Chapter 1: Introduction

Instructor: Yi-Ju Tseng, PhD
Department of Computer Science, NYCU

Database System Concepts, 7th Ed.

You Will Learn…

- What are databases and database systems?
- Why we need database?
- Some basic ideas and components in a database.
    - Data models
    - Database languages
    - Database engine

Outline

- Database-System Applications
- Data Models
- Database Languages
- Database Design & Engine

DATABASE-SYSTEM APPLICATIONS

Database and Database system

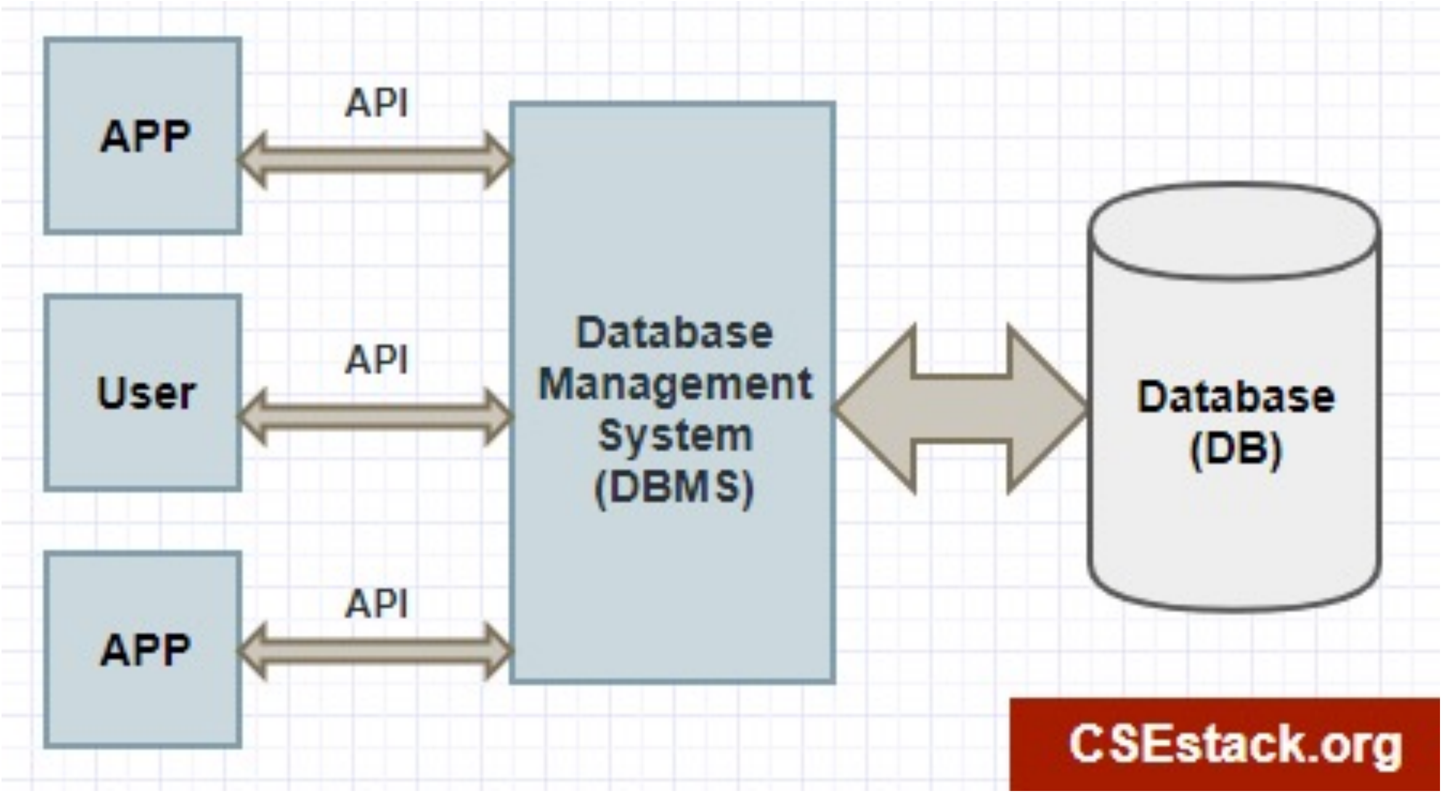
- A **database** is an **organized collection of data** stored and accessed electronically.
  - Collection of interrelated data
  - Highly valuable
  - Relatively large
  - Accessed by multiple users/applications, at the same time

