MDA104 Introduction to Databases

1. Introduction

Vlastislav Dohnal

Credits

- Slides are part of the database bible:
  - Database System Concepts, Seventh Edition. Avi Silberschatz, Henry F. Korth, S. Sudarshan.
  - https://db-book.com/
- Experience from courses of Faculty of Informatics, Masaryk University
  - PB168 - Fundamentals of Database and Information Systems
  - PB154 - Fundamentals of Database Systems

Outline

- Purpose
- Looking at the data
- Database languages
- Architecture of Database Systems
  - Data models
  - Structure of the database system
  - Relational database
  - Object databases
  - History
- Database design
- Data storage and querying
- Transaction processing
- Users and database administrators

Database system

- Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - A collection of interrelated data
  - Set of programs to access the data
  - An environment that is both convenient and efficient to use
- Database applications
  - Banking – all transactions we know
  - Airlines – booking, planning
  - Universities – registration, enrolment, evaluation, ...
  - Sales – customers, products, sales
    - Online - Shipment tracking, customized recommendations
  - Production – production, inventory, orders, transport, suppliers
  - State administration – population register, applications, tax administration, ...
- Databases can be found (almost) everywhere

Purpose of the database system

- In the early days, database applications were built directly on top of file systems,
- which leads to several disadvantages:
  - Redundancy and data inconsistency
    - Different file formats, duplicating information into multiple files
  - Data is difficult to access
    - Need to write a new program to carry out each new task
  - Data isolation
    - Separate files, different formats
  - Integrity issues
    - Integrity constraints are implemented in user programs, e.g. account balance >= 0.
    - They are hidden in programs, they are not "explicitly presented" anywhere
    - Difficult to add a new restriction or change an existing one

Purpose of the database system

- Disadvantages of storing data directly in files:
  - Atomicity of data updates
    - Outages can cause inconsistent state
      - Only some tasks were performed
    - E.g., transfer of the amount from account to account – must be done in complete or not at all
  - Concurrent multi-user access
    - Important for system performance
    - Inconsistencies can be created without concurrent access control
      - For example, two users access and update the balance of the same account
  - Restricting access to data (data security)
    - Difficult to restrict access to selected data (part of a file)
- Database systems
  - = offer solutions to these challenges

Data models

- Data model = a set of data description tools
  - and the relationships between them
  - Data semantics
  - integrity constraints

  and data manipulation

- Examples
  - Relational model
  - Entity-relational model (especially in database design)
  - Object-relational model, object-oriented model
  - Model for Semi-Structured Data (XML)
  - Other older models
    - Hierarchical model
    - Network model

Relational model

- Example of data in a tabular representation
  - can be stored in a relational database

Columns

customer_id	customer_name	customer_street	customer_city	account_number
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-101
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-201
677-89-9011	Hayes	3 Main St.	Harrison	A-102
182-73-6091	Turner	123 Putnam St.	Stamford	A-305
321-12-3123	Jones	100 Main St.	Harrison	A-217
336-66-9999	Lindsay	175 Park Ave.	Pittsfield	A-222
019-28-3746	Smith	72 North St.	Rye	A-201

Lines

Ted Codd
Turing Award 1981

Relational database example

customer_id	*customer_name*	*customer_street*	*customer_city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account_number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer_id	*account_number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Data definition language (DDL)

- Provides expressions for the definition of the database schema

  E.g.:

  ```
  create table account (
      account-number  char(10),
      balance         integer
  );
  ```
- The DDL compiler generates a set of tables
  - These are stored in a data dictionary
- A **data dictionary** contains metadata (data about data)
  - Database schema
  - Integrity constraints
    - Domain restrictions
    - Referential integrity (foreign key)
    - Assertions
  - Access rights
  - Data storage methods

Data manipulation language (DML)

- A language for accessing and manipulating data organized in a specific model
  - Often referred to as query language
- Two language classes
  - Procedural – the user specifies both the data they want and how to access it.
  - Declarative (non-procedural) – the user specifies only data without a procedure to retrieve it.
- SQL = the most common (query) language

