Neusoft

东软机密

文件编号：D05-PDS073

PLSQL 编码规范

版本：0.0.0-1.2.0

2009-6-30

东软集团股份有限公司 软件开发事业部

（版权所有，翻版必究）

文件修改控制

修改编号	版本	修改条款及内容	修改日期
1	0.0.0-1.1.0	更换 LOGO	2005-7-29
2	0.0.0-1.1.1	修改文档中有错别字，将建改成键	2008-2-20
3	0.0.0-1.1.2	1、修改公司标识为“东软集团股份有限公司”。 2、修改文件密级标识为“东软机密”。	2008-8-15
4	0.0.0-1.2.0	根据公司要求，转换为OpenOffice格式	2009-06-19

目 录

1. 风格约定

1.1 字体

在编辑或浏览代码时，请将编辑器的字体设置为字符等宽字体，这样可以保证格式上的整齐，不会出现有些缩进行无法对齐的现象。下表列出几种常用的字体。

常用的字符等宽字体	常用的字符非等宽字体
Courier	Arial
Fixedsys	MS Sans Serif
MS Gothic（支持日文字符集）	Roman
宋体	System

1.2 Tab 键与缩进

由于各种编辑器对 Tab 键宽度的解释不一样，可能会造成代码的缩进变得参差不齐，所以统一使用半角空格代替 Tab 键。我们建议以 3 个半角空格代替 Tab 键，如果客户有其他要求，遵照客户要求执行。

1.3 每行的最大宽度

每行最大字符数建议不要超过 78 个，因为超过 78 个字符时，编辑器可能会出现水平滚动条，不方便阅读。

1.4 大小写

所有关键字、内建函数、内建包名采用大写。如：BEGIN、LOOP、INSERT、DECODE、UTL_FILE。

所有用户自定义变量、参数采用小写。如：p_i_student_id、l_student_name。

用户自定义的包名、函数或过程名建议采用小写，如客户有其他要求，遵照客户要求执行。

1.5 空格

逗号后面加一空格。如：func(arg1, arg2, arg3)。

操作符前后各加一空格。l_average = (l_demo1 + l_demo2) / 2。

1.6 空行

在 DML 语句，TYPE 声明，CURSOR 声明，IF、LOOP 等逻辑块，程序单元逻辑块，标签等前面空一行。

1.7 对齐

尽量采用“垂直形式”，如：

- 罗列参数时：

```
PROCEDURE maximize_profits (
   advertising_budget     IN     NUMBER,
   bribery_budget         IN OUT NUMBER,
   merge_and_purge_on     IN     DATE := SYSDATE,
   obscene_bonus          OUT    NUMBER
);
```

- 定义变量时：

```
CREATE OR REPLACE PACKAGE genAPI
IS
   c_table       CONSTANT CHAR(5)  := 'TABLE';
   c_column      CONSTANT CHAR(6)  := 'COLUMN';
   c_genpky      CONSTANT CHAR(6)  := 'GENPKY';
   c_genpkyonly CONSTANT CHAR(10) := 'GENPKYONLY';
   c_sequence    CONSTANT CHAR(7)  := 'SEQNAME';
```

- SQL 语句的关键字采用右对齐的方式：

```
   SELECT
     FROM
    WHERE
      AND
       OR
 GROUP BY
   HAVING
      AND
```

```
   INSERT INTO
       VALUES

   INSERT INTO
       SELECT
         FROM
        WHERE
```

```
   DELETE FROM
        WHERE
```

```
   UPDATE
      SET
    WHERE
```

例子：

```
SELECT last_name,
       first_name
  FROM employee
 WHERE department_id = 15 AND hire_date < SYSDATE;

SELECT department_id,
       SUM(salary) AS total_salary
  FROM employee
 GROUP BY department_id
 ORDER BY total_salary DESC;

INSERT INTO employee
            (employee_id, ... )
     VALUES (105 ... );

DELETE FROM employee
      WHERE department_id = 15;

UPDATE employee
   SET hire_date = SYSDATE
 WHERE hire_date IS NULL
   AND termination_date IS NULL;
```

2. 命名约定

2.1 命名的最大宽度

命名在 30 个字符之内。这是 PL/SQL 语言的规定，包括 constant、variable、exception、procedure、function、package、record、PL/SQL table、cursor、reserved word。建议充分使用这个长度，一般命名不要小于 5 个字符，尽量描述清楚(Self-document)，如定义现在时间为 current_date 或 system_date 就比定义成 curdat 或 now 要好得多。

