University of Al-Qadisiyah
College of Computer Science and Information Technology
Information Systems Department

Fundamentals of Database Systems

Lecture 1

Introduction

2023

Lecture 1
Introduction

1.1 Introduction

Today, people use computers to perform many tasks formerly done with other tools. Computers have replaced typewriters for creating and modifying documents. They've surpassed electromechanical calculators as the best way to do math.

They've also replaced millions of pieces of paper, file folders, and file cabinets as the principal storage medium for important information. Compared to those old tools, of course, computers do much more, much faster — and with greater accuracy. These increased benefits do come at a cost, however. Computer users no longer have direct physical access to their data. When computers occasionally fail, office workers may wonder whether computerization really improved anything at all. In the old days, a manila file folder only "crashed" if you dropped it — then you merely knelt down, picked up the papers, and put them back in the folder. Barring earthquakes or other major disasters, file cabinets never "went down," and they never gave you an error message.

A hard drive crash is another matter entirely: You can't "pick up" lost bits and bytes. Mechanical, electrical, and human failures can make your data go away and never to return.

If you are storing important data, you have four main concerns:

1. Storing data needs to be quick and easy, because you're likely to do it often.
2. The storage medium must be reliable. You don't want to come back later and find some (or all) of your data missing.
3. Data retrieval needs to be quick and easy, regardless of how many items you store.
4. You need an easy way to separate the exact information that you want today from the tons of data that you don't want right now.

What is data?

** Data can be defined in many ways. Information science defines data an unprocessed information.

What is information?

** Information is data that have been organized and communicated in a coherent and meaningful manner.

- Data is converted into information, and information is converted into knowledge.

** **Knowledge** is information evaluated and organized so that it can be used purposefully as shown in figure (1.1)

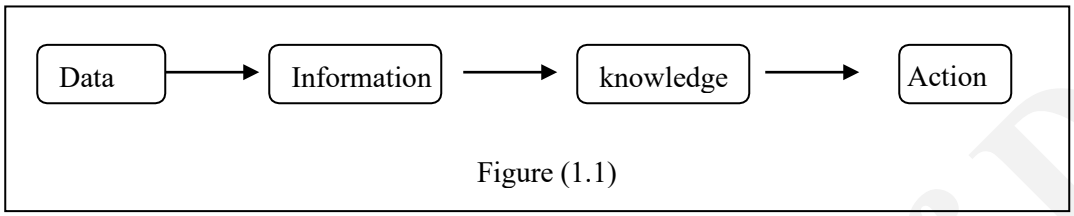


Figure (1.1)

1.2 What is a Data Base?

A **database** is an organized collection of data for one or more uses, typically in digital form. The data can be textual, like order or inventory data, or it can be pictures, programs or anything else that can be stored on a computer in binary form.

One way of classifying databases involves the type of their contents, for example: bibliographic, document-text, statistical.

The purpose of a database is to store and retrieve related information, so databases are designed to offer an organized mechanism for :

- Storing
- managing
- and retrieving information.

1.3 Files System:

The File is a block of arbitrary information, it is a place that application programs store their data in it. These application programs either database application or non-database application. Each file has a format. The information stored in the file can be organized in a record, which is acollection of fields.

The file system is typically described as various files and a number of different application programs are written to read from and add to the appropriate files.

File System Disadvantage:

- **Program dependence**: Each file has a format; the non-database application must know exactly the format of the file to deal with it. Any other application cannot access the file unless knowing the format of the file.
- **When file format updated**, then the application program must be updated, it is complicated to update all programs when data format is update.

- **Security problems existed**. Anyone can write a program to read the data in the file.
- **Data redundancy**, if there are application A deals with file A and application B deals with file B, if application A store an information in file A, and if application B need this information, application B cannot access file A , so application B must record the same information in file B.

Some basic Definitions:

- **Field**: one category of information (one data value), i.e., Name, Address, Semester Grade, Academic topic.
- **Record**: Collection of fields i.e., one student's information, a recipe, a test question. **File**: A group or collection of similar records, like student File.
- Digital databases are managed using **database management systems (DBMS**), which store database contents, allowing data creation and maintenance, and search and other access to the database.

1.4 What is DBMS ?

A **Database Management System** (**DBMS**) is a set of computer programs that controls the –

- Creation of the database
- The storing and organization of the data in the database
- Maintenance the database
- Searching, data retrieval and the use of a database.

The DBMS accepts requests for data from an application program and instructs the operatingsystem to transfer the appropriate data as shown in figure (1.2)

ADVANTAGES OF A DBMS

1. **Database Development**: It allows organizations to place control of database development in the hands of database administrators (DBAs) and other specialists.
2. **Data independence**: Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.
3. **Efficient data access**: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. It allows different user application programs to easily access the same database. Instead of having to write computer programs to extract information, user can ask simple questions in

a query language.

4. **Data integrity and security**: If data is always accessed through the DBMS, the DBMS can enforce:
   - integrity constraints on the data. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded.
   - Also, the DBMS can enforce access controls that govern what data is visible to different classes of users.
5. **Crash recovery**: the DBMS protects users from the effects of system failures.
6. **Data administration and Concurrent access**: When several users share the data (more than one user access the database at the same time), DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time.

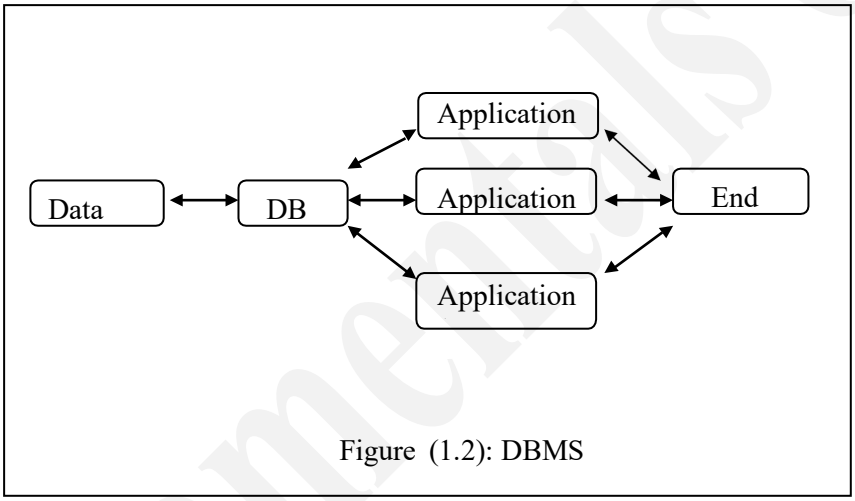


Figure (1.2): DBMS

1.5 Data Abstraction

The major purpose of a database system is to provide users with an abstract view of the data.

