STRATEGIC IMPORTANCE OF RELATIONAL DATABASE

Orland Larson
Hewlett-Packard Company
Cupertino, California

ABSTRACT

It is generally understood that the enterprises most likely to excel in the future will be those that have recognized the importance of managing information as a major resource. These companies will be competing on the basis of the accessibility, accuracy, and timeliness of their information. One of the most important technologies used to improve the productivity, speed and flexibility of information management systems is RELATIONAL DATABASE.

This paper begins by addressing the issues and concerns associated with current non-relational application development environments. This will be followed by a review of the motivation for relational and an overview of the relational approach including Structured Query Language (SQL). The things to consider when choosing relational will be addressed including a review of the characteristics of an application that would be best suited for relational. This will be followed by a summary of the relational approach and a discussion of the things to consider when preparing to "go relational". The proprietary and third-party relational products available on HP computer systems will then be discussed followed by a preview of the future trends in relational technology. Finally, this paper will summarize the strategic importance of relational database technology for your organization.

INTRODUCTION

Database management systems, which first became widely available in the early 1970's, are, for many users, the single most important piece of software they will ever own. The implementations of the network and hierarchical models have provided developers with the tools that are generally providing users with solutions to complex information management problems. However, there is a price to pay for using either of these nonrelational approaches. Because of the navigational nature if these two models, database designers spend a great deal of time predefining data relationships only to find that users' data requirements are changing dynamically. These changes in user requirements cause modifications to the database structure and, in most cases, the associated application programs. This results in an excessive amount of time maintaining applications that are inflexible and fail to meet user needs. In addition, the complexity of accessing data limits the productivity of the programmer who is forced to think and code at a low level of structural detail.

Business professionals are frustrated by the limited access to information that they know exists somewhere in the database. Their business environment is changing dynamically, and they feel MIS should keep up with these changes. They also lack powerful inquiry facilities to aid in the decision-making process, which would allow them to ask anything about any data residing in that database.

THE MOTIVATION FOR RELATIONAL

Dr. Edgar F. Codd, considered to be the originator of the relational model for databases, noted when presented the 1981 ACM Turing Award that the most important motivation for the research work resulting in the relational model was the objective of providing a sharp and clear boundary between the logical and physical aspects of data base management (including data base design, data retrieval, and data manipulation). This is called the data independence objective.

A second objective was to make the model structurally simple, so that all kinds of users and programmers could have a common understanding of the data, and could therefore communicate with one another about the database. This is called the communicability objective.

A third objective was to introduce high-level language concepts to enable users to express operations on large chunks of information at a time. This entailed providing a foundation for set-oriented processing (i.e., the ability to express in a single statement the processing of multiple sets of records at a time). This is called the set-processing objective.

Another primary motivation for development of the relational model has been to make data access more flexible. Because there are no pointers embedded with the data, relational programmers do not have to be concerned about following pre-defined access paths or navigating the database, which force them to think and code at a needlessly low level of structural detail.

RELATIONAL DATABASE DEFINED

The relational database model is the easiest to understand - at least at the most basic level. In this model, data are represented as a table, with each horizontal row representing a record and each vertical column representing one of the attributes, or fields, of the record. Users find it natural to organize and manipulate data stored in tables, having extensive familiarity with tables dating from elementary school.

The Table, or two dimensional array, in a "true" relational database is subject to some special constraints. First, no row can exactly duplicate any other row. (If it did, one of the rows would be unnecessary). Second, there must be an entry in at least one column or combination of columns that is unique for each row; the column heading for this column, or group of columns, is the "key" that identifies the table and serves as a marker for search operations. Third, there must be one and only one entry in each rowcolumn cell.

A fourth requirement, that the rows be in no particular order, is both a strength and a weakness of the relational model. Adding a new record can be thought of as adding a row at the bottom of the table; hence, there is no need to squeeze a new row in between preexisting rows as in other database structures. However, to find a particular row, the entire table may have to be searched.

There are three kinds of tables in the relational model: base tables, views, and result tables. A base table is named, defined in detail, filled with data, and is more or less a permanent structure in the database.

