RAWDATA
Section 1

Introduction to Databases and Database Systems

Henrik Bulskov & Troels Andreasen

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - In particular, a **database**: Collection of interrelated data
  - A set of programs that support access to the data

- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades, students
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases touch all aspects of our lives
- Databases can be very large.

University Database Example

- Database containing
  - student and instructor information,
  - courses and course registrations,
  - grades, …
- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems
- so why not just use file systems?

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones
- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Levels of Abstraction

- **Physical level:** describes how a record (e.g., customer) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

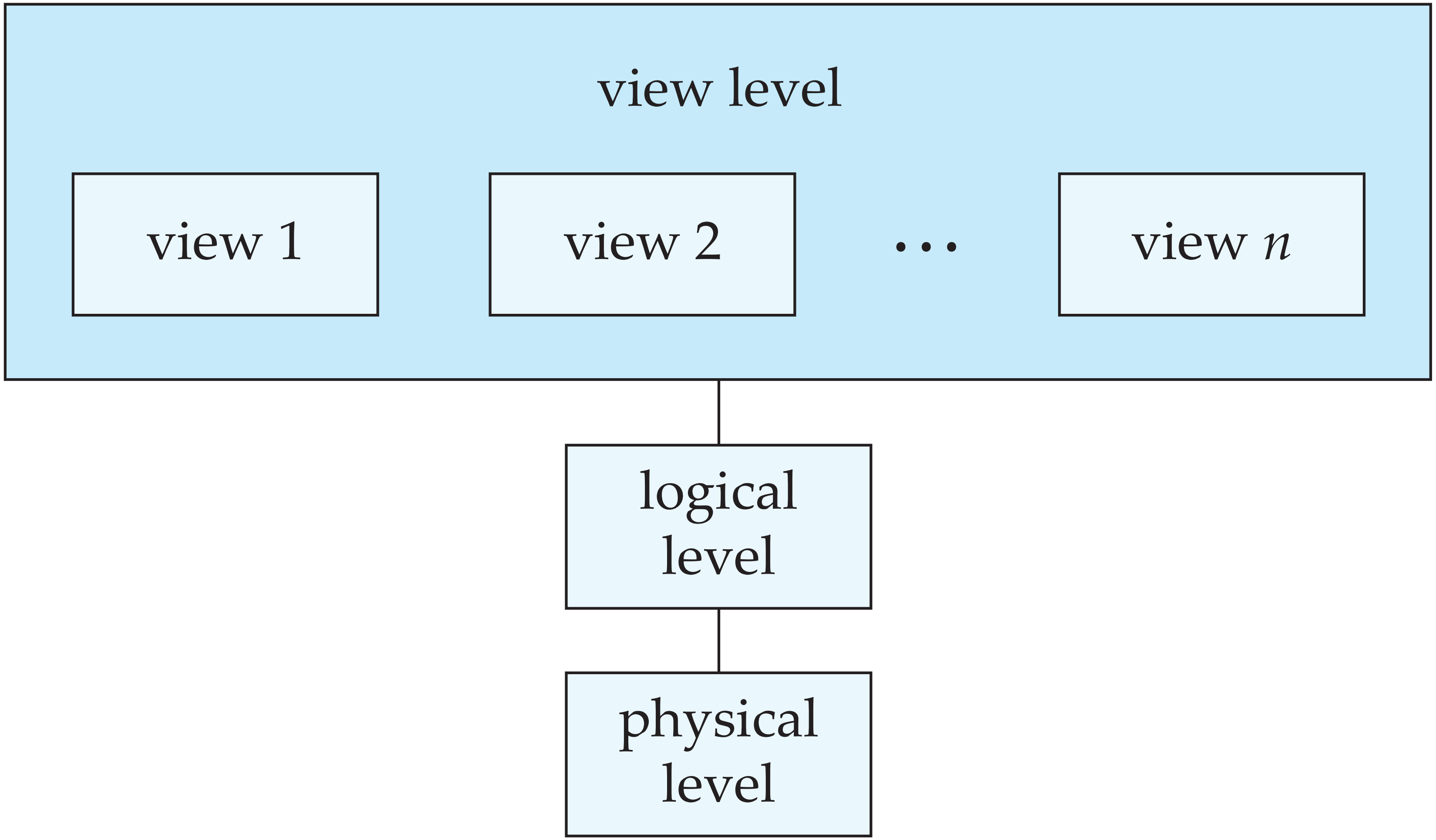
  type *instructor* = **record**

  ID : string;
  name : string;
  dept_name : string;
  salary : integer;

  end;
- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

- Similar to types and variables in programming languages

- **Schema** – the logical structure of the database
  - Example: The database consists of information about a set of customers and accounts and the relationship between them
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level

