

Database Systems

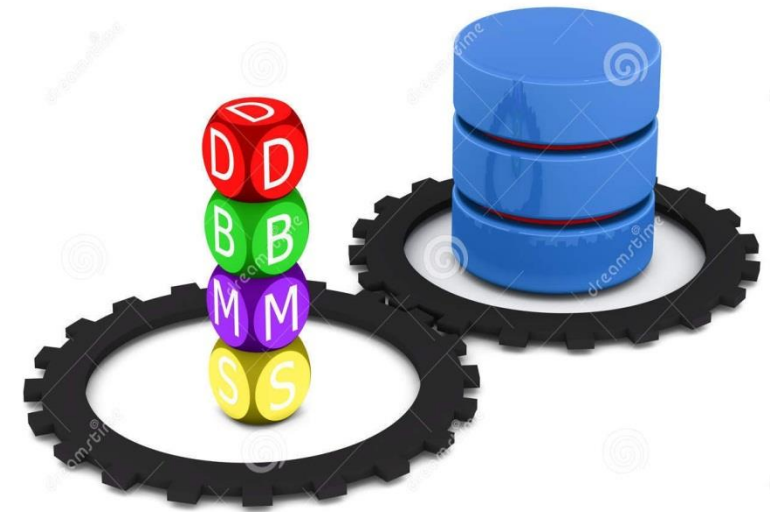
For
Third Stage

Lecture 2

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen



◆ 1.4 Main Characteristics of the Database Approach

- 1) **Self-describing nature of a database system:** A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints. This definition is stored in the DBMS catalogue, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data. The information stored in the catalogue is called meta-data, and it describes the structure of the primary database Figure 1-1. The catalogue is used by the DBMS software and also by database users who need information about the database structure. A general-purpose DBMS software package is not written for a specific database application. Therefore, it must refer to the catalogue to know the structure of the files in a specific database, such as the type and format of data it will access.





2) Insulation between programs and data: In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file. By contrast, DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalogue separately from the access programs. We call this property program-data independence. The characteristic that allows program-data independence is called data abstraction. A DBMS provides users with a conceptual representation of data that does not include many of the details of how the data is stored or how the operations are implemented.

3) Support of multiple views of the data: A database typically has many users, each of whom may require a different perspective or view of the database. A view may be a subset of the database.

4) Sharing of data and multiuser transaction processing: a multiuser DBMS, must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct. For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger.



◆ 1.5 Database related jobs:

Users may be divided into those who actually use and control the content (called “Actors on the Scene”) and those who enable the database to be developed and the DBMS software to be designed and implemented (called “Workers Behind the Scene”).

1.5.1 Actors on the scene

1) Database administrators: In any organization where many people use the same resources, there is a need for a chief administrator to oversee and manage these resources. In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software. Administering these resources is the responsibility of the database administrator (DBA). The DBA is responsible for authorizing access to the database, coordinating and monitoring its use and acquiring software and hardware resources as needed. The DBA is responsible for problems such as security breaches and poor system response time.



Top Skills for Database Administrators



**Basic
Statistics**



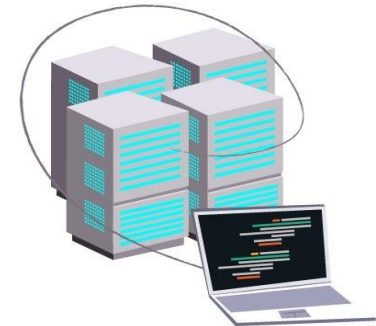
Python



R



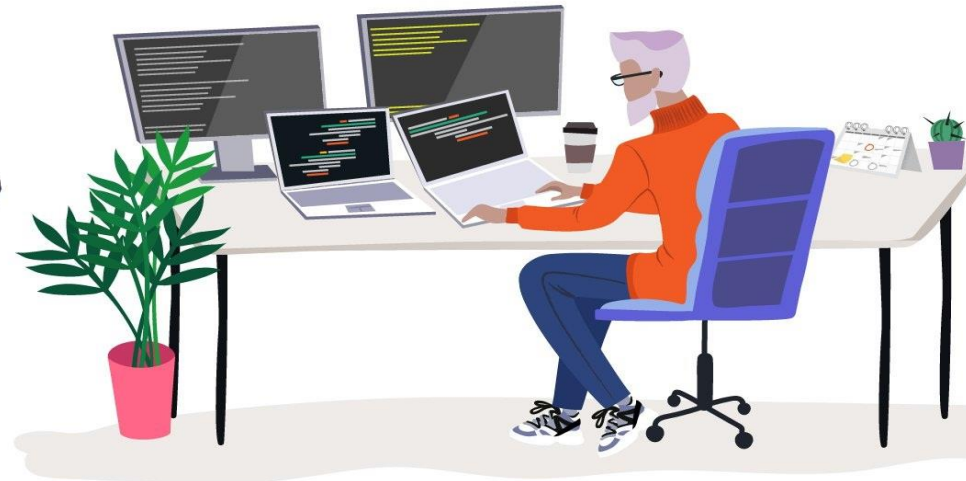
SQL



**Database
Reporting
and Querying**



**Database
Backups and
Recover**



Data Security

Figure 1-2 Database Administrators



2) 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. These tasks are mostly started before the database is actually implemented and populated with data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements.

3) End-users: they use the data for queries, reports and some of them actually update the database content .

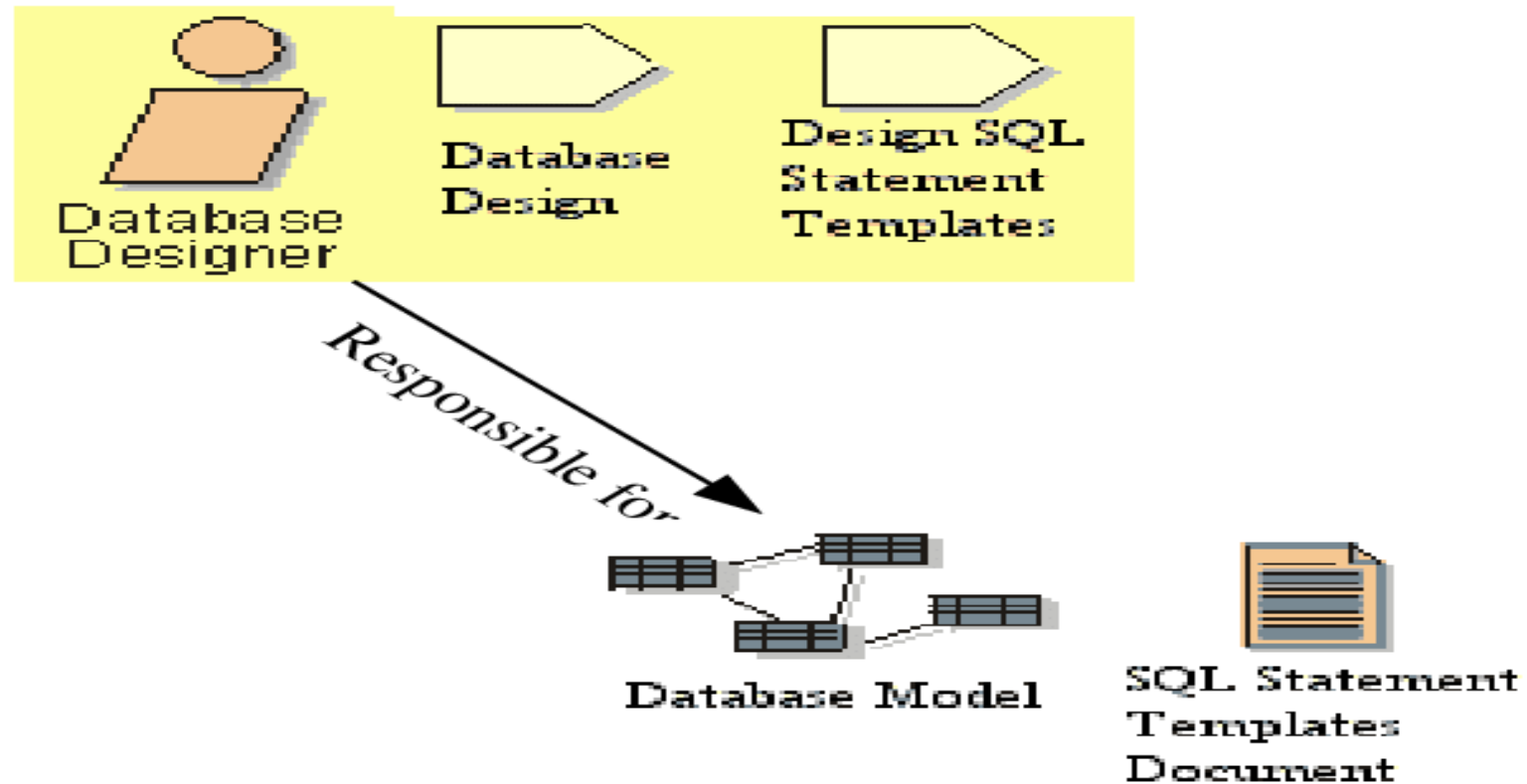


Figure 1-3 Database Designer



Figure 1-4 End Users



Thank You

Database Systems

For
Third Stage

Lecture 3

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen



◆ 2 Database System Concepts and Architecture

2.1 Data Models

- After all business requirements have been gathered for a proposed database, they must be modelled. Models are created to visually represent the proposed database so that business requirements can easily be associated with database objects to ensure that all requirements have been completely and accurately gathered.
- **Data Model:** A data model is a set of concepts to describe the structure of a database, and certain constraints that the database should obey. By structure of a database we mean the data types, relationships, and constraints that should hold on the data .Most data models also include a set of basic operations for specifying retrievals and updates on the database.



- **Data Model Operations:** Operations for specifying database retrievals and updates by referring to the concepts of the data model .In addition to the basic operations provided by the data model, it is becoming more common to include concepts in the data model to specify the dynamic aspect or behaviour of a database application .This allows the database designer to specify a set of valid user-defined operations that are allowed on the database objects. An example of a user-defined operation could be COMPUTE_TOTAL, which can be applied to a STUDENT object . On the other hand, generic operations to insert, delete, modify, or retrieve any kind of object are often included in the basic data model operations.



2.2 Categories of data models

Many data models has been proposed, and we can categorize them according to the types of concepts they use to describe the database structure.

- 1) **Conceptual (high-level, semantic) data models** : A conceptual data model is a high-level description tool of a business's informational needs. It typically includes only the main concepts and the main relationships among them. Typically, these models are used in the early stages to draw the big picture of the database with insufficient detail to build an actual database. This level describes the structure of the whole database for a group of users. Conceptual data models use concepts such as entities, attributes, and relationships (will be discussed later).

In our course, we will study the Entity-Relationship model, which is a popular conceptual data model.



2) Implementation (representational) data models: Provide concepts that fall between the above two, balancing user views with some computer storage details.

3) Physical (low-level, internal) data models: Provide concepts that describe details of how data is stored in the computer. Physical data models describe how data is stored in the computer by representing information such as record formats, record orderings, and access paths .An access path is a structure that makes the search for particular database records efficient.