A view can be seen as a "window" into one or more tables. It consists of a row and/or column subset of one or more base tables. Data is not stored in a view, so a view is often referred to as a logical or virtual table. Only the definition of a view is stored in the database, and that view definition is then invoked whenever the view is referenced in a command. Views are convenient for limiting the picture a user or program has of the data, thereby simplifying both data security and data access.

A result table contains the data that results from a retrieval request. It has no name and generally has a brief existence. This kind of table is not stored in the database, but can be directed to an output device.

THE RELATIONAL LANGUAGE

The defacto industry standard language for relational databases is SQL. SQL is pronounced "SEQUEL" and stands for Structured Query Language. This name is deceiving in that it only describes one facet of SQL's capabilities. In addition to the inquiry or data retrieval operations, SQL also includes all the commands needed for data manipulation. The user only needs to learn four commands to handle all data retrieval and manipulation of a relational database. These four commands are: SELECT, UPDATE, DELETE and INSERT.

The relational model uses three primary operations to retrieve records from one or more tables: select, project and join. These operations are based on the mathematical theories that underlie relational technology, and they all use the same command, SELECT. The select operation retrieves a subset of rows, that meet certain criteria, from a table. The project operation retrieves specific columns from a table. The join operation combines data from two or more tables by matching values in one table against values in the other tables. For all rows that contain matching values, a result row is created by combining the columns from the tables, eliminating redundant columns.

The basic form of the SELECT command is:

```
SELECT    some data (column names)
FROM      some place (table names)
WHERE     search conditions (if any) are to be met
```

In some instances WHERE, may not be neccessary. Around this SELECT..FROM..WHERE structure, the user can place other SQL clauses in order to express the many powerful operations of the language like the GROUP BY clause or the HAVING clause.

In all uses of SQL, the user does not have to be concerned with how the system should get the data. Rather, the user tells the system what data is needed. This means that the user only needs to know the meaning of the data, not its physical representation, and this feature can relieve the user from many of the complexities of data access.

The data manipulation operations include UPDATE, DELETE and INSERT. The UPDATE command changes data values in all rows that meet the WHERE qualification; the DELETE command deletes all rows that meet the WHERE qualification; the INSERT command adds new rows to a table.

When retrieving data in application programs, it is important to remember that SQL retrieves sets of data rather than individual records and consequently requires different programming techniques. There are two options for presenting selected data to programs. If an array is established in the program, a BULK SELECT can retrieve the entire set of qualifying rows and store them in the array for programmatic processing. Alternatively, it is possible to activate a cursor that will present rows to programs one at a time.

SQL has a set of built-in, aggregate functions. Some of the functions available are COUNT, SUM, AVERAGE, MINIMUM, and MAXIMUM. They operate on a collection of values and produce a single value.

In addition to commands for data retrieval and modification, SQL also includes commands for defining all database objects. The data definition commands are CREATE, ALTER and DROP. The CREATE command is used to create base tables, views, indexes and authorization groups; the ALTER command provides for the expansion of existing tables; the DROP command deletes a table, view, index and group. One of the most powerful features of SQL is its dynamic data definition capability. This function allows the user to add tables, columns, views, indexes and groups to the database without unloading and reloading existing data or changing any current programs. More importantly, these changes can be made while the databases are in use.

WHEN TO CHOOSE RELATIONAL

The choice of the "correct" database management system must be based on the environment in which the database will be used and on the needs of the particular application. The key feature of relational technology is that it allows for maximum flexibility, and will probably be the choice for many new applications.

The relational approach should be selected when the application has a large number of data relationships or when the data relationships are unknown or changing dynamically. The relational approach provides the needed flexibility to establish relationships at the time of inquiry, not when the database is designed. If the application has unknown or incomplete data specifications, which is usually the case in a prototyping environment, then a relational database may be preferable. If the application requires a quick turnaround, the quick design and implementation capabilities of a relational database can be important. The ability to handle ad hoc requests is a definite strength of the relational model as is the ability to extract data for use in a modeling, forecasting, or analytical framework.

Some questions that should be asked when deciding whether or not to "go relational" are listed below.

