DATA MODELS

14-1 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 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 and so on. Today, however, all of these flat files can be combined in a single entity; the database for the whole university.

Although it is difficult to give a universally agreed definition of a database, we use the following common definition:

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

Comparing the flat-file system, we can mention several advantages for a database system.

Less redundancy

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

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.

Efficiency

A database is usually more efficient that a flat file system, because a piece of information is stored in fewer locations.

Data integrity

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

It is easier to maintain the confidentiality of the information if the storage of data is centralized in one location.

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).

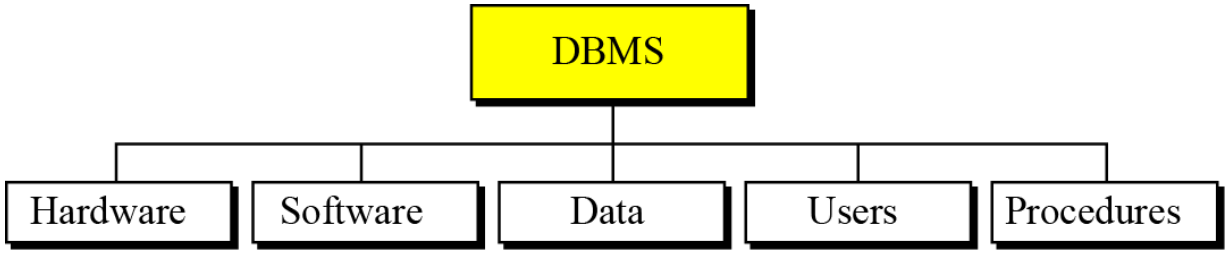


Figure 14.1 DBMS components

Hardware

The hardware is the physical computer system that allows access to data.

Software

The software is the actual program that allows users to access, maintain and update data. In addition, the software controls which user can access which parts of the data in the database.

Confidentiality

The 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

In a DBMS, the term users has a broad meaning. We can divide users into two categories: end users and application programs.

Procedures

The last component of a DBMS is a set of procedures or rules that should be clearly defined and followed by the users of the database.

14-3 DATABASE ARCHITECTURE

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).

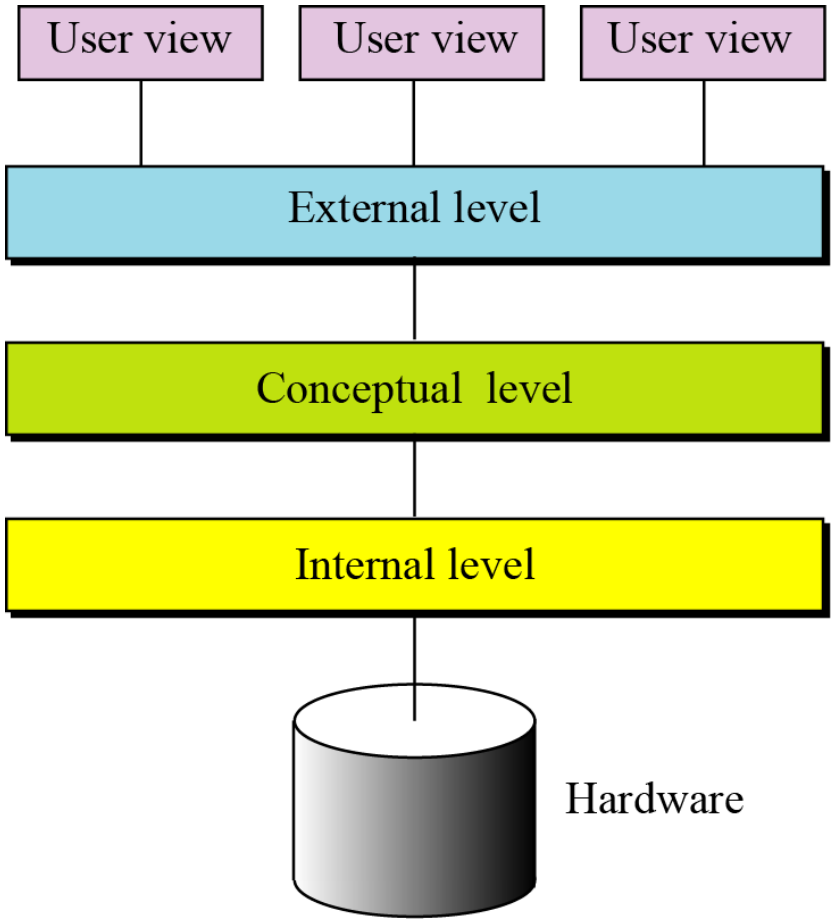


Figure 14.2 Database architecture

Internal level

The 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. In other words, the internal level interacts directly with the hardware.

Conceptual level

The 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. The DBMS changes the internal view of data to the external view that users need to see. The conceptual level is an intermediary and frees users from dealing with the internal level.

External level

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.

DATA MODELS

Data model tells how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in DBMS. Data models define how data is connected to each other and how it will be processed and stored inside the system.

14-4 DATABASE MODELS

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.

Hierarchical Model

- In a hierarchical model data is organized into a tree-like structure implying a single parent for each record. A sort field keeps sibling records in a particular order. Hierarchical structures were widely used in the early mainframe database management systems, such as the Information Management System (IMS) by IBM,

- This structure allows one one-to-many relationship between two types of data. This structure is very efficient to describe many relationships in the real world; recipes, table of contents, ordering of paragraphs/verses, any nested and sorted information.
- **This hierarchy is used as the physical order of records in storage. Record access is done by navigating through the data structure using pointers combined with sequential accessing. Because of this, the hierarchical structure is inefficient for certain database operations when a full path (as opposed to upward link and sort field) is not also included for each record. Such limitations have been compensated for in later IMS versions by additional logical hierarchies imposed on the base physical hierarchy.**

The hierarchical data model

- The hierarchical data model organizes data in a tree structure. There is a hierarchy of parent and child data segments. This structure implies that a record can have repeating information, generally in the child data segments. Data in a series of records, which have a set of field values attached to it. It collects all the instances of a specific record together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows. To create links between these record types, the hierarchical model uses Parent Child Relationships. These are a 1:N mapping between record types. This is done by using trees, like set theory used in the relational model, "borrowed" from maths. For example, an organization might store information about an employee, such as name, employee number, department, salary. The organization might also store information about an employee's children, such as name and date of birth. The employee and children data forms a hierarchy, where the employee data represents the parent segment and the children data represents the child segment. If an employee has three children, then there would be three child segments associated with one employee segment. In a hierarchical database the parent-child relationship is one to many. This restricts a child segment to having only one parent segment. Hierarchical DBMSs were popular from the late 1960s, with the introduction of IBM's Information Management System (IMS) DBMS, through the 1970s.

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.

