Course:
Database Management Systems
Credits: 3

Nguyen Thanh Tung
thanhtung@cse.hcmut.edu.vn
cse.hcmut.edu.vn/~thanhtung
Faculty of Computer Science & Engineering
HoChiMinh City University of Technology
Vietnam National University of HoChiMinh City

References

[1] R. Elmasri, S. R. Navathe, *Fundamentals of Database Systems- 4th Edition*, Pearson-Addison Wesley, 2003.

[2] H. G. Molina, J. D. Ullman, J. Widom, *Database System Implementation*, Prentice-Hall, 2000.

[3] H. G. Molina, J. D. Ullman, J. Widom, *Database Systems: The Complete Book,* Prentice-Hall, 2002

[4] A. Silberschatz, H. F. Korth, S. Sudarshan, *Database System Concepts –3rd Edition*, McGraw-Hill, 1999.

References

[1] R. Elmasri, S. R. Navathe, *Fundamentals of Database Systems- 4th Edition*, Pearson-Addison Wesley, 2003.

[2] H. G. Molina, J. D. Ullman, J. Widom, *Database System Implementation*, Prentice-Hall, 2000.

[3] H. G. Molina, J. D. Ullman, J. Widom, *Database Systems: The Complete Book,* Prentice-Hall, 2002

[4] A. Silberschatz, H. F. Korth, S. Sudarshan, *Database System Concepts –3rd Edition*, McGraw-Hill, 1999.

Course outline

- C0. Overview of a DBMS
- C1. Disk Storage, Basic File Structures, and Hashing
- C2. Indexing Structures for Files
- C3. Algorithms for Query Processing and Optimization
- C4. Introduction to Transaction Processing Concepts and Theory
- C5. Concurrency Control Techniques
- C6. Database Recovery Techniques

Course outline - Timetable

- C0. Overview of a DBMS (**w. 1**)
- C1. Disk Storage, Basic File Structures, and Hashing (**w. 1, 2**)
- C2. Indexing Structures for Files (**w. 3, 4, 5**)
- C3. Algorithms for Query Processing and Optimization (**w. 6, 7, 8**)
- C4. Introduction to Transaction Processing Concepts and Theory (**w. 9, 10**)
- C5. Concurrency Control Techniques (**w. 11, 12**)
- C6. Database Recovery Techniques (**w. 13, 14**)

Three parts

- **Storage management**: how secondary storage is used effectively to hold data and allow it to be accessed quickly
- **Query processing**: how queries expressed in a very high-level language such as SQL can be executed efficiently
- **Transaction management**: how to support transactions.

Chapter 0

An Overview of a Database Management System

What is a DBMS?

- The power of database comes from a body of knowledge and technology that has developed over several decades and is embodied in a specialized software called a ***database management system***, or DBMS.
- A DBMS is a powerful tool for creating and managing large amount of data efficiently and allowing it to persist over long periods of time safely.

DBMS Capabilities

The capabilities that a DBMS provides the user are:

- **Persistent Storage**. A DBMS supports the storage of very large amounts of data that exists independently of any processes that are using the data.
- **Programming Interface**. A DBMS allows the user to access and modify data through a powerful query language.
- **Transaction management**. A DBMS supports concurrent access to data, i.e., simultaneously access by many distinct processes (called transaction) at once. To avoid some of the undesirable consequences of simultaneous access, the DBMS supports:
  - isolation
  - atomicity
  - resiliency

Overview of a Database Management System

- In Fig. 1.1. we can see an outline of a complete DBMS.
  - Single boxes represent system components
  - Double boxes represent in-memory data structures.
  - Solid lines indicate control and data flows
  - Dashed lines indicate data flow only.
- At the top level, we suggest that there are two distinct sources of commands to the DBMS:
  - Conventional users and application programs that ask for data or modify data.
  - A DBA (database administrator): a person or persons responsible for the structure (schema) of the database.

