Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering

Overall Introduction to Database Management Systems

Database Management Systems (CO3021)

Computer Science Program

Dr. Võ Thị Ngọc Châu

(chauvtn@hcmut.edu.vn)

Semester 1 – 2018-2019

Course outline

- **Overall Introduction to Database Management Systems**
- Chapter 1. Disk Storage and Basic File Structures
- Chapter 2. Indexing Structures for Files
- Chapter 3. Algorithms for Query Processing and Optimization
- Chapter 4. Introduction to Transaction Processing Concepts and Theory
- Chapter 5. Concurrency Control Techniques
- Chapter 6. Database Recovery Techniques

References

- [1] R. Elmasri, S. R. Navathe, Fundamentals of Database Systems- 6th Edition, Pearson- Addison Wesley, 2011.
  - ***R. Elmasri, S. R. Navathe, Fundamentals of Database Systems- 7th Edition, Pearson, 2016.***
- [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.
- [Internet] …

Content

- What is a database management system (DBMS)?
- System architecture
- History of DBMS development
- Classification of database management systems
- When should(not) we use the DBMS approach?
- Human resource related to a DBMS

What is a database management system (DBMS)?

Database Management Systems

-***System***: a set of connected items or devices which operate together, a set of computer equipment and programs used together for a particular purpose

-***Management***: the control and organization of something

-***Database***: a collection of related data with an implicit meaning

What is a database management system (DBMS)?

- The power of database comes from a body of knowledge and technology that has developed over several decades and is embodied (*included as part*) 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.

System Architecture

- An outline of a DBMS in the following figure:
  - 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, two sources of commands:
  - Queries/updates, transaction commands
    - Conventional users and application programs that ask for data or modify data.
  - DDL commands
    - A database administrator (DBA) responsible for the structure (schema) of the database.

