CSH2D3 - Database System

01 | Database System Concept and Architecture

Preface

In preceding course (database modelling), we have emphasized the higher-level models of a database (*conceptual* or *logical* level).

We viewed the database, in the relational model, as a collection of tables. Indeed, the logical model of the database is the correct level for database *users* to focus on.

The goal of a database system is to simplify and facilitate access to data;

users of the system should not be burdened unnecessarily with the physical details of the implementation of the system.

In this course we probe below the higher levels as we describe various methods for implementing the data models and languages.

Goals of the Meeting

01

Students understand the advantages of using database systems and how the database system evolve from time to time

02

Students understand the components of database engine, how they worked, and who are the users

03

Students know various database architecture

Outline

- Database Management System
- Database Engine
- Database System Architecture

Database Management System

Characteristics of the Database Approach

Advantages of Using DBMS Approach

History of Database Systems

Database Management System (DBMS)

DBMS contains information about a particular enterprise

- Collection of interrelated data
- Set of programs to access the data
- An environment that is both *convenient* and *efficient* to use

Database Applications:

- Banking: transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Databases can be very large.

Characteristics of the Database Approach

- Self-describing nature of a database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing

Advantages of Using DBMS Approach

Controlling Redundancy

Restricting Unauthorized Access

Providing Persistent Storage for Program Objects

Providing Storage Structures and Search Techniques for Efficient Query Processing