That is, the system hides certain details of how the data are stored and maintained.

For the system to be usable, it must retrieve data efficiently. This had led to the design of complex data structure to represent the data in the data base. Since many database-user are not computer trained, data base developers hide the complexity from users through several levels of abstraction as shown in figure (1.3).

** **Physical level:** the lowest level of abstraction, describe ***how*** the data are actually stored. Atthe Physical level , complex low level data structure are described in detail. In this level storage locations used to specify where the data are and the data been

described by words and bytes.

** **Logical level:** This level describe ***what*** data are stored in the database and what relationshipexist among those data. The entire database is describe in terms of small number of relatively simple structure. The logical level is used by the database administrator, who must decide what information is to be kept in the database.

** **View level:** The highest level of abstraction describes only part of the entire database. Many users of the database system will not be concern with all the information. Instead the user needto access only part of the database, so a view level of abstraction is defined .

The system may provide many views for the same database.

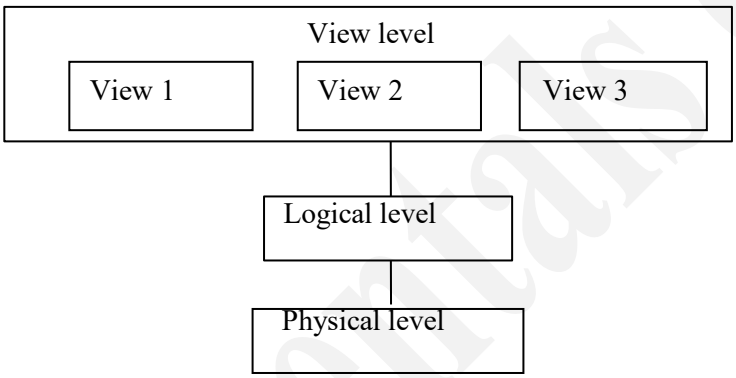


Figure (1.3)

The three level of abstraction

University of Al-Qadisiyah
College of Computer Science and Information Technology
Information Systems Department

Fundamentals of Database Systems

Lecture 2

Introduction

2023

Lecture 2
Database Models

2.1 Database Models

A data model is a collection of tools that describes how data are represented and accessed. Data models formally define data elements and relationships among data elements for a domain of interest (data element is an unit of data that has precise meaning or precise semantics)

A data model explicitly determines the structure of data . The main aim of data models is to support the development of information systems by providing the definition and format of data.
If the same data structures are used to store and access data then different applications can share data.

Communication and precision are the two key benefits that make a data model important to applications that use and exchange data. A data model is the medium which project team members from different backgrounds and with different levels of experience can communicate with one another. (Precision means that the terms and rules on a data model can be interpreted only one way and are not ambiguous).

There are many deferent models:

1- The Entity Relationship model
2- The Relational data model.
3- Object Based data model
4- The Network data model.
5- The Hierarchical data model.

2.1.1 : The Entity Relationship model

The entity-relationship model (or ER model) is a way of graphically representing the logical relationships of entities (or objects) in order to create a database.

In ER modeling, the structure for a database is portrayed as a diagram, called an entity-relationship diagram (or ER diagram),

The ER model consists of the following:

- **Entity**: a thing that exists and which can be uniquely identified
  e.g. person, automobile, department, employee
- **Entity Set**: a group of similar entities
  e.g. all persons, all automobiles, all employees
- **Relationship**: association between entities
  e.g. a person is assigned to a department
- **Relationship Set** : set of similar relationships
- **Attribute**: property of an entity or relationship
  e.g. person - name, address
- **Domain**: set of values allowed for an attribute

An entity may be a physical object such as a house or a car, an event such as a house sale or a car service, or a concept such as a customer transaction or order

A relationship captures how two or more entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns.

Examples: an owns relationship between a company and a computer.

- a **supervises** relationship between an employee and a department.
- a **performs** relationship between an artist and a song.
- a **proved** relationship between a mathematician and a theorem.

Entity sets are drawn as rectangles, attributes are drawn as oval. Entity is connected with an attribute with lines. Diamonds represent relationship among entity set.

Figure 2.1 shows an ER diagram notation for an attribute (Grade) of an entity (student)

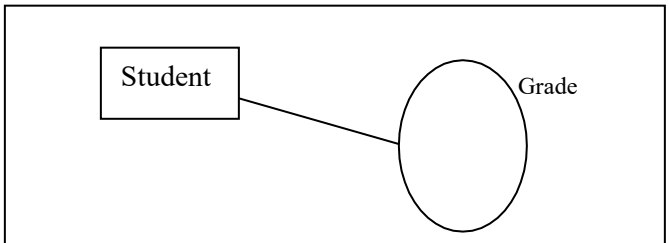


Figure 2.1 ER example 1

Figure 2.2 shows the following example :
There are three entity:

1- **Employees** : E#,ENAME, ADDRESS
2- **Departments** : D#,DNAME
3- **Projects** : PNAME