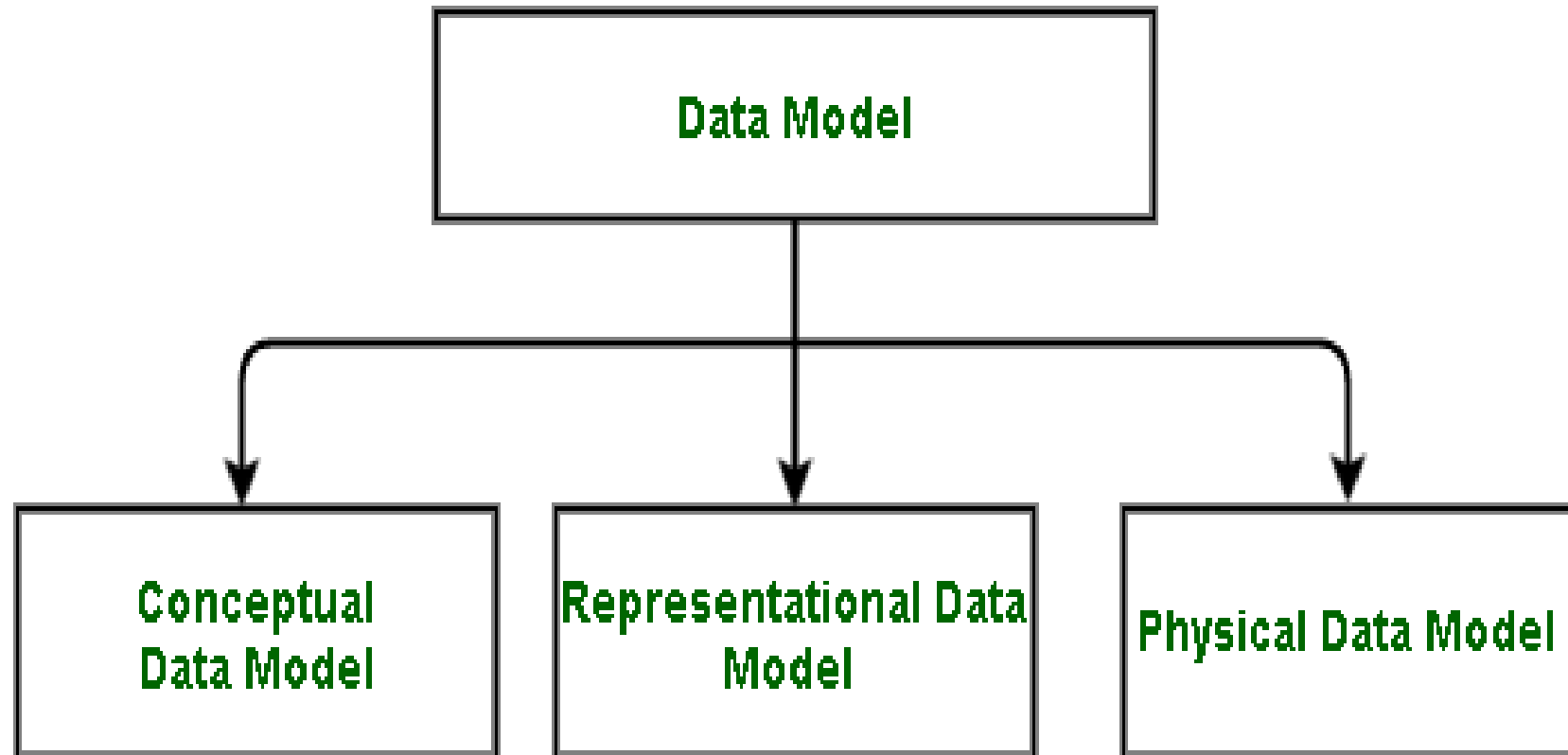


Figure 1-5 Data Model



Thank You

Database Systems

For
Third Stage

Lecture 4

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen



◆ 2 Database System Concepts and Architecture

2.3 Schemas versus Instances

- In any data model it is important to distinguish between the description of the database and the database itself .The description of a database is called the **database schema**, which is specified during database design and is not expected to change frequently .
- Most data models have certain conventions for displaying the schemas as diagrams . A displayed schema is called a **schema diagram**_.Figure 1-6 example of a schema diagram, shows an example of schema diagram .
- The diagram displays the structure of each record type but not the actual instances of records .We call each object in the schema—such as STUDENT or COURSE— a **schema construct**. A schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints .
- Other aspects are not specified in the schema diagram; for example, the below example shows neither the data type of each data item nor the relationships among the various files .





STUDENT

Name	StudentNumber	Class	Major
------	---------------	-------	-------

COURSE

CourseName	CourseNumber	CreditHours	Department
------------	--------------	-------------	------------

PREREQUISITE

CourseNumber	PrerequisiteNumber
--------------	--------------------

SECTION

SectionIdentifier	CourseNumber	Semester	Year	Instructor
-------------------	--------------	----------	------	------------

GRADE_REPORT

StudentNumber	SectionIdentifier	Grade
---------------	-------------------	-------

Figure 1-6 Example of a schema diagram

- The actual data in a database may change quite frequently .The data in the database at a particular moment in time is called **a database state or snapshot** .It is also called **the current set of occurrences or instances** in the database .
- In a given database state, each schema construct has its own current set of instances; for example, the STUDENT construct will contain the set of individual student entities (records) as its instances .Many database states can be constructed to correspond to a particular database schema .Every time we insert or delete a record, or change the value of a data item in a record, we change one state of the database into another state .



2.4 Three-Schema Architecture

- The goal of the three-schema architecture, illustrated in Figure 1-7 , is to separate the user applications from the physical database to provide three of the important characteristics of the database approach, which are:
 - 1) Use of a catalogue to store the database description (schema) so as to make it self-describing.
 - 2) Insulation of programs and data (program-data and program-operation independence).
 - 3) Support of multiple user views.



➤ In this architecture, schemas can be defined at the following three levels:

1) **The internal level** has an internal schema, 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.

2) **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 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. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.



3) 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. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.



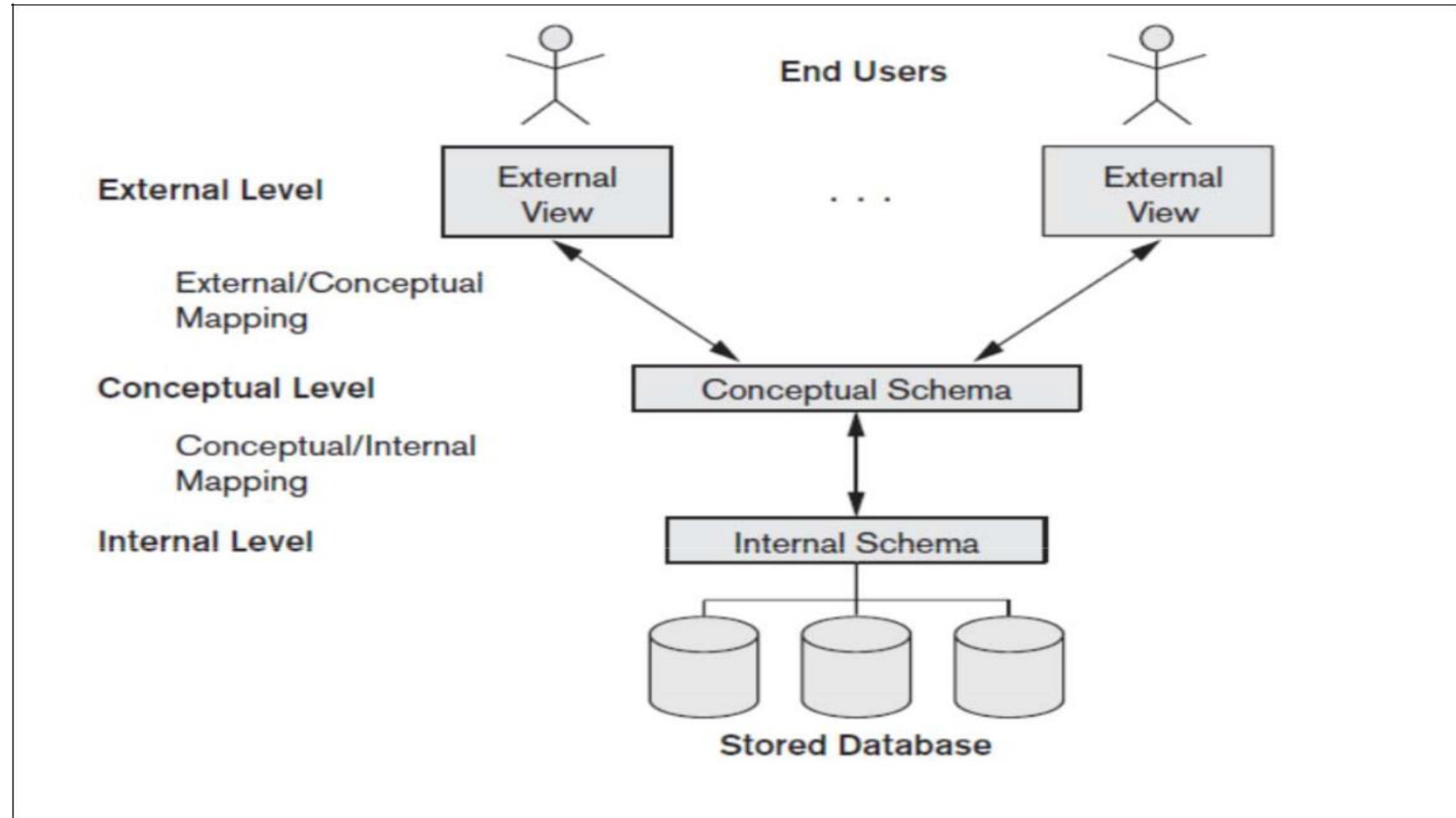


Figure 1-7 Three-schema architecture.

2.5 Data Independence

- The three-schema architecture can be used to further explain the concept of data independence, which can be **defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level**. We can define two types of data independence:

1) Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item). In the last case, external schemas that refer only to the remaining data should not be affected.



2) Physical data independence is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized—for example, by creating additional access structures (like an index) —to improve the performance of retrieval or update.





Thank You

Database Systems

For
Third Stage

Lecture 5

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen



◆ 3 Data Modelling Using the ER Model

3.1 Introduction

- In this lecture we will follow the traditional approach of concentrating on the database structures and constraints during database design .We will present the modelling concepts of the Entity-Relationship (ER) model.
- **The Entity-Relationship(E-R) model** which is a high-level data model. E-R model consists of a collection of basic objects and of the relationship among these objects. It is basically useful in the design and communication of the logical database model.
- **Entity-Relationship Diagram**, it is used to analyze to structure of the **Database**. It shows **relationships** between **entities** and their **attributes**



- Example of E-R Diagram – The below diagram shows the example of an E-R model.

