
DATABASE SYSTEMS - THE COMPLETE BOOK

TEXT NOTES AND SUMMARY

Original text by: H. Garcia-Molina, J. D. Ullman, J. Widom

Chapter 1 The Worlds of Database Systems

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* (DBMS), or “database system”. 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. We shall learn how to design databases, how to write programs in the various languages associated with a DBMS, and how to implement the DBMS itself.

1.2 Overview of a Database Management System

We present 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.

First we suggest there are two distinct sources of use to the DBMS:

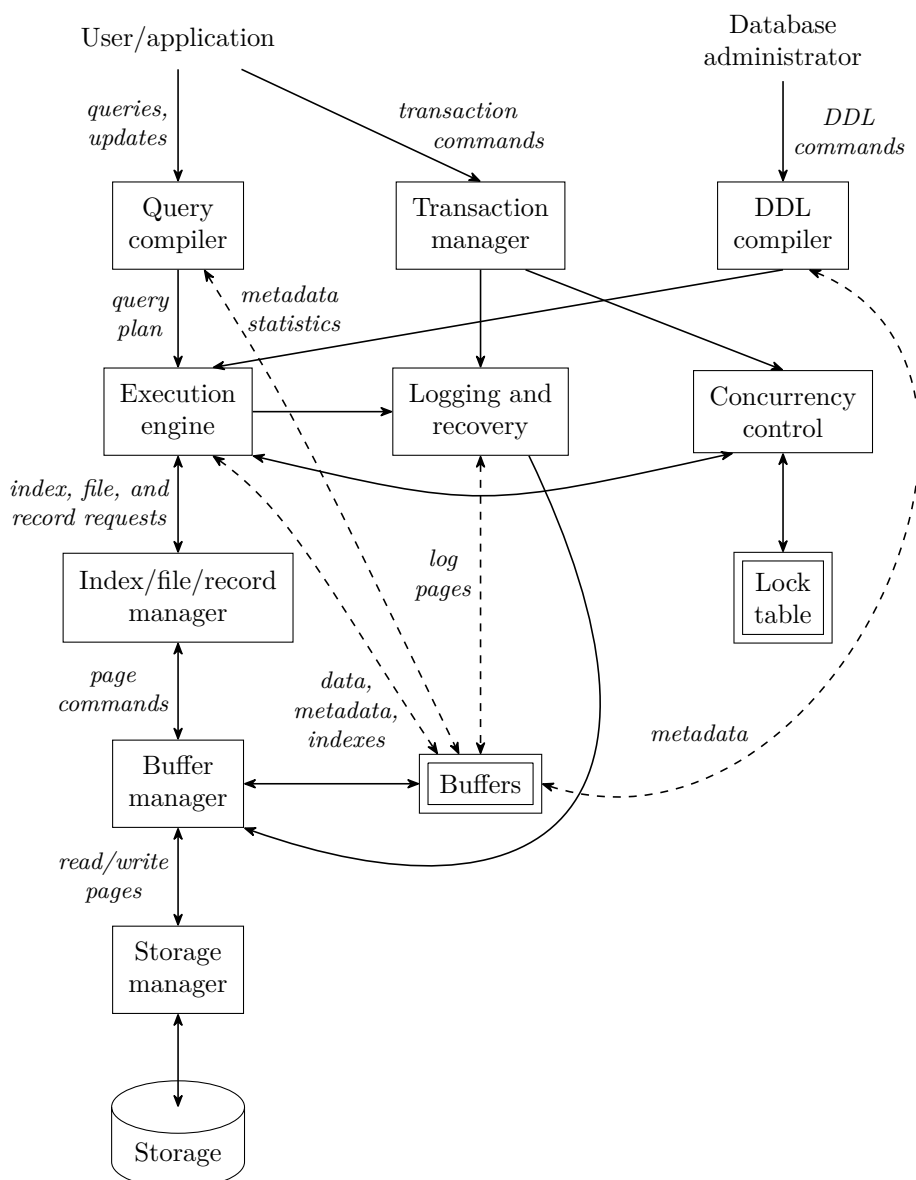
1. Conventional users and application programs that ask for data or modify data.
2. A *database administrator*, responsible for the structure or *schema* of the database.

1.2.1 Data-Definition Language Commands

The second kind of commands: schema-altering *data-definition language* (DDL) commands, are parsed by a DDL processor and passed to the execution engine, which then goes to the index/file/record manager to alter the *metadata* (schema information for the database).

1.2.2 Overview of Query Processing

A user or an application program initiates some action using the *data-manipulation-language* (DML). This command cannot affect the schema of the database, but may affect its contents, or extract data from it. These commands are handled by two separate subsystems.



Answering the Query The query is parsed and optimized by a *query compiler*. The resulting *query plan* (sequence of actions the DBMS will perform to answer the query) is passed to the *execution engine*, which 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. The requests for data are passed to the *buffer manager*, which task is to bring appropriate portions of the data from secondary storage (disk) where it is kept permanently, to the main-memory buffers, communicating 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.

1.2.3 Storage and Buffer Manager

Data of a database normally resides in secondary storage (disk). However, to perform any useful operation on data, that data must be in main memory. The *store manager* controls the placement of data on disk and its movement between disk and main memory.

The storage manager keeps track of the location of files on the disk and obtains the 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.

So all DBMS components that need information from the disk will interact with the buffers or the buffer manager (directly or through execution engine).

1.2.4 Transaction Processing

Queries and other DML actions are grouped into *transactions*, which are units that must be executed atomically and in isolation from other transactions. Any query or modification action can be a transaction by itself. The execution of transactions must be *durable*, the work of a completed transaction will never be lost.

The *transaction manager* accepts *transaction commands* from an application, which tell the transaction manager when transactions begin and end. The transaction manager performs the following tasks.

1. *Logging*: every change in the database is logged separately on disk, by a *log manager*. So when a system failure occurs, a *recovery manager* will be able to examine the log of changes and restore the database to some consistent state.
2. *Concurrency control*: Transactions must appear to execute in isolation. May not execute atomically, but in a way that the net effect is the same as if the transactions executed one at a time.
3. *Deadlock resolution*: resolve deadlocks produced by resource deadlocks between transactions.

1.2.5 The Query Processor

The portion of the DBMS that most affects the performance that the user sees is the *query processor*, represented by two components.

1. *Query compiler*: translates query into internal form called *query plan*, a sequence of operations performed on data. The compiler may use metadata and statistics about the data to decide which sequence of operations is likely to be the fastest.
2. *Execution engine*: executes each of the steps in the chosen query plan.