Characteristics of Read-Writeable External Tables

Although they are called ‘writeable’ or ‘read-writeable’ external tables, the first thing to be clear about the new flavor of 12c external tables is that you cannot:

- No Inserts, updates or deletes
- No indexes
- CTAS created within a database to store outside of DB_Cache
- Can transfer to another OS without endiness issues

.

External tables allow you to export the result set of joining three tables and applying two functions and a grouping aggregation. As you generate a read-writeable external table, however, you can apply transformations to the data (anything you can write in a `SELECT` statement will modify and define the contents of the eventual external table –so, joins with other tables, for example, are easy, as would be applying some function to the data as it is extracted).

Creating a Writeable External Table

Use the sidpers account which doesn’t need privileges granted directly (He’s a DBA), Below are the relevant grants for the sake of clarity.

```
SQL> connect / as sysdba
Connected.

SQL> create directory ext_dir as 'c:\etl';
Directory created.

SQL> grant read, write on directory ext_dir to scott;
Grant succeeded.
```

The use of database directory objects here is a nice security touch: it means that control over who can read from, or write to, the location specified in the `CREATE DIRECTORY` command can be exercised from within the database (by granting and withholding database object privileges) as well as the more obvious operating system restrictions that may (or may not) be in force for the physical directory referenced by the database object. Now connect as scott/tiger in sql*plus.

```
SQL> connect scott/tiger
Connected.

SQL> create table ext_emp (
   2 empno, ename, sal, mgr, deptno, loc, dname)
   3 organization external
   4 (type oracle_datapump
   5 default directory ext_dir
   6 location ('external_emp.etl'))
   7 as
   8 select e.empno, initcap(e.ename), e.sal*1.1,
   9 e.mgr, d.deptno, d.loc, d.dname
   10 from emp e, dept d
   11 where e.deptno=d.deptno;

Table created.
```

None of this syntax is particularly different from the way we created external tables in earlier versions, except for the inclusion of lines 7 onwards (the ‘Select’ bit of syntax that makes this a form of CTAS) –and even those lines are not exactly conceptually challenging. The only real subtlety here, and the real new feature, of course, is the use of the **`TYPE ORACLE_DATAPUMP`** line, instead of the **`TYPE ORACLE_LOADER`** as you would have used in earlier versions (and as you still can, of course, use in 10g-12c if you only need to read ‘ordinary’ external tables).

Additionally the extracted table will have its **`ENAME`** data all starting with initial capital letters, and the **`SAL`** column will have its sums inflated by 10%. This simply proves that applying functions (Oracle’s own in-built ones, or your own: it makes no difference which) to data at the point that it is extracted from the database is simply a question of being able to write a decent **`SELECT`** statement in SQL.

What gets created…

When the database responds that the table has been created, what it actually means is: ‘I have just written a file to your hard disk containing (in this case) the data extracted from two joined tables and transformed as you specified’. You can check the result of that disk writing activity, of course, by dropping to the operating system:

```
C:\etl>dir

Volume in drive C is W2K
Volume Serial Number is A0E3-CE0E

Directory of C:\etl
08/02/2005 12:12p <DIR> .
08/02/2005 12:12p <DIR> ..
08/02/2005 12:12p 12,288 EXTERNAL_EMP.ETL
08/02/2005 12:12p 45 EXT_EMP_704_504.log
```

You’ll notice that a log file is created by the table creation process: that will be populated subsequently whenever any operations are performed on the external table (for example, if someone selects from it). It’s rather like the SQL Loader log file you get when performing ordinary SQL Loads, or the one you got when using external tables in 9i. As I’ll show you later, you can actually take control over where this log file gets created and what its name will be, but the point now is simply that you’ll get an automatically-created one in any case whether you like it or not.

But the really significant file here is, of course, that EXTERNAL_EMP.ETL one. It is that which contains the joined, modified and extracted data: if I ship that to another database, it can be read simply by creating a new external table with a definition that (roughly) matches mine. I’ll show you how easy that all is in just a moment.

Earlier you noticed that the extraction file uses a proprietary Oracle binary format. So don't think of opening it in notepad or vi and actually saving it from within those programs, because you'll render it unusable. But feel free to open it in those sorts of text editors simply to have a look at what's inside it:

```
<ROWSET>
 <ROW>
  <STRMTABLE_T>
   <VERS_MAJOR>1</VERS_MAJOR>
   <VERS_MINOR>0 </VERS_MINOR>
   <VERS_DPAPI>3</VERS_DPAPI>
   <ENDIANNESS>0</ENDIANNESS>
   <CHARSET>WE8MSWIN1252</CHARSET>
   <NCHARSET>AL16UTF16</NCHARSET>
   <DBTIMEZONE>+11:00</DBTIMEZONE>
   <OWNER_NAME>SCOTT</OWNER_NAME>
   <NAME>EXT_EMP</NAME>
   <COL_LIST>
     <COL_LIST_ITEM>
     <COL_NUM>1</COL_NUM>
     <NAME>EMPNO</NAME>
     <TYPE_NUM>2</TYPE_NUM>
     <LENGTH>22</LENGTH>
     <PRECISION_NUM>4</PRECISION_NUM>
     <SCALE>0</SCALE>
     <CHARSETID>0</CHARSETID>
     <CHARSETFORM>0</CHARSETFORM>
     <CHARLENGTH>0</CHARLENGTH>
     </COL_LIST_ITEM>
     <COL_LIST_ITEM>
```

