DBA SHOPTALK

Synonyms: The appendix of SQL/DS?

BY R. G. EATON

Curing the Common Database

AS I LAY IN MY HOSpital room recovering from an emergency appendectomy, the surgeon described the operation. His confidence was comforting; that is, until he tried to explain the purpose of the tissue he had removed from me so hastily. He couldn't find the right words. And as he attempted to convince me that he really did know about appendixes, I concluded that he really didn't know their purpose. His interest in appendixes surfaced only when they became infected.

My initial impression of SQL/DS synonyms was similar to the surgeon's impression of the appendix. I wasn't sure what value they had, but as long as they did not cause a database crash, I didn't really bother to find out . . . that is, until the introduction of such database administration annoyances as:

☐ Naming conventions that are management-mandated

☐ Dynamic SQL embedded in COBOL

☐ Test-to-production promotion errors

☐ Security audits.

I had to seek new and creative solutions to these problems: I couldn't find relief in views, indexes, or even in views of views. I had to look elsewhere. My proposed solution was a liberal dose of synonyms and an additional standard or two, which could relieve the harsh sting of these DBMS irritants.

In the SQL/DS universe, synonyms are substitute names for tables and views. Naming requirements are the same as table names. Usage, however, is optional. Each synonym is SQL/DS user specific. Thus, a table may support zero, one, or multiple synonyms.

Since synonyms are user specific (rather than global), each USERID can have a synonym for any number of tables. Each synonym has its own life cycle—each remains active until explicitly dropped. A synonym is dropped when its associated table or view is dropped. Figures 1 and 2 present two types of synonym usage.

In Figure 1, users would receive the same results for each respective query:

```
Acctng:  SELECT * FROM BILL_ADDRESS
Mrktng:  SELECT * FROM PROSPECT_
           ADDRESS
Cstmrs:  SELECT * FROM CUSTOMER_
           ADDRESS
```

However, as illustrated in Figure 2, users would receive different results for each respective query:

```
Acctng:  SELECT * FROM BILL_ADDRESS
Payable: SELECT * FROM BILL_ADDRESS
```

The DBMS maintains synonym information in its system catalogs—specifically, the SYSSYNONYMS catalog. If the user hasn't referenced the table or view (thereby creating "unqualified" table or view names), the preprocessor will simply assume that their names are synonyms.

Many SQL/DS shops first used synonyms to abbreviate (or to make more memorable) table or view names, thus saving key strokes. The examples in Figures 1 and 2 represent this usage. Synonyms also give a user-specific context to a table or view name. But let's consider whether these benefits provide solutions to our burning DBMS issues:

☐ Naming conventions that are management-mandated. Typically, a simple installation includes a production and a test database. The test database reflects the actual production database structures and can use a subset of the data. In a hypothetical installation, the applications managers have decided to use different names for production tables and views than those used for test. We won't address the justification for this requirement; it's a mandate—just do it.

☐ Dynamic SQL embedded in COBOL. Preprocessing COBOL programs containing dynamic SQL occurs at normal compile/preprocess and program run times. When a preprocessed program with dynamic SQL is executed, it ignores the USER-ID of the preprocessing user and uses the actual (run time) USER-ID to resolve security privileges. This approach is contrary to nondynamic, or static, execution.

☐ Test-to-production promotion errors. In our hypothetical installation, the table names in the source code must be changed when migrating between test and production—which would expose the risk that test code could be corrupted after final testing and before promoting to production. Additionally, changing production code to perform maintenance could be incorrectly changed before the maintenance modifications are even started.

☐ Security audits. Static SQL COBOL programs resolve security privileges at preprocess time, not run time. Security is checked for the USER-ID preprocessing the program—not the actual user of the program.

Table Name:	CUST_NAME_ADDRESS	Customer Name and Address
Table Creator's USER-ID:	CSTMRS	Customer Service Department
Fully Qualified Table Name:	CSTMRS.CUST_NAME_ADDRESS	
USER-ID #1:	ACCTNG	Accountant
USER-ID #2:	MRKTNG	Marketing Staff
USER-ID #3:	CSTMRS	Customer Service Staff
USER-ID	**SYNONYM**	**TABLE NAME**
ACCTNG	BILL_ADDRESS	CSTMRS.CUST_NAME_ADDRESS
MRKTNG	PROSPECT_ADDRESS	CSTMRS.CUST_NAME_ADDRESS
CSTMRS	CUSTOMER_ADDRESS	CSTMRS.CUST_NAME_ADDRESS

FIGURE 1. *A table supporting multiple synonyms.*

Dynamic SQL COBOL programs resolve security privileges at run time. Each user must be granted appropriate privileges to execute *and* preprocess the program. A consistent method of controlling execution of both dynamic and static SQL is necessary. Additionally, a method of restricting production promotion is a fundamental requirement. With some ingenuity, synonyms can provide practical solutions.

