Tech Info Library

Shared Access to Rdb Database under DAL

Revised: 7/21/92
Security: Everyone

Shared Access to Rdb Database under DAL

==========

Article Created: 15 January 1991
Article Last Reviewed: 29 June 1992
Article Last Updated:

TOPIC ----------

I am having a problem with shared access to an Rdb database under DAL. Here are commands from the VMS $ prompt and their results:

```
$ icl1
Network Innovations Interactive CL/1 Demo
Execution begins...
 CL/1 version 1.11
Execution Completed Successfully

begin>  OPEN RDB DBMS ;
more>   USE LOCATION "CL1$SYSTEM" ;
more>   OPEN RDB DATABASE "CL1DEMO" FOR SHARED READONLY ;
more>   GO
begin>  SELECT * FROM OFFICES; PRINTALL ;
more>   GO
.
.
.
data from above database displayed
.
.
.
```

Meanwhile, from another process (once the above user has attached to the database and started a transaction), the following command:

```
$ RMU/DUMP/USERS CL1$SYSTEM:CL1DEMO
```

shows that the above DAL commands produced a read/write transaction on the RDB

database. From all the tests I have done, no read-only transaction is possible on an Rdb database using DAL.

DISCUSSION ----------

The symptom you described is one of the restrictions and limitations with DAL and Rdb. Basically, DAL provides protection using a model based on databases and tables, while Rdb uses a model based on protection. We checked with Network Innovations, and they have provided the DAL 1.2 Technical Notes, in which item #10 describes the restrictions and limitations with DAL and Rdb. The following is the text for that note.

DAL Technical Note # 010
Data Access Language (DAL) and Rdb

This tech note discusses restrictions and limitations with DAL and Rdb.

The Rdb DBMS adapter for the DAL Server is best viewed as a VAX/VMS C application which uses Rdb's Dynamic SQL interface to access Rdb databases. Accordingly, any DAL user who intends to access Rdb databases should ensure that his or her account is set up with the proper VMS privileges and quotas required by Rdb. DAL users should also note that any current restrictions or problems in Digital's Rdb and Dynamic SQL will also affect the DAL Rdb DBMS adapter.

For example, when Rdb performs arithmetic operations on floating point data types such as the SQL aggregate function AVG(), Rdb rounds any decimal result to a whole number. Consequently, DAL client queries containing aggregate functions on decimal or float datatypes will return rounded numbers when executed in Rdb.

Listed below are some specific Rdb related problems and their workarounds or explanations:

- Problem #1: Referring to multiple databases by dbaliasname (AuthID)

To support multiple Rdb databases, the DAL Server Rdb Database Adapter uses the DECLARE SCHEMA statement in Rdb's Dynamic SQL for each database opened. In Rdb, each database or schema is referenced by an "authorization identifier" or AuthID. DAL uses the dbaliasname specified in an OPEN DATABASE statement as the AuthID. (Or when a dbaliasname is not specified, the database name is used as the alias and AuthID by default). For example:

	dbaliasname	AuthID
- open database "sample";	sample	sample
- open database "reservations" alias "test";	test	test

Normally, the DAL Server keeps track of the AuthID for you. However, if you mix normal DAL statements with Rdb statements (via the DAL EXECUTE IMMEDIATE statement), you may need to use the AuthID when referring to tables, views, and indexes or when using the Rdb SET TRANSACTION statement. When referring to

tables by the dbaliasname in DAL, the correct syntax is:

```
   dbaliasname!table_name         sample!offices
```

Rdb uses a period "." instead of an exclamation point "!" when referencing a table by the schema's AuthID:

```
   AuthID.table_name            sample.offices
```

(Note: DAL uses the period "." to prepend an owner name to a table. For example, sample!mark.offices in DAL indicates the table offices owned by mark in the database whose dbaliasname is sample.)

