The SIGIMAGE Newsletter is published by the INTEREX Special Interest Group for IMAGE/SQL Databases.

INTEREX is the International Association of Hewlett-Packard Computer Users.

Volume 5.1 — Spring/Summer 1995

From the Chair....

Jerry Fochtman
SIGIMAGE Chairman
Exxon Chemical Americas
Houston, TX, U.S.A.

All the reports are in, and guess what?... IPROF was another success! But this should come as no surprise to those of you who attended this year's conference. Credit goes to Ken Sletten, Conference Chairman as well as the Interex staff, HP, the SIG Leaders, and of course the users who attended and worked together to improve the tools we use.

The comments I've received from those in attendance were extremely complimentary in terms of the topics discussed and the positive outcome of the meetings. Jim Sartain, HP IMAGE/SQL Lab Manager, has also indicated that his team was very positive about the feedback they received from the discussions with the users. So indeed the process works!

We in SIGIMAGE are also very fortunate to have so many talented individuals who support our group. When I was unable to attend this year's conference at the last minute, Steve Cooper graciously offered to lead the sessions. Given the enthusiastic feedback I've heard, Steve did his usual bang-up job...Thanks Steve!

Speaking of talent, check out Wirt Atmar's report on the IPROF conference in this issue of the newsletter. Wirt has covered many of this year's highlights and clearly his contagious enthusiasm is reflected in his article.

Interex '95

Efforts are underway for this year's Interex Conference at Toronto. SIGIMAGE's meeting will take place during 'prime time,' on Thursday, August 17th, 3:00 to 4:50 pm. The highlight of our meeting will be HP's review of their plans for the upcoming year given our feedback on users' needs. Last year HP announced their plans to include sorted sequential access (b-trees) in IMAGE by year-end 1995. I wonder what delightful surprises may be in the wings for their upcoming plans....! If you're at the conference, plan on attending to find out!

This year's conference also includes a great lineup of presentations and discussions on database topics from many knowledgable individuals. The sessions begin on Tuesday and last through Friday. A current list of sessions is included in the conference registration packet for your review. If you haven't yet received your packet, contact Interex @ 1-800-INTEREX (or +1 408 747-0227 from outside the U.S.A.)

Passing the Baton...

Over the course of time, people's available time to work on volunteer activities changes. Denys Beauchemin has taken over the role of this year's Interex Conference Chairperson. This is in addition to giving IMAGE/SQL presentations at various regional users groups, working a full-time job along with preparing for a large, 'special' event this summer. Given all this, Alfredo Rego has offered to step-in and serve as our newsletter editor for the next several issues. Knowing how busy Alfredo is, I sincerely appreciate his volunteering to work with us and fill this important role.

I want to personally thank Denys for all his efforts in producing our newsletter. As a compliment, several other SIGs have said that every time they think their newsletter is comparable to SIGIMAGE's, Denys raises the bar on them and they have to work just that much harder. I hope Denys will be able to return as our editor sometime in the future!

See you in Toronto!

New Responsibilities

Jim Sartain
Hewlett-Packard
Cupertino, CA, U.S.A.

Five years ago I accepted responsibility for leading Hewlett-Packard's TurboIMAGE Database Team. This turned out to be one of the two best professional decisions I've ever made. The other was joining HP.

A couple of high points for the team over the past few years were:

1.) The great customer acceptance of the IMAGE/SQL product. The team really had an unforgetta-

bly great time developing the product and hearing great stories about the big difference the product made to customers.

2.) At the 1993 San Francisco INTEREX I received the Marc Hoff Award for Distinguished Service by an HP employee. I've always felt that I accepted this award on behalf of my team. There is no way I could have made a big difference without the support of my extraordinary team. Another key factor was the great partnership I developed with SIGIMAGE committee members.

I recently had to make another big professional decision. HP offered me the opportunity to pursue an M.S. in Management of Technology (MOT). This intense two year program will enable me to advance my knowledge in areas that are important to HP and my own personal development. This two-year program is offered through the National Technological University (NTU) and will begin this August. Most of the lectures are broadcast over satellite. In addition there is a one to two week residency every three months.

Some of the courses I will be taking from each of the following universities are as follows: *Technology and Economic Analysis* — University of Pennsylvania. *Manufacturing Systems and Technology Strategy* — Georgia Institute of Technology. *Managing and Leading Technical People* — Rutgers. *R&D Management* — Lehigh University. *Taking Technology to Market* — University of Alabama, Huntsville. *International Business and Technology* — George Washington University. *Strategic Management of Technological Innovation* — Rensselaer Polytechnic Institute.

In order for me to be successful in the MOT program my current workload had to be reduced and it was not possible for me to continue doing the same job. My HP Database R&D responsibilities have been transferred to Reynold Schweickhardt. I think that you all know Reynold from his extensive work with RUGs and SIGALLBASE. Reynold and I shared the same job for the past two years. Reynold personally led the effort to develop and release an ODBC driver for IMAGE/SQL and ALLBASE/SQL. ODBC is a key and critical component in HP's SQL database strategy. Also, he has been responsible for managing HP's relationship with several major HP3000 customers.

I've been given some new responsibility managing the MPE kernel, System Performance and Limits. Reynold and I share the same office and I'll continue to be very involved in making CSY business decisions. I look forward to continuing to attend SIGIMAGE meetings whenever possible.

Unfortunately, I'll miss the Toronto INTEREX meeting because that is the orientation week for the MOT program.

I look forward to continuing to see the HP Database products and the customers applications that use them evolve.

SIGIMAGE at IPROF, April 1995

Wirt Atmar
AICS, Inc.
University Park, NM, U.S.A.

