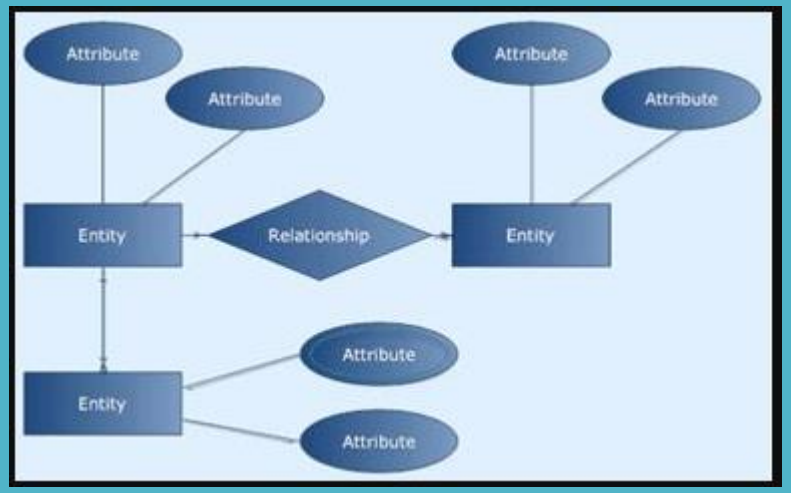
El-Shorouk Academy

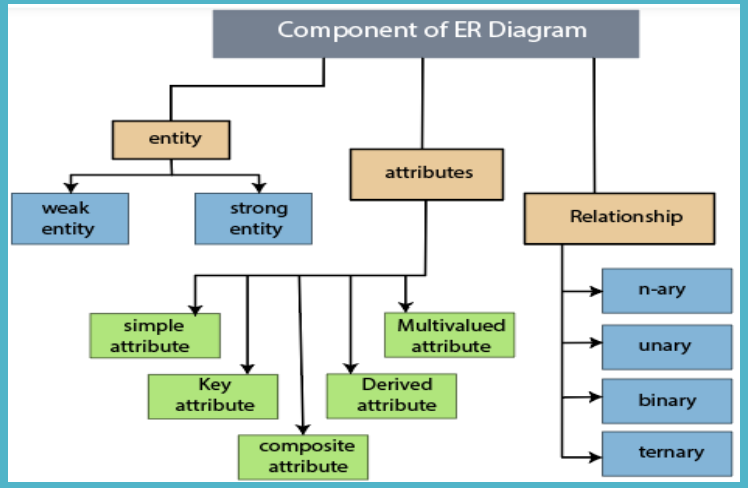
The Higher Institute of Engineering, El-Shorouk City

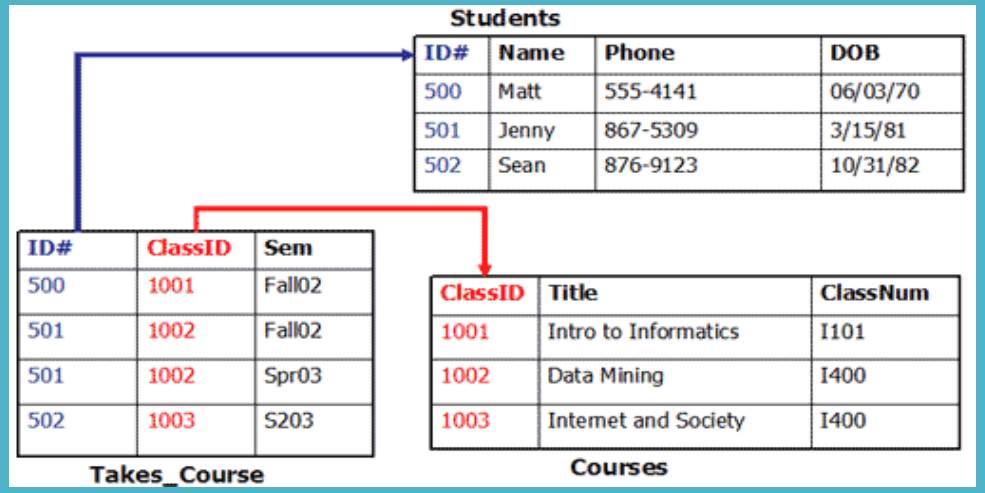
Communications and Computer Engineering Department

Data Base
CCE 395 2013
CCE 333 [2019]

Dr. Ahmed El-Shafei







Week 1 : 13/2/2023 2013

Week 1 : 16/2/2023 2019

اسم المقرر	**Data Base** **قواعد بيانات**
رقم المقرر	**CCE 395 [2013], CCE 333 [2019]** **الفرقة الثالثة هندسة الحاسبات والتحكم** ***Third Year Computer and Control Engineering***
توزيع الدرجات	***Grading System (next Slide)***
عدد الساعات	محاضرة : 22 ساعة أسبوعيا فصل : 1 - معمل: 2 2 ساعة أسبوعيا الإجمالى : 5 4 ساعات أسبوعيا

Instructor : Dr. Ahmed El-Shafei, email: dr.ahmed.alshafaay@sha.edu.eg
Graduate Teaching Assistant : Eng. Amira, email: amira.elsayed@Sha.edu.eg
Eng. Aya Mostafa, email: Aya.mostafa@sha.edu.eg
Eng. Hagar Atef, email: hagar.atef@sha.edu.eg