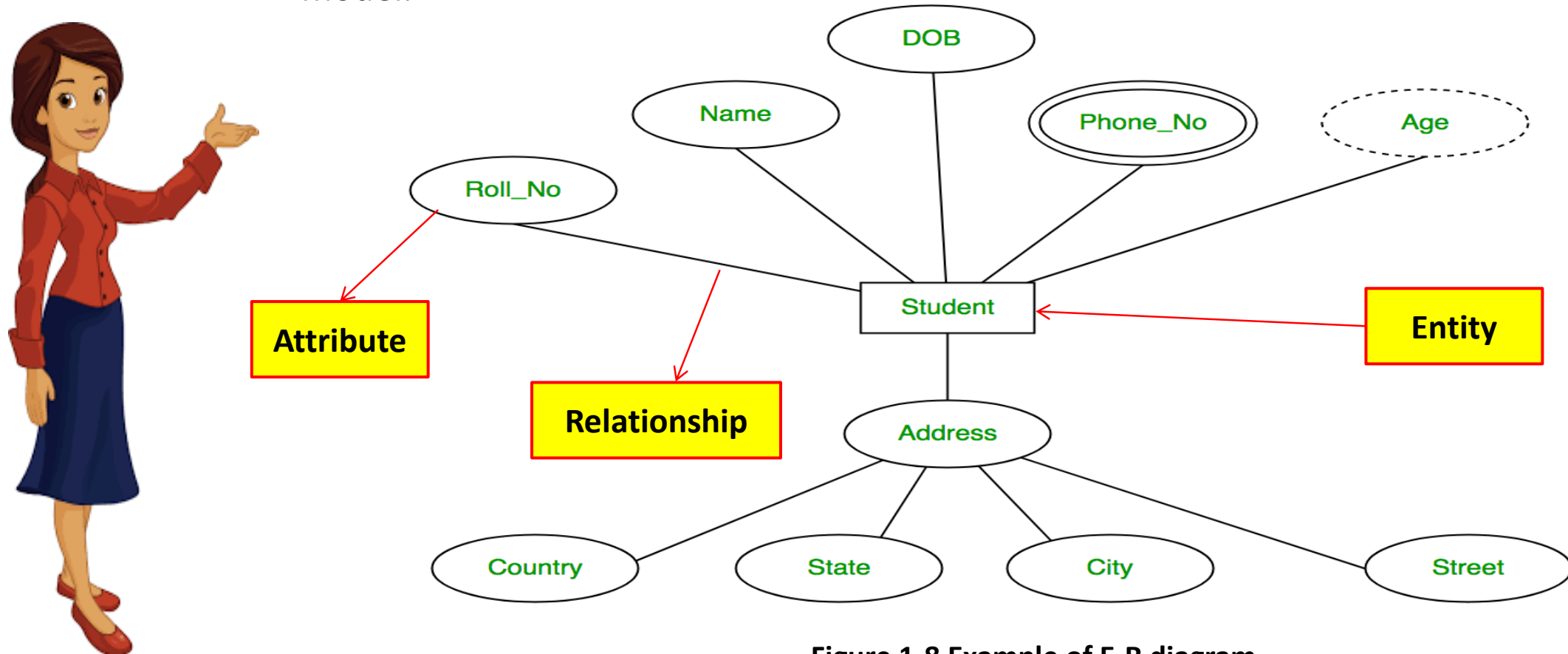


Figure 1-8 Example of E-R diagram

3.2 Example COMPANY Database

- This example will be used to demonstrate the use of ER modelling techniques and diagrams . The requirements of the Company are:
 - 1) The company is organized into **DEPARTMENTS** .Each department has **a name, number , location** and an **employee** who manages the department .We should keep track of the start date of the department manager .A department may have several locations .
 - 2) Each **department** controls a number of **PROJECTs**.
 - 3) Each **project** has a **name, number** and **Budget**.



2.3 Example COMPANY Database

- 4) We should store each **EMPLOYEE's** social security number, first name and last name. Each **employee** works for one department but may work on several projects .We keep track of the number of hours per week that an employee currently works on each project .We also keep track of the *direct supervisor* of each employee.
- 5) Each **DEPARTMENT** must have a manager at all times.
- 6) Some **departments** may control no projects.
- 7) Not every **employee** is a supervisor and not every **employee** has a supervisor.
- 8) A **PROJECT** can have several **EMPLOYEE** working on it.



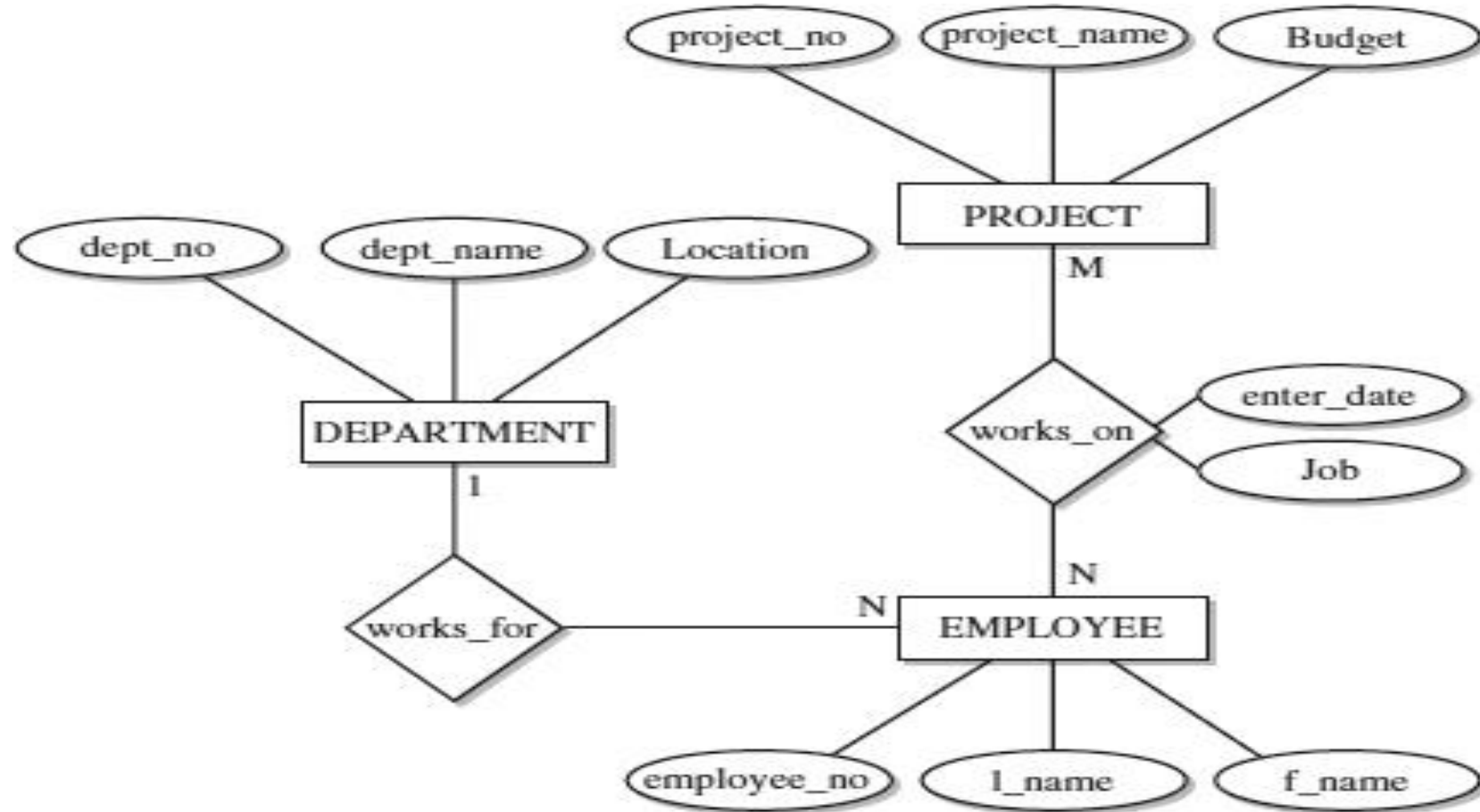


Figure 1-8 Example of E-R diagram



Thank You

Database Systems

For
Third Stage

Lecture 6

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen





3.4 Entity Types , Entity Sets, Attributes, and Keys

- The ER model describes data as **entities, relationships, and attributes** .In the next sections we introduce the concepts of entities and their attributes .We discuss entity types and key attributes and then we will specify the initial conceptual design of the entity types for the COMPANY database .





3.5 Entities and Attributes

- **Entities** are specific objects or things in the mini-world that are represented in the database .For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT.
- **Attributes** are properties used to describe an entity .For example an EMPLOYEE entity may have a Name, SSN, Address, BirthDate.
- A specific entity will have a value for each of its attributes .For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', BirthDate='09-JAN-55'.
- Each attribute has a value set (or data type) associated with it – e.g . integer, string, subrange, ...





3.5.1 Types of Attributes

- 1) **Simple** : Each entity has a single atomic value for the attribute; for example, SSN.
- 2) **Composite**: The attribute may be composed of several components. For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName). Composition may form a hierarchy where some components are themselves composite.
- 3) **Multi-valued** : An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.





3.5.1 Types of Attributes

4) **Derived Attributes**: In some cases two (or more) attribute values are related—for example, the Age and BirthDate attributes of a person .For a particular person entity, the value of Age can be determined from the current (today's) date and the value of that person's BirthDate .The Age attribute is hence called a derived attribute and is said to be derivable from the BirthDate attribute, which is called a stored attribute .Some attribute values can be derived from related entities; for example, an attribute NumberOfEmployees of a department entity can be derived by counting the number of employees related to (working for) that department .

5) **Null Values** :In some cases a particular entity may not have an applicable value for an attribute, nor don't have a value, in this case we can say that value of this attribute for this entity is Null .





3.5.2 Entity Types and Key Attributes

- Entities with the same basic attributes are grouped or typed into an entity type . For example, the EMPLOYEE entity type or the PROJECT entity type.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type .For example, SSN of EMPLOYEE.
- A key attribute may be composite .For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key .For example, the CAR entity type may have two keys:
 - VehicleIdentificationNumber (popularly called VIN) and
 - VehicleTagNumber (Number, State), also known as license_plate number.





Thank You

Database Systems

For
Third Stage

Lecture 7

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen





3.6 Relationships, Relationship Types and maximum cardinality

- A relationship relates two or more distinct entities with a specific meaning .For example, **EMPLOYEE** John Smith works on the ProductX **PROJECT** or **EMPLOYEE** Franklin Wong manages the Research **DEPARTMENT**.
- Relationships of the same type are grouped or typed into a relationship type . For example, the **WORKS_ON** relationship type in which **EMPLOYEEs** and **PROJECTs** participate, or the **MANAGES** relationship type in which **EMPLOYEEs** and **DEPARTMENTs** participate.
- The degree of a relationship type is the number of participating entity types . Both **MANAGES** and **WORKS_ON** are binary relationships (two entity types).



- **More than one relationship type** can exist with the same participating entity types .For example, **MANAGES** and **WORKS_FOR** are distinct relationships between **EMPLOYEE** and **DEPARTMENT**, but with different meanings and different relationship instances.
- Figure 1-10 The **WORKS_FOR** relationship between **EMPLOYEE** and **DEPARTMENT** , Many-to-one (N:1) and Figure 1-11 the **WORKS_ON** relationship between **EMPLOYEE** and **PROJECT**, Many-to-many (M:N)are examples of relationship instances of the **WORKS_FOR** relationship between **EMPLOYEE** and **DEPARTMENT** and the relationship instances of the **WORKS_ON** relationship between **EMPLOYEE** and **PROJECT**.



