Chapter 1: Introduction

Database System Concepts, 7th Ed.

Outline

- What is it?
- Database-System Applications
- Purpose of Database Systems
- View of Data
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

Database Systems

- A **database-management system** (**DBMS**) is
  - A collection of interrelated data and
  - A set of programs to access those data
- DBMS contains information about a particular enterprise
  - The collection of data, usually referred to as the **database**
- Goals of a DBMS system: DBMS
  - provides a way to store and retrieve database information that is both *convenient* and *efficient* to use
  - is designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.
  - must ensure the safety and security of the information stored, despite system crashes or attempts at unauthorized access.
  - If data are to be shared among several users, the system must avoid possible anomalous results.

1.1 Database Systems Applications

- The earliest database systems arose in the 1960s in response to the computerized management of commercial data and were relatively simple.
- Modern applications include highly sophisticated, worldwide enterprises.
- Database systems are used to manage collections of data that are:
  - Highly valuable
  - Relatively large
  - Accessed by multiple users and applications, often at the same time.
- 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

Database Applications Examples

- WHERE NOT? Some representative examples
- Enterprise Information
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, account balances, assets
  - Human Resources: Information about employees, salaries, payroll taxes, future benefits and generation of paychecks
- Manufacturing: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
- Banking and Finance
  - Banking: customer information, accounts, loans, and banking transactions.
  - Credit card transactions: purchases and monthly statement generation
  - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- Universities: student information, registration, grades

Database Applications Examples (Cont.)

- Airlines: reservations, schedules, among the first to use databases in a geographically distributed manner
- Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- Web-based services
  - Social media: keeping records of users, connections between users (such as friend/follows information), posts made by users, rating/like information about posts, etc.
  - Online retailers: records of sales data and orders of retailer, order tracking, tracking product view, search terms, customized recommendations
  - Online advertisements: keeping records of click history to enable targeted advertisements, product suggestions, news articles, etc.
- Document databases: maintaining collections of new articles, patents, published research papers, etc.
- Navigation systems: For maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.

Modes of Database Usage

- Broadly there are two modes in which databases are used.
- **Online transaction processing**, where a large number of users use the database, with each user retrieving relatively small amounts of data, and performing small updates.
  - Generic database applications
- **Data analytics**, that is, the processing of data to draw conclusions, and infer rules or decision procedures for business decisions.
  - For example, banks need to decide whether to give a loan to a loan applicant, online advertisers need to decide which advertisement to show to a particular user, manufacturers and retailers need to make decisions on what items to manufacture or order in what quantities
- These tasks are addressed in two steps. First, data-analysis techniques attempt to automatically discover rules and patterns from data and then create *predictive models*.
- The field of *data mining* combines knowledge-discovery techniques invented by artificial intelligence researchers and statistical analysts with efficient implementation techniques that enable them to be used on extremely large databases.

.

1.2 Purpose of Database Systems

- In the early days, database applications were built directly on top of file systems and this typical **file-processing system** is supported by a conventional operating system.
- The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files and which leads to:
  - Data redundancy and inconsistency: data is stored in multiple file formats resulting in duplication 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
- These difficulties, among others, prompted both the initial development of database systems and the transition of file-based applications to database systems, back in the 1960s and 1970s.

Database systems offer solutions to all the above problems

Some Database Management Systems

- **Commercial (Large Scale) Database Management Systems**
  - 1. Oracle – Oracle 8i, Oracle9i, Oracle 10g, Oracle 11g, Oracle 12c, Oracle 18c, Oracle 19c, Oracle 21c, Oracle 23c, Oracle 23ai
  - 2. Microsoft SQL Server
  - 3. IBM DB2/DB2UDB
  - 4. Informix (Now own by IBM)
  - 5. Sybase
  - 6. MySQL (free/public domain)
  - 7. Ingress (free/public domain)
  - 8. PostgreSQL (free/public domain)
- **Small Scale / Personal use Database Management Systems**
  - 1. Microsoft Access
  - 2. FoxPro
  - 3. dBase
  - 4. Oracle 10g / 11g Express Edition /21c Express Edition

University Database Example

- 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

1.3 View of Data

- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.

  - Data models
    - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
  - 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 Model: 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
  - Object-relational
- Semi-structured data model (XML)
- Other older models (Historical):
  - Network model
  - Hierarchical model

Data Models

- **1. Relational Model**
- This is a lower level model (than E-R model).
- It uses a collection of tables to represent both data and relationships among those data.
- Each table has multiple columns, and each column has a unique name.
- The relational model is an example of a record-based model. This is because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed no. of fields, or attributes. The columns of the table correspond to the attributes of the record type.
- The relational model is the most widely used data model and a vast majority of current database systems are based on the relational model.