Against all odds, the SIGIMAGE meetings at IPROF continue to get better and better—and more and more well attended. The SIGIMAGE meeting was conducted in two segments for a total of 7.5 hours over two days, and most attendees agreed that one more hour could have been profitably used. Attendance at the two sessions was consistently 120 to 140 persons, which was near (or a little beyond) the room's capacity.

SIGIMAGE has become the model SIG for Interex. And just as notably, the database lab's relationship with Interex, SIGIMAGE, and the IMAGE users themselves has also become the model— not only for most of CSY—but for much of the remainder of HP as well.

A great portion of the credit of the success and the effectiveness of SIGIMAGE must go to Jim Sartain, HP's R&D Database Lab Manager. Jim has consistently said that he "has the best job in HP," and he said it again at IPROF. Not only is it evident that he truly believes it, Jim's attitude and personality have been responsible in large part for the revolutionary changes in the relationship that have taken place within HP since the Boston Interex meeting of 1990. A high degree of trust and concern, accompanied by open and honest answers, set the tone for the meeting—and that alone made the meeting a great pleasure to attend.

A number of significant improvements to IMAGE were either discussed or announced at this year's IPROF meeting. Among them are the following items:

A change in design procedure

In the opening remarks (which included Harry Sterling speaking on the future of the HP3000), Jim Sartain explained the new design philosophy adopted by the database and MPE design groups. The previous method had consisted of (i) gathering up enhancement and service requests over an (often substantial) period of time, (ii) ranking them as to their necessities and desirabilities, and (iii) then locking down the design objectives of the next general release. Generally this entailed a significant follow-on period where no further input was taken while the new design modifications were made, tested, and ultimately released.

The philosophy adopted by the labs is now one of iterated evolutionary change —and there are some significant advantages to this

approach: the time to market is significantly reduced for critical feature enhancements, user feedback is more instantaneous and more relevant, and should some design path prove to be regretted, it can be reversed before every customer receives the new software and that regret is permanently locked into the design.

This approach has been commonly adopted by many of the more agile software companies, and the benefits are overwhelming—especially if it is critical that an evolving suite of software must meet rapidly changing customer needs and expectations. However, this approach has downsides as well.

At AICS Research, we similarly adopted this iterated evolutionary approach three years ago when we began updating our customers through the use of high-speed modems, and I am sure that our experiences will be repeated by HP. Perhaps the worst problem that accompanies this technique of iterated evolutionary design is that it encourages the formation of two classes of customers: (i) those that receive frequent (generally monthly) updates, and (ii) those that receive much less frequent updating than they did before with fixed releases, simply because there comes to be strong pressure to avoid setting hard-and-fast "release" dates.

Updates-by-modem has proven to be an extreme example of this iterated evolutionary design procedure, but the advantages are so significant that they cannot be ignored: software can be made exceptionally robust and reliable by iterated repair, it can be designed to customer needs and wants, and it can be done quickly (and with the modem, software installations can be handled from one central location so that the process is completely transparent to the customer).

The graph Jim drew of the iterated release process showed a series of releases, where the first beta release was the first great "bump" on the chart, the second one being smaller than the first, and each being smaller yet again. Unfortunately, those decreasing bumps haven't been our experience; rather, as our "beta" population has grown to be about 50% of our installed base, the bumps have come to be about the same size, and every release seems to carry an "X" designation with it. We have failed to guard against constantly continuing the evolutionary process by not saying at some point, "this is a release that everyone should get," simply because the iterated evolutionary process never seems to engender those decisive break points. HP will undoubtedly have to guard against that effect also.

Indexed access in IMAGE/SQL

Nonetheless, IMAGE/SQL is still an evolving product, and HP is now acting in the best traditions of an organization working hard to get a new product out in the least possible time with the highest degree of effectiveness. Because of this evolutionary approach, current assessments of IMAGE/SQL's capabilities should not be used as a measure of what it will become. Among the most significant deficiencies currently associated with IMAGE/SQL is that the query optimizer does not take advantage of either standard IMAGE keys or the third-party indexes. This represents a truly significant performance degradation—but it won't be much longer, as explained by Baharti Desai of the database lab.

Version X.G1.00 of IMAGE/SQL now properly registers the IMAGE keys to the DBE so that the ALLBASE (SQL) query optimizer knows of their existence and can properly use them. With this version's update, if you already have an IMAGE database attached to one or more DBEs, all that you must do is detach and reattach the database to properly register the indexes. If a new IMAGE database that is being attached to a DBE for the first time, the process is simply automatic.

IMAGE/SQL has the potential to become the fastest, most robust and most reliable RDBMS on the market, but it's going to take some time to get it there—and some level of patience is going to be necessary in the user base. Don't make the mistake of dismissing IMAGE/SQL too early, or setting your expectations too high too early. But it is going to happen—and IMAGE/SQL could well become Oracle's "worst nightmare."

IMAGE/SQL security enhancements

Pinky Lee, of the database lab, asked, "Where is security handled in IMAGE/SQL? Is it at the SQL level or is it at the IMAGE level?" Steve Cooper, former SIGIMAGE chair and leader-of-this-meeting, answered in his own inimitable way, with a broad smile, "Yes!"

And that's the right answer, of course. Security is handled in both arenas, in (i) the traditional user read/write passwords associated with IMAGE and (ii) with the newly modified user class passwords, similar to those used with ALLBASE and other "standard" RDBMSs.

The standard IMAGE read/write class passwords retain primary security responsibility, as they should. If the attached user class does not have sufficient read/write capability, dataitems and datasets outside of the user's capability will not be seen, read, or registered in the DBE. Ms. Lee described in some detail how the SQL user class passwords will be mapped onto the traditional IMAGE security classes. But her discussion was not merely a description; rather the IMAGE lab is now consistently taking the

view that IPROF has come to be an important part of the design review process—and the attendees have never been so polite as to refrain from offering their suggestions. It was clear to everyone that these discussions represent an extremely valuable resource, both to HP and the future users of the product.