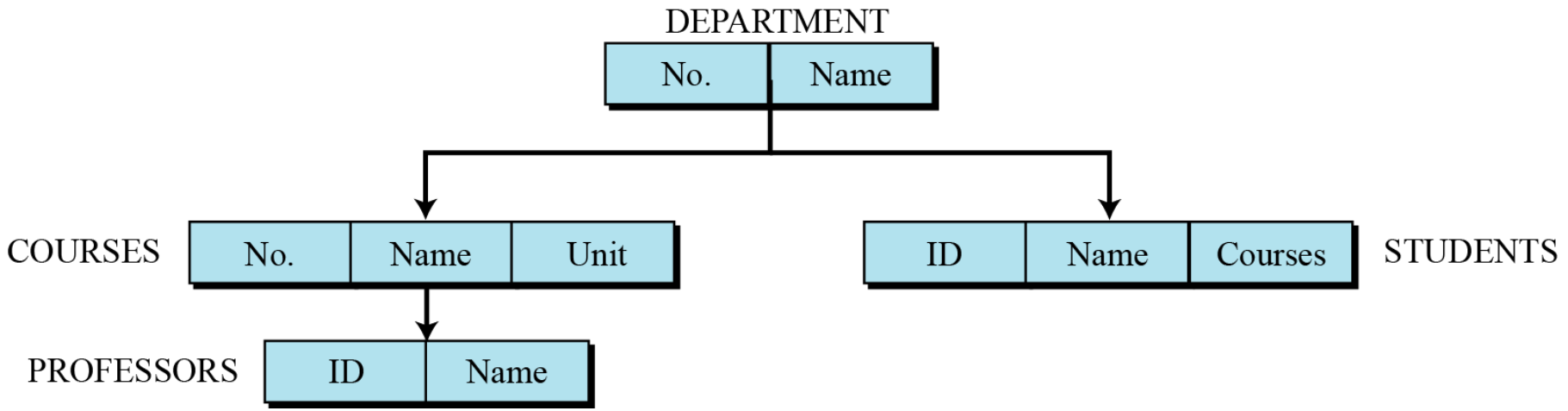
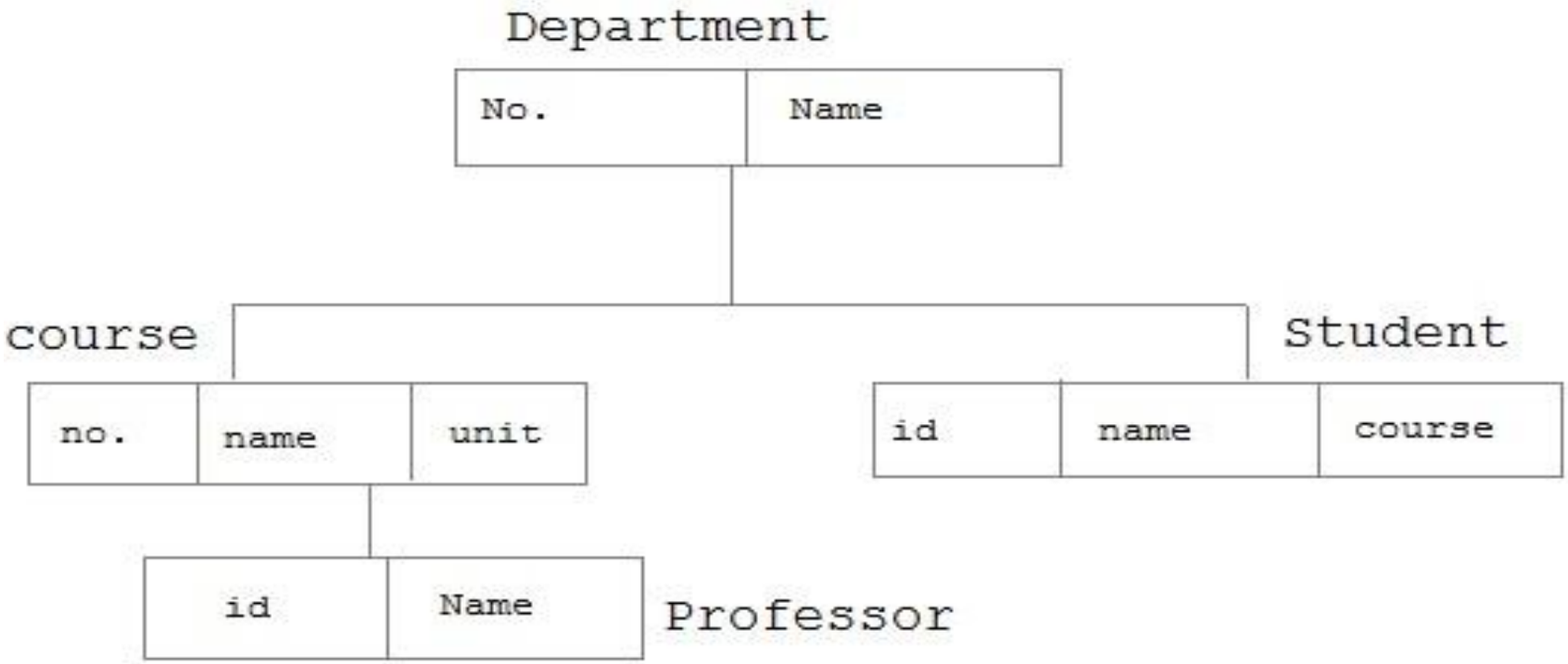


Figure 14.3 An example of the hierarchical model representing a university

Hierarchical Model



Network model

- The network model expands upon the hierarchical structure, allowing many-to-many relationships in a tree-like structure that allows multiple parents.

- Thus all the sets comprise a general directed graph (ownership defines a direction), or *network* construct. Access to records is either sequential (usually in each record type) or by navigation in the circular linked lists.

Network Model

- The popularity of the network data model coincided with the popularity of the hierarchical data model. Some data were more naturally modeled with more than one parent per child. So, the network model permitted the modeling of many-to-many relationships in data. In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model. The basic data modeling construct in the network model is the set construct. A set consists of an owner record type, a set name, and a member record type. A member record type can have that role in more than one set, hence the multiparent concept is supported. An owner record type can also be a member or owner in another set. The data model is a simple network, and link and intersection record types (called junction records by IDMS) may exist, as well as sets between them . Thus, the complete network of relationships is represented by several pairwise sets; in each set some (one) record type is owner (at the tail of the network arrow) and one or more record types are members (at the head of the relationship arrow). Usually, a set defines a 1:M relationship, although 1:1 is permitted. The CODASYL network model is based on mathematical set theory.