Relational Model

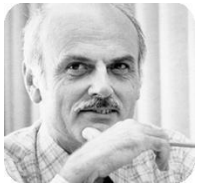
- All the data is stored in various tables.
- Example of tabular data in the relational model

Columns

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

Rows

(a) The *instructor* table



Ted Codd
Turing Award 1981

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

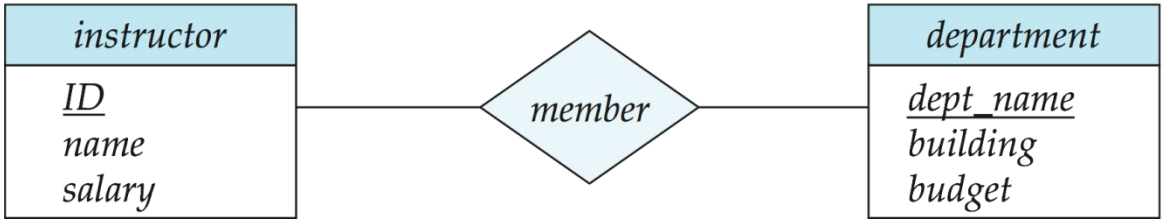
(a) The *instructor* table

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

Data Models

- **2. E-R (Entity-Relationship Model)**
- This is a higher-level data model (than relational model)
- It is based on a perception of a real world that consists of a collection of basic objects, called *entities* and the *relationship* among these objects.
- An entity is a “thing” or “object” in the real world that is distinguishable from all other objects. The entity-relationship model is widely used in database design.
- The entity-relationship model is widely used in database design.



Data Models

- **3. Object-Based Data Model**
- **i) Object-oriented data model:** Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity.
- **ii) Object-relation data model:** Combines the features of **object-oriented data model** and **relational data model.**
- **4. Semi-structured data model:** Permits the specification of data where individual data items of the same type may have different sets of attributes where in relational model every data item of a particular type must have the same set of attributes. *JSON (*JavaScript Object Notation) and the *extensible markup language* (XML) is widely used to represent semi-structured data.
- **Historical Models:** precede the relational data model. These are in little use now.
- **i) Network data model**
- **ii) Hierarchical data model**

Levels of Abstraction

- For the system to be usable, it must retrieve data efficiently.
- Since many database-system users are not computer trained, developers hide the complexity from users through several levels of **data abstraction**, to simplify users' interactions with the system. The are three levels:
- **Physical level**: The lowest level of abstraction describes *how* the data are actually stored. The physical level describes complex low-level data structures in detail.
- **Logical level**: The next-higher level of abstraction describes *what* data are stored in the database, and what relationships exist among those data.
- The logical level thus describes the entire database in terms of a small number of relatively simple structures.
- Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as **physical data independence**.
- Database administrators, who must decide what information to keep in the database, use the logical level of abstraction

Levels of Abstraction

- **For example:** creating a table in logical level

  create table instructor (

  ID **varchar** (5),

  name **varchar** (20),

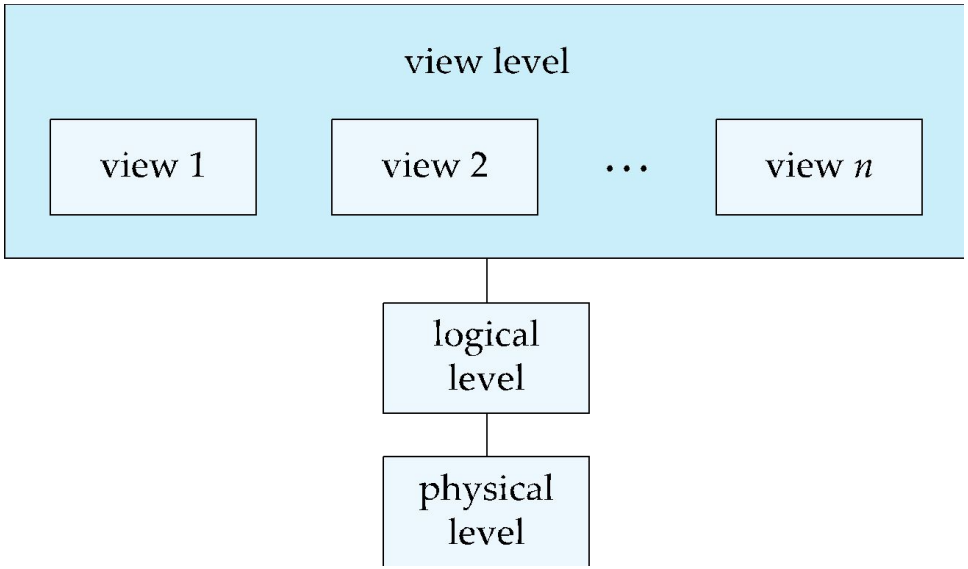
  dept_name **varchar** (20);

  salary **number** (8,2));