There are two relations that connect the three entities

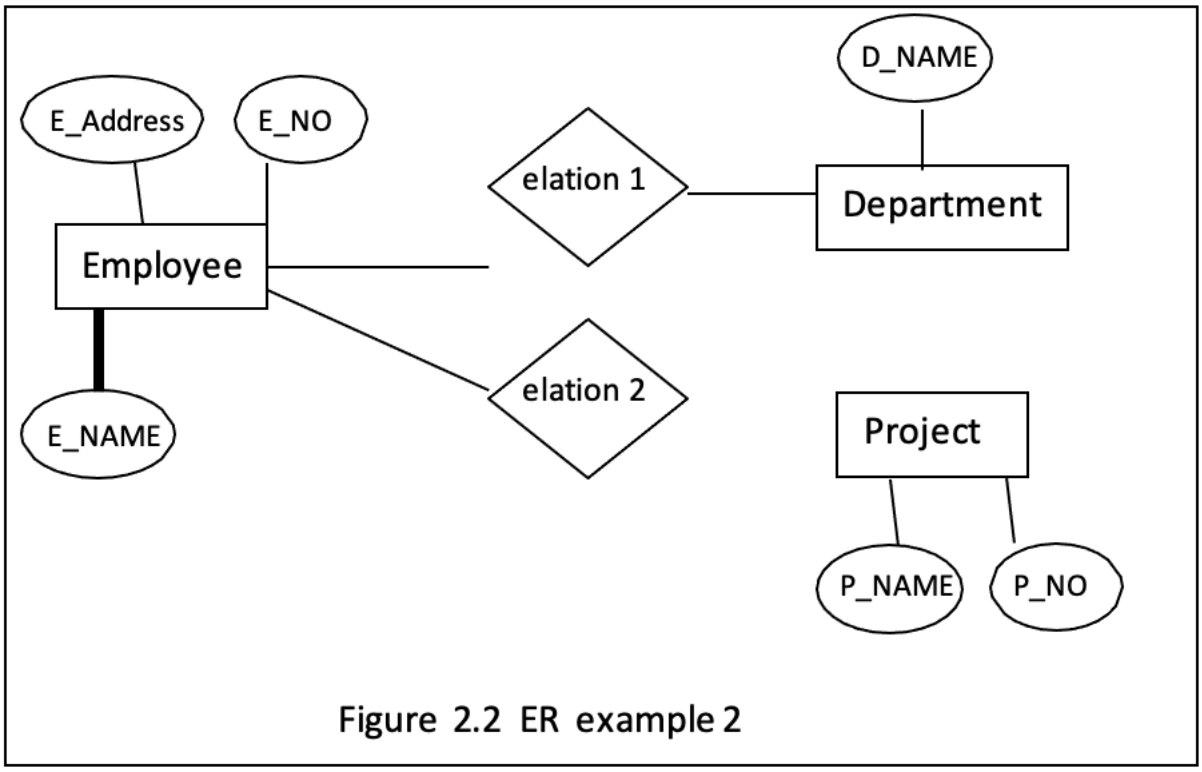


Figure 2.2 ER example 2

2.1.2 The Relational model

The Relational Model is a clean and simple model that uses the concept of a relation using a table rather than a graph or shapes. The information is put into a grid like structure that consists of columns running up and down and rows that run from left to right, this is where information can be categorized and sorted.

The relational model used the basic concept of a relation or table. The columns or fields in the table identify the attributes such as name, age, and so. A tuple or row contains all the data of a single instance of the table such as a person named Doug. In the relational model, every tuple must have a unique identification or key based on the data as shown in figure 2.3 , a social security account number (SSAN) is the key that uniquely identifies each tuple in the relation.

Often, keys are used to join data from two or more relations based on matching identification.

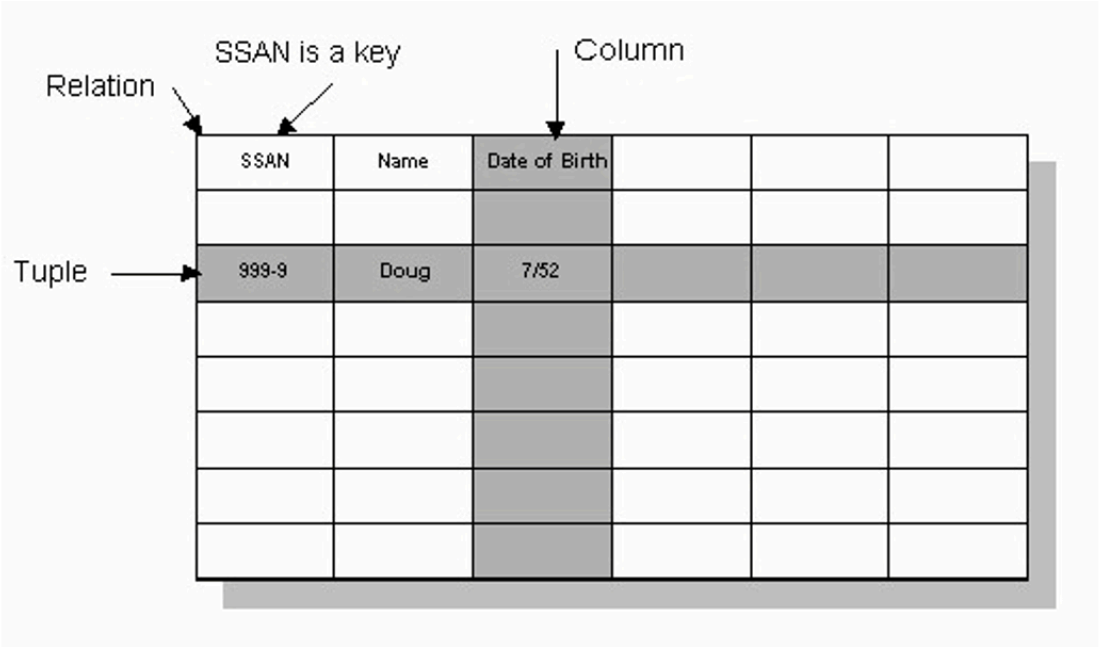


Figure 2.3 Relation (Table)

2.1.3 Object Based data model

An object database (also object-oriented database) is a database model in which information is represented in the form of objects as used in object-oriented programming

Today's trend in programming languages is to utilize objects, thereby making OODBMS ideal for OO programmers because they can develop the product, store them as objects, and can replicate or modify existing objects to make new objects within the OODBMS

2.1.4 Hierarchical and Network data model

A hierarchical database model is a data model in which the data is organized into a tree-like structure. The structure allows representing information using parent/child relationships: each parent can have many children, but each child has only one parent (also known as a 1-to-many relationship). All attributes of a specific record are listed under an entity type. Figure 2.4 shows as example.

