Panacea or pitfall? The impact of relational databases on your environment

by WILLEM STOELLER
Arthur Andersen & Co.
Chicago, Illinois

ABSTRACT

This paper discusses the impact of relational database management systems (DBMSs) on systems development. It states conditions and characteristics for which relational DBMSs are most applicable and appropriate. Finally, it suggests improvements in relational DBMSs in the areas of performance, data integrity, standardization, and a user-friendly interface.

Appendixes define relational system terms and compare DBMSs with two other data models.
INTRODUCTION

The use of database management systems (DBMSs) during the last five years has increased at a significant rate. A DBMS provides users with a more controlled and flexible environment than that offered by the basic access methods within the operating system. Since DBMSs are the caretakers and delivery mechanisms for data, their evolution will affect all information-processing disciplines and business areas. Relational DBMSs, commercially available since around 1978, could play a dramatic role within the DBMS area. These are the only DBMSs based on a theoretical data model (the relational model, which was developed by E. F. Codd in the early 1970s). Because of its mathematical foundation, the relational model's simple architecture can accommodate new features without the sharp increase in complexity common in most software systems. The most well-known products on the market today are INGRESS, NOMAD, ORACLE, QBE, and SQL/DS. Some are basic data management systems, while others offer additional facilities such as screen painting, report writing, and graphics. Implementations for the business environment, however, have lagged, primarily because of fear of poor performance. However, the current industry emphasis is on management systems that use query-oriented or end-user “fourth generation” data manipulation languages rather than on transaction systems. This trend has pushed relational systems into the spotlight and made relational a popular buzzword in the trade press and in vendor marketing strategies.

This presentation outlines the potential impact of these relational products on your application development and operations. Further, it presents suggestions for improvement of these products.

KEY ISSUES

How will the impact of relational databases relate to the key issues in your environment?

The major issues in the DBMS area include

1. The large applications backlog. Estimates set the average backlog at between two and ten years.
2. The quality of many existing applications and related databases. Current development techniques are ineffective and inefficient in meeting the end-user needs.
3. The inflexible data structures of traditional databases and lack of logical data independence of applications. Changing data structures of existing databases is costly and time consuming.
4. The difficulty of proper physical database design. Database design for products such as IMS and IDMS is complex and costly to adjust. Highly skilled people are required.
5. The design and testing of applications using traditional databases. Applications that must specify navigation through the database are error prone and hard to maintain.
6. The long development time of applications. Today's complex business environment often tolerates only short lead times for information.

PANACEA OR PITFALL?

Characteristics

The Relational Task Group of the American National Standards Institute developed a set of criteria to define relational database systems. These include the following:

1. All data in the database are represented as values in tables.
2. The data manipulation language does not contain explicit navigation links between tables.
3. The command set allows selection of subsets of tables and joining of data from different tables.

(Appendices A and B cover some terminology and a comparison of different data models.)

In addition, most relational systems have at least the following components:

1. A data definition and a data manipulation language based on the relational data model.
2. A user-friendly interface with interactive command execution; an editor for entered commands; and help functions to support the user. The commands can be used in several host languages and utilities in addition to the user-friendly interface.

IMPACT ON THE DEVELOPMENT AND OPERATION

Analysis and Design

After initial analysis of the user's information needs, a prototype of the relational database and the application can be developed with the aid of the user-friendly interface.
End users will be more able to learn about their real information requirements. Armed with this feedback, analysts and programmers can adapt the design of the application. The end result is an application and database that more closely meet the users' information requirements with fewer design bugs.

The preparation of program specifications will be simpler since access to the database can be specified without consideration for navigation or the underlying physical design of the database.

Programming

Programmer productivity will be higher because

1. Parts of the application can be created using procedures based on the user-friendly interface.
2. The database sublanguage is set instead of record-oriented and contains very powerful selection capabilities. This will simplify the logic of the host-language part.
3. The data manipulation language contains only four basic facilities: data access, add rows (records), change elements, and delete rows.
4. Navigation does not need to be specified and strong physical and logical data independence are provided.
5. Training will be much faster than it is for the traditional database sublanguages.

Data Administration

Data administration is a particularly critical function with relational systems because of the emphasis on data rather than on processing logic. A central coordinator is crucial since different user groups use existing data and create new data for particular business needs. The data administrator must keep track of these developments to ensure that data is used consistently, is properly validated, and is not redundant. The administrator must ensure that end users understand and use good, basic data-analysis techniques. The data administrator must find ways to get information about the database to the interactive user. The end user must have access to the current data structure, validation rules, column domains, and database status.