- **Instance** – the actual content of the database at a particular point in time
  - Example: actual customers, accounts, etc.
  - Analogous to the value of a variable

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

- What is a data model
  - A collection of tools for describing
    - Data
    - Data relationships
    - Data semantics
    - Data constraints

- Important models
  - **Relational model**
  - Entity-Relationship data model (mainly for database design)
  - Object-based data models (Object-oriented and Object-relational)
  - Semistructured data model (XML)
  - Other older models:
    - Network model
    - Hierarchical model

Relational Model

- Relational model
  - database ~ collection of tables
- Example of tabular data in the relational model

Columns

Rows

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

(a) The *instructor* table

Relational Model

A Sample Relational Database

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

(a) The *instructor* table

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

(b) The *department* table

Database Language

- For databases we typically separate the language into
  - Data Definition Language (DDL)
    - for defining the database schema
  - Data Manipulation Language (DML)
    - accessing and manipulating data

Data Definition Language (DDL)

- DDL for defining the database schema

  Example (DDL in SQL):

  create table *instructor* (
  ID **char**(5),
  name **varchar**(20),
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - constraints that restrict the content of the database
    - such as
      - Primary key (ID uniquely identifies instructors)
      - Referential integrity (**references** constraint in SQL)
        - » e.g. *dept_name* value in any *instructor* tuple must appear in *department* relation
  - Authorization
    - users, permissions, etc.

Data Manipulation Language (DML)

- DML for accessing and manipulating data
  - DML also known as query language
- Two classes of languages
  - **Procedural** – user specifies what data is required and how to get those data
  - **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language
  - SQL is nonprocedural

DML in SQL

- **SQL** is widely used
  - Example: Find the name of the instructor with ID 22222

    select *name*
    from *instructor*
    where *instructor.ID* = '22222'

  - Example: Find the ID and building of instructors in the Physics dept.

    select *instructor.ID*, *department.building*
    from *instructor*, *department*
    where *instructor.dept_name* = *department.dept_name* **and**
    department.dept_name = 'Physics'

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

(a) The *instructor* table

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

(b) The *department* table

SQL and Application programs

- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC / JDBC / ADO.NET) which allow SQL queries to be sent to a database

  - we will touch on JDBC and ADO.NET later in this course

- SQL-DBMS' can also be "programmed"
  - adding code to the database
  - in stored procedures and functions
    - can be called from any interface to the database (thus also from applications programs)
  - and in so-called triggers
    - event-driven,
    - activated as side-effects to other operations on the database

  - we will use these features later in this course

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  - **Business decision** – What **attributes** should we record in the database?
  - **Computer Science decision** – What **relation schemas** should we have and how should the attributes be distributed among the various relation schemas?

- Physical Design – Deciding on the physical layout of the database

- Thus Database Design is concerned with the two lower levels of abstractions (Logical and Physical) (see slide 6)
  - The view level is defined by application program

Database Design?

- Are there any problems with this design? (Exercise 1.12)

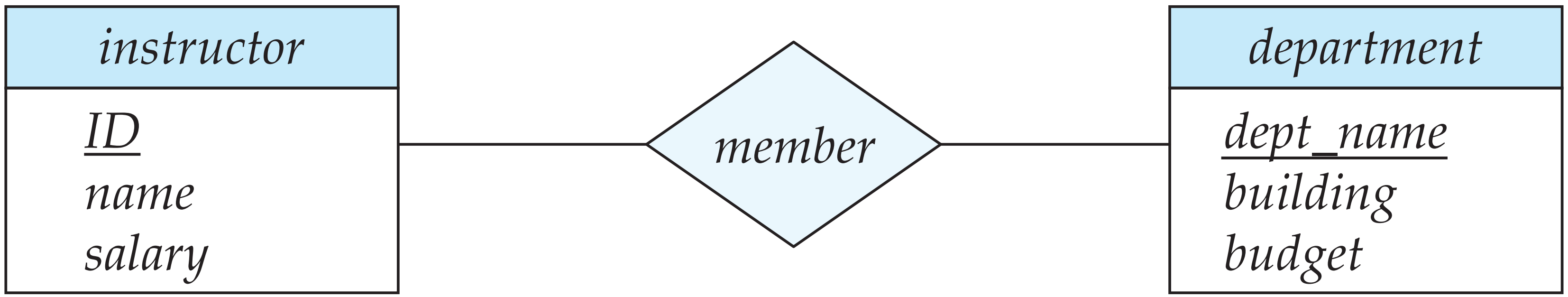
ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Design Approaches