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

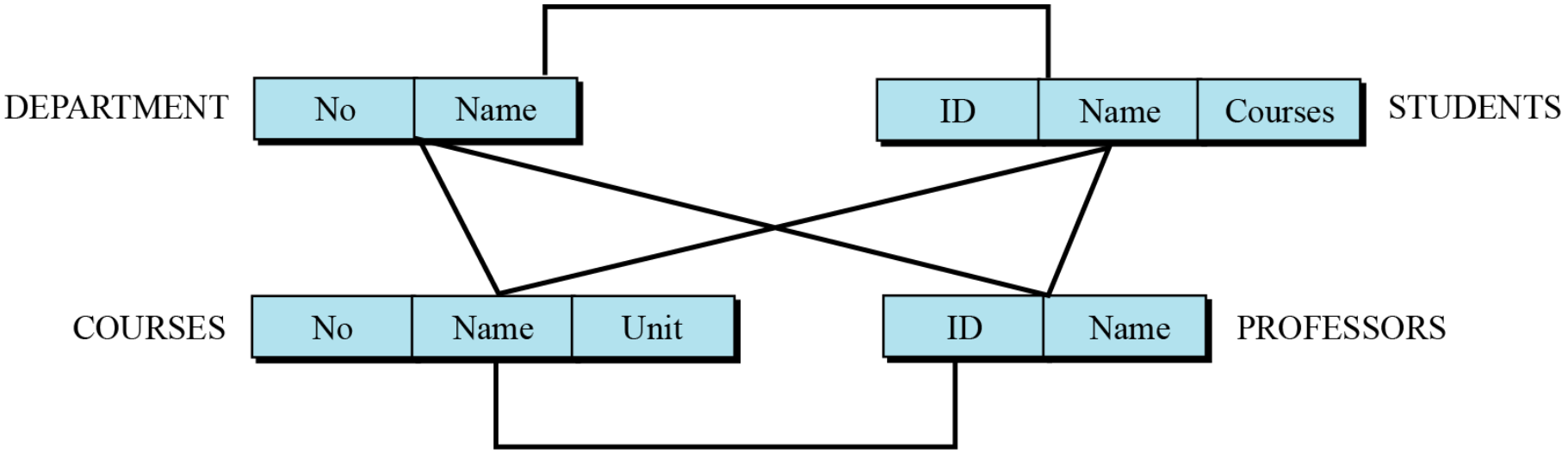
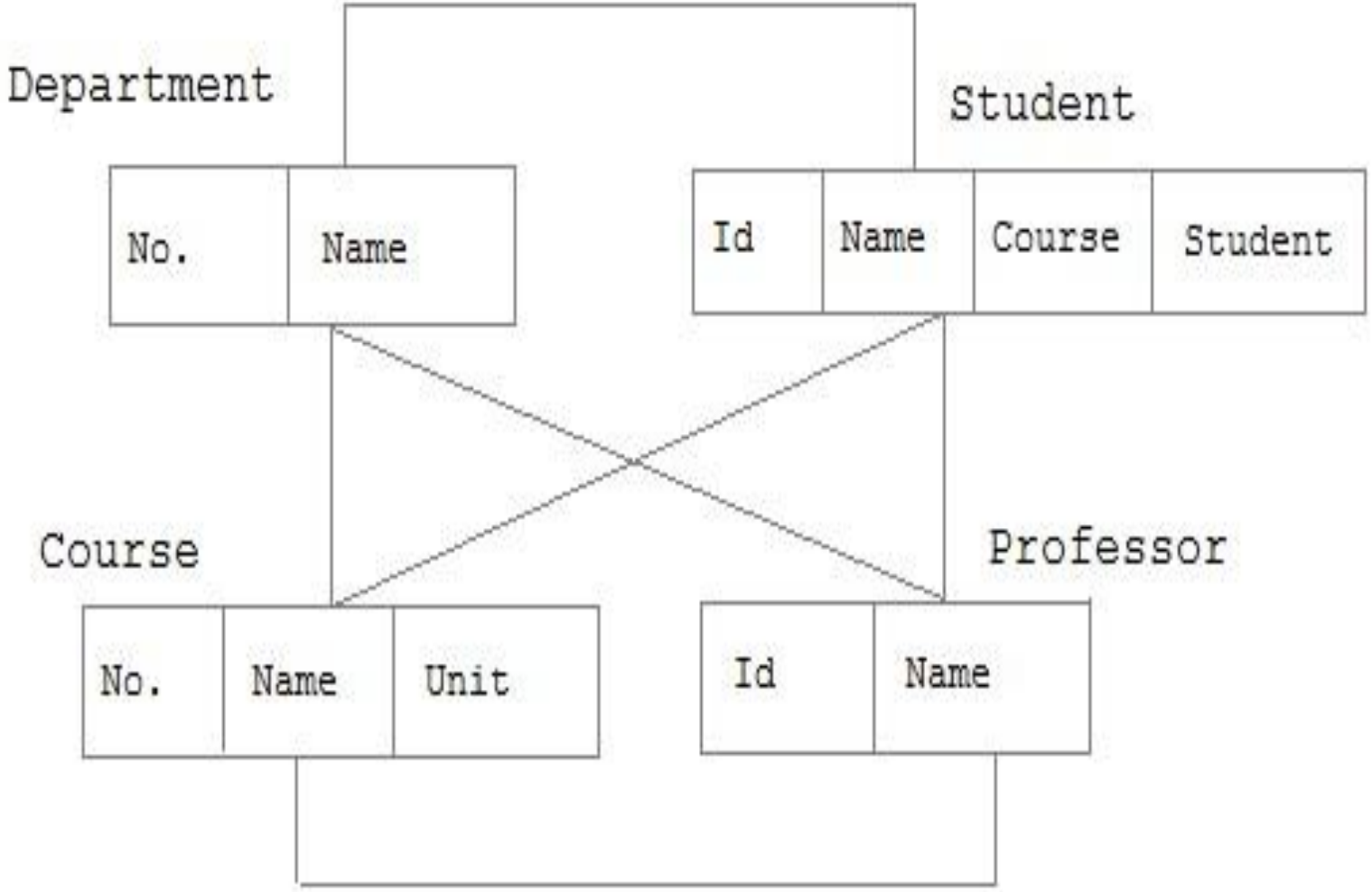


Figure 14.4 An example of the network model representing a university


Department
No.
Name
Student
Id
Name
Course
Student
Course
No.
Name
Unit
Professor
Id
Name

Relational Model

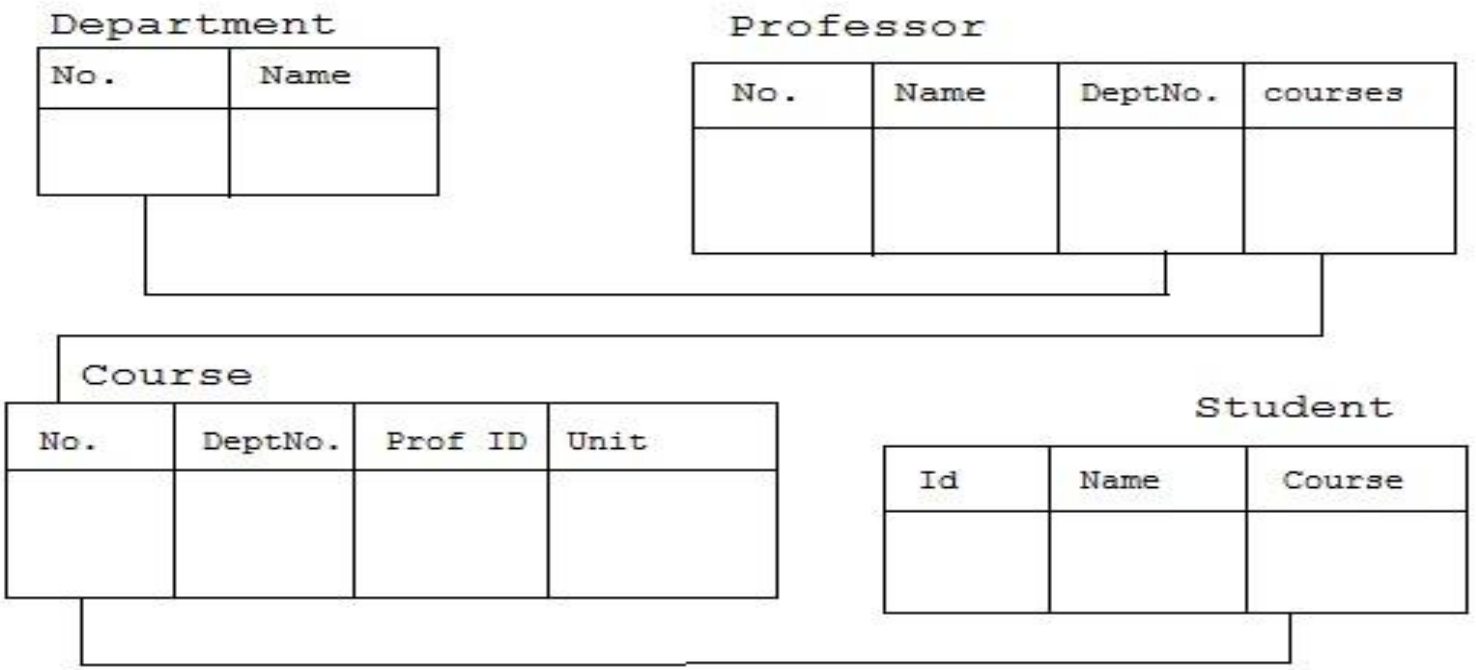
- The most popular data model in DBMS is Relational Model. It is more scientific model then others. This model is based on first-order predicate logic and defines table as an n-ary relation.
- The main highlights of this model are:
- Data is stored in tables called relations.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in relation contains unique value
- Each column in relation contains values from a same domain.
-

