February / March 2005

zData Perspectives

by Craig S. Mullins

DB2 Annoyances

If you are a regular user of any type of software you know what I mean by an "annoyance." Annoyances are those little things software does to drive intelligent people crazy. There isn't a piece of commercial software out there that is not plagued with annoyances and DB2 has its fair share of them. Let's examine a few.

Triggers: DB2's implementation of triggers has some aspects that are quite annoying. The first trigger annoyance is one that catches every new trigger user – and that is the requirement to first change the SQL terminator before you try to issue the CREATE TRIGGER statement. This is required because a trigger is composed of SQL statements, each of which is terminated by a semi-colon; but the CREATE TRIGGER statement is also a SQL statement and it needs to be terminated, too. If you use the same

SQL terminator, which defaults to a semi-colon, then DB2 gets “confused” and will refuse to create the trigger. The usual workaround is to use a semi-colon in the trigger text and change the SQL terminator for the UOW that executes the CREATE TRIGGER statement.

The exact way you specify a different SQL termination character depends on how you are issuing your SQL. For example, using DSNTEP2 you can specify SET TERMINATOR TO in a comment to use a different character to terminate the SQL. For example:

```
--#SET TERMINATOR #
CREATE TRIGGER...
  <code of trigger goes here>
#
```

That way the # character is read as the end of the CREATE TRIGGER statement (it would work the same way for any other SQL statement in the DSNTEP2 input). If you are working with DB2 on Linux, Unix, or Windows platforms you can use the command -td# in the Command Line Processor (CLP) to change the termination character. Or if you are using DB2 Control Center you can select the 'Tools Settings' option from the 'Tools' pull-down menu. Once there, check the box next to the line labeled 'Use statement termination character' and supply the character you wish to use in the little prompt box to the right of the line.

But the terminating the SQL statement is not the only annoyance when it comes to DB2 triggers. It is also very annoying that you cannot create a trigger specifying EXPLAIN YES for the SQL in that trigger at the same time. There is no EXPLAIN option for the CREATE TRIGGER statement, so the package that is created for the trigger is not explained. The way around this is to immediately follow your CREATE TRIGGER statement with a REBIND TRIGGER PACKAGE statement specifying EXPLAIN(YES). It works, but of course, it is annoying.

LOBs: After dealing with triggers, LOBs are probably the next most annoying part of managing DB2 databases and applications. LOB (stands for Large OBject) and it refers to a series of data types (BLOB, CLOB, and DBCLOB) that can be used to store very large multimedia data in DB2 tables. But the implementation for LOBs in DB2 seems to me to be a bit incomplete. Why is that?

Well, let's start with the problem that there is no easy way to LOAD them. If the total length of the LOB column and the base table row is less than 32KB, you can use the LOAD utility to populate the data into DB2. If the LOB column is greater in size you must use INSERT or UPDATE statements. That means you have to write a program to load LOBs into the database, and who wants to do that?

But that isn't the only LOB problem. A REORG of a LOB table space does not reclaim physical space. Reorganizing a LOB table space can help to improve the effectiveness of prefetch though, because it removed any embedded free space and will attempt to make the LOB pages contiguous.

Finally, and perhaps most annoying in the true sense of the word, is the need to create a unique index on the auxiliary table. The auxiliary table is the table in the LOB table space that holds the LOB data. The index is required. But why does the DBA have to create that index? No columns are specified and the index is mandatory. The CREATE statement will look something like this:

```
CREATE UNIQUE INDEX indexxname ON auxtab;
```

Couldn't DB2 create it automatically "behind the scenes" when the auxiliary table is created? There is no reason for DB2 to force the DBA to create this index.

Other Annoynaces: There are several more things that annoy me about DB2, but none are as annoying as triggers and LOBs. A somewhat annoying documentation aspect is that the SQL examples in the DB2 manuals are far too simple. In some cases, the examples are so simple that users do not utilize the available features as effectively as they could. For example, consider the simple examples used for CASE statements and table expressions. Additionally, the DSNZPARMs are not very well-documented. The best source for DSNZPARM information is in an appendix in the Installation Guide that includes links to descriptions in the manual. Of course, additional information is spread throughout the rest of the DB2 manuals, too. I'd really like to see a separate, comprehensive DSNZPARM manual.

In this column we've looked at just a few of the DB2 annoyances that drive me crazy. If I missed your favorite, please share it with me at craig@craigsmullins.com.

From zJournal, February / March 2005

.

Home.