Inside Digital's Rdb 6.0: The Last Hurrah? PAGE 74

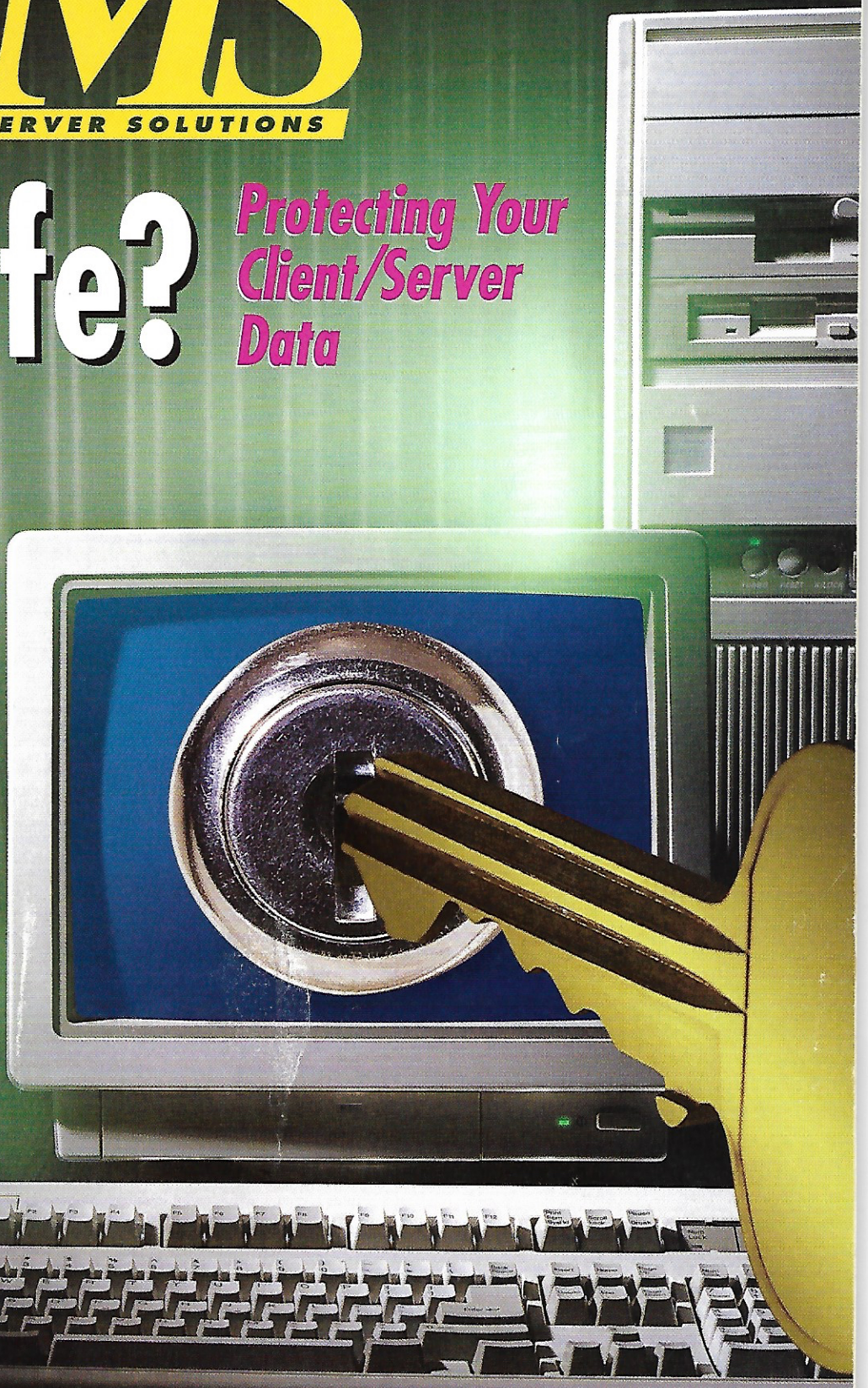
SEPTEMBER 1994
VOLUME 7 NUMBER 10

DBMS

DATABASE & CLIENT/SERVER SOLUTIONS

Is It Safe?