Jumbo datasets

Subra Ramesh of the database lab explained in detail the method that HP has taken to temporarily accommodate "jumbo" datasets in IMAGE. Without this enhancement, the largest possible IMAGE dataset could only be 4GBytes. This limitation is not a limitation with IMAGE but with the file system.

That file size limitation was put into MPE at the very beginnings of the HP3000's design, but then who would have thought in 1972 that 4GB would have proven to be a limit, especially when system discs totalling 100MB were only then a distant possibility.

With the addition of jumbo datasets, up to 10 "chunks" of these 4GB datasets can now be sewn together for a new maximum of 40GB. This linking together of these dataset chunks is accomplished through the use of the POSIX hierarchical file system names—and there is a certain "kludgey-ness" to the solution. It's much more of a "fix" than it is a solution, and it can only be of temporary value. If the pressure exists to maintain datasets that exceeds 4GB, a 10x extension will soon become unacceptable, too. The relatively "small" dataset size limit of a maximum of 4GB (or 40GB) is going to become a severe problem in the near-term future—simply because of the changing nature of what people are going to consider appropriate material to be stored in databases, accompanied by the decreasing costs of storing that information on-line.

Because the problem lies in the file system itself, and not in IMAGE, Craig Fairchild, the MPE file system architect, was asked during the conference, "Is it possible for the file system to be fixed?" Craig said, "Anything's possible." And that, too, was precisely the right answer (although Steve Cooper was correct to point out the magnitude of the task that lies before HP).

B-trees in IMAGE

Following critical item update (CIU), the addition of b-trees to IMAGE has been consistently the most requested enhancement to IMAGE over the past 17 years. Fred White, now of Adager but then one of IMAGE's designers, has said that b-trees were one of the enhancements that was scheduled for Release 2 of IMAGE. But that release never happened. The design team was broken up before a second release could be finished, or even much begun.

But now b-trees are no longer an option. If IMAGE/SQL is going to realize the promise that it potentially offers as a truly competitive RDBMS, b-trees are an absolute necessity. The method that will be employed in IMAGE is surprisingly simple—and it will be extraordinarily efficient in terms of disc space usage and CPU costs.

Tien-You Chen, of the database lab, explained the technique: a smaller-than-normal b-tree file will be attached to each master dataset that exists in an IMAGE database, whether it is an automatic or a manual master. These b-trees will only contain the key search item values found in the master datasets. There will be none of the pointers that are normally associated with the more common b-tree implementations. They simply aren't necessary; the pointer structures already exist in IMAGE's chains. The relationship between the master dataset's values and the b-trees will be one-to-one.

There are several truly significant advantages to this technique offers over externally maintained b-tree files:

- The b-trees will need to be updated (a value added or dropped) only when a key search item value is either added to or deleted from the master dataset, which is generally a very rare event. Thus process concurrency will be very high—with an absolute minimization of b-tree maintenance costs to the database or interference with other contending database processes. Indeed, on many datasets, the addition of b-trees will prove to represent an almost invisible cost.
- Because only unique search values will be stored in the b-trees, compression ratios will be intrinsically very high (average chain length x the number of affected datasets). Moreover, the disc space expansion due to the addition of b-trees over a normal IMAGE database is expected to only be in the range of 2-5%.
- No pointers are stored in the b-trees, only search item values. Thus reorganizations of detail datasets, which often engender substantial performance improvements, will require no modification of the b-trees.
- Hashed searches will be unaffected by the addition of the b-trees (the b-trees will simply be ignored), thus the method is completely compatible with all prior IMAGE usage. But because the b-trees are attached to the same masters that begin the chained searches, both search techniques (hashing or range/generic searches) will use precisely the same chain structure that already exists in IMAGE. The only difference lies in the use of the chains. Under hashed searches, only one key item is searched for at a time, as specified by DBFIND, Mode 1.

But with the b-trees, a set of chains will be searched, one after another, until the set of all qualifying search values (as determined by the b-tree search process) has been exhausted. The set of qualifying key values will be determined by a DBFIND, Mode 2 (a new mode), search of the b-tree. This set of key item values will then be applied one after another, in turn, against their respective chains, until all appropriate chains have been searched.

The SIGIMAGE ballot

The top two enhancement requests at this year's SIGIMAGE meeting were the addition of a DBQUIESCE intrinsic and the installation of the DDL (data description language) that will begin to complete the transition of IMAGE into a fully functioning RDBMS.

The DBQUIESCE intrinsic will allow third-party vendors, as well as HP itself, to begin to perform some real magic on the database—while users are (at least nominally) still logged on and have their databases open.

The top two enhancement requests are closely related to each other. Indeed, DDL requires DBQUIESCE. The DDL enhancements of being able to add and drop dataitems, datasets, and indexes are consistently touted as among the most attractive attributes of relational databases. However, the capacity for this sort of database restructuring in a live database are obviously limited in regard to what most people have come to expect from products such as Adager, DBGeneral, Flexibase, and DBChange—but changes such as (i) the capacity to add new dataitems to end of a dataset, (ii) the capacity to add or drop indexes, (iii) the capacity to add new detail datasets, will be done on-line, in real time, and without having to have users or processes be clear of the database first.

The following is the ranking of the SIGIMAGE ballot, with the number of votes per item in parentheses.

