(1)SEQNAME.NEXTVAL里面的值如何读出来？可以直接在insert into test values(SEQNAME.NEXTVAL) 是可以用 这样：

SELECT tmp#\_seq.NEXTVAL

INTO id\_temp

FROM DUAL; 然后可以用id\_temp

(2)PLS-00103: 出现符号 ">"在需要下列之一时：

代码如下：

IF (sum>0)

THEN

begin

INSERT INTO emesp.tp\_sn\_PRoduction\_log

VALUES (r\_serial\_number, , id\_temp);

EXIT;

end;

一直报sum>0 这是个很郁闷的问题 因为变量用了sum 所以不行，后改为i\_sum>0

(3)Oracle 语法

1. Oracle应用编辑方法概览

答：1) Pro\*C/C++/... : C语言和数据库打交道的方法，比OCI更常用;

2) ODBC

3) OCI: C语言和数据库打交道的方法，和ProC很相似，更底层，很少用;

4) SQLJ: 很新的一种用java访问Oracle数据库的方法，会的人不多;

5) JDBC

6) PL/SQL: 存储在数据内运行, 其他方法为在数据库外对数据库访问;

2. PL/SQL

答：1) PL/SQL(Procedual language/SQL)是在标准SQL的基础上增加了过程化处理的语言;

2) Oracle客户端工具访问Oracle服务器的操作语言;

3) Oracle对SQL的扩充;

4. PL/SQL的优缺点

答：优点：

1) 结构化模块化编程，不是面向对象;

2) 良好的可移植性(不管Oracle运行在何种操作系统);

3) 良好的可维护性(编译通过后存储在数据库里);

4) 提升系统性能;

第二章

PL/SQL程序结构

1. PL/SQL块

答：1) 申明部分, DECLARE(不可少); 2) 执行部分, BEGIN...END;

3) 异常处理，EXCEPTION(可以没有);

2. PL/SQL开发环境

答：可以运用任何纯文本的编辑器编辑，例如：VI ；toad很好用

3. PL/SQL字符集

答：PL/SQL对大小写不敏感

4. 标识符命名规则

答：1) 字母开头;

2) 后跟任意的非空格字符、数字、货币符号、下划线、或# ;

3) 最大长度为30个字符(八个字符左右最合适);

5. 变量声明

答：语法

Var\_name type [CONSTANT][NOT NULL][:=value];

注：1) 申明时可以有默认值也可以没有;

2) 如有[CONSTANT][NOT NULL], 变量一定要有一个初始值;

3) 赋值语句为“:=”;

4) 变量可以认为是数据库里一个字段;

5) 规定没有初始化的变量为NULL;

第三章

1. 数据类型

答：1) 标量型：数字型、字符型、布尔型、日期型;

2) 组合型：RECORD(常用)、TABLE(常用)、VARRAY(较少用)

3) 参考型：REF CURSOR(游标)、REF object\_type

4) LOB(Large Object)

2. %TYPE

答：变量具有与数据库的表中某一字段相同的类型

例：v\_FirstName studengts.first\_name%TYPE;

3. RECORD类型

答：TYPE record\_name IS RECORD( /\*其中TYPE，IS，RECORD为关键字，record\_name为变量名称\*/

field1 type [NOT NULL][:=expr1], /\*每个等价的成员间用逗号分隔\*/

field2 type [NOT NULL][:=expr2], /\*如果一个字段限定NOT NULL，那么它必须拥有一个初始值\*/

... /\*所有没有初始化的字段都会初始为NULL

fieldn type [NOT NULL][:=exprn]);

4. %ROWTYPE

答：返回一个基于数据库定义的类型

DECLARE

v\_StuRec Student%ROWTYPE; /\*Student为表的名字\*/

注：与3中定一个record相比，一步就完成，而3中定义分二步：a. 所有的成员变量都要申明; b. 实例化变量;

5. TABLE类型

答：TYPE tabletype IS TABLE OF type INDEX BY BINARY\_INTEGER;

例：DECLARE

TYPE t\_StuTable IS TABLE OF Student%ROWTYPE INDEX BY BINARY\_INTERGER;

v\_Student t\_StuTable;

BEGIN

SELECT \* INTO v\_Student(100) FROM Student WHERE id = 1001;

END;

注：1) 行的数目的限制由BINARY\_INTEGER的范围决定;

6. 变量的作用域和可见性

答：1) 执行块里可以嵌入执行块;

2) 里层执行块的变量对外层不可见;

3) 里层执行块对外层执行块变量的修改会影响外层块变量的值;

第四章

1. 条件语句

答：IF boolean\_expression1 THEN

...

