Teaching Mathematics and Statistics in Sciences - IPA HU-SRB/0901/221/088
Interesting Mathematical Problems in Sciences and Everyday Life - 2011

Basic and Advanced Database Courses

Srdjan Škrbić
Faculty of Science, University of Novi Sad
Trg Dositeja Obradovića 4, 21000 Novi Sad, Serbia
shkrba@uns.ac.rs
http://www.is.pmf.uns.ac.rs/skrbics

At the beginning of the course, we explore some basic database concepts and adopt terminology. We give an overview of the most important data models. First we give brief remarks on historical network and hierarchical data models, and then we continue to investigate entity-relationship and relational data model. Only most important facts about entity-relationship model, together with some examples will be covered. The relational data model will be presented in much more detail, but we concentrate on exploring possibilities for its practical usage. Important remarks about theoretical background of the relational model are covered later in the text. We continue with the most important constructions of the Structured Query Language.

In the advanced level course, we concentrate on the theoretical background of the relational model, explore it in some detail, and explain implications these concepts make to the usage of the relational model in practice. Among other topics, we explore functional dependencies in detail. Next, we give some pointers on update anomalies and the need to introduce normal (canonical) forms to relational database theory. We describe normal forms from first to third, including the Boyce-Codd normal form that belongs somewhere between the third and the fourth. We also give pointers about other normal forms. At the end we introduce two algorithms for relational database normalization - decomposition and synthesis.

1. Introduction

Databases today are essential to every business. Whenever you visit a major Web site Google, Yahoo!, Amazon.com, or thousands of smaller sites that provide information there is a database behind the scenes serving up the information you request. Corporations maintain all their important records in databases. Databases are likewise found at the core of many scientific investigations. They represent the data gathered by astronomers, by investigators of the human genome, and by biochemists exploring properties of proteins, among many other scientific activities. The power of databases comes from a body of knowledge and technology that has developed over several decades and is embodied in specialized software called a database management system, or DBMS. A DBMS is a powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely. These systems are among the most complex types of software available.

In essence a database is nothing more than a collection of information that exists over a long period of time. The term database also refers to a collection of data that is managed by a DBMS. The DBMS is expected to allow users to:

- create new databases and specify their schemas,
- query the data and modify the data,
- support the storage of very large amounts of data,

- enable durability, the recovery of the database in the face of failures and
- control access to data from many users at once, without allowing unexpected interactions among users and without actions on the data to be performed partially but not completely.

The early DBMS's required the programmer to visualize data much as it was stored. These database systems used several different data models for describing the structure of the information in a database. Two most important models were "hierarchical" or tree-based model and the graph-based "network" model. A problem with these early models and systems was that they did not support high-level query languages. That is why there was considerable effort needed to write such programs, even for very simple queries.

Following a famous paper written by Codd [3] in 1970, database systems changed significantly. Codd proposed that database systems should present the user with a view of data organized as tables called relations. Behind the scenes, there might be a complex data structure that allowed rapid response to a variety of queries. But, unlike the programmers for earlier database systems, the programmer of a relational system would not be concerned with the storage structure. Queries could be expressed in a very high-level language, which greatly increased the efficiency of database programmers.

By 1990, relational database systems were the norm. Yet the database field continues to evolve, and new issues and approaches to the management of data surface regularly. Object-oriented features have infiltrated the relational model. Some of the largest databases are organized rather differently from those using relational methodology.

1.1. Overview of a Database Management System

In Fig. 1 we see an outline of a complete DBMS. Single boxes represent system components, while double boxes represent in-memory data structures. The solid lines indicate control and data flow, while dashed lines indicate data flow only. Since the diagram is complicated, we shall consider the details in several stages. First, at the top, we suggest that there are two distinct sources of commands to the DBMS:

1. Conventional users and application programs that ask for data or modify data.
2. A database administrator: a person or persons responsible for the structure or schema of the database.

1.1.1. Query Processing

The second kind of command is the simpler to process, and we show its trail beginning at the upper right side of Fig. 1. For example, the database administrator, or DBA, for a university registrar's database might decide that there should be a table or relation with columns for a student, a course the student has taken, and a grade for that student in that course. The DBA might also decide that the only allowable grades are A, B, C, D, and F. This structure and constraint information is all part of the schema of the database. It is shown in Fig. 1 as entered by the DBA, who needs special authority to execute schema-altering commands, since these can have profound effects on the database. These schema-altering data-definition language (DDL) commands are parsed by a DDL processor and passed to the execution engine, which then goes through the index/file/record manager to alter the meta data, that is, the schema information for the database.

