IMAGE/SQL: The Inside Story

Fred White
Adager Corporation
Sun Valley, Idaho 83353-3000 • *USA*
http://www.adager.com

IMAGE/SQL is an HP3000 MPE/iX product which contains TurboIMAGE as a component along with other software components which enable you to provide SQL read/write access to your TurboIMAGE databases.

Simply installing IMAGE/SQL on your HP3000 system does not automatically provide SQL access to any of your databases. To prepare for SQL access to a database you must provide IMAGE/SQL with descriptive information about the database and its authorized SQL users. This so-called "attachment" process is accomplished by running the IMAGESQL.PUB.SYS program and entering the appropriate commands.

Attaching to a DBE & detaching from a DBE

The information provided is stored in a set of privileged files called a Database Environment (DBE).

If, subsequent to attachment, perhaps weeks or months later, you need to make "structural changes" (add fields, delete sets, rename items, etc.) to your attached database, IMAGE/SQL requires that you first "detach" the database from the DBE and, after completing the changes, redo the attachment process in order that the access information needed by SQL corresponds to the revised state of the database.

History

The first release of IMAGE/SQL was made available to HP3000 MPE/iX users in 1993.

Unfortunately, the design of the data structures of the DBE files combined with a poorly designed method of maintaining those structures resulted in the attachment process being

tedious, error-prone and annoying, especially if you have many databases and users to deal with.

In 1993, as a member of the Adager development and support team, I shared in the effort of enhancing Adager to automatically perform the required detaching and re-attaching for you thus relieving you of this burdensome re-attachment process.

During this effort, I became aware of other design flaws and errors of omission within IMAGE/SQL which, while having no perceived impact on SQL access, needlessly threaten the integrity of your DBEs and require more disk space than necessary without providing commensurate administrative or functional benefits.

I have chosen a "question and answer" format to clarify my concerns and to identify the issues that HP should address to improve the usability and flexibility of future releases of IMAGE/SQL.

Is the database attachment process easy?

No. If you have only one or two databases and a handful of users, using the program IMAGESQL.PUB.SYS to establish and maintain the DBE metadata about users and databases should not be too painful.

For medium-to-large installations, the process is tedious and error-prone, the major reasons being:

1. The "ADD USER <username>" command requires the account name and the text: "WITH PASSWORD=<password> AND MODE=<mode>" for each SQL user of a database.

This could be simplified by establishing an access class, a DBOPEN mode and, possibly, an account name and have them apply to all subsequent "ADD USER" commands.

Actually, if my suggestions to HP are adopted (see below), there will be no need to specify the access class or DBOPEN mode within the "ADD USER" command.

2. There is no provision for SQL-accessible databases to share user names.

Instead of allowing a given set of users to be shared by more than one database, you are forced to enter the same set of "ADD USER" commands for each database.

With hundreds of users and/or databases, this wastes a lot of your time.

Do the IMAGESQL commands employ syntax familiar to MPE users and not pose any privacy or security threats?

No. The "ADD USER" command requires (who knows why?) an account name separated from the user name by a commercial at sign (@) instead of the familiar period (.).

The same command also requires an IMAGE password which

1. is neither necessary nor desirable since it is error-prone and less flexible than entering the corresponding IMAGE access class number and
2. unnecessarily creates a privacy and security threat since the password is logged to a non-privileged text file.

Is it easy to make structural changes to "attached" TurboIMAGE databases?

No. My "no" response here is prompted by the detachment/re-attachment complexities.

Making the structural changes is still as easy as ever. The problem is that, as mentioned earlier, the database must be detached and, after the changes, re-attached.

These re-attachment problems would be profoundly simpler and less error-prone, if the metadata about the database (and its OWNER and SQL users) were retained (instead of deleted) at detach time.

Also, it would make it much simpler for Adager to programmatically detach and re-attach if IMAGE/SQL included a set of procedures for accessing and updating the DBE metadata.

Are the safeguards against the DBE files getting out-of-synch adequate?

No. Since keeping database files undamaged and "in-synch" are two of the most difficult challenges facing database systems, I was shocked to find little or no discussion of a) transaction logging and recovery or b) the storing and/or restoring of DBEs or c) DBE diagnostic and repair facilities.

The SQLUTIL program does permit the "stopping" (see below) of DBEs and the "storing" of stopped DBEs.