ELSIF boolean\_expression2 THEN /\*注意是ELSIF，而不是ELSEIF\*/

... /\*ELSE语句不是必须的，但END IF;是必须的\*/

ELSE ...

END IF;

2. 循环语句

答：1) Loop

..

IF boolean\_expr THEN /\* \*/

EXIT; /\* EXIT WHEN boolean\_expr \*/

END IF; /\* \*/

END LOOP;

2) WHILE boolean\_expr LOOP

...

END LOOP;

3) FOR loop\_counter IN [REVERSE] low\_blound..high\_bound LOOP

...

END LOOP;

注：a. 加上REVERSE 表示递减，从结束边界到起始边界，递减步长为一;

b. low\_blound 起始边界; high\_bound 结束边界;

3. GOTO语句

答：GOTO label\_name;

1) 只能由内部块跳往外部块;

2) 设置标签：<<label\_name>>

3) 示例：

LOOP

...

IF D%ROWCOUNT = 50 THEN

GOTO l\_close;

END IF;

...

END LOOP;

<<l\_close>>;

...

4. NULL语句

答：在语句块中加空语句，用于补充语句的完整性。示例：

IF boolean\_expr THEN

...

ELSE

NULL;

END IF;

5. SQL in PL/SQL

答：1) 只有DML SQL可以直接在PL/SQL中使用;

第五章

1. 游标(CURSOR)

答：1) 作用：用于提取多行数据集;

2) 声明：a. 普通申明：DELCARE CURSOR CURSOR\_NAME IS select\_statement /\* CURSOR的内容必须是一条查询语句\*/

b. 带参数申明：DELCARE CURSOR c\_stu(p\_id student.ID%TYPE) SELECT \* FROM student WHERE ID = p\_id;

3) 打开游标：OPEN Cursor\_name; /\*相当于执行select语句，且把执行结果存入CURSOR;

4) 从游标中取数：a. FETCH cursor\_name INTO var1, var2, ...; /\*变量的数量、类型、顺序要和Table中字段一致;\*/

b. FETCH cursor\_name INTO record\_var;

注：将值从CURSOR取出放入变量中，每FETCH一次取一条记录;

5) 关闭游标: CLOSE Cursor\_name;

注：a. 游标使用后应该关闭;

b. 关闭后的游标不能FETCH和再次CLOSE;

c. 关闭游标相当于将内存中CURSOR的内容清空;

2. 游标的属性

答：1) %FOUND: 是否有值;

2) %NOTFOUND: 是否没有值;

3) %ISOPEN: 是否是打开状态;

4) %ROWCOUNT: CURSOR当前的记录号;

3. 游标的FETCH循环

答：1) LOOP

FETCH cursor INTO ...

EXIT WHEN cursor%NOTFOUND; /\*当cursor中没记录后退出\*/

END LOOP;

2) WHILE cursor%FOUND LOOP

FETCH cursor INTO ...

END LOOP;

3) FOR var IN cursor LOOP

FETCH cursor INTO...

END LOOP;

第六章

1. 异常

答：DECLARE

...

e\_TooManyStudents EXCEPTION; /\* 申明异常 \*/

...

BEGIN

...

RAISE e\_TooManyStudents; /\* 触发异常 \*/

...

EXCEPTION

WHEN e\_TooManyStudents THEN /\* 触发异常 \*/

...

WHEN OTHERS THEN /\* 处理所有其他异常 \*/

...