SQL

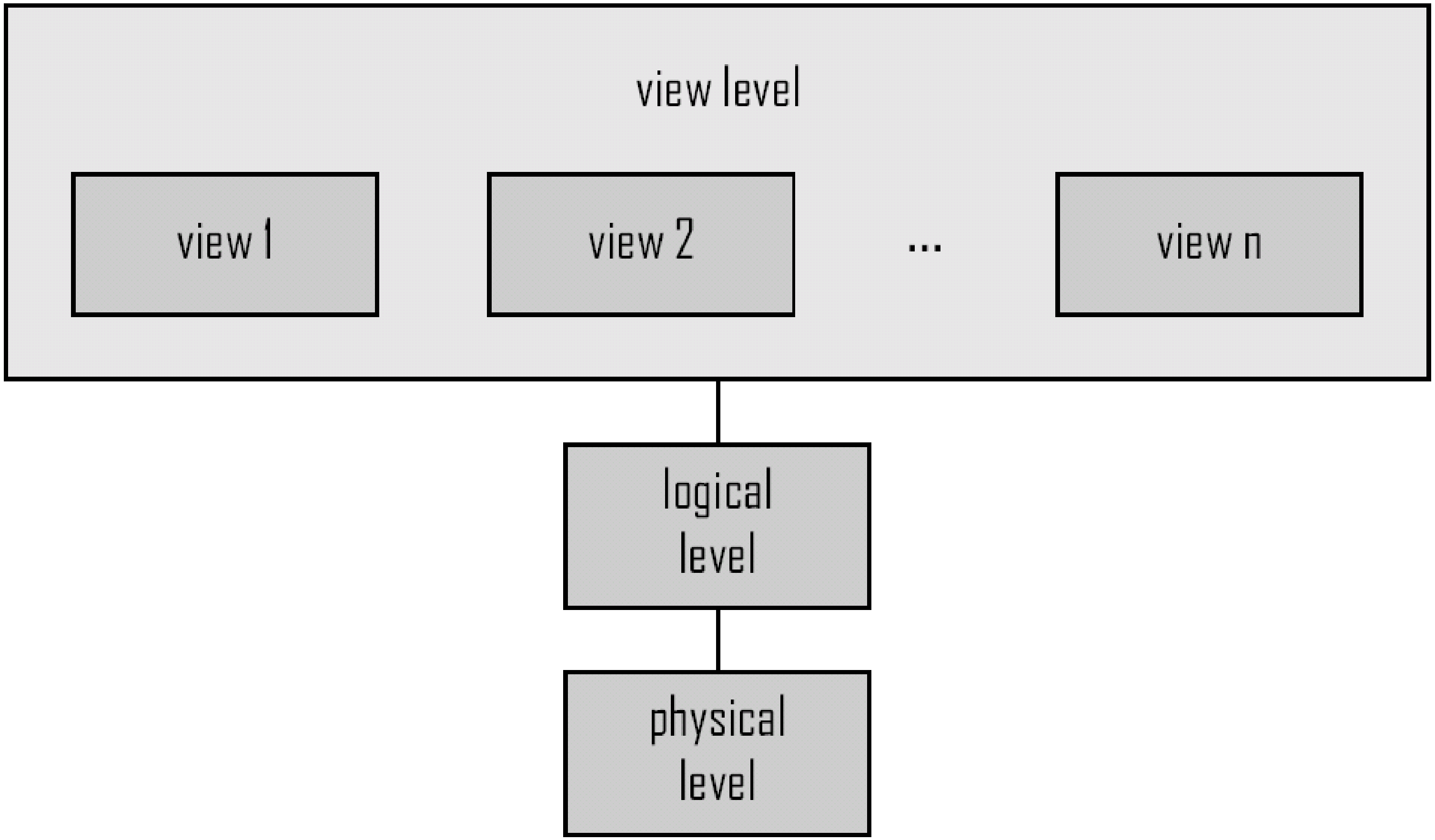
- SQL – Structured Query Language
- Frequently used non-procedural language
  - E.g.: Find the name of the customer with id 7465

    select *customer_name*
    from *customer*
    where *customer_id* = 7465
  - E.g. List the balances of all accounts owned by a customer having id 7465

    select *account.balance*
    from *depositor*, *account*
    where *depositor.customer_id* = 7465 **and**
    depositor.account_number = *account.account_number*
- Applications generally access the database using
  - Extension of the programming language by encapsulation of SQL
  - Application programming interface (e.g. ODBC, JDBC)
    - allows sending SQL expressions