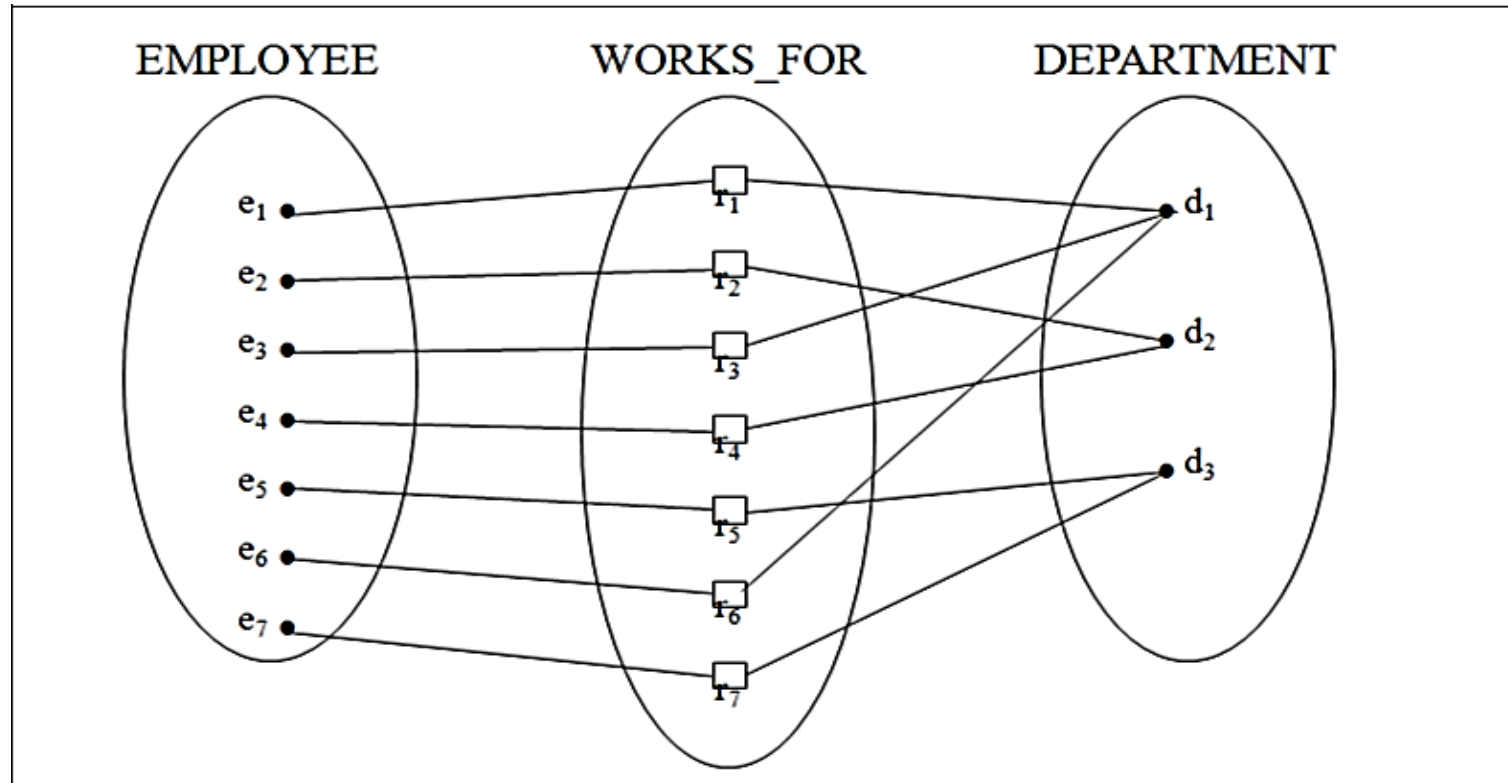
- To find out the **maximum cardinality** for a relationship type between 2 entity types for example the WORKS_FOR relationship between EMPLOYEE and DEPARTMENT you can ask the following questions:

1) For the EMPLOYEE side: How many EMPLOYEE can have a relationship with a single DEPARTMENT? the answer is N

2) For the DEPARTMENT side: How many DEPARTMENT can have a relationship with a single EMPLOYEE? the answer is 1

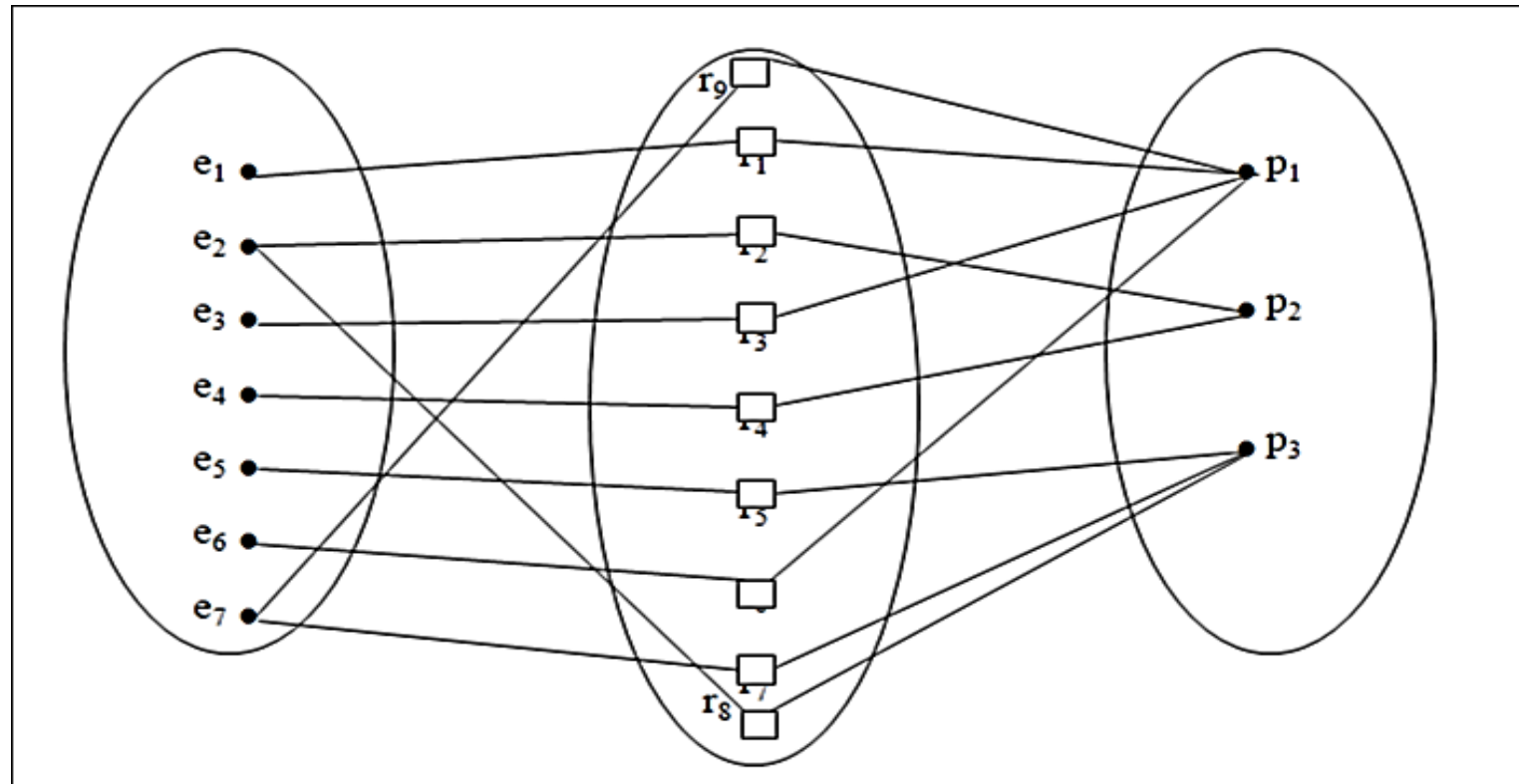
So the relationship is N:1





**Figure 1-10 The WORKS_FOR relationship between EMPLOYEE and DEPARTMENT ,
Many-to-one (N:1)**





**Figure 1-11 the WORKS_ON relationship between EMPLOYEE and PROJECT,
Many-to-many (M:N)**





Thank You

Database Systems

For
Third Stage

Lecture 8

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen





3.7 Types of Keys

1- Primary Key is the field that identify each table in the database, and identifies each record in a table uniquely.

2- Foreign Key It is a field that represents the primary key in another table, it called a foreign keys because it is not from an existing fields in the table, but its added to the table to link with another table.



Primary Key :

customer_id	customer_name	address	phone
01	Mohammed	Mosul	09567
02	Ali	Baghdad	07654
03	Saad	Baghdad	08654
04	Ali	Basrah	

Foreign Key:

order_id	cust_id	order_name	order_date
001	01	a	2011-01-05
002	02	b	2010-07-10
003	02	c	2009-05-05
004	04	d	2011-11-17
005	01	e	2006-12-07





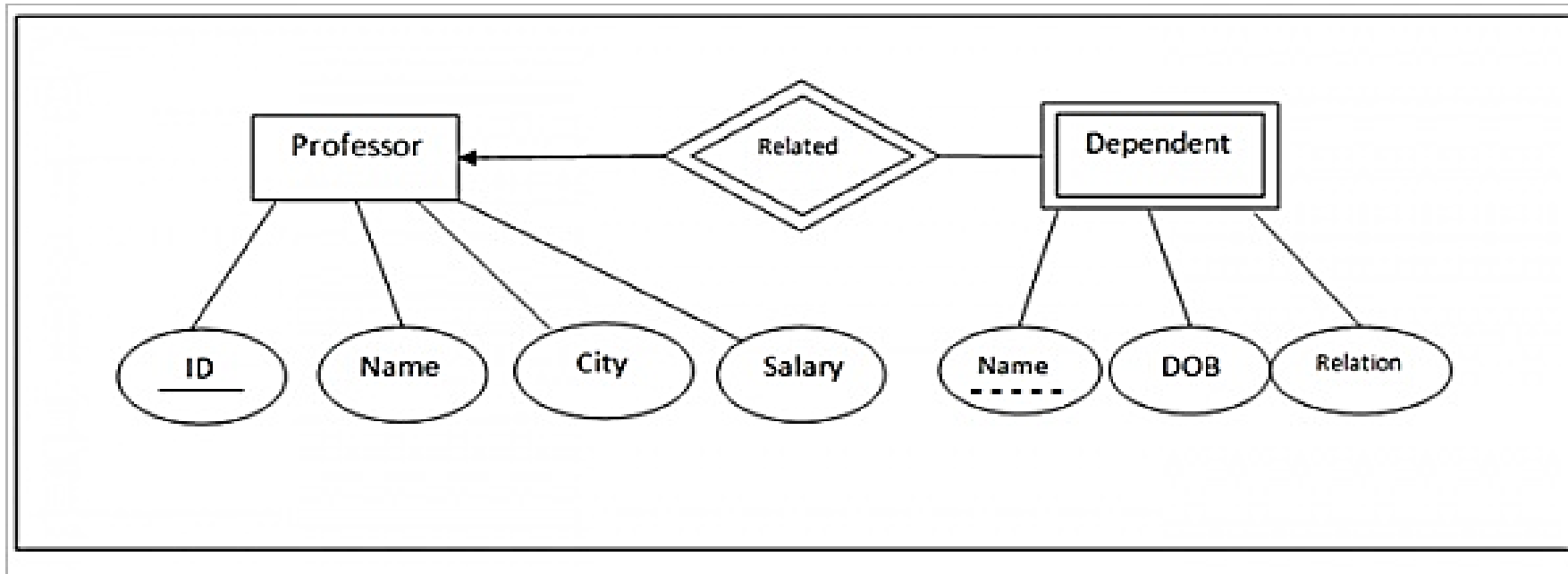
3.8 Weak Entity Types

- An entity that does not have a key attribute.
- A weak entity do not have a primary key and are dependent on the parent entity. It mainly depends on other entities..
- Entities are identified by the combination of:
 - A partial key of the weak entity type
 - The particular entity they are related to in the identifying entity type



Example of Strong and Weak Entity

The example of strong and weak entity can be understood by the below figure.