END;

2004-9-8 星期三 阴

PL/SQL数据库编程(下)

1. 存储过程(PROCEDURE)

答：创建过程：

CREATE [OR REPLACE] PROCEDURE proc\_name

[(arg\_name[{IN|OUT|IN OUT}]TYPE,

arg\_name[{IN|OUT|IN OUT}]TYPE)]

{IS|AS}

procedure\_body

1) IN: 表示该参数不能被赋值(只能位于等号右边);

2) OUT:表示该参数只能被赋值(只能位于等号左边);

3) IN OUT: 表示该类型既能被赋值也能传值;

2. 存储过程例子

答：CREATE OR REPLACE PROCEDURE ModeTest(

p\_InParm IN NUMBER,

p\_OutParm OUT NUMBER,

p\_InOutParm IN OUT NUMBER)

IS

v\_LocalVar NUMBER; /\* 声明部分 \*/

BEGIN

v\_LocalVar:=p\_InParm; /\* 执行部分 \*/

p\_OutParm:=7;

p\_InOutParm:=7;

...

EXCEPTION

... /\* 异常处理部分 \*/

END ModeTest;

3. 调用PROCEDURE的例子

答：1) 匿名块可以调;

2) 其他PROCDEURE可以调用;

例：

DECLARE

v\_var1 NUMBER;

BEGIN

ModeTest(12, v\_var1, 10);

END;

注：此时v\_var1等于7

4. 指定实参的模式

答：1) 位置标示法：调用时添入所有参数，实参与形参按顺序一一对应;

2) 名字标示法：调用时给出形参名字，并给出实参

ModeTest(p\_InParm=>12, p\_OutParm=>v\_var1, p\_Inout=>10);

注：a. 两种方法可以混用;

b. 混用时第一个参数必须通过位置来指定。

5. 函数(Function)与过程(Procedure)的区别

答：1) 过程调用本身是一个PL/SQL语句(可以在命令行中通过exec语句直接调用);

2) 函数调用是表达式的一部分;

6. 函数的声明

答：CREATE [OR REPLACE] PROCEDURE proc\_name

[(arg\_name[{IN|OUT|IN OUT}]TYPE,

arg\_name[{IN|OUT|IN OUT}]TYPE)]

RETURN TYPE

{IS|AS}

procedure\_body

注：1) 没有返回语句的函数将是一个错误;

7. 删除过程与函数

答：DROP PROCEDURE proc\_name;

DROP FUNCTION func\_name;

第八章

1. 包

答：1) 包是可以将相关对象存储在一起的PL/SQL的结构;

2) 包只能存储在数据库中，不能是本地的;

3) 包是一个带有名字的声明;

4) 相当于一个PL/SQL块的声明部分;

5) 在块的声明部分出现的任何东西都能出现在包中;

6) 包中可以包含过程、函数、游标与变量;

7) 可以从其他PL/SQL块中引用包，包提供了可用于PL/SQL的全局变量。

8) 包有包头和包主体，如包头中没有任何函数与过程，则包主体可以不需要。

2. 包头

答：1) 包头包含了有关包的内容的信息，包头不含任何过程的代码。

2) 语法：

CREATE [OR REPLACE] PACKAGE pack\_name {IS|AS}

procedure\_specification|function\_specification|variable\_declaration|type\_definition|exception\_declaration|cursor\_declaration

END pack\_name;

3) 示例：

CREATE OR REPLACE PACKAGE pak\_test AS

PROCEDURE RemoveStudent(p\_StuID IN students.id%TYPE);

TYPE t\_StuIDTable IS TABLE OF students.id%TYPE INDEX BY BINARY\_INTEGER;

END pak\_test;

3. 包主体

答：1) 包主体是可选的，如包头中没有任何函数与过程，则包主体可以不需要。

2) 包主体与包头存放在不同的数据字典中。

3) 如包头编译不成功，包主体无法正确编译。

4) 包主体包含了所有在包头中声明的所有过程与函数的代码。

5) 示例：

CREATE OR REPLACE PACKAGE BODY pak\_test AS

PROCEDURE RemoveStudent(p\_StuID IN students.id%TYPE) IS

BEGIN

...