- A **database system** (or DBMS, database management system) is a complex **software system** whose task is to **manage a large, complex collection of data (=database)**.
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use

Database-System Applications

multiple users/applications

Collection of interrelated data



Set of programs to access the data

https://www.csestack.org/difference-database-database-management-system-dbms/

Database Applications Examples

- **Universities: registration, grades, … and others**

- Enterprise Information
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes.
- Manufacturing: management of production, inventory, orders, supply chain.
- Banking and finance
  - Credit card transactions
  - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data

University Database Example

- Data consists of information about:
  - Students
  - Instructors
  - Classes
- Application program examples:
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute GPA and generate transcripts

Purpose of Database Systems

What if …. You store the data directly in the file systems.

- **Data redundancy and inconsistency**
  - Data is stored in multiple file formats resulting induplication of information in different files
- **Difficulty in accessing data**
  - Need to write a new program to carry out each new task
- **Data isolation**
  - Multiple files and formats
- **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

Purpose of Database Systems (Cont.)

- **Atomicity of updates**
  - Failures may leave database in an inconsistent state with partial updates carried out
    - Ex: Transfer of funds from one account to another should either complete or not happen at all
- **Concurrent access by multiple users**
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Ex: Two people reading a balance (100) and updating it by withdrawing money (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

Purpose of Database Systems (Cont.)

- A major purpose of a database system is to provide users with an **abstract view** of the data.

- **Data abstraction**
  - **Hide the complexity** of data structures to represent data in the database from users through several levels of **data abstraction**.

DATA MODELS

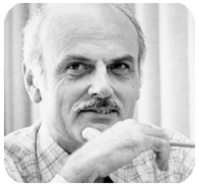
Data Models

- **Data models -** A collection of tools for describing:
  - Data
  - Data relationships
  - Data semantics
  - Data constraints

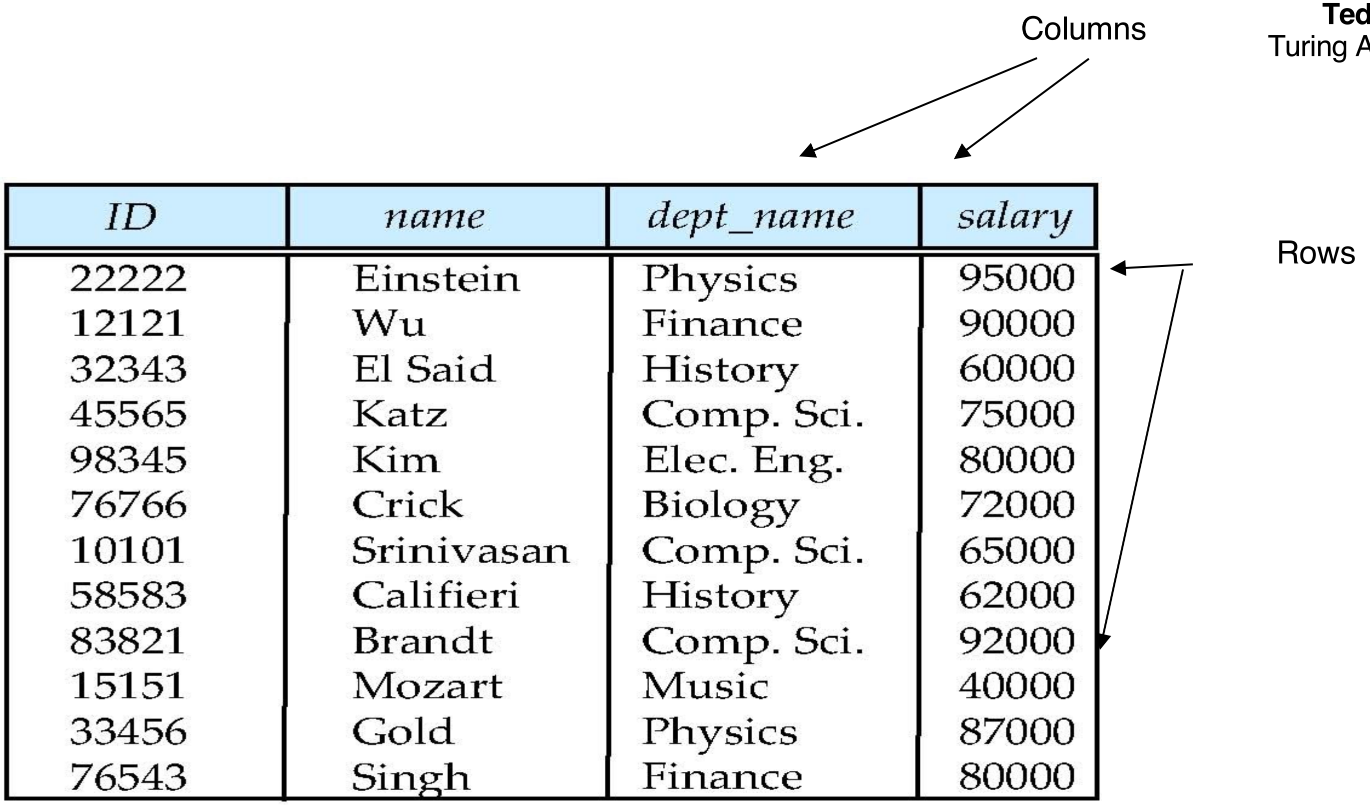
- **Relational model**
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Relational Model