1. (1165) Provide DBQuiesce intrinsic.
2. (1023) Ability to add/drop datasets, indexes and items.
3. (869) Provide dynamic master dataset expansion.
4. (695) Extend IMAGE/SQL data access to KSAM and MPE files.
5. (548) Improvement of DBA functions for managing AuthIDs.
6. (544) Tracking files associated with databases.
7. (479) Provide ability to retain/reuse attach parameters across a detach/attach.
8. (459) Read-only DBLOCK mode (someone placed his entire $100 on this one).
9. (426) Provide ability to enable/disable SQL access without detaching.
10. (411) IMAGE measurement interface for performance data.
11. (406) Multi-record/Multiset DBGET/DBPUT/DBUPDATE.
12. (406) Multiple database dynamic transactions.
13. (389) GETUPDATE to update record w/o DBGET.
14. (262) Provide ability to better influence SQL query optimizer.
15. (261) Support SQL NULL item.
16. (222) Support SQL savepoint.
17. (220) Performance improvements with DBUPDATE of sort fields.
18. (200) Dynamic transactions in QUERY.
19. (197) IMAGE/SQL transactions spanning ALLBASE and IMAGE.
20. (192) Provide default value for data items.
21. (163) Tool memory area.
22. (148) DBCONTROL/DBINFO access in QUERY.

The following is a list of new requests:

1. Support of BLOBs.
2. Capacity changes without ATTACH/DETACH.
3. Date/Time datatypes.
4. Ability to specify flags and settings in DBSCHEMA.
5. Non-destructive creator access across groups for DBUTIL.
6. Global files.
7. User setting of maximum chain lengths.
8. DBINFO mode to return fully qualified DB name.
9. DBINFO from QUERY.
10. DBGET using record # in QUERY.
11. DCE compatibility for IMAGE (security, directory services, and so on).
12. SQL-savvy DBSCHEMA.
13. Make CIUPDATE the default.
14. Have DBUTIL ignore file equations.
15. SHOW INDEX command to show other than composite keys.
16. MULTIFIND update/delete.
17. Enable rename of IMAGE table in SQL.
18. Find FIELDA = FIELDB in QUERY

Other matters of interest

In two separate management roundtable discussions, pointed questions were asked of HP management—and only partially answered.

The most important of these was the now-perennial and continually perplexing question, "Why doesn't HP advertise the HP3000, and especially its IMAGE database outside of the installed base, given its extraordinary quality and the loyalty and satisfaction of its large user base?"

Harry Sterling responded that HP must walk a delicate line so not

as to offend their other database partners—meaning primarily Oracle.

In a recent poll reported in a trade magazine, Oracle has come to be regarded as the second most respected software company in the world (Microsoft being first). Clearly that poll wasn't conducted at IPROF. Nonetheless, that result has clear business implications to HP management. They feel that they are not going to sell the large HP3000 systems to the largest customers without Oracle, and they seem willing to ignore IMAGE to the point of almost killing it to make these sales. However, Oracle was not even in the top 10 companies in last year's poll; their rise has been the result of aggressive marketing over the past several years. There is no reason that IMAGE and the HP3000 cannot see a similar rise, if only HP would adopt similarly aggressive marketing. But it doesn't seem likely to happen: the other respondent to the question was Olivier Helleboid, and his answer was, "Advertising is expensive."

Acknowledgments

Although there are a great number of people who contribute to the success of SIGIMAGE and IPROF, there are also several who stand in need of special recognition for their efforts in the continuing success of SIGIMAGE at IPROF. They are most especially: Ken Sletten, Jerry Fochtman, and Steve Cooper.

Perspectives on Managing IMAGE/SQL

Evan Rudderow
GEC Marconi Electronic Systems
Wayne, NJ, U.S.A.

Fred White's articles *Migrating to IMAGE/SQL* (The SIGIMAGE Newsletter 4.03) and *IMAGE/SQL: Enhancements for the DBA* (which was sent with that issue of the newsletter) present many thoughtful observations and suggestions about IMAGE/SQL; those articles prompted me to reflect on how I manage IMAGE/SQL databases.

I have both Allbase/SQL and IMAGE/SQL; I'm running version F0 under MPE/iX 4.0 on one system and version G0 under MPE/iX 4.0 on my other system. The databases in my DBEs are about equally divided between IMAGE/SQL and native Allbase/SQL databases; generally native Allbase/SQL databases, containing warehoused data, are used for decision support applications while IMAGE/SQL databases are used for light reporting and for processes which require "real time" access to application data. Decision support applications use warehoused data instead of production data for a couple of reasons:

- I don't want decision support processing interfering with OLTP processing.
- Production data is constantly in a state of flux, so a query executed now will probably return different result from a query executed, say, 15 minutes ago. Warehoused data, on the other hand, is relatively stable, making queries reproducible.

I mentioned that decision support applications generally use native Allbase/SQL databases; there is one exception—an IMAGE/SQL database named MHSUM. MHSUM was initially designed as a quick and dirty prototype to show the value of implementing client-server based decision support tools. Additionally it was designed before this site understood that they had the Information Access/SQL server components merely as a consequence of owning Allbase/SQL; hence MHSUM's TurboIMAGE implementation—the plan had been to use Information Access Server (the server for TurboIMAGE databases) to access it.

In the discussion which follows I'll be using MHSUM as my example. I'd like the reader to be forewarned that MHSUM has a trick that enables it to be managed in the manner describe below; I'll explain later what the trick is. Additionally, since Fred White's articles prompted my reflection on my IMAGE/SQL management practices, I'll be juxtaposing my management practices with Fred's suggestions; that juxtaposition isn't meant to be critical of Fred's suggestions—I'll be using his comments to better illustrate mine.

MHSUM User Management and Security

In both of the articles cited above Fred discusses the related issues of security and user management at length; he does this for good reasons: the security and user management aspects of IMAGE/SQL are tedious, labor intensive, prone to error, ridden with inadequacies and security holes, and unlike anything else you I'm familiar with. Let me explain how I've implemented security and user management for MHSUM.