* Does your company have an excessive backlog of applications to be developed, including an invisible backlog?

* Are your programmers spending too much time maintaining applications caused by changing data requirements or relationships?

* Are your programmers spending an excessive amount of time writing code to navigate through a nonrelational database?

* Do your users' requirements for information change dynamically?

* Do your users feel restricted by a nonrelational database?

* Would your users find it natural to organize and manipulate data in tables? For example, are they currently using spreadsheets?

* Is your company moving towards a distributed database environment?

If you answered yes to more than one of the above questions, you should seriously consider taking advantage of relational database technology.

RELATIONAL HIGHLIGHTS

The following are the key points associated with relational technology:

* Relational concepts are easy to understand and use.
* SQL is a multifunctional language
  - Database definition and creation
  - Data retrieval and manipulation
  - Authorization and security
  - Transaction management and recovery
  - Database environment management and restructuring
  - Interactive and programmatic use
* SQL allows you to specify which information you want - not how to retrieve it.
* SQL increases programmer productivity and raises programming closer to the level of problem solving.
* Data independence is ensured and minimizes maintenance of programs.
* Data access is automatically optimized as the DB structure changes.
* The DBA has unprecedented power and control over the database.
* New systems are implemented much faster.
* Relational databases provide a cost effective, powerful solution.

It is to the advantage of most dataprocessing management to learn to use this technology creatively and to manage it effectively.

PREPARING FOR RELATIONAL

There are several things to consider when going to a relational database environment. The first thing to be aware of are the additional computing resources required to effectively support this technology. For example, the dynamic capability of the relational approach and the intelligence built into the relational software usually requires additional CPU cycles and memory.

Training of the database administrators in database design and data modeling is something that should not be overlooked when developing the relational application environment. The programmer should also be trained in the use of SQL and associated application development tools.

Performance is usually brought up as an issue when discussing relational and often depends on the maturity of the optimizer software which is built into the relational software. The optimizer is used to determine the most efficient way of accessing the database. The Database Administrator (DBA) plays a major role in monitoring and improving performance by creating and dropping indexes when appropriate. The DBA can also elect to use "clustering" or "keeping like data together" which affects performance by reducing the number of times a disc is accessed. Fortunately, Hewlett-Packard's Precision Architecture Computers are providing a high performance platform for relational database applications.

The command-driven nature of SQL may be difficult for some users to understand because they usually have to know the names of the tables and columns in order to properly construct a SQL command. The user may prefer a much more "friendly" interactive menu-driven interface such as Hewlett-Packard's ALLBASE/QUERY on MPE/XL or HP VISOR on MPE/V systems.

Security of the data resources is usually very important. The DBA has the capability to implement some very comprehensive security schemes. In addition, to ensure data integrity, logging of transactions against the database is mandatory and provides for the automatic recovery of the database.

If compatibility with SQL in an IBM environment is important, HP's SQL is very similar. The user and programmer interface is essentially the same; however, there are some DBA functions which are system dependent. Both of these products, as well as others on the market, are converging on industry standard interfaces.

RELATIONAL PRODUCTS IN THE "HP WORLD"

Hewlett-Packard is strategically committed to relational database technology. HP is deeply involved in the various standards committees such as the American National Standards Institute (ANSI) and the international standards oganization called X/OPEN. HP's development team is also aware of the importance of SQL and DB2 from IBM and Codd's rules for a fully relational database.

Significant investment is being made to broaden HP's relational offering and to integrate additional products with this key technology The first of these integrated products, ALLBASE/4GL and ALLBASE/QUERY, were introduced at the recent INTEREX meeting in Sweden.

ALLBASE/4GL (previously HP TODAY) is a highly integrated fourth generation environment for developing and maintaining transaction processing applications. Its outstanding qualities include its excellent prototyping facilities and complete integration of its component tools (screen painter, report writer and high level logic commands) resulting in improved programmer productivity.

ALLBASE/QUERY (previously HP VISOR) is an easy to use, terminal-based query and reporting tool for HP SQL end users. It combines a simple forms-based interface with function key operation to allow end users to perform queries and generate their own reports.