- **View level**: The highest level of abstraction describes only part of the entire database.
- Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database.
- Many users of the database system do not need all this information; instead, they need to access only a part of the database.
- The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

View of Data

The three levels of data abstractions:



- At the view level, computer users see a set of application programs that hide details of the data types. At the view level, several views of the database are defined, and a database user sees some or all of these views.
- In addition to hiding details of the logical level of the database, the views also provide a security mechanism to prevent users from accessing certain parts of the database.
- For example, clerks in the university registrar office can see only that part of the database that has information about students; they cannot access information about salaries of instructors.

Instances and Schemas

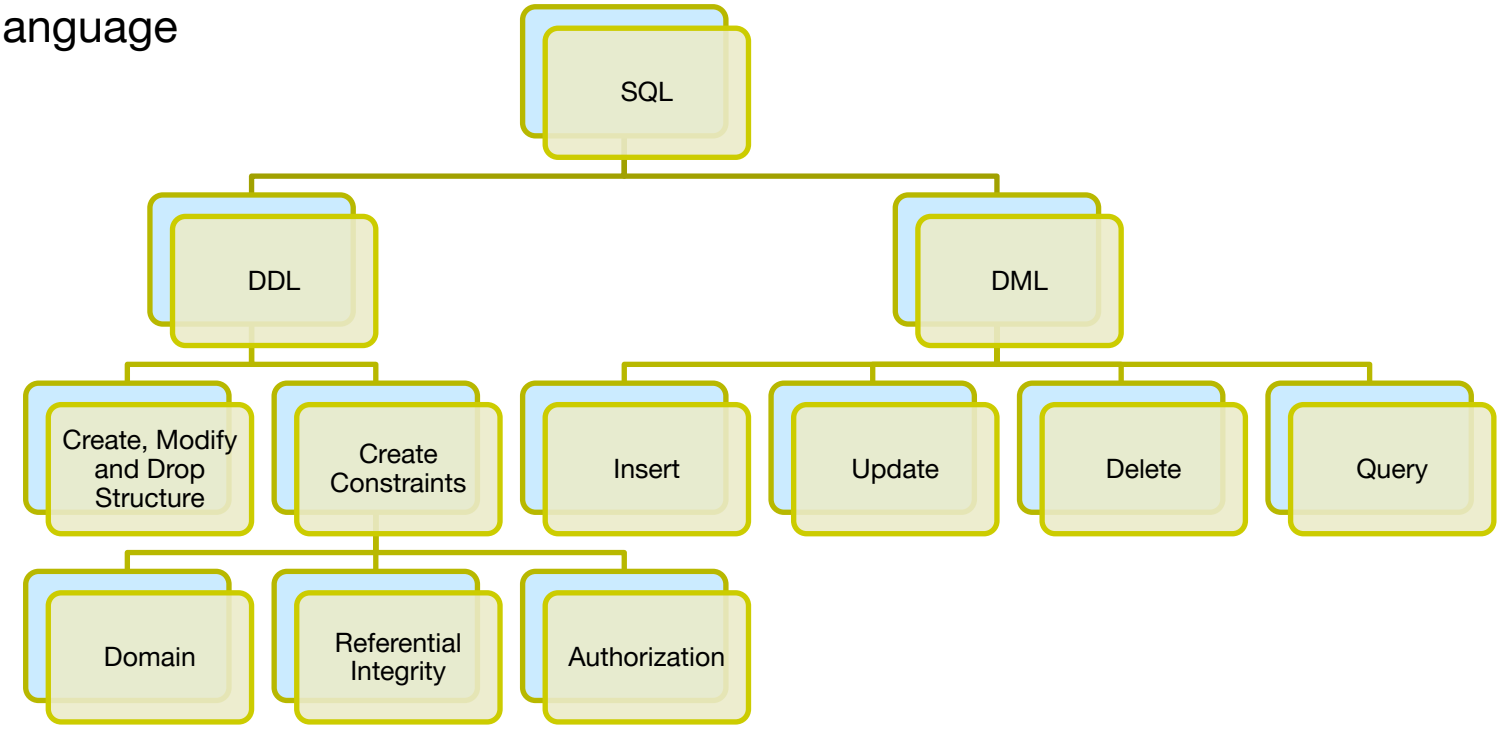
- Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an **instance** of the database.
- The overall design of the database is called the database **schema**.
- The concept of database schemas and instances can be understood by analogy to a program written in a programming language.
- A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an *instance* of a database schema.
- There are several schemas according to the levels of abstraction.
- **Physical schema** – the overall physical structure of the database
- **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
- **View schema (Subschema)** – describe different views of the database.

