Craig S. Mullins

Return to Home Page

Vol. 15, No. 3 (Winter 2008-2009)

From IDUG Solutions Journal...

The Buffer Pool

Database Fundamentals

By Craig S. Mullins

For this issue’s column I’d like to broaden the scope of coverage from DB2 specifically, to databases and database management in general. I hope you’ll forgive this diversion of coverage because I think it is necessary that as IT professionals we have a grounded educational foundation in proper terminology, usage, and understanding.

I know, most readers of this column probably think that they understand the basic concepts and fundamentals of database technology. However, some who think they understand the basics might actually have less of a firm grip on the topic than they think. So let's review some of the fundamentals of database management systems.

First, what is a database? Again, I'd wager that most readers think they know the answer to that question. I'd also bet that many of you are wrong. DB2 is not a database; neither are Informix, Oracle and SQL Server. Each of these is a DBMS, or Database Management System. You can use DB2 (or Informix or SQL Server) to create a database, but DB2, in and of itself, is not a database.

So what is a database? ***A database is a large structured set of persistent data.*** So a phone book is a database. But within the world of IT a database usually is associated with software. A simple database might be a single file containing many records, each of which contains the same set of fields where each field is a certain data type and length. In short, a database is an organized store of data wherein the data is accessible by named data elements.

A Database Management System (DBMS) is a software package designed to create, store, and manage databases. The DBMS software enables end users or application programmers to share data. It provides a systematic method of creating, updating, retrieving and storing information in a database. DBMSs also are generally responsible for data integrity, data access control, and automated rollback, restart and recovery.

In layman's terms, you can think of a database as a file of information. You can think of the filing cabinet itself along with the file folders and labels as the DBMS. A DBMS manages databases. You implement and access database instances using the capabilities of the DBMS.

So, DB2 (and Oracle, Informix, IMS, etc.) are database management systems. Your payroll application uses the payroll database, which may be implemented using DB2 (or IMS or …). Why is this distinction important? If we do not use precise terms when we write, speak, and work confusion can result. And confusion leads to over budget projects, improperly developed systems, and lost productivity. So precision must be important to us.

Why Use a DBMS?

The main advantage of using a DBMS is to impose a logical, structured organization on the data. A DBMS delivers economy of scale for processing large amounts of data because it is optimized for such operations.

A DBMS can be distinguished by the model of data upon which it is based. A data model is a collection of concepts used to describe data. There are two fundamental components of a data model: its structure, that is, the way data is stored, and its operations, that is the way that data can be manipulated. There are four major data models used by the most popular DBMS products:

1. Hierarchical

2. Network (or CODASYL)

3. Relational

4. Object-oriented

The hierarchical model arranges data into hierarchic trees that stores data at lower levels in the hierarchy as subordinate to data stored in higher levels. A hierarchy is a network with the additional restriction that access into a record can only be accomplished in one way. The hierarchical model is tree-structured. IMS is an example of a DBMS based on the hierarchical model.

The network model is structured as a collection of record types and the relationships between these record types. All relationships are explicitly specified and stored as part of the structure of the DBMS. Another common name for the network model is CODASYL. CODASYL is named after the Conference on Data SYStems Languages, the committee that formulated the model in the early 1970s. Data is manipulated using the location of a given record and following links to related records. IDMS is an example of a DBMS based on the hierarchical model.

The relational model consists of a collection of tables (more properly, relations) with relationships between tables being provided by columns. The relational model is based on the mathematics of set theory. Contrary to popular belief, the relational model is not named after “relationships,” but after the relations of set theory. A relation is basically a set with no duplicate values. Data can be manipulated in many ways, but the most common is through SQL. DB2, Oracle, Informix, and SQL Server are examples of DBMSs based on the relational model (though none of them is “fully” relational).

The object-oriented (OO) model consists of a collection of entities, or objects, where each object includes the actions that can take place on that object. In other words, an object encapsulates data and process. With OO systems data typically is manipulated using an OO programming language. ObjectStore and Cache are examples of DBMSs based on the OO model.

Each of these four data models is referred to as a data model for simplicity sakes. Actually, only the relational and network models have any true, formal data model specification. Different models of data lead to different logical and structural data organizations. The relational model is the most popular data model because it is the most abstract and easiest to apply to data, while providing powerful data manipulation and access capabilities.