- All the data is stored in various **tables**.



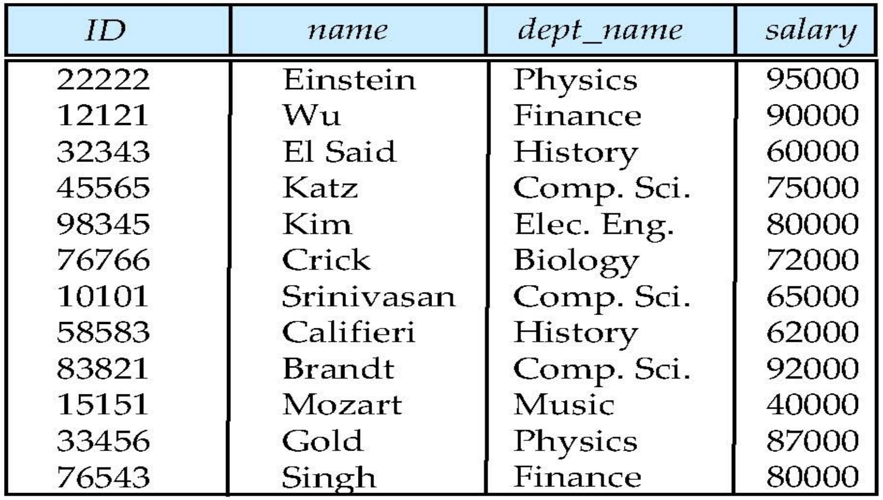
Ted Codd
Turing Award 1981


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

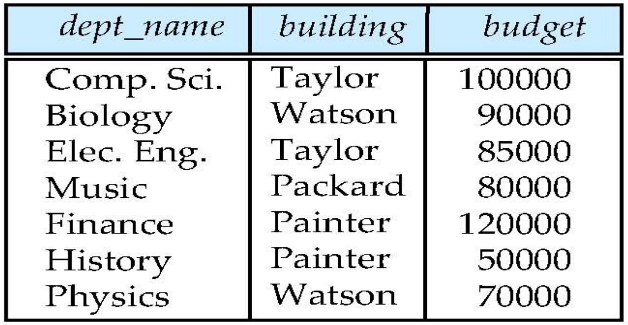
(a) The *instructor* table

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

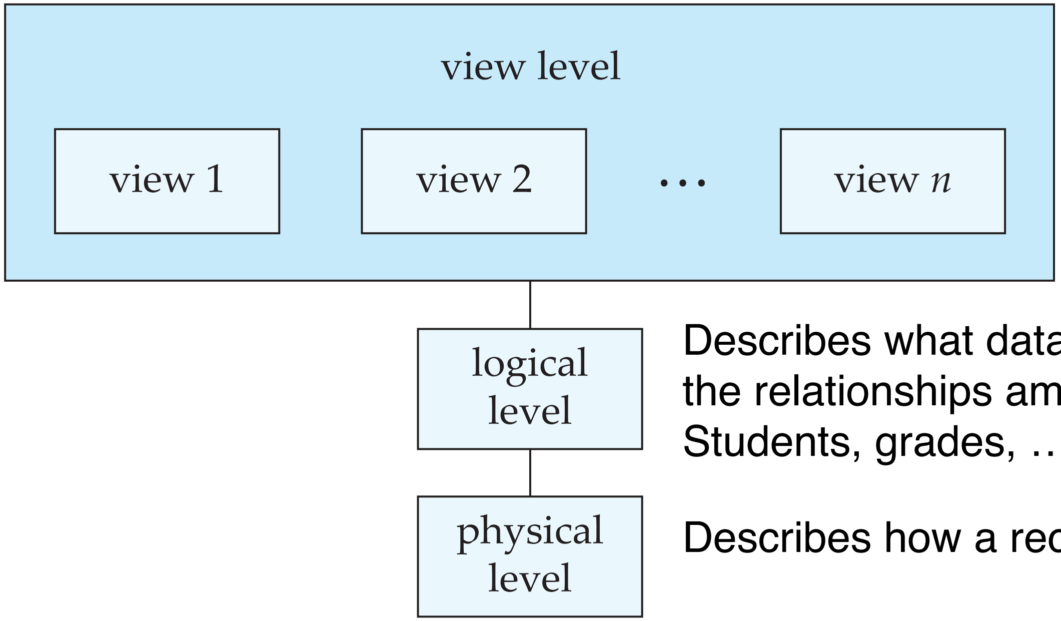


dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Levels of Data Abstraction

An architecture for a database system



Application programs hide details
Like DBMS UI

Describes what data is stored in database, and
the relationships among the data
Students, grades, … and others