View of data

- Architecture of the database system



Data abstraction levels

- Physical level
  - Describes how a data record (e.g. an order) is stored in memory
- Logical level
  - Describes the structure of data stored in a database and the relationships between data.

```
type order = record
        order_id : integer;
        created : date;
        goods : string;
        customer_id : integer;
    end;
```

```
class order {
        int order_id;
        date created;
        string goods;
        int customer_id;
}
```

- View Level (Application)
  - Simplification for applications / application users
  - Hiding part of the data, e.g., employee’s salary (for security reasons)
  - Making summary data available

Data abstraction levels

- Physical data independence
  - = Ability to change the physical schema without changing the logical schema
- Applications are dependent on a logical scheme or views
- Define the interface between the levels
  - Changes in one level affected the other levels as little as possible

Database Instance and Schema

- **Database schema**
  - = database structure
    - E.g. collections of customers, accounts and relationships between them
  - Logical schema
    - the overall logical structure of the database
    - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
  - Physical schema
    - the overall physical structure of the database
- **Database,** or **database instance**
  - Current data content (at a given time)
- **Database system**
  - Implementation of a specific data model including additional software
  - E.g. MariaDB, PostgreSQL, ….

Database design

- Database design process:
  1. Logical design
     - Decide how the database schema should look
     - Requires finding suitable relational schemes
     - Customer's decision
       - What information (descriptions) will we store in the database?
     - IT decisions
       - What relations we will create and what attributes they will have
  2. Physical design
     - Deciding on the physical layout of the database
       - What disks to use, logical disk arrays, what indexes to build, …?

Database design example

- Requirements
  - We want to keep records of university teachers and their affiliation to the faculty
  - A teacher has a name and a salary
  - The faculty has its building and budget
- Example of data
  - Einstein earns 95000 and belongs to Physics in Watson with a budget of 70000.

Database design example

- Is this proposal correct?

Table *instructor*

name	*salary*	*dept_name*	*building*	*budget*
Einstein	95000	Physics	Watson	70000
Wu	90000	Finance	Painter	120000
El Said	60000	History	Painter	50000
Katz	75000	Comp. Sci.	Taylor	100000
Kim	80000	Elec. Eng.	Taylor	85000
Crick	72000	Biology	Watson	90000
Srinivasan	65000	Comp. Sci.	Taylor	100000
Califieri	62000	History	Painter	50000
Brandt	92000	Comp. Sci	Taylor	100000
Mozart	40000	Music	Packard	80000
Gold	87000	Physics	Watson	70000
Singh	80000	Finance	Painter	120000

Database design example

Table *instructor*

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Table *department*

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

Database design

- Theory on normalization
  - Formal tools that determine what is right and what is wrong.
  - Procedures for making the right design

- Entity-relationship model
  - Models enterprise data as a collection of entities and relationships
    - An entity is a "thing" or "object" identifies in the enterprise. It is uniquely distinguishable from others
      - Described by an attribute set
    - A relationship is a link between entities
  - Represented by E-R diagram