This model is recognized as the first database model created by IBM in the 1960s.

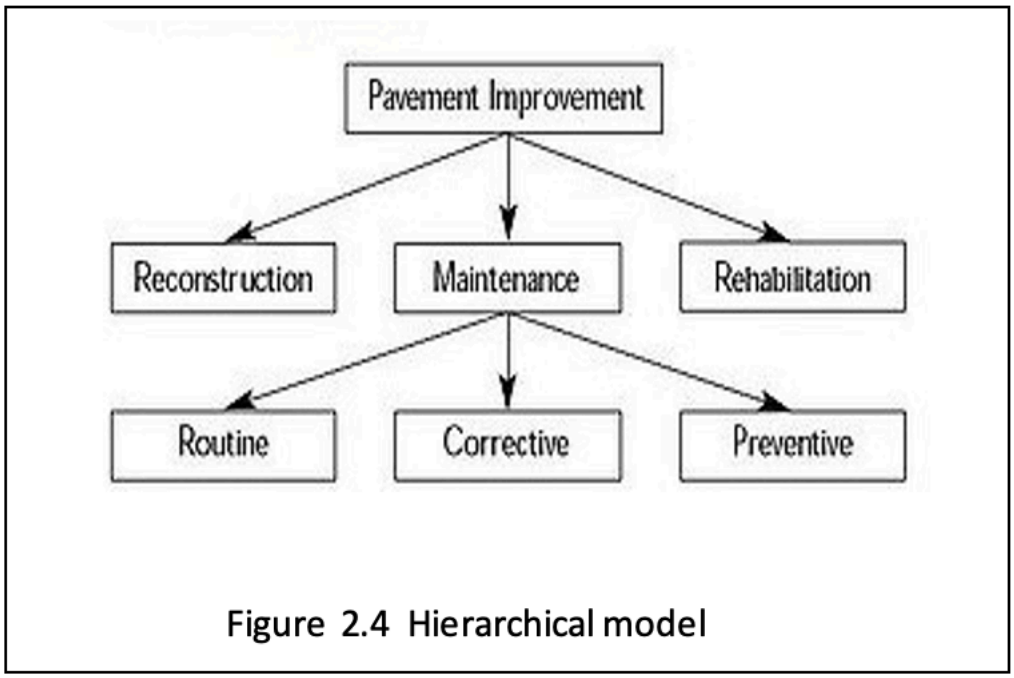
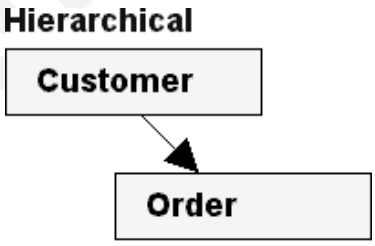


Figure 2.4 Hierarchical model

In the following example, orders are owned by only one customer. This model often restrictive in linking real-world structures.

In network databases, data is organized as a graph ,a record type can have multiple owners. In the hierarchical model, each record having one parent record and many children, the network model allows each record to have multiple parent and child records, forming a generalized graph structure.

In the example below, orders are owned by both customers and products, reflecting their natural relationship in business.

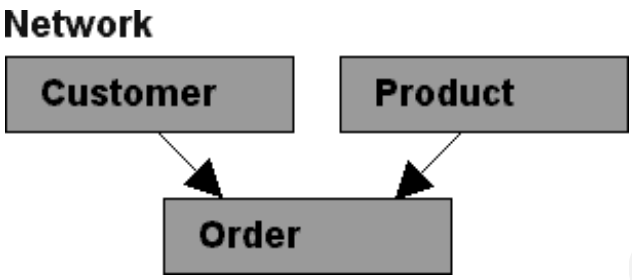


Relational databases do not link records together physically, but the design of the records must provide a common field, such as account number, to allow for matching. Often, the fields used for matching are indexed in order to speed up the process.

In the following example, customers, orders and products are linked by comparing data fields and/or indexes when information from more than one record type is needed. This method is more flexible for ad hoc inquiries.

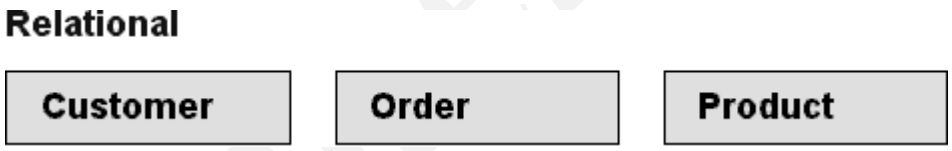
In ***network databases***, data is organized as a graph ,a record type can have multiple owners. In the hierarchical model, each record having one parent record and many children, the network model allows each record to have multiple parent and child records, forming a generalized graph structure.

In the example below, orders are owned by both customers and products, reflecting their natural relationship in business.



Relational databases do not link records together physically, but the design of the records must provide a common field, such as account number, to allow for matching. Often, the fields used for matching are indexed in order to speed up the process.

In the following example, customers, orders and products are linked by comparing data fields and/or indexes when information from more than one record type is needed. This method is more flexible for ad hoc inquiries.



University of Al-Qadisiyah
College of Computer Science and Information Technology
Information Systems Department

Fundamentals of Database Systems

Lecture 3

Introduction

2023

Lecture 3
Relational Database

3. Relational Database

A relational database consists of a collection of tables, each of which assigned a unique name.

The table consists of a number of rows. A row in a table represents a relation among a set of values.

A relational database is a collection of data items organized as a set of tables from which data can be accessed in many different ways without having to reorganize the database tables. The relational database was invented by E. F. Codd at IBM in 1970.

A relational database is a set of tables containing data fitted into predefined categories. Each table (which is sometimes called a relation) contains one or more data categories in columns. Each row contains a unique instance of data for the categories defined by the columns.

Consider the account table as shown in figure 3.1 . It has three column header (three fields) :

- Branch-name
- Account-number
- Balance

Branch-name	Account-umber	Balance
A	A-101	500
B	A-215	234
C	A-234	800

Figure 3.1 : Account table

3.1 Instance and schemas

Database changes over time as information is inserted and deleted.