Physical Data Independence

- The logical schema is by far the most important in terms of its effect on application programs, since programmers construct applications by using the logical schema.
- The physical schema is hidden beneath the logical schema and can usually be changed easily without affecting application programs.
- **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.
- Traditionally, logical schemas were changed infrequently, if at all.
- Many newer database applications, however, require more flexible logical schemas where, for example, different records in a single relation may have different attributes.

1.4 Database Languages

- A database system provides a **data-definition language** (**DDL**) to specify the database schema and a **data-manipulation language** (**DML**) to express database queries and updates.
- In practice, the data-definition and data-manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the SQL (**Structured Query Language**) language



1.4.1 Data Definition Language (DDL)

- **1. Data storage and definition language:**
- The storage structure (B+ tree, B-tree, Heap, Hash) and access methods (sequential, Binary, Indexing, Hashing) used by the database system are specified by a set of statements in a special type of DDL called data storage and definition language. These statements define the implementation details of the database schemas, which are usually hidden from the users.
- **2. Consistency constraints specification:**
- The data values stored in the database must satisfy certain consistency constraints. The DDL provides facilities to specify such constraints. The database system checks these constraints every time the database is updated.
- **i) Domain constraints:** A domain of possible values must be associated with every attribute (integer types, character types, date/time types). Declaring a attribute to be of a particular domain acts as a constraint on the values that it can take. They are easily tested by the system whenever a new data is entered into the database.

1.4.1 Data Definition Language (DDL)

- **ii) Referential integrity:** There are cases where it is to ensure that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation (referential integrity). Database modification can cause violations of referential integrity. When it is violated, the normal procedure is to reject the action that caused the violation.

- **iii) Authorization:** It is required to differentiate among the users as far as the types of access they are permitted on various data values in the database. These differentiations are expressed in terms of authorizations, the most common instance related authorization being: **read** (reading but not modification)**, insert** (but not modification of existing data)**, update** (but not deletion)**, delete**. A user may be assigned all, none or a combination of these types of authorization. Other schema related authorizations might be: **resource** (create), **alter** (object structure modification **alter-add** and **alter-modify**), **drop** (delete object), **rename** (object or component of object) etc.

1.4.1 Data Dictionary

- The processing of DDL statements, just like those of any other programming language, generates some output.
- As DDL creates schema of a table it also updates a special set of tables called **data directory** or **data dictionary**, which contains **metadata**—that is, data about data.
- The data dictionary is considered to be a special type of table that can be accessed and updated only by the database system itself (not a regular user). The database system consults the data dictionary before reading or modifying actual data.
  - 1. Relation-metadata (relation-name, no-of-attributes, storage-organization, location)
  - 2. Attribute-metadata (attribute-name, relation-name, domain-type, position, length)
  - 3. User-metadata (user-name, encrypted-password, group)
  - 4. Index-metadata (index-name, relation-name, index-type, index-attributes)
  - 5. View-metadata (view-name, definition)

1.4.2 The SQL Data Definition Language (DDL)

- A database schema is specified by a set of definitions expressed by a special language called a **data-definition language**. The DDL is also used to specify additional properties of data.

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20) not null**,**
  dept_name **varchar**(20) not null,
  salary **numeric**(8,2) check (salary > 29000)
  primary Key (*ID*),
  foreign key (dept_name) **references** department
  (dept_name) **on delete set null**));

1.4.3 Data Manipulation Language (DML)

- A **data-manipulation language** (**DML**) is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:
  - insert - Insertion of new information into the database.
  - delete - Deletion of information from the database.
  - update - Modification of information stored in the database
  - query - Retrieval of information stored in the database.
- 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 DMLs are usually easier to learn and use than are procedural DMLs (also referred to as non-procedural DMLs).
- A **query** is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a **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.'

- May involve more than one table
- Example to find the instructor ID and department name of all instructors associated with a department with a budget of more than $95,000.

  select *instructor.ID*, *department.dept name*
  from *instructor*, *department*
  where *instructor.dept name= department.dept name* **and**
  department.budget > 95000;

Database Access from Application Program

- Non-procedural query languages such as SQL are not as powerful as a universal Turing machine; that is, there are some computations that are possible using a general-purpose programming language but are not possible using SQL
- 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.
- The Open Database Connectivity (ODBC) standard defines application program interfaces for use with C and several other languages. The Java Database Connectivity (JDBC) standard defines a corresponding interface for the Java language.
- **Application programs** -- are programs that are used to interact with the database in this fashion i.e. 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

1.5 Database Design

- The process of moving from an abstract data model to the implementation of the database proceeds in two final design phases.
- 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?
  - The designer maps the high-level conceptual schema onto the implementation data model of the database system that will be used.
- Physical Design – Deciding on the physical layout of the database
  - file organization and the internal storage structures

1.6 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.