- The Strong Entity is **Professor**, whereas **Dependent** is a Weak Entity. **ID** is the primary key (represented with a line) and Name in **Dependent** entity is called **Partial Key** (represented with a dotted line).



3.9 ER Notation :

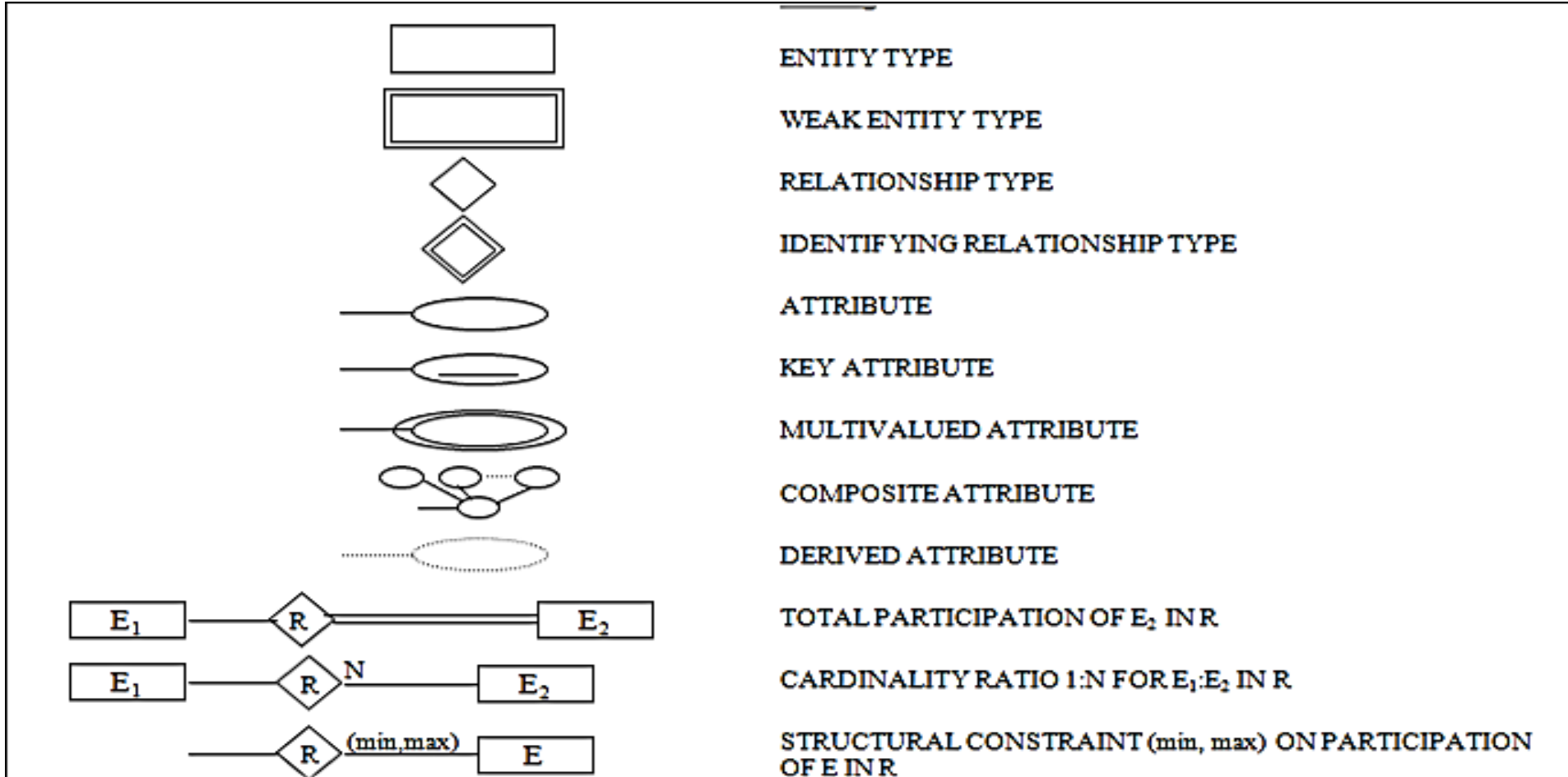
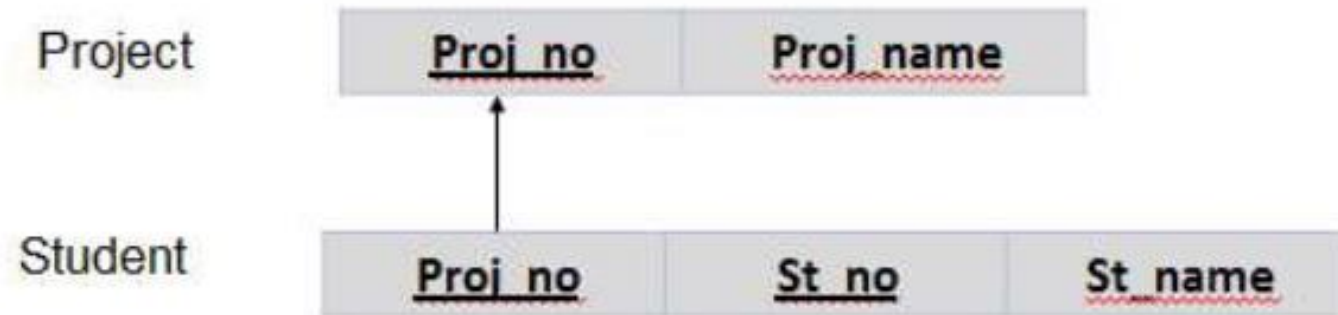
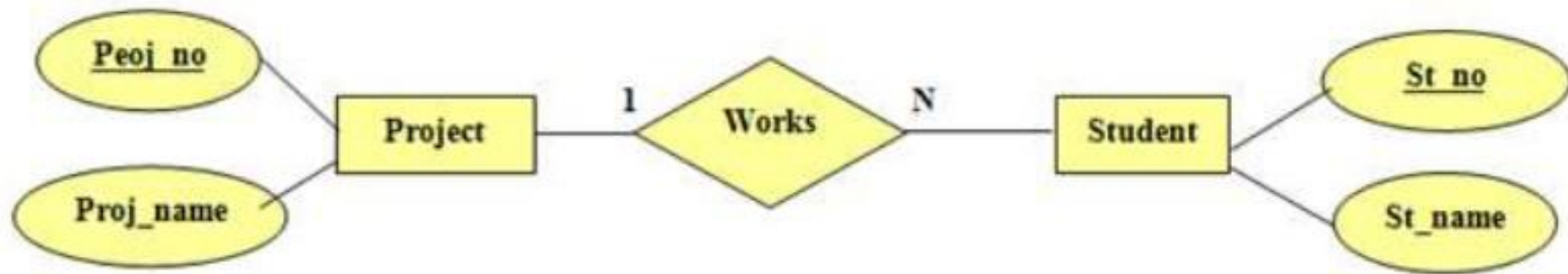


Figure 1-12 ER notation





Example :





Thank You

Database Systems

For
Third Stage

Lecture 9

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen





Example

- Suppose we have a (Company management System), this company contains many Employees that are belongs to Departments, and works on Projects One department have many employees, and every employee is works to one project, as well as every one project is worked by many employees.
- Propose ER diagram to this system?





Solution:

الحل:

1- Entities:

Employee (employee Id, employee name)

Department (department Id, department name)

Project (project Id, project name)

1- الكيانات:

الموظف (رمز الموظف، اسم الموظف)

القسم (رمز القسم، اسم القسم)

المشروع (رمز المشروع، اسم المشروع)

2- Relationships:

The employee's relationship with the department

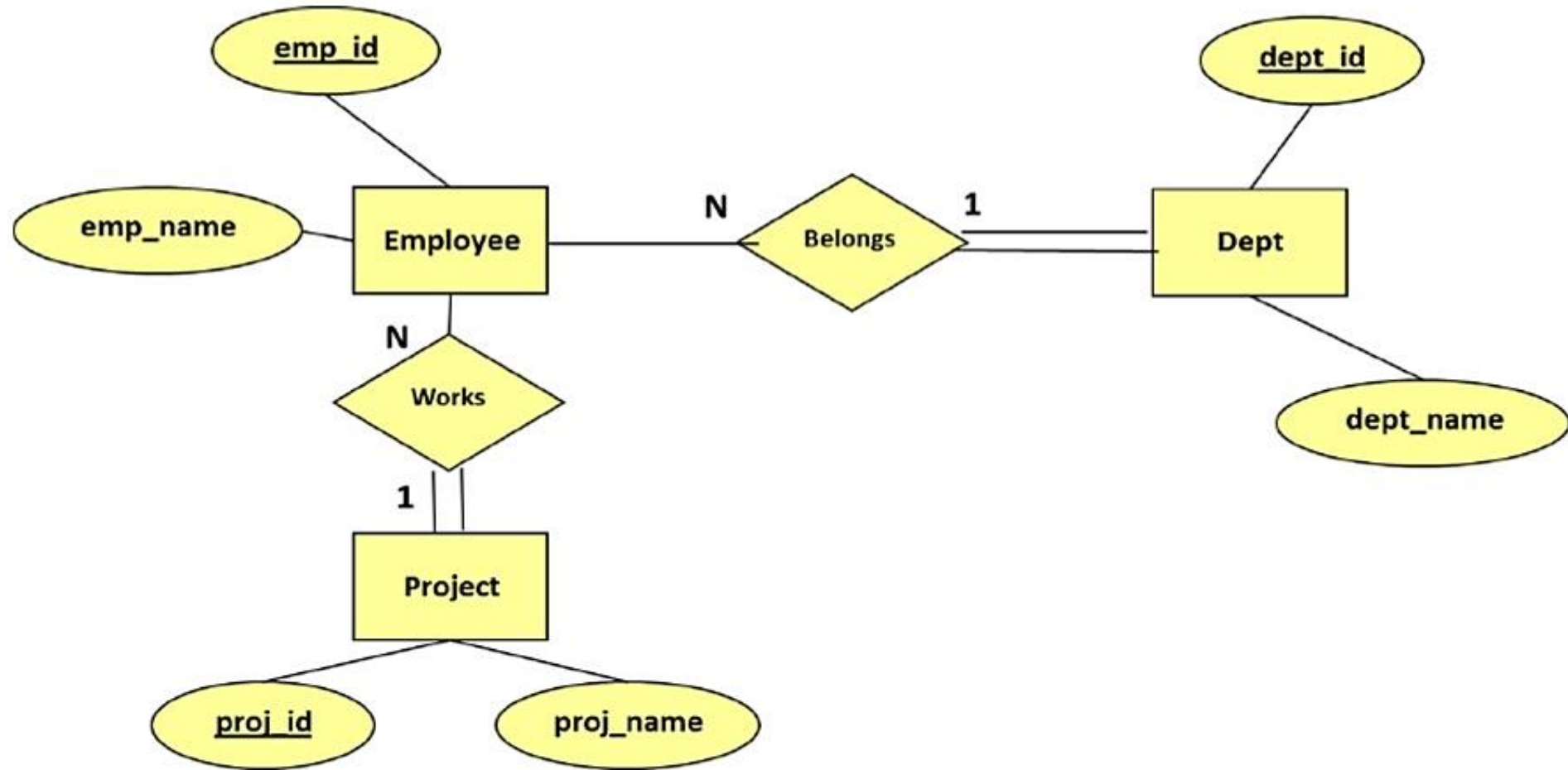
The employee's relationship with the project

2- العلاقات:

علاقة الموظف مع القسم

علاقة الموظف مع المشروع







Notes:

- The employee belongs to one department and the department has several employees (1 N).