Instance: is the collection of information stored in the database at a particular moment is called an instance of the database.

Schema: the overall design of the database is called the database schema.

Schemas are changed infrequently.

The database schema is the logical design of the database, database instance is a snapshot of the data in the database at a given instance of time.

Schemas, like tables and fields, have names. For example the schema that describe the table in figure 3.1 could be:

Account table-Schema= (Branch-name, Account-number , Balance)

3.2 Keys

As you may already know, databases use tables to organize information. Each table consists of a number of rows, each of which corresponds to a single database record. So, how do databases keep all of these records straight? It's through the use of keys.

Primary Keys

The first type of key we'll discuss is the primary key. **Every database table should have one or more columns designated as the primary key**. The value this key holds should be unique for each record in the database. For example, assume we have a table called Employees that contains personnel information for every employee in our firm. We'd need to select an appropriate primary key that would uniquely identify each employee. Your first thought might be to use the employee's name.

This wouldn't work out very well because it's conceivable that you'd hire two employees with the same name. A better choice might be to use a unique employee ID number that you assign to each employee when they're hired

Once you decide upon a primary key and set it up in the database, **the database management system (DBMS)** will enforce the uniqueness of the key. If you try to insert a record into a table with a primary key that duplicates an existing record, the insert will fail.

Foreign Keys

The other type of key that we'll discuss in this course is the foreign key. These keys are used to create relationships between tables. Natural relationships exist between tables in most database structures

The foreign key link is set up by matching columns in one table (the child) to the primary key columns in another table (the parent).

Example 1: In the example of figure 3.2, there is a link between the Company and Contact tables. The Company table is the parent table in the link. The Contact table is the child: the Company ID field in the Contact table indicates which Company a Contact belongs to.

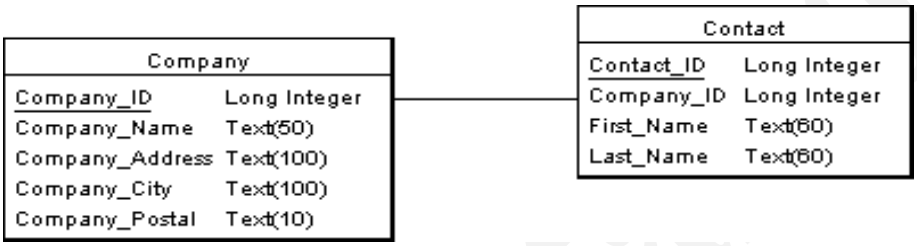


Figure 3.2 Example 1 of a relation between two tables

Example 2: suppose there is a need to design a database to hold the history of each student of a school. The information of each student should have :

- Student name
- Student phone
- Information about the student in each year in the school.

This database should have at least two tables , the first one like the following schema:

St Table-Schema=(st-name, st-phone)

The second table contain the information about the history of the student :

History-schema=(year , level , state)

To link those two tables a primary key in the first table and a foreign key in the second table must be added to the tables as shown in figure 3.3 .

The new schemas will be:

St Table-Schema=(S-ID , st-name, st-phone)

History-schema =(S-ID , year , level , state)

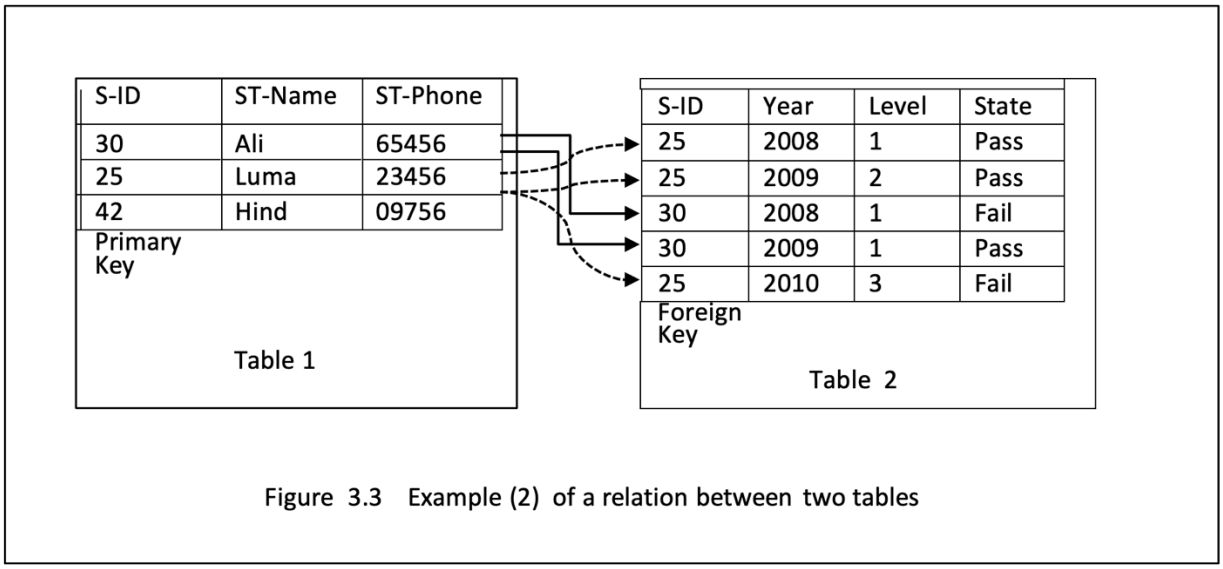


Figure 3.3 Example (2) of a relation between two tables

3.3 Structured Query Language (SQL)

SQL (Structured Query Language) is a database computer language designed for managing data in relational database management systems (RDBMS), and originally based upon Relational Algebra.

SQL designed to organize and simplify the process of getting information out of a database in a usable form, and also used to reorganize data within databases.

It is a set of statements to manage databases, tables, and data.

Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server and Microsoft Access.

SQL falls into two classes

1. **Data Definition Language (DDL)** - SQL for creating, altering and dropping tables
2. **Data Manipulation Language (DML)** - SQL for retrieving and storing data.