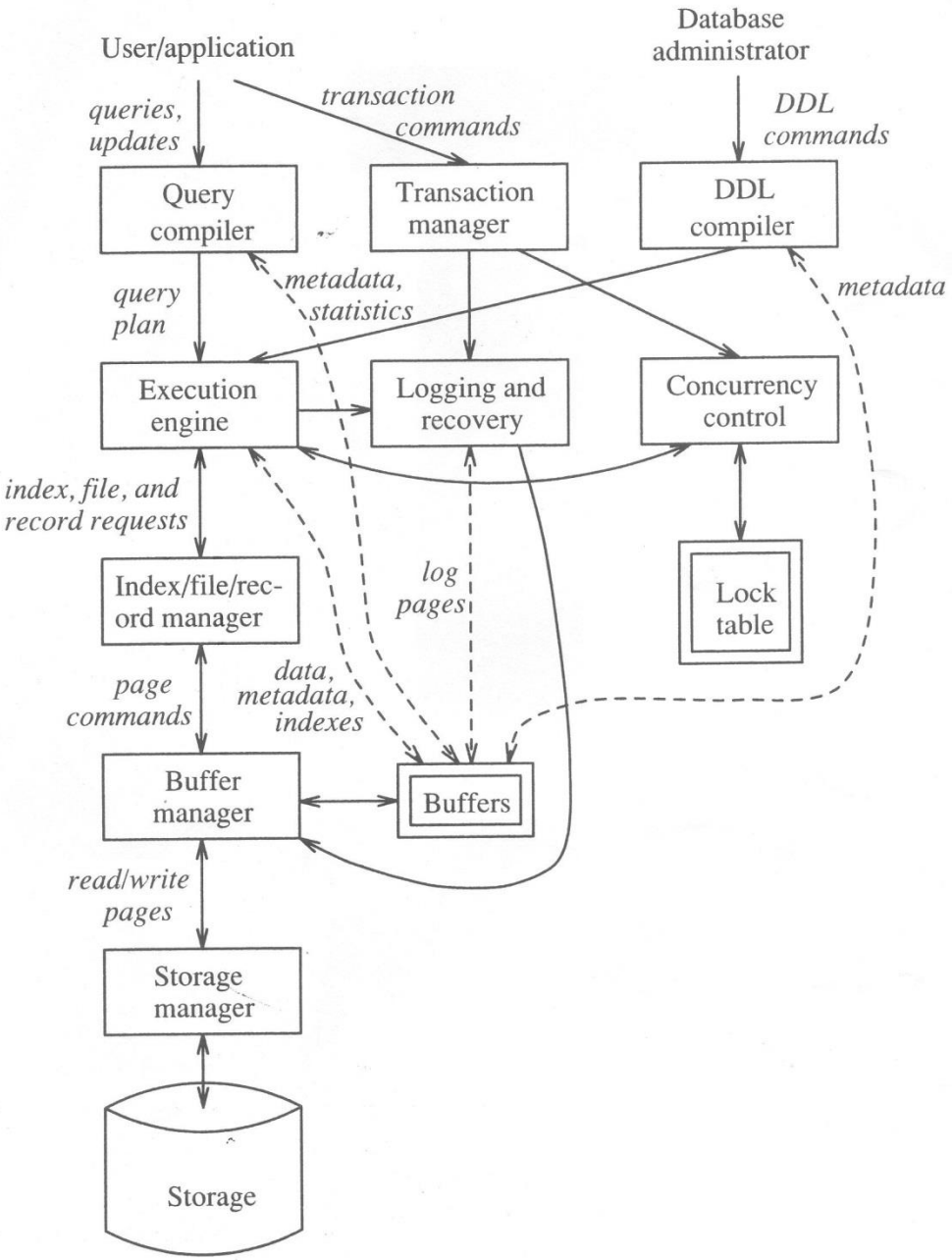


Figure 1.1: Database management system components

Data-Definition Language Commands

- The second kind of commands is the simpler to process and we can see its trail beginning at the upper right side of Fig.1.1.
- E.g., DBA for a university registrar's database might decide that there should be a **relation** with columns for a student, a course that student has taken and a grade for that student in that course. This structure and constraint information is all part of the **schema** of the database. It is entered by the DBA, who needs special authority to execute schema-altering commands.
- These schema-altering ***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 **metadata**, that is, the schema information for the database.