Grading System

Final written Exam :		***60%***	**40** **90**
Teacher opinion ……..			**30** **30**
Attendance	**-**	***20%***	**-**
Reports / sheets / Activities	**-**		**-**
Quiz (1)	**40% of 30 marks**		***12*** ***12***
Quiz (2)	**-**		**-**
Mid-term exam	**60% of 30 marks**		***18*** ***18***
Practical / Oral ……..			**30** **30**
Practical Attendance	**10% of 30 marks**	***20%***	***3*** ***3***
Lab. reports	**10% of 30 marks**		***3*** ***3***
Lab. Activities / Projects	**-**		**-**
Final oral / practical exam	**80% of 30 marks**		***24*** ***24***
Total		***100%***	**100** **150**

- *Class and section participation*
- *Homework assignments*
- *Group Projects*
- *Two quizzes*
- *Mid Term*
- *Practical and Oral exam + Project discussion*
- *Final Exam*

Course Description:

- Data Base (DB) concepts, relational data base concepts, data models, data base languages, Structured Query Language (SQL), data base design using Entity Relationship Diagram (ERD) model, data base design using normalization, data base security and integrity, data base recovery and concurrency, data base applications development.

❑ *Essential book:*

- Abraham Silberschatz, Henry Korth , and S. Sudarshan, "Database System Concepts", 7th Edition, 2020, (Published: February 28, 2019). ISBN-13: 978-0078022159, ISBN-10: 0078022150

❑ *Recommended books:*

- Ramez Elmasri and Shamkant B. Navathe, "FUNDAMENTALS OF Database Systems", SEVENTH EDITION, 2016, ISBN-10: 0-13-397077-9, ISBN-13: 978-0-13-397077-7
- Dr. Mukesh Negi, "Fundamentals of Database Management System: Learn essential concepts of Database Systems", 1st Edition, BPB Publication, India, 2019. ISBN-13: 978-9388176620, ISBN-10: 9388176626

❑ Scientific papers, publications, websites ... etc.:

- Egyptian Knowledge Bank
- www.elseiver.com, Springer Series
- www.geeksforgeeks.com
- www.tutorialspoint.com

❑ Handout:

Course Learning Objectives :

- **By the end of the course the student should be able to:**
  - **Illustrate the basic concepts of Data Base (DB).**
  - **Differentiate between different types of database models.**
  - **Understanding data base languages Structured Query Language (SQL), Data Definition Language (DDL) and Data Manipulation Language (DML).**
  - **Apply database design and dependencies between database tables.**

Course Content:

1. Databases and database users.
2. Database system concepts and architecture.
3. Relational data model.
4. Relational database constraints.
5. Basic Structured Query language (SQL).
6. Complex queries, triggers, views.
7. Relational algebra.
8. Entity-Relationship (ER) model.
9. Weak entity types.
10. Basics of functional dependencies.
11. Normal forms.
12. Multivalued and join dependency.

Reports Formats:

- **All reports in PDF format unless otherwise stated.**
- **File name is the student *ID_xx*, *where xx : report number***
- **The cover page should include:**
  - **Student ID,**
  - **Student Name,**
  - **Section number,**
  - **Student Serial,**
  - **Report title, and**
  - **report date**

Chapter 1

Introduction

Introduction to Database Systems

- The world is drowning in data!
- Need computer scientists to help manage this data
  - Help domain scientists achieve new discoveries
  - Help companies provide better services (e.g., Facebook)
  - Help governments (and universities!) become more efficient

Outline

- Introduction to Database System
- Data Management Approaches
- Database Management System (DBMS)
- Actors on the Scene

Introduction to Database Systems

- **What is Database ?**
  - A collection of related data
  - It is a logically coherent collection of data with some inherent meaning
- **What is Data ?**
  - It is known facts (but unprocessed) that can be recorded and have an implicit meaning
- **What is Mini-world Unprocessed ?**
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university
- **What is Database Management System (DBMS) ?**
  - A software package/system to facilitate the creation and maintenance of a computerized database
- **What is Database System ?**
  - The DBMS software together with the data itself. Sometimes, the applications are also included

Data Management Approaches

- There are three approaches (levels) with their own advantage and disadvantage
  - Manual Approach
  - File-Based Approach
  - Database Approach

Data Management Approaches …

Manual File Handling System

- The primitive and traditional way of information handling where cards and papers are used for the purpose
  1. This work may well if the number of items to be stored is small
  2. Events and objects are written on files (paper)
  3. Each of the files containing various kinds of information that labeled and stored in one or more cabinets
  4. The cabinets could be kept in safe places for security purpose based on the sensitivity of the information contained in it
  5. Insertion and retrieval are done by searching first for the right cabinet then for the right file then the information
  6. It is possible to use manual indexing system to facilitate access to the data

Data Management Approaches …

Limitations of Manual File Handling Systems

- Problem of Data Organization
- It requires intensive human labor
- Problem of Efficiency
- Prone to error
- Difficult to update, retrieve, manipulate, duplicate and integrate
- Limited to small size information (Scale up problem)
- Cross referencing is difficult

