

Chương 11: DATABASES

Objectives After studying this chapter, the student should be able to: ▪ Define a database and a database management system (DBMS) and describe the components of a DBMS. (định nghĩa database là gì và quản lý database và cấu trúc của nó DBMS-database manager system)

- Describe the architecture of a DBMS based on the ANSI/SPARC definition.
- Define the three traditional database models: hierarchical, networking, and relational.
- Describe the relational model and relations
- Understand operations on a relational database based on commands available in SQL.
- Describe the steps in database design.
- Define ERM and E-R diagrams and explain the entities and relationships in this model.
- Define the hierarchical levels of normalization and understand the rationale for normalizing the relations.
- List database types other than the relational model

1 - INTRODUCTION

1. Introduction

- **Data storage** traditionally used individual, unrelated files, sometimes called **flat-files**. In the past, each application program in an organization used its own file.
 - In a university, for example, each department might have its own set of files: the record office kept a file about the student information and their grades, the financial aid office kept its own file about students that needed financial aid to continue their education, the scheduling office kept the name of the professors and the courses they were teaching, the payroll department kept its own file about the whole staff (including professors), and so on.
- Today, however, all of these flat-files can be combined in a single entity, the database for the whole university.

Definition: A database is a collection of related, logically coherent, data used by the application programs in an organization.

2. Advantages of databases

- **Less redundancy** (giảm sự phí phạm): In a flat-file system there is a lot of redundancy. For example, in the flat-file system for a university, the names of professors and students are stored in more than one file.
- **Inconsistency avoidance** (tránh mâu thuẫn dữ liệu): If the same piece of information is stored in more than one place, then any changes in the data need to occur in all places that data is stored. For example, if a female student marries and accepts the last name of her husband, the last name of the student needs to be changed in all files that hold information about the student. Lack of care may create inconsistency in the data.
- **Efficiency**: A database is usually more efficient than a flat-file system, because a piece of information is stored in fewer locations.
- **Data integrity (toàn vẹn dữ liệu)**: In a database system it is easier to maintain data integrity (see Chapter 16) because a piece of data is stored in fewer locations.
- **Confidentiality (bảo mật)**: It is easier to maintain the confidentiality of the information if the storage of data is Centralized in one location

3. Database management systems

- A database management system (DBMS) defines, creates, and maintains a database. The DBMS also allows controlled access to data in the database. A DBMS is a combination of five components: hardware, software, data, users, and procedures (Figure 14.1).
- **Hardware** is the physical computer system that allows **access to data**.
- **Software** is the actual program that allows users to access, maintain, and update data
- Data in a database is stored physically on the storage devices. In a database, data is a separate entity from the software that accesses it.
- Users are the people who control and manage the databases and perform different types of operations the databases in the database management system.
- Procedures (quy trình, quy tắc) refer to general rules and instructions help to design the database and to use a database management system

