A PEDAGOGICAL COMPARISON OF TWO DATABASE MANAGEMENT SYSTEMS

by George Wright
Laurette Poulos Simmons
Loyola College
4501 North Charles Street
Baltimore, MD 21210
301/323-1010

ABSTRACT: Database course objectives typically include introduction of the major database models, logical and conceptual database design to meet management needs, and various technical aspects involved in the database approach. In selecting a database management system to meet these objectives, many factors must be taken into consideration. Cost and popularity are important as is the ability of the product to create structures and illustrate concepts that are in the course textbook. In this research, we look at the functionality of dBase III+ and PC-ORACLE. Their ability to illustrate concepts found in popular texts to meet course objectives is examined.

Some of the important concepts that appear in database textbooks include data dictionary, data independence, data manipulation language, display mechanisms, high level language interface, data integrity, security and recovery, and structures or organizational models: networks, hierarchies, and relations. Understanding of these concepts and structures is enhanced by hands-on experience with them.

While no database product is an ideal pedagogical tool for all database courses, some products have advantages over others.. This research compares dBase and ORACLE. The result can be generalized to similar products.

KEYWORDS: Database, Database Management System, DBMS, dBaseIII+, Database Pedagogy, ORACLE, Relational DBMS

INTRODUCTION

In this paper we discuss a dozen major database topics. For each, we state definitions and importance in a database course.

We then compare how well dBase III+ and ORACLE accommodate or illustrate the topic. Order is arbitrarily alphabetic. Unless otherwise noted, statements about dBase and ORACLE capabilities are taken from the manuals that accompany the products.

Our choice of dBase and ORACLE is pragmatic. Both are available to us. There are other reasons why treatment of these two software products is worthwhile. In spite of intense competition at the low end of the microcomputer DBMS market, dBase still retains about a 50% market share. In the last half of 1987, dBase III+ was the fourth-largest selling program, in dollars, through computer stores. (It was bested by Lotus 1-2-3, Display-Write 4, and WordPerfect.) [10,p. 95]

ORACLE is another interesting product. Available in common format for the DOS, Unix/Xenix, and VM operating systems, ORACLE is being marketed aggressively. Inexpensive DOS and Xenix versions are available for micros. Free (albeit unsupported) VM versions are often donated to educational institutions.
In all that follows, we use the somewhat idiosyncratic spelling used in both products' documentation: "dBase" and "ORACLE."

DATA DICTIONARY

Marketing demands that all commercial database products include -- or at least claim to include -- a "data dictionary." And no wonder -- the data dictionary is the cornerstone of every modern DBMS. [8, p. 382] The data dictionary is a software product that, at minimum, maintains

information about the database and its contents. More elaborate data dictionaries provide for automatic creation and recompilation of schemas. [8, p. 221] The difference between dBase and ORACLE dictionaries reflects their histories. DBase descends from a single-user PC environment. Consequently its files are small and self-contained. Data dictionary information is correspondingly sparse. ORACLE comes from a multi-user minicomputer environment. Its files are contained in a single, large, contiguous, master file. A more elaborate data dictionary is therefore required.

We believe that ORACLE's approach comes closest to the active data dictionaries that students will encounter in business applications. DBase simply stores field descriptors in its database files. ORACLE presents an extensively documented, "full service" data dictionary.

The DBase Data Dictionary

DBase stores database content information in each of its .DBF files. Each .DBF file contains a header having structure information in binary format. Information on the structure of each file and the number of data records it contains as well as the date of the last update is readily available to the user. The dBase data dictionary capability is rudimentary. The information is there to support dBase functions, but the data dictionary provides little insight into the functionality offered by more powerful dictionaries.

DBase also has a catalog feature. The dBase .CAT file combines database files, index files, format files, form files, and label files into a related set. In some ways, this is analogous to the user views provided by more sophisticated data dictionaries. The catalog feature is invoked through the SETCATALOG command. Once the catalog is "on", any requests for database files, for example, result in access to only those files that are within the requested domain.

The ORACLE Data Dictionary

The ORACLE data dictionary capability is closer to the concept as covered in database texts. The ORACLE dictionary is a set of tables and views, installed during initial ORACLE initialization. The ORACLE DD records the names and descriptions of users, tables, and views. It also stores information about user privileges and data storage. ORACLE Data Dictionary files can be queried through regular SQL SELECT statements.

DATA INDEPENDENCE

Data independence is the separation of data storage details from applications programs that use the data. [9, p. 475] It contributes to the maintainability of both data and related applications.

Both dBase and ORACLE provide some -- but not complete--data independence. "Field type" is stored in the data dictionary and is available to applications. Applications can access fields without worrying about whether they are character strings, numerics, dates, etc. On the other hand, elaborate, customized report routines can be written which incorporate knowledge of field type and format. If the underlying data is changed, screens, report generators, etc., must each be updated to accommodate the change as required. In this respect, neither dBase nor ORACLE provides as much data independence support as some large mainframe database management systems.