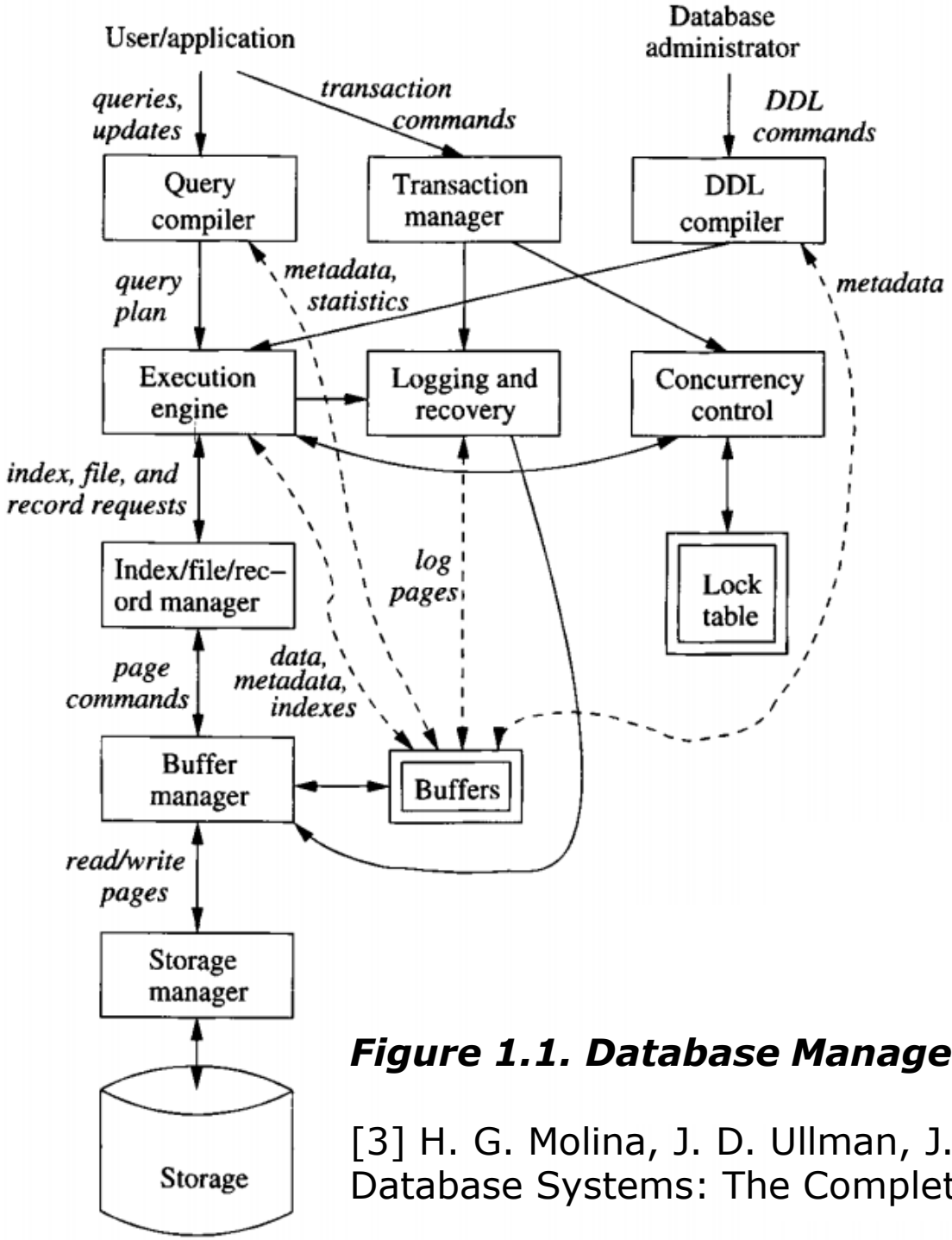


Figure 1.1. Database Management System Components

[3] H. G. Molina, J. D. Ullman, J. Widom,
Database Systems: The Complete Book, Prentice-Hall, 2009

Data-Definition Language Commands

```
CREATE TABLE courses (
        cid      VARCHAR2(6) PRIMARY KEY,
        cname    VARCHAR2(50) NOT NULL,
        credit   NUMBER
);
```

- These schema-altering ***DDL commands*** are parsed by a **DDL compiler**
- then passed to the **execution engine**,
- then goes through the **index/file/record manager** to alter the **metadata**, that is, the schema information for the database.

Queries/Updates with Query Processing

- Queries/Updates with DML statements

```
SELECT cname
FROM courses
WHERE cid = 'CO3021';

UPDATE courses
SET credit = 3
WHERE cid = 'CO3021';
```

- DML statements are handled by two separate subsystems:
  - Answering the query
  - Transaction processing

Queries/Updates with Query 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 commands, but more typically, DBMS issues commands directly to the **disk controller**.

Queries/Updates with Query Processing

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.

Main-memory buffers and Buffer Manager

The kinds of information in the *buffer* 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.

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**.

The Query Processor – Query Compiler

- **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.

The Query Processor – 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.

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 Processing

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 Processing

- **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 Processing

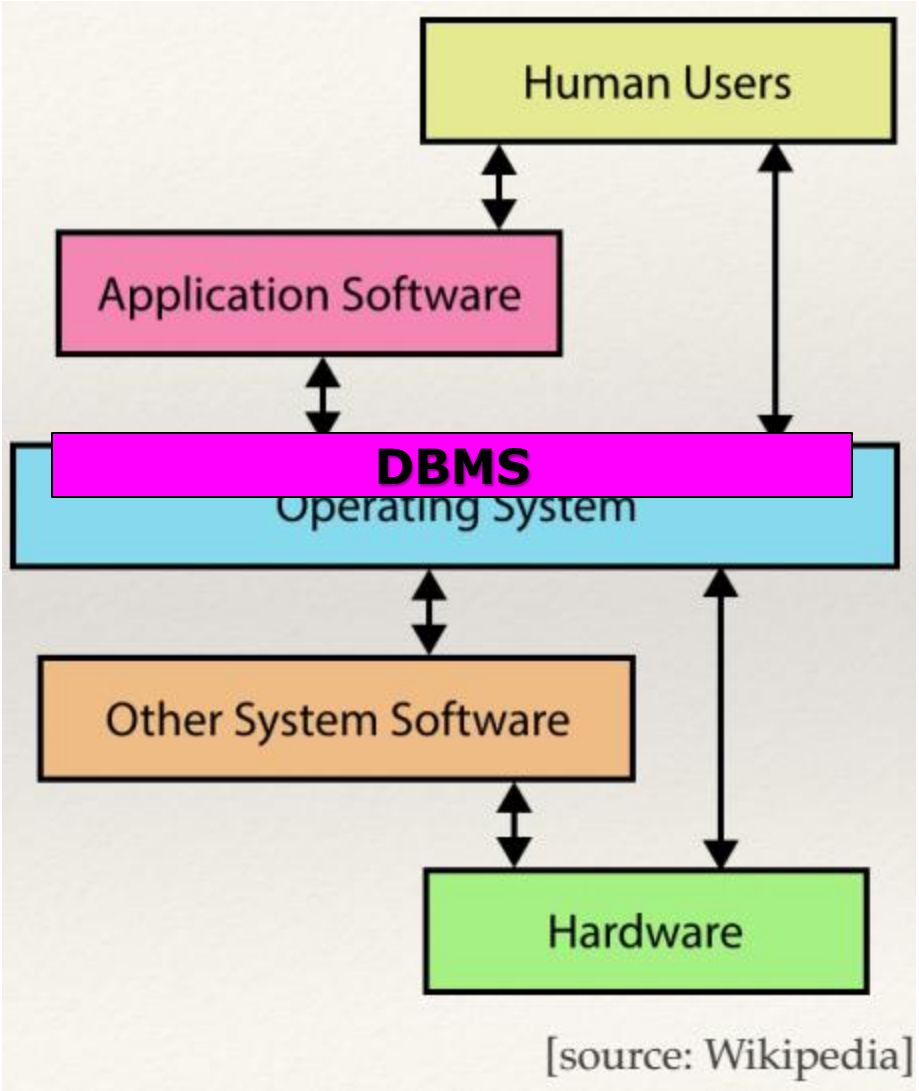
- **Deadlock resolution**: As transactions compete for resources through the locks that the scheduler grants, they can get into a situation where none can proceed because each needs something another transaction has. The transaction manager has the duty to ***intervene*** and ***cancel*** one or more transactions to let the others proceed.

