Oracle Lob类型存储浅析

在Oracle中，为数据表字段column和PL/SQL语言，分别提供了多种数据类型，以应对实际开发中的多种类型。Lob类型是Oracle推出一种保存大对象的数据类型。当我们考虑将信息文件（十进制、二进制）、图像甚至音频信息采用数据库作为保存载体时，就需要使用lob类型数据。

目前Oracle支持的Lob类型具体包括四个子类型（subtype），分别为CLOB、BLOB、NLOB和BFILE。其中，CLOB、BLOB和NLOB都是将数据保存在数据库内部，而BFILE类型保存的核心是文件指针，真正的文件是保存在数据库外。

与传统的数据类型相比较，lob类型数据无论在管理上还是空间使用上，都有很多特殊之处。本篇主要介绍lob类型一些基本的存储特性。

**1、**  **环境准备和数据段segment特性**

我们选择在Oracle 10gR2下进行试验。

SQL> select \* from v$version;

BANNER

----------------------------------------------------------------

Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Prod

PL/SQL Release 10.2.0.1.0 - Production

CORE     10.2.0.1.0       Production

TNS for 32-bit Windows: Version 10.2.0.1.0 - Production

NLSRTL Version 10.2.0.1.0 – Production

使用create table命令，可以构建出一个实验数据表T。

SQL> create table t (id number, cl clob);

Table created

对于一般的数据表而言，一个数据表只会对应一个存储数据段data segment对象。这里的特殊情况是分区表，通常一个分区就对应一个单独的存储对象。当数据表中包括lob类型的数据列时，也会有独特的段对象建立。

SQL> select segment\_name, segment\_type, tablespace\_name from user\_segments;

SEGMENT\_NAME                   SEGMENT\_TY TABLESPACE\_NAME

------------------------------ ---------- ------------------------------

SYS\_LOB0000056069C00002$$      LOBSEGMENT USERS

T                  TABLE      USERS

SYS\_IL0000056069C00002$$       LOBINDEX    USERS

（篇幅原因，无关对象省略。。。。。。）

9 rows selected

我们发现，除了常规的数据段T之外，另外增加了两个明显是系统命名的段对象，类型分别为lobsegment和lobindex。

对Oracle lob类型数据表而言，一个带lob列的数据表创建是要对应多个数据段创建的。除了传统的数据表创建的数据段Table Data Segment之外，一个lob列都会生成两个专门的段：lob段和lob索引段。

Lob段（LobSegment）对应的是存放在数据表lob列上的数据。在Oracle的lob类型数据列，有两种保存位置结构。一个是in-row storage，也就是每一行的lob数据同其他列的数据以行的形式一起保存在数据块中。这种情况的lob列取值较小。而另一种为out-of-row storage，当lob对象较大，不能保存在一个数据块中时，可以将其放置在一个独立lobsegment中进行保存。而out-of-row storage时数据行中lob列上保存的只是一个指向lobsegment对应位置的指针引用。

Lob索引段（LobIndex）是Oracle为每一个lob类型列强制生成的索引，主要作用是用于进行lob类型数据检索加速的操作。Lobindex与lob列共生，如果强制进行删除操作，是会报错的。

SQL> drop index SYS\_IL0000056069C00002$$;

drop index SYS\_IL0000056069C00002$$

ORA-22864: 无法 ALTER 或 DROP LOB 索引

**2、lob类型数据表原始定义分析**

使用dbms\_metadata包，我们可以获取到数据表的全部定义，包括各种缺省参数和细节信息。

CREATE TABLE "SCOTT"."T"

   ( "ID" NUMBER,

       "CL" CLOB

   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING

  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT)

  TABLESPACE "USERS"

 LOB ("CL") STORE AS (

  TABLESPACE "USERS" ENABLE STORAGE IN ROW CHUNK 8192 PCTVERSION 10

  NOCACHE LOGGING

  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT)) ;

  CREATE UNIQUE INDEX "SCOTT"."SYS\_IL0000056069C00002$$" ON "SCOTT"."T" (

  PCTFREE 10 INITRANS 2 MAXTRANS 255

  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT)

  TABLESPACE "USERS"

  PARALLEL (DEGREE 0 INSTANCES 0) ;

可以看到我们抽取出的元数据包括两个组成部分，数据表创建部分和lob索引创建部分。在数据表创建部分，我们可以看到将lob作为一个独立段对象进行存储设置和参数设置。

在lob索引创建部分，我们可以看到虽然是对应索引创建语句，但是从索引名称上显然是系统自动生成的对象名称。

此外，还有很多是针对lob特殊的参数，如cache、enable storage in-row等，这些参数在数据表lob的行为和访问性能上有巨大的影响。由于篇幅和内容所限，我们在本篇中不加以累述。

**3、lob段与lob索引存储转移**

对于一个数据表涉及的多个段，很多时候我们需要将其进行移动处理。其中最常用的方法是使用move进行对象表空间的移动。

SQL> select segment\_name, segment\_type, tablespace\_name from user\_segments;