It's interesting to note that some of the XML tags you see here describe the character set and time zone of the source database, plus the endianness of the data itself. This makes the file extremely portable: take it over to a Unix database, for example, and the different byte ordering of the data (Little Endian versus Big Endian) can be sorted, because the extraction file itself tells us what byte ordering was used to generate it. Take it to a database in another part of the planet, and all TIMESTAMP WITH TIMEZONE data that may be encased within the extraction file can be appropriately re-processed, because we know where it was originally created.

3.2 Verifying Creation

Of course, having created the table, you will want to check that it works as intended:

```
EMPNO  ENAME     SAL MGR   DEPTNO LOC      DNAME
------ ------ ------ ----- ------ -------- --------------
7782   Clark    2695 7839      10 NEW YORK ACCOUNTING
7839   King     5500           10 NEW YORK ACCOUNTING
7934   Miller   1430 7782      10 NEW YORK ACCOUNTING
7369   Smith     880 7902      20 DALLAS   RESEARCH
7876   Adams    1210 7788      20 DALLAS   RESEARCH
7902   Ford     3300 7566      20 DALLAS   RESEARCH
7788   Scott    3300 7566      20 DALLAS   RESEARCH
7566   Jones  3272.5 7839      20 DALLAS   RESEARCH
7499   Allen    1760 7698      30 CHICAGO  SALES
7698   Blake    3135 7839      30 CHICAGO  SALES
7654   Martin   1375 7698      30 CHICAGO  SALES
7900   James    1045 7698      30 CHICAGO  SALES
7844   Turner   1650 7698      30 CHICAGO  SALES
7521   Ward     1375 7698      30 CHICAGO  SALES
```

And as you can see, just like the external tables in earlier versions, you query an external table just exactly as if it were a normal table

- Can be joined to oracle tables or other external tables
- Reduces result sets stored in the database buffer cache

. Only when you're absolutely ready would you bring the data inside the database with a perfectly standard CTAS statement.

Illegal moves...

Whilst selecting from your new external table is one thing, don’t try any of these sorts of things:

```
SQL> insert into ext_emp (empno, ename)
  2 values (7385,'ROGERS');
insert into ext_emp (empno, ename)
*
ERROR at line 1:
ORA-30657: operation not supported on external organized table

SQL> create index extidx1 on ext_emp(empno);
create index extidx1 on ext_emp(empno)
*
ERROR at line 1:
ORA-30657: operation not supported on external organized table

SQL> delete from ext_emp;
delete from ext_emp
*
ERROR at line 1:
ORA-30657: operation not supported on external organized table
```

This new feature is called ‘writeable external tables’ in many books and in Oracle’s own documentation, but that’s clearly not really very true! Perhaps they should better be thought of as ‘wrote-able’: the database wrote them out once, and that’s the end of their writeable-ness. Thereafter, they’re just as read-only as external tables ever were in earlier versions.

Transporting Data

If that's all there was to external tables, they probably wouldn't rate much of a mention by anyone.

- simple flat file, housed on a regular file system just like any Word document, or spreadsheet file
- Can port to other Operating Systems (Windows to Linux)

```
SQL> create directory receive_data as
  2  '/home/oracle/ext_data';

Directory created.SQL> create table ext_emp (
  2  empno number(4),
  3  ename varchar2(15),
  4  sal number(8,2),
  5  mgr number(4),
  6  deptno number(4),
  7  loc varchar2(20),
  8  dname varchar2(20))
  9  organization external
 10  (type oracle_datapump
 11  default directory receive_data
 12  location ('EXTERNAL_EMP.ETL'));

Table created.

 EMPNO ENAME      SAL   MGR DEPTNO LOC      DNAME
------ ------ ------ ----- ------ -------- --------------
  7782 Clark    2695  7839     10 NEW YORK ACCOUNTING
  7839 King     5500           10 NEW YORK ACCOUNTING
  7934 Miller   1430  7782     10 NEW YORK ACCOUNTING
  7369 Smith     880  7902     20 DALLAS   RESEARCH
  7876 Adams    1210  7788     20 DALLAS   RESEARCH
  7902 Ford     3300  7566     20 DALLAS   RESEARCH
  7788 Scott    3300  7566     20 DALLAS   RESEARCH
  7566 Jones  3272.5  7839     20 DALLAS   RESEARCH
  7499 Allen    1760  7698     30 CHICAGO  SALES
  7698 Blake    3135  7839     30 CHICAGO  SALES
  7654 Martin   1375  7698     30 CHICAGO  SALES
  7900 James    1045  7698     30 CHICAGO  SALES
  7844 Turner   1650  7698     30 CHICAGO  SALES
  7521 Ward     1375  7698     30 CHICAGO  SALES
```

4.1 Earlier versus 12c Differences

In 9i, you can create external tables and you can even update them via PL/SQL, C, C++, VB Net or just directly open it in Notepad and add your information. It's less secure than 12c which is encrypted but it also has additional flexibility in that you can insert, update or delete via applications. Within the Oracle database and front ends though, you can only SELECT the data. However, it is like a snapshot in time and can be very usefull in that manner.