Data Management Approaches …

File-Based Approach

- It were developed as better alternatives to paper based (manual) file systems
- It were an early attempt to computerize the manual file system
- Is a decentralized computerized data handling method
- Is a collection of application programs performing services for end users
- Every application defines and manages its own data, the system is subjected to serious data duplication problem
- Files, in traditional files-based approach, is a collection of records which contains logically related data (as pros)
- Files stored on computers could be accessed more effciently (as pros)

Data Management Approaches …

Limitations of File-Based Systems

- Data Redundancy (Duplication of data)
- Same data is held by different programs
- Wasted space (due to uncontrolled duplication of data)
- Separation and isolation of data

Week 2 : 20/2/2023 2013

Week 2 : 23/2/2023 2019

Data Management Approaches …

Database Approach

- It is a shared corporate resource that is integrated within a minimum amount or no duplication
- Has locally related data comprised entities, attributes, and relationships of an organization's information
- The key idea behind the database concept is separating the data from the application program (data independence)
- There is a separation between data definition and data application
- In addition to containing data required by an organization, database also contains a description of the data which called as “Metadata" or “Data Dictionary". It is a self descriptive collection on integrated records

Data Management Approaches …

Limitations and Risks of Database Approach

- It needs a new professional or specialized personnel called Database Administrator
- It needs high cost to incurred to develop and maintain the system
- There is complexity in designing and managing data
- It requires complex backup and recovery services from the user's perspective
- Has high impact on the system when failure occurs to the central system
- May reduce the performance due to centralization
- It requires high effort to transfer data from current system

Database Management System (DBMS)

- **What is a DBMS ?**
  - A software system that enables user to create, define and maintain the database and provides controlled access to this database
  - it is software package that makes it possible to define, construct, manipulate and share databases among various users and applications
  - Defining Database involves specifying the data types, structures and constraints for the data to be stored in the database
  - Constructing Database: the process of storing the data itself on some storage medium that is controlled by the DBMS
  - Manipulating Database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the mini world, and generating reports from the data
  - Sharing Database: it allows multiple users and programs to access the database concurrently

Database Management System (DBMS) …

Services by DBMS

- A full-scale DBMS should at least have the following services to provide to users
  - Data storage, retrieval and update in the database:

    must furnish users with the ability to store, retrieve and update data
  - A user accessible catalogue: it must furnish a catalog in which descriptions of data items are stored and which is accessible to users
  - Transaction support service: ALL or NONE transaction, which minimizes data inconsistency
  - Concurrency control services: access and update on the database by different users simultaneously should be implemented correctly
  - Recovery services: A mechanism for recovering the database after failure must be available
  - Authorization Services (Security): must support the implementation of access and authorization service to database administrator and users

Database Management System (DBMS) …

DBMS Components

The DBMS environment has five components

- **Hardware**
- **Software**
- **Data**
- Procedure
- People

- **Hardware**

1. **Might range from single PC to network of computers**
2. **Depends on the organization's requirement and the DBMS used (Some run on a particular hardware while others on a wide variety of Hardware) E.g., Client Server architecture**

Database Management System (DBMS) …

DBMS Components…

- **Software:**
  - The DBMS software itself
    1. MS Access
    2. Oracle
    3. My SQL
    4. SQL Server
    5. Other Application Programs
    6. Operating System
- **Data:**
  - the most important component to the user of the database. There are two types of data in the database approach
    1. **Operational Data:** data stored in the system to be used by the users
    2. **Meta-Data:** it is used to store information about the database itself

Database Management System (DBMS) …

DBMS Components…

- **Procedure:**
  - it is rules and regulations on how to govern the design and use of database. It include procedures like how to log on to the DBMS, how to start and stop DBMS
- **People:**
  - People in the organization responsible to design, implement, manage and administer and use of the database

Database Management System (DBMS) …

Database Users

- Users may be divided into:
  1. Those who use and control the database content, and those who design, develop and maintain database applications (called “Actors on the Scene"), and
  2. Those who design and develop the DBMS software and related tools, and the computer systems operators (called “Workers Behind the Scene")

Actors on the Scene

Database Administrators

- Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations

Database Designers

- Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs

End-Users

- They use the data for queries, reports and some of them update the database content

Actors on the Scene …

Categories of End-Users

- **Casual:** access database occasionally when needed
- **Naive or Parametric:** they make up a large section of the end-user population; e.g., bank-tellers
- **Sophisticated:**
  - ✓ These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities
  - ✓ Many use tools in the form of software packages that work closely with the stored database
- **Stand-Alone:**
  - ✓ Mostly maintain personal databases using ready-to-use packaged applications
  - ✓ An example is a tax program user that creates its own internal database
  - ✓ Another example is a user that maintains an address book

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.
- Banking and finance
  - Banking: 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
  - Social-media: For keeping records of users, connections between users (such as friend/follows information), posts made by users, rating/like information about posts, etc.
  - Online advertisements
- Document databases
- Navigation systems: For maintaining the locations of various places of interest along with the exact routes of roads, train systems, buses, etc.

Database Applications Examples (Cont.)

- These tasks are addressed in two steps.