The great majority of interactions with the DBMS follow the path on the left side of Fig. 1. A user or an application program initiates some action, using the data-manipulation language (DML). This command does not affect the schema of the database, but may affect the content of the database or will extract data from the database.

The query is parsed and optimized by a query compiler. The resulting query plan, or sequence of actions the DBMS will perform to answer the query, is passed to the execution engine. The execution engine issues a sequence of requests for small pieces of data, typically records or tuples of a relation, to a resource manager that knows about data files (holding relations), the format and size of records in those files, and index files, which help find elements of data files quickly.

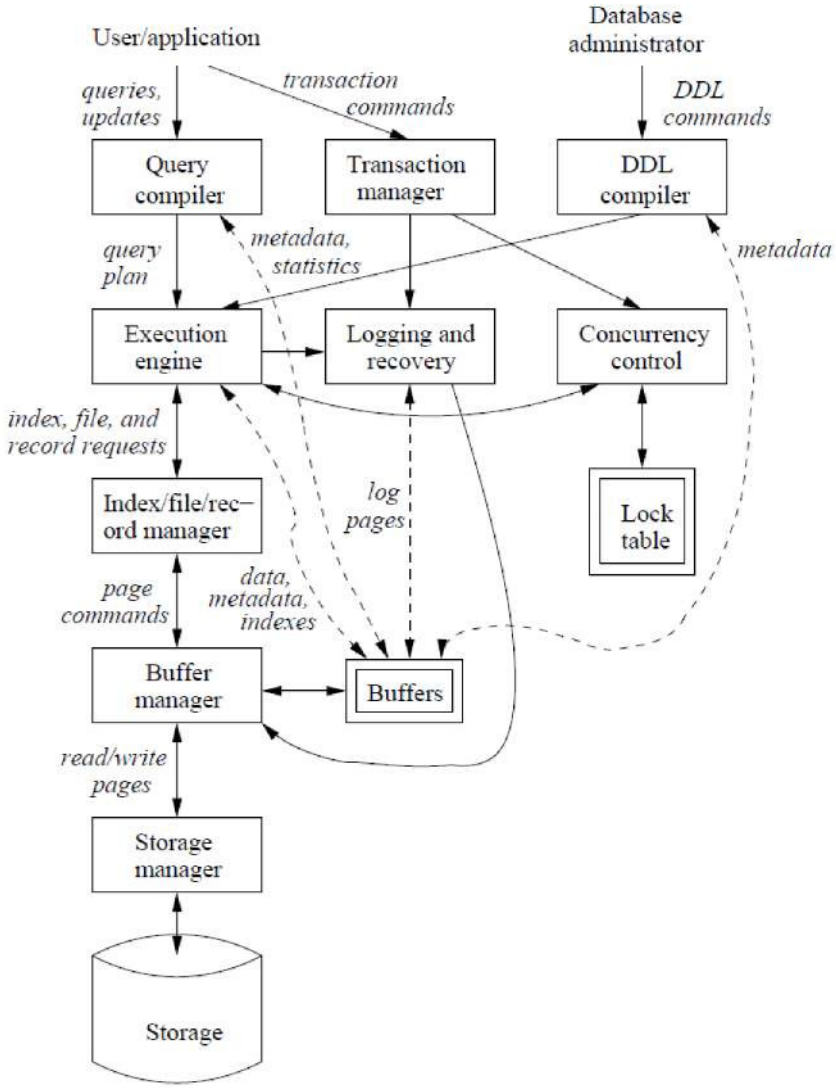

Figure 1: DBMS components.

The requests for data are passed to the buffer manager. The buffer manager's task is to bring appropriate portions of the data from secondary storage (disk) where it is kept permanently, to the main-memory buffers. Normally, the page or "disk block" is the unit of transfer between buffers and disk.

The buffer manager communicates with a storage manager to get data from disk. The storage manager might involve operating-system commands, but more typically, the DBMS issues commands directly to the disk controller.

Queries and other DML actions are grouped into transactions, which are units that must be executed atomically and in isolation from one another. Any query or modification action can be a transaction by itself. In addition, the execution of transactions must be durable, meaning that the effect of any completed transaction must be preserved even if the system fails in some way right after completion of the transaction.