1.6.1 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:** tests for the satisfaction of integrity constraints and checks the authority of users to access data.
  - **Transaction manager:** ensures consistency of the DB despite system failures.
  - **File manager:** allocation of space + data structure used to represent information stored in the disk.
  - **Buffer manager:** responsible for fetching data from disk storage into main memory, and deciding what data to cache in the main memory.

Storage Manager (Cont.)

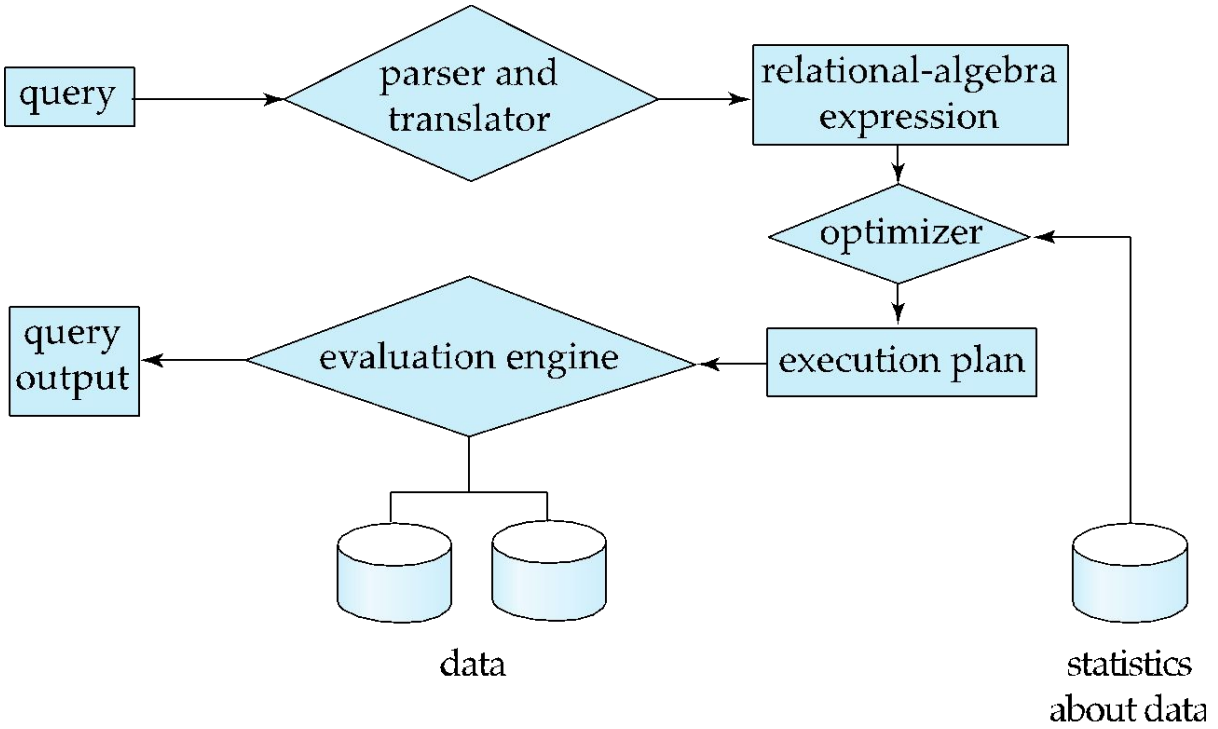
- The storage manager implements several data structures as part of the physical system implementation:
  - Data files -- store the database itself (relations)
  - Data dictionary -- stores metadata about the structure of the database, in particular the schema of the database.
  - Indices -- can provide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

1.6.2 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.
    - The DML compiler 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