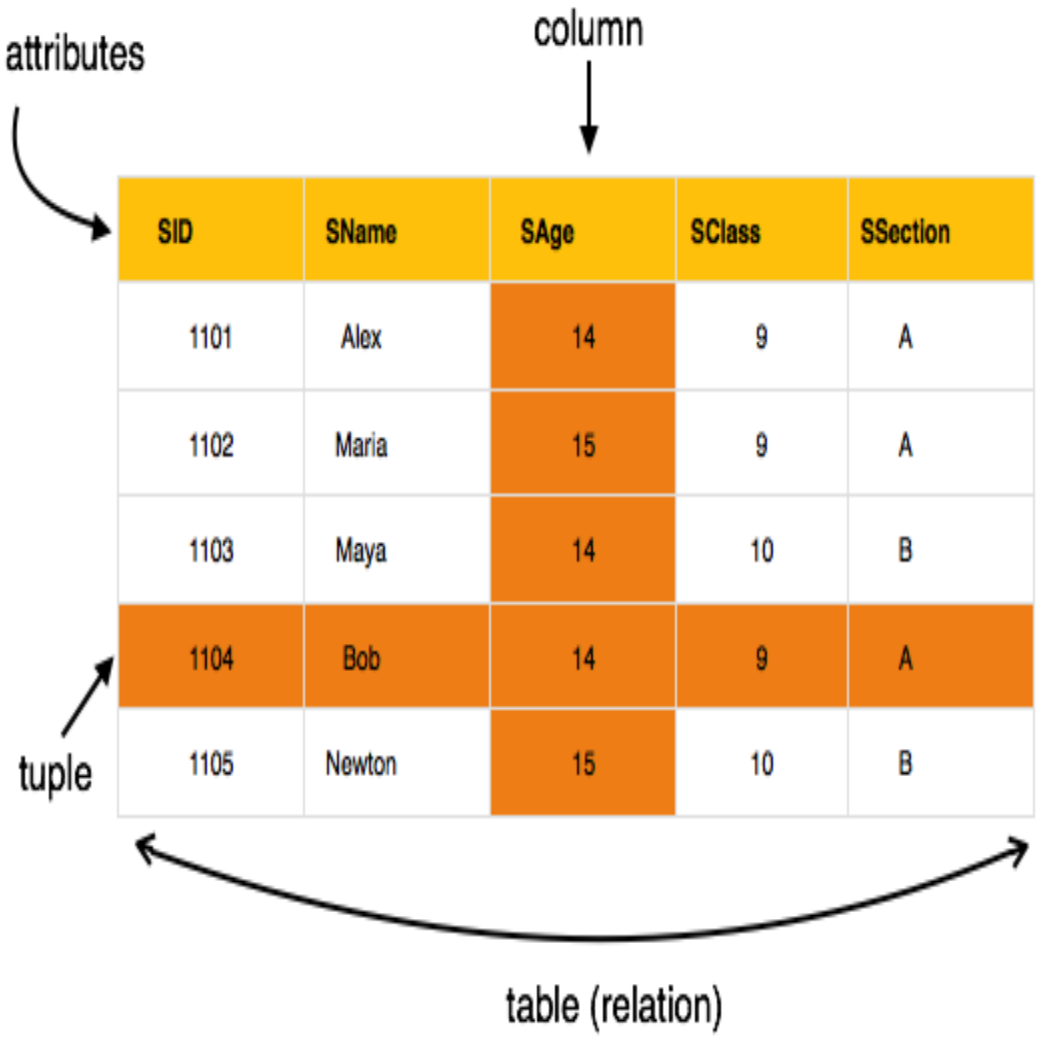
Relational Data Model

- (RDBMS - relational database management system) A database based on the relational model developed by E.F. Codd. A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organised in tables. A table is a collection of records and each record in a table contains the same fields.

  Properties of Relational Tables:Values Are AtomicEach Row is UniqueColumn Values Are of the Same KindThe Sequence of Columns is InsignificantThe Sequence of Rows is InsignificantEach Column Has a Unique Name

Relational Model




attributes
column
SID
SName
SAge
SClass
SSection
1101
Alex
14
9
A
1102
Maria
15
9
A
1103
Maya
14
10
B
1104
Bob
14
9
A
1105
Newton
15
10
B
tuple
table (relation)