1.1.2. Storage and Buffer Management

The data of a database normally resides in secondary storage - a magnetic disk, for example. However, to perform any useful operation on data, that data must be in main memory. It is the job of the storage manager to control the placement of data on disk and its movement between disk and main memory. For efficiency purposes, DBMS's normally control storage on the disk directly, at least under some circumstances. The storage manager keeps track of the location of files on the disk and obtains the block or blocks containing a file on request from the buffer manager.

The buffer manager is responsible for partitioning the available main memory into buffers, which are page-sized regions into which disk blocks can be transferred. Thus, all DBMS components that need information from the disk will interact with the buffers and the buffer manager, either directly or through the execution engine. The kinds of information that various components may need include: data, meta data, log records, statistics, indexes, etc.

1.1.3. Transaction Processing

It is normal to group one or more database operations into a transaction, which is a unit of work that must be executed atomically and in apparent isolation from other transactions. In addition, a DBMS offers the guarantee of durability: that the work of a completed transaction will never be lost. The transaction manager therefore accepts transaction commands from an application, which tell the transaction manager when transactions begin and end, as well as information about the expectations of the application.

The transaction processor performs the following tasks:

1. Logging: In order to assure durability, every change in the database is logged separately on disk.
2. Concurrency control: The concurrency-control manager must assure that the individual actions of multiple transactions are executed in such an order that the net effect is the same as if the transactions had in fact executed in their entirety, one-at-a-time.
3. Deadlock resolution: As transactions compete for resources through the locks that the scheduler grants, they can get into a situation where none can proceed because each needs something another transaction has. The transaction manager has the responsibility to intervene and cancel ("rollback" or "abort") one or more transactions to let the others proceed.

Properly implemented transactions are commonly said to meet the "ACID test", where:

- "A" stands for "atomicity," the all-or-nothing execution of transactions.
- "C," stands for "consistency." That is, all databases have consistency constraints, or expectations about relationships among data elements.
- "I" stands for "isolation," the fact that each transaction must appear to be executed as if no other transaction is executing at the same time.

- ”D” stands for ”durability,” the condition that the effect on the database of a transaction must never be lost, once the transaction has completed.

1.1.4. The Query Processor

The portion of the DBMS that most affects the performance that the user sees is the query processor. In Fig. 1 the query processor is represented by two components - the query compiler and the execution engine.

The query compiler, which translates the query into an internal form called a query plan. The latter is a sequence of operations to be performed on the data. The query compiler consists of three major units:

1. A query parser, which builds a tree structure from the textual form of the query.
2. A query pre processor, which performs semantic checks on the query and performing some tree transformations to turn the parse tree into a tree of algebraic operators representing the initial query plan.
3. A query optimizer, which transforms the initial query plan into the best available sequence of operations on the actual data.

The execution engine, which has the responsibility for executing each of the steps in the chosen query plan. The execution engine interacts with most of the other components of the DBMS, either directly or through the buffers. It must get the data from the database into buffers in order to manipulate that data. It needs to interact with the scheduler to avoid accessing data that is locked, and with the log manager to make sure that all database changes are properly logged.

2. Data models

The notion of a ”data model” is one of the most fundamental in the study of database systems. In this brief summary of the concept, we define some basic terminology and mention the most important data models.

A data model is a notation for describing data or information. The description generally consists of three parts:

1. Structure of the data. The data structures used to implement data in the computer are sometimes referred to as a physical data model. In the database world, data models are at a somewhat higher level than data structures, and are sometimes referred to as a conceptual model to emphasize the difference in level.
2. Operations on the data. In database data models, there is usually a limited set of operations that can be performed. We are generally allowed to perform a limited set of queries (operations that retrieve information) and modifications (operations that change the database). By limiting operations, it is possible for programmers to describe database operations at a very high level, yet have the database management system implement the operations efficiently.
3. Constraints on the data. Database data models usually have a way to describe limitations on what the data can be. These constraints can range from the simple (e.g., ”a day of the week is an integer between 1 and 7” or ”a movie has at most one title”) to some very complex forms of limitations.

Here we give an overview of the two most important data models - the entity-relationship and the relational model.

2.1. Entity-relationship model

Entity-relationship diagrams were first proposed as a means of quickly obtaining, with minimum effort, a good sense of the structure of a database [14]. They are used to plan and design a database and to model a system’s data.