

NON-MODAL DATABASE SYSTEM WITH METHODS FOR INCREMENTAL MAINTENANCE OF LIVE REPORTS

RELATED APPLICATIONS

The present application is related to co-pending application Ser. No. 08/379,226, filed on Jan. 27, 1995 and now pending.

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

BACKGROUND OF THE INVENTION

The present invention relates generally to information processing environments and, more particularly, to end-user data processing systems, such as a PC Database Management System (PC DBMS).

Computers are a powerful tool for the acquisition and processing of information. Computerized databases can be regarded as a kind of electronic filing cabinet or repository for collecting computerized data files; they are particularly adept at processing vast amounts of information quickly. As such, these systems serve to maintain information in database files or tables and make that information available on demand.

In 1970, Dr. E. F. Codd invented the "relational model", a prescription for how a DBMS should operate. The relational model provides a foundation for representing and manipulating data, that is, a way of looking at data. The model includes three basic components: structure, integrity, and manipulation. Each will be briefly described.

The first of these, structure, is how data should be presented to users. A database management system is defined as "relational" when it is able to support a relational view of data. This means that data which a user can access and the operators which the user can use to operate upon that data are themselves relational. Data are organized as relations in a mathematical sense, with operators existing to accept relations as input and produce relations as output. Relations are perhaps best interpreted by users as tables, composed of rows (tuples) and columns (attributes).

Ideally, data in a relational system is perceived by users as tables and nothing but tables. This precludes the user from seeing explicit connections or links between tables, or having to traverse between tables on the basis of such links. It also precludes user-visible indexes on fields and, in fact, precludes users from seeing anything that smacks of the physical storage implementation. Thus, tables are a logical abstraction of what is physically stored.

The integrity aspect, on the other hand, dictates that every relation (i.e., table) should have a unique, primary key to identify table entries or rows. The integrity of the data for the user is of course crucial. If accuracy and consistency of the data cannot be achieved, then the data may not be relied upon for decision-making purposes.

Data manipulation, the last component, may be thought of as cut-and-paste operators for tables. Data manipulation is of course the purpose for which databases exist in the first place. The superiority of manipulating tables relationally

(i.e., as a whole, or sets of rows) is substantial. Users can combine data in various tables logically by matching values in common columns, without having to specify any internal details or the order in which tables are accessed; this provides users with a conceptual view of the database that is removed from the hardware level. Non-relational DBMSs, in contrast, require complex programming skills that form an inherently unreliable means to interact with databases.

The general construction and operation of a database management system is known in the art. See e.g., Date, C., *An Introduction to Database Systems*, Volume I and II, Addison Wesley, 1990; the disclosures of which are hereby incorporated by reference.

Today, relational systems are everywhere—commonly seen operating in corporate, government, academic settings, and other shared environments. A typical installation will employ one of the popular UNIX-based RDBMS running on a minicomputer. By submitting queries to the DBMS from a remote terminal (e.g., using a SQL "query editor"), users are often able to handle many of their own data processing needs directly. Thus, relational technology is not only just another way to build a database system, but it also offers a set of underlying principles that provide very direct practical benefits to the user.

The strong theoretical underpinnings of relational systems which account for their superior design over prior non-relational systems have also created some unexpected problems. With the ever-increasing trend towards "down-sizing," more and more data processing tasks are being moved off mainframes and minicomputers and onto desktop PCs, often operating in a Local Area Network (LAN). Although relational systems are easier for database-savvy users to use (e.g., for querying), they are by no means easy for untrained users. And with the movement of data processing chores to desktop PCs, ordinary PC users are nevertheless often faced with the responsibility of designing and implementing a database system, one having the reliability and integrity typically associated with a relational system.

Consider the following issues attendant to setting up a relational database management system (RDBMS). Tables in a relational system are not just any tables but are, instead, special "disciplined" tables. Relational systems require, for instance, that tables not store duplicates (so that each row may be uniquely identified by one or more column values). Thus, relations or "R-tables" are subject to particular constraints, such as "first normal form." As another example, to preserve simplicity and take advantage of relational operations, database tables should not contain "repeating groups"—that is, multi-valued columns. Such multi-valued columns remove table resemblance to relations and thus prevent tables from taking advantage of the latter's mathematical properties. Instead, relational tables should contain only single-value cells or "atomic" data values. Thus while relational tables are simple and flexible in theory, they nevertheless entail rigorous constraints which must be obeyed to implement them in practice.

While trained database administrators have the expertise to tackle such issues, ordinary PC users for the most part have received no formal data processing education. They cannot be expected to be familiar with such seemingly esoteric concepts as "joins," "one-to-many relations," "foreign keys," or any of the other myriad of issues which must be considered when applying the relational approach to database management.

All told, relational database products have over time become more and more advanced. And technical advances