文档名称	密级
Oracle数据库设计开发规范	机密
文档版本	共12页
1.0	

Oracle数据库设计开发规范

拟制		日期	2017-06-30
审核		日期	yyyy-mm-dd
批准		日期	yyyy-mm-dd

北京君德财富投资管理股份有限公司

北京微金客科技有限公司

版权所有 侵权必究

（仅供内部使用）

修订记录

日期	修订版本	CR号	修改章节	修改描述	作者
2017-06-30	1.00			初稿完成	

目录

Oracle数据库设计开发规范

关键词：君德财富，微金客，Oracle，数据库，设计，开发，规范

摘　要：本文档针对北京君德财富投资管理股份有限公司和北京微金客科技有限公司开发人员制定Oracle数据库规范，可作为公司研发部门人员的参考文档。

缩略语清单：

缩略语	英文全名	中文解释
DBA	Database Administrator	数据库管理员，从事管理和维护数据库管理系统的相关工作人员的统称，属于运维工程师的一个分支，主要负责业务数据库从设计、测试到部署交付的全生命周期管理
DCL	Data Control Language	数据控制语言，用来设置或更改数据库用户或角色权限的语句，包括：GRANT、DENY、REVOKE等语句。在默认状态下，只有sysadmin、dbcreator、db_owner或db_securityadmin等人员才有权力执行DCL
DDL	Data Definition Language	数据定义语言，使用如CREATE、DROP、ALTER等语句以改变表属性
DML	Data Manipulation Language	数据操纵语言，使用户能够查询数据库以及操作已有数据库中的数据的计算机语言。包括：INSERT、UPDATE、DELETE
DQL	Data Query Language	数据查询语言，关键字为SELECT
E-R	Entity Relationship Diagram	实体-联系图，提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。
SQL	Structured Query Language	结构化查询语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名
UML	Unified Modeling Language	统一建模语言或标准建模语言，它是一个支持模型化和软件系统开发的图形化语言，为软件开发的所有阶段提供模型化和可视化支持，包括由需求分析到规格，到构造和配置

1 目的

本规范是针对关系型数据库ORACLE的相关特性，拟定的用于指导相关开发过程的规范，其旨在通过该规范的约束和建议，使开发人员可以在他们所编写的代码中保持统一正确的风格，提升数据库和应用的性能，提供代码的可读性以及减少出现错误的几率。

2 建表规则

1. 命名尽量采用富有意义、易于记忆、描述性强、简短及具有唯一性的英文词汇，不要采用汉语拼音。
2. 所有命名均采用英文单数，不准采用复数形式。除非特别需要，或有特殊含义，否则命名中不准出现数字。
3. 命名不能和Oracle的关键字重复。
4. 命名中若使用特殊约定或缩写，要有注释说明。
5. 变量命名禁止取单个字符（如i、j、k...），除非在局部循环体中。
6. 用varchar2代替varchar类型。
7. 如非必须的情况下不要使用blob、clob、nclob大字段，用varchar2(4000)代替。
8. 严禁用系统表空间作为用户默认表空间；严禁在系统表空间上创建用户数据库对象；严禁在SYSTEM/SYS等系统用户下，创建用户数据库对象。
9. 严禁使用带空格的名称来给字段和表命名，会导致错误而终止。
10. 分区表的表名可以遵循普通表的正常命名规则，但是具体分区的命名要能准确表达分区的含义。

    【示例】

    对一个表按时间范围进行分区，则分区名称可定为：MONYYYY。例如2006年1~12月的分区名字分别为：JAN2006、FEB2006、MAR2006、APR2006、MAY2006、JUN2006、JUL2006、AUG2006、SEP2006、OCT2006、NOV2006、DEC2006

    如果每个分区内还有子分区，则子分区的命名可定为：part_yyyymm_sub_no。例如2006年4月里有三个子分区分别为：part_200604_sub_01、part_200604_sub_02、part_200604_sub_03

3 索引规则

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子句中常用的字段上；
   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，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，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。

4 语法的建议与约束

4.1 建议

1. 使用统一的SQL编码格式规范。

   说明：统一的SQL编码格式规范不但可以使阅读者感到清晰明了，而且可以最大程度上避免同一SQL语句在不同地方处理时由于书写格式的不统一，而造成无法共享从而增加SQL解析负担的问题。

   比如，如下的三条SQL，其达到的目的是一样的，但是在ORACLE看来这是三条完全不同的语句，所以要进行三次硬解析。

select * from employees where department_id = 60;

SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 60;

select /*+ PARALLEL */ * from employees where department_id = 60;

对于联机交易型系统来说，SQL的软解析率是非常关注的一个指标。而引起SQL不能共享的主要因素包括：大小写、空格、注释、提示等。一些第三方开发工具，例如TOAD等，都有比较好的格式化功能，大家可以用其做风格统一和美化使用。