I don't use the security and user management features of IMAGE/SQL at all for MHSUM; there are a

number of reasons for this, among them:

• The TurboIMAGE database has only two passwords.

• I don't like having my DBEs cluttered with all the views that are created for each new user class that is added by an IMAGESQL ADD USER command.

• The IMAGESQL utility isn't very good about cleaning up after itself—some things are left lying around in the DBE after a TurboIMAGE database has been detached.

• I don't want to deal with two different security/user management mechanisms—one for IMAGE/SQL tables, the other for native Allbase/SQL tables.

So, how do I manage security and users? The procedure is something like this:

1. I attach MHSUM to the Allbase/SQL DBE using IMAGESQL. This creates a set of tables whose owner is <dbname>; that owner has all authorities on the tables. IMAGESQL also creates a view for each dataset, by appending "_V0" to the dataset name.

2. Still in IMAGESQL, I perform the required SPLITs and UPDATE TYPEs.

3. I invoke SQL and REVOKE all authorities from the IMAGESQL-created tables and views; e.g.: REVOKE ALL ON MHSUM.MH_SUMMARY_D FROM PUBLIC;

4. I then use SQL to create the authorization groups which will have access to the tables and views; e.g.: CREATE GROUP MHSumUsers;

5. Then I add the authorization group to a generic authorization group which has connect authority to the DBE; e.g.: ADD MHSumUsers TO GROUP EndUsers;

6. I then use ISQL to grant the necessary authorities (e.g. SELECT, UPDATE, INSERT, DELETE) to the groups I've created; e.g.: GRANT SELECT ON MHSUM.MH_SUMMARY_D TO MHSumUsers;

7. Finally I use ISQL to establish security for individual users by issuing the appropriate ADD <user@account> TO GROUP <authorizationgroup> command; e.g.: ADD EVAN@ENDUSER TO GROUP MHSumUsers;

Astute readers will note that this seems to grants creator access to all users; actually that's not true—it's part of the trick. In any event user authorities are fine tuned by judiciously granting privileges: users requiring read-only access are added to authorization groups which have only SELECT authority while users requiring both read and write access are added to authorization groups which have SELECT, INSERT, DELETE, and UPDATE authority. And, although I've not yet had the occasion to do it, if I had to implement column security (i.e. TurboIMAGE item security) I would do so merely by:

1. Creating a view containing only the columns to which the user requires access.

2. Creating an authorization group.

3. Adding the newly created authorization group to the generic EndUsers authorization group.

4. Granting SELECT authority on the view to the authorization group.

5. Adding the particular user to the newly created authorization group.

Admittedly this creation of views and authorization groups is what IMAGESQL is doing for me when it creates its "_V#" views for each new user class during an ADD USER, but I prefer to select my own view names.

A nice feature of my method of user management with this database is that I can easily restrict users to only those tables that I want them to have access to; an example of how this is beneficial is my site's use of HP's Information Access for decision support applications: When a user requests a list of tables in Information Access he is presented with a listing of all tables to which he has been granted SELECT authority (it probably works for more than just SELECT, but for decision support all I ever grant is SELECT). Using the IMAGESQL method of user management the user would see all of the datasets in the TurboIMAGE database—details, manual masters, and automatic master—in the list of tables. Well, it's doubtful that he needs to see the automatic masters; after all, if showing automatic masters is a good thing, then Information Access should show him Allbase/SQL indices as tables as well! Furthermore, depending on his user class, he might see each dataset twice: once under its IMAGESQL converted name and again with the "_V#" suffix; all this does is confuse him.

So, you see, I perform all of the security and user management tasks for MHSUM within Allbase/SQL just as though MHSUM were a native Allbase/SQL database. Indeed, if I use IMAGESQL to detach, then re-attach MHSUM to the DBE (and because of the way the data refresh process was designed this happens every week) there are no user management issues involved.

MHSUM Management and Fred White

In *Migrating to IMAGE/SQL* Fred suggests:

There is no provision for the attached database to share a set of users. Consequently you must reenter the same set of user names each time you attach a new database. Pity the lady from the University of Notre Dame who must correctly

enter 16,000 unique logon names. Let's hope she has only one database.

With databases like MHSUM I can use ISQL to grant a single authorization group access privileges to multiple databases (irrespective of whether they are IMAGE/SQL or native Allbase/SQL databases). Then I merely add users to that single authorization group. For example, given IMAGE/SQL databases FOO and BAR, I would invoke ISQL and issue commands like these:

```
CREATE GROUP
FOO_BAR_USERS;
GRANT SELECT ON
FOO.FOO_DETAIL to
FOO_BAR_USERS;
GRANT SELECT ON
BAR.BAR_DETAIL to
FOO_BAR_USERS;
ADD MGR@EVAN TO GROUP
FOO_BAR_USERS;
```

Fred also suggests:

For each user name that you enter, you must supply the IMAGE access class PASSWORD and the user's DBOPEN mode and the name must include the account name even if all the users are in one account. In addition to this being tedious and error-prone, it creates a security risk since the entire command is logged to an EDITOR file for later re-attachment purposes. (Note that this "locks in" the PASSWORD in that subsequent changes to access class PASSWORDs in the database will cause ADD USER commands referencing old PASSWORDs to be rejected.) It also precludes referencing access classes with non printable PASSWORDs and, finally, when you are performing hundreds of ADD USER commands, it is very annoying for you to be "warned" about each of them, especially when you have no choice.

As I've demonstrated, with MHSUM I don't have to muck around with access class passwords and DBOPEN modes; consequently that security risk has been eliminated—as has the worry about subsequent changes to passwords. Nor do I have to deal with the annoying "command containing a password has been logged" messages or the issue of non-printable characters in passwords.