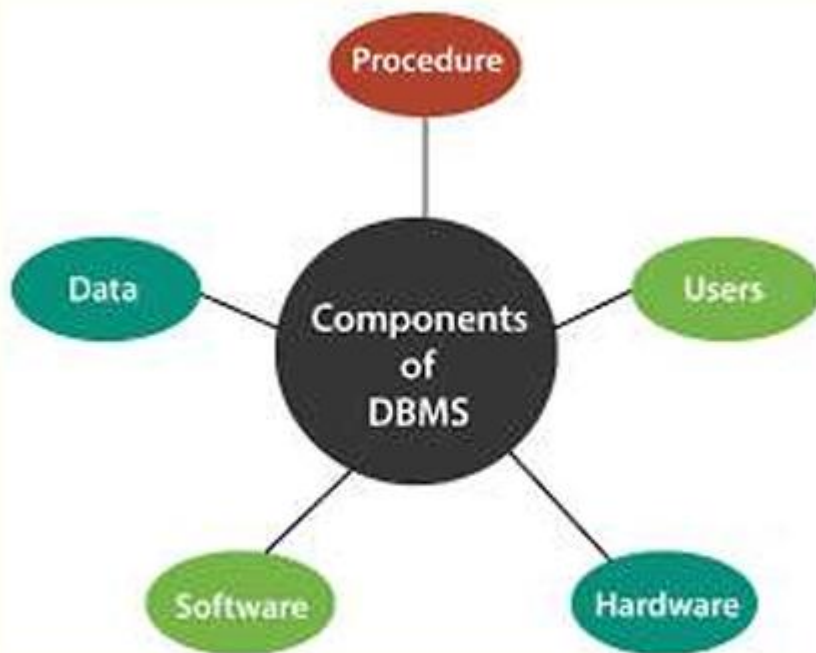
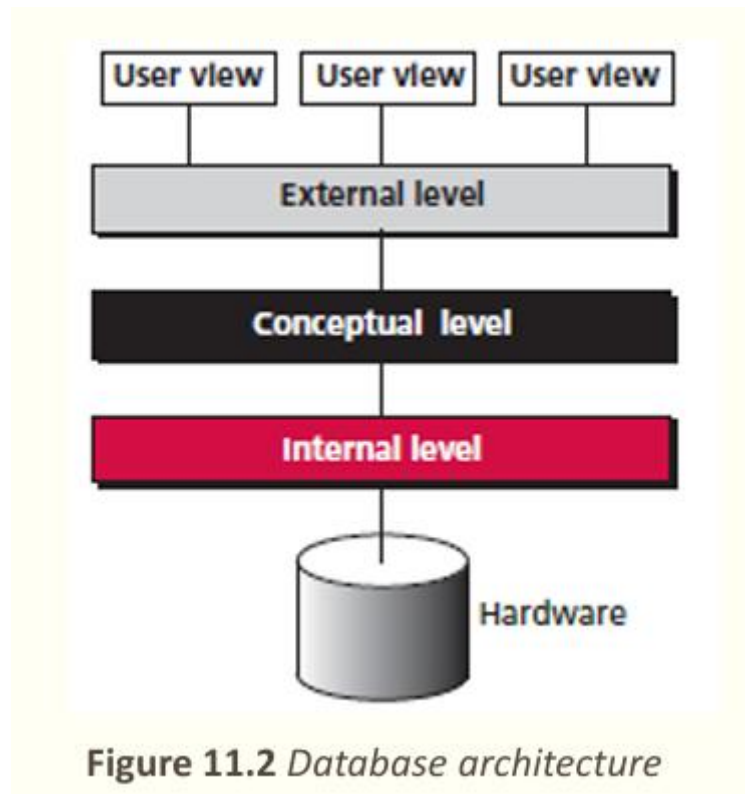


Figure 11.1 *DBMS components*

2- DATABASE ARCHITECTURE

1. Introduction

- The American National Standards Institute/Standards Planning and Requirements Committee (ANSI/SPARC) has established a three-level architecture for a DBMS: **internal, conceptual, and external (Figure 14.2).**
 - Internal level determines where data is actually stored on the storage devices. This level deals with low-level access methods and how bytes are transferred to and from storage devices.
 - Conceptual level defines the logical view of the data. The data model is defined on this level, and the main functions of the DBMS, such as queries, are also on this level.
 - External level interacts directly with the user (end users or application programs). It changes the data coming from the conceptual level to a format and view that is familiar to the users.



2. The internal level

The internal level has an internal schema (lược đồ), which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

- ❑ Physical representation of the DB on the computer.
- ❑ How the data is stored in the database. describes structure the of the complete
- ❑ Physical implementation of the DB to achieve optimal run, time performance and storage space utilization, Storage space allocation for data and indexes, Record description for storage, Record placement, Data compression, encryption

3. The conceptual level

The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and for a concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented.

- What data is stored in the database.
- The logical structure of the entire database as seen by DBA.
- The relationships among the data.

- Complete view of the data requirements of the organization, independent of any storage consideration.
- Represents:- Entities, attributes, relations , constraints on data , semantic information on data and security integrity information

4. The external or view level

The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. The external level interacts directly with the user (end users or application programs). It changes the data coming from the conceptual level to a format and view that is familiar to the users.

- For example, one user may view dates in the form (day, month, year), while another may view dates as (year, month, day).
- Consists of a number of different external views of the DB.
- The user's view of the database
- Describes part of the DB for particular group of users.
- Provides a powerful and flexible security mechanism by hiding parts of the DB from certain users. The user is not aware of the existence of any attributes that are missing from the view.
- It permits users to access data in a way that is customized to their needs, so that the same data can be seen by different users in different ways, at the same time

3 - DATABASE MODELS

1. Introduction

A database model defines the logical design of data. The model also describes the relationships between different parts of the data. In the history of database design, three models have been in use:

- ☐ the hierarchical model
- ☐ the network model
- ☐ the relational model

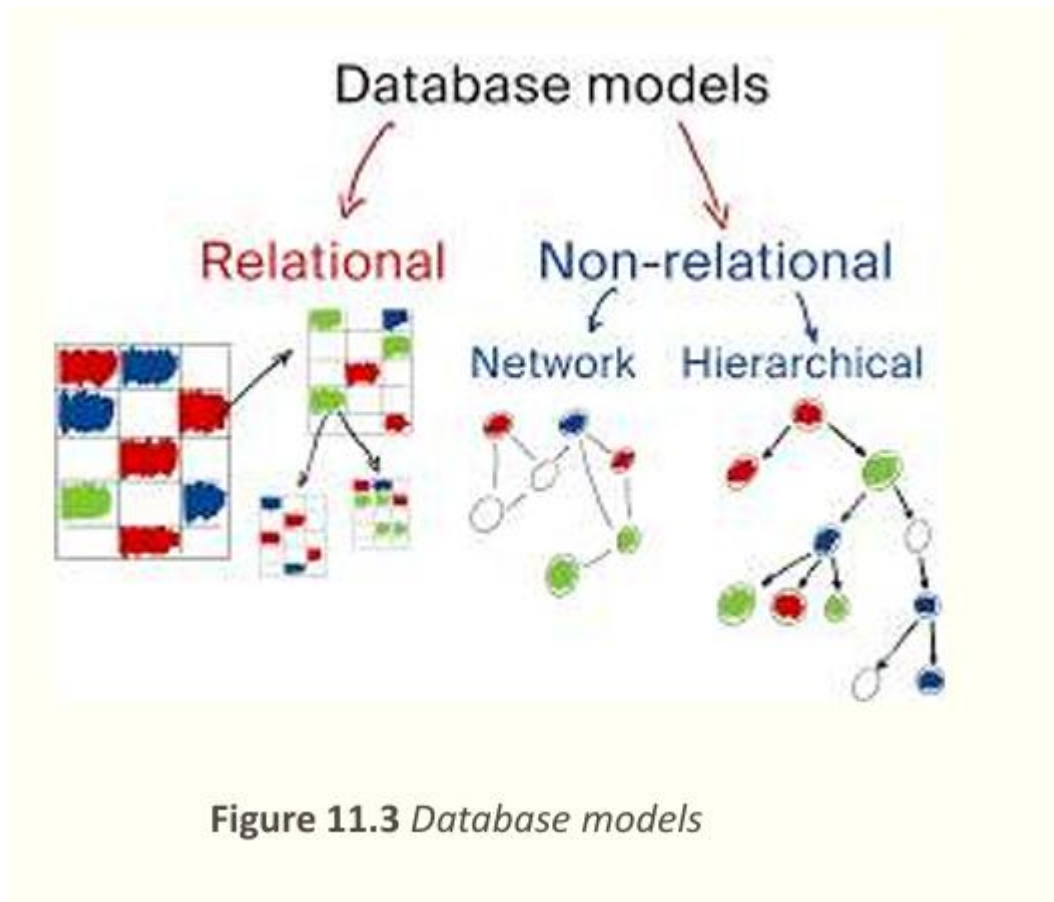
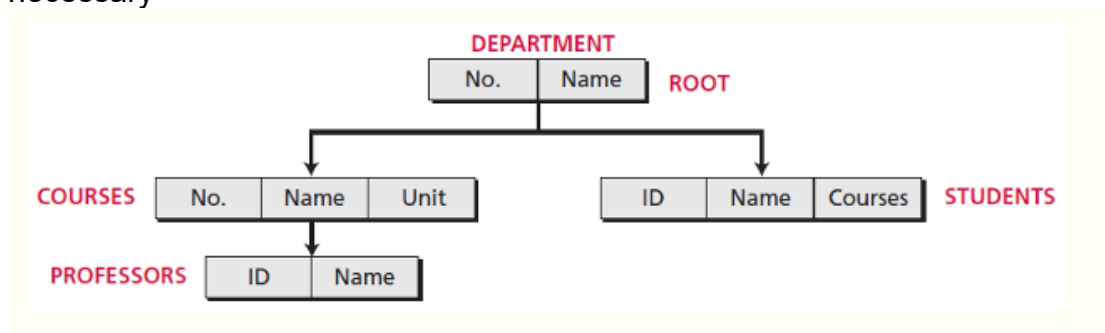


