编程规范

1：所有数据库关键字和保留字都大写；字段、变量的大小写

2：程序块采用缩进风格书写，保证代码清晰易读，风格一致，缩进格数统一为2/4个。
必须使用空格，不允许使用【tab】键。

3：当同一条语句暂用多于一行时，每行的其他关键字与第一行的关键字进行右对齐。

4：不允许多个语句写到一行，即一行只写一条语句。

5：避免把复杂的SQL语句写到同一行，建议要在关键字和谓词处换行。

6：相对独立的程序块之间必须加空行。BEGIN、END独立成行。

7:太长的表达式应在低优先级操作符处换行,操作符或关键字应放在新行之首。不同类型的操作符混合使用时，用括号隔离，使得代码清晰。

8: 不同类型的操作符混合使用时，应使用括号明确的表达运算的先后关系。

9：运算符以及比较符左边或者右边只要不是链接的括弧，则空一格。

10：if 后的条件要用括号括起来，括号内每行最多两个条件。

11：减少控制语句的检查次数，如在 else(if..else)控制语句中，对最常用符合条件，尽量往前被检查到。尽量避免使用嵌套的 if 语句，在这种情况应使用多个 if 语句来判断其可能。

命名规范

1：不使用数据库关键字和保留字，为了避免不必要的冲突和麻烦。

2：严禁使用带空格的名称来给字段和表命名，会出错误而终止。

3：用户自定义数据库对象：表，视图，主外键，索引，触发器，函数，存储过程，序列，同义词，数据库连接，包，包体风格要保持一致。

数据库名称 1-8 个字符，其他对象 1-30 个字符，数据库连接不操过 30 个字符。使用英文字母、数字、下划线。

除表外，其他对象命名最好用不同的前缀来区别。

表	tbl_/t_
视图	v_
序列	seq_
簇	c_
触发器	trg_
存储过程	sp_/p_
函数	f_/fn_
物化视图	mv_
包和包体	pkg_
类和类体	typ_
主键	pk_
外键	fk_
唯一索引	uk_
普通索引	idx_
位图索引	bk_

4：PL/SQL 对象和变量命名规则

输入变量	i_
输出变量	o_
输入输出变量	io_
普通变量	v_
全局变量	gv_
常量	大写
游标	cur_
用户自定义类型	type_
保存点	spt_

不允许使用中文和特殊字符

用户对象命名应全部为小写，且不允许使用控制符号强制转换对象为小写字符

变量命名，要有具体含义，能表明变量类型。

5：注释规范

源程序有效注释量必须在 30%左右。

统一文件头的注释，针对存储过程，函数进行功能性描述，入出参数说明。

/**
名称：
功能描述：

修订记录
版本号　编辑时间　编辑人　修改描述

入出参数说明

返回值描述（针对函数）
**/

所有变量定义需要添加注释，说明该变量的用途和含义。
程序分支必须书写注释，这些语句是程序实现某一特定功能的关键。
在程序块的结束行加注释，表明程序块结束。
注释应与描述的代码相似，对代码的注释应在其上方或右方现今为止，不能放在下面。
禁止在注释中使用缩写，特别是非常用的缩写。
注释要与描述的内容进行相同的缩排。
注释上面的代码应空行隔开。
注释用中文书写。
尽量使用”--” 进行注注释。
行尾注释须使用”--” 。

6：分区表命名

分区表的表名可以遵循普通表的正常命名规则。主要用途的缩写+下划线+yymm。

按地域分布的子公司库存表（ 每个区域一个分区），分区名这为表的主要用途的缩写+区域的缩写。

最小分区名字为 before_data 规则 3.4.5:最大分区名字为 after_data 规则 3.4.5:子分区的名字为：父分区名+下划线+sub+ 下划线+no(区域缩写)，根据实际情况进行组合。

分区表本地索引命名在正常索引名的最后一个下划线前加 L。
分区表全局索引命名在正常索引名的最后一个下划线前加 G。

DML 操作规范

1：减少控制语句的检查次数，应将最常用的符合条件前置以便被检查到。

2：避免使用 SELECT *语句，给出字段列表，避免出现在表结构变化时程序无法识别的情况。

3：INSERT 语句必须给出字段列表，避免在表结构变化时发生编译错误。

4：从表中同一笔记录中获取记录的字段值，须使用一 **SQL** 语句得到，不允许分多条 **SQL** 语句。

5：当一个 **PL/SQL** 或 **SQL** 语句中涉及到多个表时，始终使用别名来限定字段名，这使其它人阅读起来更方便，避免了含议模糊的引用，其中能够别名中清晰地判断出表名。