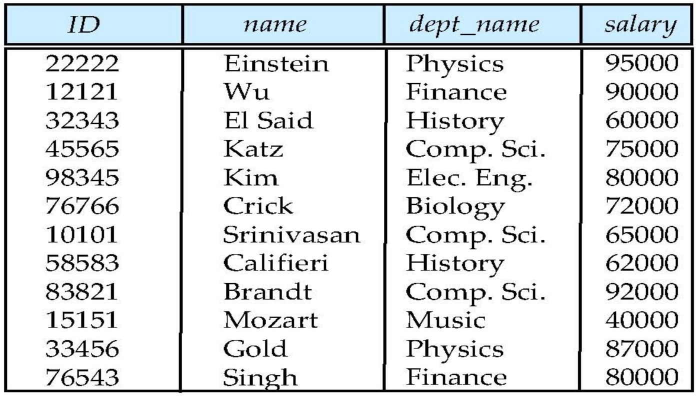
Describes how a record is stored

Instances and Schemas

- Similar to types (= **Schema**) and variables (= **Instance**) in programming languages

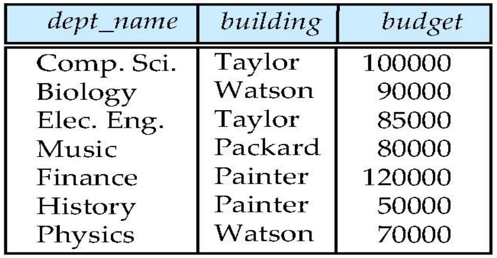
int a = 5;

- **Schema**
  - **Logical schema** – the **overall logical structure of the database**
    - Ex: database consists of information about instructors and departments in a university and the relationship between them
  - **Physical schema** – the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time



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

3 mins Quiz!