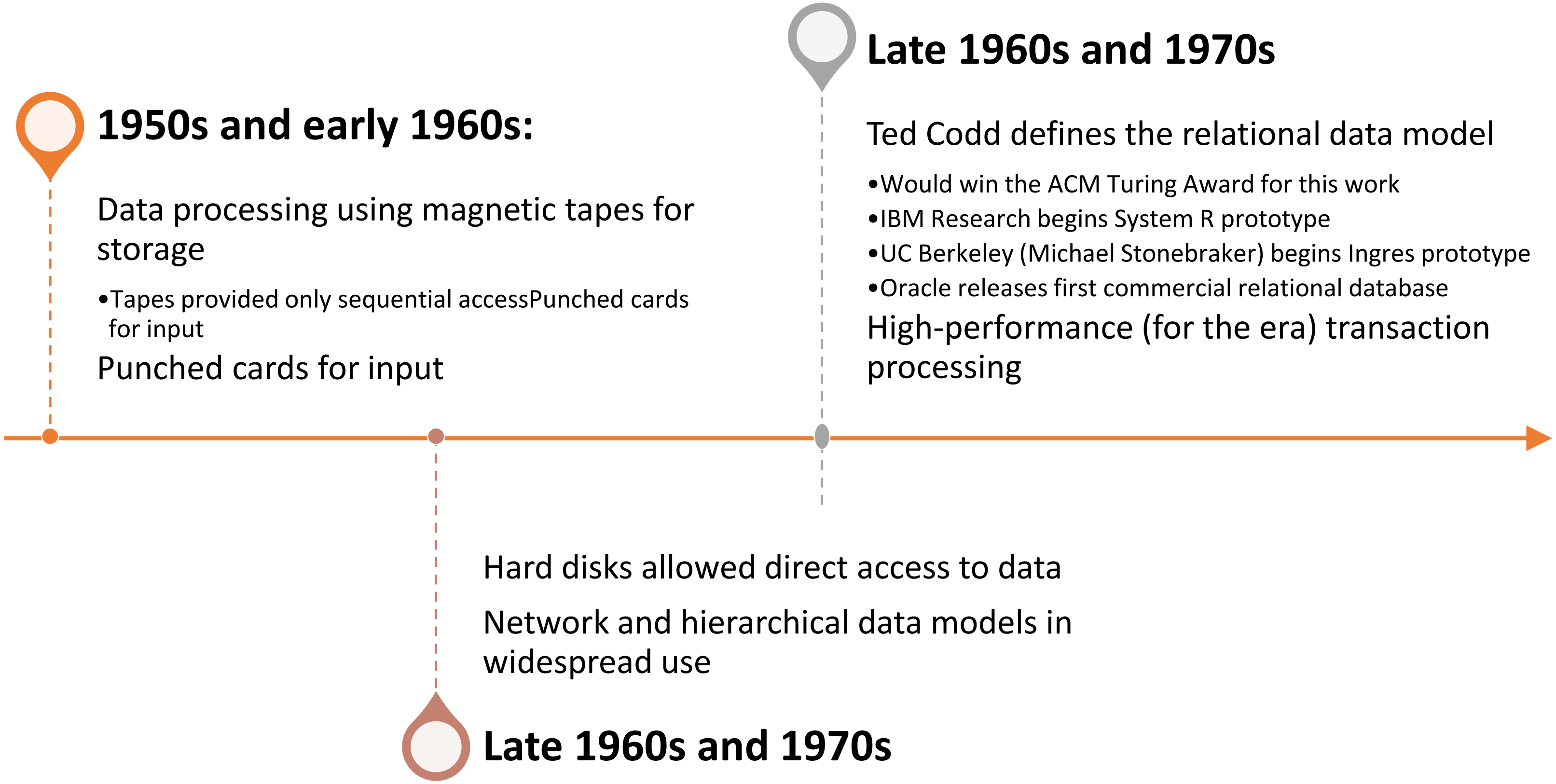
Providing Backup and Recovery

Providing Multiple User Interfaces

Advantages of Using DBMS Approach (cont.)

- 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

History of Database Systems



1950s and early 1960s:

Data processing using magnetic tapes for storage

- Tapes provided only sequential accessPunched cards for input

Punched cards for input

Late 1960s and 1970s

Hard disks allowed direct access to data

Network and hierarchical data models in widespread use

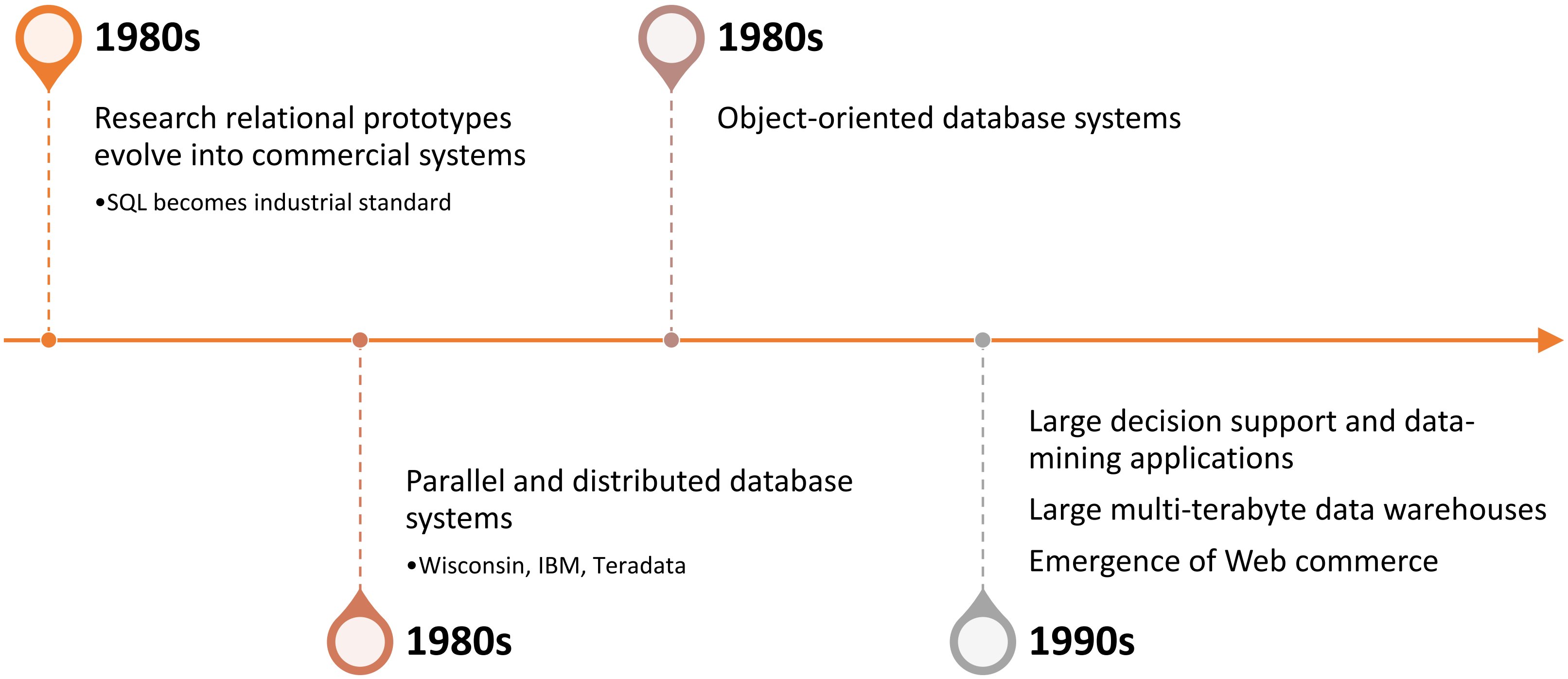
Late 1960s and 1970s

Ted Codd defines the relational data model

- Would win the ACM Turing Award for this work
- IBM Research begins System R prototype
- UC Berkeley (Michael Stonebraker) begins Ingres prototype
- Oracle releases first commercial relational database

High-performance (for the era) transaction processing

History of Database Systems (Cont.)