  - The first mode is to support online transaction processing, where many users use the database, with each user retrieving relatively small amounts of data, and performing small updates. This is the primary mode of use for most users of database applications such as those that we outlined earlier.

  - The second mode is to support data analytics, that is, the processing of data to draw conclusions, and infer rules or decision procedures, which are then used to drive 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 particular user.

Database Applications Examples (Cont.)

- These tasks are addressed in two steps
  - First, data-analysis techniques attempt to automatically discover rules and patterns from data and create predictive models. These models take as input attributes ("features") of individuals, and output predictions such as likelihood of paying back a loan, or clicking on an advertisement, which are then used to make the business decision.

As another example, manufacturers and retailers need to make decisions on what items to manufacture or order in what quantities; these decisions are driven significantly by techniques for analyzing past data and predicting trends.

The cost of making wrong decisions can be very high, and organizations are therefore willing to invest a lot of money to gather or purchase required data and build systems that can use the data to make accurate predictions.

Database Applications Examples (Cont.)

The field of datamining 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.

Purpose of Database Systems

- **Database systems offer solutions to all problems of file management which are as follows:**
  - **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
    - 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

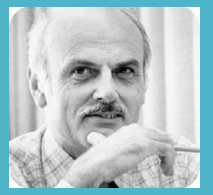
View of Data

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
  - Data consistency constraints
- Relational model (Chapter 2 and Chapter 7)
- Entity-Relationship (E-R) data model (mainly for database design) (Chapter 6)
- Object-based data models (Object-oriented and Object-relational) (Chapter 8)
- Semi-structured data model (XML) (Chapter 8)
- Other older models:
  - Network model
  - Hierarchical model

Relational Model



Ted Codd
Turing Award 1981

- **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

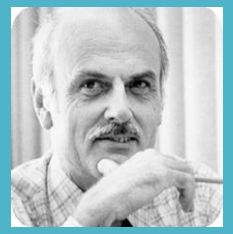
(a) The *instructor* table

Introduction to Database Systems

- Turing Awards in Data Management

Charles Bachman, 1973

IDS and CODASYL



Ted Codd, 1981

Relational model

Jim Gray, 1998

Transaction processing

Michael Stonebraker, 2014

INGRES and Postgres

You could be next!!

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

Week 2 : 27/2/2023 2013

Week 2 : --/2/2023 2019

Data Abstraction

- For the system to be usable, it must retrieve data efficiently.
- The need for efficiency has led database system developers to use complex data structures to represent data in the database. 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:

  - Physical level. The lowest level of abstraction describes *how* the data are 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.

Data Abstraction

- Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.
- 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.

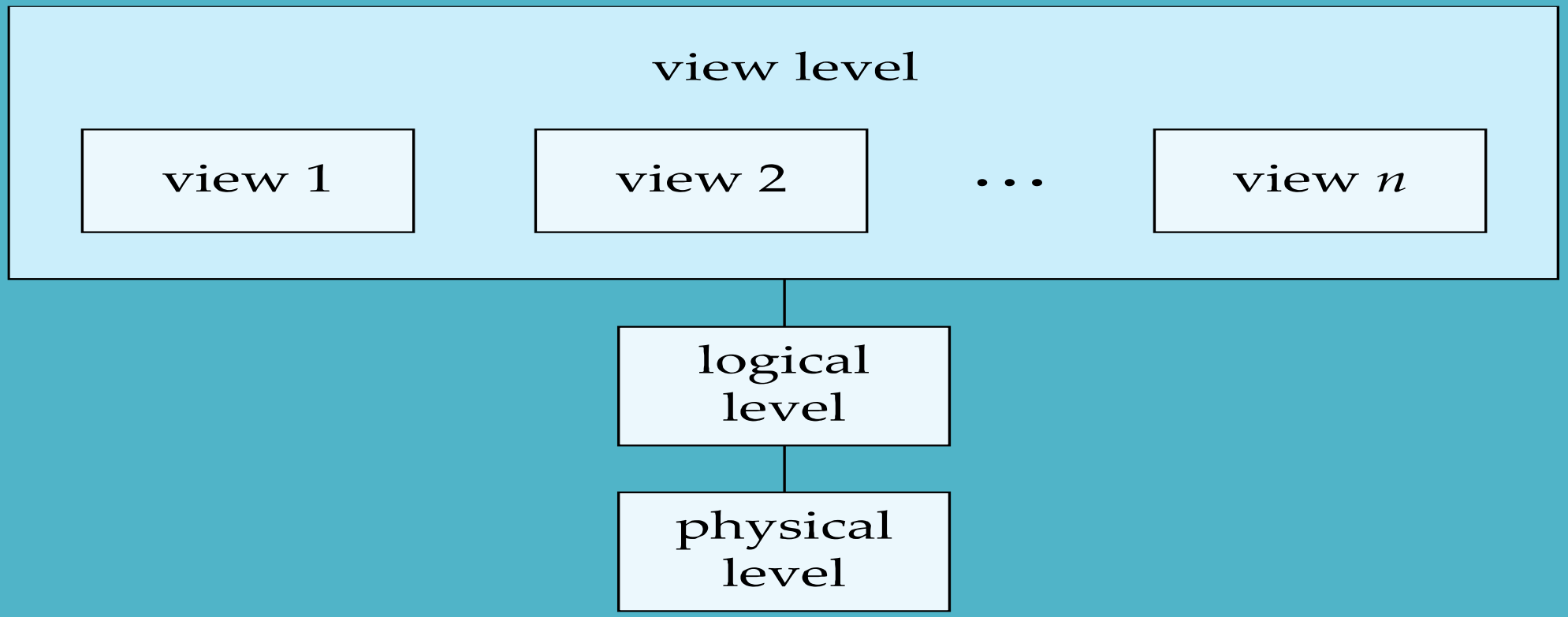
  Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system.
- The system may provide many views for the same database.