Entity-relationship model

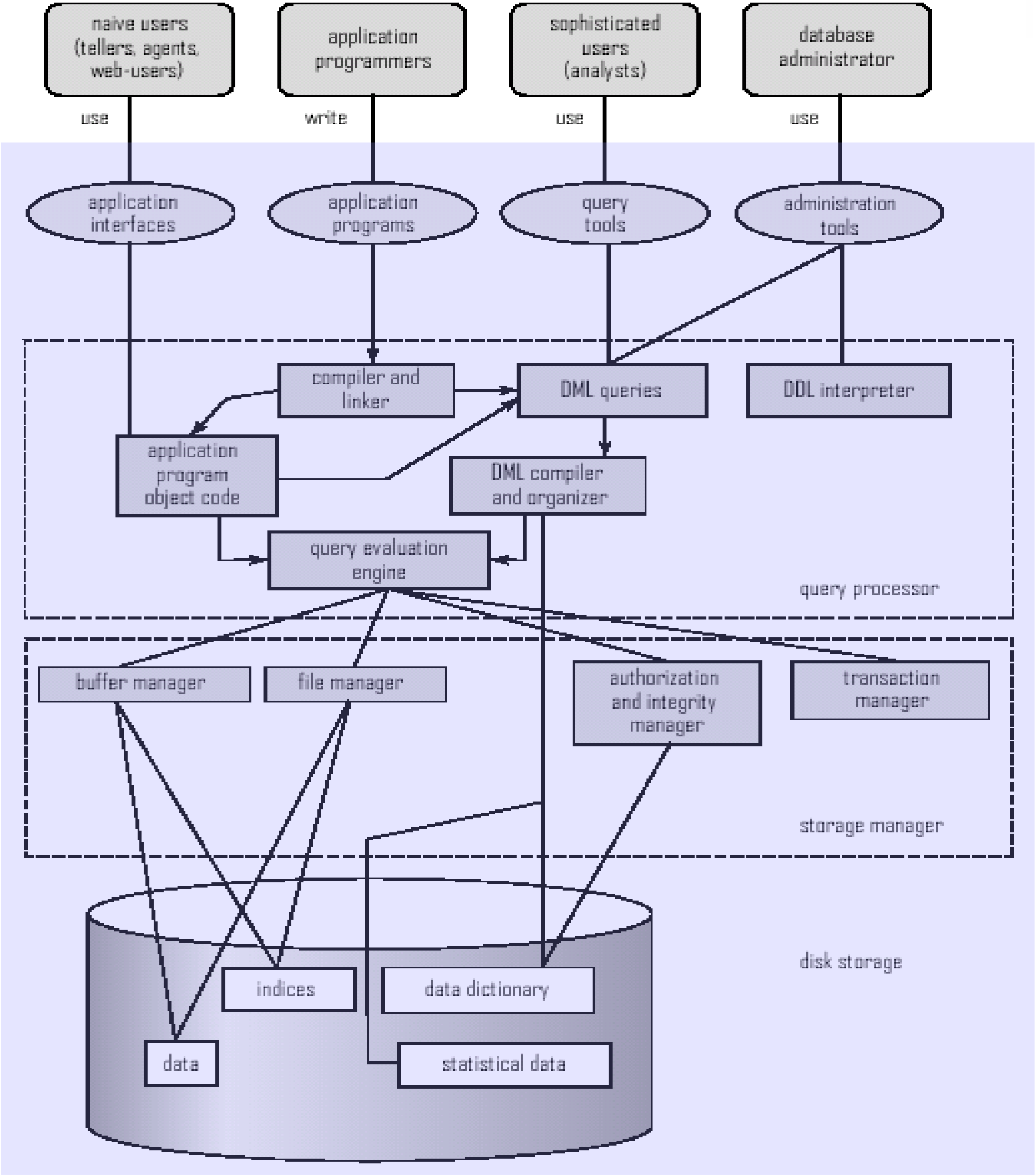
- Entity sets
  - *instructor* and *department*
  - marked primary and foreign keys
- Relationships, e.g. *homedepartment*
  - cardinality n:1 (many-to-one)
  - total (from *instructor*)
    - marked with a perpendicular line at *department*