If you perceive this as a solution to the problem of backing up your DBE files and databases to keep them "in-synch", you have been deceived. In reality it is a non-solution since

a) stopping a DBE terminates and prevents further SQL access but doesn't terminate or prevent non-SQL access and

b) other available MPE/iX store and restore facilities are not DBE-aware (i.e., any non-DBE related STORE or RESTORE may unknowingly include files of a DBE).

Consequently, the use of either backup/restore facility can unknowingly destroy the integrity of your DBEs.

Last, but by no means least, can anyone explain to me how stopping and storing a DBE (and later restoring a DBE) can maintain file synchronization when the current version of

IMAGE/SQL permits databases to be "attached" to more than one DBE?

Are the DBE metadata files and their structures designed so that the system is flexible and powerful while still conserving disk space?

No.

1. Each attached database requires its own table of users.

If databases could share groups, systems with many databases would conserve disk space and simplify group maintenance.

2. Each group entry contains a password and DBOPEN mode.

On large systems with large numbers of users, it is common for many of the users to be granted the same password and DBOPEN mode. In those environments, it is inefficient to have to repeat the same values in each "ADD USER" command as well as to waste the disk space required to store the values for each user instead of just storing them once for a group of users.

Proposals

These defects could be removed by changes to the attachment process such as:

1. Let databases be added to uniquely named SQL "domains" with each database a member of only one SQL domain.
2. Let SQL mapping information ("SPLITs" and "UPDATE TYPEs") be shared by all members of an SQL domain.
3. Let users be added to uniquely named SQL "groups".
4. Allow users to be added to more than one SQL group.
5. "ALLOW" a specified SQL group to access a specified SQL domain with a specified access class and DBOPEN mode.
6. "ALLOW" each SQL group other access classes and/or DBOPEN modes to other SQL domains.
7. "ALLOW" two SQL groups access to the same SQL domain only if both groups have no users in common.
8. Let utilities have procedural access to the DBE metadata.

For example, a utility may need to shutdown, or quiesce, or otherwise gain exclusive access to, a DBE (or a subset) in order to restructure and/or diagnose and repair or backup, restore or recover the DBE (or the subset).

Is there a good transaction logging facility covering DBE files supported with a fast, reliable recovery facility which is intelligent and has minimal input requirements?

No. A facility is desperately needed to control transaction logging and recovery at either the DBE level or, perhaps, the domain level, if that makes sense.

All file utilities, such as STORE and DBUTIL must be made DBE-aware so that they don't violate the integrity of DBEs.

(Hopefully, in the course of making utilities DBE-aware, the lab could also make them TurboIMAGE database aware even in non-SQL environments. This has never been attempted and its absence has led to out-of-synch databases for the last 19 years.)

Are there other issues which need addressing?

Yes. Miscellaneous issues have arisen and will continue to arise. For instance:

1. Will the STORE command bypass DBE files if the "TRANSPORT" option is present?
2. Will the STORE command have a "NODBES" option (or default)?
3. Will the RESTORE command have a "NODBES" option (or default)?
4. Will DBE files be restored on pre-SQL systems?
5. Is there a facility for diagnosing and repairing DBEs?
6. Etc., etc.,... etc.

Good news

Jim Sartain, HP's SQL Database Program Manager, invited me to Cupertino where we held an all-day discussion of my concerns and suggestions.

I am pleased to report that Jim's team was already addressing my concerns along with enhancement requests submitted by others.

Some enhancements have already been implemented and others are in progress.

Some issues are extensive in scope and will require the cooperative efforts of the MPE/iX Lab and the people working on HP's utilities. These, of course, will take more time.

Practical suggestions

If you are planning on using IMAGE/SQL, I have a few simple suggestions which will simplify your life and make it easier for you to maintain the synchrony of your database and DBE files:

1. Try to place them within a single MPE group so that they will share the same volume set as well as be easier to

backup and restore separately from the other files on your system.

2. Until future releases of IMAGE/SQL have simplified the detachment and re-attachment processes, establish and maintain job stream files and XEQ files to minimize errors and save your time.
3. When you perform stores and restores provide some fool-proof steps in your operational practices and job streams for maximizing success and for detecting and reporting failures.
4. Avoid partial stores and/or restores of databases and/or DBEs. Such practices should be resorted to only when attempting to repair database (or DBE) damage and only after experts have explored all other options.