6：禁止进行字段数据类型的隐式转换，所有转换必须进行明确的数据类型转换

说明：隐式转换会导致字段上的索引失效，而进行显式转换，会提醒到开发人员该种操作会导致索引失效

7：禁止在多表关联的时候，在非索引字段上的关联；

8：进行模糊查询时，禁止条件中字符串直接以**“%”**开头；

9：尽量使用 **DECODE** 来简化 **SQL** 访问数据库的次数

10:避免使用 **HAVING** 子句, **HAVING** 只会在检索出所有记录之后才对结果集进行过滤. 这个处理需要排序,总计等操作. 如果能通过 **WHERE** 子句限制记录的数目,那就能减少这方面的开销.

11：当 PL/SQL 或 SQL 语句中涉及多个表时，始终使用别名来限定表名和字段名。

12：确保变量和参数在类型和长度上与表数据列相匹配，否则较宽或较大数据进来时会异常。

13：使用 **EXISTS/NOT EXISTS** 替代 **IN/NOT IN**

14：采用表连接替代 **EXIST**

15：复杂的 SQL 是否由设计不当引起，复杂的 SQL 考虑用程序块来执行。
处理的优先级

静态 SQL>动态 SQL
绑定变量的 SQL>动态 SQL

SQL>PL/SQL 过程
SQL>游标遍历
ORACLE 函数>自定义函数

16：使用 ORACLE 分析函数来代替同一表多次的关联。

17：使用动态 SQL 时要绑定变量。

18：不要把空的变量直接与比较运算符比较，如果结果可能为空，应使用 IS NULL 货 IS NOT NULL 或 NVL 函数进行比较。

19：order by 后面字段不唯一时分页会出现问题，分页时如果 order by 后面的字段不唯一，一定要让 order by 唯一，最佳方案是增加一 pk，如实在没办法则可以追加 rowid，order by 后尽量避免使用 rowid。

20：聚集函数 max、min、sum 在没有记录得符合查询条件的情况下返回 null，不会产生 no_data_found 异常。

21：避免频繁 commit，尤其是把 commit 写在循环体中每次循环都进行 commit。

22：使用绑定变量，避免常量的直接引用。

23：避免不必要的排序

24：对于数字型的唯一键值，用序列 sequence 产生。

25：索引的规则：

建立索引常用的原则如下：

1)、表的主键、外键必须有索引

2)、数据量超过 1000 行的表应该有索引

3)、经常与其它表进行连接的表，在边接字段上应建立索引

4)、经常出现在 where 子句中的字段且过滤性极强的，特别是大表的字段，应该建立索

引

5)、索引字段，尽量避免值为 null

6)、复合索引的建立需要仔细分析； 尽量考虑用单字段索引代替；

A. 正确选择复合索引中的第一个字段，一般是选择性较好的且在 where 子句中常的字段上；

B. 复合索引的几个字段是否经常同时以 and 方式出现在 where 子句中？ 单字段查询是否极少其至没有？ 如果是，则可以建立复合索引； 否则考虑单字段索引；

C. 如果复合索引中包含的字段经常单独出现在 where 子句中，则分解为多个单字段索引；

D. 如果复合索引所包含的字段超过 3 个，那么仔细考虑其必要性，考虑减少复合的字段；

E. 如果既有单字段索引，又有这几个字段上的复合索引，一般可以删除复合索引；

7). 频繁 DDL 的表，不要建立太多的索引；

8). 删除无用的索引，避免对执行计划造成负面影响；让 SQL 语句用上合理的索引。

合理让 SQL 语句使用索引的原则如下：

首先，看是否用上了索引，对于该使用索引而没有用上索引的 SQL 语句，应该想办法用上索引。

其次，看是否用上了索引，特别复杂的 SQL 语句，当其中 where 子句包含多个带有索引的字段时，更应该注意索引的选择是否合理。错误的索引不仅不会带来性能的提高，相反往往导致性能的降低。

26：like 子句尽量前端匹配

数据库设计

1：数据库设计文档中，必须包含表数据保留时间；

2：数据库设计文档中，必须包含表在最大保留时间下的数据量；

3：数据库设计文档中，如果表为分区表，必须包含分区条件；

4：数据库设计文档中，必须包含表的读写频率；

5：单 **SEGMENT** （如单个普通表，分区表的单个分区，单个普通索引，分区索引的单个分区）原则上不得超过 **2GB** 大小；

6：和其他表有关联的表，和其他表功能一致的字段类型以及长度，尽量使用相同的列名；

7：禁止依靠设计数据库表之间的主外键关系来保证数据一致性；

:

8：需要 **UPDATE** 的字段，不得设计为分区条件字段；

9：如无特别需要，原则上，字符类型选择变长字段，数字类型选择 **NUMBER**；

10：如无特别需要，原则上不得设定表的并发度，压缩等属性；

11：无特别说明，每个表的索引，不得超过 **5** 个；单字段上的索引不得超过 **2** 个；（即一个单字段最多可在上面建立一个单字段索引和一个组合索引包含这个字段）；复合索引原则上不得超过 **3** 个字段；

12：原则上，分区表的索引必须是分区索引；

13：频繁出现在 **where** 字句里的字段建议建立索引；

14：用来和其他表关联的字段建议建立索引；

15：索引字段建议有高的选择性和过滤性（**count(distinctid)/count(*)>0.6**）；

16：在 **where** 子句里作为函数参数的字段，不能创建索引，不建议建立函数索引**;**

17：建立索引的时候，建议考虑到 **SELECT** 和 **INSERT,UPDATE,DELETE** 的平衡；

18：一般建议在查询数据量 **10%**以下使用索引；

19：**WHERE** 子句的查询条件构成索引字段前导字段；选择性更高的字段放在组合字段索

引的前导字段；如果字段选择性接近，则把频繁查询的字段放在前面；如果字段查询频率相同，则把表中的数据的排列顺序所依据的字段放在前面；

20：进行 **GROUP BY** 或者是 **ORDER BY** 的字段应在组合字段索引的前导字段；

21：所有序列应设置 **CACHE** 值为不低于 **100**

22：如果要求获得的字段具有强连续和强排序性，则不适宜使用序列

23：视图中不允许出现 **ORDER BY** 排序

24：基于多表关联的视图，必须在字段名前指定表别名；视图的基础数据尽量从表中获取，尽量不要嵌套视图

25：存储过程，必须有异常捕获代码

在存储过程中变量的声明集中在 AS 和 BEGIN 中完成，不允许在代码中随意定义变量。完成相同功能模块的变量放在一起，不同模块一空行隔开。

存储过程中严禁使用 **GOTO** 语句进行跳转

有循环更新的存储过程，必须进行批量提交，且必须进行事务控制

存储过程中如果打开了 **dblink**，则在存储过程正常或者异常退出必须关闭所有打开的 **dblink**

存储过程中如果使用了游标，则在存储过程正常或者异常退出必须关闭所有打开的游标

存储过程中如果有更新，必须在异常捕获代码中做回退操作。

异常处理时，把收集机到的错误信息计入错误日志表。

pl/sql 使用短路径法，当计算逻辑表达式，即：一旦确定后，pl/sql 停止计算表达式。

26：函数中，如果进行了事务处理，必须有异常捕获代码

函数尽量只是实现复杂的计算功能，不对数据库进行更新操作

27：一次 **UPDATE** 多个字段的时候，应一次查询完成

脚本规范

1：脚本按分类或内容分开存放，按下列顺序存储
    1：创建数据库表空间、用户文件脚本。
    2：创建数据库角色、用户脚本。
    3：创建数据类型脚本、自定义的数据类型。
    4：创建业务表脚本。
    5：创建临时表脚本。
    6：创建视图脚本。
    7：创建主外键脚本。
    8：创建索引脚本。
    9：创建触发器脚本。
    10：创建函数、存储过程脚本。
    11：初始化数据脚本。
    12：创建作业脚本。
2：创建每个对象代码的首部应该有对象注释。

3：函数，存储过程应单独创建脚本，在相应目录下，创建一个运行所有脚本的总脚本。

技巧

1：触发器尽量考虑内部代码过程封装，用过程封装 sql，减少解析次数。

```
create or replace procedure p_test_tri(p_deptno in number)
as
    begin
       insert into test_tri_tab2 (deptno,cnt) values (p_deptno,1);
end;
/
create or replace trigger test_tab2_trigger
after insert on test_tri_tab1
for each row
begin
   p_test_tri(:new.deptno);
end test_tri_tab2_trigger;
/
```

2:避免动态 sql，动态 sql 在执行过程中变异，普通 sql 在过程执行前就已经编译过了 。等价静态语句替换动态 sql

3：OLTP 系统尽量使用绑定变量，sql 在 shared_pool 中介西完成逻辑优化，物理优化，生成计划等一系列动作

```
select x from t where x=:x;
```