Protecting Your Client/Server Data



MISSION-CRITICAL VIEW
The Documentum Solution

INTERVIEW
David Kroenke's Semantic Object Modeling

DESKTOP DEVELOPER
Borland Database Update

A Miller Freeman Publication

$3.95 (Canada $4.95)
0 70989 35564 4

NEW AND REVIEWED
Skribe Professional 4.0, RaSQL, DataEdit 2.1

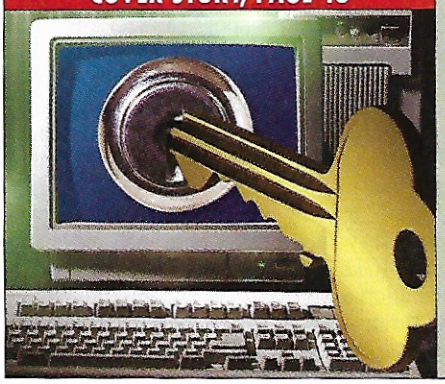
DBMS
DATABASE & CLIENT/SERVER SOLUTIONS

Database Security in a Client/Server World 48

by Steve Bobrowski

As database environments become more distributed, the chances of security violations increase dramatically. What's even more alarming is that many independent database software vendors are not addressing the special need for security in these distributed computing environments. This in-depth cover story looks at various security options, including data encryption, that can make your "leaky" client/server database environment airtight.

COVER STORY, PAGE 48



September 1994
Vol. 7 No. 10

Cover Art Direction by Barbara Licata

Cover Photography by Sandra Frank

Image Editing by Ron Licata

Inside Rdb 6.0 74

by Ken England

Although the industry continues to speculate on the future of Rdb, Digital Equipment Corp. recently released version 6.0 of its relational database management system. This comprehensive article discusses Rdb's newest and most important features, including support for stored procedures, outer joins, and asynchronous I/O.

The Competitive Spirit 84

edited by David M. Kalman

At its March CODA Conference, Oracle Corp. sponsored a Cooperative Development Environment (CDE) Programming Competition, in which developers had five hours to create a stock-trading application for tracking stock prices, managing customer profiles, and analyzing portfolios. This feature provides the exact specifications of the competition, as well as screen shots from the winning application.

PRODUCTS

Hands-On Reviews...28
Edited by Marjorie Thorne

Brio DataEdit 2.1, *an application development tool with fine features for simple, low-end environments.*

Skribe Professional 4.0, *a report writer that creates sophisticated reports, but is still a bit rough around the edges.*

RaSQL/B for FoxPro, *an apt solution for Xbase developers seeking access to alternative file systems.*

Soft Notes, *new client/server products and upgrades usher in the Fall season.*

COLUMNS

From the Editor
by David M. Kalman.....10
Pondering the industry's more imponderable conundrums.

Mission-Critical View
by Robin Bloor.....12
The ins and outs of workflow and document management systems.

SQL Explorer
by Joe Celko.....17
The debate rages about surrogate keys versus intelligent keys; plus a new SQL puzzle.

C/S Developer
by David Linthicum.....24
A look at the latest CASE links to PowerBuilder.

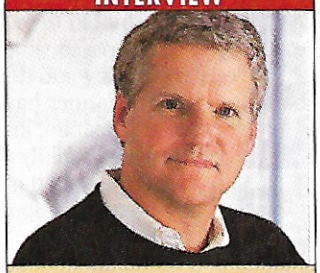
Server Side
by Steve Bobrowski.....97
How to estimate the space requirements for database objects in Oracle7.

Desktop DBMS
by Tom Spitzer.....109
Even with good technology, can Borland pull itself out of its downward spiral?

DEPARTMENTS

Letters.....4
Calendar.....44
Advertisers Index.....96
C/S Connection.....112

INTERVIEW



Wall Data Inc.'s David Kroenke 60
Wall Data's chief technologist explains how Semantic Object Modeling can turn everyday computer users into expert database designers.

COMING NEXT MONTH

A discussion of the state of relational technology (has it really come as far as vendors say?); and training issues in client/server environments.