- Normalization Theory (Chapter 8)
  - Formalize what designs are bad, and test for them
- Entity Relationship (E-R) Model (Chapter 7)
  - a widely used data model for describing the data of a business domain in an abstract way, distinguishing *entities, relationships* and *attributes*
  - an abstraction that tends to lead to better designs
  - major E-R notation: diagrams

The Entity-Relationship Model

- ❑ Models an enterprise as a collection of *entities* and *relationships*
  - – Entity: a "thing" or "object" in the enterprise that is distinguishable from other objects
    - Described by a set of *attributes*
  - – Relationship: an association among several entities
- ❑ Represented diagrammatically by an *entity-relationship diagram:*



What happened to dept_name of instructor (slide 13)?

Object-Relational Data Models

- Relational model: flat, “atomic” values
- Object Relational Data Models
  - Extend the relational data model by including object orientation and constructs to deal with added data types.
  - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
  - Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
  - Provide upward compatibility with existing relational languages.

- Postgres is object-relational

Database System Internals

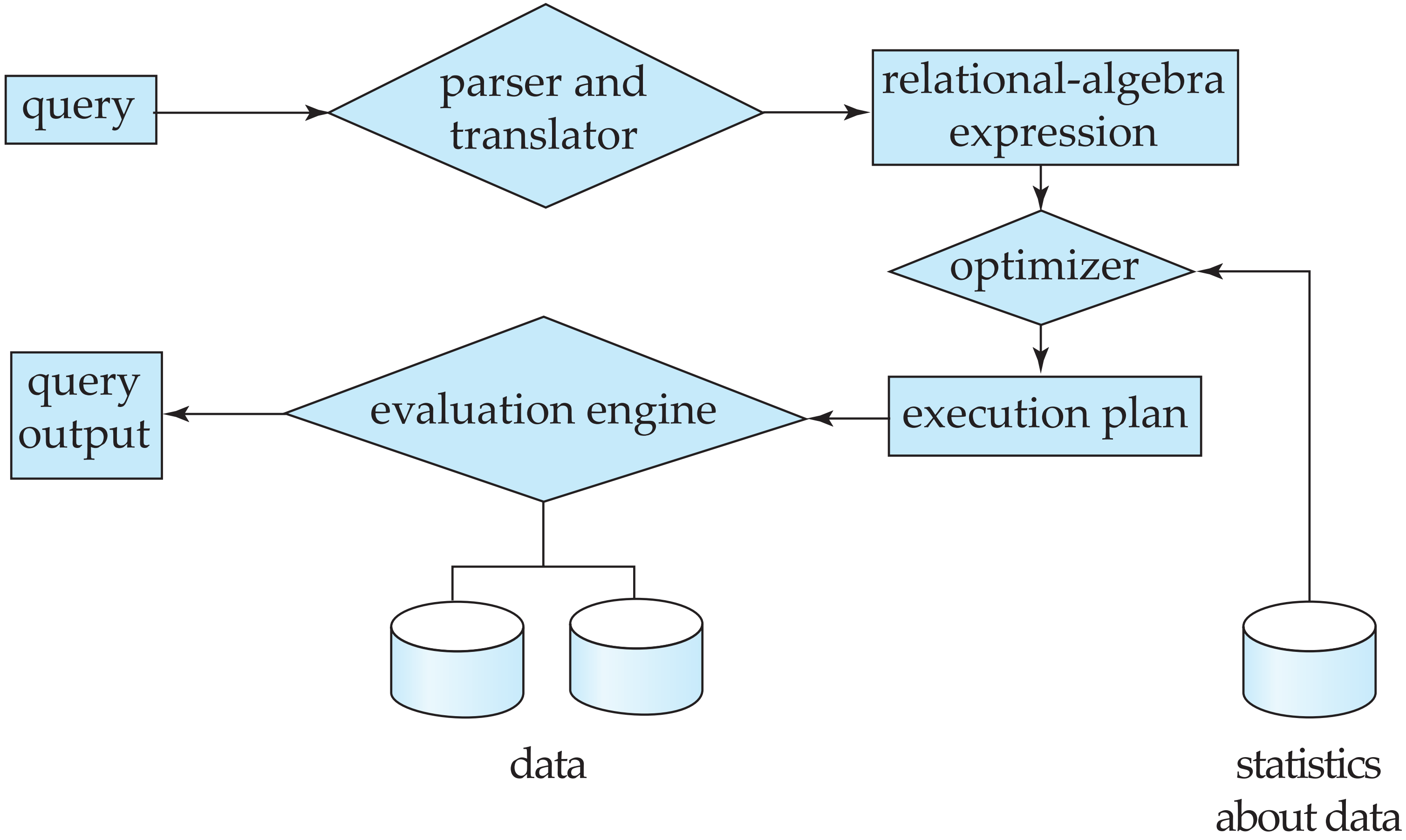
- Components
  - Storage manager
  - Query Optimizer
  - Transaction-management component
  - Concurrency-control manager
  - …