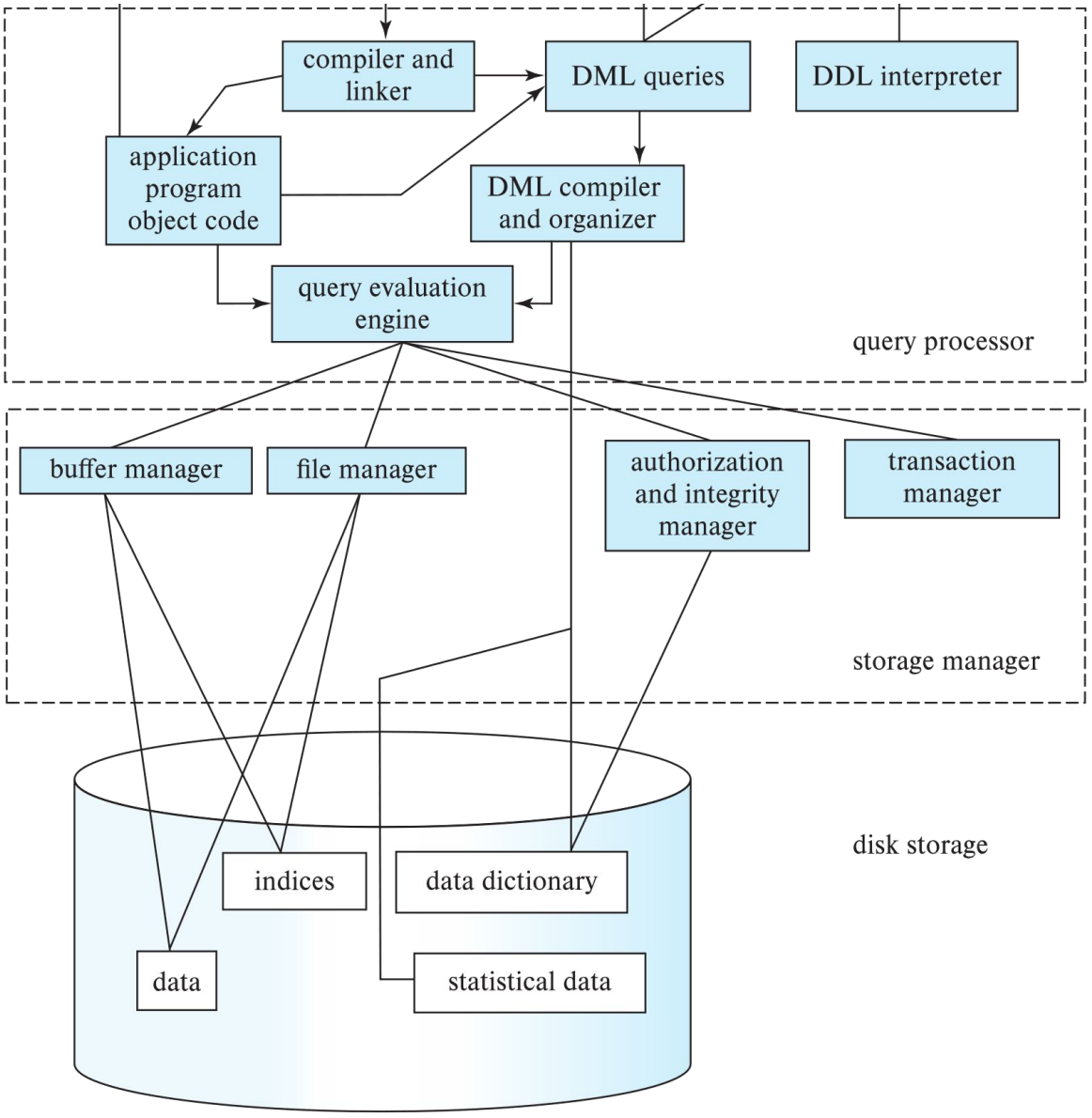
1.6.3 Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
  - e.g., deposit, withdrawal, transfer between accounts
- **ACID** Properties of Transaction:
  - A - Atomicity
  - C - Consistency
  - I - Isolation
  - D – Durability
- **Transaction Manager** = concurrency control manager + recovery manager
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.
  - e.g., two users accessing the same bank account cannot “corrupt” the system or withdraw more than allowed
- **Recovery manager** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
  - e.g., system crash cannot wipe out “committed” transactions

1.7 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: exploits parallel computer architecture
  - Many core shared memory
  - Shared disk
  - Shared nothing
- Distributed databases
  - Geographical distribution
  - Schema/data heterogeneity