Figure 11.3 *Database models*

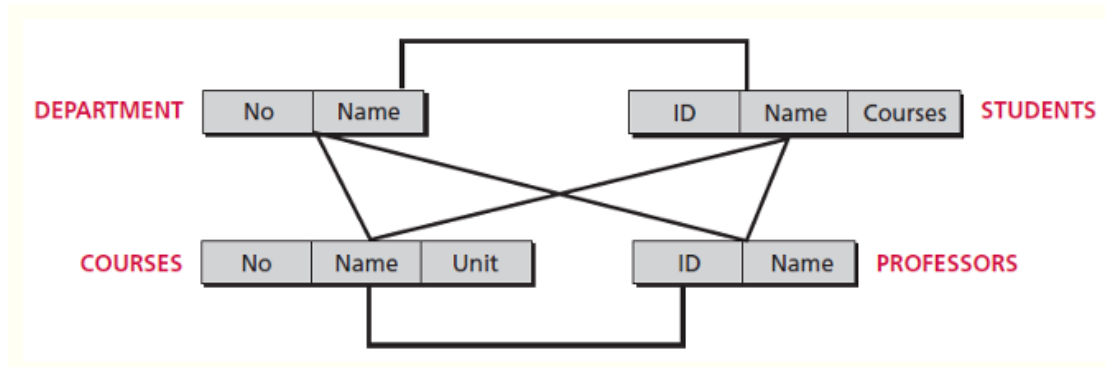
2. The hierarchical model

- In the hierarchical model, data is organized as an inverted tree. Each entity has only one parent but can have several children. At the top of the hierarchy, there is one entity, which is called the root. Figure 14.3 shows a logical view of an example of the hierarchical model.
- As the hierarchical model is obsolete, no further discussion of this model is necessary



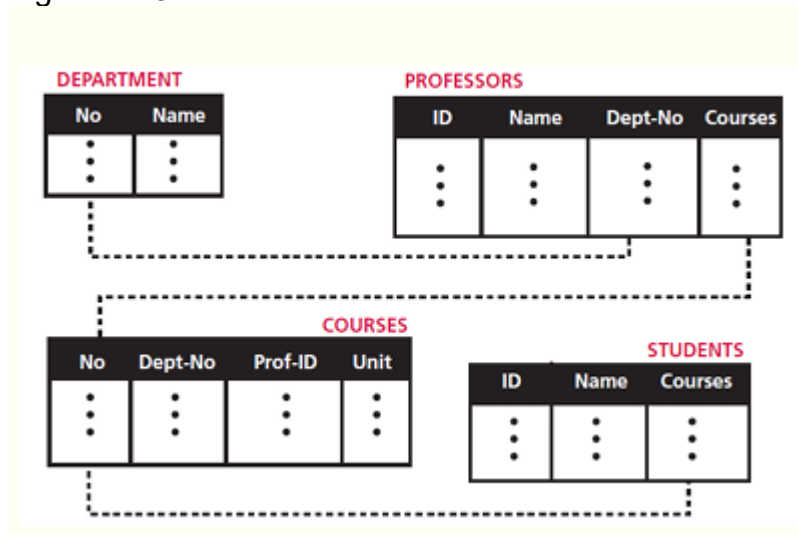
3. The network model

- In the network model, the entities are organized in a graph, in which some entities can be accessed through several paths (Figure 14.4). There is no hierarchy. This model is also obsolete and needs no further discussion.



4. The relational model

▪ In the relational model, data is organized in **two-dimensional tables called relations**. There is no hierarchical or network structure imposed on the data. The tables or relations are, however, related to each other, as we see in Figure 14.5.



3- THE RELATIONSHIP DATABASE MODEL

1. Relation

▪ Relation, in appearance, is a two-dimensional table. The RDBMS organizes the data so that its external view is a set of relations or tables. This does not mean that data are stored as tables: the physical storage of the data is independent of the way in which the data is logically organized. Figure 14.6 shows an example of a relation.

• A relation in an RDBMS has the following features:

□ **Name.** Each relation in a relational database should have a name that is unique among other relations.

□ **Attributes.** Each column in a relation is called an attribute. The attributes are the column headings in the table in Figure 14.6. Each attribute gives meaning to the data

- Tuples. Each row in a relation is called a tuple. A tuple defines a collection of attribute values. The total number of rows in a relation is called the cardinality of the relation

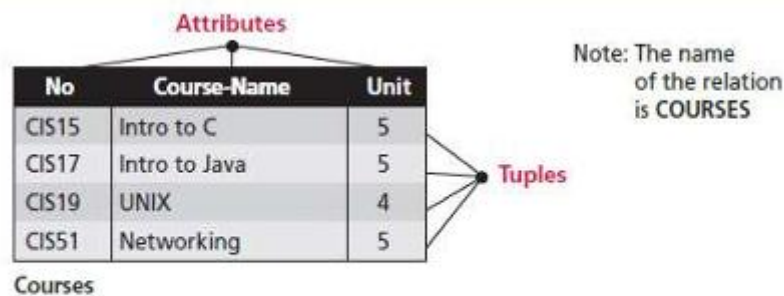


Figure 11.7 *An example of a relation*

2. Operations on relations

- In a relational database we can define several operations to create new relations based on existing ones. We define nine operations in this section:
Instead of discussing these operations in the abstract, we describe each operation as defined in the database query language SQL (Structured Query Language).
- Structured Query Language (SQL) is the language standardized by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) for use on relational databases