THE SIMPLEST WAY to explain how to implement synonyms is by using an outline to show the procedure. Each element of this implementation progresses toward the next. These techniques will satisfy all described requirements with relative ease and minimal impact to performance.

I. *Naming conventions that are management-mandated.* Develop test and production naming standards for the following:

☐ USER-ID. Define a USER-ID for the preprocessor/compile execution. Synonyms will be defined for each USER-ID. (Examples are PROD1 for the production environment and TEST1 for the test environment.)

☐ CREATOR-ID. The CREATOR-ID will indicate the application. It should describe the application; for example, ACCTNG represents an accounting application. Using a prefix to specify the environment satisfies the requirement of different test and production table names. The CREATOR-ID is used when creating tables for the application. (Two examples include PACCTNG for the production creator, and TACCTNG for the test creator.)

☐ Tables. The table name should be a descriptive abbreviation of the table's purpose or type.

☐ Views. The view name should be a descriptive abbreviation of the view's purpose or type.

☐ Synonyms. The synonym is a concatenation of the CREATOR-ID and the table or view name (the prefix is dropped). Because the potential length of the concatenated CREATOR-ID and table/view name exceeds the 18-byte maximum, the table/view name will be truncated accordingly. Synonyms are the same for both test and production USER-IDs. (PROD1 creates the synonym CSTMRS_CUST_NAME_A for table PCSTMRS.CUST_NAME_ADDRESS, and TEST1 creates the synonym CSTMRS_CUST_NAME_A for table TCSTMRS.CUST_NAME_ADDRESS.

II. *Dynamic SQL embedded in COBOL.* Include explicit CONNECT instruction in all COBOL/SQL programs. Standardize the USER-ID and PASSWORD host variable names; all programs must use the same variable name. Identify tables by their PROD1 and TEST1 synonyms.

III. *Test-to-production promotion errors.* Develop an interactive preprocess/compile procedure (such as a REXX EXEC) to: First, insert the USER-ID and PASSWORD into the source code; second, preprocess and compile the program; third, remove the USER-ID and PASSWORD from the source code;

Table Name:	CUST_NAME_ADDRESS	Customer Name and Address
Table Creator's USER-ID:	CSTMRS	Customer Service Department
Fully Qualified Table Name:	CSTMRS.CUST_NAME_ADDRESS	
Table Name:	VENDOR_NAME_ADDR	Vendor Name and Address
Table Creator's USER-ID:	PAYABLE	Accounts Payable
Fully Qualified Table Name:	PAYABLE.VENDOR_NAME_ADDR	
USER-ID #1:	ACCTNG	Accountant
USER-ID #2:	CLERK	Accounts Payable Staff
USER-ID	**SYNONYM**	**TABLE NAME**
ACCTNG	BILL_ADDRESS	CSTMRS.CUST_NAME_ADDRESS
PAYABLE	BILL_ADDRESS	PAYABLE.VENDOR_NAME_ADDR

FIGURE 2. *A synonym supporting multiple tables.*

and fourth, grant RUN privileges to the appropriate users. The fourth step may be performed by a user with DBA authority rather than including it in the preprocess/compile exec.

The authorized preprocessor/compile user would enter the USER-ID, PASSWORD, and list of the program's users. Use its synonym to reference a table or view.

IV. *Security audits.* By implementing the interactive preprocess/compile procedure, all COBOL/SQL programs (including dynamic SQL) will be protected from unauthorized execution, thereby satisfying our security audit requirements. Furthermore, the source code is relieved of the USER-ID and PASSWORD. At preprocess/compile time, the synonyms are resolved to the USER-ID that is furnished to the preprocessor/compile exec.

The following three applications and creator-IDs are included in Figure 3:

Customer Service	CSTMRS
Accounting	ACCTNG
Marketing	MRKTNG

Each application has one table and program. The corresponding test tables are:

TCSTMRS.CUST_NAME_ADDRESS
TMRKTNG.PURCHASE_HISTORY
TACCTNG.ACCOUNT_BALANCE

All test table creators grant all privileges with grant option to the TEST1 creator-ID. Note that test tables are distinguished by the first character in the creator-ID.

Each program contains standard host variable names for the connect-ID and password. (The actual connect-ID and password are stored within a protected table.) Table/view names aren't qualified in the program; that is, the creator-ID isn't included.

REXX EXEC inserts the actual connect-ID and password into the COBOL source code just before the preprocessor and compiler are executed. After the preprocessor and compiler complete, the REXX EXEC removes the connect-ID and password. The preprocessing is performed with PROD1 as the creator/connect-ID.