DBMS Capabilities

The capabilities that a DBMS provides its users 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 (*recovery*)

Where is a DBMS?



History of DBMS development

- 1960s, navigational DBMSs
  - IBM's IMS with the hierarchical model,
  - IDMS with the CODASYL network model, …
- 1970s-late 1980s, relational DBMSs with SQL
  - Oracle,
  - MS SQL Server,
  - IBM's DB2,
  - MySQL, …
- 1990s, object-oriented DBMSs (object, object-relational)
  - Oracle,
  - PostgreSQL,
  - Informix, …
- 2000s, NoSQL and NewSQL
  - XML DBMSs: Oracle Berkely DB XML, …
  - NoSQL DBMSs: MongoDB, Hbase, Cassandra, …
  - NewSQL DBMSs: ScaleBase, VoltDB, …

Classification of database management systems

- DBMS classification based on:
  - Data model
    - Hierarchical, network, relational, object, object-relational, XML, document-based, graph-based, column-based, key-value, …, NewSQL data models
  - The number of users
    - Single-user systems vs. multiuser systems
  - The number of sites
    - Centralized vs. distributed
  - Cost
  - Purpose
    - General purpose vs. special purpose

When should(not) we use the DBMS approach?

- Should
  - Controlling Redundancy
  - Restricting Unauthorized Access
  - Providing Persistent Storage for Program Objects
  - Providing Storage Structures and Search Techniques for Efficient Query Processing
  - Providing Backup and Recovery
  - Providing Multiple User Interfaces
  - Representing Complex Relationships among Data
  - Enforcing Integrity Constraints
  - Permitting Inferencing and Actions Using Rules and Triggers
  - Additional Implications of Using the Database Approach
    - Potential for Enforcing Standards
    - Reduced Application Development Time
    - Flexibility
    - Availability of Up-to-Date Information
    - Economies of Scale

When should(not) we use the DBMS approach?

- Should *not*
  - Simple, well-defined database applications that are not expected to change at all
  - Stringent (*severe*), real-time requirements for some application programs that may not be met because of DBMS overhead
  - Embedded systems with limited storage capacity, where a general-purpose DBMS would not fit
  - No multiple-user access to data

Human resource related to a DBMS

- Database administrators
- Database end-users
- Database designers
- System analysts
- Application developers
- DBMS developers

Summary

- Database management systems
  - *Relational* database management systems
  - Powerful tools for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely
- Three main parts of a DBMS
  - Storage management
  - Efficiently query processing
  - Transaction management
- Make an appropriate choice of a DBMS!

Overall Introduction to Database Management Systems

Check for understandings

- 1. What is a database management system?
- 2. Describe the capabilities of a DBMS
- 3. Describe the processing of DDL commands
- 4. Describe the processing of DML commands
- 5. Describe transaction processing
- 6. Describe the buffer and give examples for its content

Check for understandings

- 7. Classify the following DBMSs
  - Oracle 8i
  - Oracle 12c
  - MS SQL Server 2008
  - MySQL
  - MongoDB
  - Apache Cache
  - IBM DB2