深入了解 Oracle 数据字典

——深入了解 Oracle 数据字典,提高自学习 Oracle 能力

本文作者: eygle (eygle.com@gmail.com)

摘要:

我们知道 Oracle 通过数据字典来管理和展现数据库信息，这些信息至关重要。正确理解这部分内容有助于加强我们的 oracle 学习能力。

本文对 Oracle 数据字典的关系进行探讨。

接下来我们介绍一下怎样通过数据库本身来学习研究数据库。

首先，Oracle 的字典表和视图基本上可以分为三个层次。

1.1 X$表

这一部分表是 Oracle 数据库的运行基础，在数据库启动时由 Oracle 应用程序动态创建。

这部分表对数据库来说至关重要，所以 Oracle 不允许 SYSDBA 之外的用户直接访问，显示授权不被允许。

如果显示授权你会收到如下错误:

```
SQL> grant select on x$ksppi to eygle;
grant select on x$ksppi to eygle
                 *
ERROR at line 1:
ORA-02030: can only select from fixed tables/views
```

Oracle 的解释是:

```
ORA-02030 can only select from fixed tables/views
Cause: An attempt is being made to perform an operation other than a retrieval from a fixed table/view.
Action: You may only select rows from fixed tables/views.
```

一句话，这些对象你只能查询。

1.2 GV$和 V$视图

在创建了 X$表之后，Oracle 创建了 GV$和 V$视图。 从 Oracle8 开始，GV$视图开始被引入，其含义为 Global V$。

除了一些特例以外，每个 V$视图都有一个对应的 GV$视图存在。

GV$视图的产生是为了满足 OPS 环境的需要，在 OPS 环境中，查询 GV$视图返回所有实例信息，而每个 V$视图是基于 GV$视图，增加了 INST_ID 列的 WHERE 条件限制建立，只包含当前连接实例信息。

注意，每个 V$视图都包含类似语句：

```
where inst_id = USERENV('Instance')
```

用于限制返回当前实例信息。

我们从 GV$FIXED_TABLE 和 V$FIXED_TABLE 开始，看一下 GV$视图和 V$视图的结构：

```
SQL> select view_definition from v$fixed_view_definition where view_name='V$FIXED_TABLE';

VIEW_DEFINITION
--------------------------------------------------------------------------------
select NAME , OBJECT_ID , TYPE , TABLE_NUM from GV$FIXED_TABLE where inst_id = USERENV('Instance')
```

这里我们看到 V$FIXED_TABLE 基于 GV$FIXED_TABLE 创建。

```
SQL> select view_definition from v$fixed_view_definition where view_name='GV$FIXED_TABLE';

VIEW_DEFINITION
--------------------------------------------------------------------------------
select inst_id,kqftanam, kqftaobj, 'TABLE', indx from x$kqfta
union all
select inst_id,kqfvinam, kqfviobj, 'VIEW', 65537 from x$kqfvi
union all
select inst_id,kqfdtnam, kqfdtobj, 'TABLE', 65537 from x$kqfdt
```

这样我们找到了 GV$FIXED_TABLE 视图的创建语句，该视图基于 X$表创建。

我们知道，GV$视图和 V$视图是在数据库创建过程中建立起来的，内置于数据库中，Oracle 通过 v$fixed_view_definition 视图为我们展现这些定义。

1.3 GV_$,V_$视图和 V$,GV$同义词

在 GV$和 V$之后，Oracle 建立了 GV_$和 V_$视图，随后为这些视图建立了公用同义词。这些工作都是通过 catalog.sql 脚本实现的。

我们从 catalog.sql 脚本中摘录一段:

```
create or replace view v_$fixed_table as select * from v$fixed_table;
create or replace public synonym v$fixed_table for v_$fixed_table;
```

```
create or replace view gv_$fixed_table as select * from gv$fixed_table;
create or replace public synonym gv$fixed_table for gv_$fixed_table;
```

