Chapter 1: Intro to Database Management Systems

Database System Concepts, 7th Ed.

Outline

- Database-System Applications
- Purpose of Database Systems
- Database Languages
- Database Engine
- Database Users and Administrators

Database Applications Examples

- 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.

Database Applications Examples (Cont.)

- Banking and finance
  - ✓ customer information, accounts, loans, and banking transactions.
  - ✓ Credit card transactions
  - ✓ Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data)
- Universities: registration, grades

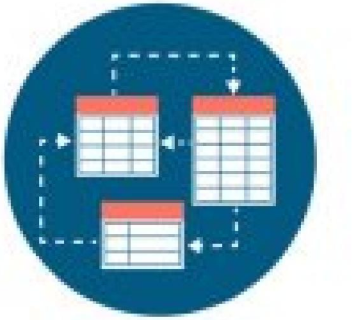
Database Applications Examples (Cont.)

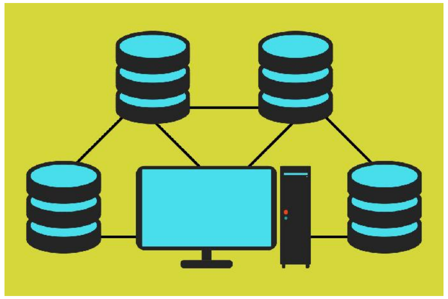
- Airlines: reservations, schedules
- Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- Web-based services
  - ✓ Online retailers: order tracking, customized recommendations
  - ✓ Online advertisements

- Document databases
- Navigation systems: For maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.

Database Management Systems (DBMS)

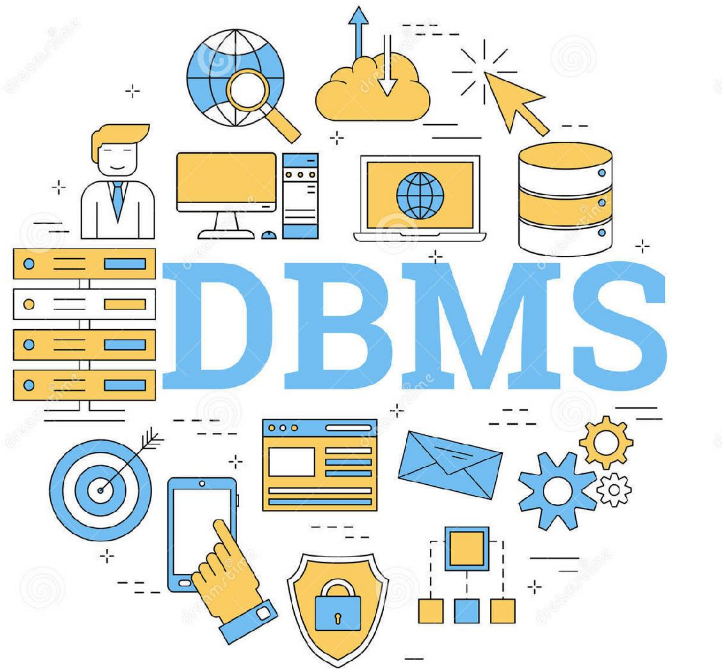
- DBMS contains information about a particular enterprise
  - ✓ Collection of interrelated data
  - ✓ Set of programs to access the data
  - ✓ An environment that is both *convenient* and *efficient* to use





Database Management Systems (DBMS)

- A modern database system is a complex software system whose task is to manage a large, complex collection of data.
- Databases touch all aspects of our lives



Purpose of Database Systems

- In the early days, database applications were built directly on top of **file systems**, which leads to:
  - ✓ 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
  - 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
    - Ex: 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