DBMS (ISSN:1041-5173) is published monthly, except in June, which is semimonthly and contains the *DBMS* Buyer's Guide Issue, by Miller Freeman Inc., 411 Borel Ave., Suite 100, San Mateo, CA 94402. Second-class postage paid at San Mateo, CA, and additional offices. Direct requests for permission to Editor in Chief, *DBMS*, 411 Borel Ave., Suite 100, San Mateo, CA 94402. ***Subscriptions:*** For subscription inquiries, change of address, call: 800-334-8152 (U.S. and Canada), 619-745-2809 (all other countries). Or write *DBMS* at P.O. Box 469039, Escondido, CA 92046-9039; fax 415-905-2233. Please allow 4-6 weeks for an address change to take effect. Single copy price is $3.95 (Canada $4.50). Subscriptions are $24.95 for one year (13 issues); $43.95 for two years (26 issues) in the U.S. and its territories. Canada (GST# R124771239) and Mexico: $35.00 per year. All other foreign: $60.00 per year. Foreign orders must be prepaid in U.S. Dollars on a U.S. bank. ***Postmaster:*** Send address changes to: *DBMS*, P.O. Box 469039, Escondido, CA 92046-9039. ***Foreign Newsstand Distributor:*** Worldwide Media Service Inc., 115 E. 23rd St., New York, NY 10010; 212-420-0588.

■ FROM THE EDITOR ■

By David M. Kalman

Why Ask Why?

In the DBMS business, things happen for reasons that are not always clear. Why do small companies have the best ideas? Why does software get harder to use? Why do computers seem to run more slowly today than they did in 1987? Don't ask. Here are a few more imponderables for the month:

• Every product manager who demonstrates a new client/server application development tool feels compelled to spend at least 10 minutes out of every briefing comparing his or her tool to Powersoft's PowerBuilder. If 30 magazine editors and analysts each sit in 50 such briefings, Powersoft's competitors have drilled into the heads of the industry's opinion leaders 250 hours of expert testimony that Powersoft is the market leader.

• Graphical icons were intended to make software friendly, and easy to use. Now, Microsoft applications have Tool Tips — text labels that appear when you move the mouse pointer over icons. We used to call these "menus."

• At Database & Client/Server World (Boston, June 28 to 30), Alpha Software (Burlington, Mass.) previewed its Alpha 5 for Windows. Without breaking my nondisclosure agreement, I can say that Alpha 5 has the most clever, intuitive, and productive GUI implementation of any end-user database package I've seen. Alpha Software also deserves kudos for its Xbase language implementation — trademarked as "Xbasic." (Why didn't somebody think of that before?) As users become more comfortable with Alpha 5 for Windows, they can extend its power by writing Xbasic scripts.

■ CPUs keep getting faster, but applications keep getting slower.

• Gupta's SQLWindows 5.0 is easier than Microsoft Access 2.0 for designing multitable forms.

• XDB Systems' (Laurel, Md.) products continue to be "more compatible with DB2 than DB2." At Database & Client/Server World, the company announced XDB-Link Application Server for DRDA, which lets DB2 for MVS, DB2/VM (SQL/DS), or SQL/400 initiate dialogs to access and manipulate data stored on XDB-Servers running on LANs, desktops, or portable computers. IBM does not offer a comparable DRDA server.

• CPUs keep getting faster, but applications keep getting slower.

• At its Developer and User Conference (San Jose, Calif., July 12 to 14), ACI US Inc. (Cupertino, Calif.) demonstrated its 4D Universal platform-independent DBMS (database engine, language, and interface) for Macintosh, Windows (3.1, 4.0, and NT), Sun Solaris, and Unix. A true engineering marvel, this product was developed using what ACI calls the Virtual Machine Engine (VME), which consists of an engine and a collection of adapters for specific platforms. Instead of coding for specific machines, ACI's developers wrote 4D Universal for VME. At compile time, ACI replaces calls to VME with platform-specific calls. The company maintains only a single version of the 4D Universal C++ code; the platform-specific versions of 4D Universal are just "instances" of the virtual machine.

