IMAGE's coming of age

Breaking free from restrictions to Database transformations

F. Alfredo Rego

Profesor del Instituto de Informática y Ciencias de Computación
Universidad Francisco Marroquín
Ciudad de Guatemala, Guatemala

The original research report was presented by the founder of Adager Corporation during the 1978 HP3000 International Conference in Denver. Some of the terminology has evolved through the decades, but the fundamental ideas are timeless. In 1988, after 10 years in Guatemala, Adager Corporation moved its operations to Sun Valley, Idaho, U.S.A.

www.adager.com

Abstract A computerized database should reflect an organization's way of behaving. As real-world circumstances change — forcing the organization to adopt new ways and abandon old ones — the database should also adapt itself. This is easier said than done.

Hewlett-Packard provides tools, such as *DBUNLOAD* and *DBLOAD*, which allow a limited set of transformations to IMAGE/3000 databases. But these tools do not lend themselves to the easy implementation of the radical transformations that are sometimes necessary. The restrictions of these tools, we feel, are analogous to the restrictions imposed on children by loving parents.

Taking into consideration that children (just like computer users) eventually come of age and will do their own thing despite formidable obstacles, we have developed a software system called *DATABASE.UTILITY* to help IMAGE/3000 users out of their database transformation predicaments.

Buzzword-compliant editorial notes (written 21 years after the article's publication in 1978)

We shall use the term "Adager" (coined in 1979) instead of *DATABASE.UTILITY.* Adager means ***The Adapter/Manager for IMAGE/3000 Databases.***

This article reflects the state of the art in 1978. Computers have changed dramatically since then and, as Hewlett-Packard evolved IMAGE/3000 into TurboIMAGE and IMAGE/SQL, Adager changed accordingly. The few "functions" described in this paper have grown immensely and their names have also become less cryptic ("PAVEPATH" became "Repack Dataset" and "CAPDTAIL/CAPMASTR" became "Change Capacity").

With the Internet, the HP3000 has become a powerful web server that can support millions of transactions based on IMAGE databases — on a daily basis and with thousands of concurrent users who can be anywhere in the world. As a result, the HP3000 is known as HP's *Enterprise Database Server for Business-Critical Computing.*

Adager Corporation is conducting leading-edge research and development with direct Java access to IMAGE/SQL databases (bypassing the overhead of SQL, ODBC and JDBC used by common database systems such as Oracle). This Adager technology, called *ADBC* (a registered trade mark for *"Adager's DataBase Connectivity"*), allows any Java-enabled client to have direct access to IMAGE/SQL databases via the Internet from any place in the world (as well as via private intranets).

The performance of IMAGE/SQL databases has benefitted from improvements in computer hardware, as 16-bit CISC gave way to 32-bit RISC in the mid 1980s. For the new millennium, Hewlett-Packard is taking the HP3000 into 64-bit hardware architectures pioneered by HP in conjunction with Intel.

This article, however, is about the state of the database art in 1978. So, please rewind your calendar for this flashback. The only difference is that we now have to worry about terabytes instead of megabytes and about doing, in one thousandth of the time, what we could do leisurely back in the 1970s.

Motivation for the development of Adager

How can I be sure that my database design is perfect? How can I guarantee that I will *never* have to change my database to meet unexpected shifts in my organization's way of doing things?

If I can not answer these questions to my satisfaction, then what type of tuning (and fine-tuning) tools do I need to facilitate the constant and inevitable evolution of my database?

What worries me about the tools I have currently available to me? What type of questions linger in my mind as I dream of better and more effective ways to do what I have to do anyway?

- Why do I have to stop the operation of my live database for a long time, even when I only want to make minor changes like password reassignments? Could I minimize "down" time while I do certain non-radical transformations? *(The answer is "yes" with Adager.)*
- Why do I have to *DBUNLOAD* my whole database to magnetic tape before I transform my schema (assuming, of course, that I do not want to lose the live data I presently have!)? *(The answer is "you don't have to do so" with Adager.)*
- Why do I have to spend (a sometimes even longer) time to *DBLOAD* my previous database, even though I merely want to optimize the storage locations of a primary path's entries? Could I simply reshuffle these entries without having to worry about the consequences of having to manipulate the whole database as well? *(The answer is "yes" with Adager.)*
- Why do I have to *PURGE* my entire database, when all I want is to change the name of a data item? Could I simply make changes such as this without having to kill (and then re-issue life to) my database? *(The answer is "yes" with Adager.)*
- Why do I have to *EDIT* and recompile my schema, when I simply want to change the read/write capabilities of a user class? *(The answer is "you don't have to do so" with Adager.)*
- Why do I have to *CREATE*, from the newly produced Root-File, a brand-new database if the old one was just fine except for the capacity of a dataset? Could I change the capacity of a dataset without having to go through this onerous process? *(The answer is "yes" with Adager.)*
- Why am I at the mercy of subtle schema changes that *can* cause very unpleasant surprises, even after my database has apparently been successfully reloaded? Could I have some "editor" which would make sure I do not clobber my schema? Could I know, before I ruin anything, whether my new specifications are illegal? Could I have a dialogue to discuss the possible consequences of "slight" changes in transformation requests? *(The answer is "yes" with Adager.)*
- Why do I have to write special application programs whenever I need to transform my database in ways that are not supported by IMAGE/3000's transformation utilities? Could I have a flexible, non-procedural system? Could I do data-type conversions if the source data item does not match the target data item? *(The answer is "yes" with Adager.)*