Example: University Database

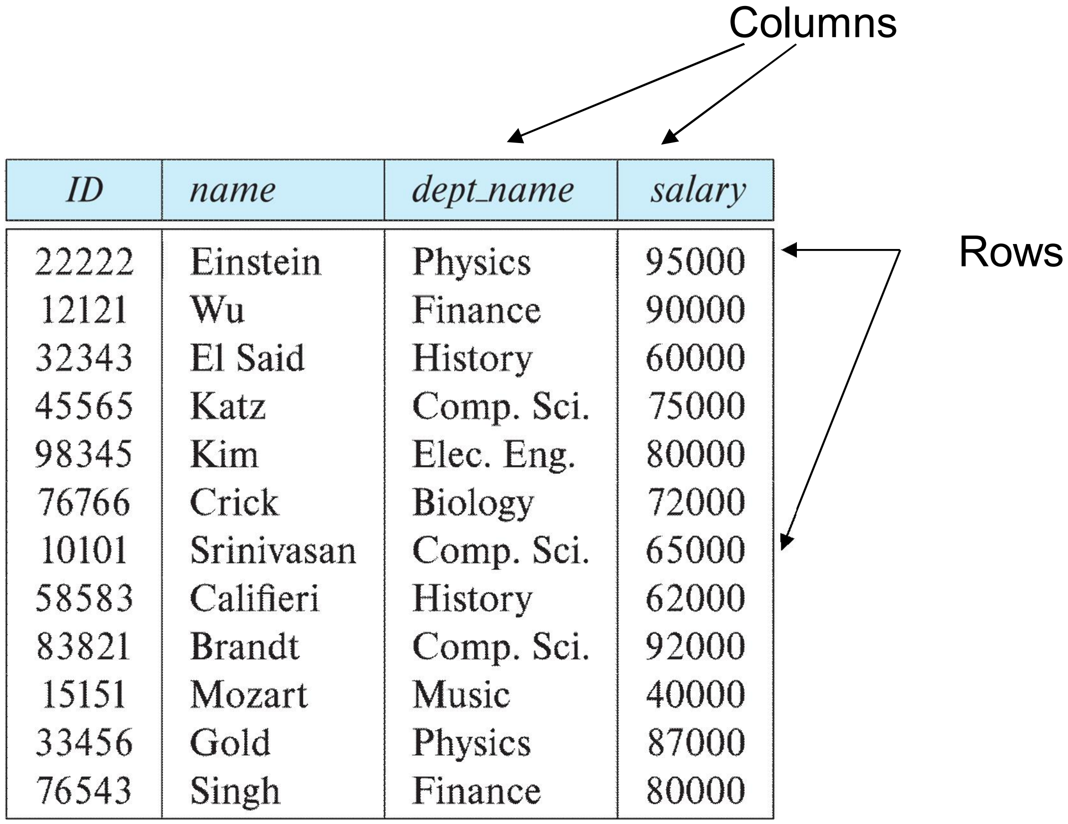
- In this text we will be using a university database to illustrate all the concepts
- Data consists of information about:
  - Students
  - Instructors
  - Classes
- Application program examples:
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts

Data Models

- A collection of tools for describing
  - ✓ Data
  - ✓ Data relationships
  - ✓ Data semantics
  - ✓ Data constraints
- Four different categories:
  1. Relational model
  2. Entity-Relationship data model (mainly for database design)
  3. Object-based data models (Object-oriented and Object-relational)
  4. Semi-structured data model (XML)
  - ✓ Other older models:
    - ❖ Network model
    - ❖ Hierarchical model

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model



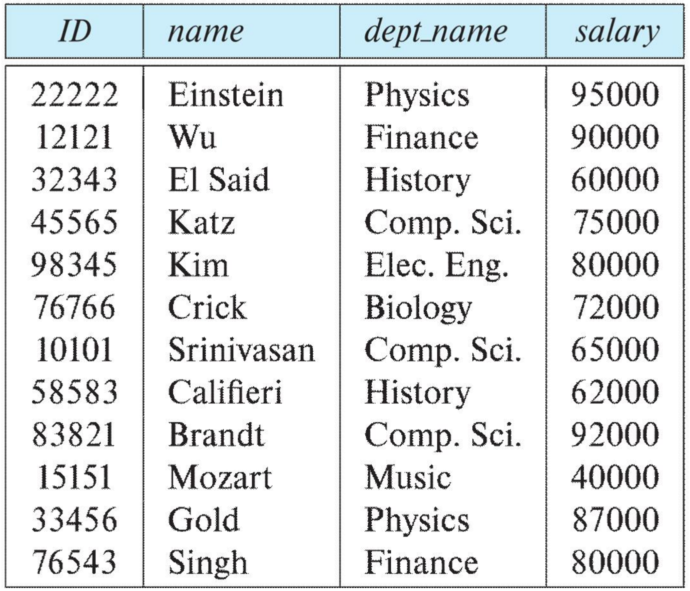
Columns

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

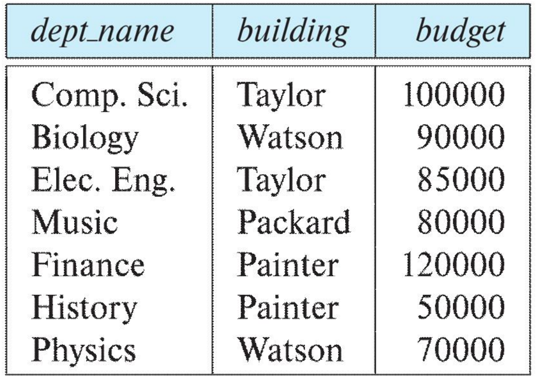
(a) The *instructor* table

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

Instances and Schemas

- Similar to types and variables in programming languages
- **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
    - Analogous to type information of a variable in a program
- **Physical schema** – the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time
  - ✓ Analogous to the value of a variable

Database Languages

- **Data-definition language (DDL)**
  - ✓ specify the database schema
- **Data-manipulation language (DML)**
  - ✓ express database queries and updates

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)

- ►DML is a language that enables users to access or manipulate data as organized by the appropriate data model.