Object-relational data model

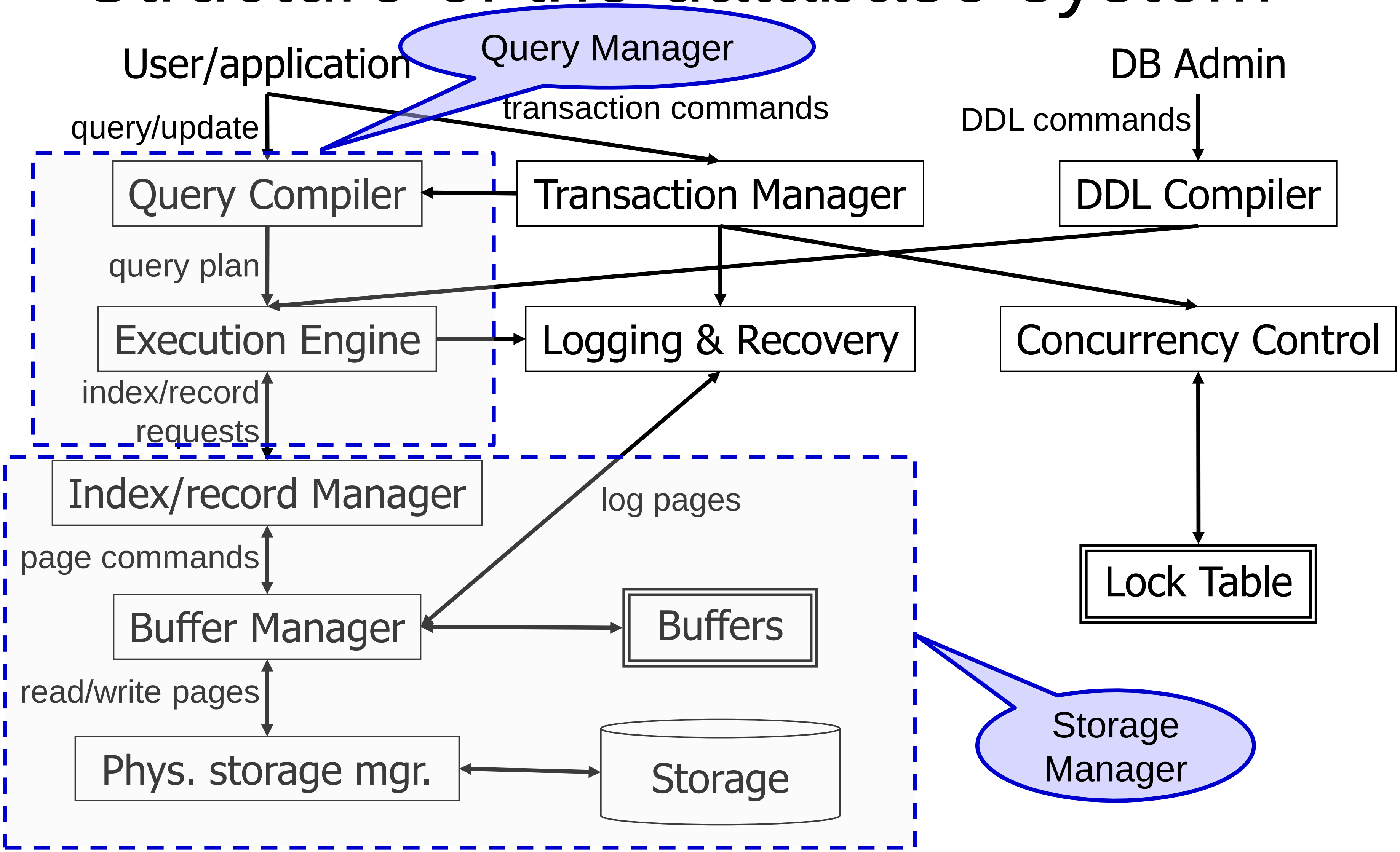
- Extends the relational model
  - by objects and structures for working with them
- Allows attributes to have complex structures
  - E.g. nested relations
- Preserves relational access to data
  - Extends it
  - Backward compatible with existing relational languages
    - SELECT object.name FROM table WHERE object.isValid();