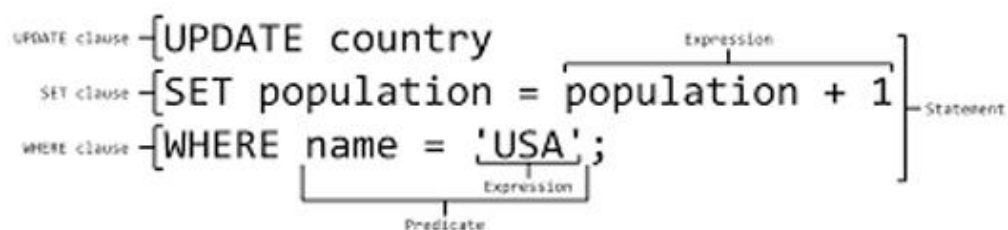
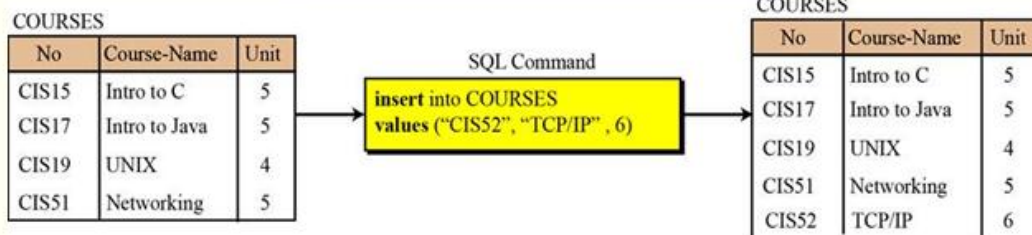


Figure 11.8 *An example of a SQL sentence*

3. Insert

- The insert operation is a unary operation—that is, it is applied to a single relation. The operation inserts a new tuple into the relation. The insert operation uses the following format:

insert into RELATION-NAME
values (... , ... , ...)



4. Delete

- The delete operation is also a unary operation. The operation deletes a tuple defined by a criterion from the relation. The delete operation uses the following format:

delete from RELATION-NAME
where criteria



Figure 11.10 *An example of a Delete sentence*

5. Update

- The update operation is also a unary operation that is applied to a single relation. The operation changes the value of some attributes of a tuple. The update operation uses the following format:

following format:

```
update RELATION-NAME
set attribute1 = value1, attribute2 = value2, ...
where criteria
```



Figure 11.11 An example of a Update sentence

6. Select

- The select operation is a unary operation. The tuples (rows) in the resulting relation are a subset of the tuples in the original relation.

```
select *
from RELATION-NAME
where criteria
```

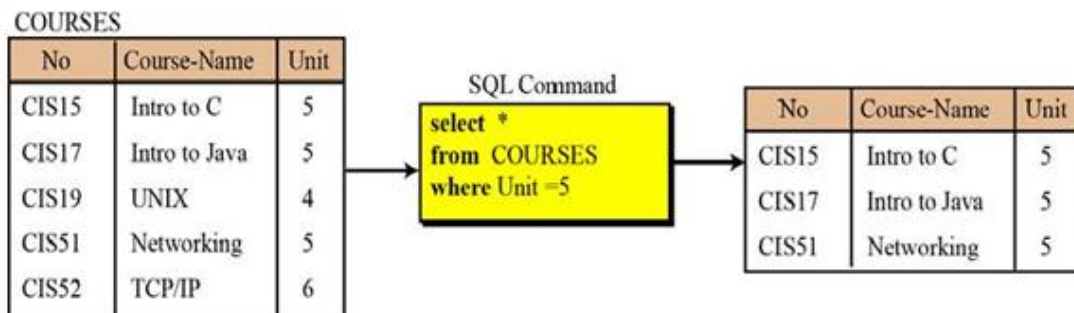


Figure 11.12 An example of a Select sentence

7. Join

- The join operation is a binary operation that combines two relations on common attributes.

```

select attribute-list
from RELATION1, RELATION2
where criteria