ملاحظات:

- الموظف ينتمي لقسم واحد والقسم به عدة موظفون 1:N

- The employee works on one project and the project is working on several employees (1 N).

- الموظف يعمل على مشروع واحد والمشروع يعمل عليه عدة موظفون 1:N

- It may be that some employees have no departments (partial participation). It cannot be The department has no employees (full participation).

- يمكن أن يكون بعض الموظفين ليس لديهم اقسام (اشتراك جزئي) ولا يمكن أن يكون القسم إلا وبه موظفون (اشتراك كلي)

- Some employees may not have projects (partial participation) The project is only and employees work on it (full participation).

- يمكن أن يكون بعض الموظفين ليس لديهم مشاريع (اشتراك جزئي) ولا يمكن أن يكون المشروع إلا ويعمل عليه موظفون (اشتراك كلي).



Thank You

Database Systems

For
Third Stage

Lecture 10

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen





Example

- Suppose we have a (Library management System), this library contains many Books that composed by Authors, every author can compose one or more book These books are published by one Publisher that can publish one or more book
- Propose ER diagram to this system?





Solution:

الحل:

1- Entities:

الكيانات:

- Book (book number, book title)
- Author (author Id, author name, title)
- Publisher (publisher Id, publisher name)

الكتاب (رقم الكتاب، عنوان الكتاب)
المؤلف (رمز المؤلف، اسم المؤلف، العنوان)
الناشر (رمز الناشر، اسم الناشر)

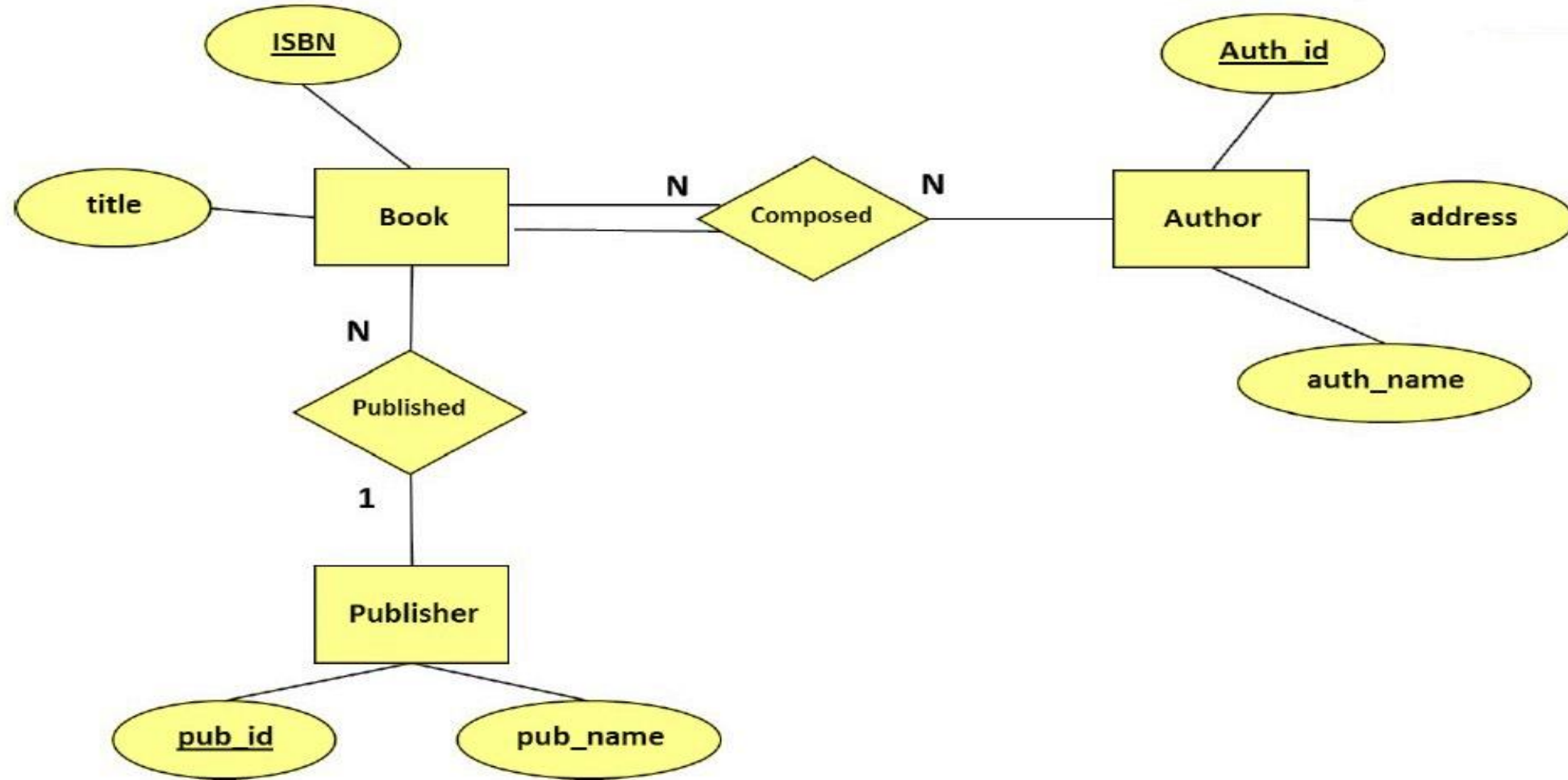
2- Relations:

العلاقات:

- The book's relationship with the author
- The book's relationship with the publisher

علاقة الكتاب مع المؤلف
علاقة الكتاب مع الناشر







Notes:

- The book is author by one or more authors in cooperation. The author can author more than one book, N: N

- The book is published in one publishing , and the publishing can publish more than one book N: 1

- A book can only be contain by an author (full particapation), and the author may not have a book (partial particapation).

- The book can be unpublished (partial particapation), and the publishing can not contain books (partial particapation).

ملاحظات:

- الكتاب يؤلفه مؤلف واحد او أكثر من مؤلف بالتعاون، ويمكن للمؤلف تأليف أكثر من كتاب N: N

- ينشر الكتاب في دار نشر واحد، ودار النشر يمكنه ان ينشر أكثر من كتاب N: 1

- لا يمكن للكتاب إلا وبه مؤلف (اشتراك كلي)، ويمكن أن يكون المؤلف ليس لديه كتاب (اشتراك جزئي).

- يمكن ان يكون الكتاب غير منشور (اشتراك جزئي)، ويمكن لدار النشر أن لا يحتوي على كتب (اشتراك جزئي)



Thank You

Database Systems

For
Third Stage

Lecture 11

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen





Example

Suppose you have a (Learning management System), this system have a number of students those registered to many courses, The registrar of learning system is write the year and the class number when registering a student for any course

- **Propose ER diagram to this system?**





Solution:

الحل:

1- Entities:

الكيانات:

- Student (student name - university number(id) - address) الطالب (اسم الطالب - الرقم الجامعي - العنوان)
- Course (course name - course number - number of hours) المقرر (اسم المقرر - رقم المقرر - عدد الساعات)

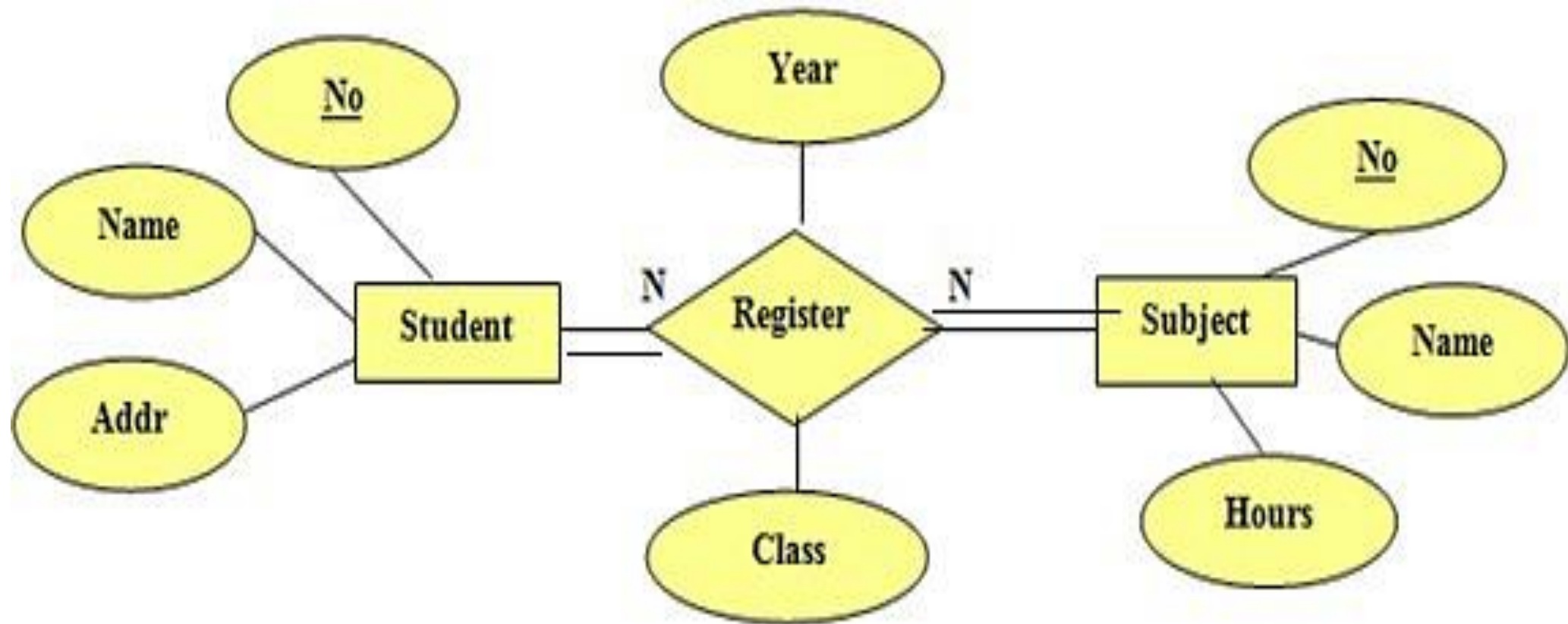
2- Relations:

العلاقات:

- The relationship of the student's registration to a course.

