# History of Data Models

- Hierarchical Model
- Network Model
- Relational Model
- Object-oriented Data Models
- Object-Relational Models

Slide 2- 1



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 and the relational model.

14.2

## Hierarchical database 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.

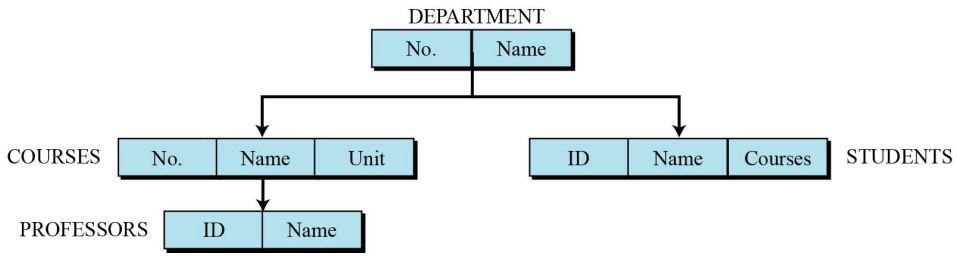



**Figure 14.3** An example of the hierarchical model representing a university

14.3

## Network database model

In the network model, the entities are organized in a graph, in which one entity can be accessed through several paths (Figure 14.4).

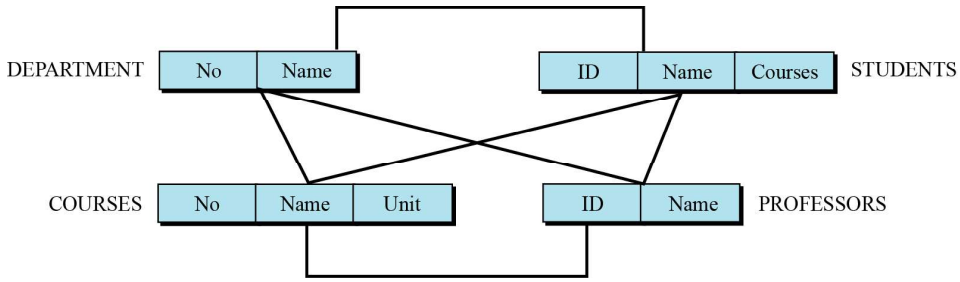



**Figure 14.4** An example of the network model representing a university

14.4

## Relational database model

In the relational model, data is organized in two-dimensional tables called relations. The tables or relations are, however, related to each other.

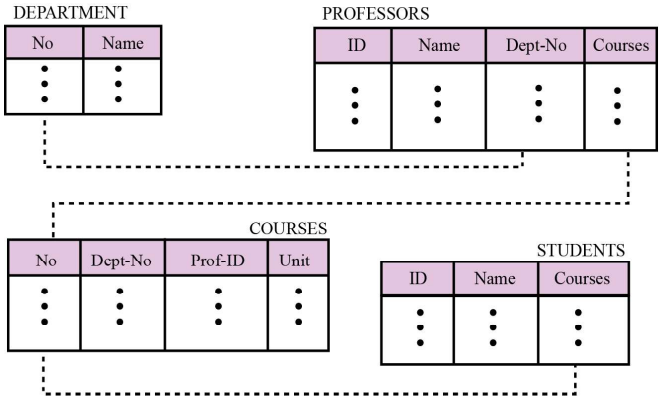



**Figure 14.5** An example of the relational model representing a university



# Outline

- Relational Model
  - History
  - Concepts
  - Constraints



# Relational Model History

- Introduced by Ted Codd in 1970
- Ted Codd was an **IBM** Research Fellow
- Laid the foundation for database theory
- Many database concepts & products based on this model



# Why is the relational model so popular?

- supported by a mathematical model
- relations (tables) are a good tool for communicating information to users and developers
- efficient implementations exist for the storing of relational information in the form of Relational DBMSs (RDBMSs)

# What is a Relation?

- A Relation is a 2-dimensional table of values (rows and columns)
- each row, or **tuple**, is a collection of related facts
- the degree of the relation is the number of attributes in the relation
- each column represents an **attribute**
- each row is an **instance** of the relation



# What is a Relation (cont'd)?

- So, a relation is a big table of facts.
  - Each column contains the attribute data with the data type
  - Each row describes a real-world instance of the relation
- A Relational database contains one or more relations (or tables).