Please take the quiz on the E3 system

DDL and DML

DATABASE LANGUAGE

Data Definition Language (DDL)

- Specification notation for **defining the database schema**

  Example:

  ```
  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**
    - Primary key (ID uniquely identifies instructors)
  - **Authorization**
    - Who can access what

Data Manipulation Language (DML)

- Language for **accessing** and **updating** the data organized by the appropriate data model
- Two classes of languages
  - **Pure**
    - used for proving properties about computational power and for optimization
  - **Commercial**
    - used in commercial systems
    - **SQL** (Structured Query Language) is the most widely used commercial language

Data Manipulation Language (Cont.)

- There are basically two types of data-manipulation language
  - **Procedural DML**
    - require a user to specify what data are needed and **how to get those data.**
  - **Declarative DML**
    - require a user to specify what data are needed **without specifying how to get those data.**
- Declarative (non-procedural) DMLs are usually easier to learn and use than are procedural DMLs.
- The portion of a **DML** that involves **information retrieval** is called a **query language**.

SQL Query Language

- **Declarative DML (query language)**
- Example to find all instructors in Comp. Sci. dept

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.’

- SQL does not support actions such as **input** from users, **output** to displays, or **communication** over the network.
- Applications generally access databases through one of
  - Such computations and actions must be written in a **host language**, such as Java or Python, with embedded SQL queries that access the data in the database.
  - Application program interface (API, e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

DATABASE DESIGN AND ENGINE

Database Design

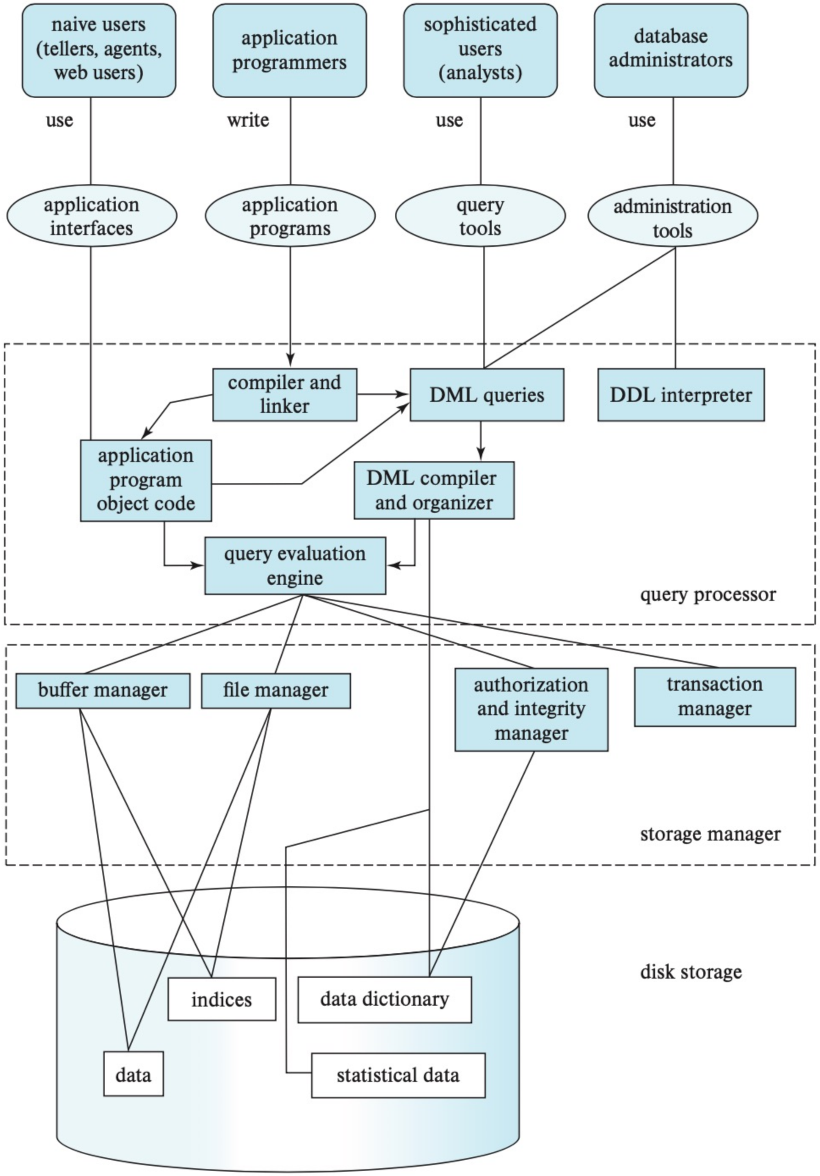
The process of designing the general structure of the database:

- **Logical Design** – Deciding on the **database schema**. Database design requires that we find a “good” collection of relation schemas.
    - The logical relationships among the objects

- **Physical Design** – Deciding on the physical layout of the database
    - The most effective way of storing and retrieving the objects

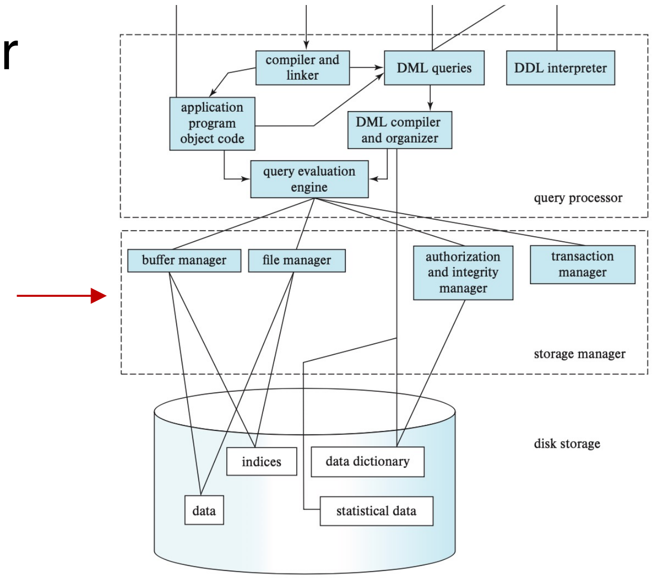
Database Engine

- The functional components of a database system can be divided into
  - The storage manager
  - The query processor component,
  - The transaction management component



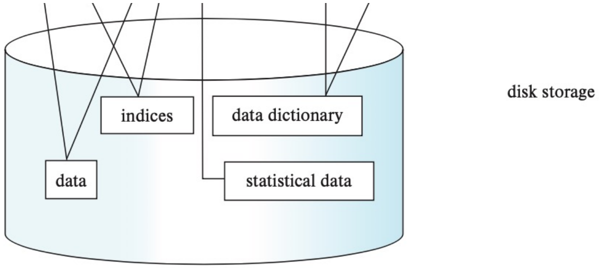
Storage Manager

- 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
- The storage manager components include:
  - Authorization and integrity manager
  - Transaction manager
  - File manager
  - Buffer manager

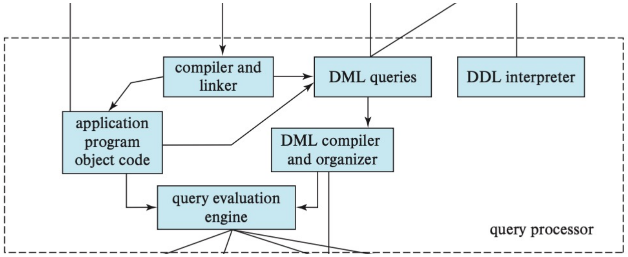


Storage Manager (Cont.)