علاقة تسجيل الطالب لمقرر





Note / Course = Subject





Notes:

ملاحظات:

- The student can register a set of courses (relationship type N).
 - The course is registered by a set of students (relationship type N).
 - Some students cannot have courses (full participation).
 - Courses cannot be register by students (full participation).
 - “year, and class” are adjectives for the relationship “register” and therefore have been added to it.
- الطالب يمكن أن يسجل مجموعة من المقررات (نوع العلاقة N)
 - (المقرر يسجله مجموعة من الطلبة (نوع العلاقة N)
 - لا يمكن أن يكون بعض الطلبة ليس لديهم مقررات (اشتراك كلي)
 - لا يمكن أن تكون المقررات غير مسجل فيها طلبة (اشتراك كلي)
 - “السنة، والشعبة” هي صفات للعلاقة “يسجل” ولذلك اضيفت لها.



Thank You

Database Systems

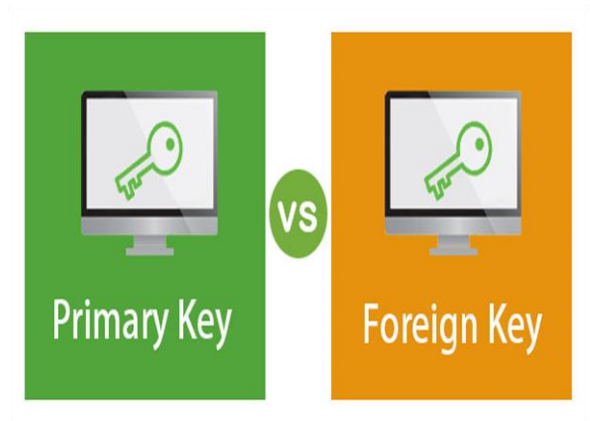
For
Third Stage

Lecture 12

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen



Foreign Key

A foreign key is a key used to **link two tables** together. This is sometimes called a **referencing key**. Foreign Key is a column or a combination of columns **whose values match a Primary Key in a different table**. The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table. If a table has a primary key defined on any fields, then you can not have two records having the same value of that fields.

An example might be a student table that contains the **course_id** the student is attending. Another table lists the courses on offer with **course_id** being the primary key. The 2 tables are linked through **course_id** and as such **course_id** would be a foreign key in the student table.





Example of Foreign Key

Let's discuss an example to understand the working of a foreign key.

Consider two tables **Student** and **Department** having their respective attributes as shown in the below table structure:

Student

Stud_Id	Name	Course
101	John	Computer
105	Merry	AI
107	Sheero	Biology
108	Bisle	Maths

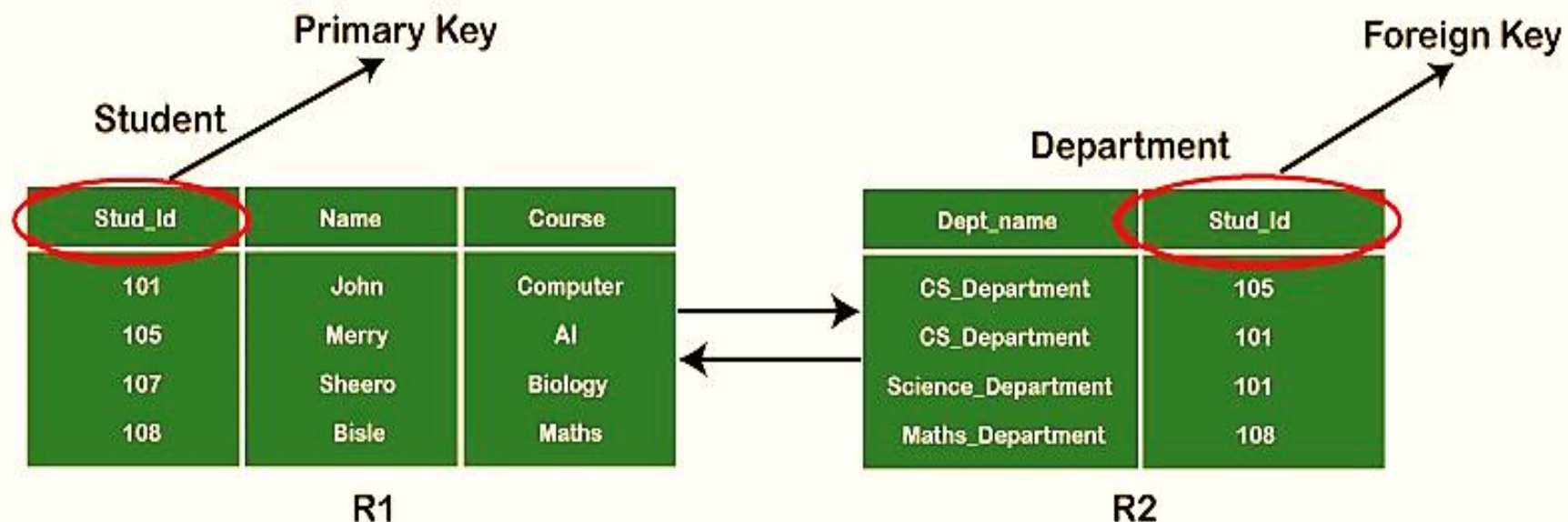
Department

Dept_name	Stud_Id
CS_Department	105
CS_Department	101
Science_Department	101
Math_Department	108



In the tables, one attribute, you can see, is common, that is **Stud_Id**, but it has different key constraints for both tables. In the Student table, the field Stud_Id is a **primary key** because it is uniquely identifying all other fields of the Student table. On the other hand, Stud_Id is a **foreign key** attribute for the Department table because it is acting as a primary key attribute for the Student table. It means that both the Student and Department table are linked with one another because of the Stud_Id attribute.

In the below-shown figure, you can view the following structure of the relationship between the two tables.





Thank You

Database Systems

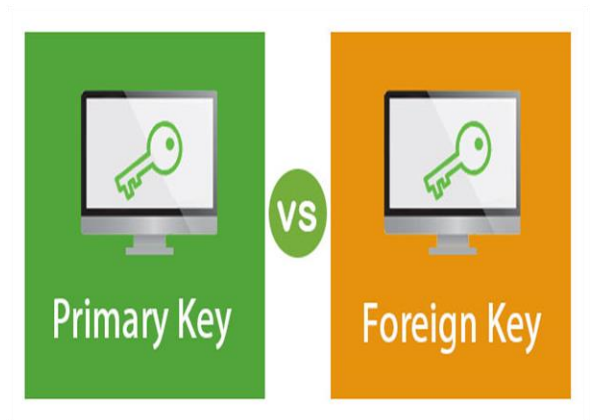
For
Third Stage

Lecture 13

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen





The difference between Primary Key and Foreign Key:

S.NO.	PRIMARY KEY	FOREIGN KEY
1	A primary key is used to ensure data in the specific column is unique.	A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables.
2	It uniquely identifies a record in the relational database table.	It refers to the field in a table which is the primary key of another table.
3	Only one primary key is allowed in a table.	Whereas more than one foreign key are allowed in a table.
4	It does not allow NULL values.	It can also contain NULL values.
5	Its value cannot be deleted from the parent table.	Its value can be deleted from the child table.
6	It is a combination of UNIQUE and Not Null constraints.	It can contain duplicate values and a table in a relational database.





Example of Primary Key and Foreign Key

<u>studentId</u>	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042

Foreign Keys

Relationship

Primary Keys

<u>courseId</u>	courseName
A004	Accounts
C002	Computing
P301	History
S042	Short Course



Example of Primary Key and Foreign Key

Rep

RepNum	LastName	FirstName	Street	City	State	Zip	Commission	Rate
20	Kaiser	Valerie	624 Randall	Grove	FL	33321	\$20,542.50	0.05
35	Hull	Richard	532 Jackson	Sheldon	FL	33553	\$39,216.00	0.07
65	Perez	Juan	1626 Taylor	Fillmore	FL	33336	\$23,487.00	0.05

Repnum is a *Foreign key* borrowed from Rep table



Customer

CustomerNum	CustomerName	Street	City	State	Zip	Balance	CreditLimit	RepNum
148	Al's Appliance and Sport	2837 Greenway	Fillmore	FL	33336	\$6,550.00	\$7,500.00	20
282	Brookings Direct	3827 Devon	Grove	FL	33321	\$431.50	\$10,000.00	35
356	Ferguson's	382 Wildwood	Northfield	FL	33146	\$5,785.00	\$7,500.00	65
408	The Everything Shop	1828 Raven	Crystal	FL	33503	\$5,285.25	\$5,000.00	35
462	Bargains Galore	3829 Central	Grove	FL	33321	\$3,412.00	\$10,000.00	65
524	Kline's	838 Ridgeland	Fillmore	FL	33336	\$12,762.00	\$15,000.00	20
608	Johnson's Department Store	372 Oxford	Sheldon	FL	33553	\$2,106.00	\$10,000.00	65
687	Lee's Sport and Appliance	282 Evergreen	Altonville	FL	32543	\$2,851.00	\$5,000.00	35
725	Deerfield's Four Seasons	282 Columbia	Sheldon	FL	33553	\$248.00	\$7,500.00	35
842	All Season	28 Lakeview	Grove	FL	33321	\$8,221.00	\$7,500.00	20



Thank You

Database Systems

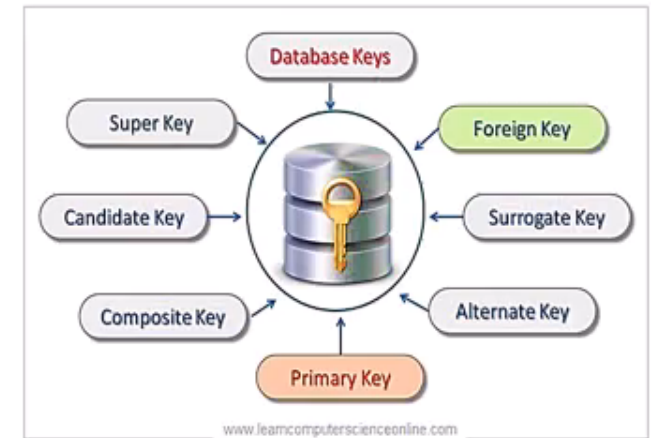
For
Third Stage

Lecture 14

Prepared By

Assistant Lecturer

Mays Adel Abdul-Ameen



Key Fields

➤ Candidate Key

- It holds the same properties of primary key, but it is not a primary key for the table, therefore its candidate to be a primary key.
- When starting to design the table, a number of fields are candidate to become primary keys, and upon entering data , A key that takes a Null value, may prove that these keys can take a Null value are Excluded and keys that do not take a Null value and not repeated (Unique) it choose a primary keys.



➤ Example

customer_id	customer_name	address	phone
01	Mohammed	Mosul	09567
02	Ali	Baghdad	07654
03	Saad	Baghdad	08654
04	Ali	Basrah	

Note

From the previous example that the customer_id and phone number are candidate keys to be primary key, but it turns out Later on, the phone number may be **Null**, so it will be **excluded** from being a primary key .

