1. 引言

本规范用于说明在利用 PL/SQL 开发 Oracle 数据库后台应用程序时，应遵守的某些原则。

本规范的制定主要出于以下几方面的考虑：

- ✓ 效率
- ✓ 可读性
- ✓ 可维护性
- ✓ 规范性

规范中未做声明的内容，以满足开发总则为准。

2. 命名规范

2.1 基本命名规则

本规范中采用的命名规则，基本模式为：

前缀_描述

其中前缀根据定制对象，变量类型的不同而变化。

描述的命名规则为：信息结构通则命名规则与字母标识符命名两者相结合

（1）与表中字段相对应的变量命名采取代码方式，以与字段命名相对应

（2）其余用有意义的字母标识符（如：拼音——各个字汉语拼音首字母联结成的字符串，对只有两个字符的，采用名称全拼）

2.2 定制对象命名规则

如表 2-1 所示

前缀	定制对象	举例说明
pkg	程序包	个人帐户 pkg_grzh
prc	过程	生成个人帐户 prc_scgrzh
fun	函数	确定缴费比例 fun_qdjfbl
ind	索引	ind_grzh
tri	触发器	tri_grcbzl
vi	视图	vi_
seq	序列发生器	seq_
pk	主键	
fk	外键	
syn	同义词	
dlk	DB LINK	
snp	快照	

表 2-1 定制对象命名规则

2.3 变量命名规则

变量命名规则中前缀由两部分组成：变量类型前缀+数据类型前缀

变量类型前缀规则如下表 2-2：

前 缀	变量类型
g	全局变量
def	常量

表 2-2 变量类型命名规则

数据类型前缀命名规则如下表 2-3：

前 缀	数据类型
bin	Binary_Integer

b	Boolean
c	Char
d	Date
l	Long
lob	LOB
n	Number
dec	Decimal
dbl	Double
i	Integer
f	Float
r	Real
pls	Pls_Integer
Rec	%ROWTYPE
raw	RAW
row	ROWID
str	Varchar2
cur	CURSOR
rec	Record
tab	Table
refcur	REF CURSOR
u	用户自定义数据类型
typ	TYPE 自定义类型类型
e	异常（EXCEPTION）

表 2-3 数据类型前缀命名规则

例：声明一全局类数值型：gn_grsxh

声明一局部类：str_name

说明：对于未在上述数据类型中列出的其它数据类型（包括系统提供的数据类型），其数据类型前缀的命名必须不与表 3-4 中的前缀重名，它们的前缀命名规则是未规定的。

3．程序书写规范

3.1 大小写风格：

类型	约定	举例
保留字	大写	BEGIN、DECLARE、ELSIF
内置函数	大写	SUBSTR、COUNT、TO_NUMBER
预定义类型	大写	NUMBER(7,2)、BOOLEAN
SQL 关键字	大写	SELECT、INTO、WHERE
数据库对象	小写	abc007、ac021
变量名	小写	gn_dwhrbl

表 3-1 大小写书写规范

3.2 注释风格：

注释总是加在程序的需要一个概括性说明或不易理解或易理解错的地方。注释应语言简炼、易懂而又准确。

3.2.1 源代码文件的注释

（1）在文件的头部必须标明程序名称，它所完成的主要功能。

（2）文件的作者，及完成时间。

（3）文件的状态：测试/未测试。

（4）主要修改活动的修改人、时间、简单原因说明列表、版本号。

（5）维护过程中需要修改程序时，应在被修改语句前面注明修改时间和原因说明。

例：

```
REM 文件名：
REM 功能描述：
REM 状态：
REM 作者：
REM 完成时间：
REM 修改：时间        版本号          修改人          修改原因
```