- The **storage manager** implements several **data structures** as part of the physical system implementation:
  - **Data files**
    - store the **database itself**
  - **Data dictionary**
    - stores **metadata** about the structure of the database
    - in particular the schema of the database.
  - **Indices**
    - provide fast access to data items.
    - A database index provides pointers to those data items that hold a particular value.



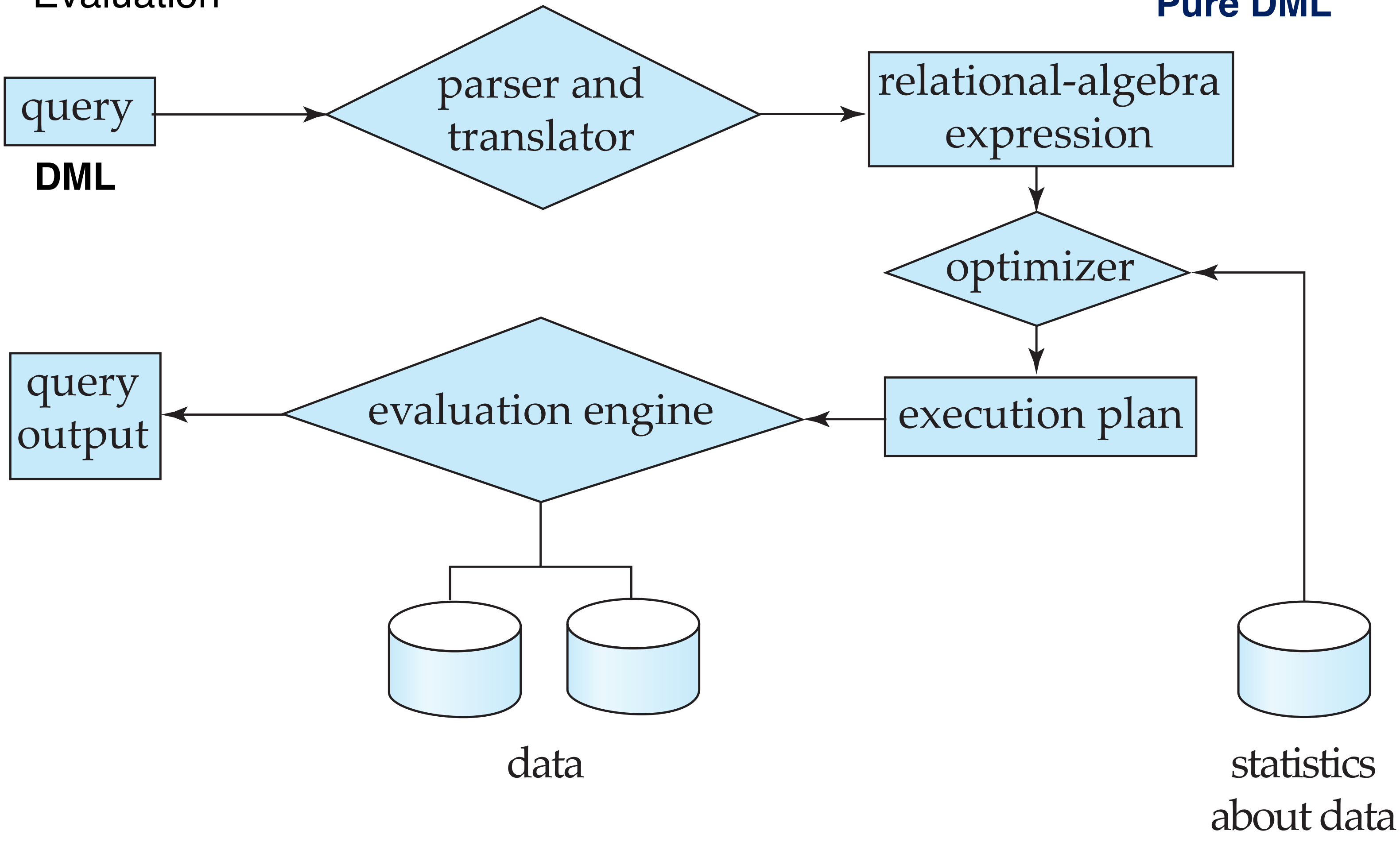
Query Processor



- The query processor components include:
  - **DDL interpreter**
    - interprets DDL statements and records the definitions in the data dictionary.
  - **DML compiler**
    - translates **DML statements in a query language** into an evaluation plan consisting of **low-level instructions** that the query evaluation engine understands.
    - performs **query optimization**; that is, it picks the lowest cost evaluation plan from among the various alternatives.
  - **Query evaluation engine**
    - executes low-level instructions generated by the DML compiler.

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Pure DML

DML

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.
  - Ex: Transfer of funds from one account to another should either complete or not happen at all

- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.
  - Ex: Two people reading a balance (100) and updating it by withdrawing money (50 each) at the same time

3 mins Quiz!

Please take the quiz on the E3 system

Questions?