Database Management & Security:
DBMS Components

by M. E. Kabay, PhD, CISSP-ISSMP
Associate Professor of Information Assurance
School of Business & Management
Norwich University, Northfield VT

This is the fifth in a series of articles on database management and security. In this article, we look at terminology and tools of database management systems.

A database management system (DBMS) includes databases (DBs) and the tools for building, modifying, monitoring, securing, and repairing them. A DB contains files – sometimes people call them tables and in some systems they are called relations or datasets – each of which consists of records (also called rows or tuples) in which there are fields (also called columns or attributes). Designers establish relationships or links among tables to help locate data; these linkages among the tables help users navigate through the DB.

It’s the DB application that provides a particular way of accessing the DB; for example, a particular DB application might be an accounting program or a production-control program or a patient-records program. DB applications provide for our control over what constitutes acceptable data; for example an inventory system can include provisions for defining reorder points to prevent exhaustion of parts supplies.

DB applications provide user interfaces so that employees can enter data quickly and correctly as well as locating data using interactive queries or stored procedures. Applications normally include capabilities for creating new reports as well as for generating predefined reports.

The DBMS also includes a DB application with general capabilities often called a query program. Users with little or no programming experience can use the query program to work on subsets of their data to answer specific questions, for calculations and to generate reasonably sophisticated reports with features like sorting, subtotals, headers and even graphs.

Unfortunately, query programs run by database administrators (DBAs) may also bypass normal security restrictions, resulting in security problems if they tamper with the self-consistent data of the DB.

Access by an application program to the data flows through an application program interface or API which in turn depends on the internals of the DBMS to interpret metadata from the data dictionary. The metadata translate requests for functional definitions of the data such as the name of a patient into pointers to records and descriptors of the specific parts of the records that correspond to the needed data.

The relational model has a number of strict requirements. The most important is that every single record must be unique. That means in practice that we are going to have to name an attribute of the information that is naturally unique or alternatively, to impose one that we can force to be unique. We call the unique identifier of the record the key.

Keys make an enormous difference to the structure and performance of databases. Keys can consist of a single field or of several fields that are concatenated to form a compound key. Keys can be used to create a special set of pointers called an index (plural indexes or indices) that can
greatly speed access to records. For example, if patient_ID is a key for treatment records, then one could have the database almost instantly retrieve the records of treatments for a specific patient without having to read all of the patient records to find the right ones. The patient_ID key allows the DBMS to use random access (direct I/O) with a specific record number instead of serial access (serial I/O).

Defining appropriate keys is critically important in designing databases. The choice of keys depends very much on the kinds of questions that users typically ask; one of the concerns is that keys add to the overhead of the database – both in terms of requiring extra storage space for the pointers (some define a chain of values with the same key and others point to the start or end of the chain) and in imposing a performance cost whenever we add or delete records (because we have to modify pointers to keep the chain descriptors correct).

Choosing the right keys is at the heart of the database designers skillset. Just as an example, imagine an order-entry system in which the only key to the order dataset were mistakenly defined as customer_ID; it would be impossible to have more than one order in the dataset per customer – a ridiculous constraint.

Continuing our imaginary order-entry system example, suppose an amateur database designer decided to include full customer information (name, address, telephone number) in the same record as the order number and the date of an order (what we call an order-header, which corresponds to the top of an order form). Suppose a very large customer had 3,000 orders in the database – are we to accept that there should be 3,000 copies of the same information stored uselessly in the order-header file? Ridiculous. We prevent these problems in the process called normalization, which is the subject of the next articles in this series.

* * *

This series of articles is based on the narrated lecture “Introduction to Database Management System Administration & Security” <http://www.mekabay.com/courses/academic/norwich/msia/index.htm> prepared for the MSIA program <http://www.graduate.norwich.edu/infoassurance/> at Norwich University <http://www.norwich.edu>. You are welcome to download the lecture files at any time.

* * *

M. E. Kabay, PhD, CISSP-ISSMP <mailto:mekabay@gmail.com> specializes in security and operations management consulting services. CV online.<http://www.mekabay.com/cv/>

Copyright © 2009 M. E. Kabay. All rights reserved.

Permission is hereby granted to Network World to distribute this article at will, to post it without limit on any Web site, and to republish it in any way they see fit.