```

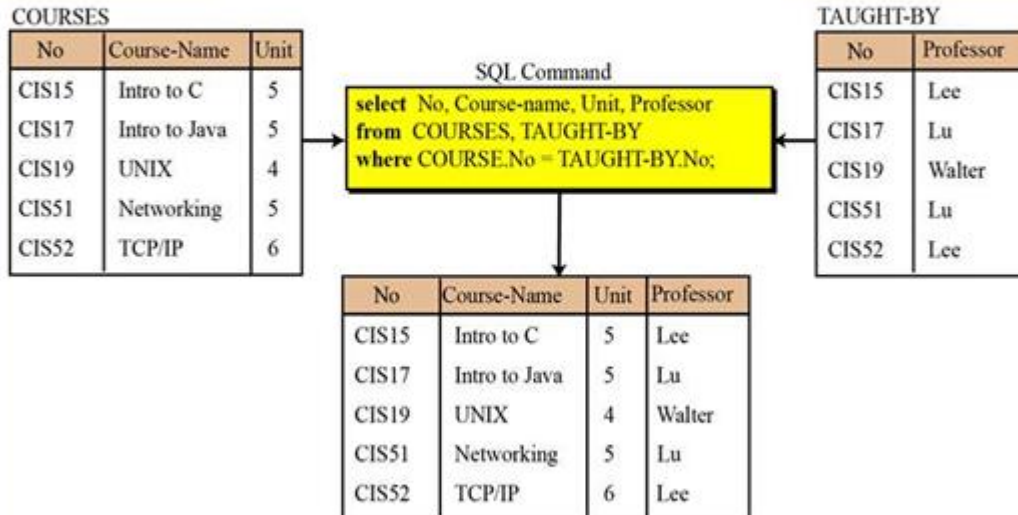


Figure 11.13 An example of a Join sentence

8. Union

- The union operation takes two relations with the same set of attributes.

```

select *
from RELATION1
union
select *
from RELATION2

```

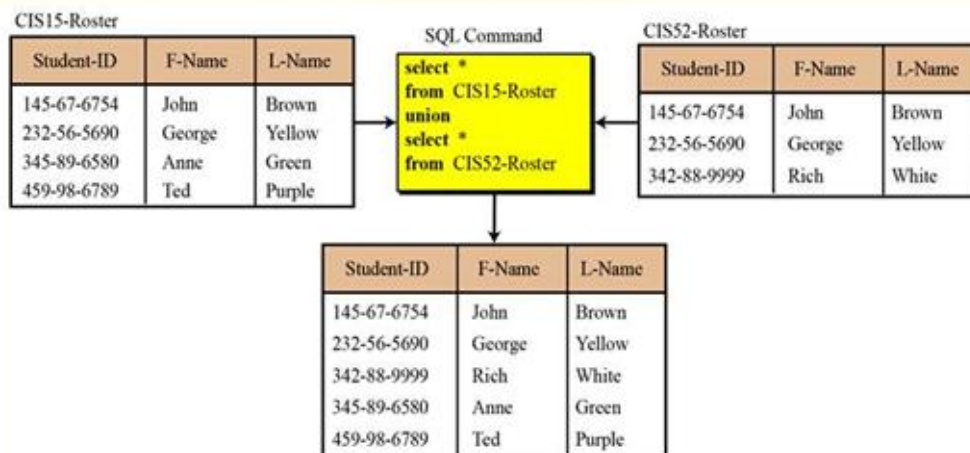


Figure 11.14 An example of a Union sentence

9. Project

Delete a column (attribute) (unary)

10. intersection

You want to create a relation called New that contains tuples that belong to both relation A and relation B