Structure of the database system

Detailed structure of a typical RDBMS



Structure of the database system

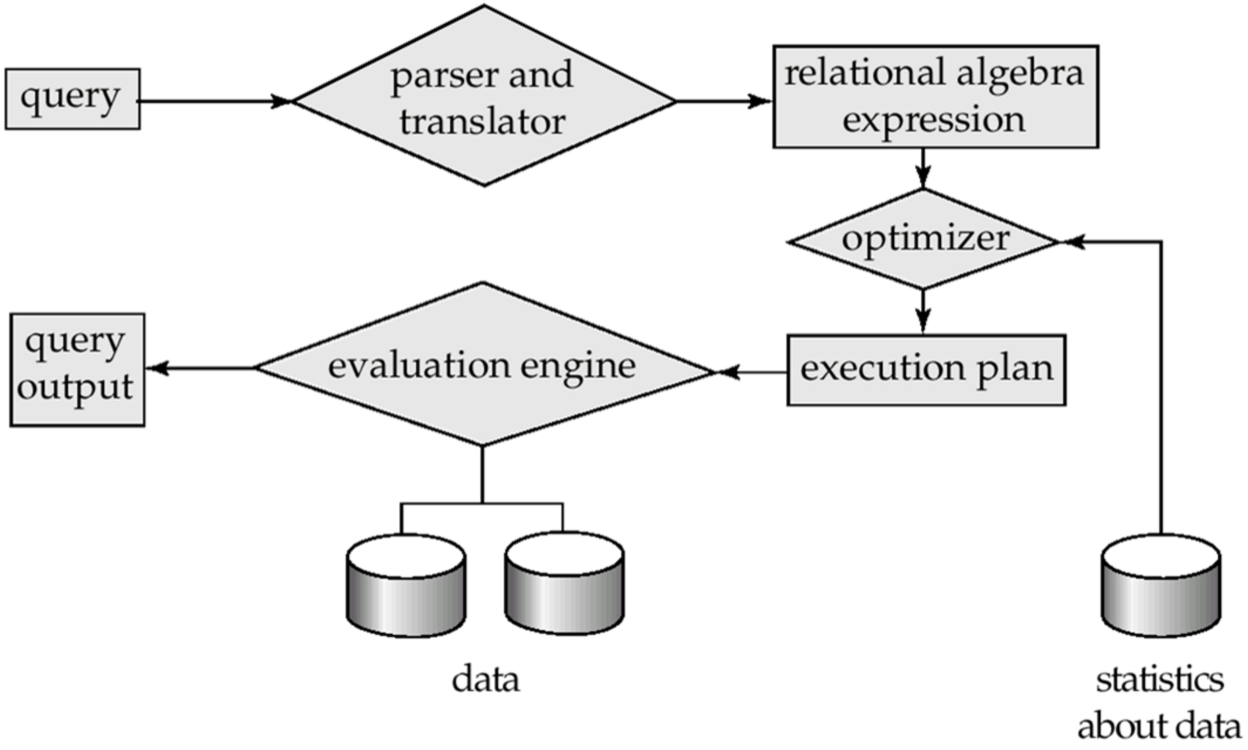


Storage manager

- **Storage manager**
  - A module that provides an interface for physical data storage
  - The task is
    - Working with the file system
    - Efficient data storage, update and retrieval
  - Manages
    - Access to storage
    - Organizes data, e.g. into files
    - Performs indexing

Query processing

- Parsing and translation
- Optimization
- Execution



Query optimization

- Multiple options for processing the same query
  - Equivalence of expressions
  - Different algorithms for each operation
- Different costs of individual options
  - Processing cost, usually time
  - The differences can be very significant
- The need to estimate these costs
  - Use statistics about data (session size, ...)
  - Statistics on intermediate query results
    - For estimating complex queries

Transaction processing

- Transaction
  - Sequence of operations that make up a logical block, function, database application
- Transaction processing (transaction manager)
  - Ensuring a consistent (correct) state
    - regardless of database system failures
    - e.g., power failure, OS crash, memory error.
- Concurrency
  - Ensures data consistency when multiple transactions are executed simultaneously
    - i.e., it ensures the isolation of transactions

Database users

- Types of users according to their activities
  - Application Programmer
    - Uses DML
  - Knowledgeable users
    - They use the database query language directly
  - Specialized users
    - They create applications for which traditional data processing is not sufficient
  - Naïve users
    - It uses prepared application programs, interfaces
      - e.g., websites

Database Manager

- Coordinates activities performed with the database system
  - Has knowledge of available resources and business needs
- Activities:
  - Database schema definition
  - Define methods for storing and accessing data
  - Schema and physical organization changes
  - Authorizing access rights, creating users
  - Definition of integrity constraints
  - An intermediary for communication between users
  - Monitoring performance and system tuning