Data Definition Language (DDL) : a database schema is specified by a set of definitions expressed by the a language called (DDL).

It is the subset of SQL used for defining and examining the structure of a database. It used to define databases and their components.

Data Manipulation Language (DML): The data in the database are manipulated by the (DML) . With (DML) we can

- Retrieve information from the database
- Inserting new information into the database
- Delete information from the database
- Updating the data in the database

3.4 Table Joining

A JOIN is a means for combining fields from two tables by using values common to each.

Table Joining is a formal specification of which column(s) in a row in one table should be matched to a column(s) in a row in another table.

To combine tables, i.e. to perform a join, we have to select data from the tables, and relate the tables to each other with conditions on some attributes (often keys attributes).

Joining tables is used when retrieving information from the database is required.

In SQL the word (SELECT) is used to display information from the database followed by the name of the tables.

to explain the types of joins in the database, the following three tables are used as an example:

```
teachers table

 id |  name
----+--------------------
  1 | Volker
  2 | Elke
(2 rows)
```

Projects table

id	name	duration	teacher
1	compiler	180	1
2	xpaint	120	1
3	game	250	2
4	Perl	80	4

(4 rows)

Assign table

project	stud	percentage
1	2	10
1	4	60
1	1	30
2	1	50
2	4	50
3	2	70
3	4	30

(7 rows)

3.4.1 Cross Join:

Each row from table one is arbitrarily combine with each row from table two.

This is known as a Cartesian product. In practical terms a cross join is a join without a join condition

Each row in teachers is arbitrarily combined with each row in projects.:

Example: Cross join of tables teachers and project will be

```
> SELECT * FROM teachers, projects;
```

id	name	id	name	duration	teacher
1	Volker	1	compiler	180	1
1	Volker	2	xpaint	120	1
1	Volker	3	game	250	2
1	Volker	4	Perl	80	4
2	Elke	1	compiler	180	1
2	Elke	2	xpaint	120	1
2	Elke	3	game	250	2
2	Elke	4	Perl	80	4

(8 rows)

3.4.2 Inner Join:

Inner joins are the most common type of joins, they combine two or more tables using where clause as main condition with equality "=" sign or inequality "!=" .

A field from the first table is compared to another field from the second table and if they are equal (or not equal) the two records from the two tables are combined.

Example : inner join of tables teachers and project if the condition is

teachers.id = projects.id will be

SELECT * FROM teachers, projects ***where*** teachers.id =projects.id;

```
id |  name   | id |   name    | duration | teacher
---+---------+----+-----------+----------+---------
  1 | Volker |  1 | compiler |      180 |       1
  2 | Elke   |  2 | xpaint   |      120 |       1
```

Example : inner join of tables teachers and project if the condition is

teachers.id ! = projects.id will be

SELECT * FROM teachers, projects where teachers.id != projects.id;

```
id |  name   | id |   name    | duration | teacher
---+---------+----+-----------+----------+---------
  1 | Volker |  2 | xpaint   |      180 |       1
  1 | Volker |  3 | game     |      180 |       1
  1 | Volker |  4 | Perl     |      180 |       1
  2 | Elke   |  1 | compiler |      120 |       1
  2 | Elke   |  3 | game     |      120 |       1
  2 | Elke   |  4 | Perl     |      120 |       1
```

There is another format for inner join

select T1.F1, F2,F3,…,FN from T1 inner join T2 on (T1.F1 = T2.F1) inner join T3 on (T2.F1=T3.F1) inner join ……

3.4.3 LEFT OUTER JOIN:

this type of join will first select all shared records from those tables, and then will bring all records from left table which are not exist in the second table.

3.4.4 RIGHT OUTER JOIN

this type of join will first select all shared records from those tables, and then will bring all records from right table which are not exist in the first table

3.4.5 FULL OUTER JOIN

this type of join will first select all shared records from those tables, and then will bring all records exist in first table which are not exist in the second table and then bring all records exist in the second table which are not exist in first table.

select teachers.id , teachers.name , projects.id , projects.name from teachers full OUTER join projects on (teachers.id = projects.id);

```
id |  name   | id |    name
---------+-----------------+---------+---------------------
  1 | Volker |  1 | compiler
  2 | Elke   |  2 | xpaint
                4 | Perl
                3 | games
```

University of Al-Qadisiyah
College of Computer Science and Information Technology
Information Systems Department

Fundamentals of Database Systems

Lecture 4

Database Design

2023

Lecture 4
Database Design

4: Database Administrator

A database administrator (DBA) is a person responsible for:

- Design and Implementation of the database
- maintenance and repair of the database
- monitoring and improving database performance and capacity
- planning for future expansion requirements.

A basic responsibility for just about every database administrator involves the installation of new databases. As part of the database installation, the database administrator will set up login credentials to authorized persons, define the privileges associated with each authorized user, and ensure that every work station attached to the network is set up to access the new database.

5: Database Design

Database design is the process of producing a detailed data model of a database , this data model which can then be used to create a database.

The term database design can be used to describe many different parts of the design , it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views.

5.1: The Design Process

The design process consists of the following steps:

1. Determine the purpose of your database - This helps prepare you for the remaining steps.
2. Find and organize the information required - Gather all of the types of information you might want to record in the database, such as product name and order number.
3. Divide the information into tables - Divide your information items into major

entities or subjects, such as Products or Orders. Each subject then becomes a table.

4. Turn information items into columns - Decide what information you want to store in each table. Each item becomes a field, and is displayed as a column in the table. For example, an Employees table might include fields such as Last Name and Hire Date.

5. Specify primary keys - Choose each table's primary key. The primary key is a column that is used to uniquely identify each row. An example might be Product ID or Order ID.

6. Set up the table relationships - Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.

7. Refine your design - Analyze your design for errors. Create the tables and add a few records of sample data. See if you can get the results you want from your tables. Make adjustments to the design, as needed.

8. Apply the normalization rules - Apply the data normalization rules to see if your tables are structured correctly. Make adjustments to the tables, as needed.

5.2: Database Cardinality

In data modeling, the cardinality of one data table with respect to another data table is a critical aspect of database design. Relationships between data tables define cardinality when explaining how each table links to another.