2.2 命名风格

由于 PL/SQL 是不区分大小写的，所以不采用“驼峰式”命名法，而采用“_”连接的方式。如：get_student_name、p_i_student_id。

2.3 前缀

在变量、参数前适当加前缀。

1）范围类型前缀

全局变量加 g_。如：g_for_global_variable

局部变量加 l_。如：l_for_local_variable

常量加 c_。如：c_for_constant_variable

参数加 p_。如：p_for_parameter

2）输入、输出前缀

输入参数加 i_。如：p_i_for_param

输出参数加 o_。如：p_o_for_param

输入/输出参数加 io_。如：p_io_for_param

3）数据类型前缀

数据类型	前缀
BOOLEAN	bol_
CHAR	ch_
DATE	dt_
NUMBER	num_
RAW	raw_
ROWID	rid_
VARCHAR2	vch_

2.4 后缀

一般自定义类型加上类型后缀。

- TABLE 类型加_tbl
- RECORD 类型加_rec
- CURSOR 类型加_cur

例子：

```
TYPE cust_fname_tbl IS TABLE OF customer.cust_fname%TYPE;
TYPE employee_id_rec IS RECORD(employee_id employee.employee_id%TYPE);
CURSOR yearly_analysis_cur IS SELECT ...;
```

3. 其他考虑

3.1 可读性

- 使用注释

只有一个原则：在你编码的时候就加注释，而不是编码完成后加注释。注释应该与被注释的块对齐，如下：

```
-- If the total compensation is more than the maximum...
IF :employee.total_comp > maximum_salary
THEN
   -- Inform the user of the problem.
      MESSAGE  ('Total  compensation  exceeds  maximum.  Please  re-
enter!');

   -- Reset the counter to zero.
   :employee.comp_counter := 0;

   -- Raise the exception to stop trigger processing.
   RAISE FORM_TRIGGER_FAILURE;
END IF;
```

- 使用标签

在嵌套的匿名块和循环前面加标签以达到 Self-document 的效果。这个特别有用，当你有多个嵌套时。示例代码：

```
CREATE OR REPLACE PROCEDURE display_book_usage
IS
BEGIN
   <<best_seller_review>>
   DECLARE
      CURSOR yearly_analysis_cur IS SELECT ...;
      CURSOR monthly_analysis_cur IS SELECT ...;
   BEGIN
      <<yearly_analysis>>
      FOR book_rec IN yearly_analysis_cur (2000)
      LOOP
         <<monthly_analysis>>
         FOR month_rec IN
             monthly_analysis_cur (
                yearly_analysis_cur%rowcount)
         LOOP
            ... lots of month-related code ...
         END LOOP monthly_analysis;
         ... lots of year-related code ...
      END LOOP yearly_analysis;
   END best_seller_review;
END display_book_usage;
```

- 复杂表达式请用()明显地显示其逻辑关系。

3.2 可维护性

尽量使用%TYPE 去定义参数或变量的类型。例子：

```
PROCEDURE format_customer
IS
   first_name    VARCHAR2(30);
   last_name     VARCHAR2(30);
   address       VARCHAR2(60);
BEGIN
   ...
END;
```

```
PROCEDURE format_customer
IS
   first_name    customer.cust_fname%TYPE;
   last_name     customer.cust_lname%TYPE;
   address       customer.cust_addr_l1%TYPE;
BEGIN
   ...
END;
```

4. SQL 优化规则

1.1. 索引的使用原则

1. **尽量避免对索引列进行计算。**
   例：
   X WHERE sal*1.1>950
   O WHERE sal>950/1.1

   X WHERE SUBSTR(name,1,7)=’CAPITAL’
   O WHERE name LIKE ‘CAPITAL%’

2. **尽量注意比较值与索引列数据类型的一致性。**
   例：
   emp_no： NUMBER 型
   O WHERE emp_no=123（好）
   WHERE emp_no=’123’（也可）

   emp_type：CHAR 型
   X WHERE emp_type=123 （此时，查询时，不利用索引列）
   O WHERE emp_type=’123’

3. **尽量避免使用 NULL**
   例：
   X WHERE comm IS NOT NULL
   X WHERE comm IS NULL
   O WHERE comm>=0

4. **尽量避免使用 NOT=（!=）**

例：

X WHERE deptno!=0

O WHERE deptno>0

5. **对于复合索引，SQL 语句必须使用主索引列**

例：复合索引(deptno,job)

O WHERE deptno=20 AND job='MANAGER'

O WHERE deptno=20

O WHERE job='MANAGER' AND deptno=20

X WHERE job='MANAGER'

6. 4.1.6 ORDER BY 子句

O 子句中，列的顺序与索引列的顺序一致。

O 子句中，列应为非空列。

7. 查询列与索引列次序（WHERE）的一致性

O SELECT empno,job FROM emp WHERE empno<100 AND job='MANAGER';

1.2.其他优化的原则

1. 查询的 WHERE 过滤原则，应使过滤记录数最多的条件放在最前面

例：
```
SELECT info
    FROM taba a,tabb b,tabc c
    WHERE a.acol between :alow and :ahigh
    AND   b.bcol between :blow and :bhigh
    AND   c.ccol between :clow and :chigh
    AND   a.key1 = b.key1
    AND   a.key2 = c.key2;
```
其中，A 表的 acol 列可以最多减少查询的记录数目，其次为 B 表的 bcol 列，依次类推。

2. 尽量少用嵌套查询。

3. 使用表的别名

多表连接时，使用表的别名来引用列。

例：

X SELECT abc002,abd003
```
    FROM ab001 ,ab020
    WHERE ab001.col2=ab020.col3
```
.....