- ►Language for accessing and updating the data organized by the appropriate data model
  - ✓DML also known as query language

SQL Query Language

- SQL query language is nonprocedural. A query takes as input several tables (possibly only one) and always returns a single table.
- Example to find all instructors in Comp. Sci. dept

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

- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - ✓ Language extensions to allow embedded SQL
  - ✓ Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Database Access from Application Program

- Non-procedural query languages such as SQL are not as powerful as a universal Turing machine.
- SQL does not support actions such as input from users, output to displays, or communication over the network.
- Such computations and actions must be written in a **host language**, such as C/C++, Java or Python, with embedded SQL queries that access the data in the database.
- **Application programs** -- are programs that are used to interact with the database in this fashion.

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

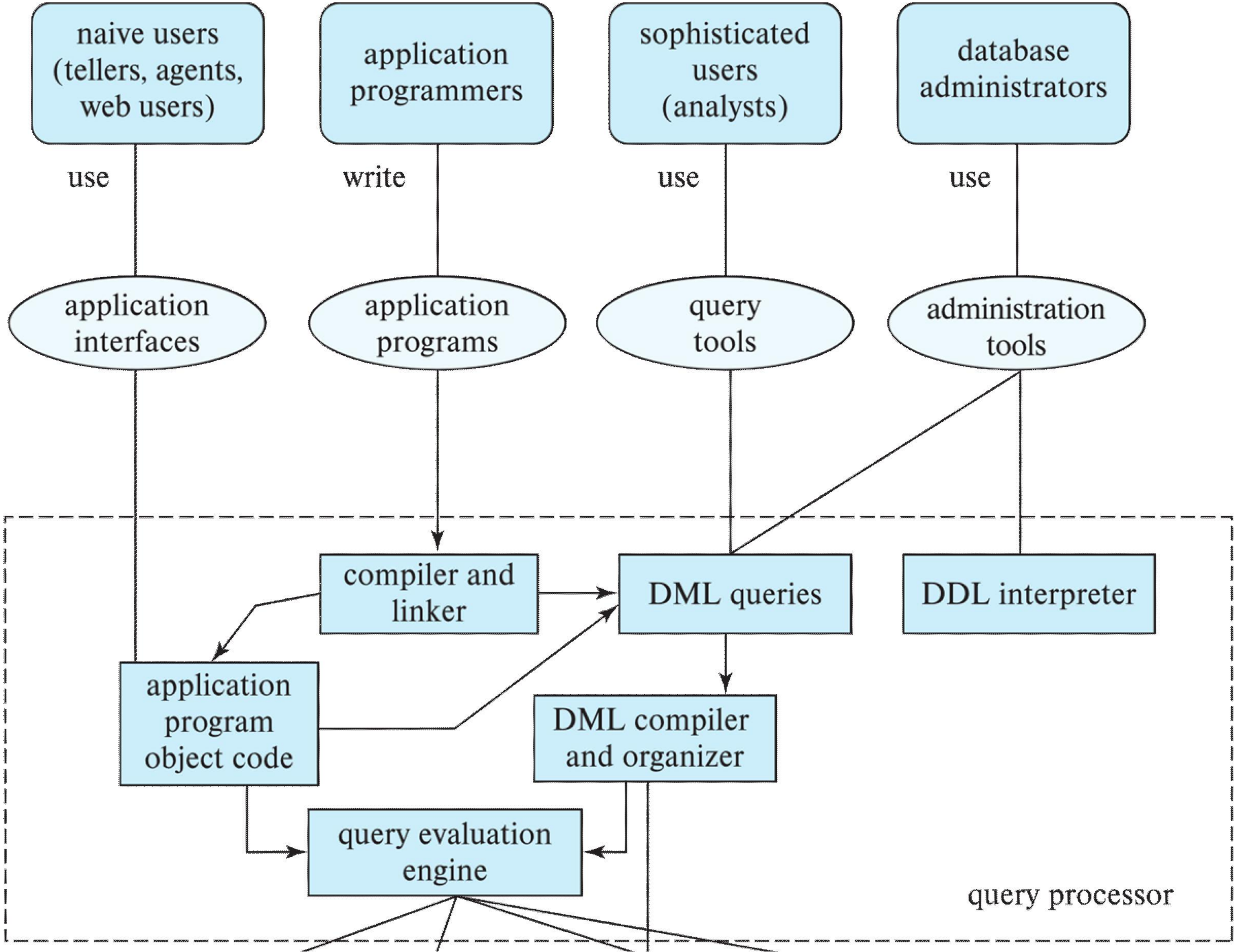
Query Processor Components

- DDL interpreter
  - ✓ interprets DDL statements
  - ✓ records the definitions in the data dictionary.
- DML compiler
  - ✓ translates DML statements into a low-level instructions.
- Query evaluation engine
  - ✓ **executes** low-level instructions generated by the DML compiler.