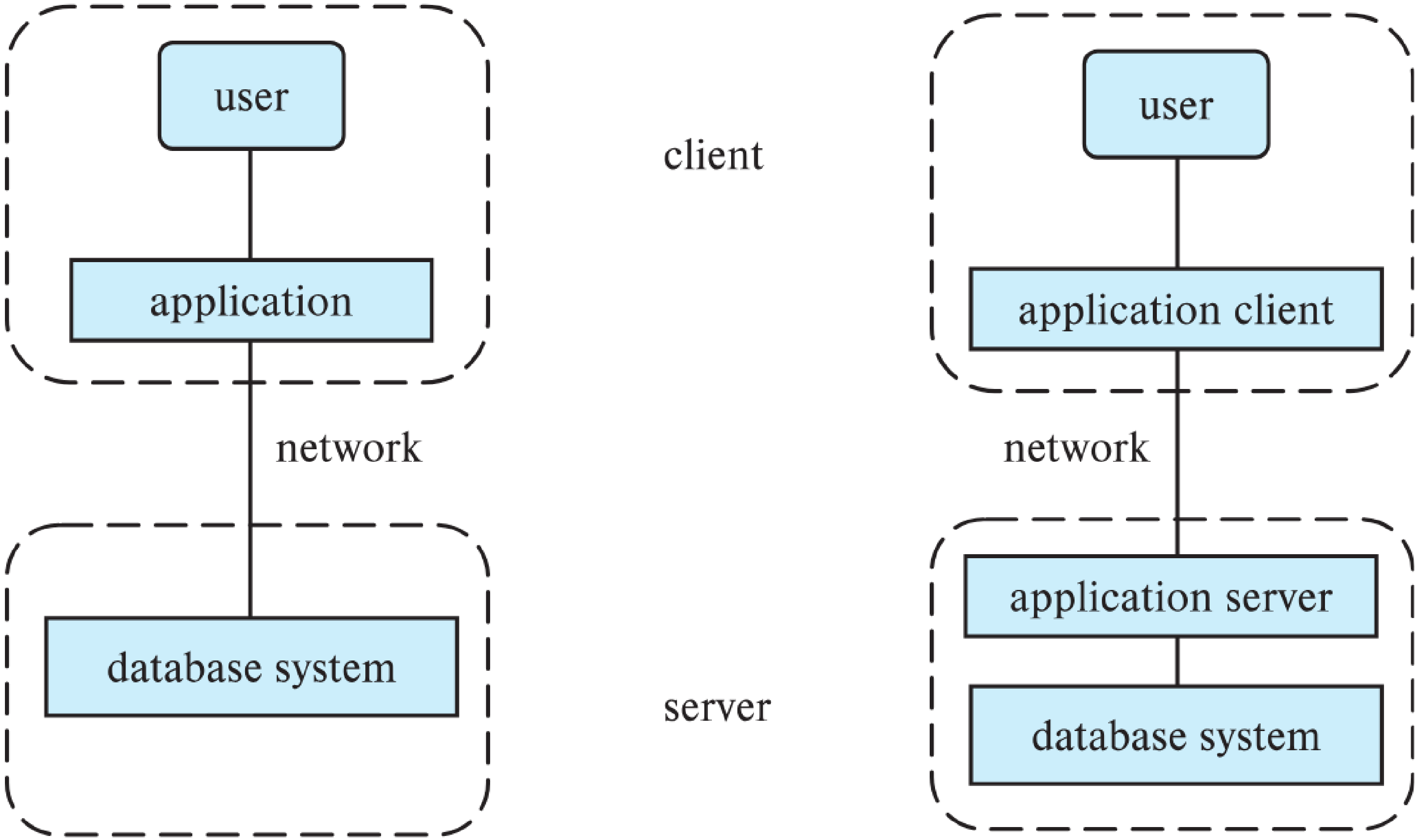
Database architectures

- The architecture of the database system is influenced by the environment in which the DB system is to run.
  - Centralized
  - Client-server
  - Parallel (multiprocessors)
  - Distributed

Database Applications

- 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

History of Database Systems

- 50s and beginning of 60s
  - Data processing on magnetic tapes
    - Sequential access only
  - Punched cards for data input
- 60s and 70s
  - Hard drive and direct data access
  - Network and hierarchical data model
  - Ted Codd defines a relational model
    - IBM Research begins to implement System R (now as DB2)
    - UC Berkeley – prototype of the Ingres system
  - High-performance transaction processing
    - For its time

History of Database Systems

- 80s
  - Research on relational systems led to commercial systems
    - SQL has become an industry standard
  - Parallel and distributed databases
  - Object-oriented databases
- 90s
  - Decision support and knowledge mining
  - Huge data warehouses (terabytes)
  - The beginning of internet (web) trading
- 2000s
  - XML and XQuery standards
  - Automatic database administration and tuning
  - Specialized database systems:
    - Massive data storage, distributed, NoSQL databases, NewSQL
    - Google BigTable, Yahoo PNuts, Amazon Redshift, …

History of Database Systems

- 2010s
  - SQL reloaded
    - SQL front end to Map Reduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases

Takeaways

- Why we have database systems
  - Data abstraction
  - Structure of the database system
- Data models
  - relational, ER, object-relational
- Terminology
  - DBMS, database (instance), database schema