O SELECT t1.abc002,t2.abd003
```
    FROM ab001 t1,ab020 t2
    WHERE t1.col2=t2.col3
```

.....

4. 用 NOT EXISTS 代替 NOT IN

例：

X SELECT .....
FROM emp
WHERE dept_no NOT IN (SELECT dept_no
FROM dept
WHERE dept_cat='A');

O SELECT .....
FROM emp e
WHERE NOT EXISTS (SELECT 'X'
FROM dept
WHERE dept_no=e.dept_no
AND dept_cat='A');

5. 用多表连接代替 EXISTS 子句

例：

X SELECT .....
FROM emp
WHERE EXISTS (SELECT 'X'
FROM dept
WHERE dept_no=e.dept_no
AND dept_cat='A');

O SELECT .....
FROM emp e,dept d
WHERE e.dept_no=d.dept_no
AND dept_cat='A';

6. 少用 DISTINCT，用 EXISTS 代替

X SELECT DISTINCT d.dept_code,d.dept_name
FROM dept d ,emp e
WHERE e.dept_code=d.dept_code;

O SELECT dept_code,dept_name
FROM dept d
WHERE EXISTS (SELECT 'X'
FROM emp e
WHERE e.dept_code=d.dept_code);

7. 使用 UNION ALL、MINUS、INTERSECT 提高性能

8. 使用 ROWID 提高检索速度

对 SELECT 得到的单行记录，需进行 DELETE、UPDATE 操作时，使用 ROWID 将会使效率大大提高。

例：SELECT rowid
INTO v_rowid

```
        FROM t1
       WHERE con1
       FOR UPDATE OF col2;
••••••
••••••
   UPDATE t1
      SET col2=.....
    WHERE rowid=v_rowid;
```

9. 查询的 WHERE 过滤原则，应使过滤记录数最多的条件放在最前面。

例：SELECT info
FROM taba a,tabb b,tabc c
WHERE a.acol between :alow and :ahigh
AND b.bcol between :blow and :bhigh
AND c.ccol between :clow and :chigh
AND a.key1 = b.key1
AND a.key2 = c.key2;

其中，A 表的 acol 列可以最多减少查询的记录数目，其次为 B 表的 bcol 列，依次类推。

10. 尽量使用共享的 SQL 语句。

如经常使用 select * from dept where deptno=值

如果每一个‘值’都是常量，则每一次都会重新解释，不能共享内存中的 SQL 语句优化结果。

应把‘值’设置为一个变量，所有的共同语句都可以优化一次，高度共享语句解释优化的结果。

例：SELECT * FROM dept WHERE deptno=:d;

11. 使用优化线索机制进行访问路径控制。

```
SELECT e.ename
  FROM emp e
 WHERE e.job||' ' =' CLERK' ;
```

不如下面的语句好：

```
SELECT /*+FULL(EMP)*/ E.ENAME
  FROM emp e
WHERE e.job='CLERK';
```

12. 使用纯 SQL 来提交效率（Use Pure SQL）

如果可以用 SQL 语句解决问题就不要用 PL/SQL 的形式书写。

考察下面两段代码：

PL/SQL 版本：

```
  DECLARE
     CURSOR checked_out_cur IS
        SELECT pet_id, name, checkout_date
```

```
          FROM occupancy
         WHERE checkout_date IS NOT NULL;
BEGIN
   FOR checked_out_rec IN checked_out_cur
   LOOP
      INSERT INTO occupancy_history (pet_id, name, checkout_date)
           VALUES (checked_out_rec.pet_id, checked_out_rec.name,
                   checked_out_rec.checkout_date);

      DELETE FROM occupancy WHERE pet_id = checked_out_rec.pet_id;
   END LOOP;
END;
```

纯 SQL 版本：

```
INSERT INTO occupancy_history (pet_id, name, checkout_date)
     SELECT pet_id, name, checkout_date
       FROM occupancy
      WHERE checkout_date IS NOT NULL;

DELETE FROM occupancy WHERE checkout_date IS NOT NULL;
```

假如上面代码会处理 30 条记录，那么 PL/SQL 版本将提交 60 条 SQL 语句给 RDBMS，而纯 SQL 版本只提交 2 条 SQL 语句，很大程度上提高了效率。