从以上脚本中，我们注意到，第一个视图 V_$和 GV_$视图基于 V$和 GV$视图首先被创建，然后基于 V_$和 GV_$视图的同义词被创建。

通过 V_$视图，Oracle 把 V$视图和普通用户隔离，V_$视图的权限可以授予其他用户，而 Oracle 不允许任何对于 V$视图的直接授权，我们看以下例子：

```
[oracle@jumper udump]$ sqlplus '/ as sysdba'

SQL*Plus: Release 9.2.0.4.0 - Production on Mon Jun 13 16:41:41 2005

Copyright (c) 1982, 2002, Oracle Corporation.    All rights reserved.



Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production

SQL> grant select on v$sga to eygle;
grant select on v$sga to eygle
                        *
ERROR at line 1:
ORA-02030: can only select from fixed tables/views


SQL> grant select on v_$sga to eygle;

Grant succeeded.
```

对于内部 X$表及 V$视图的限制，我猜测是通过软件代码实现的，而并非通过数据库权限控制。

所以，实际上通常我们大部分用户访问的 V$对象，并不是视图，而是指向 V_$视图的同义词；而 V_$视图是基于真正的 V$视图(这个视图是基于 X$表建立的)。

在进行数据访问时，Oracle 访问 VIEW 优先，然后是同义词。我们通过以下实验来验证一下这个结论。
首先参考 Oracle 处理机制，创建 X$EYGLE,V$EYGLE,V_$EYGLE 和公用同义词 V$EYGLE：

```
[oracle@jumper udump]$ sqlplus eygle/eygle

SQL*Plus: Release 9.2.0.4.0 - Production on Mon Jun 13 17:37:25 2005
```

```
Copyright (c) 1982, 2002, Oracle Corporation.    All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production

SQL> create table x$eygle as select username from dba_users;

Table created.

SQL> create view v$eygle as select * from x$eygle;

View created.

SQL> create view v_$eygle as select * from v$eygle;

View created.

SQL> create public synonym v$eygle for v_$eygle;

Synonym created.
```

然后我们在 sys 用户下创建 V$EYGLE 视图：

```
SQL> connect / as sysdba
Connected.

SQL> create view v$eygle as select username,user_id from dba_users;

View created.
```

此时查询，我们得到的 SYS 的 V$EYGLE 信息：

```
SQL> desc v$eygle;
 Name                                      Null?    Type
 ----------------------------------------------------------------
 USERNAME                                  NOT NULL VARCHAR2(30)
 USER_ID                                   NOT NULL NUMBER
```

当我们删除这个视图以后，再次访问时，Oracle 选择访问了 V$EYGLE 同义词：

```
SQL> drop view v$eygle ;

View dropped.

SQL> desc v$eygle
 Name                                      Null?    Type
 ----------------------------------------------------------------
 USERNAME                                  NOT NULL VARCHAR2(30)

SQL>
```

v$fixed_view_definition 视图是我们研究 Oracle 对象关系的一个入口，仔细理解 Oracle 的数据字典机制，有助于深入了解和学习 Oracle 数据库知识。