# Schema vs. Instance

- the name of the relation and the set of attributes is called the **schema** (or the **intension**)
- the current values in the relation represent an **instance** (or **extension**) of the data



# More formally……

- A *domain* **D** is a set of atomic values
  - local phone number – The set of 7-digit numbers
  - names – The set of names of persons
  - date of birth – Possible dates of birth for people
- A *relation schema* $\mathbf{R(A_1, A_2, \ldots, A_n)}$ is a:
  - relation name (**R**)
  - list of attributes ($\mathbf{A_1, A_2, \ldots, A_n}$)
  - each attribute $\mathbf{A_i}$ is the name of a role played by some domain **D** in the relation schema **R**

## 14-8 OTHER DATABASE MODELS

The relational database is not the only database model in use today. Two other common models are distributed databases and object-oriented databases. We briefly discuss these here.

14.13

### Replicated distributed databases

In a replicated distributed database, each site holds an exact replica of another site. Any modification to data stored in one site is repeated exactly at every site. The reason for having such a database is security. If the system at one site fails, users at that site can access data from another site.

14.15

## Distributed databases

The distributed database model is not a new model, but is based on the relational model. However, the data is stored on several computers that communicate through the Internet or a private wide area network. Each computer (or site) maintains either part of the database or the whole database.

### Fragmented distributed databases

In a fragmented distributed database, data is localized—locally used data is stored at the corresponding site. However, this does not mean that a site cannot access data stored in another site. Access is mostly local, but occasionally global.

14.14

## Object-oriented databases

An object-oriented database tries to keep the advantages of the relational model and at the same time allows applications to access structured data. In an object-oriented database, objects and their relations are defined. In addition, each object can have attributes that can be expressed as fields.

### XML

The query language normally used for objected-oriented databases is XML (Extensible Markup Language). XML was originally designed to add markup information to text documents, but it has also found its application as a query language in databases. XML can represent data with nested structures.

14.16

## Extended Relational Data Model (ERDM)

- Semantic data model
  - Developed in response to increasing complexity of applications
  - Based heavily on relational model
    - Relational DB response to OODM
  - Primarily geared to business applications
    - Typically scientific or engineering apps
- Object/relational database management system (O/RDBMS)
  - DBMS based on the ERDM



In the relational database management system (RDBMS), the data is represented as a set of *relations*.



## Relations

A relation appears as 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 is stored as tables: the physical storage of the data is independent of the way in which the data is logically organized.

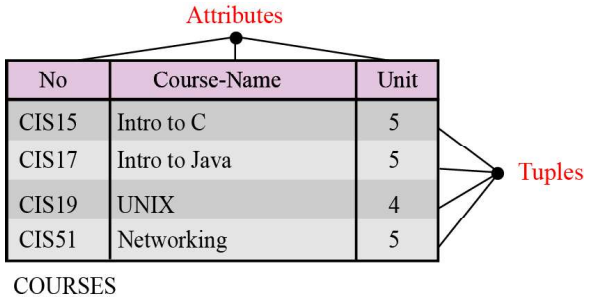



**Figure 14.6** **An example of a relation**



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.
- 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. Note that the cardinality of a relation changes when tuples are added or deleted. This makes the database dynamic.

In a relational database we can define several operations to create new relations based on existing ones. We define nine operations in this section: insert, delete, update, select, project, join, union, intersection and difference. Instead of discussing these operations in the abstract, it describes each **operation** as **defined** in the database query language named SQL (Structured Query Language).



## 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. It is a *declarative* rather than *procedural* language, which means that users declare what they want without having to write a step-by-step procedure. The SQL language was first implemented by the Oracle Corporation in 1979, with various versions of SQL being released since then.