• Event-driven programming was intended to make software more flexible. Users could do things in any order by selecting non-modal windows and controls. Now we have Experts, Wizards, and Cue Cards, which guide users through modal, predefined steps. We used to call this "procedural programming." ■

DBMS

Publisher
Philip Chapnick

EDITORIAL

Editor in Chief David M. Kalman
Managing Editor Kathleen B. O'Connor
Art and Production Supervisor Barbara Licata
Associate Editor Marjorie Thorne
Associate Editor Theresa Rigney
Technical Illustrator Margaret Anderson
Contributing Editors Shaku Atre, Robin Bloor, Ron Carnahan, Joe Celko, Maurice Frank, David Linthicum, Stewart McKie, P. L. Olympia, Steve Roti, Tom Spitzer

PRODUCTION

Director of Production Andrew A. Mickus

ADVERTISING

Eastern United States Len Ganz
Account Representative Kimberly Tuells-DiMascio
Western United States Susan Dawson
Account Representative Sydnee Sanchez
Ad Coordinator Gae Reck

MARKETING

Marketing Account Manager Sue Singh
Marketing Production Designer Lynn LaRocca

CIRCULATION

VP/Director of Circulation Jerry Okabe
Group Circulation Manager Kathy Henry
Assistant Circulation Manager Phil Payton
Newsstand Manager Pam Santoro

Vice President/Group Director
Steve Schneiderman

Miller Freeman, Inc.
A MEMBER OF THE UNITED NEWSPAPERS GROUP

A publication of M&T Publishing, a division of Miller Freeman, Inc.

un A United Newspapers publication

Chairman of the Board Graham J.S. Wilson
President/CEO Marshall W. Freeman
Executive Vice President/COO Thomas L. Kemp,
Senior Vice President H. Verne Packer
Senior Vice President Donald A. Pazour
Senior Vice President Wini D. Ragus
Vice President/CFO Warren "Andy" Ambrose
Vice President/Administration Charles H. Benz
Vice President/Production Andrew A. Mickus
Vice President/Circulation Jerry Okabe

PRINTED IN THE USA

BPA INTERNATIONAL

For subscription orders or change of address: call 619-745-0685 or toll free 800-334-8152, write *DBMS*, P.O. Box 469039, Escondido, CA 92046, or contact our customer service department via CompuServe at 71572,341.

A New Generation of Rdb

ALTHOUGH ITS FUTURE IS UNCLEAR, RDB 6.0 LEAPS INTO THE DATABASE SERVER FRAY WITH SUPPORT FOR STORED PROCEDURES AND OTHER FUNCTIONS.

BY KEN ENGLAND

At the time of this printing, Digital Equipment Corp. (DEC) still owned and developed Rdb, its 10-year-old relational database server. However, Digital has hinted that it is looking for a buyer for the database, and rumors abound as to which company is in the running (most rumors point to Oracle). Even so, Digital recently released the newest incarnation of the database, version 6.0. This article takes a look at its most important new features.

This year marks the 10th birthday of Rdb, which Digital launched in 1984 on its VAX/VMS operating system. Now, 10 years later, Rdb runs on Digital's OpenVMS operating system, and the company is also porting it to DEC OSF/1 and Microsoft Windows NT. These ports will initially be exclusive to Digital's Alpha AXP RISC hardware platforms, but I expect that ports to other vendors' hardware will soon follow.

In this respect, Rdb will be walking a path similar to IBM's DB2/MVS, which is now available as DB2/2 on OS/2 and DB2/6000 on AIX, and will also be available on other vendors' hardware platforms in the future. Of course, there are many similarities between Rdb and DB2/MVS: Both are the flagship relational database products of major hardware vendors, both have been highly integrated with the underlying operating systems and hardware architectures, and both have been criticized as being proprietary. Both products, however, also have excellent reputations for their speed and robustness.

Digital recently released version 6.0 of Rdb in April 1994, and this version is the major baseline that will be used for upcoming ports to other operating systems. Rdb provides a lot of functionality as well as many tuning and administration options, but prior to this release there were a number of notable gaps, such as the lack of stored procedure support. Rdb 6.0 plugs these gaps with new SQL capabilities, additional performance and tuning options, and database administration functions. I will look at the new SQL capabilities first and then discuss the other features.