4:减少对 sysdate，mod 的调用，避免 sql 中的函数调用，大量递归调用影响性能。可以改用表关联来代替函数调用。函数调用有代价。

5：设法减少表扫描次数

6：尽量使用简单 sql 来代替 PL-SQL 逻辑

7：避免不必要的排序

1：确认 order by 是否多余

2：union 是否可以被 union all 替代

3：不可避免排序，要降低开销，降序索引

8：使用 pls_integer 类型

变量时整数型可使用。内部算法改进可提高性能。

9：避免数据类型转换，隐式类型转换

1：在 insert 和 update 语句中，oracle 将赋值的类型转换为目标列的类型。sysdate 根据参数 NLS_DATE_FORMAT 和 NLS_DATE_LANGUAGE 转换为字符

2：SELECT 中，oracle 会将查询到的数据类型自动转换为目标变量的类型。

3：对数值类型的操作，oracle 经常调整其精度 precision 和刻度 scale，允许最大容量。

4：当比较字符和数值的时候，数值有更高的优先级，讲字符转化为数值进行比较。

5：字符类型(可转换成数值)，number 类型与浮点数类型转换，可能会丢失精度，数值和 number 以十进制表示数字，浮点数以二进制表示。

6：讲 clob 转换为字符类型（varchar2），获奖 blob 转换成 raw 类型的时候，

被转换的类型长度长的话，会出错。

7：binary_float 自动转换为 binary_double 是准确的，反之不准确。binary_double>binary_float>number

8：字符串与 date 类型比较，date 具有较高优先级，将字符串转化为 date 类型，需要上下文的支持。

9：当使用 sql 函数或操作符时候，传入类型和实际接收的类型不一致，会将传入的类型根据需要转化为一致。

10：赋值运算=的时候，oracle 会将右边被赋值的类型转化为何左边类型一致的类型。

11：在做连接操作的时候，oracle 会将飞字符类型转化为字符类型，根据上下文转换。

12：在字符和非字符之间的算术和比较运算中，oracle 会将字符转换成日期，rowid、数值类性，算术操作转化为数值，rowid 比较的将字符转化为 rowid，日期比较的转化为日期类型。

13：字符类型将的类型转换，char，varchar，nchar，nvaechar2，nchar 和 nvarchar2 需要国家字符集 utf8 和 al16utf16 的支持，按字符存储的。char，varchar2 手数据库默认字符机支持

类型	到 CHAR	到 VARCHAR2	到 NCHAR	到 NVARCHAR2
CHAR	--	VARCHAR2	NCHAR	NVARCHAR2
VARCHAR2	VARCHAR2	--	NVARCHAR2	NVARCHAR2
NCHAR	NCHAR	NCHAR	--	NVARCHAR2
NVARCHAR2	NVARCHAR2	NVARCHAR2	NVARCHAR2	--

14：sql 字符函数可以接受 clob 类型，substr，instr，对不接受 clob 类型的自动转换为字符类型

15：空格

10：if 的顺序，入参越是频繁调用的值，对应的 if 逻辑越需要靠前，条件靠前可以减少判断的次数。

11：设计开发对列是否为空慎重决定，null 会影响 oracle 的执行计划。索引能回答问题时，非空索引能用上全索引扫描提高性能。

空索引会导致 count(*)记录出错

索引列不可能为空，不要加 is not null 的 check

Not in 查询中，空值会限制 unnest 转换，导致优化器无法选择 anti 算法，走抵消的 filter

Oracle 如果是 not exists 或 exists 和类似 group by 子句连用，cbo 不做查询转换，会慢，改成 not in 或 in

12：不要对列运算

Select * from a where trunk(log_time)=to_date('2013-09-01','yyyy-mm-dd');

分析执行计划

一般，每获取一行开销 5 个以下的逻辑读是接收的范围
Sql>set autotrace traceonly
Sql>sql clause
关注： consistent gets
Rows processed
聚合函数 sum，count 以返回的记录数量为 rows processed

使用锁定

1：执行 DML 前，先执行 SELECT+ FOR UPDATE NOWAIT 来判断自己能否加上锁。
2：在 select+for update nowait 失败后，立即退出不执行后续更新语句，通过自定义异常来让后面的语句不执行，在 for update 加不上锁就退出，ooo4 异常单独捕获。
3：对插入为遇到重复记录就插，重复不插。Dup_val_on_index 异常来实现
4：获取机器 ip，终端号，sid 等信息

```
select userenv('sessionid') from dual;
select userenv('terminal') ) from dual;
select Sys_context('userenv','ip_aaddress') ) from dual;
select Sys_context('userenv','current_user') ) from dual;
select sid from v$mystat where rownum=1;
```