DEADLOCK

Deadlock occurs when two or more concurrently running programs need exclusive control over the same file. [2, p. 228] This concept is important in any discussion of shared and distributed databases. Of course, both dBase and ORACLE can avoid deadlock in a networked environment. By write-protecting the network file server, network users are forced to use local disks for all data storage. Only dBase or ORACLE program files are available from the server. Consequently deadlock on data files is completely obviated while network software handles contention for program files.

When data files are available simultaneously to multiple users, the following may happen.

1. User U1 locks file F1.
2. User U2 locks file F2.
3. User U2 tries to lock F1 and waits, pending a successful lock of F1.
4. User U1 tries to lock F2 and waits, pending a successful lock of F2.

This is deadlock or "deadly embrace."

From a teaching point of view, both dBase and ORACLE illustrate the deadlock problem and approaches for dealing with it. From a practical point of view, multi-user configurations can be arranged to obviate deadlock completely. If such configuration limits are undesirable, ORACLE's automatic approach to deadlock resolution is the more practical.

DBase Deadlock-Avoidance Features

DBase has all the features necessary to produce the described deadlock situation but leaves it to a programmer to make sure that when multiple users are accessing the same files, a file deadlock will not occur. To quote from the dBase documentation:

> "It is the programmer's responsibility to make sure that, when multiple users are serially accessing the same files, a program sequence that includes [code which sets up a potential deadlock] will not result in a file deadlock." [1, p. N4-9]

ORACLE Deadlock-Avoidance Features

ORACLE contains a set of true locking features which minimizes the likelihood of conflicts and deadlocks, both at the table and row levels (or the file and record level, in dBase terminology). But, these locking features, as with dBase, still permit deadlocks. Unlike dBase,

detecting and resolving deadlocks. When a deadlock occurs, ORACLE aborts one of the commands or command procedures that is causing the deadlock. (Which process is aborted is unpredictable.)

ORACLE Lock Interaction

	If one user has:		
Another user may:	SHARE LOCK	UPDATE LOCK	EXCLUSIVE LOCK
Establish SHARE lock	yes	no	no
Establish SHARE UPDATE lock	no	yes	no
Establish EXCLUSIVE lock	no	no	no
Query the table	yes	yes	yes
Modify the table	no	yes*	no

*Provided the other user places a SHARE UPDATE lock on a different row. Once either user performs an update, the other cannot update any row until the first user commits. [4, p.20-5

ORACLE has built-in provisions for

DISPLAY MECHANISMS

Current DBMSs feature various display mechanisms:

1. There is the interactive screen display, as if the screenwere a glass teletype. This mechanism usually features the ability to echo the screen display to a disk file for later display and modification.

2. There is a windowing display, as if the screen were a window into the database file. The view may be restricted to certain subsets of the file. Cursor motion keys function to page the viewing window over the database contents.

3. There is the report generator. A facility is available to specify the contents of a hardcopy report. The ultimate result is a printed report with little or no corresponding screen display.

Both dBase and ORACLE offer all three of these display mechanisms. We prefer the ORACLE interactive display because of its SQL compatibility. The dBase windowing display is much more convenient because it is automatic, yet can be completely customized if desired. There are some advantages to dBase report generation because of the rich programming capabilities. Yet the reporting functions available through the user interface (ASSIST) mode of dBase are not very powerful. After-market products (such as R&R Relational Report Writer by Concentric Data Systems) exist to automate the creation of customized reporting using dBase files.

Report generation under ORACLE is less automated than under dBase. Embedded formatting commands must be placed in the data stream for later interpretation by the ORACLE report formatter. The complexity of the output reports is limited by the user or programmer's SQL expertise and the capabilities of the formatter.

HIGH-LEVEL LANGUAGE INTERFACE

A high-level language (HLL), also called a higher-order language, is any of the compiled computer languages: FORTRAN, COBOL, C, Pascal, etc. A high-level language interface (HLLI) allows interaction between the database and compiled code.

A full HLLI is a two-way street. First, to the database the HLLI provides syntax for accessing routines written by the user in a high-level language. The object, of course, is to deal with situations beyond the capabilities of the native database syntax. Second, to the HLL the HLLI provides routines, usually in the form of object libraries, allowing HLL calls to database data and index files. The object is to allow programmers to achieve highly optimized or unusual access to the database through the HLL.

Both dBase and ORACLE offer HLLI features. The dBase interface is available only in the DBMS-to-HLL direction; ORACLE is available only for HLL-to-DBMS. In practice, both ultimately offer the same degree of practical functionality.