3.2.2 包、过程、函数的注释

(1) 头部必须进行功能和参数说明；

(2) 主体部分，如算法复杂时，应以注释的方式对其算法结构作出说明；

3.2.3 语句的注释

注释单独成行、放在语句前面。

(1) 应对不易理解的分支条件表达式加注释；

(2) 不易理解的循环，应说明出口条件(有GOTO的程序还应说明入口条件)；

(3) 对重要的计算应说明其功能；

(4) 过长的函数实现，应将其语句按实现的功能分段加以概括性说明；

(5) 供别的文件或函数调用的函数，绝不应使用全局变量交换数据；

(6) 每条SQL语句均应有注释说明（表名、字段名）。

例：

```
SELECT abc007,abc008,abc009,        --上年计入金额，本年计入金额，本年缴费月数
       abc010,abc011,              --年度、本年缴费基数、
       abc012,abc013               --本年帐户支付累计金额、本年统筹支付金额
  INTO ln_abc007,ln_abc008,ln_abc009,
       ln_abc010,ln_abc011,
       ln_abc012,ln_abc013
  FROM ab003                       --个人帐户
 WHERE abc001 = ivc_shbzh          --
   AND abd004 = ...;               --


UPDATE ab003                       --
   SET abc004 = ln_abc004,         --
       abc005 = ln_abc005,         --
       ...
 WHERE abc001 = ivc_shbzh          --
   AND ...                         -

INSERT INTO table_name            --
     ( col1,col2,col3,            --
       col4,col5,...)             --
VALUES
     ( v1,v2,v3,
       v4,v5,...)
```

3.2.4 常量和变量的注释

注释说明放在常量和变量定义语句的后面，注释说明的要点是：

(1)被保存值的含义(必须)

(2)合法取值的范围(可选)

(3)全局量需要对以上逐点做充分的说明。

3.2.5 注释的书写规范

可采用单行/多行注释。(-- 或 /* */ 方式)

3.3 缩进规则:

3.3.1 SQL 语句的缩进风格

(1) 查询列表的书写风格(与注释综合考虑)

一行有多列，超过 80 个字符时，基于列对齐原则，采用下行缩进

```
SELECT col1,col2,....
       colm,coln,...
  INTO v_col1,v_col2,...
       v_colm,...
```

(2) WHERE 子句的书写规范

①每个条件占一行

②嵌套查询条件书写规范

```
WHERE con1
AND con2
AND col3 NOT IN ( SELECT col3
                  FROM t2
                  WHERE ......);
```