So, when using DAL's EXECUTE IMMEDIATE statement to set a transaction mode for a database or to reference a table, be sure to use the database's dbaliasname as the AuthID. For example:

```
open rdb dbms;
open rdb database "daldemo" alias  test  in location "msad$system";
execute immediate "create table small (text char(20))";
commit;
insert into test!small values ("xyz");
execute immediate "update small set text = 'small fails' where text = 'xyz' ";
rollback;
insert into test!small values ("xyz");
execute immediate "update test.small set text = 'test.small works' where text =
'xyz' ";
commit;
```

- Problem #2: Table Locking with DAL and Rdb

DAL provides a model for specifying the type of "protection" or "locking" desired when using a database or specific table. Both the OPEN DATABASE and OPEN TABLE statements allow for update modes (READONLY/UPDATE) and sharing modes (SHARED/PROTECTED/EXCLUSIVE). The default mode is UPDATE and PROTECTED. Typically this would mean that the user has a "READ/WRITE" lock on the particular table or database. The sharing and update modes are advisory and are passed on to the DBMS only WHEN APPROPRIATE.

Rdb provides a some what different model for specifying protections that does not map well with the DAL model. Rdb uses an extension to SQL transactions — the SET TRANSACTION statement — to signal the type of locking mechanism desired for that transaction. SET TRANSACTION modes are not persistent across subsequent transactions; they must be specified at the beginning of each new transaction or the default mode of "READ WRITE" and "SHARED" will be used.

DAL update and shared modes are not transaction based; that is, in the DAL model, update and shared modes are "global" session modes and are independent of beginning and committing sessions. Consequently, when a DAL user opens a Rdb database or table, specifying a particular update or shared mode, the Rdb transaction modes are NOT changed. The DAL Server does not alter the Rdb transaction modes because they CANNOT be changed in the middle of a Rdb transaction. One would have to commit the current transaction and begin a new

transaction, setting the transaction modes as desired.

There are two possible problems:

1) PROBLEM: Lock Conflicts caused inserts and updates while using the Rdb default transaction modes.

SOLUTION: Commit as early and often as possible. Generic DAL applications should commit often to free any locks against tables and indexes accessed. Otherwise, other users might be locked out of one or more tables.

2) PROBLEM: User wants to specify READ ONLY to prevent locking.

SOLUTION: Applications tailored to Rdb can explicitly begin new transactions with the desired locking and update modes by using the Rdb SET TRANSACTION statement via the DAL EXECUTE IMMEDIATE statement. Below are examples:

```
open rdb dbms;
open rdb database "daldemo";
execute immediate "SET TRANSACTION READ ONLY;" in Rdb;
...queries....
commit;      /* ends this transaction with READ ONLY */
insert into ....
commit;
```

- Problem #3: Attempting to query directly against the Rdb System Relations occasionally fails. For example, the following query fails:

```
SELECT RDB$RELATION_NAME FROM RDB$RELATIONS;
```

Explanation: When DAL is asked to do a query, it checks the syntax and content of the query statement to ensure that the columns and tables asked for exist in the database and to determine the data types of the results. Certain system relations contain columns with a "SEGMENTED STRING" data type (seg_str). This data type is not supported in SQL interface for Rdb 3.xxx (but will be supported in Rdb 4.0). Even though the column queried is not of the type seg_str, DAL looks at the entire structure (column names, data types, and lengths) to check for correct syntax and content. And in doing so, Rdb generates the following error code when it encounters a column with a seg_str: SQL-F-INVSSCONV, Invalid conversion for segmented string column RDB$VIEW_BLR.

WORKAROUND: Use the execute statement to create a view that specifies the non-seg_str columns desired. Then use DAL to query the view. This way, the DAL server's table look-up won't stumble on columns with the data type seg_str. The following example does this:

```
Execute immediate "create view temp as
    select rdb$relation_name from rdb$relations";
commit ;                    /* don't forget to commit */
    Select rdb$relation_name from temp;
```

Copyright 1991, Apple Computer, Inc.

Tech Info Library Article Number:6405