Multistatement Procedures

Rdb has supported the SQL Module Language for a long time. This functionality lets a developer write modules of SQL code that consist of an arbitrary number of procedures, which, after compilation and linking, you can call from the host program with appropriate parameters. This approach has its advantages, espe-

Ken England is president of Database Technologies, a U.K.-based company specializing in database consulting and training. Ken is also the coauthor of Rdb/VMS — A Comprehensive Guide *and* Digital's CDD/Repository *(Butterworth-Heinemann). You can reach Ken via CompuServe at 100114,166 or via Internet at 100114.166@compuserve.com.*

LISTING 1

```
BEGIN
  IF :account_type = '7' THEN
    IF :account_balance < 0 THEN
      BEGIN
          SET :acct_desc = 'Not Good';
      END;
    END IF;
  ELSEIF :account_balance > 0 THEN
      BEGIN
          SET :acct_desc = 'Not Bad';
      END;
  ELSE
      BEGIN
          SET :acct_desc = 'Not Sure';
      END;
  END IF;
END;
```

cially considering that the SQL is held in its own module file and not mixed in with the 3GL host program. However, until recently, a major restriction was that a procedure could contain only a single SQL statement, such as an INSERT. To solve this problem, Rdb 6.0 provides powerful multistatement procedures, which are SQL Module Language procedures that are no longer restricted to a single SQL statement. In effect, Rdb provides SQL compound statements — in other words, BEGIN...END blocks. You can nest compound statements, which can contain any number of SQL statements. Also, Rdb 6.0 provides flow-control statements and variables to let developers write fairly complex multistatement procedures, including CASE, IF, LOOP, FOR, and various verbs to retrieve diagnostic information and trace the values of variables for diagnostic purposes.

The example in Listing 1 shows a multistatement procedure-code fragment using an IF statement.

In addition to SQL Module Language procedures, you can use multistatement procedures in embedded SQL, dynamic SQL, and interactive SQL. The multistatement procedures can be marked as being atomic or non-atomic. If a compound statement is marked as being atomic (in the event of a SQL verb inside it failing), any other SQL verb in that compound statement that has already successfully executed is rolled back and the compound statement as a whole is rolled back. If a compound statement is marked as being non-atomic (in the event of a SQL verb inside it failing), any other SQL verb in that compound statement that has already successfully executed is not rolled back.

Stored Procedures

Multistatement procedures are useful but they really come into their own within another new Rdb feature — stored procedures. Stored procedures have been available in several database servers for a while. Through conversations with many Rdb customers, I found that they are most excited about the stored procedure feature. Basically, a stored procedure is a collection of SQL and flow-control statements that are held inside the database as a schema object. When you call a stored procedure, parameters are passed to it and received from it. The example in Listing 2 creates the stored procedure "Close_Branch." Note that a procedure is created as part of a module in Rdb 6.0.

There are three main advantages to using stored procedures. First, developers can encapsulate complex operations. They merely call the procedure, passing the appropriate parameters, without regard to the operations performed behind the scenes. Second, the definer of a stored procedure can specify that it runs with the privileges of that definer. The user of the stored procedure may not have permission to access the objects referenced by it, but if the definer has permission, the stored procedure will execute successfully. This is similar to writing privileged code whereby the user temporarily inherits privileges that are removed when the code finishes executing. Third, in a client/server environment, placing complex logic in a stored procedure on the server database results in simpler logic on the client, better maintainability, and less network traffic.

LISTING 2

```
CREATE MODULE Bank_Module LANGUAGE SQL
     PROCEDURE Close_Branch (:bcode CHAR(4));
       BEGIN
         UPDATE branch
           SET branch_desc = 'Closed Down'
            WHERE branch_code = :bcode;
       END;
END MODULE;
```

External Functions

For those developers who find Rdb's built-in functions such as SUM and MAX too restrictive, external functions will let them call any piece of 3GL code from any place in SQL where a value expression can be used. This 3GL code can be homegrown or a built-in library routine that comes with OpenVMS. The example in Listing 3 shows two user-written functions, "Encrypt" and "Decrypt."

Because developers can now write their own functions, the only limitations to SQL are in the minds of the developers. Of course, portability tends to go out of the window at this point, but each developer has the freedom to make that choice.

Joins

Rdb 6.0 now supports outer joins. Previously, to produce the same effect as an outer join, you had to execute multiple SELECTs and then concatenate those results with a UNION operator. Rdb 6.0 supports left, right, and full outer joins. Listing 4 is an example of a left outer join and its results.