(3) SET 子句的书写规范

每个表达式占一行。

```
SET col1 = v1,
    col2 = v2,
    ...
```

3.3.2 控制结构的缩进

程序应以缩进形式展现程序的块结构和控制结构。

下列保留字的下一行缩进三格

BEGIN、THEN、ELSE、ELSIF、LOOP

下列保留字所在行前移三格

END、ELSE、ELSIF、END IF、END LOOP

3.3.3 缩进的限制

（1）每次缩进标准为 3 个空格，不准使用 TAB 键。

（2）任何一个程序最大行宽不得超过80列，第一行续行语句缩进三格，后续续行语句与第一行续行语句对齐。长语句的下一语句以长语句为对齐基准参照上面规定执行。

3.4 其他

（1）后台过程禁止使用 COMMIT。在出错处理中可用 ROLLBACK。

（2）过程的输入/输出参数声明必须指定为'IN'或'OUT'，要与参数命名一致。特别，不能指定为'IN OUT'。

3.5 例子

3.5.1 例一.

```
REM 文件名：jlgrzh.sql
REM 功能描述：该模块主要用于人员新参保时帐户的处理
REM 状态：未测试
REM 作者：丁蓉
REM 完成时间：1999/11/3
REM 修改：1999/11/5，版本号，王建，(所修改信息描述)

CREATE OR REPLACE PACKAGE pkg_jlgrzh AS
  --对于新增，统筹范围外转入人员，新建个人帐户信息
  PROCEDURE pro_xjgrzh (
    ic_shbzh     IN   CHAR,                    --个人社会保障号
    in_dwsxh     IN   NUMBER,                  --单位顺序号
    id_cbrq      IN   DATE,                    --参保日期
    on_fhz       OUT NUMBER                    --返回值
  );

  --对统筹范围外转入的参保人员，如实记载职工个人帐户。
  PROCEDURE pro_sjgrzh (
    ic_shbzh     IN   CHAR,                    --个人社会保障号
    in_snjrje    IN   NUMBER,                  --上年计入金额
    on_fhz       OUT NUMBER                    --返回值
  );

  --功能说明
  FUNCTION fun_函数名 (
```

```
    参数 1 IN 类型,                          --参数说明
    参数 2 IN 类型,)                         --参数说明
    RETURN 类型;

END pkg_jlgrzh;

CREATE OR REPLACE PACKAGE BODY pkg_jlgrzh AS
  --功能说明
  PROCEDURE 过程名 (
    参数 1 IN  类型,                         --参数说明
    参数 2 IN  类型,                         --参数说明
    参数 1 OUT 类型                          --参数说明
  ) IS
     变量名 数据类型;                              --变量说明
  BEGIN
    --语句说明
    语句;
    长语句第一行
       续行 1
       续行 2;
    语句;
    IF 判断条件组合 THEN
    --分支处理说明
       语句;
    ELSIF ... THEN
       语句;
    ELSE
       语句;
    END IF;
    FOR ... LOOP
       循环体;
    END LOOP;
  EXCEPTION
    WHEN ... THEN
       例外处理语句;
  END 过程名;

  --对于新增，统筹范围外转入人员，新建个人帐户信息
  PROCEDURE pro_xjgrzh (
     ic_shbzh     IN   CHAR,                  --个人社会保障号
     in_dwsxh     IN   NUMBER,                --单位顺序号
     id_cbrq      IN   DATE,                  --参保日期
     on_fhz       OUT NUMBER                  --返回值
  ) IS
```

```
    ln_bz      NUMBER;                          --标志
    ln_abc007 NUMBER;                           --上年计入金额
    ln_abc008 NUMBER;                           --本年计入金额
    ln_abc009 NUMBER;                           --本年缴费月数
  BEGIN
    SELECT abc007,abc008,                       --上年计入金额,本年计入金额
           abc009                               --本年缴费月数
      INTO ln_abc007,ln_abc008,
           ln_abc009
      FROM ab003                                --个人帐户
     WHERE abc001 = ic_shbzh;

    --生成个人帐户记录
    INSERT INTO ab004
         ( abc007,abc008,                       --
           abc009,...)                          --
    VALUES
         ( ln_abc007,ln_abc008,                 --
           ln_abc009,...);                      --

    ...
  EXCEPTION
    WHEN OTHERS THEN
       on_fhz := -10110.9;
  END pro_xjgrzh;

  ...
END pkg_jlgrzh;
```

4. SQL 优化规则

4.1 索引的使用原则

4.1.1 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%'

4.1.2 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'

4.1.3 尽量避免使用 NULL

例：

X WHERE comm IS NOT NULL

X WHERE comm IS NULL

O WHERE comm>=0

4.1.4 尽量避免使用 NOT=（!=）

例：

X WHERE deptno!=0

O WHERE deptno>0

4.1.5 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'

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。

4.1.7 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';

4.2 其他

4.2.1 语句书写要规范

尽量避免相同语句由于书写格式的不同，而导致多次语法分析。

4.2.2 尽量少用嵌套查询。

4.2.3 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
    ......
```

4.2.4 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');
```

4.2.5 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';
```

4.2.6 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);
```

4.2.7 使用 UNION ALL、MINUS、INTERSECT 提高性能

4.2.8 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;
```

4.2.9 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。

4.2.10 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;

4.2.11 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';
```

4.2.12 显示光标优于隐式光标

如：

```
update target
set t_field = (select s_information
               from source
               where source.key = target.key)
where exists (select ...
              from source
```

```
            where source.key  = target.key)
```

不如下面的显示光标语句好：

```
declare
   cursor src is
   select * from source;
begin
   for row in src loop
      update target
      set t_field = row.s_information
      where key = row.key;
   end loop;
end;
```