1980s

Research relational prototypes evolve into commercial systems

- SQL becomes industrial standard

1980s

Parallel and distributed database systems

- Wisconsin, IBM, Teradata

1980s

Object-oriented database systems

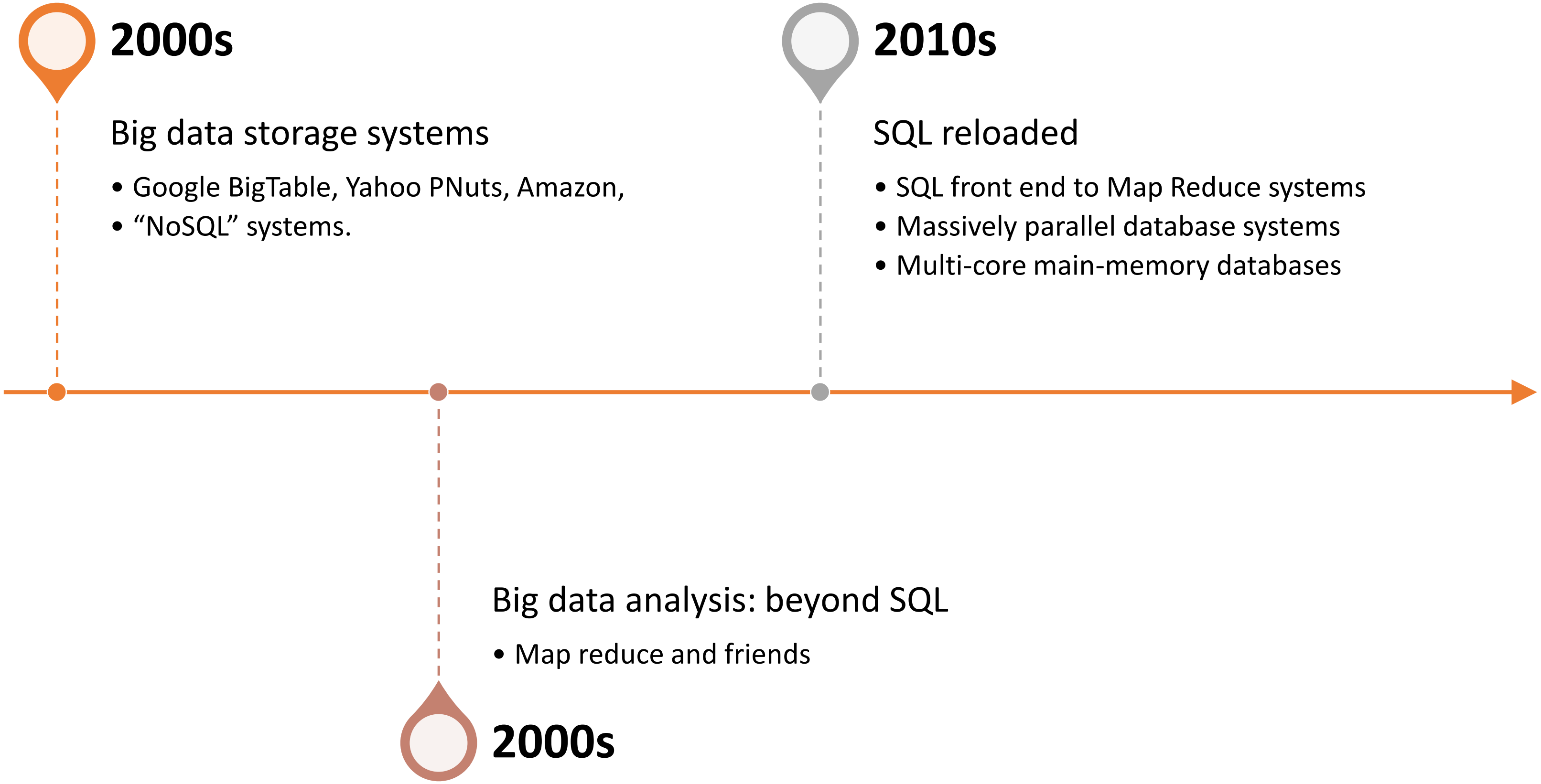
1990s

Large decision support and data-mining applications

Large multi-terabyte data warehouses

Emergence of Web commerce

History of Database Systems (Cont.)



2000s

Big data storage systems

- Google BigTable, Yahoo PNuts, Amazon,
- “NoSQL” systems.

2000s

Big data analysis: beyond SQL

- Map reduce and friends

2010s

SQL reloaded

- SQL front end to Map Reduce systems
- Massively parallel database systems
- Multi-core main-memory databases