Storage Management

- **Storage manager** is 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 file manager
  - Efficient storing, retrieving and updating of data
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Query Processing

- **Query Processor** takes care of efficient evaluation of queries
  1. Parsing and translation
  2. Optimization
  3. Evaluation



Query Processing (Cont.)

- Consider alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
  - Use of statistical information about relations

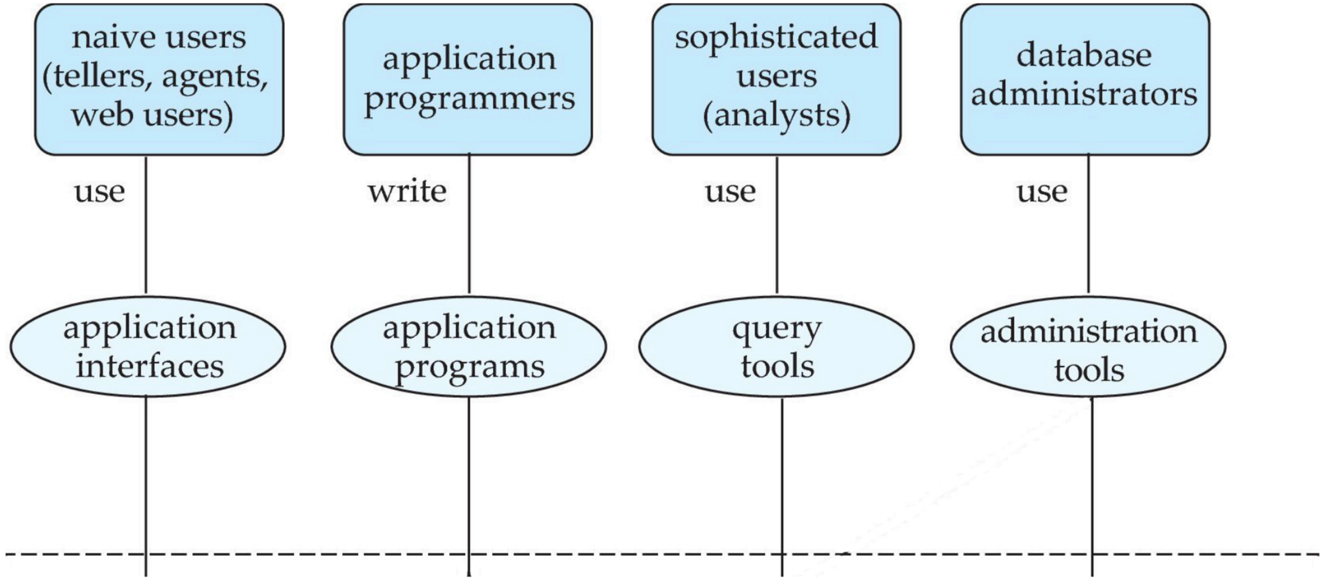
- Cost difference between a good and a bad way of evaluating a query can be enormous

Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?

- 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.

Database Users and Administrators



Database

Database System Internals overview