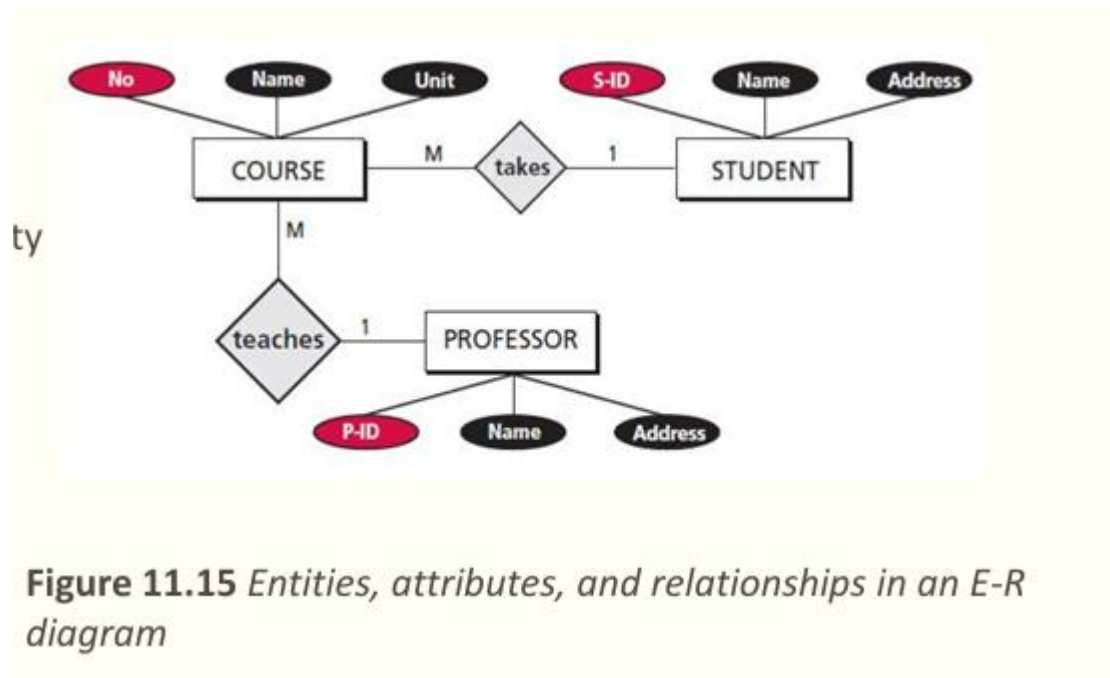
5 - DATABASE DESIGN

1. Introduction

- The design of any database is a lengthy and involved task that can only be done through a step-by-step process.
- ☐ The first step normally involves a lot of interviewing of potential users of the database, for example in a university, to collect the information needed to be stored and the access requirements of each department.
- ☐ The second step is to build an entity–relation model (ERM) that defines the entities for which some information must be maintained, the attributes of these entities, and the relationship between these entities.

2. Entity–relation model (ERM)

- In this step, the database designer creates an entity–relationship (E-R) diagram to show the entities for which information needs to be stored and the relationship between those entities. E-R diagrams uses several geometric shapes,
- ☐ Rectangles represent entity sets
- ☐ Ellipses represent attributes
- ☐ Diamonds represent relationship sets
- ☐ Lines link attributes to entity sets and link entity sets to relationship sets



3. From E-R diagrams to relations

- Relations for entity sets
- For each entity set in the E-R diagram, we create a relation (table) in which there are n columns related to the n attributes defined for that set. Example 14.2 We can have three relations (tables), one for each entity set defined in Figure 14.16, as shown in Figure 14.17

COURSE			STUDENT			PROFESSOR		
No	Name	Unit	S-ID	Name	Address	P-ID	Name	Address
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 11.16 Relations for entity set in Figure 14.16

- Relations for relationship sets
- For each relationship set in the E-R diagram, we create a relation (table). This relation has one column for the key of each entity set involved in this relationship and also one column for each attribute of the relationship itself if the relationship has attributes (not in our case). Example 11.3 There are two relationship sets in Figure 14.16, teaches and takes, each connected to two entity sets. The relations for these relationship sets are added to the previous relations for the entity set and shown in Figure 14.18

COURSE			STUDENT			PROFESSOR		
No	Name	Unit	S-ID	Name	Address	P-ID	Name	Address
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

TEACHES		TAKES	
P-ID	No.	S-ID	No.
⋮	⋮	⋮	⋮

Figure 11.17 Relations for E-R diagram in Figure 14.16

4. Normalization

- Normalization is the process by which a given set of relations are transformed to a new set of relations with a more solid structure.
- Normalization is needed to allow any relation in the database to be represented, to allow a languages like SQL to **use powerful retrieval operations composed of atomic operations**("sử dụng các thao tác truy xuất mạnh mẽ được cấu thành từ các thao tác nguyên tử."), to **remove anomalies**(Bất bình thường) in insertion, deletion, and updating, and **reduce the need for restructuring the database as new data type are added.**
 - The normalization process defines a set of hierarchical normal forms (NFs). Several normal forms have been proposed, including 1NF, 2NF, 3NF, BCNF (Boyce–Codd Normal Form), 4NF, PJNF (Projection/Joint Normal Form), 5NF, and so on

5. First normal form (1NF)

- When we transform entities or relationships into tabular relations, there may be some relations in which there are more values in the intersection of a row or column.
- For example, in our set of relations in Figure 14.18, there are two relations, teaches and takes, that are not in first normal form. A professor can teach more than one course, and a student can take more than one course. These two relations can be normalized by repeating the rows in which this problem exists