Next Slide shows the relationship among the three levels of abstraction.

- An important feature of data models, such as the relational model, is that they hide such low-level implementation details from not just database users, but even from database-application developers.

Data Abstraction

An architecture for a database system. The three levels of data abstraction.



Data Abstraction

- The database system allows application developers to store and retrieve data using the abstractions of the data model and converts the abstract operations into operations on the low-level implementation.
- An analogy to the concept of data types in programming languages may clarify the distinction among levels of abstraction. Many high-level programming languages support the notion of a structured type. We may describe the type of a record abstractly as follows: (language dependent C and C++. In JAVA, a simple class can be defined to the same effect)

```
type instructor = record
        ID : char (5);
        name : char (20);
        dept name : char (20);
        salary : numeric (8,2);
end;
```

Data Abstraction

- **A university organization may have several such record types, including:**
  - ***department*, with fields *dept name*, *building*, and *budget*.**
  - ***course*, with fields *course id*, *title*, *dept name*, and *credits*.**
  - ***student*, with fields *ID*, *name*, *dept name*, and *tot cred*.**
- **At the physical level, an instructor, department, or student record can be described as a block of consecutive bytes. The compiler hides this level of detail from programmers.**
- **Similarly, the database system hides many of the lowest-level storage details from database programmers.**
- **Database administrators, on the other hand, may be aware of certain details of the physical organization of the data.**

Data Abstraction

- For example, there are many possible ways to store tables in files.
- One way is to store a table as a sequence of records in a file, with a special character (such as a comma) used to delimit the different attributes of a record, and another special character (such as a new-line character) may be used to delimit records.

  If all attributes have fixed length, the lengths of attributes may be stored separately, and delimiters may be omitted from the file.

  Variable length attributes could be handled by storing the length, followed by the data.

  Databases use a type of data structure called an index to support efficient retrieval of records; these too form part of the physical level.

Data Abstraction

- **At the logical level, each such record is described by a type definition, as in the previous code segment.**

  The interrelationship of these record types is also defined at the logical level; a requirement that the *dept name* value of an *instructor* record must appear in the *department* table is an example of such an interrelationship. Programmers using a programming language work at this level of abstraction.

  Similarly, database administrators usually work at this level of abstraction.

Data Abstraction

- **Finally, 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 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.*

Instances and Schemas

- Database systems have several schemas, partitioned according to the levels of abstraction. The *physical schema* describes the database design at the physical level, while the *logical schema* describes the database design at the logical level.

- A database may also have several schemas at the view level, sometimes called *subschemas*, that describe different views of the database.

- Of these, 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.

Instances and Schemas

- The *physical schema* is hidden beneath the *logical schema* and can usually be changed easily without affecting application programs.

- Application programs are said to exhibit physical data independence if they do not depend on the physical schema and thus need not be rewritten if the physical schema changes.

- We also note that it is possible to create schemas that have problems, such as unnecessarily duplicated information.
  - For example, suppose we store the department budget as an attribute of the instructor record. Then, whenever the value of the budget for a department (say the Physics department) changes, that change must be reflected in the records of all instructors associated with the department.

Instances and Schemas – Summary

- Like 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

Physical Data Independence

- **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.**

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 language. Almost all relational database systems employ the SQL language, which we cover in detail in Chapter 3, Chapter 4, and Chapter 5.

