代码号:'||sqlcode||',异常信息:'||sqlerrm);
467 end;
468
469 结果：
470     异常描述：年龄不能为负数！

```

* 能够掌握Oracle的游标

* 游标概述

* 游标是指oracle在执行增删改查操作时，会把执行结果放在内存分配的缓冲区中，游标就是指向该区的一个指针，可以对结果集的每一行数据分别进行处理。

* 游标属性：

- * %found 是否存在结果集或影响的行数，如果存在返回true
- * %notfound 是否存在结果集或影响的行数，如果不存在返回true
- * %rowcount 返回受影响的行数
- * %isopen 游标是否已经打开。隐式游标中一般是自动打开和关闭的，查询都返回

False

* 游标的分类

* 隐式游标

* 由Oracle在内部声明，数据库自动创建，管理。

* 主要用途是用于增删改数据时，可以返回一个操作成功或失败的相关信息，名称是 (sql)

* 显式游标

* 可以定义游标，自己使用

```
1 * 隐式游标
2 示例：访问游标的属性
3 /*
4  创建员工表，插入测试数据
5 */
6 --drop table employee;
7 CREATE TABLE employee
8 (
9     emp_no CHAR(8) PRIMARY KEY NOT NULL,    --工号，主键，非空
10    emp_name VARCHAR2(30) NOT NULL, --姓名,非空
11    emp_id VARCHAR2(18), --身份证号，代表18位整数
12    emp_age NUMBER(3,0) --年龄
13 );
14 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg001','张大','441
15 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg002','张二','441
16 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg003','张三','441
17 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg004','张四','441
18 commit;
19
20 --隐匿游标使用
21 declare
22 begin
23     --insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg005','张6'
24     update employee set emp_age=17 where emp_id='441521199909092111' ;
25     --delete from employee where emp_id='441521199909092115' ;
26     if sql%found then
27         dbms_output.put_line('已经影响了数据');
28     end if;
29     if sql%notfound then
30         dbms_output.put_line('没有影响数据');
```

```

31     end if;
32     if sql%isopen then
33         dbms_output.put_line('游标open了');--隐式游标自动打开和关闭的，此时查询已经
34     else
35         dbms_output.put_line('游标close了');
36     end if;
37     dbms_output.put_line('受影响的行:' || sql%rowcount);
38 end;
39
40 * 显示游标
41 使用步骤：
42 * 在declare 声明游标：Cursor 游标名 is 查询语句(可以多个查询字段);
43 * 打开游标：open 游标名;
44 * 提取游标：fetch 游标名 into 变量1, 变量2（变量的个数对应字段数）;
45 * 关闭游标：close 游标名;
46 示例：
47 /*
48 创建员工表，插入测试数据
49 */
50 --drop table employee;
51 CREATE TABLE employee
52 (
53     emp_no varchar2(8) PRIMARY KEY NOT NULL,    --工号，主键，非空
54     emp_name VARCHAR2(30) NOT NULL,--姓名,非空
55     emp_id VARCHAR2(18), --身份证号，代表18位整数
56     emp_age NUMBER(3,0) --年龄
57 );
58 insert into employee(emp_no,emp_name,emp_id,emp_age) values('gh001','张大','441
59 insert into employee(emp_no,emp_name,emp_id,emp_age) values('gh002','张二','441
60 insert into employee(emp_no,emp_name,emp_id,emp_age) values('gh003','张三','441
61 insert into employee(emp_no,emp_name,emp_id,emp_age) values('gh004','张四','441
62 commit;
63
64 --使用显式游标遍历（loop），查询并输出所有员工的工号和姓名
65 declare
66     e_no employee.emp_no%type;
67     e_name employee.emp_name%type;
68     Cursor emp_cursor is select emp_no,emp_name from employee;-- 定义游标
69 begin
70     -- 打开游标

```

```

71 open emp_cursor;
72 loop
73     fetch emp_cursor into e_no,e_name;
74     exit when emp_cursor%notfound;
75     dbms_output.put_line('工号:'||e_no||',姓名:'||e_name);
76 end loop;
77 -- 关闭游标
78 close emp_cursor;
79 end;
80 结果:
81 工号:lg001,姓名:张大
82 工号:lg002,姓名:张二
83 工号:lg003,姓名:张三
84 工号:lg004,姓名:张四
85
86 /*
87 使用显式游标(for)，查询并输出所有员工的工号和姓名，
88 for循环游标会隐式打开游标、自动创建%ROWTYPE类型变量对应记录行，
89 处理完所有行后会自动关闭游标.使用起来较方便。
90 */
91 declare
92     Cursor emp_cursor is select emp_no,emp_name from employee;
93 begin
94     for empinfo in emp_cursor loop
95         dbms_output.put_line('工号:'||empinfo.emp_no||',姓名:'||empinfo.emp_name);
96     end loop;
97 end;
98 工号:lg001,姓名:张大
99 工号:lg002,姓名:张二
100 工号:lg003,姓名:张三
101 工号:lg004,姓名:张四
102
103 --使用显式游标修改数据，把年龄为负的员工年龄改为18岁,其它都加一岁
104 declare
105     Cursor emp_cursor is select emp_no,emp_age from employee for update;
106 begin
107     for empinfo in emp_cursor loop
108         if empinfo.emp_age<0 then
109             update employee set emp_age=18 where emp_no=empinfo.emp_no;
110         else

```