1.4 再进一步

1.4.1 数据库的初始化

首先我们考察 bootstrap$表，该表中记录了数据库启动的基本及驱动信息。

```
SQL> col line# for 99
SQL> col obj# for 99
SQL> select * from bootstrap$ order by line#;

LI OB SQL_TEXT
-- -- --------------------------------------------------------------------------------
-1 -1 8.0.0.0.0
 0  0 CREATE ROLLBACK SEGMENT SYSTEM STORAGE (    INITIAL 112K NEXT 1024K MINEXTENTS 1 M
 2  2 CREATE CLUSTER C_OBJ#("OBJ#" NUMBER) PCTFREE 5 PCTUSED 40 INITRANS 2 MAXTRANS 25
 3  3 CREATE INDEX I_OBJ# ON CLUSTER C_OBJ# PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE
 4  4 CREATE TABLE TAB$("OBJ#" NUMBER NOT NULL,"DATAOBJ#" NUMBER,"TS#" NUMBER NOT NULL
 5  5 CREATE TABLE CLU$("OBJ#" NUMBER NOT NULL,"DATAOBJ#" NUMBER,"TS#" NUMBER NOT NULL
 6  6 CREATE CLUSTER C_TS#("TS#" NUMBER) PCTFREE 10 PCTUSED 40 INITRANS 2 MAXTRANS 255
 7  7 CREATE INDEX I_TS# ON CLUSTER C_TS# PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE (
 8  8 CREATE CLUSTER C_FILE#_BLOCK#("TS#" NUMBER,"SEGFILE#" NUMBER,"SEGBLOCK#" NUMBER)
 9  9 CREATE INDEX I_FILE#_BLOCK# ON CLUSTER C_FILE#_BLOCK# PCTFREE 10 INITRANS 2 MAXT
10 10 CREATE CLUSTER C_USER#("USER#" NUMBER) PCTFREE 10 PCTUSED 40 INITRANS 2 MAXTRANS
11 11 CREATE INDEX I_USER# ON CLUSTER C_USER# PCTFREE 10 INITRANS 2 MAXTRANS 255 STORA
```

```
12 12 CREATE TABLE FET$("TS#" NUMBER NOT NULL,"FILE#" NUMBER NOT NULL,"BLOCK#" NUMBER
13 13 CREATE TABLE UET$("SEGFILE#" NUMBER NOT NULL,"SEGBLOCK#" NUMBER NOT NULL,"EXT#"
14 14 CREATE TABLE SEG$("FILE#" NUMBER NOT NULL,"BLOCK#" NUMBER NOT NULL,"TYPE#" NUMBE
15 15 CREATE TABLE UNDO$("US#" NUMBER NOT NULL,"NAME" VARCHAR2(30) NOT NULL,"USER#" NU
16 16 CREATE TABLE TS$("TS#" NUMBER NOT NULL,"NAME" VARCHAR2(30) NOT NULL,"OWNER#" NUM
17 17 CREATE TABLE FILE$("FILE#" NUMBER NOT NULL,"STATUS$" NUMBER NOT NULL,"BLOCKS" NU
18 18 CREATE TABLE OBJ$("OBJ#" NUMBER NOT NULL,"DATAOBJ#" NUMBER,"OWNER#" NUMBER NOT N
19 19 CREATE TABLE IND$("OBJ#" NUMBER NOT NULL,"DATAOBJ#" NUMBER,"TS#" NUMBER NOT NULL
....
```

这部分信息，在数据库启动时最先被加载，跟踪数据库的启动过程，我们发现数据库启动的第一个动作就是:

```
create table bootstrap$ ( line#              number not null,     obj#
   number not null,     sql_text     varchar2(4000) not null)     storage (initial
   50K objno 56 extents (file 1 block 377))
```

这部分代码是写在 Oracle 应用程序中的，在内存中创建了 bootstrap$以后，Oracle 就可以从 file 1,block 377 上读取其他信息，创建重要的数据库对象。从而根据这一部分信息启动数据库，这就实现了数据库的引导，类似于操作系统的初始化。

这部分你可以参考biti_rainy在Itpub上的文章（http://www.itpub.net/199099.html）。

Oracle 的 X$表信息可以从 v$fixed_table 中查到：

```
SQL> select count(*) from v$fixed_table where name like 'X$%';
   COUNT(*)
----------
          394
```

对于 Oracle9iR2，共有 394 个 X$对象被记录。

1.4.2 GV$和 V$视图

X$表建立以后，基于 X$表的 GV$和 V$视图得以创建。

这部分视图我们也可以通过查询 V$FIXED_TABLE 得到。

```
SQL> select count(*) from v$fixed_table where name like 'GV$%';
```

```
  COUNT(*)
----------
       259
```