In the relational model, tables can be related as any of:

- **many-to-many**
- **many-to-one (**rev. **one-to-many)**
- **one-to-one**

This is said to be the **cardinality** of a given table in relation to another.

For example, considering a database designed to keep track of hospital records. Such a database could have many tables like:

- a Doctor table full of doctor information
- a Patient table with patient information
- and a Department table with an entry for each department of the hospital.

In that model:

- There is **a many-to-many** relationship between the records in the doctor table and records in the patient table (Doctors have many patients, and a patient could have several doctors);
- a **one-to-many** relation between the department table and the doctor table (each doctor works for one department, but one department could have many doctors).
- **one-to-one** relationship is mostly used to split a table in two in order to optimize access or limit the visibility of some information. In the hospital example, such a relationship could be used to keep apart doctor's personal or administrative information.

Example 1: A chemical factory producing chemical materials, each material identified by a name and a formula .

The supplier, identified by his name and his ID, purchase from the factory by an order. The order has date, amount and total.

To draw the ER model

Each supplier can **have** any material, and any material can go to any supplier. The relation is of type **many-to-many**, as shown in figure 5.1.

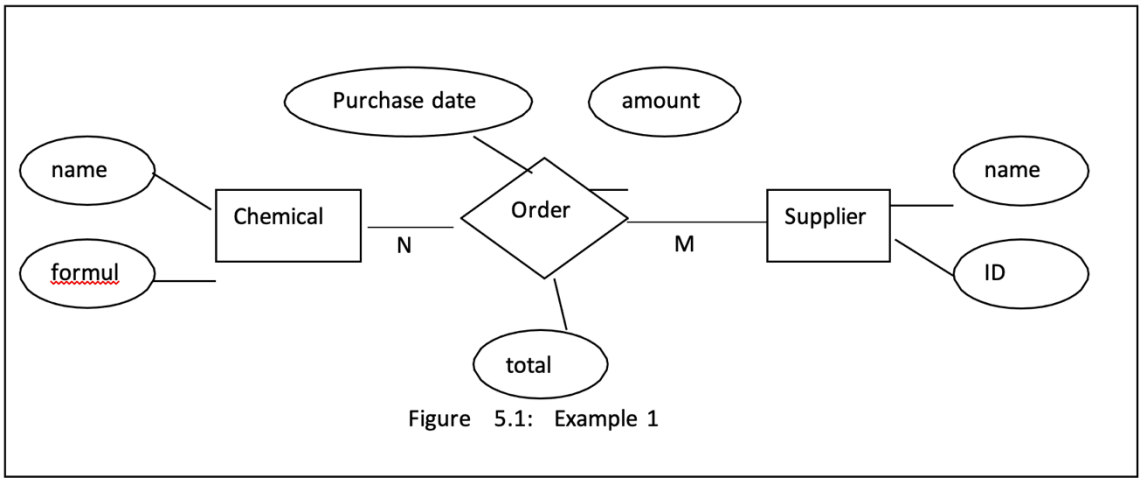


Figure 5.1: Example 1

5.3: Weak Entity

An entity may not have sufficient attributes to form a primary key. Such an entity is termed a **weak entity**. An entity that has a primary key is termed a **strong entity**.

Consider the relationship Have which links a student with his activates as shown in figure (5.2).

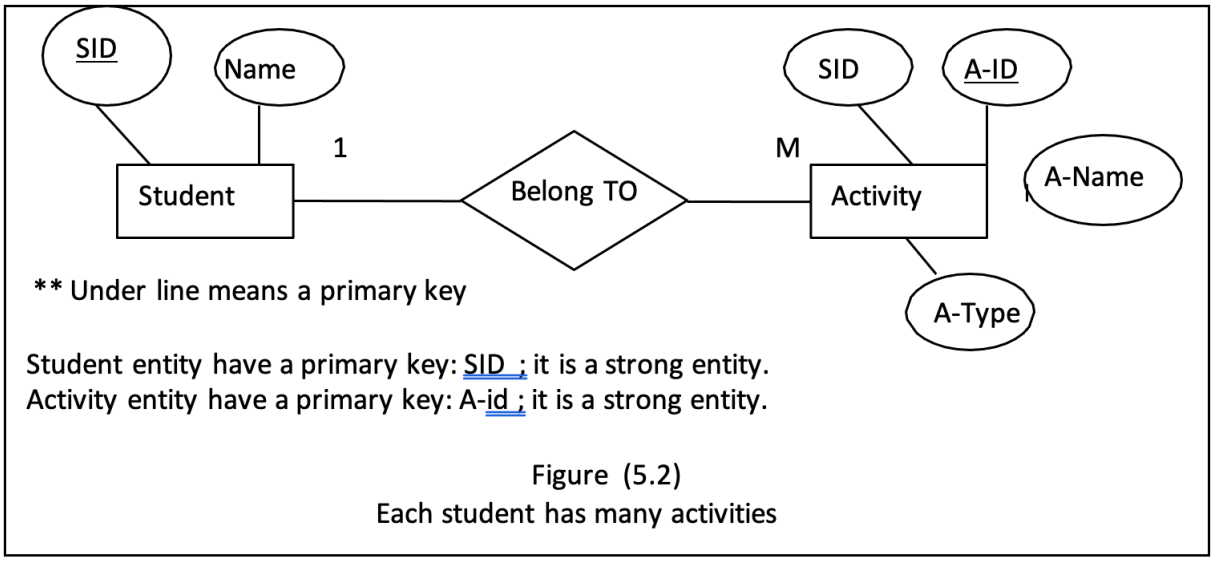


Figure (5.2)
Each student has many activities

If more details is needed about all the activities the student have (history) , another entity will be added as shown if figure (5.3).