In a left outer join, rows from the left table are preserved if there are no matching rows in the right table. The result of the operation is rows from the left table joined to matching rows of the right table, and rows from the left table joined to nulled columns of the right table.

The right outer join is the mirror image of the left outer join with rows from the right table preserved if there are no

LISTING 3

```
INSERT INTO Very_Special_Accounts(account_num, account_name)
     VALUES ('99886543', Encrypt('Lord Charles Richfellow'));

SELECT account_num, Decrypt(account_name) FROM
Very_Special_Accounts;
```

LISTING 4

```
SELECT c.customer_num, c.surname, a.account_balance FROM
   customer c LEFT OUTER JOIN account a
   ON c.customer_num = a.customer_num;

C.CUSTOMER_NUM    C.SURNAME     A.ACCOUNT_BALANCE
1111111111        England              10000
5566443399        Hobbs                NULL
2222876222        Hagan                70000
2200222222        Carpenter            16600
5678906222        Horn                 12200
5 rows selected
```

matching rows in the left table. The full outer join is a combination of the left and right outer joins.

Asynchronous I/O

In previous versions of Rdb, I/O operations to disk were mostly synchronous. This meant that the server process always waited for a read or write to complete before it could continue processing. With version 6.0, this is no longer the case. With an asynchronous I/O operation, the server process can continue processing without waiting for a read or write to complete. This means that for operations that are sequentially scanning a table, Rdb can now perform an asynchronous prefetch of database pages. In other words, while Rdb is processing the pages it has just read into its buffer, other pages that it suspects will be needed next are fetched from disk. In this way, the server process never has to wait for pages to be read.

The server can also perform write operations asynchronously. A batch write function will write modified pages in the database buffer pool to disk in the background when it reaches certain thresholds. This avoids the server process delay that can occur when a database page must be written back to disk in order to make room for a new page.

Query Outlines

Rdb 6.0 provides one of the most sophisticated query optimizers of any relational database in the marketplace today. When a user submits a query to the server, the optimizer calculates the best strategy with which to execute the query. It will take into account the query predicates, the number of rows in the affected tables, the number of unique values in the indexes, and other factors.

Generally, a query optimizer will try to choose the most efficient strategy, but occasionally it may choose one that is not optimal. In fact, DBAs may decide that a strategy using an alternative set of indexes might be more efficient for their needs. Prior to version 6.0, it was difficult (if not impossible) to force the Rdb query optimizer to use a different strategy. Now a DBA can specify the complete or partial strategy that the optimizer should follow to execute a specific query. The DBA does this by specifying a query outline.

■ Rdb 6.0 provides one of the most sophisticated query optimizers of any relational database in the marketplace today.

Within a query outline, a DBA can specify: a join order, in which tables and views are joined; a join method, which is the mechanism the query optimizer uses to relate records in a join; and an access path, which is the method the optimizer uses to retrieve rows from the table, such as sequential reading or indexed access via a particular index. DBAs can also define a query outline to have a mandatory or optional compliance level. A mandatory compliance level means that the query outline must be followed to the letter; if Rdb cannot do this, an error will result. An optional compliance means that Rdb may try to follow the query outline, but it doesn't matter if it cannot do so exactly.

By setting a logical name to a specific value, the default query execution strategy generated by the query optimizer is captured in a file. This file is then edited by the DBA to apply local changes, and it is stored in the database system tables. When a user asks Rdb to execute the query again, the server realizes that a query outline already exists, so it retrieves and executes it. A typical example of a query outline is shown in Listing 5.

Note that the strange 'id' string is a unique tag by which DEC Rdb can relate the query outline to the query.

LISTING 5

```
create outline ACCOUNTS_IN_BRANCHES
            id 'B426956C2BC632FD14A284D6FDF89988'
            mode -1
            as (
              query (
                subquery (
                  BRANCHES 0  access path index    BRANCH_NAME
                    join by cross to
                  ACCOUNTS 1  access path index ACCOUNT_BRANCH_CODE
                        )
                  )
              )
            compliance optional
            execution options ( total time );
```

Index Enhancements

Rdb supports sorted and hash indexes. The sorted index is a B-tree structure consisting of a tree of index node records with a leaf level that contains database key pointers to each row in the table. Sorted indexes are good general-purpose indexes that support exact-match retrieval using the equality operator (=). They can also support range retrieval using operators such as >, <, and BETWEEN, as well as partial key retrieval.