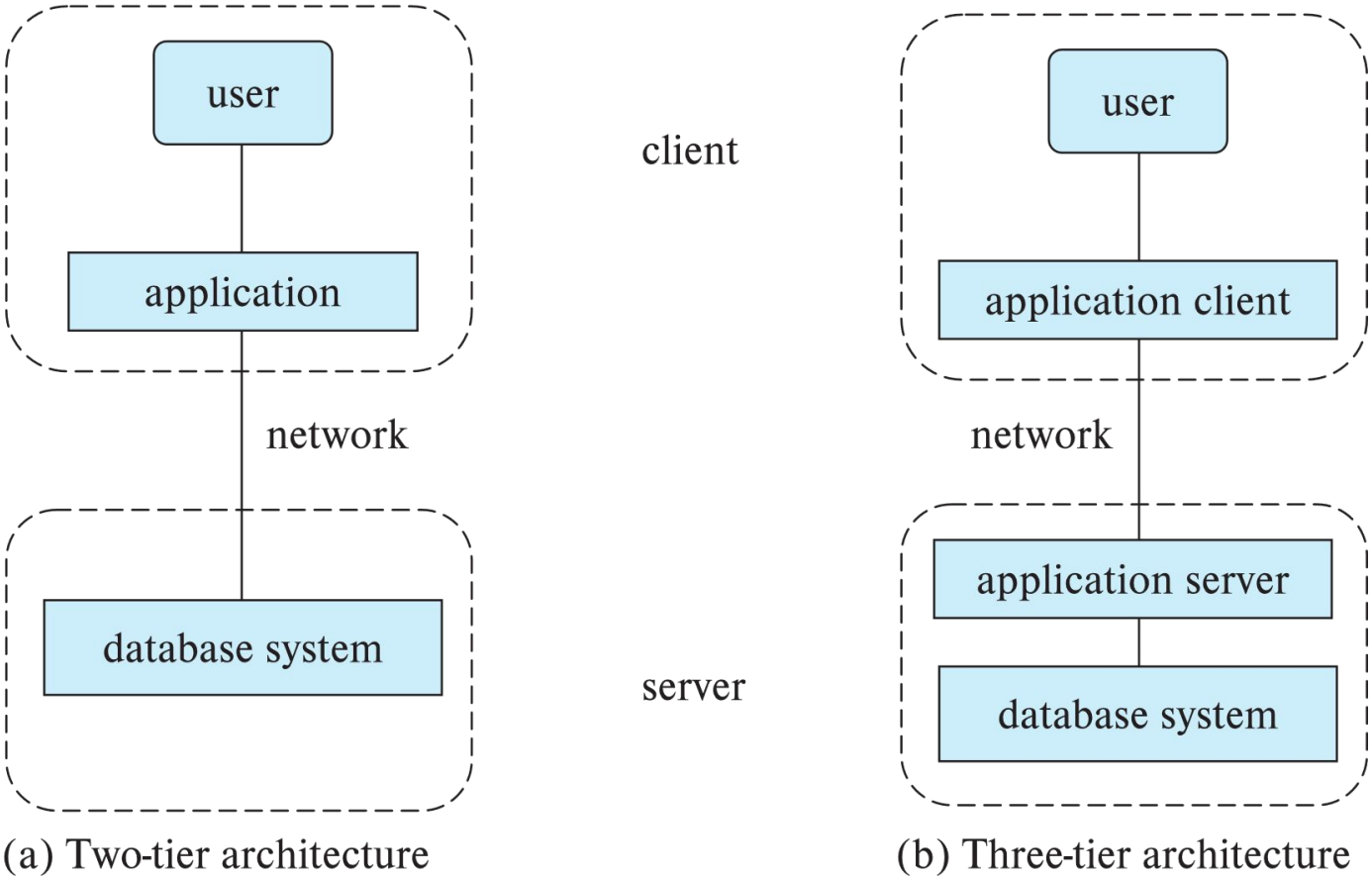
Database Architecture (Centralized/Shared-Memory)



Database Applications

- Most user of a database system today are not present at the site of the database system, But connect to it through a network. So we can differentiate between **client** machines, on which remote database users work, and **server** machines, on which the database systems runs.
- 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 through query language statements. Application program interface standard like ODBC and JDBC are used for interaction between client and server.
  - 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.
- Three-tire applications are more appropriate large applications that run on the World Wide Web.