Under the heading *Global Design Changes in IMAGE/SQL: Enhancements for the DBA* Fred suggests:

3. Create SQL GROUPs in a database independent manner.

I just described how to do that for MHSUM.

And under the heading *IMAGESQL Command Changes* he suggests:

11. An ALLOW command would give permission for an SQL GROUP to access a DB (with "ACCESS=" and "MODE=" parameters). There would also be a DISALLOW command.

With MHSUM I don't need new ALLOW and DISALLOW commands: for ALLOW I use the ISQL command:

```
GRANT <function> ON
<owner>.<table/view> TO
<authorizationgroup>;
```

And for DISALLOW I use the ISQL command:

```
REVOKE ALL ON
<owner>.<table/view> FROM
<authorizationgroup>;
```

15. A DISPLAY GROUP command would permit the DBA to see the names of any and all usergroups along with their states.

I don't need a DISPLAY GROUP command for MHSUM. Instead, I invoke ISQL and issue the command:

```
SELECT * FROM SYSTEM.TABAUTH;
```

(or some variation thereof). Better yet, this shows me users of native Allbase/SQL tables in addition to users of IMAGE/SQL tables!

16. The DISPLAY USERS command should permit display of the users within a single GROUP. Within each group the user names should appear in alphabetical order.

To do this for MHSUM, I issue the ISQL command:

```
SELECT * FROM SYSTEM.GROUP WHERE GROUPID
= '<authorizationgroup>';
```

And if I want the result in alphabetical order, then I just add an

```
ORDER BY USERID
```

clause to the query.

Fred raises an important issue that's independent of MHSUM's management techniques: that of wildcarded user identifiers. In *Migrating to IMAGE/SQL* Fred states:

There is no provision for the user names including MPE group names or for containing wild cards. The lady from the University of Notre Dame would LOVE such capabilities. All 16,000 of her users might require just a handful of ADD USER commands instead of 16,000.

And under the heading *Global Design Changes in IMAGE/SQL: Enhancements for the DBA* Fred suggests:

4. Provide username syntax of the forms:

[<jsname>,]<name>[.<account>
[,<group>]]

[<jsname>,]@[.<account>
[,<group>]]

Using the latter form permits the DBA to "add" and "delete" SQL users externally to IMAGE/SQL (and without needing to STOP the DBE) by using various MPE commands (:PURGEUSER and :NEWUSER, for example).

I agree that wildcarded user identifiers and fully qualified user identifiers should be supported; in fact I think that not only IMAGE/SQL, but Allbase/SQL and Allbase/NET should support this enhancement as well (and while we're at it, how about TurboIMAGE Remote Data Base Access files, too.) However I do have some reservations about the way Fred suggests these enhancements be implemented.

There's a conflict between MPE and Allbase/SQL regarding wildcard characters: for example, MPE considers "@" to be a wildcard character while Allbase/SQL considers "@" to be the separator between user name and account name. Wildcards must be resolved in a manner that is consistent with open systems, SQL standards, and MPE usage.

The MHSUM Trick

I promised to reveal the "trick" which allows me to manage MHSUM's security and users as though it were a native Allbase/SQL database; it's that the user class for read access to MHSUM is "0"—the user class to which no password applies or is needed.

The crux of the matter is that only TurboIMAGE databases which define the "0" user class as a valid reader and/or writer can be managed this way—of course the downside is that defining the "0" user class entails giving away the farm: anybody's got access. MHSUM was the first IMAGE/SQL database that I did anything more than dabble with, and having managed it this way for more than a year (without realizing that I was using a trick) I had naively assumed that any IMAGE/SQL database could be managed this way; imagine my shock and disappointment when I found out that MHSUM was a special case!

Having managed this IMAGE/SQL database as a native Allbase/SQL database for as long as I have, I'm now of the opinion that I ought to be able to manage all IMAGE/SQL databases this way—IMAGE/SQL user/security management then would merely be Allbase/SQL user/security management. That's why I described, at length, how I manage MHSUM as a native Allbase/SQL database and why I related my management practices to Fred's comments.

One might argue that implementing IMAGE/SQL user and security management within Allbase/SQL would require enabling some sort of proxy user capability (i.e.: that one or more Allbase/SQL user IDs would be mapped to an IMAGE/SQL user ID) and that database access information would consequently be lost. While the concern about losing accessor information is valid, I'm not sure how pertinent it is. Consider TurboIMAGE remote database access for a moment; there are two ways to access remote database: (1) establish a DSLINE then logon to the remote system as a user capable of accessing the remote database, or (2) establish an RDBA access file on the local system that maps local users to users on the remote system. Using either of these methods entails using a proxy user ID to access the remote database. So for that matter does using Allbase/NET (Allbase/SQL's version of Remote Data Base Access) to access a remote Allbase/SQL database: NETUTIL is used on the local system (the client) to configure an alias for the remote Allbase/SQL DBE, then NETUTIL is used on the remote system (the server) to map the client user ID to a server user ID.

I should note that Allbase/NET can be used in "loopback" mode—with the same node acting as both client and server. Conceivably I could use Allbase/NET as another avenue to managing IMAGE/SQL security and user management with minimal use of IMAGESQL; but doing so would only move the management mechanism from one awkward utility (IMAGESQL) to another one (NETUTIL) and at an unknown performance cost.

Other IMAGE/SQL and DBE Management Issues

In Migrating to IMAGE/SQL Fred points out that:

There are no documented procedures for obtaining information about DBEs and their attached databases/files. This makes it difficult for 1st or 3rd party vendors to provide DBE tools which are not pathologically dependent on IMAGE/SQL utilities.