Database Engine

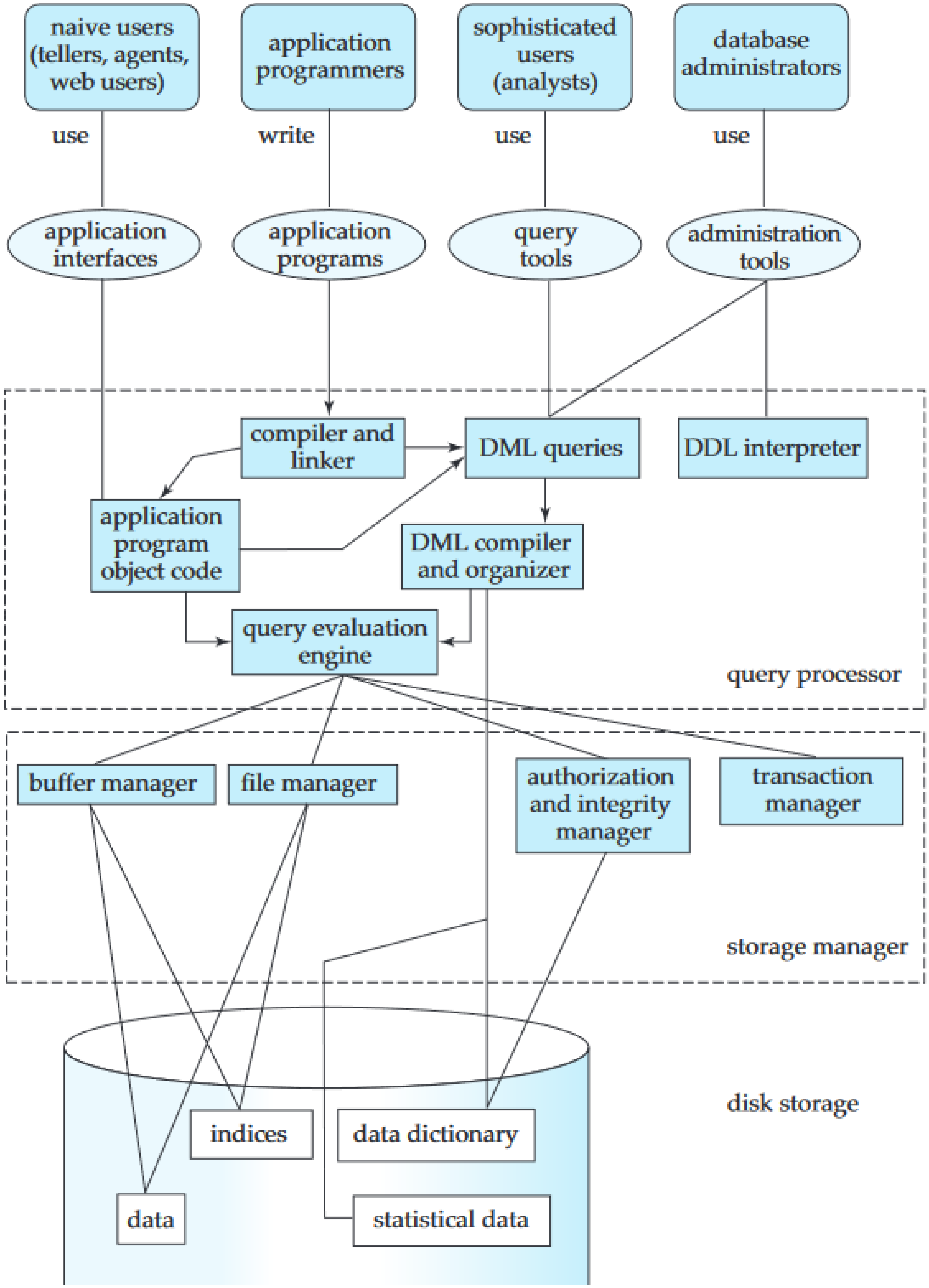
Storage manager

Query processor

Transaction management

Database Engine

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be divided into
    - The storage manager,
    - The query processor component,
    - The transaction management component.



Storage Manager

- A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data

Component of Storage Manager

- **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
- **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
- **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

Disk Storage

The storage manager implements several data structures as part of the physical system implementation:

- **Data files**, which store the database itself.
- **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.
- **Indices**, which can provide fast access to data items. A database index provides pointers to those data items that hold a particular value. For example, we could use an index to find the instructor record with a particular ID, or all instructor records with a particular name
- **Hashing** is an alternative to indexing that is faster in some but not all cases.