ORACLE's HLLI will always require more programming skill on the part of the user, as well as the appropriate language compiler environment. C language object libraries are bundled with PC-ORACLE. Fortran, Pascal, PL/1 and Cobol pre-compilers are also available.

INTEGRITY

There are several aspects of integrity addressed in database texts.

1. T*ransaction integrity:* ensuring that operations of a transaction either are performed in their entirety or are not performed at all. [3, p. 10]. We consider this aspect below under the heading ROLLBACK/ RECOVERY.

2. E*ntity integrity:* a constraint on a database relation stating that primary keys must be unique and no part of a primary key can be null.

3. R*eferential integrity:* a constraint on a database relation stating that all attributes which refer to primary

keys in another relation (i.e. foreign keys) must reference existing primary key values in that other relation or have null values. [6,pp.354, 368]

The entity integrity constraint ensures that we can't insert a new record into a relational database with the same primary key value as an existing record. The referential integrity constraint ensures that there is a parent record for each child record. [2, pp. 333]

ORACLE has facilities for ensuring entity integrity. It refuses to accept a duplicate entry (providing an index was created with the 'uNIQUE' operator). Neither dBase nor ORACLE stores sufficient information to allow automatic enforcement of referential integrity.

MANIPULATION LANGUAGE (QUERY AND UPDATE)

Textbook discussions of data manipulation languages (DMLs) consider an important aspect to be syntax used with HLLs for data access and update. [5, p. 9; 9, p. 110] We covered that aspect above under the heading HIGH LEVEL LANGUAGE INTERFACE. Here we consider manipulation language (ML) capabilities simply to be a non-procedural language for data handling. The ML is meant to provide ease of use and data independence.

The MLs of dBase and ORACLE are quite different. The dBase ML is its own command language, while ORACLE uses SQL. DBase's is record-oriented; ORACLE's is table-oriented. ORACLE's use of SQL as its ML makes it more appealing to us for classroom use.

DBase Manipulation Language

DBase has a rich syntax of procedures, functions, and programming constructs. Its documentation is extensive, with frequent examples of code. DBase's ML syntax has become a microcomputer standard. Several competing database products advertise dBase syntax compatibility. Others claim to feature supersets of the dBase ML. There is a host of after-market software products which assume the dBase ML standard.

The only problem is that the dBase standard is idiosyncratic. Its ML is similar to but not exactly like other relational DBMS programming languages. For example, dBase applications can be ported to run under Informix or Ingres, but only by translation of the source code to the new host's syntax.

The new release of dBase even includes SQL as a manipulation option. However, our experience with a Beta release of dBase IV indicates that the SQL interface is not smoothly integrated.

Manipulation Language Paradigms

dBase

```
DO WHILE .NOT. EOF()
                *Process current record
                SKIP
;ENDDO
```

ORACLE

```
SELECT FIELD FROM TABLE
WHERE FIELD = VALUE
```

ORACLE Manipulation Language

ORACLE uses SQL -- primarily a query language -- as its manipulation language. There are some advantages to this. SQL, largely due to its IBM mainframe heritage, is more standardized and widespread. Students are far more likely to encounter SQL in large or distributed, mainframe-supported databases. Moreover, SQL seems to be the "wave of the future." The new release of dBase even includes SQL as a manipulation option. However, our experience with a Beta release of dBase IV indicates that the SQL interface is not smoothly integrated.

There are differences between a record-oriented ML, such as dBase, and a set (SQL)-oriented ML, such as ORACLE. The dBase ML is built on the single-thread program paradigm. It is rich with structured programming constructs. The ORACLE ML is built on the table paradigm. It is rich with data query and selection functionality. It has few programming constructs. A practical implication is that conversion of dBase applications to ORACLE applications is not at all a straightforward translation task. On the other hand, it may well be that the tabular orientation is more appealing to students more familiar with spreadsheets than with programming.

MANY-TO-MANY/ONE-TO-MANY RELATIONSHIPS

The many-to-many relationship is a common one in database applications: a part can be stored in many warehouses, a warehouse can stock many parts; an engineer can work on many projects, a project may occupy many engineers; a student may attend many classes, a class may contain many students; etc. Such situations can be handled without redundancy by means of a junction record. The junction record contains a concatenated key and the intersection data of the records concerned. Alternatively a foreign key and redundant (non-normalized) data can be used to reflect this many-to-many relationship.

Resolution of many-to-many relationships into two one-to-many relationships by means of junction records is a generic topic. That is, it is addressed the same way regardless of database model. Any DBMS that can interrelate files by means of keys or pointers can illustrate the concept. Both dBase and ORACLE serve equally well.

NETWORK AND SECURITY ACCOMMODATION

A common approach to campus computing involves a network of individual PCs, most often with one or more PCs devoted to network management and

file service. Under networking, more care must be paid to security and concurrency. We've discussed concurrency above under the heading DEADLOCK.