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

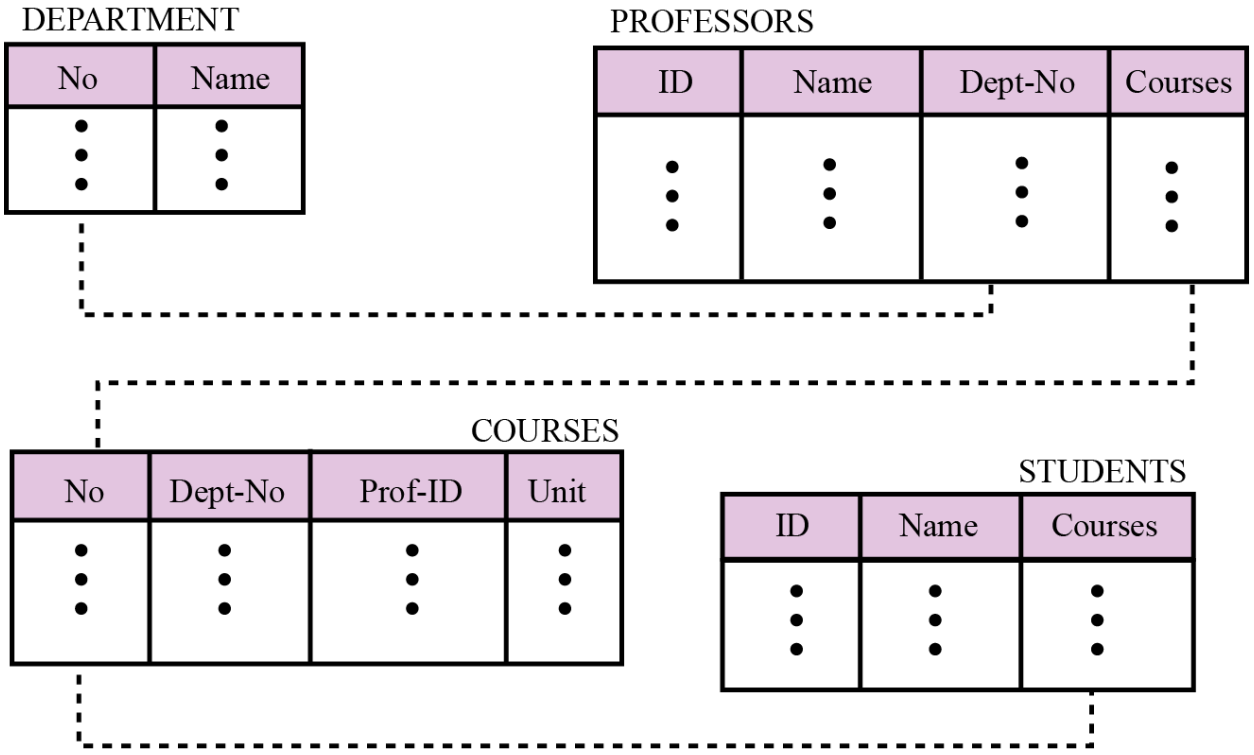


Figure 14.5 An example of the relational model representing a university

14.5 THE RELATIONAL DATABASE MODEL

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

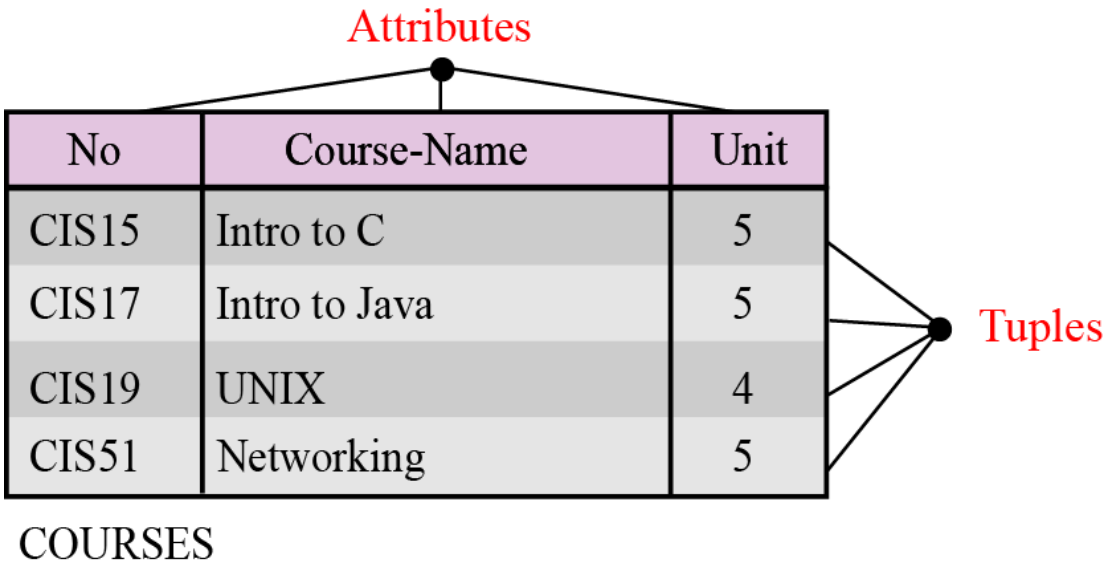


Figure 14.6 An example of a relation

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.

14-6 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: insert, delete, update, select, project, join, union, intersection and difference. Instead of discussing these operations in the abstract, we describe each operation as defined in the database query language SQL (Structured Query Language).

INTRODUCTION TO SQL

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.

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

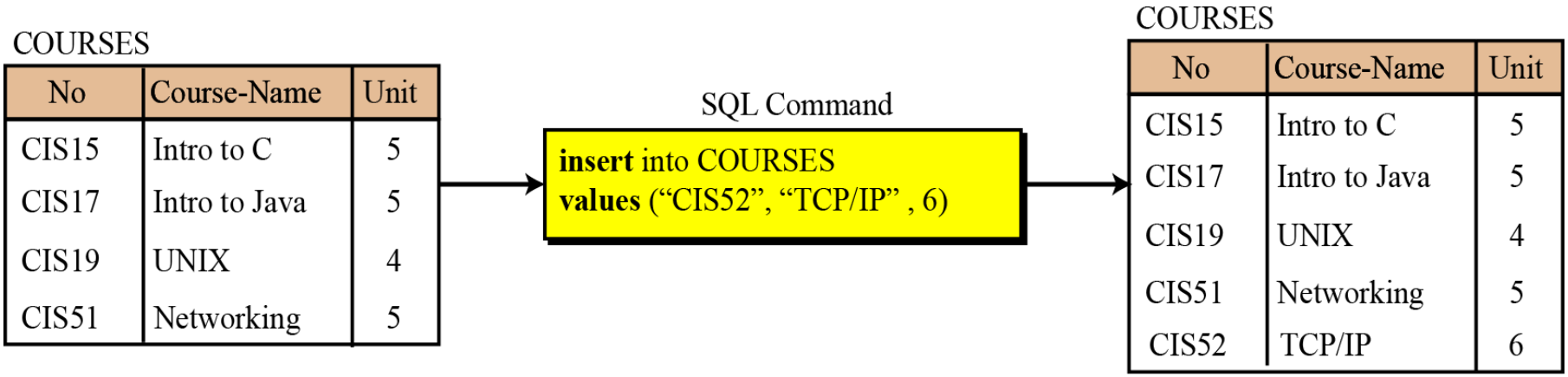


Figure 14.7 An example of an insert operation

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
```

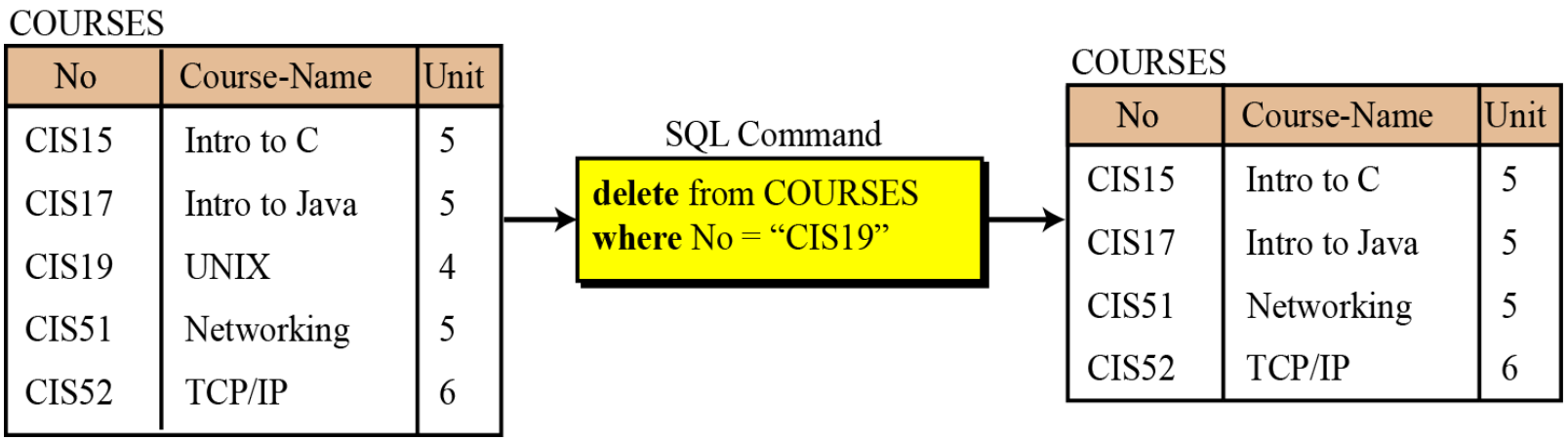


Figure 14.8 An example of a delete operation

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
```