END RemoveStudent;

TYPE t\_StuIDTable IS TABLE OF students.id%TYPE INDEX BY BINARY\_INTEGER;

END pak\_test;

4. 包的作用域

答：1) 在包外调用包中过程(需加包名)：pak\_test.AddStudent(100010, 'CS', 101);

2) 在包主体中可以直接使用包头中声明的对象和过程(不需加包名);

5. 包中子程序的重载

答：1) 同一个包中的过程与函数都可以重载;

2) 相同的过程或函数名字，但参数不同;

6. 包的初始化

答：1) 包存放在数据库中;

2) 在第一次被调用的时候，包从数据库中调入内存并被初始化;

3) 包中定义的所有变量都被分配内存;

4) 每个会话都将拥有自己的包内变量的副本。

第九章

1. 触发器

答：1) 触发器与过程/函数的相同点

a. 都是带有名字的执行块;

b. 都有声明、执行体和异常部分;

2) 触发器与过程/函数的不同点

a. 触发器必须存储在数据库中;

b. 触发器自动执行;

2. 创建触发器

答：1) 语法：

CREATE [OR REPLACE] TRIGGER trigger\_name

{BEFORE|AFTER} triggering\_event ON table\_reference

[FOR EACH ROW [WHEN trigger\_condition]]

trigger\_body;

2) 范例：

CREATE OR REPLACE TRIGGER UpdateMajorStats AFTER INSERT OR DELETE OR UPDATE ON students

DECLARE

CURSOR c\_Statistics IS

SELECT \* FROM students GROUP BY major;

BEGIN

...

END Up;

3. 触发器

答：1) 三个语句(INSERT/UPDATE/DELETE);

2) 二种类型(之前/之后);

3) 二种级别(row-level/statement-level);

所以一共有 3 X 2 X 2 = 12

4. 触发器的限制

答：1) 不应该使用事务控制语句;

2) 不能声明任何LONG或LONG RAW变量;

3) 可以访问的表有限。

5. 触发器的主体可以访问的表

答：1) 不可以读取或修改任何变化表(被DML语句正在修改的表);

2) 不可以读取或修改限制表(带有约束的表)的主键、唯一值、外键列。

(4)Java开发中使用Oracle的ORA-01000

很多朋友在Java开发中，使用Oracle数据库的时候，经常会碰到有ORA-01000: maximum open cursors exceeded.的错误。

实际上，这个错误的原因，主要还是代码问题引起的。

ora-01000: maximum open cursors exceeded.

表示已经达到一个进程打开的最大游标数。

这 样的错误很容易出现在Java代码中的主要原因是：Java代码在执行conn.createStatement()和 conn.prepareStatement()的时候，实际上都是相当与在数据库中打开了一个cursor。尤其是，如果你的 createStatement和prepareStatement是在一个循环里面的话，就会非常容易出现这个问题。因为游标一直在不停的打开，而且没 有关闭。

一般来说，我们在写Java代码的时候，createStatement和prepareStatement都应该要放在循环外面，而且 使用了这些Statment后，及时关闭。最好是在执行了一次executeQuery、executeUpdate等之后，如果不需要使用结果集 （ResultSet）的数据，就马上将Statment关闭。

对于出现ORA-01000错误这种情况，单纯的加大open\_cursors并不是好办法，那只是治标不治本。实际上，代码中的隐患并没有解除。

而且，绝大部分情况下，open\_cursors只需要设置一个比较小的值，就足够使用了，除非有非常特别的要求。

(5)在store procedure中执行 DDL语句

一 是：execute immediate 'update '||table\_chan||' set '||column\_changed||' = '''||v\_trans\_name||''' where empid = '''||v\_empid||'''' ;

二是：The DBMS\_SQL package can be used to execute DDL statements directly from PL/SQL.

这是一个创建一个表的过程的例子。该过程有两个参数：表名和字段及其类型的列表。

CREATE OR REPLACE PROCEDURE ddlproc (tablename varchar2, cols varchar2) AS

cursor1 INTEGER;

BEGIN

cursor1 := dbms\_sql.open\_cursor;

dbms\_sql.parse(cursor1, 'CREATE TABLE ' || tablename || ' ( ' || cols || ' )', dbms\_sql.v7);

dbms\_sql.close\_cursor(cursor1);

end;

/

2 如何找数据库表的主键字段的名称?

SQL>SELECT \* FROM user\_constraints

WHERE CONSTRAINT\_TYPE='P' and table\_name='TABLE\_NAME';

3 如何查询数据库有多少表?