With relational systems, data definition commands are very straightforward. Any knowledgeable user can easily create or change table structures. Restricting this capability is the only way the administration function can exercise control over the database.

Relational DBMSs offer excellent, easy-to-use tools for administration of data security and privacy, usually on the element and row (record) occurrence level.

The most important aid to effective data administration is the data dictionary. Although most relational systems have no fully integrated dictionary, many of the same features are offered by the system catalogs. (The system catalogs are tables used and updated by the system to keep track of information about the database.)

Database Administration

Database administration is considerably easier with relational systems. In a hierarchical DBMS such as IMS, the database administrator must coordinate multiple databases, select efficient access methods for each, and choose pointers and other parameters to define the access path a user may follow. With relational systems, these tasks are eliminated since the database software will automatically optimize access to the data. Database performance tuning is reduced to defining or dropping indexes for specific tables and requesting clustering of frequently linked tables. Changes to the data structures are easy to implement without a lengthy unload/reload procedure. Existing applications are often unaffected by data structure changes owing to a high degree of physical and logical data independence.

End Users

Because of the powerful user-friendly interfaces, many non-DP professionals can use most of the data-manipulation commands with a small training investment.

For data definition and update procedures, end users still need the guidance of DP professionals.

APPLICABILITY OF CURRENT PRODUCTS

Relational DBMSs should be considered a viable possibility for applications with the following characteristics:

1. Table sizes are less than 500,000 rows (records) and performance requirements are modest.
2. Transaction volumes are low, even in peak periods.
3. Interactive and/or ad hoc, database queries are fundamental.

The applications also should have one or more of the following characteristics:

1. Data must be accessed dynamically—data requirements and combinations are not known until execution time.
2. Data structure may evolve over time.
3. Data integrity, data security, data privacy, and data sharing are important.
4. Data will be updated on line.
5. Prototyping of database and query functions is needed to properly define complex requests or to communicate effectively with users.

These latter five conditions are indicators of how appropriate the situation may be for usage of current relational products. They do not all need to be true for the relational systems to be a good fit.

Relational systems, as they stand today, have some limitations that make them inappropriate for applications with one or more of these characteristics:
1. Data volume is large (over a million rows) and/or performance requirements critical.
2. Transaction volumes are large.
3. Data structure is naturally hierarchical, with considerable hierarchical reporting expected.
4. Users of interactive query facility will be casual (less than once a week).

SUGGESTED IMPROVEMENTS

Performance

Relational systems can accommodate medium-volume databases, but their performance will be questionable. Since relational products are all fairly new, further software enhancements will improve performance. Some new storage media, such as content addressable files (CAF from ICL) may provide a solution.

Data Integrity

As with the traditional DBMSs, data integrity relationships between tables are weak in current products. (This is where a row/column value in one table is verified against a required key value in another table.)

Standardization

Standardization of the relational sublanguage will stimulate development and use of relational systems. Currently, relational database standards are being developed by the ANSI Database Standards Committee.

User-Friendly Interface

To appeal to the large community of non-DP professionals, the user-friendly interface needs improvement in the following areas:

1. Syntax. QBE showed that a totally different approach is feasible.
2. Editing. Easy-to-use full-screen editing is desirable.
3. Guidance for interactive users. They should be provided with prompting and menu facilities, clear warnings and help functions, and more forgiving software.

CONCLUSION

Relational systems will claim a sizable database market share in the next five years.
Simple, flexible data structures and supporting English-like command language are what the industry is looking for.

APPENDIX A: RELATIONAL SYSTEM TERMS

Introduction

The area of relational systems has emerged with its own unique language. The mathematical terms reflect the theoretical origins of the relational model. This appendix explains some of the common concepts and terms of relational models.

Relation

A relation is a normalized data aggregate represented by a table. (The terms relation and table can be used interchangeably.) A relation consists of any number of columns in any order and rows in any order. Figure A-1 shows an example of an Employee relation.

Row

Each row is one occurrence of the relation. A row gives a complete piece of information about the relation. No two rows are completely identical, and the unique portion of each row can be in one column or may span several columns. A row (sometimes called a tuple) is similar to a record in a file. The Employee relation in Figure A-1 contains two rows. The meaning of a relation does not change when rows are added or deleted.