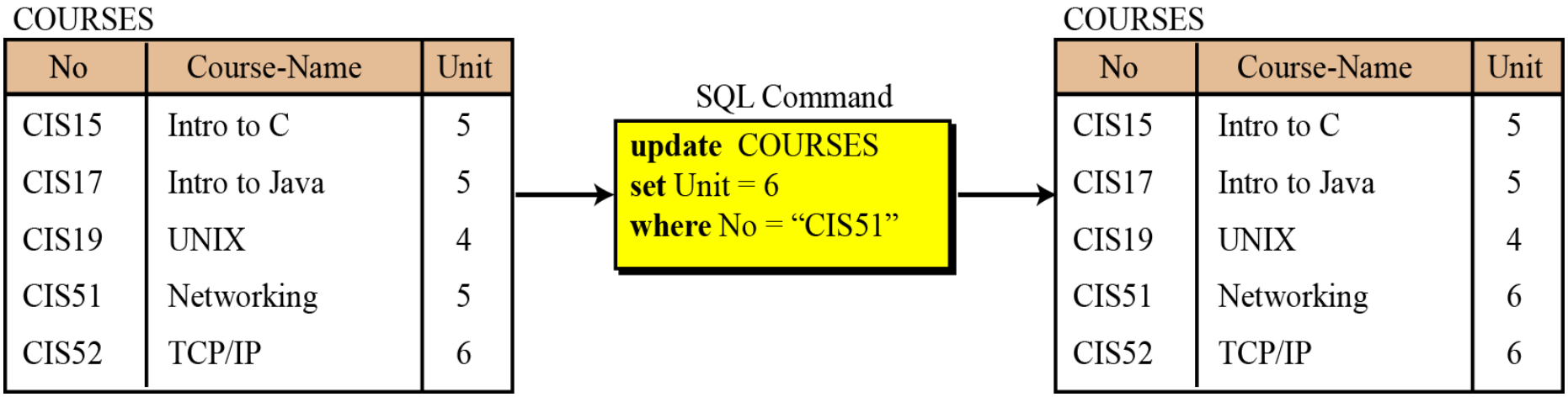


Figure 14.9 An example of an update operation

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
```

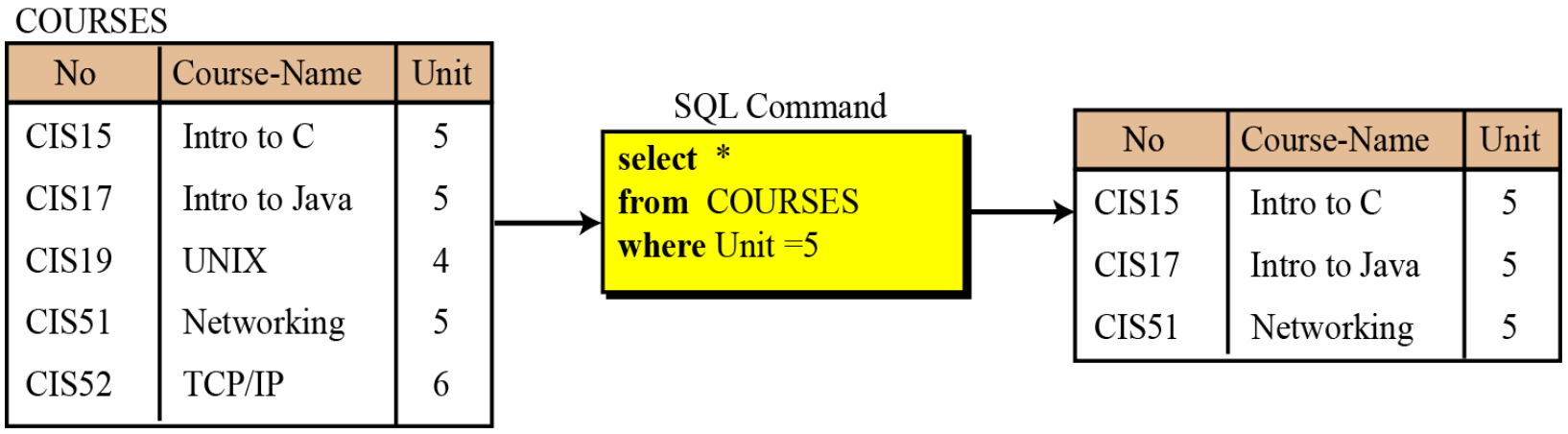


Figure 14.10 An example of an select operation

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

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

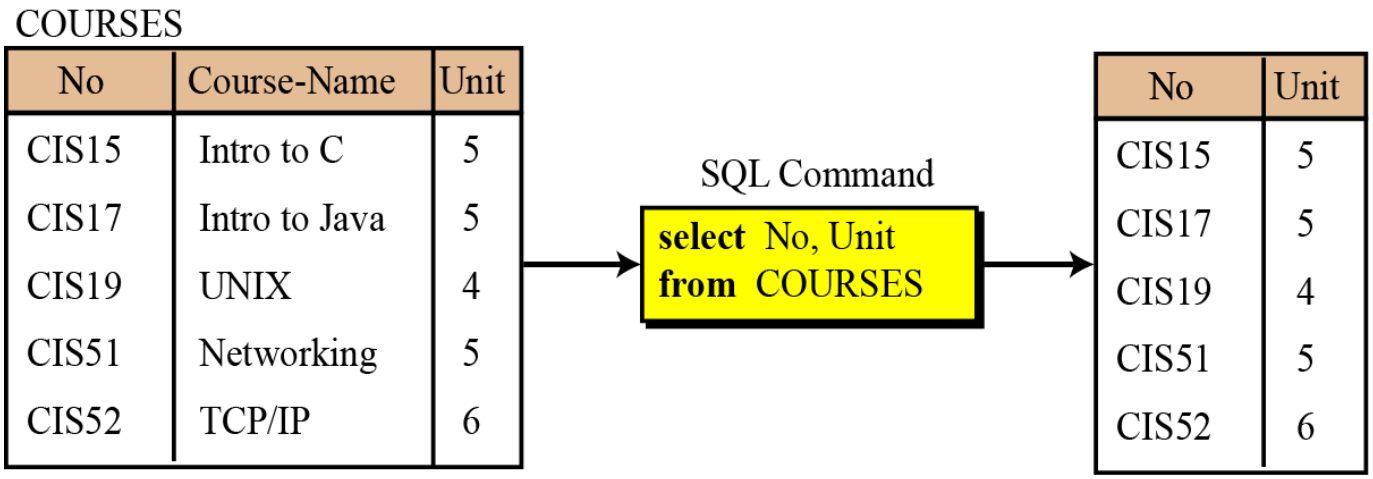