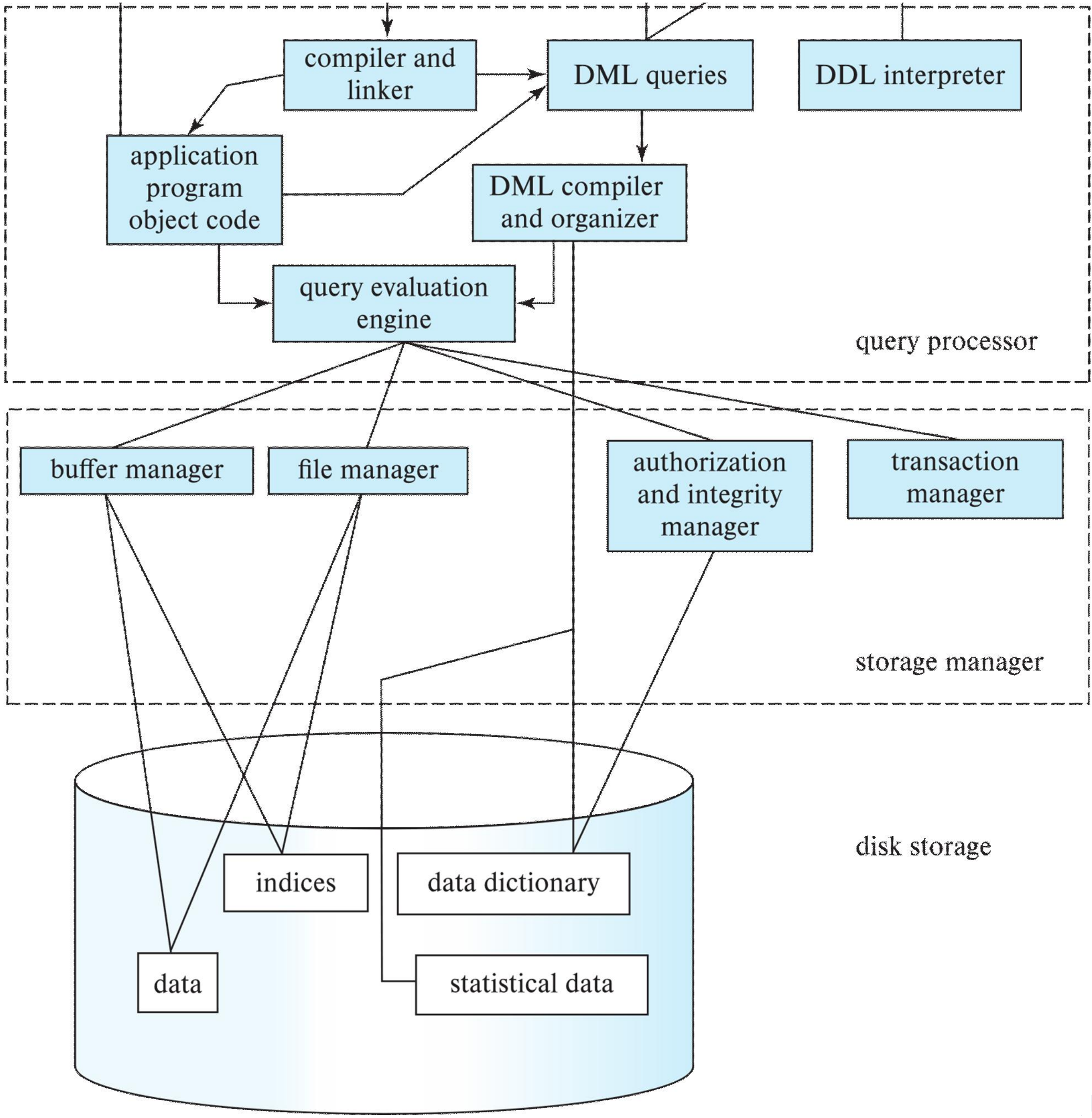
Transaction Management

- ► **Transaction**
  - ✓ 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
- ► **Concurrency-control manager**
  - ✓ controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Users



Database Architecture



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

End of Chapter 1