Actions As SYS	

Data Dictionary Objects Related To External Tables	`user_external_tables`
	`all_external_tables`
	`dba_external_tables`

System Privileges Related To External Tables	`create table`
	`create any table`
	`drop any table`

Actions As SYS	
Connect	`/ AS SYSDBA`
Create Directory	Create the directory external – mkdir external `CREATE OR REPLACE DIRECTORY ext AS 'c:\external'; -- '/oracle/external';`

Grant Directory Access To An End User	`GRANT READ ON DIRECTORY ext TO scott;`
	`GRANT WRITE ON DIRECTORY ext TO scott;`

External Table	

Create Text File Using a Text Editor	`7369,SMITH,CLERK,20`
	`7499,ALLEN,SALESMAN,30`
	`7521,WARD,SALESMAN,30`
	`7566,JONES,MANAGER,20`
	`7654,MARTIN,SALESMAN,30`
	
	`Save external file as c:\external\demo1.dat`
	`(if UNIX or LINUX use /oracle/external)`

	1111,MORGAN,DIRECTOR,10 2222,CLINE,MANAGER,30 3333,HAVEMEYER,VP MKTG,10 4444,LOFSTROM,MANAGER,10 5555,ALLEN,SECURITY,30 Save external file as c:\external\demo2.dat (if UNIX or LINUX use /oracle/external)

As End User

Create Internal Representation of the External Table

```
conn scott/tiger

CREATE TABLE ext_tab (
empno CHAR(4),
ename CHAR(20),
job CHAR(20),
deptno CHAR(2))
ORGANIZATION EXTERNAL
(TYPE oracle_loader
DEFAULT DIRECTORY ext
ACCESS PARAMETERS
(FIELDS TERMINATED BY ','
MISSING FIELD VALUES ARE NULL
(empno, ename, job, deptno))
LOCATION ('demo1.dat'))
PARALLEL
REJECT LIMIT 0;

SELECT * FROM EXT_TAB; -- SEE DATA

DROP TABLE ext_tab PURGE;
```

```
CREATE TABLE ext_tab (
empno CHAR(4),
ename CHAR(20),
job CHAR(20),
deptno CHAR(2))
ORGANIZATION EXTERNAL
(TYPE oracle_loader
DEFAULT DIRECTORY ext
ACCESS PARAMETERS
(FIELDS TERMINATED BY ','
MISSING FIELD VALUES ARE NULL
(empno, ename, job, deptno))
LOCATION ('demo1.dat','demo2.dat'))
PARALLEL
REJECT LIMIT 0;

SELECT * FROM EXT_TAB;
-- See data from both files
```

External Table For Writing and Reading

```
CREATE TABLE ext_write (
tab_name, tblspname, numblocks)
ORGANIZATION EXTERNAL
(TYPE oracle_datapump
DEFAULT DIRECTORY ext
LOCATION ('table_history.exp'))
PARALLEL
AS
SELECT table_name, tablespace_name,
blocks
FROM user_tables;

SELECT *
FROM ext_write;

SELECT *
FROM ext_write
WHERE numblocks > 100;

-- open ext_write_####_####.log files
-- open c:\external\table_history.exp

DROP TABLE ext_write;
```

Tab Delimited External Table	CREATE TABLE ext_tab (empno CHAR(4), ename CHAR(20), job CHAR(20), deptno CHAR(2)) ORGANIZATION EXTERNAL (TYPE oracle_loader DEFAULT DIRECTORY ext ACCESS PARAMETERS (RECORDS DELIMITED BY NEWLINE FIELDS TERMINATED BY X'09' MISSING FIELD VALUES ARE NULL (empno, ename, job, deptno)) LOCATION ('demo1.dat')) PARALLEL REJECT LIMIT 0;

External Table For Viewing Alert Logs	CREATE OR REPLACE DIRECTORY bdump AS
	'c:\oracle\product\admin\orabase\bdump\';
	
	CREATE TABLE alert_log (text
	VARCHAR2(400))
	ORGANIZATION EXTERNAL (
	TYPE oracle_loader
	DEFAULT DIRECTORY bdump
	ACCESS PARAMETERS (
	  RECORDS DELIMITED BY NEWLINE
	  NOBADFILE NODISCARDFILE NOLOGFILE)
	  location('alert_orabase.log'))
	REJECT LIMIT unlimited;
	
	SELECT * FROM system.alert_log;

Query External Table	SELECT *
	FROM ext_tab;

NOTE:	External tables are READ ONLY. Insert, update, and delete can not be performed.