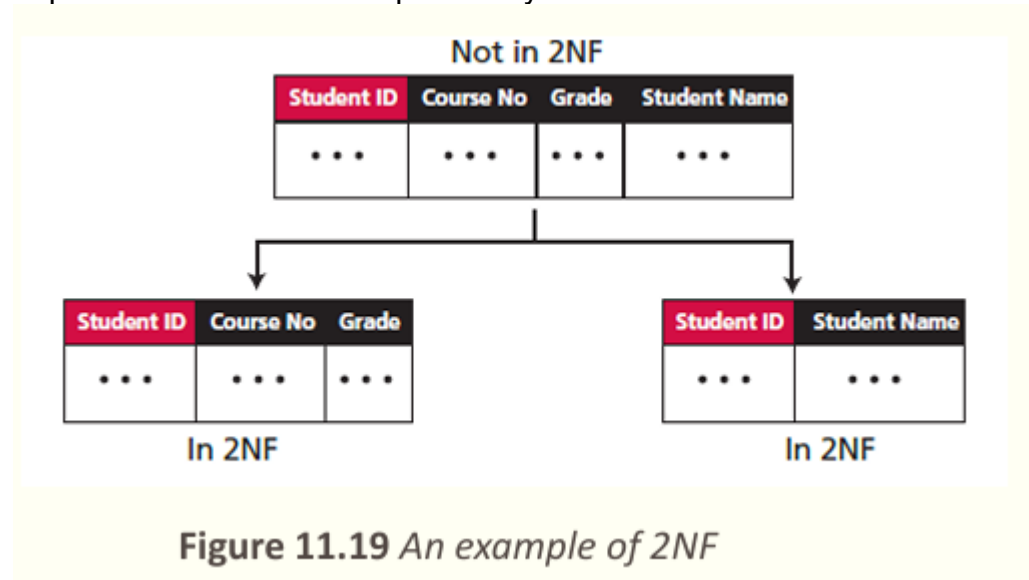
TEACHES			TEACHES		
ID	No.		ID	No.	
⋮	⋮	Three values in one intersection	⋮	⋮	Only one value in each intersection
8256	CIS15		8256	CIS15	
	CIS18		8256	CIS18	
	CIS21		8256	CIS21	
⋮	⋮		⋮	⋮	

a. Not in 1NF b. In 1NF

Figure 11.18 An example of 1NF

6. Second normal form (2NF)

▪ In each relation we need to have a key (called a primary key) on which all other attributes (column values) needs to depend. ▪ For example, if the ID of a student is given, it should be possible to find the student's name. However, it may happen that when relations are established based on the E-R diagram, we may have some composite keys (a combination of two or more keys). ▪ In this case, a relation is in second normal form if every non-key attribute depends on the whole composite key



1. Insert (Thêm bản ghi)

Mục đích: Thêm dữ liệu mới vào bảng.

Ví dụ: Thêm sinh viên tên "Lan" vào bảng SinhVien.

2. Delete (Xóa bản ghi)

Mục đích: Xóa bản ghi thỏa điều kiện.

Ví dụ: Xóa sinh viên có MSSV = "SV001" khỏi bảng SinhVien.

3. Update (Cập nhật bản ghi)

Mục đích: Sửa giá trị của bản ghi.

Ví dụ: Đổi điểm của sinh viên "Lan" từ 7.0 thành 8.5.

4. Select (Chọn bản ghi)

Mục đích: Lọc bản ghi theo điều kiện.

Ví dụ: Tìm sinh viên có điểm ≥ 8.0 trong bảng SinhVien.

5. Project (Phép chiếu)

Mục đích: Chọn một số cột cụ thể, bỏ qua cột không cần.

Ví dụ: Chỉ lấy cột HoTen và Diem từ bảng SinhVien.

6. Join (Phép kết nối)

Mục đích: Ghép hai bảng dựa trên liên kết.

Ví dụ: Kết hợp bảng SinhVien và MonHoc để biết sinh viên học môn nào.

7. Union (Phép hợp)

Mục đích: Gộp kết quả từ hai truy vấn (loại trùng lặp).

Ví dụ: Gộp danh sách sinh viên lớp "A" và lớp "B" thành một danh sách duy nhất.

8. Intersection (Phép giao)

Mục đích: Lấy phần chung của hai truy vấn.

Ví dụ: Tìm sinh viên vừa học "Toán" vừa học "Lý".

9. Difference (Phép hiệu)

Mục đích: Lấy bản ghi có trong bảng này nhưng không có trong bảng kia.

Ví dụ: Tìm sinh viên học "Toán" nhưng không học "Lý".