<table>
<thead>
<tr>
<th>EMPLOYEE:</th>
<th>NUMBER</th>
<th>NAME</th>
<th>SALARY</th>
<th>DEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>61256</td>
<td>Jones</td>
<td>8000</td>
<td></td>
<td>Household</td>
</tr>
<tr>
<td>38972</td>
<td>Anderson</td>
<td>6000</td>
<td></td>
<td>Toy</td>
</tr>
</tbody>
</table>

Figure A-1—Employee table

Column

Each column is a single data item and represents an attribute of the relation. Repeating groups of data items is not allowed within a column or be repeating columns. A column is similar to the name of a field on a record. The Employee relation in Figure A-1 contains four columns: NUMBER, NAME, SALARY, and DEPT. The meaning of a relation changes when columns are added or deleted.

Domain

The collection of all possible allowable values for each column is called its domain. The domain of the NAME column in the Employee relation is the last name of all employees.

Relational Database

A relational database is simply a collection of relations. The relational database used as an example for the relational operations discussed here.
Relational operations access and manipulate the relations of a relational database. These operations are nonprocedural since they define what to do but not how to do it. In a relational system, data can be accessed only by matching data values. The database user defines what values to match and the system determines the access path to the desired data.

Relational operators work on entire tables (rather than on individual records) and result in new relations. The three principal operations are:

1. Project
2. Select
3. Join

Project

The project operation extracts columns from a relation to create a new relation. Duplicate rows are dropped. The following list demonstrates the project operation where the NAME and SALARY columns were selected from the Employee relation in Figure A-2.

<table>
<thead>
<tr>
<th>NAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>8000</td>
</tr>
<tr>
<td>Anderson</td>
<td>6000</td>
</tr>
<tr>
<td>Morgan</td>
<td>10000</td>
</tr>
<tr>
<td>Murphy</td>
<td>9000</td>
</tr>
</tbody>
</table>

Select

The select operation extracts from a relation to create a new relation. The database user defines the criteria to use in the selection by describing desired data values. The example in Figure A-3 shows a select operation on the Employee relation where only employees in the Toy Department are selected.

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>NAME</th>
<th>SALARY</th>
<th>DEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>38972</td>
<td>Anderson</td>
<td>6000</td>
<td>Toy</td>
</tr>
<tr>
<td>22318</td>
<td>Murphy</td>
<td>9000</td>
<td>Toy</td>
</tr>
</tbody>
</table>

Join

The join operation combines two or more relations to create a new relation. One column in each table must share a common domain to make the join meaningful. Figure A-4 shows an example where the Employee and Manager relations from Figure A-2 are joined on the basis of the DEPT column in each table.

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>NAME</th>
<th>SALARY</th>
<th>DEPT</th>
<th>MGR</th>
</tr>
</thead>
<tbody>
<tr>
<td>61256</td>
<td>Jones</td>
<td>8000</td>
<td>Household</td>
<td>Smith</td>
</tr>
<tr>
<td>38972</td>
<td>Anderson</td>
<td>6000</td>
<td>Toy</td>
<td>Murphy</td>
</tr>
<tr>
<td>09181</td>
<td>Morgan</td>
<td>10000</td>
<td>Cosmetics</td>
<td>Hoffman</td>
</tr>
<tr>
<td>22318</td>
<td>Murphy</td>
<td>9000</td>
<td>Toy</td>
<td>Murphy</td>
</tr>
</tbody>
</table>

Other Terms

An element is a single field value in a relation. It is the intersection of a row and column.

The degree of a relation is the number of columns it has. A N-ary relation is one with N columns. A relation with two columns is a binary relation.

The number of rows in a relation is called its cardinality. A row or record occurrence is sometimes called a tuple. An N-tuple is a row from a relation of N columns.

APPENDIX B: COMPARISON WITH OTHER DATA MODELS

Introduction

A database usually fits one of the three data models: hierarchical, network, or relational. This appendix compares the relational model to the hierarchical and network models.

Access Path

In a hierarchical or network data model, the connections and relationships between data aggregates remain in the data structure. The data designer must carefully predict all data access to ensure that needed data can be extracted via some access path. Two techniques used in that process are to identify alternate entry points and to use unidirectional business-function data models.

With relational systems, access paths are not pre-determined. The relationships between data aggregates are considered in design but are not implemented in the database. Since join operations simply combine tables, the direction of access is irrelevant.

This flexibility in accessing data is the main reason why relational systems excel in unplanned (ad hoc) data requests.

Navigation

The hierarchical or network database user must enter the database at an acceptable entry point and then navigate from...
aggregate to aggregate through the structure to get the needed data. This usually requires the data manipulation language to be coded in a language such as COBOL and embedded in program logic. Conventional programs are needed just to provide the logic to navigate to the data.