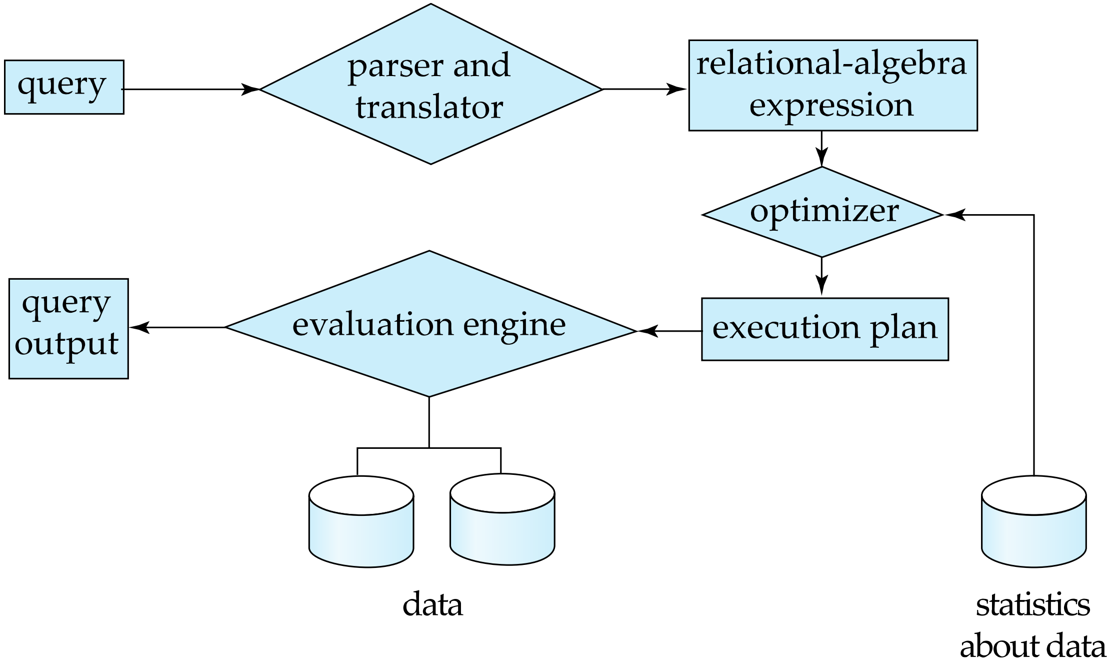
Query Processor

The query processor components include:

- **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.
- **DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands
  - A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization**; that is, it picks the lowest cost evaluation plan from among the alternatives.
- **Query evaluation engine**, which executes low-level instructions generated by the DML compiler

Query Processing

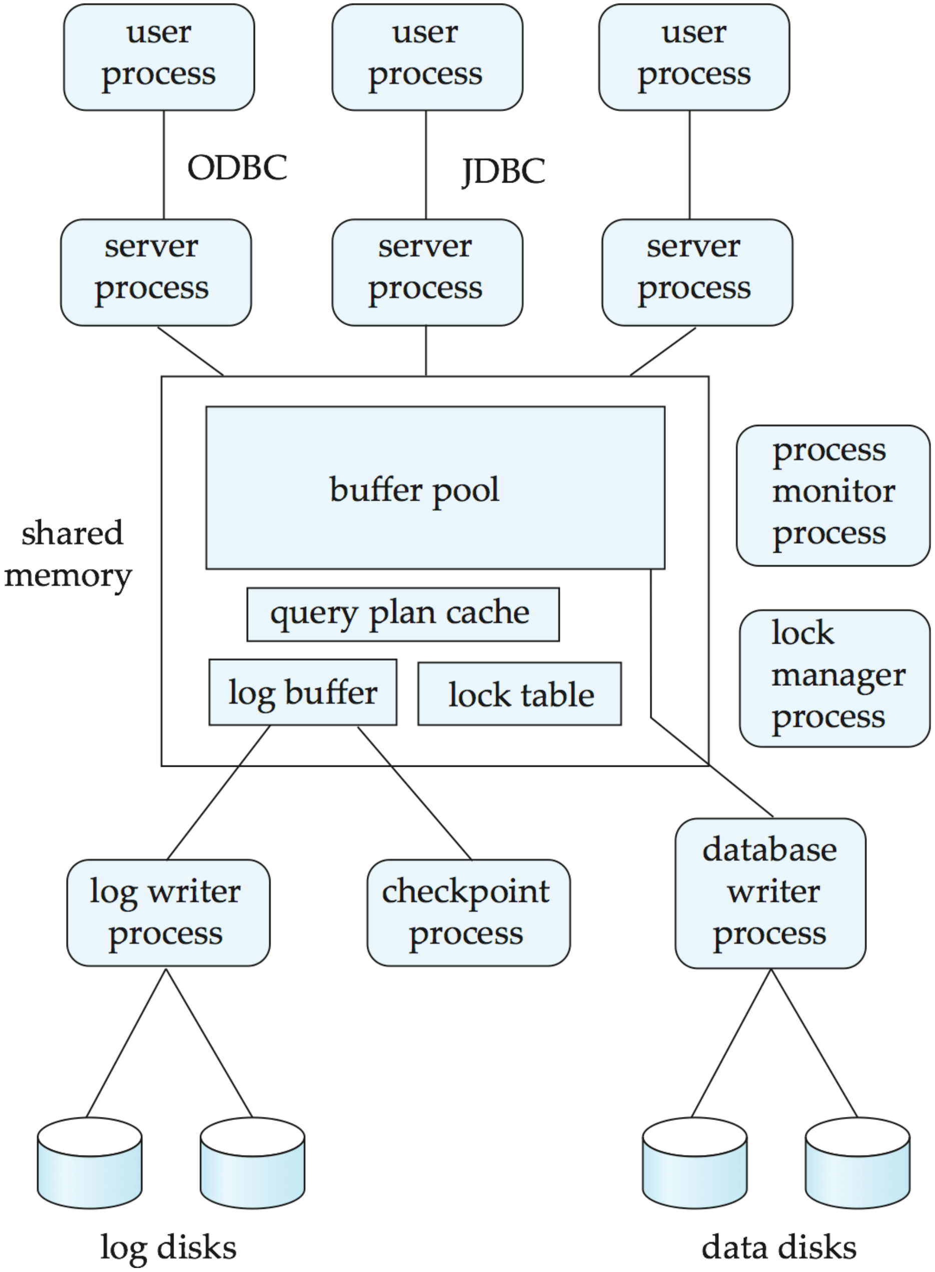
1. Parsing and translation
2. Optimization
3. Evaluation



Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Transaction System Processes



Transaction System Process Structure

- A typical transaction server consists of multiple processes accessing data in shared memory.
- Server processes
  - These receive user queries (transactions), execute them and send results back
  - Processes may be **multithreaded**, allowing a single process to execute several user queries concurrently
  - Typically multiple multithreaded server processes
- Lock manager process
- Database writer process
  - Output modified buffer blocks to disks continually

Transaction System Processes Structure (Cont.)

- Log writer process
  - Server processes simply add log records to log record buffer
  - Log writer process outputs log records to stable storage.
- Checkpoint process
  - Performs periodic checkpoints
- Process monitor process
  - Monitors other processes, and takes recovery actions if any of the other processes fail
    - E.g., aborting any transactions being executed by a server process and restarting it

Transaction System Processes Structure (Cont.)

- Shared memory contains shared data
  - Buffer pool
  - Lock table
  - Log buffer
  - Cached query plans (reused if same query submitted again)
- All database processes can access shared memory
- To ensure that no two processes are accessing the same data structure at the same time, databases systems implement **mutual exclusion** using either
  - Operating system semaphores
  - Atomic instructions such as test-and-set
- To avoid overhead of interprocess communication for lock request/grant, each database process operates directly on the lock table
  - instead of sending requests to lock manager process
- Lock manager process still used for deadlock detection

Database Users

Application programmers

interact with system through DML calls

Sophisticated users

form requests in a database query language

Specialized users

write specialized database applications that do not fit into the traditional data processing framework

Naïve users

Invoke one of the permanent application programs that have been written previously

E.g. people accessing database over the web, bank tellers, clerical staff

Database Administrator

Database Administrator

A person who has central control over the system is called a **database administrator (DBA).**

Functions of a DBA include:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database

Database Architecture

Database Architecture

Centralized databases

One to a few cores, shared memory

Client-server

One server machine executes work on behalf of multiple client machines.