SQL>select \* from all\_tables;

4 使用sql统配符

通 配符 描述 示例 % 包含零个或更多字符的任意字符串。 WHERE title LIKE '%computer%' 将查找处于书名任意位置的包含单词 computer 的所有书名。 \_（下划线） 任何单个字符。 WHERE au\_fname LIKE '\_ean' 将查找以 ean 结尾的所有 4 个字母的名字（Dean、Sean 等）。 [ ] 指定范围 ([a-f]) 或集合 ([abcdef]) 中的任何单个字符。 WHERE au\_lname LIKE '[C-P]arsen' 将查找以arsen 结尾且以介于 C 与 P 之间的任何单个字符开始的作者姓氏，例如，Carsen、Larsen、Karsen 等。 [^] 不属于指定范围 ([a-f]) 或集合 ([abcdef]) 的任何单个字符。 WHERE au\_lname LIKE 'de[^l]%' 将查找以 de 开始且其后的字母不为 l 的所有作者的姓氏。

5使普通用户有查看v$session的权限

GRANT SELECT

ON "SYS"."V\_$OPEN\_CURSOR" TO "SFISM4";

GRANT SELECT

ON "SYS"."V\_$SESSION" TO "SFISM4";

常用函数

distinct

去掉重复的

minus 相减

在第一个表但不在第二个表

SELECT \* FROM FOOTBALL MINUS SELECT \* FROM SOFTBALL;

intersect 相交

INTERSECT 返回两个表中共有的行。

SELECT \* FROM FOOTBAL；

UNION ALL 与UNION 一样对表进行了合并但是它不去掉重复的记录。

汇总函数

count

select count(\*) from test;

SUM

SUM 就如同它的本意一样它返回某一列的所有数值的和。

SELECT SUM(SINGLES) TOTAL\_SINGLES FROM TEST;

SUM 只能处理数字如果它的处理目标不是数字你将会收到如下信息

输入/输出

SQL>SELECT SUM(NAME) FROM TEAMSTATS;

ERROR

ORA-01722 invalid number

no rows selected

该错误信息当然的合理的因为NAME 字段是无法进行汇总的。

AVG

AVG 可以返回某一列的平均值。

SELECT AVG(SO) AVE\_STRIKE\_OUTS FROM TEAMSTATS；

MAX

如果你想知道某一列中的最大值请使用MAX。

SELECT MAX(HITS) FROM TEAMSTATS;

MIN

MIN 与MAX 类似它返回一列中的最小数值。

VARIANCE

VARIANCE 方差不是标准中所定义的但它却是统计领域中的一个至关重要的数值。

SELECT VARIANCE(HITS) FROM TEAMSTATS;

STDDEV

这是最后一个统计函数STDDEV 返回某一列数值的标准差。

SELECT STDDEV HITS FROM TEAMSTATS；

日期时间函数

ADD\_MONTHS

ADD\_MONTHS也可以工作在select 之外

该函数的功能是将给定的日期增加一个月举例来说由于一些特殊的原因上述的计划需要推迟两个月那么就用到了。

LAST\_DAY

LAST\_DAY 可以返回指定月份的最后一天.

MONTHS\_BETWEEN

如果你想知道在给定的两个日期中有多少个月可以使用MONTHS\_BETWEEN。

select task, startdate, enddate ,months between(Startdate,enddate) duration from project;

返回结果有可能是负值.

可以利用负值来判断某一日期是否在另一个日期之前下例将会显示所有在1995 年5 月19 日以前开始的比赛.

SELECT \* FROM PROJECT

WHERE MONTHS\_BETWEEN ('19-MAY-95', STARTDATE)>0;

NEW\_TIME

如果你想把时间调整到你所在的时区你可以使用NEW\_TIME.

SQL>SELECT ENDDATE EDT, NEW\_TIME('ENDDATE',' EDT',' PDT') FROM PROJECT;

NEXT\_DAY

NEXT\_DAY 将返回与指定日期在同一个星期或之后一个星期内的你所要求的星期天

数的确切日期如果你想知道你所指定的日期的星期五是几号可以这样做.