这一部分共 259 个对象。

```
SQL> select count(*) from v$fixed_table where name like 'V$%';
  COUNT(*)
----------
       259
```

同样是 259 个对象。

v$fixed_table 共记录了:
394 + 259 + 259 共 912 个对象。

```
SQL> select count(*) from v$fixed_table;

  COUNT(*)
----------
       912
```

以上是 Oracle9iR2 单机环境中的数据：

```
SQL> select * from v$version;

BANNER
----------------------------------------------------------------
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
PL/SQL Release 9.2.0.4.0 - Production
CORE    9.2.0.3.0       Production
TNS for Linux: Version 9.2.0.4.0 - Production
NLSRTL Version 9.2.0.4.0 - Production
```

1.5 最后的验证

最后然我们通过 V$PARAMETER 视图来追踪一下数据库的架构：

1.5.1 V$PARAMETER 的结构

```
SQL> select view_definition from v$fixed_view_definition a where a.VIEW_NAME='V$PARAMETER';
```

```
VIEW_DEFINITION
--------------------------------------------------------------------------------
select    NUM , NAME , TYPE , VALUE , ISDEFAULT , ISSES_MODIFIABLE , ISSYS_MODIFIA
BLE , ISMODIFIED , ISADJUSTED , DESCRIPTION, UPDATE_COMMENT from GV$PARAMETER wh
ere inst_id = USERENV('Instance')
```

我们看到 V$PARAMETER 是由 GV$PARAMETER 创建的，GV$PARAMETER 则是由 X$创建的。

```
SQL> select view_definition from v$fixed_view_definition a where a.VIEW_NAME='GV$PARAMETER';
VIEW_DEFINITION
--------------------------------------------------------------------------------
select x.inst_id,x.indx+1,ksppinm,ksppity,ksppstvl,ksppstdf,    decode(bitand(kspp
iflg/256,1),1,'TRUE','FALSE'),    decode(bitand(ksppiflg/65536,3),1,'IMMEDIATE',2,
'DEFERRED',                                          3,'IMMEDIATE','FALSE'),    decode(bit
and(ksppstvf,7),1,'MODIFIED',4,'SYSTEM_MOD','FALSE'),    decode(bitand(ksppstvf,2)
,2,'TRUE','FALSE'),    ksppdesc, ksppstcmnt    from x$ksppi x, x$ksppcv y where (x.i
ndx = y.indx) and    ((translate(ksppinm,'_','#') not like '#%') or (ksppstdf = 'F
ALSE'))
```

说明：在这里我们看到 GV$PARAMETER 来源于 x$ksppi,x$ksppcv 两个 X$表。 x$ksppi,x$ksppcv 基本上包含所有数据库参数，v$parameter 展现的是不包含"_"开头的参数。以"_"开头的参数我们通常称为隐含参数，一般不建议修改，但很多因为功能强大经常使用而广为人知。

1.5.2 视图还是同义词

在非 SYS 用户下查询，很多朋友曾经提出过疑问，那就是，当我访问 V$PARAMETER 对象时，访问的是视图还是同义词?

如果你还记得我们前面讲过的内容，那么你会知道，毫无疑问，这里访问的是同义词，因为除了 SYS 用户以外，其他用户不能查询 V$视图，V$视图也不能被授权给其他用户。