2. 在不使用DISTINCT、UNION、ORDER BY、GROUP BY情况下，也能实现业务功能的情况，一定不要使用这些功能。使用这些功能会导致对应的SQL语句排序，增加系统的开销。

【示例】

错误的用法：

```
SELECT COUNT(*)
  FROM (
        SELECT SERIALNO
          FROM T_PUB_COMMONINFO
         WHERE A.PARTID >= '0127'
           AND A.PARTID <= '1227'
         ORDER BY ACCEPTBEGINTIME DESC -- 没有用的 ORDER BY
       )
```

3. 在WHERE条件表达式中，尽可能避免在要使用到索引的字段上使用函数，如果要使用函数建议创建相应的函数索引。

【示例】

错误的用法：

```
SELECT FIELD
  FROM TABLENAME
 WHERE SUBSTRB(FIELD, 1, 4) = '5378'
```

正确的用法：

```
SELECT FIELD
  FROM TABLENAME
 WHERE FIELD LIKE '5378%'
```

错误的用法：

```
WHERE TO_CHAR(zip) = '94002'
```

正确的用法：

```
WHERE zip = TO_NUMBER('94002')
```

4. 当查询条件选择性很低时使用索引反而降低效率，这种情况下，应该用特殊的方法屏蔽该索引，如果字段为数值型的就在表达式的字段名后+0，为字符型的就并上空串。

【示例】

```
SELECT NUM_FIELD
  FROM TABLENAME
 WHERE NUM_FIELD + 0 > 30
SELECT STRING_FIELD
  FROM TABLENAME
 WHERE STRING_FIELD || '' = 'EXAMPLE'
```

5. 如果业务逻辑允许的情况下，尽量用UNION ALL代替UNION，用UNION ALL代替OR。

【示例】

错误的用法：

```
SELECT SERIALNO
  FROM T_WF_DISPOSALSTATUSHIS
 WHERE (ACCEPTPHONE = :B4 OR CALLERNO = :B3 OR USERPHONE1 = :B2)
   AND ACCEPTTIME > SYSDATE - :B1
```

正确的用法：

```
SELECT SERIALNO
  FROM T_WF_DISPOSALSTATUSHIS
 WHERE (ACCEPTPHONE = :B4)
   AND ACCEPTTIME > SYSDATE - :B1
UNION ALL
SELECT SERIALNO
  FROM T_WF_DISPOSALSTATUSHIS
 WHERE (CALLERNO = :B3)
   AND ACCEPTTIME > SYSDATE - :B1
UNION ALL
SELECT SERIALNO
  FROM T_WF_DISPOSALSTATUSHIS A
 WHERE (USERPHONE1 = :B2)
   AND ACCEPTTIME > SYSDATE - :B1
```

6. 如果要对整个表或分区的数据删除，建议使用TRUNCATE替代DELETE。

7. 避免通过DUAL表赋值。

说明：过多的对DUAL表的访问，导致调用该表的等待时间事件比较长。比如取系统时间之类的操作，往一个表插入记录等。

【示例】

错误的用法：

```
SELECT SYSDATE INTO v_Date
  FROM DUAL
```

正确的用法：

```
v_Date := SYSDATE
```

错误的用法：

```
INSERT INTO TABLENAME(FIELD1, FIELD2, FIELD3)
  SELECT '2', SYSDATE, SUSBTR(v_Name, 1, 30)
    FROM DUAL;
```

正确的用法：

```
INSERT INTO TABLENAME(FIELD1, FIELD2, FIELD3)
  VALUES('2', SYSDATE, SUSBTR(v_Name, 1, 30));
```

8. 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。
9. 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。
10. 避免频繁commit，尤其是把commit写在循环体中每次循环都进行commit。
11. 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。

4.2 约束

1. SQL语句的WHERE子句中应尽可能将字段放在等式左边，将计算操作放在等式的右边，除非是要屏蔽该字段的的索引，否则禁止字段参与表达式运算。

   说明：任何对字段的操作都将造成此字段上的索引被屏蔽，导致全表扫描，这里所谓的操作包括数据库函数、计算表达式等等。

   【示例】

   错误的用法：

```
SELECT SOME_FIELD
  FROM TABLENAME
 WHERE NUM_FIELD / 30 < 1000
```

正确的用法：

```
SELECT SOME_FIELD
  FROM TABLENAME
 WHERE NUM_FIELD < 1000 * 30
```

错误的用法：

```
SELECT SOME_FIELD
  FROM TABLENAME
 WHERE TO_CHAR(LOGDATE, 'YYYYMMDD') = '19991201'
```

正确的用法：

```
SELECT SOME_FIELD
  FROM TABLENAME
 WHERE LOGDATE >= TO_DATE('19991201', 'YYYYMMDD')
   AND LOGDATE < TO_DATE('19991202', 'YYYYMMDD')
```