## 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    (..., ..., ...)
```

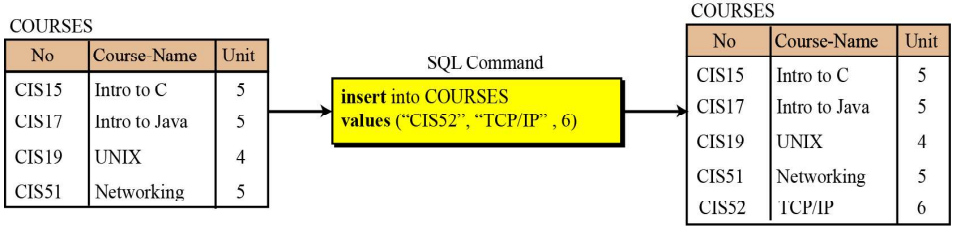





**Figure 14.7** An example of an insert operation

## Delete

The delete operation is also a unary operation that is applied to a single relation. 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 14.8** An example of a delete operation

## 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:

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

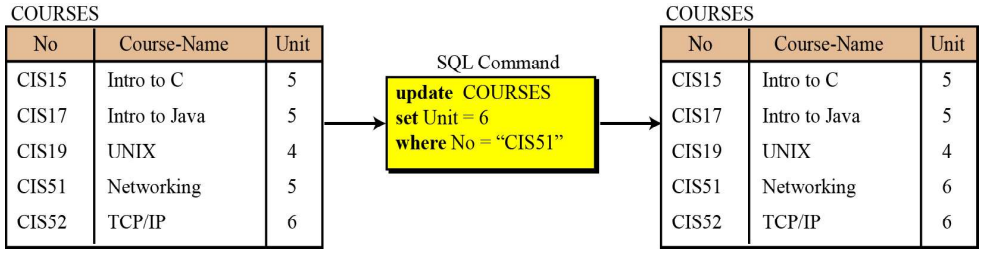



**Figure 14.9** An example of an update operation

## Select

The select operation is a unary operation that is applied to a single relation. The resulting tuples (rows) are a subset of the tuples (rows) in the original relation. The select operation uses the following format:

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

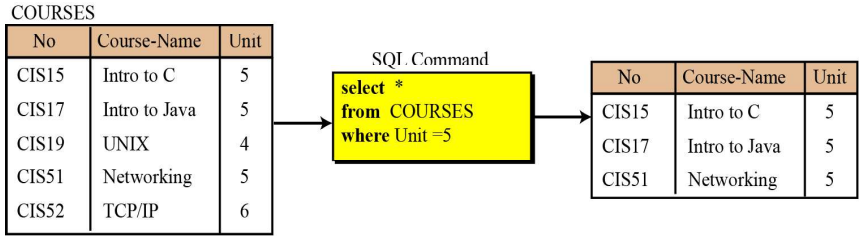



**Figure 14.10** An example of an select operation

## Project

The project operation is also a unary operation and creates another relation temporarily. The attributes (columns) in the resulting relation are a subset of the attributes in the original relation.

```
select    attribute-list
from RELATION-NAME
```

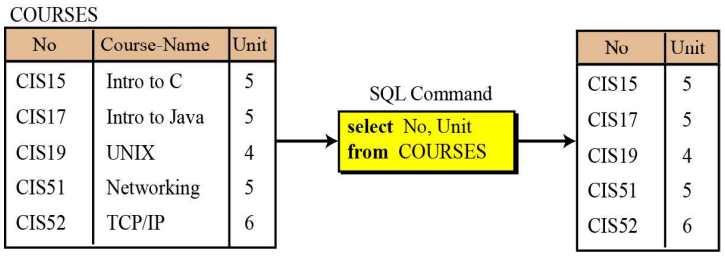



**Figure 14.11** An example of a project operation

## Join

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