Advantages of Using a DBMS

Additionally, using a DBMS provides a central store of data that can be accessed by multiple users, from multiple locations. Data can be shared among multiple applications, instead of new iterations of the same data being propagated and stored in new files for every new application. Central storage and management of data within the DBMS provides:

- Data abstraction and independence.
- Data security.
- A locking mechanism for concurrent access with ACID properties (ACID is an acronym for atomicity, consistency, isolation, and durability).
- An efficient handler to balance the needs of multiple applications using the same data.
- The ability to swiftly recover from crashes and errors.
- Robust data integrity capabilities.
- Simple access using a standard API.
- Uniform administration procedures for data.

Levels of Data Abstraction

A DBMS offers the ability to provide many views of a single database schema. A view defines what data the user sees and how that user sees the data. The DBMS provides a level of abstraction between the conceptual schema that defines the logical structure of the database and the physical schema that describes the files, indexes and other physical mechanisms used by the database. Users function at the conceptual level -- for example, by querying columns within rows of tables -- instead of having to figure out how to access data using the many different types of physical structures used by the DBMS to store the data.

When a DBMS is used, systems can be modified much more easily when business requirements change. New categories of data can be added to the database without disruption to the existing system.

Data Independence

A DBMS provides a layer of independence between the data and the applications that use the data. In other words, applications are insulated from how data is structured and stored. The DBMS provides two types of data independence:

1. *Logical data independence*: Protection from changes to the logical structure of data.
2. *Physical data independence*: Protection from changes to the physical structure of data.

As long as the program uses the API (application programming interface) to the database ad provided by the DBMS, developers can avoid changing programs because of database changes.

Note: The primary API to “relational” databases is SQL. In general, most application SQL statements need not change when database structures change (e.g. a new column is added to a table).

Data Security

Data security prevents unauthorized users from viewing or updating the database. The DBMS uses IDs and passwords to control which users are allowed access to which portions of the database. For example, consider an employee database containing all data about individual employees. Using the DBMS security mechanisms, payroll personnel can be authorized to view payroll data, whereas managers could be permitted to access only data related to work and project history.

Concurrency Control

A DBMS can serve data to multiple, concurrently executing user programs. To do so, requires a locking mechanism to deliver concurrency control. The actions of different user programs running at the same time could conceivably cause data inconsistency without a locking mechanism. For example, multiple bank ATM users might be able to withdraw $100 each from a checking account containing only $150. Using a DBMS ensures that such problems are avoided because the locking mechanism isolates transactions competing for the same exact data.

Transaction Logging

The DBMS uses database logging to record before and after images of database objects as they are modified. It is important to note that the database log captures information about every data modification (except in certain circumstances as directed by the DBA). The information on the database logs can be used to un-do and to re-do transactions. Database logging related activities are handled transparently by the DBMS -- that is, the user need not request or code anything to ensure it is done.

Ensuring Atomicity and Durability

A DBMS can be used to assure the all-or-nothing quality of tranactions. This is referred to as atomicity and it means data integrity is maintained that even if the system crashes in the middle of a transaction. Furthermore, a DBMS provides recoverability. After a system failure data can be recovered to a prior state -- either immediately before the crash, or to some other consistent point-in-time.

Data Integrity

The DBMS provides mechanisms for defining rules that govern the type of data that can be stored in specific fields, or columns. Only data that conforms to the business rules will ever be stored in the database. Furthermore, the DBMS can be set up to manage relationships between different types of data and to ensure that changes to related data elements are accurately implemented.

Data Access

DBMSs provide a standard query language to enable users to interactively interrogate the database and analyze its data. For relational databases this standard API is SQL, or Structured Query Language. Furthermore, many DBMS products ship with analytical tools and report writers to further simplify data access.

Summary

A primary benefit of a DBMS is its ability to maintain and query large amounts of data while assuring data integrity and consistency. It offers transparent recovery from failures, concurrent access, and data independence. In fact, most modern computer applications rely on DBMS and database technology to manage data. Understanding the topic is of benefit to all IT professionals.

This short review of DBMS fundamentals is necessarily brief because of its publication format – and most readers should find this material to be familiar. If you require additional details on the basic operations and qualities of DBMSs and databases, please refer to some of the books listed below for additional knowledge on the topic.

- Chris Date’s "An Introduction to Database Systems, 8th edition" for an academic and theoretical approach to the material.
- Fabian Pascal’s two books, "Practical Issues in Database Management" and "Understanding Relational Databases" for an opinionated, yet informed approach to the topic.
- Joe Celko’s “Data & Databases: Concepts In Practice” for a good practical overview of the topic.
- And, Pratt and Adamski’s “The Concepts of Database Management” for a good high-level overview of DBMS concepts.

From *IDUG Solutions Journal*, Winter 2008 - 2009.