Overview of Query Processing

- The great majority of interactions with the DBMS follow the path on the left side of Fig. 1.1.
- A user or application program initiates some action that does not affect the schema of the database, but may affect the content of the database (if the action is a **modification** command) or will extract data from the database (if the action is a **query**).
- There are two paths along which user actions affect the database:
  - Answering the query
  - Transaction processing.

Answering the query

- The query is parsed and optimized by a **query compiler**. The resulting query plan is passed to the **execution engine**.
- The execution engine issues a sequence of requests for small pieces of data, typically tuples of a relation, to a **resource manager** that knows about data files, the format and size of records in those files and index files.
- The requests for data are translated into pages and these requests are passed to **buffer manager**. Buffer manager's task is to bring appropriate portions of the data from secondary storage to main-memory buffers.
- Normally, the page or "disk blocks" 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 command, but more typically, DBMS issues commands directly to the **disk controller**.

Transaction processing

- Queries and other actions are grouped into **transaction**s, which are units that must be executed **atomically** and in **isolation**; often each query or modification action is a transaction 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 after completion of the transaction.
- The **transaction processor** consists of two parts:
  - A **concurrency-control manager** (scheduler) responsible for assuring atomicity and isolation of transaction,
  - A **logging and recovery manager** responsible for the durability of transactions.

Main-memory buffers and Buffer Manager

- The data of a database normally resides in secondary storage (magnetic disk). However, to perform any operation on data, that data must be in main memory.
- Buffer manager is responsible for partitioning the available main memory into **buffers**, which are page-sized regions into which *disk blocks* can be transferred.
- 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.

Buffer manager

The kinds of information that various components in DBMS may need include:

- **Data**: the content of the database itself
- **Metadata**: the database schema that describes the structure of, and the constraints on, the database.
- **Statistics**: information gathered and stored by the DBMS about data properties such as the size of, and the values in various relations or other components of the database.
- **Indexes**: data structures that support efficient access to the data.

Transaction Processing

- It's 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.
- Besides, a DBMS offers the guarantee of durability: that 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
  - information about the expectations of the application.

Transaction processor's tasks

The transaction processor performs the tasks:

- **Logging**: In order to assure durability, every change in the database is logged separately on disk. The **log manager** follows one of several policies designed to assure that no matter when a system failure or "crash" occurs, the **recovery manager** will be able to examine the log of changes and restore the database to some consistent state. The log manager initially writes the log in buffers and negotiates with the **buffer manager** to make sure that buffers are written to disk at appropriate times.

Transaction processor's tasks (cont.)

- **Concurrent control**: Transactions must appear to execute in isolation. But in most systems, there will be many transactions executing at once. Thus, the **scheduler** (*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 been executed in their entirety, one-at-a-time.
  - A typical scheduler does its work by maintaining **locks** on certain pieces of the database. These locks prevent two transactions from accessing the same piece of data in ways that interact badly.
  - Locks are stored in a main-memory **lock table**. The scheduler affects the execution of queries and other database operations by forbidding the execution engine from accessing locked parts of the database.

Transaction processor's tasks (cont.)

- **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 duty to ***intervene*** and ***cancel*** one or more transactions to let the others proceed.

The Query Processor

- The part of DBMS that most affects the performance that the user sees is the **query processor**.
- The query processor consists of two components: **query compiler** and **execution engine**.
- The **query compiler**, translates the query into an internal form called a query plan. A query plan is a sequence of operations to be performed on the data. Often the operations in a query plan are “relational algebra” operations.
- The query compiler consists of 3 major units: a **query parser**, a **query preprocessor**, and a **query optimizer**.