2. SQL语句的WHERE子句中每个条件的操作符两边类型应相同，禁止潜在的数据类型转换。

说明：潜在的字段数据类型转换将造成索引被屏蔽，导致全表扫描。例如将字符型数据与数值型数据比较，ORACLE会自动将字符类型字段用TO_NUMBER函数进行转换。

【示例】

错误的用法（表TABLENAME中的列STRING_FIELD是字符型VARCHAR，则以下语句存在类型转换）：

```
SELECT SOMEFIELD
  FROM TABLENAME
 WHERE STRING_FIELD > 10
```

正确的用法：

```
SELECT SOMEFIELD
  FROM TABLENAME
 WHERE STRING_FIELD > '10'
```

3. SQL语句的WHERE子句中避免使用IN操作，严禁使用NOT IN操作。

说明：在SQL语句中，能用表连接尽量使用表连接，不能使用表连接则使用EXISTS，严禁使用IN。

【示例】

错误的用法：

```
SELECT SOME_FIELD
  FROM TABLE1
 WHERE FIELD1 IN (
                  SELECT FIELD2
                    FROM TABLE2
                 )
```

正确的用法：

```
SELECT T1.SOME_FIELD
  FROM TABLE1 T1, TABLE2 T2
 WHERE T1.FIELD1 = T2.FIELD2
```

错误的用法：

```
SELECT SOME_FIELD
  FROM TABLE1
 WHERE FIELD1 NOT IN (
                       SELECT FIELD2
                         FROM TABLE2
                      )
```

正确的用法：

```
SELECT SOME_FIELD
  FROM TABLE1 T1
 WHERE NOT EXISTS (
                    SELECT 1
                      FROM TABLE2 T2
                     WHERE T2.FIELD2 = T1.FIELD1
                   )
```

4. 禁止对VARCHAR(2000)之类的大字段值进行ORDER BY、DISTINCT、GROUP BY、UNION之类的操作。

   说明：此类操作将消耗大量的CPU和内存资源。

5. 禁止利用SQL语句做一些业务逻辑的判断或操作。

   【示例】

   错误的用法：

```
SELECT STAFFNO, STAFFNAME
  FROM T_PUB_STAFF
 WHERE (i_StaffNo IS NULL OR STAFFNO = i_StaffNo)
   AND (i_StaffName IS NULL OR STAFFNAME LIKE '%' || i_StaffName || '%')
```

   错误分析：上面的SQL语句中，利用SQL引擎对变量的值进行判断，导致在使用过程中，对该表进行全表扫描。

   正确的用法：通过代码中对变量的值进行判断然后决定执行对应的SQL语句。

5 工作中发生过的其它问题

使用PL/SQL Developer的DEBUG功能执行pl/sql代码时,需要确保捕获了所有的异常，否则会导致数据库异常。

6 工具推荐

推荐使用以下几种数据库脚本开发工具。

6.1 UltraEdit-32

全称：UltraEdit-32 Professional Text/HEX Editor

公司：IDM Computer Solutions, Inc.

网址：http://www.ultraedit.com

简介：UltraEdit是目前比较流行的文本编辑器，它可以将文本中的字符串按自己所期望的字体及格式显示出来，对显示代码中关键字、字符串、系统函数等尤为有用，并且可以在编辑过程中自动按照自己所定义关键字列表（WORDFILE.TXT文件）校正大小写。

6.2 PL/SQL Developer

全称：PL/SQL Developer

公司：Allround Automations

网址：http://www.allroundautomations.nl

简介：PL/SQL Developer是能支持断点单步调试ORACLE脚本的工具，通过它可以方便的查看修改ORACLE的数据库对象及其属性、执行各种SQL语句，调试测试存储过程函数等，是开发ORACLE脚本和现场解决ORACLE脚本问题的好工具。

6.3 TOAD

全称：TOOLS FOR ORACLE APPLICATION DEVELOPER

公司：Quest Software

网址：http://www.quest.com

简介：TOAD能支持断点单步调试ORACLE脚本，同时还有较强的DBA管理功能。

6.4 PowerDesigner

全称：PowerDesigner

公司：Sybase

网址：http://www.sybase.com

简介：PowerDesigner是一个独具特色的建模工具集，它融合了以下几种标准建模技术：使用UML的应用程序建模、业务流程建模和传统数据库建模。我们主要利用该工具完成数据库建模，PowerDesigner能以E-R图的方式体现各实体之间的关系，尤其适用于项目设计阶段的表结构设计。