```

111         update employee set emp_age=emp_age+1 where emp_no=empinfo.emp_no;
112     end if;
113 end loop;
114 end;
115

```

* 能够掌握Oracle的存储过程

* 存储过程概述

* 存储过程属于已命名的pl/sql程序块，封装数据业务操作，具有模块化、可重用、可维护、更安全特点。

* 存储过程类型

* 不带参数

* 带输入参数

* 带输出参数

* 带输入输出参数

```

1 * 创建与调用语法
2 * 创建语法
3 CREATE [OR REPLACE] PROCEDURE procedure_name[(param_list)]
4     IS|AS
5     ....
6 BEGIN
7     执行语句;
8 [EXCEPTION]
9     异常处理;
10 END[procedure_name];
11 说明:
12 OR REPLACE: 如果系统已存在该存储过程，将被替换
13 procedure_name: 存储过程名称
14 param_list: 参数列表，参数不需要声明长度，可选
15 DECLARE: 局部声明，可选
16 * 调用语法
17 * 命令行调用:
18     * exec|execute procedure_name(paramlist);
19 * pl/sql块调用:

```

```
20  begin
21      procedure_name(paramlist);
22  end;
23
24  * 测试
25  /*
26      创建员工表，插入测试数据
27  */
28  --drop table employee;
29  CREATE TABLE employee
30  (
31      emp_no varchar2(8) PRIMARY KEY NOT NULL,    --工号，主键，非空
32      emp_name VARCHAR2(30) NOT NULL, --姓名,非空
33      emp_id VARCHAR2(18), --身份证号，代表18位整数
34      emp_age NUMBER(3,0) --年龄
35  );
36  insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg001','张大','441
37  insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg002','张二','441
38  insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg003','张三','441
39  insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg004','张四','441
40  commit;
41
42  * 无参存储过程
43  --新建存储过程(不带参数)，可以新增员工信息
44  create or replace procedure pro_add_procedure is
45  begin
46      insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg005','张五','4
47      commit;
48  end;
49
50  --调用存储过程
51  begin
52      pro_add_procedure;
53  end;
54  select * from employee;
55
56  * 有输入参数存储过程
57  --新建存储过程(带输入参数,输入参数类型in可以省略不写)，可以新增员工信息
58  create or replace procedure pro_add_procedure(
59      e_no employee.emp_no%type,
```

```

60     e_name employee.emp_name%type,
61     e_id employee.emp_id%type,
62     e_age employee.emp_age%type
63 ) is
64 begin
65     insert into employee(emp_no,emp_name,emp_id,emp_age) values(e_no,e_name,e_id,
66     commit;
67 end;
68 /
69 --调用存储过程
70 begin
71     --按位置传递参数
72     -- pro_add_procedure('lg008','凌凌漆','441521199909092115',20);
73     --按名称传递参数,顺序可变
74     pro_add_procedure(e_no=>'lg009',e_id => '441521199909092115',e_name => '凌凌
75 end;
76
77 * 有输出参数存储过程
78 --新建存储过程(带输出参数)，可以新增员工信息
79 create or replace procedure pro_add_procedure(
80     s_flag out number,
81     s_message out varchar
82 )
83 is
84 begin
85     insert into employee(emp_no,emp_name,emp_id,emp_age) values('a010','张10','44
86     commit;
87     s_flag:=1;
88     s_message:='添加成功';
89     EXCEPTION
90     when DUP_VAL_ON_INDEX then
91         s_flag:=20001;
92         s_message:='工号已经被使用，请核查! ';
93     when others then
94         s_flag:=sqlcode;
95         s_message:=sqlerrm;
96 end;
97
98 /
99 --调用存储过程

```

```

100 declare
101     p_flag number(10,0);
102     p_message varchar2(50);
103 begin
104     pro_add_procedure(p_flag,p_message);
105     dbms_output.put_line(p_flag||':'||p_message);
106 end;
107
108 * 有输入输出参数存储过程
109 --新建存储过程(带输入输出参数,输入参数类型in可以省略不写), 可以新增员工信息
110 create or replace procedure pro_add_procedure(
111     e_no employee.emp_no%type,
112     e_name employee.emp_name%type,
113     e_id employee.emp_id%type,
114     e_age employee.emp_age%type,
115     s_flag out number,
116     s_message out varchar
117 )is
118 begin
119     insert into employee(emp_no,emp_name,emp_id,emp_age) values(e_no,e_name,e_id,
120     commit;
121     s_flag:=1;
122     s_message:='添加成功';
123     Exception
124         when DUP_VAL_ON_INDEX then
125             s_flag:=20001;
126             s_message:='工号已经被使用, 请核查! ';
127         when others then
128             s_flag:=sqlcode;
129             s_message:=sqlerrm;
130 end;
131 /
132 --调用存储过程
133 declare
134     p_flag number(10,0);
135     p_message varchar2(50);
136 begin
137     pro_add_procedure('lg1231','小白','441521199909092119',30,p_flag,p_message);
138     dbms_output.put_line(p_flag||':'||p_message);
139 end;

```


140

141 * 删除存储过程

142 `drop` procedure pro_add_procedure;