那么这个问题实际上是不成立的。

```
SQL> connect / as sysdba
Connected.
SQL> grant select on v$parameter to eygle;
grant select on v$parameter to eygle
                *
ERROR at line 1:
ORA-02030: can only select from fixed tables/views

SQL> connect eygle/eygle
Connected.
```

```
SQL> desc sys.v$parameter
ERROR:
ORA-04043: object sys.v$parameter does not exist

SQL> desc v$parameter
 Name                                      Null?    Type
 ----------------------------------------------------------------
 NUM                                                NUMBER
 NAME                                               VARCHAR2(64)
 TYPE                                               NUMBER
 VALUE                                              VARCHAR2(512)
 ISDEFAULT                                          VARCHAR2(9)
 ISSES_MODIFIABLE                                   VARCHAR2(5)
 ISSYS_MODIFIABLE                                   VARCHAR2(9)
 ISMODIFIED                                         VARCHAR2(10)
 ISADJUSTED                                         VARCHAR2(5)
 DESCRIPTION                                        VARCHAR2(64)
 UPDATE_COMMENT                                     VARCHAR2(255)
```

1.5.3 Oracle 如何通过同义词定位对象

如果愿意的话，我们可以进一步来进行追溯，使用 10046 事件，我们可以看到更多的东西。

通过 10046 事件跟踪查询：

```
[oracle@jumper udump]$ sqlplus eygle/eygle

SQL*Plus: Release 9.2.0.4.0 - Production on Mon Jun 13 18:29:22 2005

Copyright (c) 1982, 2002, Oracle Corporation.    All rights reserved.



Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production

SQL> alter session set events '10046 trace name context forever,level 12';

Session altered.

SQL> select count(*) from v$parameter;
```

```
  COUNT(*)
----------
       262

SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production
```

10046 事件的使用请参考：
http://www.eygle.com/case/Use.sql_trace.to.Diagnose.database.htm

Ok，在这里我们不要使用 tkprof 格式化，因为 tkprof 可能会隐去重要信息（本文仅摘取几段重要跟踪信息，你完全可以通过实验获得相同的输出）：

第一段重要代码是：

```
PARSING IN CURSOR #2 len=198 dep=1 uid=0 oct=3 lid=0 tim=1092440257023120 hv=2703824309 ad='567681f0'
select obj#,type#,ctime,mtime,stime,status,dataobj#,flags,oid$, spare1, spare2 from obj$ where owner#=:1 and name=:2 and namespace=:3 and remoteowner is null and linkname is null and subname is null
END OF STMT
PARSE #2:c=0,e=1601,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=0,tim=1092440257023088
BINDS #2:
 bind 0: dty=2 mxl=22(22) mal=00 scl=00 pre=00 oacflg=08 oacfl2=1 size=24 offset=0
    bfp=b701cf24 bln=22 avl=02 flg=05
    value=25
 bind 1: dty=1 mxl=32(11) mal=00 scl=00 pre=00 oacflg=18 oacfl2=1 size=32 offset=0
    bfp=b701c7b4 bln=32 avl=11 flg=05
    value="V$PARAMETER"
 bind 2: dty=2 mxl=22(22) mal=00 scl=00 pre=00 oacflg=08 oacfl2=1 size=24 offset=0
    bfp=b701c790 bln=24 avl=02 flg=05
    value=1
```

Oracle 根据三个传入参数 owner#=25,name=V$PARAMETER,namespace=1，来判断对象类型，按照表、视图优先规则来定位判断，对于本例这个查询是不会有结果的。

接下来 Oracle 继续判断，那么此时需要验证同一词了：

```
PARSING IN CURSOR #4 len=46 dep=1 uid=0 oct=3 lid=0 tim=1092440257028409 hv=3378994511 ad='576eb040'
select node,owner,name from syn$ where obj#=:1
END OF STMT
PARSE #4:c=0,e=1278,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=0,tim=1092440257028379
```

```
BINDS #4:
 bind 0: dty=2 mxl=22(22) mal=00 scl=00 pre=00 oacflg=08 oacfl2=1 size=24 offset=0
    bfp=b701b3cc bln=22 avl=03 flg=05
    value=841
```

传入绑定变量值是 841，我们看看 841 是什么：

```
SQL> select object_name,object_id,object_type from dba_objects where object_id=841;

OBJECT_NAME                    OBJECT_ID OBJECT_TYPE
------------------------------ ---------- ------------------
V$PARAMETER                           841 SYNONYM
```