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
  - DML also known as query language (Retrieval, Insertion, Deletion, Modification)
- 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.
- Declarative DMLs are also referred to as non-procedural DMLs
- 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.'
```

- SQL is NOT a Turing machine equivalent language
- 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

- **ODBC is an SQL-based Application Programming Interface (API) created by Microsoft that is used by Windows software applications to access databases via SQL.**

- **JDBC is an SQL-based API created by Sun Microsystems to enable Java applications to use SQL for database access.**

  - **These APIs provide communications between an application residing on a client machine and a data source residing on the same client machine or on another server computer.**

  - **Micro Focus XDBC includes both ODBC and JDBC drivers for COBOL data files.**

  - **Micro Focus XDBC gives ODBC-enabled Windows applications (like those in Microsoft Office) and Java applications access to indexed, relative, and fixed-length sequential data files.**

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 Design

- **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**

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

Storage Manager (Cont.)

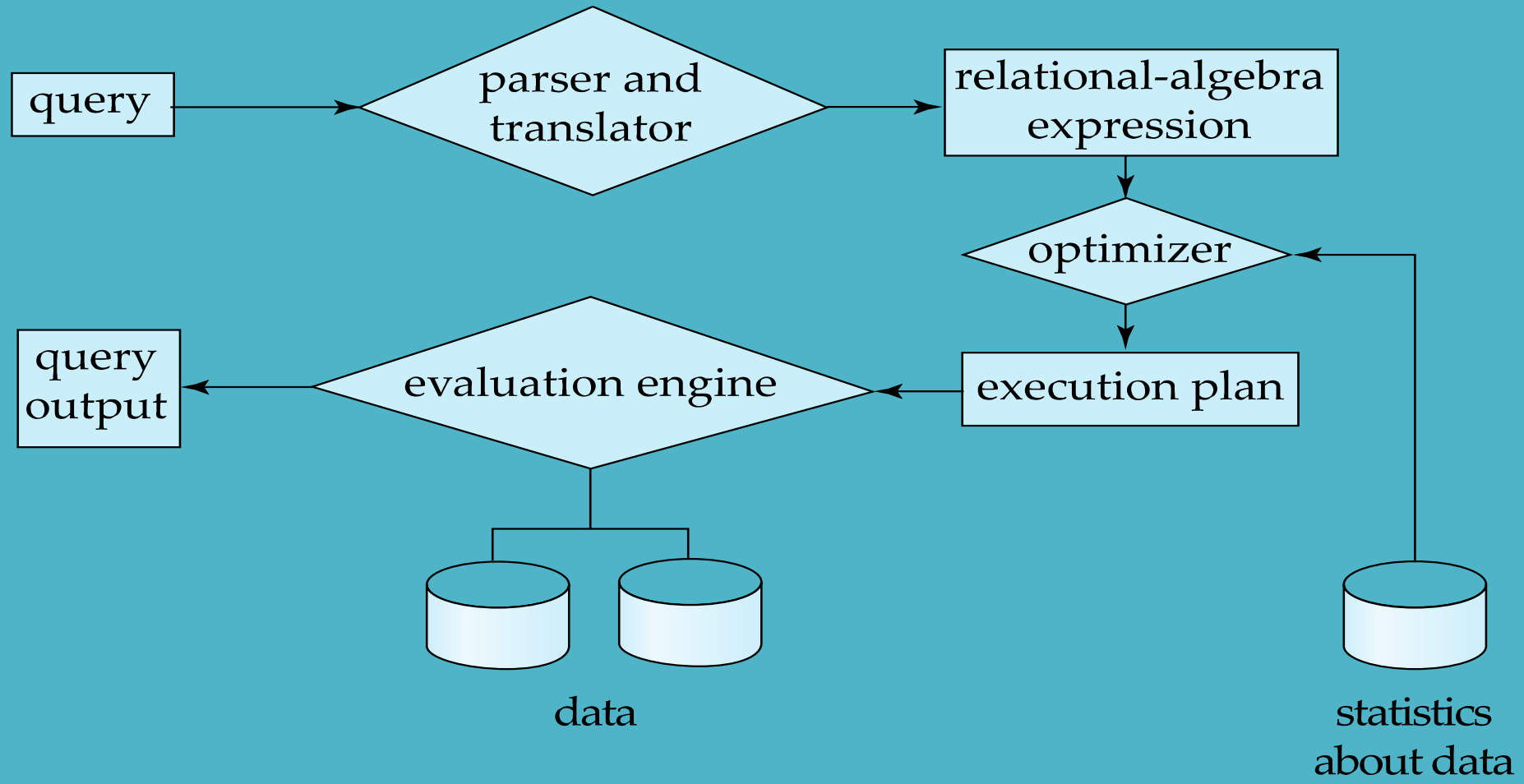
- The storage manager implements several data structures as part of the physical system implementation:
  - Data files -- store the database itself
  - Data dictionary -- stores metadata about the structure of the database, in particular the schema of the database.
  - Indices -- can provide fast access 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.**