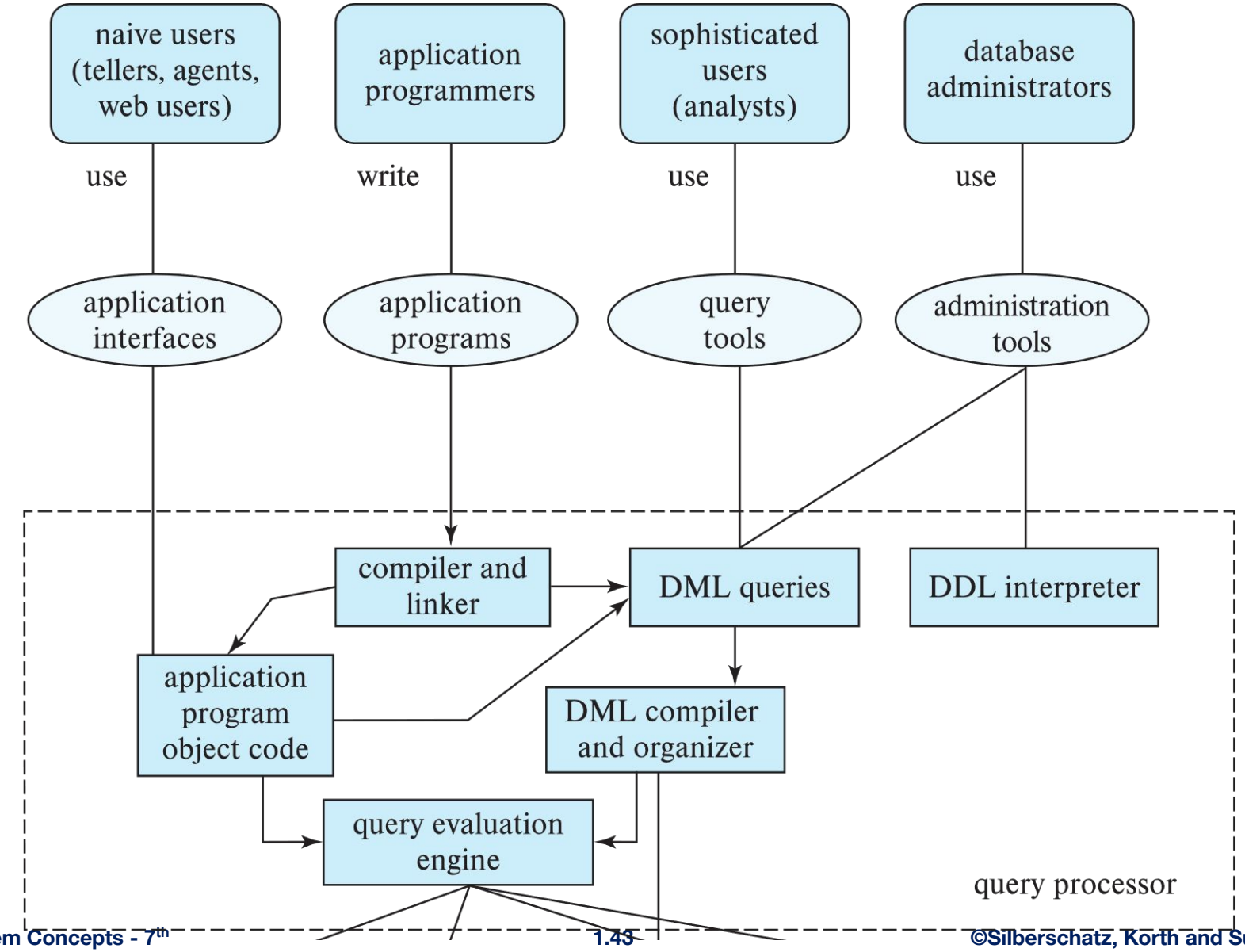
Two-tier and three-tier architectures



(a) Two-tier architecture

(b) Three-tier architecture

Database Users



1.8 Database Users and Administrators

- A primary goal of a database system is to retrieve information from and store new information in the database. People who work with a database can be categorized as database users or database administrators.
- **1.8.1 Database Users**
- Users are differentiated by the way they expect to interact with the system. Four different types:
- **1. Naive users** – are unsophisticated users who interact with the system by invoking one of the permanent application programs that have been written previously.
  - E.g. people accessing database over the web, bank tellers, and clerical staff.
  - The typical user interface for naïve users is a forms interface, where the user can fill in appropriate fields of the form. They may also simply read reports generated from the database.

1.8.1 Database Users

- **2. Application programmers** – are computer professionals who write application programs. Application programmers can choose from many tools to develop user interface.
  - **Rapid application development (RAD)** tools are tools that enable an application programmer to construct forms and reports with minimal programming effort.
- **3. Sophisticated users** – interact with the system without writing programs. Instead, they form their requests in a database query language. Analysts who submit queries to explore data in the database fall in this category.
  - e.g., analyst looking at sales data (OLAP – Online analytical processing), data mining

1.8.1 Database Administrators

- One of the main reasons for using DBMS is to have central control both data and the programs that access that data. A person who has such central control over the system is called **database administrator (DBA)**.
- The functions of a database administrator (DBA) include:
- **1. Schema definition:** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **2. Storage structure and access method definition:** File organization (sequential, heap, hash, B^+ tree), organization of records in a file (fixed length or variable length), index definition (ordered index, hash index).
- **3. Schema and physical-organization modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve the performance.

1.8.1 Database Administrators

- **4. Granting of authorization for data access:** By granting different types of authorization, the DBA can regulate which parts of the database various users can access.
- **5. Specifying integrity constraints:** The DBA implements key declaration (primary key, foreign key), trigger, assertion, business rules of the organization.
- **6. Routine maintenance:**
- i) Periodically backing up the database, either onto tapes or remote servers, to prevent loss of data in case of disasters
- ii) Ensuring that enough disk space is available for normal operations and upgrading disk space as required
- iii) Monitoring jobs running on the database and ensuring better performance

1.9 History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - 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
  - Parallel and distributed database systems
    - Wisconsin, IBM, Teradata
  - 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.
  - 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

End of Chapter 1