The hash index is a structure that enables a table row to be stored and retrieved in one disk I/O. It employs a hashing algorithm that takes the key value and converts it into a database page number. Hash indexes support only the equality operator and cannot be used for range or partial key retrieval.

Rdb 6.0 has improved indexing. First, the DBA can turn on run-length compression when creating an index. Sorted indexes can use up a large amount of disk space, so any compression mechanism is usually welcome. With run-length compression, a text key can have its space characters compressed, or a numeric key can have its binary zeros compressed. For example, a 20-character key column containing the name:

```
Dan⊕de⊕Lion⊕⊕⊕⊕⊕⊕⊕⊕⊕
```

(where ⊕ represents a space character) can be represented by:

```
Dan⊕de⊕Lion⊕⊕⊕
```

(assuming you choose a run-length compression of two).

Choosing a run-length compression of two means that Rdb compresses each sequence of two or more space characters, replacing the number of space characters by the minimum run-length value plus an extra byte that contains information about the compression (in our example, two plus one). This can significantly reduce the disk space used by the index and, because the keys are compressed, fewer index node records are used, resulting in fewer index levels and therefore fewer disk I/Os to retrieve a table row.

Rdb 6.0 also provides enhanced hash indexes. Prior to Rdb 6.0, one hash algorithm, known as "hashed scattered," randomly placed table rows across a database storage area. The result was a random distribution of rows in which a database page in the storage area could end up holding any number of rows.

However, you may now choose an alternative algorithm known as "hashed or-

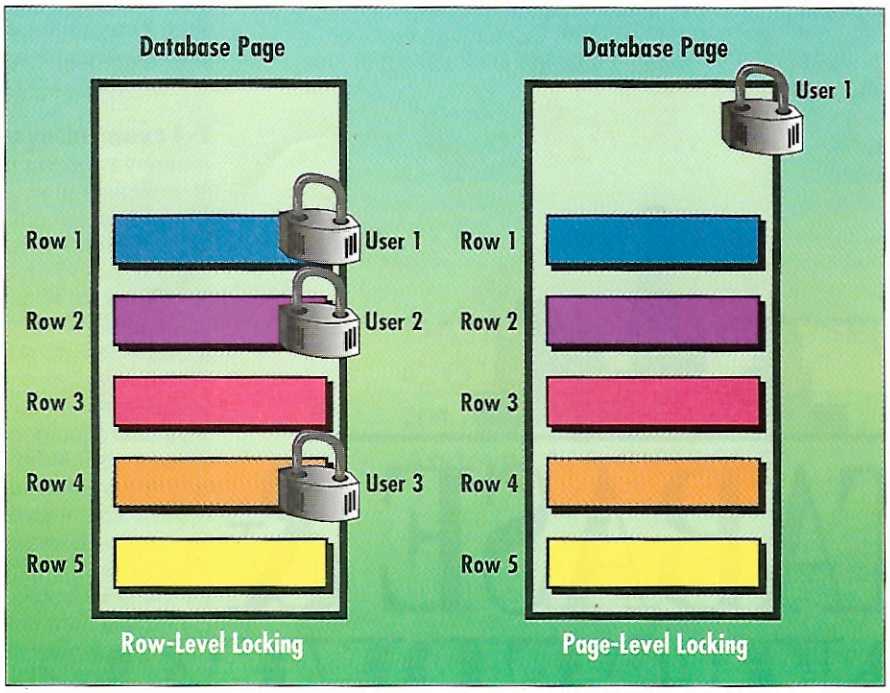


FIGURE 1 **A database page is subject to row-level and page-level locking. In some instances, locking at the database page level can improve performance.**

dered." This algorithm is ideal for data that has key values uniformly distributed across a range. This approach results in a more uniform distribution where database pages in the storage area will generally end up holding the same number of rows. Therefore, page sizing becomes simpler and hash collisions are avoided.