    - **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



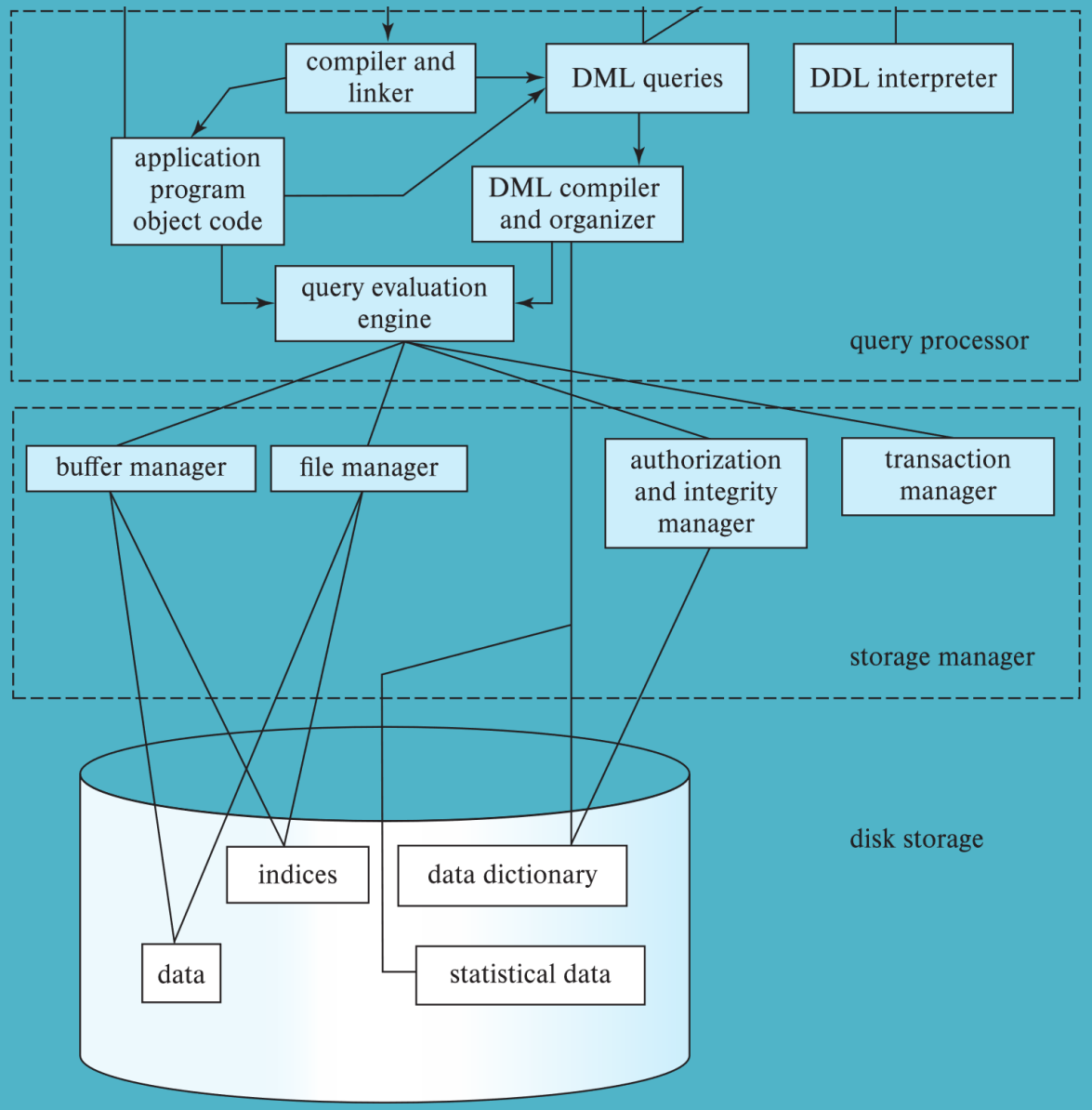
Transaction Management

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

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
  - Many core shared memory
  - Shared disk
  - Shared nothing
- Distributed databases
  - Geographical distribution
  - Schema/data heterogeneity

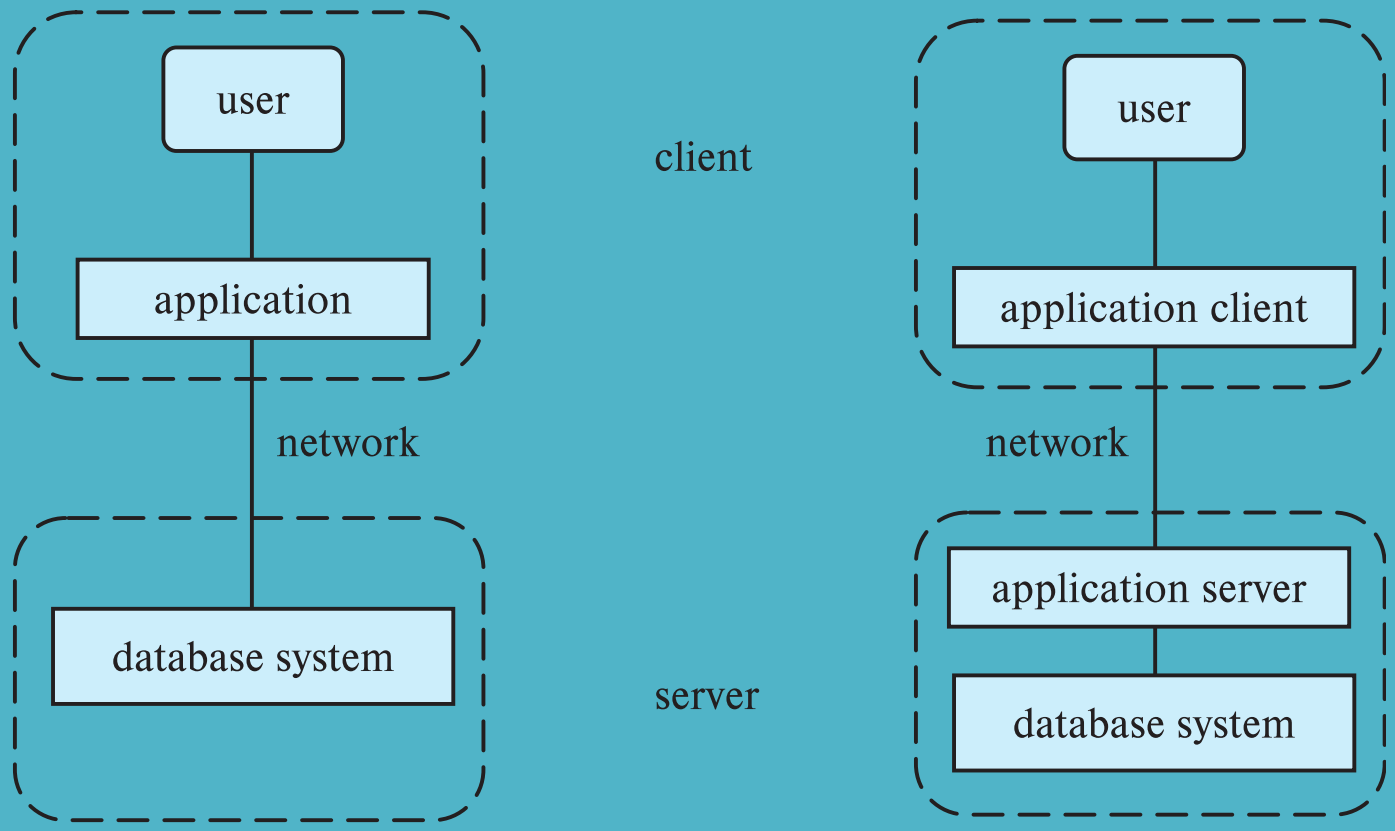
Database Architecture (Centralized/Shared-Memory)



Database Applications

- 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.

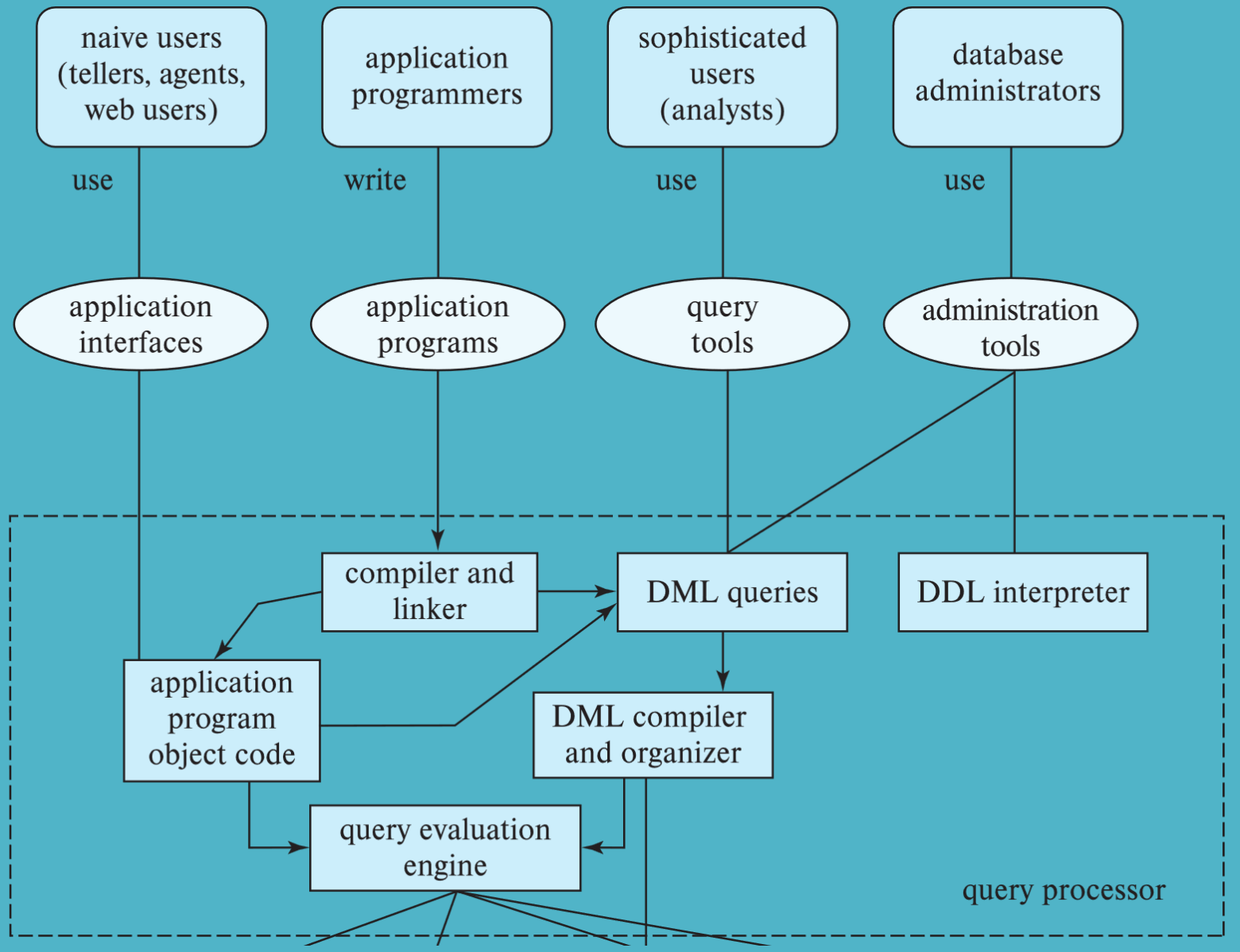
Two-tier and three-tier architectures



(a) Two-tier architecture

(b) Three-tier architecture

Database Users



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**

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

ThankYou