The following table shows the relational database products that are currently supported by Hewlett-Packard. HP VISOR will be changed to ALLBASE/QUERY and HPTODAY will be changed to ALLBASE/4GL on HP-UX systems in November 1988.

HEWLETT-PACKARD'S PROPRIETARY RELATIONAL PRODUCTS

MPE V	MPE XL	HP-UX
HP SQL	HP SQL	HP SQL
HP VISOR	ALLBASE/QUERY	HP VISOR
	ALLBASE/4GL	HP TODAY (4GL)

The following table shows the relational database products that are supplied by HP's third party independent software vendors.

HEWLETT-PACKARD THIRD PARTY RELATIONAL PRODUCTS

MPE V	MPE XL	HP-UX	MSDOS
RELATE/3000	ORACLE (2H88)	ORACLE	ORACLE
	INGRES (1H89)	INGRES	RBASE
	RELATE/3000	INFORMIX	INFORMIX
		UNIFY	

Hewlett-Packard is currently evaluating other relational database and 4GL vendors with an interest in porting their products to HP computer systems.

FUTURE TRENDS IN RELATIONAL TECHNOLOGY

Relational database is becoming a dominant technology in today's information management marketplace. There are several enhancements planned to improve functionality and performance. It eventually will be appropriate for most applications and gain wide acceptance by all users.

Within the next few years, distributed database will provide users with the ability to access data transparently and simultaneously across multiple relational databases on computers in a network. Information will be viewed as one logical database even though it may be fragmented across tables, databases and different vendors' computers. Efficient distributed database will only be possible with relational database. The continued development, growth and maturity of relational databases has made new technologies such as distributed database an extremely important direction for Hewlett-Packard in the future.

STRATEGIC IMPORTANCE

Relational databases can significantly improve the quality, control and accessibility to your organization's extremely important and valuable information resources. It can result in an improved competitive position by aiding business analysis that can help to determine ways to improve products and services.

Unlike non-relational database environments, relational databases adapt easily to dynamic business requirements. In addition, unrestricted access to important data means better information for more effective decision making.

Relational database can also have a very positive effect on many MIS development environments by reducing the application backlog and reducing the time and cost required to develop applications. The improved database flexibility and ease of change also results in a significant reduction in the maintenance of applications.

Overall, the use of relational technology can increase the MIS professionals' effectiveness and productivity which results in improved user satisfaction and confidence. Choosing relational now will position your organization to take full advantage of the technological advances of the future.

SUMMARY

Relational technology can have a profound effect on the way organizations operate. In short, the use of relational databases, within the correct environment, can help turn the computer into the effective tool most managers need to run their organizations effectively. It would be advantageous for your company to become very familiar with relational technology and to begin using this powerful tool. Paul Hessinger, Vice President of Research and Technology at Computer Task Group, said in a recent Computerworld article, "Successful, insightful users of relational DBMS have used 'going relational' as a catalyst for refining and in some cases redefining their entire approach for building and delivering information systems."

REFERENCES

Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," CACM, 13 6,(June 1970) pp. 377-387.

Codd, E.F.,"Relational Database: A Practical Foundation for Productivity," CACM, 25 2,(February 1982) pp. 109-117.

Date, C.J., An Introduction to Database Systems, Addison-Wesley, 1977.

Date, C.J., An Introduction to Database Systems Vol II, Addison-Wesley, 1983.

Larson, O.J.,"Relational Database: How Do You Know You Need One?", Proceedings 1987 North American Conference of Hewlett-Packard Business Computer Users,Las Vegas, Nevada, September 20-25, 1987.

Schussel, George,"Relational Database-Management Concepts, "Proceedings 1988 Database and Fourth/Fifth Generation Language Symposium, Washington, DC, March 21-23, 1988.

_____,Relational Technology: A Productivity Solution, Hewlett-Packard Co., Computer Systems Division, Cupertino, Ca., 5954-6676, January 1986.

_____,SQL/Data System for VSE: A Relational Data System for Application Development, IBM Corp. Data Processing Division, White Plains, N.Y.,G320-6590,Feb 1981.