Figure 14.11 An example of a project operation

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

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

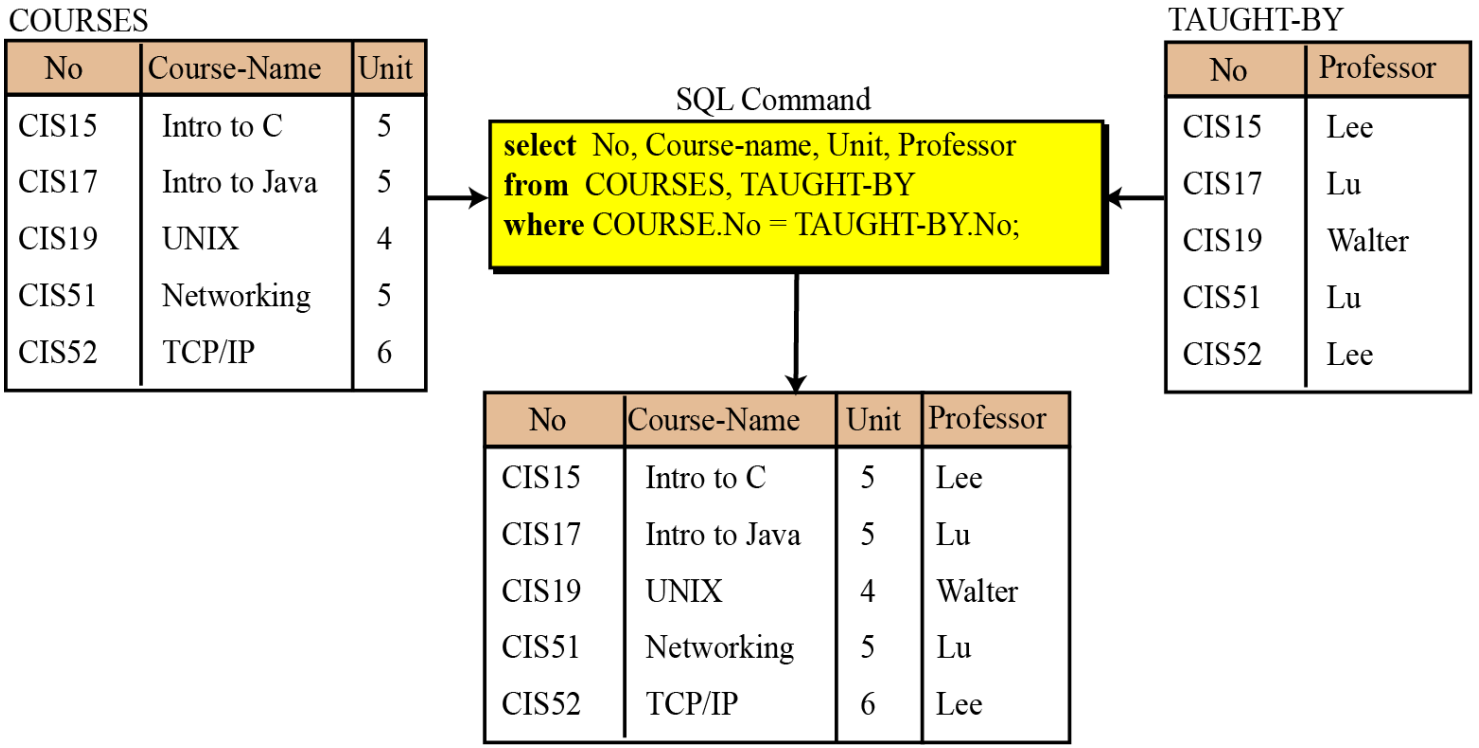


Figure 14.12 An example of a join operation

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

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

CIS15-Roster

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

SQL Command