Parallel databases

Many core shared memory

Shared disk

Shared nothing

Distributed databases

Geographical distribution

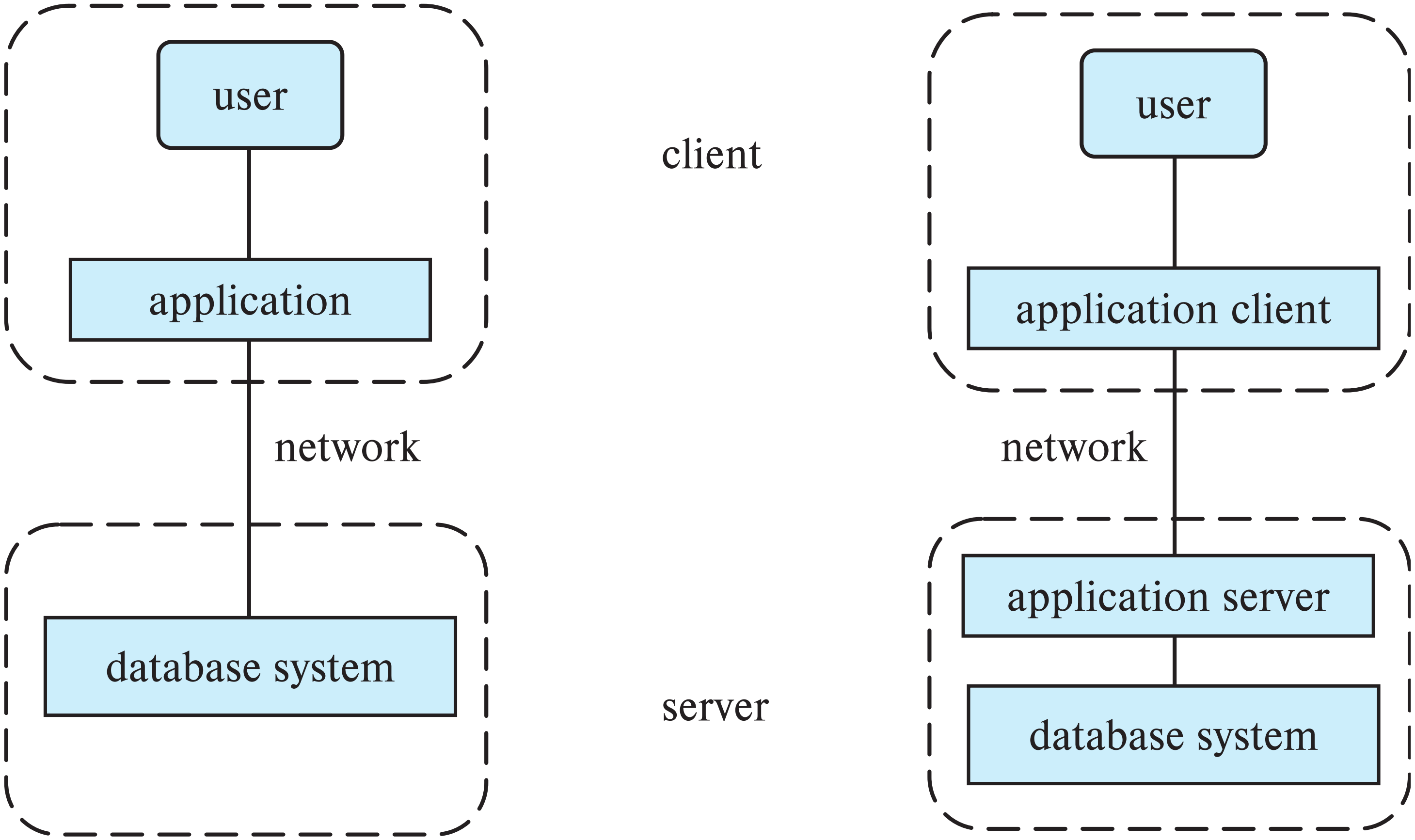
Schema/data heterogeneity

Database Application

Database applications are usually partitioned into two or three parts

- Two-tier architecture -- the application resides at the client machine, where it invokes database system functionality at the server machine
- Three-tier architecture -- the client machine acts as a front end and does not contain any direct database calls.
  - The client end communicates with an application server, usually through a forms interface.
  - The application server in turn communicates with a database system to access data.

Two-tier and three-tier architectures



(a) Two-tier architecture

(b) Three-tier architecture

References

Silberschatz, Korth, and Sudarshan. *Database System Concepts* – 7th Edition. McGraw-Hill. 2019.

Slides adapted from Database System Concepts Slide.

Source: https://www.db-book.com/db7/slides-dir/index.html

Elmasri, Navathe, “Fundamental of Database Systems”, Seventh Edition, Pearson, 2015.

Any
Questions