SEGMENT\_NAME                   SEGMENT\_TY TABLESPACE\_NAME

------------------------------ ---------- ------------------------------

SYS\_LOB0000056099C00002$$      LOBSEGMENT USERS

T                  TABLE      USERS

SYS\_IL0000056099C00002$$       LOBINDEX    USERS

9 rows selected

SQL> alter table t move tablespace system;

Table altered

SQL> select segment\_name, segment\_type, tablespace\_name from user\_segments;

SEGMENT\_NAME                   SEGMENT\_TY TABLESPACE\_NAME

------------------------------ ---------- ------------------------------

T                  TABLE      SYSTEM

SYS\_LOB0000056099C00002$$      LOBSEGMENT USERS

SYS\_IL0000056099C00002$$       LOBINDEX   USERS

9 rows selected

上面实验可以明确看到，当使用一般的move命令时，只会将数据表T段进行移动到新表空间。Lob段和对应的lobindex段没有变化。如果需要移动lob/lobindex，需要额外的单独操作。

SQL> alter index SYS\_IL0000056069C00002$$ rebuild tablespace users;

alter index SYS\_IL0000056069C00002$$ rebuild tablespace users

ORA-02327: 无法以数据类型 LOB 的表达式创建索引

Lobindex是不能使用rebuild直接重构的。

SQL> select segment\_name, segment\_type, tablespace\_name from user\_segments;

SEGMENT\_NAME                   SEGMENT\_TY TABLESPACE\_NAME

------------------------------ ---------- ------------------------------

T\_LOGSEGMENT              LOBSEGMENT USERS

SYS\_IL0000056069C00002$$       LOBINDEX   USERS

T                   TABLE      EXAMPLE

9 rows selected

SQL> alter table t move lob(cl) store as t\_logsegment (tablespace example);

Table altered

SQL> select segment\_name, segment\_type, tablespace\_name from user\_segments;

SEGMENT\_NAME                   SEGMENT\_TY TABLESPACE\_NAME

------------------------------ ---------- ------------------------------

T                  TABLE      EXAMPLE

SYS\_IL0000056069C00002$$       LOBINDEX   EXAMPLE

T\_LOGSEGMENT              LOBSEGMENT EXAMPLE

9 rows selected

使用alter table xxx move lob(xx) store as xxx (tablespace xxx);命令，我们可以进行lob列的存储位置调节。

在创建数据表的时候，同样可以使用lob(xxx)对应的数据表空间字句，执行存储lob对象的空间信息。

SQL> Create table DemoLob ( A number, B clob )

  2         LOB(b)

  3       STORE AS lobsegname (

  4         TABLESPACE users

  5         --STORAGE (lobsegment storage clause)

  6         INDEX lobindexname (

  7              TABLESPACE example

  8              --STORAGE ( lobindex storage clause )

  9         )

 10       )

 11        TABLESPACE system

 12       --STORAGE( tables storage clause )

 13       ;

Table created

SQL> select table\_name, column\_name, segment\_name, tablespace\_name, index\_name from user\_lobs;

TABLE\_NAME COLUMN\_NAM SEGMENT\_NAME      TABLESPACE\_NAME   INDEX\_NAME

---------- ---------- ------------------------------ --------------------------

T    CL    T\_LOGSEGMENT EXAMPLE    SYS\_IL0000056069C00002$$

在实际物理设计部署过程中，经常有将大对象分区和存储单独部署表空间的情况。可以根据实际的情况，将一些很大的lob列连同索引保存在单独的表空间上。

但是注意，一般数据表而言，lob段和lobindex段是在一个表空间上。即使在SQL语法上存在支持，但是将lob段和lobindex分开存储的语句通常被忽略掉。

SQL> alter table t move lob(cl) store as T\_LOGSEGMENT (tablespace example index t\_logindex (tablespace users));

Table altered

SQL> select table\_name, column\_name, segment\_name, tablespace\_name, index\_name from user\_lobs;

TABLE\_NAME COLUMN\_NAM SEGMENT\_NAME    TABLESPACE\_NAME      INDEX\_NAME

---------- ---------- ------------------------------ --------------------------

T    CL   T\_LOGSEGMENT     EXAMPLE    SYS\_IL0000056069C00002$$

SQL> select segment\_name, segment\_type, tablespace\_name from user\_segments;

SEGMENT\_NAME                   SEGMENT\_TY TABLESPACE\_NAME

------------------------------ ---------- ------------------------------

T         TABLE      EXAMPLE

SYS\_IL0000056069C00002$$       LOBINDEX   EXAMPLE

T\_LOGSEGMENT            LOBSEGMENT EXAMPLE

9 rows selected

**4、结论**

Lob类型是一种我们经常使用的复杂数据类型。处理和管理lob类型的方法和我们常规的手段存在很大差异，无论是开发还是运维过程中都要特别注意。

 >