```
select    attribute-list
from RELATION1, RELATION2
where criteria
```

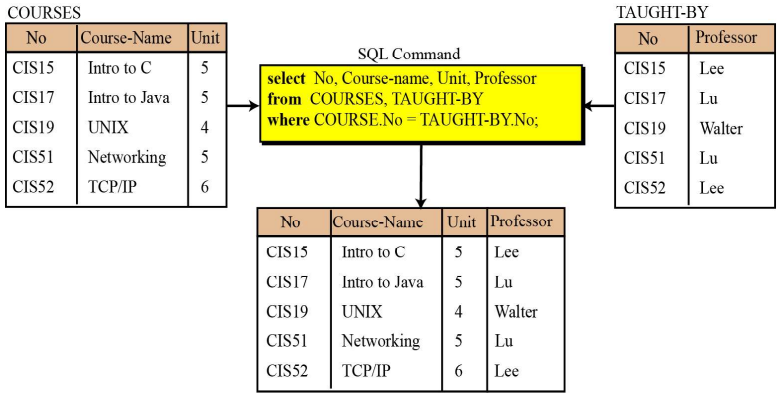
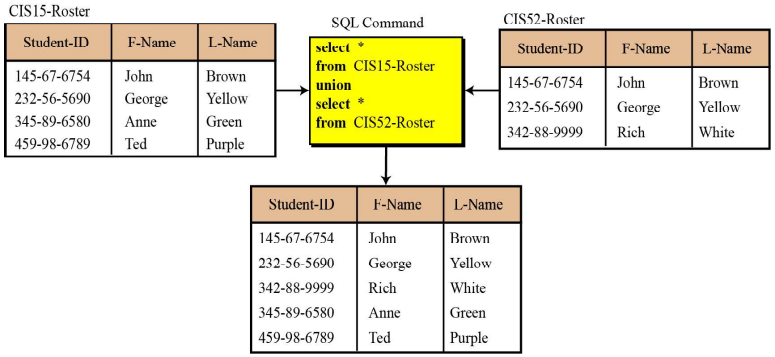



**Figure 14.12** An example of a join operation

## Union

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

```
select *
from RELATION1
union
select *
from RELATION2
```



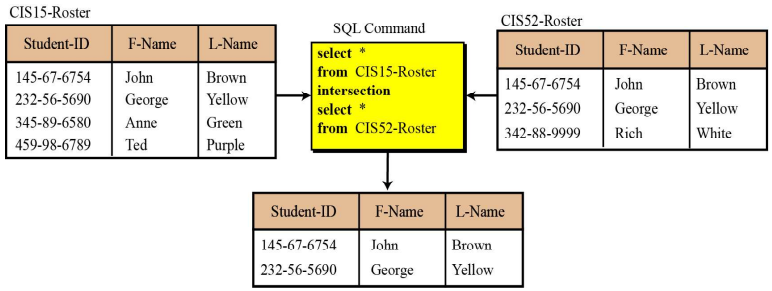

14.29

**Figure 14.13** An example of a union operation

## Intersection

The intersection operation takes two relations and creates a new relation, which is the common in the two relations.

```
select *
from RELATION1
intersection
select *
from RELATION2
```



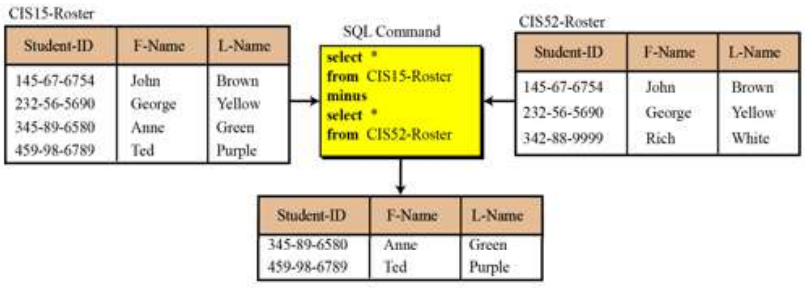

14.30

**Figure 14.14** An example of an intersection operation

## Difference

The difference operation is applied to two relations with the same attributes. The resulting tuples are those that are in the first relation but not in the second relation.

```
select *
from RELATION1
minus
select *
from RELATION2
```




14.31

**Figure 14.15** An example of a difference operation