I couldn't agree more. I'd like to see some of this kind of information available in Allbase/SQL system tables; that way I would be able to use existing relational access tools to review this information—after all, what better place is there to keep metadata than in the database itself? For information that does not properly belong in system tables I'd like to see documented and supported APIs provided so that access to the information could be obtained in a version-independent manner; the APIs could either be distributed as part of IMAGE/SQL and Allbase/SQL or they could be provided for a fee as an Architected Interface. I know this would make my IMAGE/SQL tool vendor (I use HICOMP's DBTune/SQL) happy, it would make me happy as well: I'd have a greater comfort level knowing that my tools were using HP provided APIs and I'd know that the tool vendors could invest more resources in enhancing their products functionality to meet my needs rather than in reverse engineering IMAGE/SQL and Allbase/SQL data structures just to ensure the tools work with the different database software versions.

Under the heading *Global Design Changes in IMAGE/SQL: Enhancements for the DBA* Fred suggests:

1. Permit a DB to be attached to at most one DBE and allow "remote DBE access". That is, allow users con-

nected to one DBE to access DBs in another DBE (even if the other DBE is on a different hardware system).

While I don't like to contemplate the management complications and synchronization risks that arise when a TurboIMAGE database is attached to multiple DBEs, I question the feasibility of this suggestion. Fred proposes more than simple remote database access—he proposes that HP provide what is called (I think) "star" access. "Star" access provides the ability to do things like perform joins of tables that, rather than being in different databases within the same DBE, are in different DBEs (and perhaps on different machines). This seem like an expensive product to "give away" as part of IMAGE/SQL. I also wonder whether executing a query against multiple DBEs would consume many more system resources than executing the same query against a single DBE.

Under the heading *IMAGESQL Command Changes* in the same article Fred suggests:

3. A "MAP" command should be provided which replaces the current UPDATE TYPE and SPLIT commands which should be obsoleted.

The MAP command operates at the DB level (not at the dataset level) and is performed only while the DB is not attached to a DBE. The resulting metadata is stored in the TC-file where it is available to all IMAGE/SQL software which deals with view creation.

Note 1: If DBAs wish to map differently fields of two or more datasets defined by the same dataitem, they will be forced to add new dataitems and rename some of the fields so that all fields defined by the same dataitem are mapped in the same manner.

While I agree that like-named fields should have the same format in different datasets, I also know that such is not always the case. Furthermore, sometimes the poor practice of using generic fields is outside the control of the DBA or a site's DP staff; for example, the field naming scheme may have been selected by the application software vendor. Mandating that mapping commands apply at the database level, with the only recourse being renaming fields in the TurboIMAGE database, is much too restrictive and will only serve to hinder IMAGE/SQL deployment and acceptance. I suggest that, when performing a mapping operation, the DBA should be able to specify whether the mapping is to apply to the dataset only or to the entire database.

I note that, as far as I know, there's nothing in Allbase/SQL that mandates that like-named columns in different tables of a single database must have the same format. There's also a more pernicious problem than data items having different formats in different datasets: a friend of mine suffers from generic fields which have different formats within the same dataset—he can't use IMAGE/SQL at all for those datasets.

9. The DETACH command should not "drop" all of the metadata from the DBE. The name of the DB and its creator and its OWNER should be retained. This reserves the OWNER name for subsequent reattachment purposes after DB maintenance is finished.

The fact is that at present a DETACH does not drop all of the information from the DBE. For example, during an ATTACH IMAGE/SQL creates entries in (at least) these tables: SYSTEM.TABLE, SYSTEM.GROUP, SYSTEM.TABAUTH, SYSTEM.COLUMN, and SYSTEM.VIEWDEF. However after a DETACH I have found there are still entries in SYSTEM.GROUP. (I suppose one might justify this by reasoning that dependencies on the IMAGESQL-created authorization groups might have been defined and that it might therefore be dangerous to drop the SYSTEM.GROUP entries; I'm not certain whether or not that would be a bogus argument.)

Furthermore, let's say I have done an ATTACH, then an ADD USER, then a DETACH; as I noted above, after the DETACH there's still a SYSTEM.GROUP entry for that user. I've found that if I then re-ATTACH, a DISPLAY USER will not show that previously added user. IMAGESQL should be consistent about cleaning up after itself.

Also, IMAGESQL will create a set of views based on the TurboIMAGE user class the first time that user class is referenced in an ADD USER command; however when you DELETE USERs through IMAGESQL the corresponding views are not dropped from the DBE. Obviously, dropping the views when deleting a user shouldn't be unconditional—it should occur only when the last user in a given TurboIMAGE user class has been deleted. This is another instance of IMAGE SQL not doing a good job of cleaning up after itself. (But then, I've found that I can DROP a native Allbase/SQL table, but the associated views are not dropped).

Are you still with me? Managing the objects that are created by an ATTACH or ADD USER, but which are not deleted by a DETACH or DELETE USER, is certainly confusing and can be very messy; this issue extends beyond IMAGE/SQL objects to include native Allbase/SQL objects as well. One of the best examples of this is the task of deleting a native Allbase/SQL database from a DBE; that's such a painful, exacting, and precarious process that when I develop a new database I do so in a stand-alone DBE. I do this for the simple reason that it is easier and faster to delete the entire DBE and

start from scratch if I make a mistake than it is to issue the many REMOVEs, DROPs, REVOKEs and I forget what else (and I usually do manage to forget something, which invariably leaves me with a toasted DBE.) After having worked with TurboIMAGE for so long I'm used to thinking that if I mess up while defining the database I need only run DBUTIL, purge the database, tweak the schema, run DBSCHEMA to recreate the root file, and run DBUTIL to recreate the database. In Allbase/SQL, however, the root unit is the DBE, which can contain many databases; so I can't be impulsively blowing away DBEs unless there's nothing in them other than the database I'm working on. Allbase/SQL needs some kind of DROP ALL command to make deleting a database from a DBE a simple—and complete—task.