Description of Adager

Adager is a suite of programs designed specifically to allow a large selection of transformations to IMAGE/3000 databases without having to mess around with magnetic tapes or schema recompilations.

Adager invests a good 90% of its time making reasonably sure that your requested transformations are legal and will not produce unpleasant results. Strict log-on subsystems verify that only authorized users access a database.

All of our design trade-offs have one main objective: to preserve database consistency and integrity. At the least sign of trouble, Adager does its best to back out and to salvage the original database.

When necessary, the RootFile is appropriately updated; MPE files are created or purged as needed; datasets are reorganized to include or exclude structural information; datasets and data items are renumbered if any intermediate elements have been eliminated, etc. You don't have to worry about the myriad details involved in even the simplest database transformation, because Adager automatically takes care of all the overhead.

Global database transformations

- COPY: Copies the RootFile and all of the datasets of a database.
- RENAME: Assigns a new name to a database and changes MPE file names as well as internally-kept IMAGE names.
- PASSES: Reports, lists, modifies, assigns, re-assigns, takes away, etc., maintenance passwords and read/write passwords and level/class numbers.

Transformations of detail datasets

- NEWDTAIL: Adds new detail datasets to the database (with the appropriate new data items for new fields, if needed).
- CAPDTAIL: Modifies the capacity of a detail dataset, preserving all current chains and making sure, in the case of a decrease in capacity, that the target capacity is not less than the lowest permissible capacity for the given dataset's status.
- KILLDET: Deletes a detail dataset.

Transformations of master datasets

- NEWMASTR: Adds new automatic or manual master datasets to the database (with the appropriate new data items for new fields, if needed).

- CAPMASTR: Modifies the capacity of a master dataset, preserving hashing properties and chainhead structural information. Attempts to minimize synonym probabilities by suggesting prime-number capacities in the neighborhood of your requested capacity, according to the key's hashing type.
- KILLMAST: Deletes a master dataset, making sure that it is safe to do so and that no detail orphan chains will be left hanging without master chainheads.

Transformations of data items

- NEWITEM: Adds new items to existing datasets.
- KILLITEM: Deletes a data item from the global item table.
- NEWFIELD: Adds a new field to a dataset (a field is a specific reference, in a dataset, of a global data item).
- KILLFIELD: Deletes a field from a dataset.
- RDEFITEM: Redefines the type of a data item (from integer to byte, for instance) and does all the appropriate data conversions if necessary.

Transformations of element references

- NEWNAME: Assigns a new name to a data item or a dataset. Checks non-duplicity and legality of the requested name.

Transformations of access paths

- NEWPATH: Defines an optimized access path between a master dataset and a detail dataset.
- CLOSPATH: Deletes a path between a master dataset and a detail dataset.
- SORT: Adds a collating specification for all detail chains in a given path.
- UNSORT: Removes the collating specification for all detail chains in a given path.
- PRIMARY: The path most frequently accessed in chained mode should be specified as the primary path for a detail dataset. Should this state of affairs change, you can redefine the primary path for the detail dataset.
- PAVEPATH: Reshuffles the entries of a detail dataset so that the entries of each chain in a given path will be in contiguous storage locations for efficiency's sake in chained retrieval.

Conclusion

Adager keeps track of all IMAGE/3000 internal housekeeping, while your database evolves.

You are now free to concentrate your attention on the *only* housekeeping task that really matters: Your database's accurate reflection of your organization's way of doing business.

You can now specify — without fear — whatever you need in your database today, according to today's requirements. Tomorrow, you can easily remodel your database according to the constantly changing conditions of your business.

Thanks to Adager, ***The Adapter/Manager for IMAGE/3000 Databases***, you have broken free from restrictions to database transformations.