Relational database users can manipulate each relation independently or can join relations together as needed. Because the data can be directly accessed, the relational model lends itself to fourth-generation languages. The access logic is in the DBMS, not in the application.

Level of Access

The data manipulation languages for hierarchical and network systems generally return one record at a time. Therefore, the application logic must interact frequently with the DBMS to obtain multiple records.

Relational operations manipulate entire relations in a single request to the DBMS. The relational system always creates a new relation as a result of an operation. The fact that table operations create tables is called the closure property.

Maintenance

Even a small change to a data aggregate in a hierarchical or network system may have a significant impact on the physical implementation and application programs. Programs may need to be recompiled, database definition utilities rerun, and the entire database reloaded. This maintenance is almost always performed via batch processing.

Relational systems generally allow new relations to be defined interactively and new columns to be added to existing tables. Since columns do not have any inherent sequence, each operation can order the columns in any sequence.

Physical Database

Hierarchical and network systems use indexes, required pointers, and key fields to keep occurrences of an aggregate in sequence. Physical database design uses the concept of physical proximity of related records to improve performance. Application program logic must know about and properly use indexes to support processing and database structuring.

Indexes are never required in a relational system. Rows are not physically stored in any particular order, so the sequence of occurrences cannot be used to store information about the rows or their relationship. Rows may be sequenced when a relational operation is used. Physical design is shielded from the database user.

Language

Hierarchical and network systems have a data-manipulation language to access data and a distinct data-definition language to create the data structures. The data-definition process typically consists of a set of control cards that are input to a batch job, and the data manipulation syntax is programmer oriented and embedded in application programs.

In contrast, the data-definition and data-manipulation languages for a relational system are one and the same. The syntax is end-user oriented and can be used either interactively or embedded in source programs.

Conclusion

Relational systems separate logical data design from physical design. This data independence gives relational systems more flexibility in accessing data, allows user of higher-level languages, and lessens the impact of database changes. Hierarchical and network systems provide less data independence, so changes to the physical database affect the logical data design.

BIBLIOGRAPHY

For the first time, the National Computer Conference devotes an entire program track to the human, organizational, and social effects of computers. This provides an opportunity to bring together an outstanding group of speakers, many of whom have spent years studying the computer’s impact on work, organizations, human behavior, and society in general. The theme running through their presentations is that by understanding and responding to these impacts, systems, organizations, and society become more effective.

The program begins with “Organizations, Information Systems, and Office Effectiveness.” The panel will present the results of two recent National Science Foundation studies on the use of organizational design models to insure effective implementation and adaptation of systems. Their findings support the use of organizational redesign as a tool for increasing systems’ effectiveness.

The next panel takes a more futuristic and international view in exploring the impact of “Fifth Generation Computers.” The panel will assess Japan’s recently announced projects and present U.S. industry leaders’ plans for building up our technology base for the 1990s. They will discuss why, when, how, by whom, and how much.

The panel “Managing Computer Change” brings together an experienced group of panelists who will address the problems of change and uncertainty between managers and the system, managers and their personnel, and system developers and users. The speakers will give illustrations of how system developers and managers must appear not to manipulate while they help the users to deal with uncertainty.

The next panel, “Measuring the Impact of Information Techniques,” will discuss and present case studies on the effect of computers in the office environment. They will go beyond the narrow “productivity” measures to throw light on the broader organizational impacts and how they can be measured.

For the panel “Living with Computers: The Multi-Societal Effects,” the panelists will present current research findings on socioeconomic impacts within the computer industry, organizational impacts within local government, and impacts on individual behavior within organizations.

“Computerized Society—Resilient or Vulnerable?” will discuss whether a new and serious vulnerability to disruptions or computer unavailability exists, or whether the society has an intrinsic resiliency to major disruptions.

The ergonomic and human factor issues are discussed in the “Man-Machine Interaction” panel. The presentations will cover current ergonomic problems and their solutions, plus improved interface aids to mainframe IBM systems.

The panel “Computer Aids to the Handicapped” presents three applications: (1) sensory and information processing aids for the blind; (2) communications aids for the nonvocal; and (3) a lip-reading aid for the deaf. The applications’ possible wider use in office automation systems will be explored.

The final panel is titled “The Institutional Dimensions of Computing in Organizations and Society”; it will discuss dimensions such as persistence and rigidity that can be introduced by computers. The pros and cons of such dimensions will be explored in organizations and in the larger society, including the Soviet Union.