841 正是这个同义词，我们再继续看这个递归 SQL 的作用：

```
SQL> select node,owner,name from syn$ where obj#=841;

NODE        OWNER                          NAME
-------- ------------------------------ ------------------------------
            SYS                            V_$PARAMETER
```

原来这个 SQL 获得的是同义词的底层对象，这里得到了 V_$PARAMETER。

我们继续向下看：

```
PARSING IN CURSOR #8 len=37 dep=1 uid=0 oct=3 lid=0 tim=1092440257074273 hv=3468666020 ad='576db210'
select text from view$ where rowid=:1
END OF STMT
PARSE #8:c=0,e=1214,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=0,tim=1092440257074242
BINDS #8:
 bind 0: dty=11 mxl=16(16) mal=00 scl=00 pre=00 oacflg=18 oacfl2=1 size=16 offset=0
    bfp=b7018770 bln=16 avl=16 flg=05
    value=000001CD.0013.0001
EXEC #8:c=0,e=972,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=1092440257075602
```

注意这里，Oracle 执行查询访问 view$视图，获得视图定义文本，我们看一下这里访问的是什么对象，绑定变量传入的 rowid 值为 000001CD.0013.0001，注意这是个受限 rowid,查询时需要转换一下处理：

```
SQL> select obj# from view$ where dbms_rowid.rowid_to_restricted(rowid,0) = '000001CD.0013.0001';

      OBJ#
----------
```

```
        840


    SQL> select object_name,object_type from dba_objects where object_id=840;

    OBJECT_NAME                    OBJECT_TYPE
    ------------------------------ -----------------
    V_$PARAMETER                   VIEW
```

这里 Oracle 访问的正是 V_$PARAMETER 视图的定义方式。执行查询可以得到：

```
    select text from view$ where obj#=840;

    TEXT
    --------------------------------------------------------------------------------
    select
"NUM","NAME","TYPE","VALUE","ISDEFAULT","ISSES_MODIFIABLE","ISSYS_MODIFIABLE","ISMODIFIED","ISADJUSTE
D","DESCRIPTION","UPDATE_COMMENT" from v$parameter
```

至此就完成了查询中的回溯及定位，当然，实际过程中 Oracle 后台的递归操作比这还要复杂的多，感兴趣的朋友可以按照文中的方法测试研究一下，文中不再赘述。

参考文献：

使用 SQL_TRACE 进行数据库诊断
http://www.eygle.com/case/Use.sql_trace.to.Diagnose.database.htm
Oracle 数据库创建脚本 sql.bsq 文件
关于数据库 open 的深入探究
http://www.itpub.net/199099.html

作者简介：

盖国强，网名 eygle

Itpub Oracle 管理版版主，Itpub 论坛超级版主，曾任 ITPUB MS 版版主。
CSDN eMag Oracle 电子杂志主编。

曾任职于某国家大型企业，服务于烟草行业,开发过基于 Oracle 数据库的大型 ERP 系统，属国家信息产业部重点工程。同时负责 Oracle 数据库管理及优化，并为多家烟草企业提供 Oracle 数据库管理、优化及技术支持。

目前任职于北京某电信增值业务系统提供商企业，首席 DBA，负责数据库业务。管理全国 30 多个数据库系统。项目经验丰富，曾设计规划及支持中国联通增值业务等大型数据库系统。

实践经验丰富，长于数据库诊断、性能调整与 SQL 优化等。对于 Oracle 内部技术具有深入研究。

高级培训讲师，培训经验丰富，曾主讲 itpub dba 培训及 itpub 高级性能调整等主要课程。
《Oracle 数据库 DBA 专题技术精粹》、《Oracle 数据库性能优化》两书的主编及主要作者。

你可以在http://www.eygle.com上找到关于作者的更多信息。