Developers can preprocess and compile under their own creator/connect-ID. Therefore, if they require changes to an existing table, they can create the table under their creator-ID without changing the table name in the program. Meanwhile, the altered table can coexist with the test and production tables.

ALTHOUGH THEIR value wasn't initially obvious, I am now convinced that synonyms can play an important role in database implementations. They are much more significant than the novelty they were once considered. As discussed earlier, synonyms are a vital database component. While I hope this approach clarifies the value of SQL/DS synonyms, medical science will have to continue their search to determine a use for the human appendix.

Editor's note: The SQL/DS environment is the lead feature in this month's Product Watch. *See page 75.*

REFERENCES

IBM. "SQL/DS Concepts and Facilities," 1984, IBM order number, GH24-5013.

IBM. "SQL/DS Database Planning and Administration," 1987, IBM order number, SH09-8024.

IBM. "SQL/DS Terminal User's Reference," IBM order number, SH24-5017.

R. G. Eaton is a development team leader at Twentieth Century Services Inc. in Kansas City, Missouri.

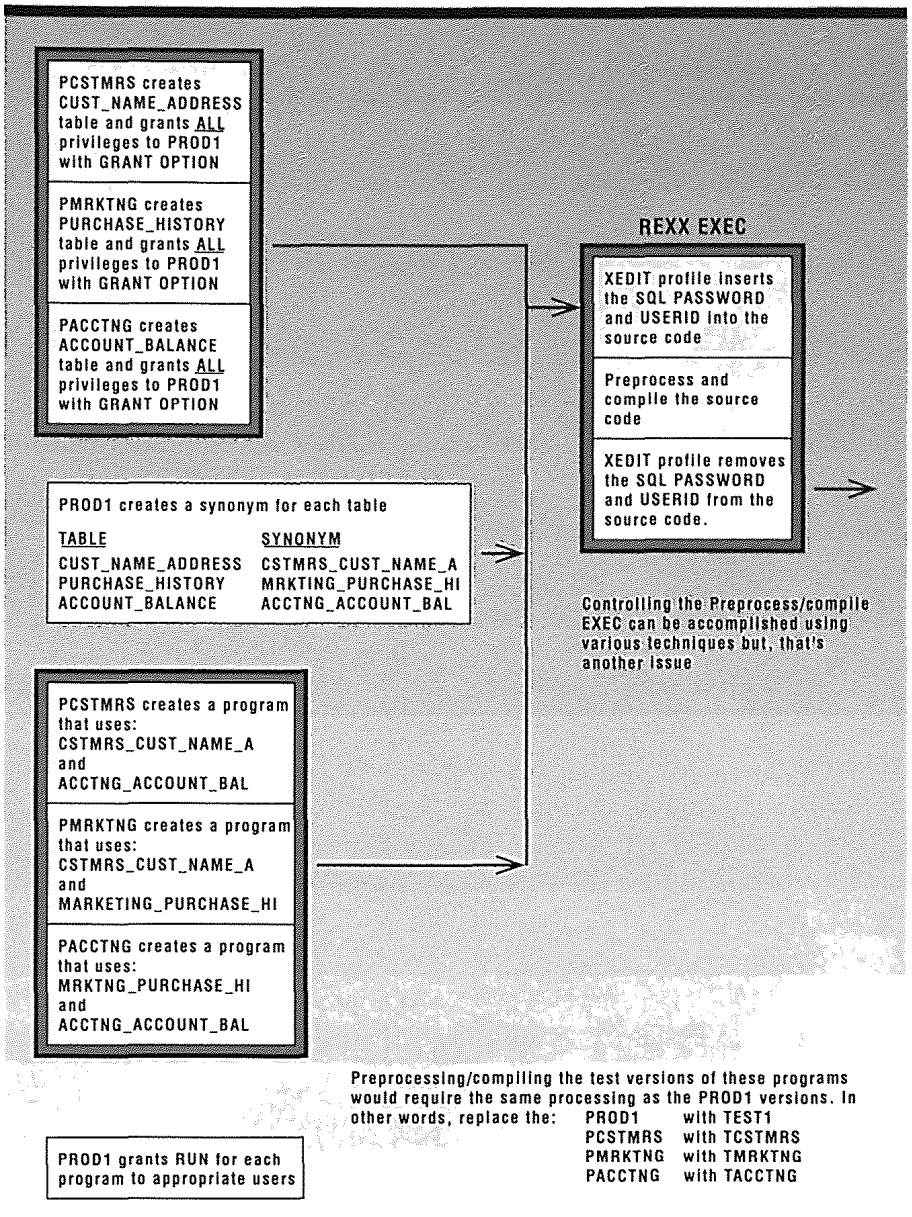

FIGURE 3. *Using synonyms to streamline test-to-production turnover.*