Query Compiler's components

- **A query parser**, which builds a tree structure from the textual form of the query.
- **A query preprocessor**, which performs semantic checks on the query (e.g., making sure all relations mentioned by the query actually exist), and performing some tree transformations to turn the **parse tree** into **tree of algebraic operators** representing the initial query plan.
- **A query optimizer**, which transforms the initial query plan into the best available sequence of operations on the actual data.

Note: The query compiler uses *metadata* and *statistics* about the data to decide which sequence of operations is likely to be the fastest.

Execution Engine

- The **execution engine** 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 the database changes are properly logged.

(Relational) DBMSs in Practice

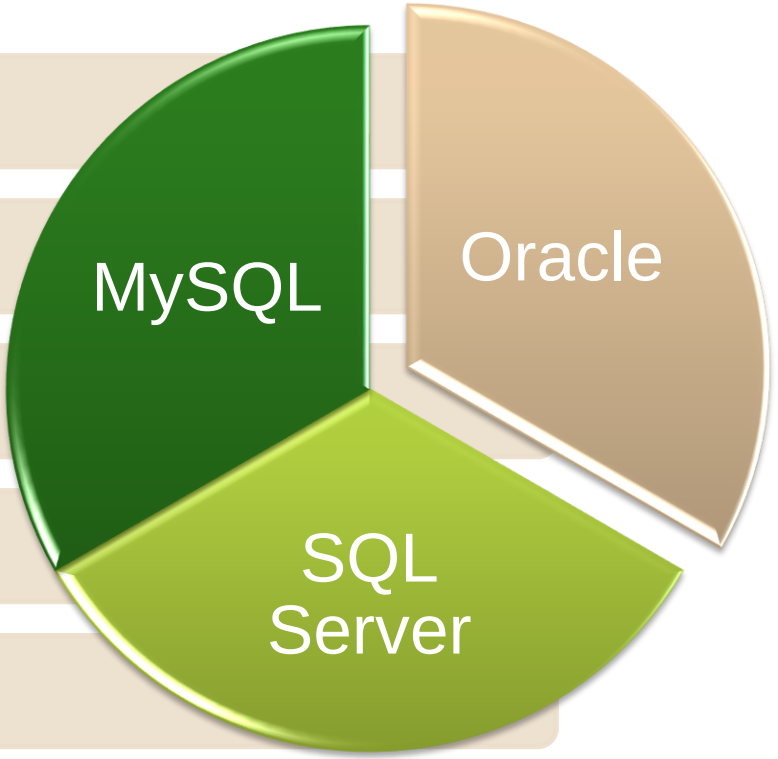
- MySQL
- Oracle
- MS SQL Server
- IBM DB2
- …

Assignments

- File structure
  - Oracle
- Index
  - Index in Oracle
  - R-Tree/ Hilbert R-Tree
  - Bitmap index
  - How to implement in Oracle
- Cache
  - In Oracle
  - In MySQL
- Query Processing
  - In Oracle
  - In SQL Server
- Transaction
  - In Oracle
  - In SQL Server
- Recovery
  - In Oracle
  - In SQL Server
- Distributed DBMS
  - Oracle

Assignments - else

File Structure	• Record • File
Index	• B^+-Tree • Bitmap • ?
Cache	• How
Query Processing	• Execution Plan • ?
Transaction	• ?
Recovery	• ?
Distributed DBMS	• ?



Assessment

- Midterm: 50%
  - Preliminary tests in class: 10%
    - Test 1 (c. 1 + 2; w. 5): 5%
    - Test 2 (c. 3 p.2 + 4 + 5; w. 13): 5%
  - Midterm exam: 25%
    - 20 questions in 60 mins: 0.5 point/question
    - Reviews: c.0-3
  - Individual assignment: 15%
    - Deadline for assignment submission: w. 14
    - Oral presentation for an added bonus (max 10%): w. 14
- Final exam: 50%
  - 45 questions in 120 mins: 0.25 point/question
  - Reviews: c. 0-6

Success in the course = { Midterm * 50% + Final exam * 50% ≥ 5.0 }