SQL>SELECT STARTDATE, NEXT\_DAY(STARTDATE,' FRIDAY') FROM PROJECT;

SYSDATE

SYSDATE 将返回系统的日期和时间。

SELECT DISTINCT SYSDATE FROM PROJECT；

数学函数

ABS

ABS 函数返回给定数字的绝对值

CEIL 和FLOOR

CEIL 返回与给定参数相等或比给定参数在的最小整数.FLOOR 则正好相反它返回

与给定参数相等或比给定参数小的最大整数.

COS COSH SIN SINH TAN TANH

COS SIN TAN 函数可以返回给定参数的三角函数值默认的参数认定为弧度制.

EXP

EXP 将会返回以给定的参数为指数以e 为底数的幂.

LN and LOG

这是两个对数函数其中LN 返回给定参数的自然对数.

MOD

知道在ANSI 标准中规定取模运算的符号为%在一些解释器中被函数MOD 所取代.

POWER

该函数可以返回某一个数对另一个数的幂在使用幂函数时第一个参数为底数第二个为指数。

SIGN

如果参数的值为负数那么SIGN 返回-1 如果参数的值为正数那么SIGN 返回1,如果参数为零那么SIGN 也返回零.

SQRT

该函数返回参数的平方根,由于负数是不能开平方的所以我们不能将该函数应用于负数.

字符函数

CHR

该函数返回与所给数值参数等当的字符返回的字符取决于数据库所依赖的字符集.

CONCAT

和||一个作用，把两个字符串连接起来.

INITCAP

该函数将参数的第一个字母变为大写此外其它的字母则转换成小写.

LOWER 和UPPER

如你所料LOWER 将参数转换为全部小写字母而UPPER 则把参数全部转换成大写字母.

LPAD 与RPAD

这两个函数最少需要两个参数最多需要三个参数每一个参数是需要处理的字符串

第二个参数是需要将字符串扩充的宽度第三个参数表示加宽部分用什么字符来做填补

第三个参数的默认值为空格但也可以是单个的字符或字符串.

LTRIM 与RTRIM

LTRIM 和RTRIM至少需要一个参数最多允许两个参数第一个参数与LPAD 和RPAD

类似是一个字符串第二个参数也是一个字符或字符串默认则是空格如果第二个

数不是空格的话那么该函数将会像剪除空格那样剪除所指定的字符.

REPLACE

它的工作就如果它的名字所说的那样该函数需要三个参数第一个参数是需要搜索

的字符串(列名），第二个参数是搜索的内容第三个参数则是需要替换成的字符串如果第三个

参数省略或者是NULL 那么将只执行搜索操作而不会替换任何内容.

SUBSTR

这个函数有三个参数允许你将目标字符串的一部份输出，第一个参数为目标字符串，

第二个字符串是将要输出的子串的起点，第三个参数是将要输出的子串的长度。

如果第二个参数为负数那么将会从源串的尾部开始向前定位至负数的绝对值的位置。

TRANSLATE

这一函数有三个参数，目标字符串、源字符串和目的字符串，在目标字符串与源字符

串中均出现的字符将会被替换成对应的目的字符串的字符。

INSTR

如果需要知道在一个字符串中满足特定的内容的位置可以使用INSTR ，它的第一个参

数是目标字符串，第二个参数是匹配的内容，第三和第四个参数是数字用以指定开始搜

索的起点以及指出第几个满足条件的将会被返回。

LENGTH

LENGTH将返回指定字符串的长度。

转换函数

转换函数有三个可以使你方便地将数据从一种类型变换为另一种类型。

TO\_CHAR

该函数的最初功能是将一个数字转换为字符型，不同的解释器可能会使用它来转换其它的

数据类型例如日期型转换为字符型或者是拥有更多的参数。

TO\_NUMBER

该函数与TO\_CHAR 函数相对应显而易见它是将一个字符串型数字转换为数值型。

其他函数

GREATEST 与LEAST

这两个函数将返回几个表达式中最大的和最小的。

SELECT GREATEST（‘ALPHA’，‘ BRAVO’，‘ FOXTROT’，‘ DELTA’） FROM CONVERT；

看来有必要使用FROM 子句可是如果FROM 子句没有的话你将会收到一个错误信息每

一个SELECT 语句都需要FROM 子句。

USER

该函数返回当前使用数据库的用户的名