Both dBase and ORACLE serve to illustrate the features a DBMS should have in order to be secure in a multi-user or distributed environment. Both feature login security, file security, and field access security. DBase additionally provides data encryption and decryption.

ORGANIZATION MODELS

Standard DBMS models are the hierarchical, the network, and the relational models. Both dBase and ORACLE are considered to be relational DBMSs. They both serve to illustrate this model. Since neither can accommodate explicit pointers, neither can readily illustrate the network or hierarchical models. We know of only one PC-based network DBMS and one PC-based hierarchical DBMS. PC-Focus, an offspring of the Focus mainframe DBMS, uses the hierarchical approach. MDBS III, another PC DBMS, relies completely on the network model. [10, p. 94]

RECORD RETRIEVAL METHODS

Several competing techniques for retrieving records are usually covered in database courses: [9, pp. 36-51]

1. Key transformation or hashing: calculation of a record location on a mass storage device from the value of the data element used as the access key.

2. The index approach: a cross-reference list providing record location of a direct access storage device that has a specific value of a data element used as the access key.

3. The indexed sequential access method: a specific indexing technique providing some benefits of both direct and sequential access to records.

4. Binary search trees: storage method supporting rapid search for a specific record without use of an index.

5. B-tree indexing: storage method using a multi-way tree for record storage.

6. List processing: storage of records with embedded pointers to next, prior, or owner records (the basis of the network model).

DBase and ORACLE both employ only the index approach to record retrieval.

ORACLE far surpasses dBase in facilities to support transactional integrity.

ROLLBACK/RECOVERY

An important concept in any database course is that of a transaction. [7, p.224] A transaction is an atomic unit of work. That is, it is a sequence of operations either all of which or none of which must be performed. The standard example is a transfer of funds from a savings account to a checking account. Funds must first be withdrawn from one account and then added to the second. If only one of these to components of the transaction occurs, database integrity is lost.

ORACLE far surpasses dBase in facilities to support transactional integrity. ORACLE automatically provides the buffers, audit trails, journaling, and rollback characteristics required. [6,p. 164] DBase provides none of these. Transactional integrity enforcement is solely up to the dBase applications programmer.

CONCLUSION

The choice of DBMS software for use in a particular course depends on many things. In this discussion, we have focused on how two products accommodate topics that appear in database texts. For the range of topics discussed, ORACLE emerges as our choice. The SQL interface, the extensive data dictionary, and the rollback/recovery provisions furnish sufficient advantages to warrant serious consideration of ORACLE over dBase III+ as the software to be used to teach a database course. Ashton-Tate has recently released dBase IV, and Oracle Corporation is currently beta-testing ORACLE release 6. Release of both these products, and others, will ultimately require our re-consideration. But until other DBMSs achieve the market acceptance of the current versions of dBase and ORACLE, this comparison should prove useful.

REFERENCES

[1] Ashton_Tate. Programming with dBase III Plus. Torrance, CA: Ashton_Tate, 1986.

[2] Bradley, James. Introduction to Data Base Management in Business. New York: Holt, Rinehart and Winston, Inc., 1983.

[3] Ceri, Stefano, and Pelagatti, Giuseppe. Distributed Databases: Principles and Systems. New York: McGraw-Hill Book Company, 1984.

[4] Dimmick, Shelley and Fachs, Jonathan. ORACLE MANUALS. Belmont, CA: Oracle Corporation, 1986.

[5] Gaydasch, Alexander, Jr. Effective Database Management. Englewood Cliffs, NJ: Prentice Hall, Inc., 1988.

[6] Harrington, Jan L. Relational Database Management for Microcomputers: Design and Implementation. New York: Holt, Rinehart and Winston, Inc., 1988.

[7] Kroenke, David M. and Dolan, Kathleen A. Database Processing: Fundamentals, Design and Implementation. Chicago: Science Research Associates, Inc., 1988.

[8] McFadden, Fred, and Hoffer, Jeffrey A. Database Management. Menlo Park, CA: Benjamin Cummings, 1988.

[9] McNichols, Charles W., and Rushinek, Sara F. Data Base Management: A Microcomputer Approach. Englewood Cliffs, NJ: Prentice Hall, Inc., 1988.

AUTHORS' BIOGRAPHIES

An Assistant Professor of Management Information Systems at Loyola College in Maryland, George Wright is new to academia. His fifteen years' business experience includes database design and implementation, computer systems management, and operations research.

Laurette Poulos Simmons is an Assistant Professor of Management Information Systems at Loyola College Maryland. She has worked extensively in end user computing as a teacher, researcher, and consultant. Laurette's articles appear in a diverse range of journals including *International Journal of Forecasting, Journal of Computer Information Systems, Computers and Operations Research, and Journal of Business Forecasting.*

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096