17. The functionality of SQLGEN and a few of the functions of ISQL and SQLUTIL should be available through IMAGESQL.

In addition to enhancing IMAGESQL, think about enhancing SQLGEN, ISQL, and SQLUTIL to handle IMAGE/SQL related functions as well.

Fred suggest that DBSCHEMA should be enhanced:

Since IMAGE/SQL is being sold as a product and TurboIMAGE is no longer being significantly enhanced, it seems to me that DBSCHEMA should be SQL-oriented rather than TurboIMAGE oriented, at least over time.

I'd like to see DBSCHEMA enhanced so that it can be run in the same "mode" as at present or in an "SQL compliant" mode. The SQL compliant mode would generate errors when it detects anything that would make an IMAGESQL ATTACH less than straightforward. Furthermore, I suggest that the SQL compliant mode be the default.

Fred has some ideas about ISQL as well, for example:

Currently the STOP DBE command stops and disables SQL access. It needs to be augmented to stop non-SQL access and to be able to stop either or both.

I'd like to see ISQL enhanced so that instead of being forced to stop the entire DBE, the DBA could choose to stop access to a particular database. Since databases are defined by their ownership this would be, in effect, a STOP OWNER command.

Finally, I've got a few suggestions that weren't prompted by Fred's articles:

1. IMAGESQL should allow a TurboIMAGE database to be partially attached to a DBE. The DBA would be able to specify which datasets he does, or does not, want to be attached. For example, is it necessary to attach automatic master data sets? It seems to me that attaching automatic masters as tables is like treating native Allbase/SQL indices as tables. Alternatively, automatic masters could be attached as indices rather than as tables; whoops, not indices—hash structures. (Note that judiciously configuring TurboIMAGE user classes might provide a way to "hide" automatic masters in IMAGE/SQL: just don't define the user class with access to the master. I really don't know if this is feasible—it might prevent the user from reading from or writing to a detail data sets associated with the automatic master; I'll have to try it out in my copious spare time.)

2. Allbase/SQL is schizophrenic with regards to case sensitivity; it shouldn't be. For example, in ISQL I can create a table using mixed case for the columns names, but if I then attempt to perform a LOAD FROM EXTERNAL, ISQL insists that the column names be entered in upper case. This is only one example —ISQL is fraught with this stuff.

3. Allbase/NET shouldn't access DBEs on the server just because one or more of the Allbase/NET servers are running; it shouldn't access a DBE until a process on the client requests access and it should release the DBE when the client process is finished.

In conclusion, in his articles Fred White offers many valuable observations and suggestions; I hope that, like me, those articles prompted you to think about defining, using, and managing IMAGE/SQL and Allbase/SQL databases. And I hope the perspectives and suggestions I've expressed in this article will keep you thinking. IMAGE/SQL and Allbase/SQL continue to evolve; and while HP has done a wonderful job of providing us with relational access to TurboIMAGE databases, there's still plenty of room for improvement. I urge you to play a part in the evolution of IMAGE/SQL and Allbase/SQL by letting your views be known; either to HP directly or though SIGIMAGE and SIG/ALLBASE.

SIGIMAGE Mission Statement

SIGIMAGE's mission is to provide a forum for fostering mutual help and cooperation among its members and by serving as a collective voice, represent and advocate the interests of its members to Hewlett-Packard. SIGIMAGE is dedicated to working with HP in furthering the capabilities of IMAGE for the continuing benefits of all IMAGE users.

Are you a member of SIGIMAGE?

Membership in SIGIMAGE (The INTEREX Special Interest Group for IMAGE/SQL Databases) is free.

You receive this SIGIMAGE Newsletter as a courtesy of INTEREX, the International Association of Hewlett-Packard Computer Users.

To ensure that you and your colleagues enjoy all the benefits of membership, contact INTEREX for further information:

INTEREX
1192 Borregas Avenue
Sunnyvale, CA 94088-3439
U.S.A.

Telephone +1 (408) 747-0227
Fax +1 (408) 747-0947
E-mail: membership@interex.org

Copyright information

Editorial contact

F. Alfredo Rego
Adager Corporation
The Adager Way
Sun Valley, ID 83353-3000
U.S.A.

Telephone +1 (208) 726-9100
Fax +1 (208) 726-8191
E-mail: alfredo@adager.com

Articles and authors in this issue

Computer-based training course "HP CBT: Using Access Tools in a Client/Server Environment"

Madeline Lombaerde
Hewlett-Packard
Cupertino, CA, U.S.A.

A test version of a computer-based training course "*HP CBT: Using Access Tools in a Client/Server Environment*" is now available on the HP3000 World-Wide Web Server.

This is a training course for TurboIMAGE/iX users who want to learn how to use IMAGE/SQL with PC client tools to access their TurboIMAGE/iX data. To find the HP CBT Web page, go to the HP3000 WWW page using the following URL: http://jazz.external.hp.com/

Follow the link for HP 3000 Related Training Materials, then follow the link for the HP CBT. You can get there directly by using this URL: http://jazz.external.hp.com/training/cbt/hpcbtidx.html

You can download the installation files for the test version of the HP CBT from the Web page. Also, you can take a look at the Web version of HP CBT Companion Guide to give you an idea of how the CBT works. We'd like you to give us feedback on the usefulness of the CBT. Just fill out and return the HP CBT feedback form to us either by fax or e-mail (see the feedback form file for details). Your feedback is important to us.

Please take note of the hardware requirements for running the CBT. It runs on a PC under Windows 3.1 and will run only in SVGA mode (800x600).