```
select *
from CIS15-Roster
union
select *
from CIS52-Roster
```

CIS52-Roster

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
342-88-9999	Rich	White

Student-ID	F-Name	L-Name
145-67-6754	John	Brown
232-56-5690	George	Yellow
342-88-9999	Rich	White
345-89-6580	Anne	Green
459-98-6789	Ted	Purple

Figure 14.13 An example of a union operation

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

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

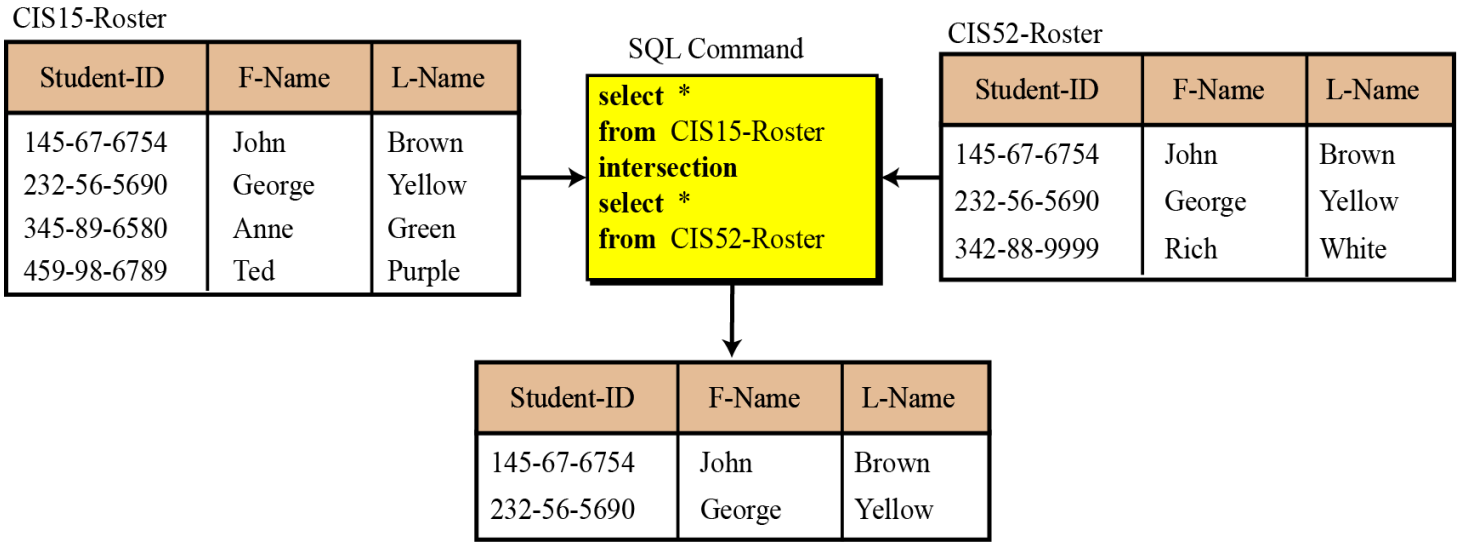


Figure 14.14 An example of an intersection operation

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

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

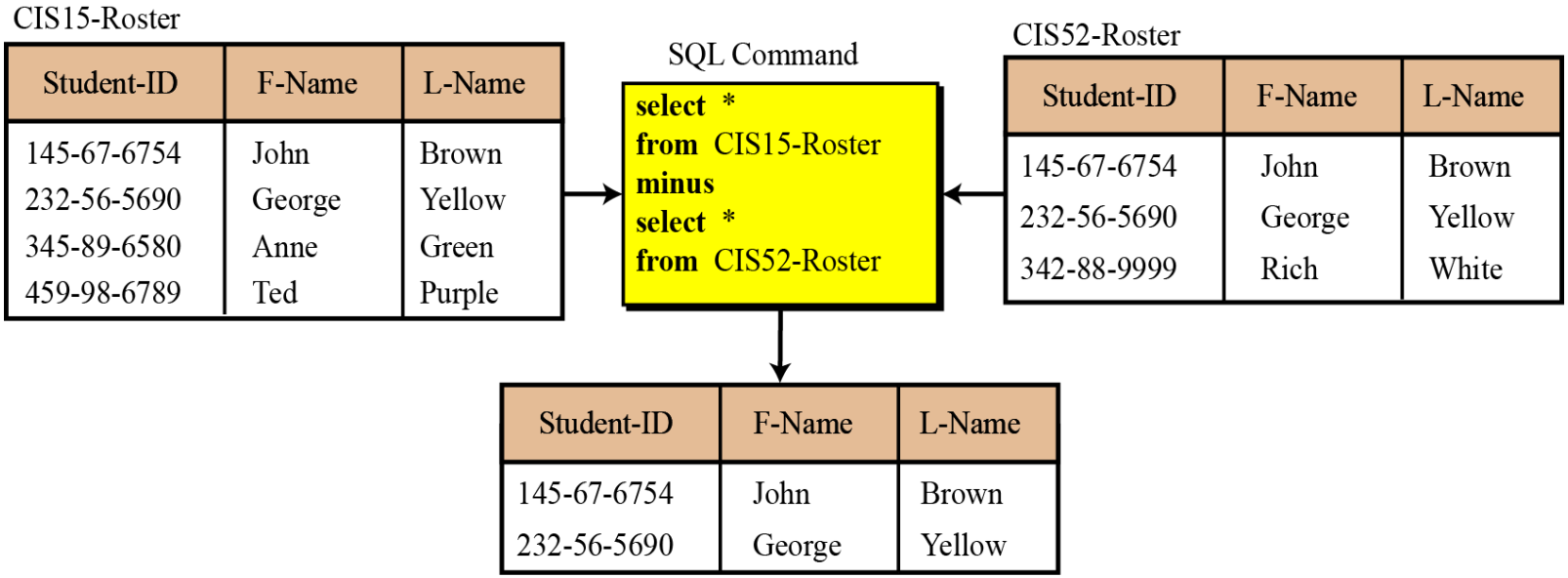
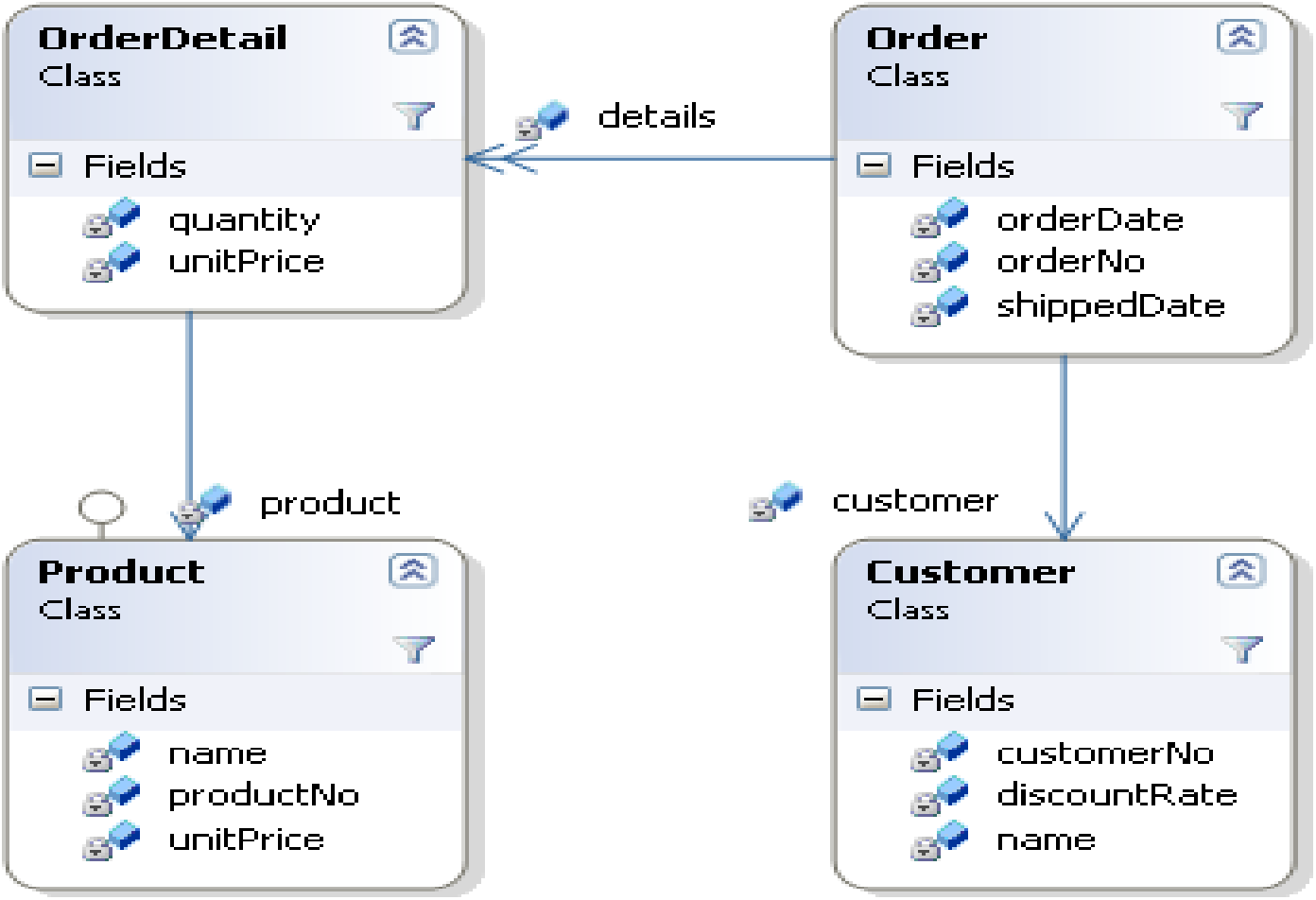


Figure 14.15 An example of a difference operation

Object oriented Model

- In the 1990s, the object-oriented programming paradigm was applied to database technology, creating a new database model known as object databases.
- Object databases also introduce the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases.


OrderDetail
Class
Fields
quantity
unitPrice
details
Order
Class
Fields
orderDate
orderNo
shippedDate
product
customer
Product
Class
Fields
name
productNo
unitPrice
Customer
Class
Fields
customerNo
discountRate
name