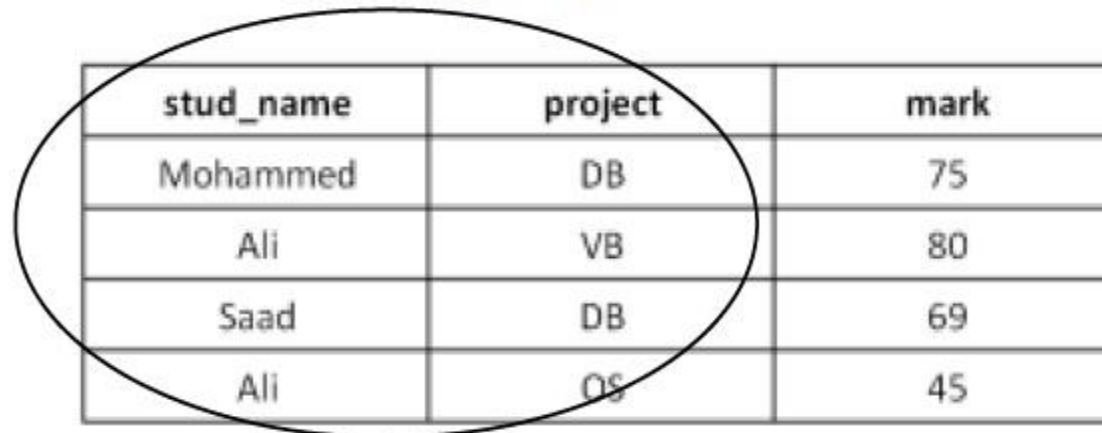


Key Fields

➤ Composite Key

- It is the key field that is used to identify a record uniquely, but differs from the primary key that includes more than one attribute.

➤ Example



stud_name	project	mark
Mohammed	DB	75
Ali	VB	80
Saad	DB	69
Ali	OS	45



- Notice in the previous example, that the student name, project name, or mark cannot be considered as a primary key.
- The record is determined exclusively and uniquely, and in this case, the student's name is considered a name with the project is a Composite Key , given that the student's name may be repeated and the project name may be repeated, but a student name with the project name as a Composite Key will not be duplicated.



Key Fields

➤ Super Key

It's a less number of attributes that can be distinguished a record in the table from the rest of the other records.

➤ Example

	St.no	St.name	dep	birth
	0001	ali	math	1980
	0002	ahmed	eng	1990
	0003	jasem	arabic	1991

Super key →

1.st.no →

2.st.name →

3.dep →





Thank You

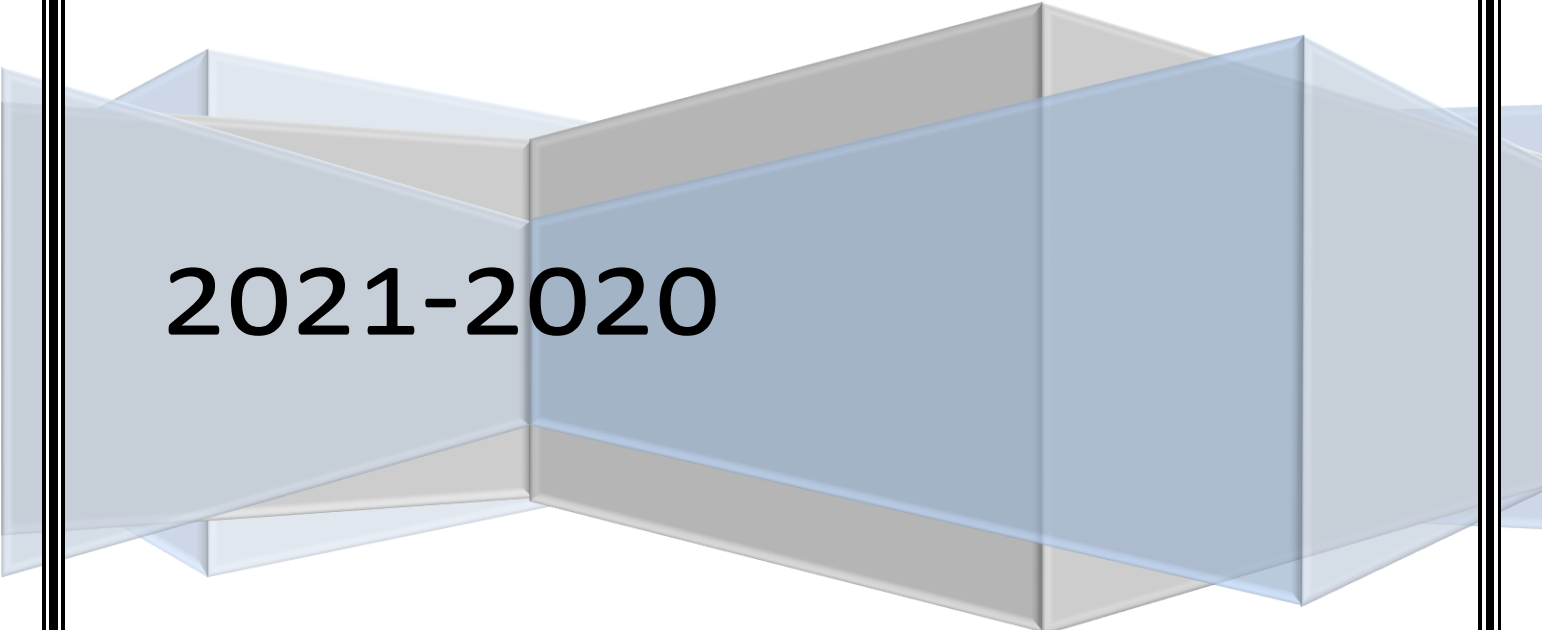
Mays A.

Al-Safwa University College

Third Stage

Database Systems

Lecture 1



2021-2020

1 Introduction to DB

1.1 Mini-world, Data and Database Definitions:

- ❖ **Mini-world** :Some part of the real world about which data is stored in a database .For example, student grades at a university.
- ❖ **Data** :Known facts that can be recorded and have an implicit meaning.
- ❖ **Database**: database is a collection of related data. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Excel. This collection of related data with an implicit meaning is a database. The preceding definition of database is quite general; for example, we may consider the collection of words that make up this page of text to be related data and hence to create a database. However, the common use of the term database is usually more restricted. A database has the following implicit properties:
 - A database represents some aspect of the real world, sometimes called the mini-world. Changes to the mini-world are reflected in the database.
 - A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.
 - A database is designed, built, and filled with data for a specific purpose. It has an intended group of users and some defined applications in which these users are interested.

1.2 Database Management System and Database systems

Definitions:

Database Management System (DBMS) :is a collection of programs that enables users to create and maintain a database. The DBMS is a general purpose software system that facilitates the processes of defining,

constructing, manipulating, and sharing databases among various users and applications.

1. Defining a database involves specifying the data types, structures, and constraints of the data to be stored in the database. The database definition or descriptive information is also stored by the DBMS in the form of a database catalogue or dictionary; it is called meta-data. The main functions provided by the DBMS include:
2. Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS.
3. Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the mini-world, and generating reports from the data.
4. Sharing a database allows multiple users and programs to access the database simultaneously.
5. Protecting the database and maintaining it over a long period of time. Protection includes system protection against hardware or software malfunction (or crashes) and security protection against unauthorized or malicious access. A typical large database may have a life cycle of many years, so the DBMS must be able to maintain the database system by allowing the system to evolve as requirements change over time.

Database System :The DBMS software together with application and the data itself .

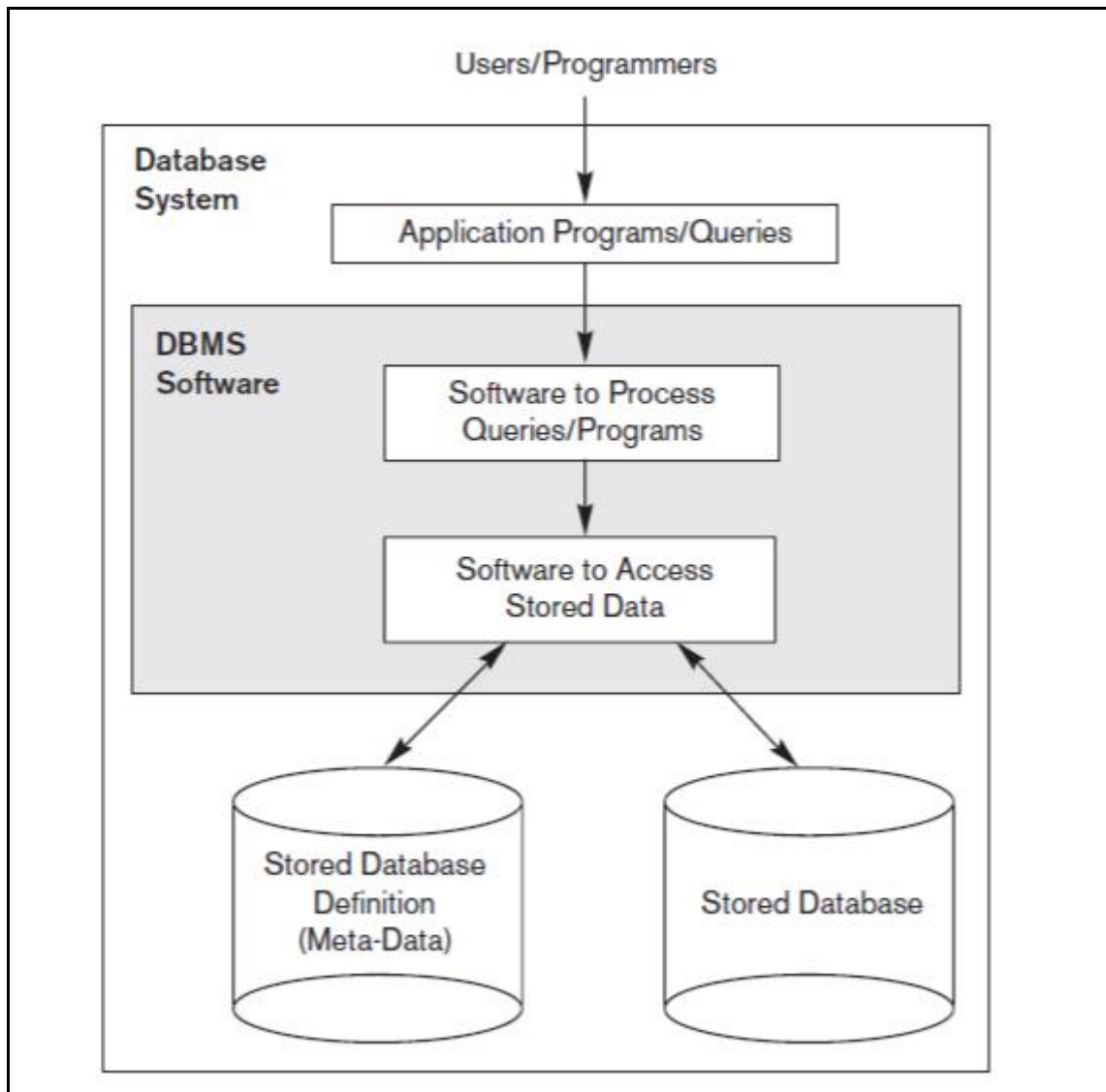


Figure 1-1 Database System

1.3 Example of a Database (with a Conceptual Data Model)

Mini-world for part of a UNIVERSITY environment.

Some of this mini-world entities:

- ❖ STUDENTs
- ❖ COURSEs
- ❖ DEPARTMENTs
- ❖ INSTRUCTORs

Some mini-world *relationships*:

- ❖ STUDENTs *take* COURSEs
- ❖ COURSEs *are offered by* DEPARTMENTs
- ❖ INSTRUCTORs *teach* COURSEs