Page- and Row-Level Locking

Row-level locking lets different users lock individual table rows on the same database page without conflict. Many RDBMSs on the market today do not provide row-level locking, but vendors are rushing to implement it. However, there are advantages and disadvantages to row-level locking. On one hand, it provides for a greater level of concurrency, but, on the other, it can use a great deal of resources when locking many individual rows.

Rdb is unusual in that it has provided row-level locking since version 1.0, in addition to table-level locking. Recognizing that the row is not always the most efficient level of granularity at which to lock for some applications, Rdb 6.0 offers the DBA the option to lock at the database page level for specified tables. This approach can provide better performance in some instances. Figure 1 shows a database page subject to row- and page-level locking.

Circular After-Image Journals

Rdb 6.0 has also brought a number of changes into the database administration arena, especially in the area of after-image journaling. Rdb has always used multiple before-image journal files but only a single after-image journal file. Using a

single file can cause problems, however, if the disk containing the after-image journal file fills such that that after-image journal file cannot extend, and therefore users cannot access the database.

Rdb 6.0 has introduced the concept of multiple fixed-size journal files. As shown in Figure 2, a DBA can specify that a number of after-image journal files will be created on different disks. The system writes to only one of these journal files at a time, and, when it is full, the system journals to the second journal file and so on. When a journal file is full, you can automatically or manually back it up. In this way, you avoid the problem of disk space, and the use of multiple after-image journal files makes for a much more resilient database application.

Recovery By Database Page

In the move to very large database support, restoring an entire database when it is corrupted becomes impractical —it takes too long! Rdb has always supported the restoration of individual database storage areas, but, for large databases, this strategy may not be feasible. Rdb 6.0 helps to solve this problem by letting you restore an individual database page from a backup file and then roll it forward. One single corrupt database page can therefore be replaced in a multi-gigabyte database.

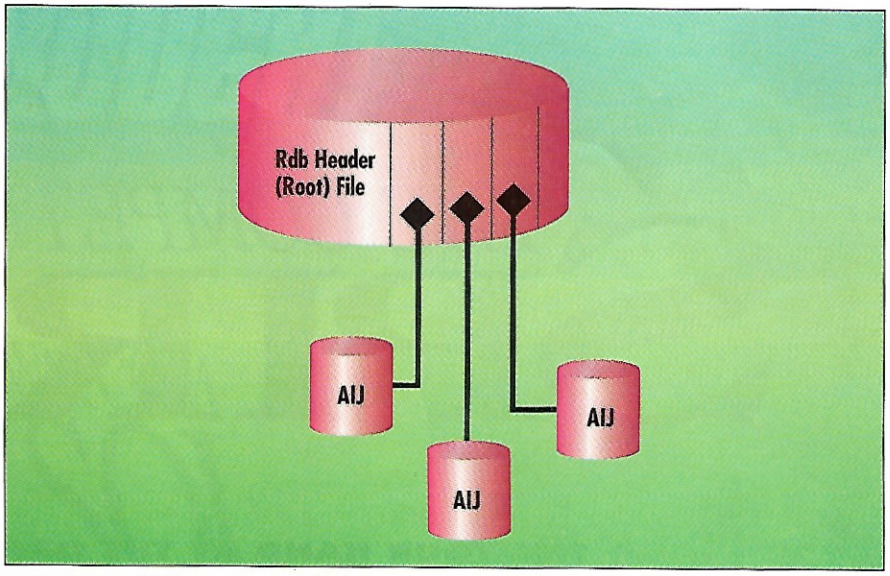


FIGURE 2 DBAs can specify that a number of after-image journal files will be created on different disks. The system writes to only one of these journal files at a time.

A Step Forward

DEC Rdb has always been an RDBMS with a high level of functionality, security, distributed capability, and robustness. Rdb is supported by many front-end tools (some via Microsoft ODBC), and it provides a number of gateways to most foreign databases.

Version 6.0 takes a great step forward with many new features, and it is now difficult to point to any major weaknesses in the product. Rdb's portability is still low, but we can hope to look forward to this changing over the next year. ■

• Digital Equipment Corp., Maynard, MA 01754-2571; 800-344-4825, 603-884-4825, or fax 603-884-1571.

ILLUSTRATIONS BY MARGARET ANDERSON