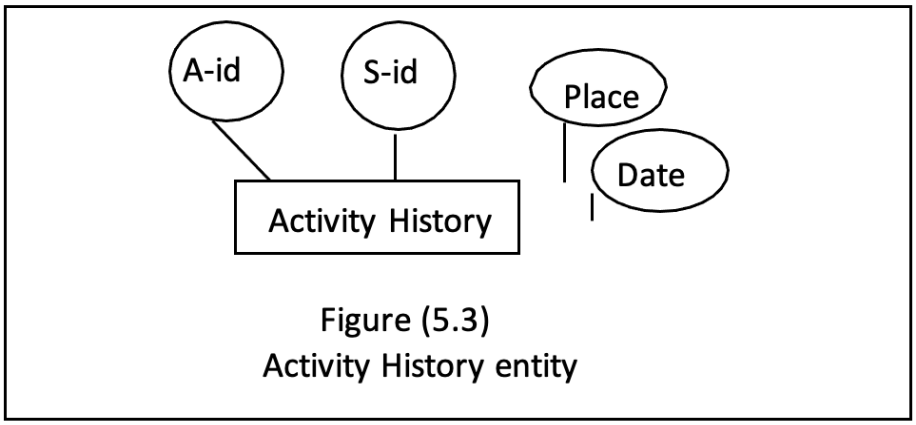


Figure (5.3)
Activity History entity

This new entity has S-ID and A-ID as a foreign key to recognize each record belong to what student and what activity. This means for each activity a student have, he/she can carry it many times.

Figure (5.4) shows how the new entity, Activity History , is linked with activity entity

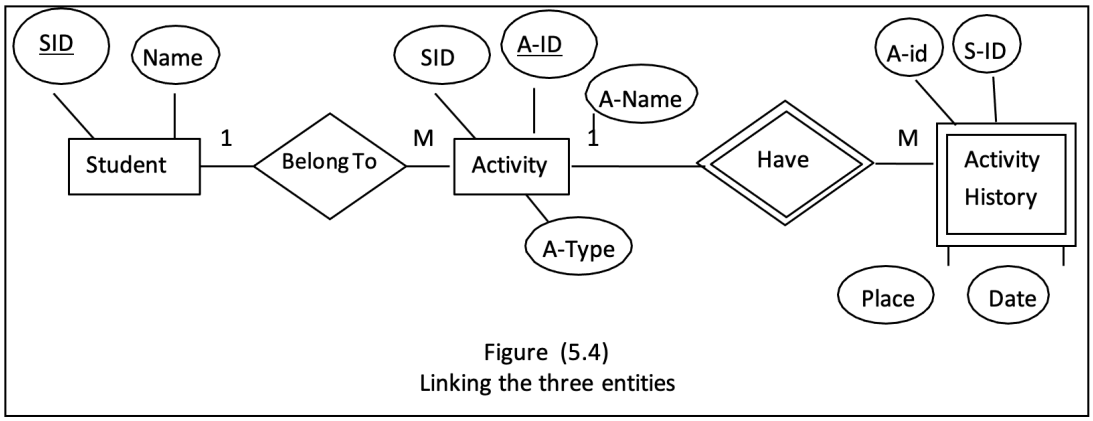


Figure (5.4)
Linking the three entities

In the Activity History, there is no primary key, so it is a weak entity.

The weak entity to be meaningful, it must be part of a one-to-many relationship. The weak entity is depending on the strong entity, that means if the strong entity is delete then the weak entity must be deleted.

In the example of figure (5.4) if the activity is deleted then all the history for that activity must be deleted.

But if the history is deleted, the activity no need to be deleted.

The DOMINANT entity is the strong entity. The SUBORDINATE entity is the weak entity.

The weak entity in the ER diagram is indicated by a double outline box, and the corresponding relationship by a double outline diamond.

5.4: More Designs Considerations

In this section, we consider how a database designer may select from the wide range of alternatives. Among the decision to be made are the following:

- weather to use an attribute or an entity to represent an object: consider the entity **employee** with attribute **employee-name**.

if we want to add telephone number to the employee:

case 1: another attribute , **telephone-number** , is added to the entity.

Case2: the **telephone** can be consider as an entity in its own with an attribute:

telephone-number and **location**, ant the two entities are connected with some relation.

In the first case every employee has one telephone number.

In the second case, each employee has several telephone numbers

- weather a real-world concept is expressed by an entity or by a relationship. For example a playing can be modeled as an entity, or it can be a relation between player and game, with playing-time attribute.

If every game is played by one player, then playing is better be a relationship as in figure (5.5)

If several players playing the same game, then we must replicate the playing information (time) for each player, and the problems of repetition and updating are arise.

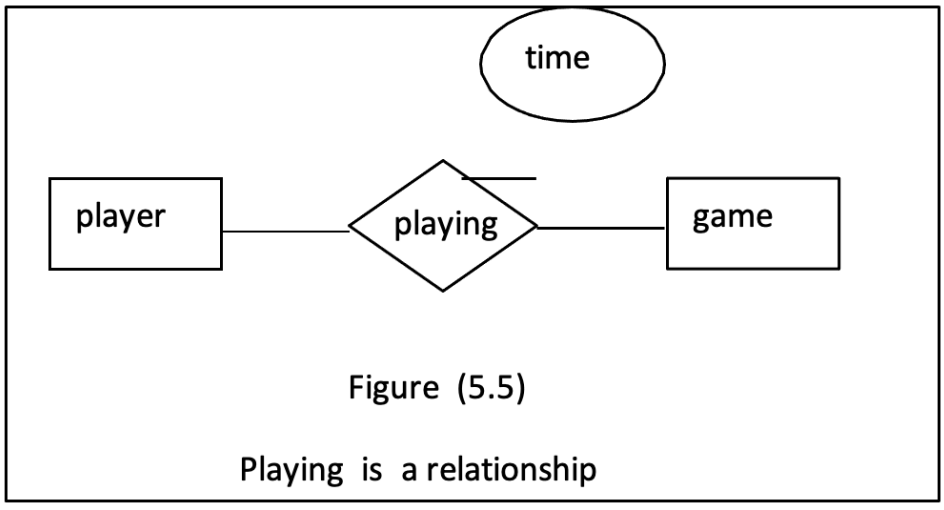


Figure (5.5)

Playing is a relationship

If the information of the playing is updated then this update must be done to each player.

Playing can be an entity rather than an relation. In this case there will be only one existence of the playing information, and the player and the games is connected with the

playing by keys as shown in figure (5.6)

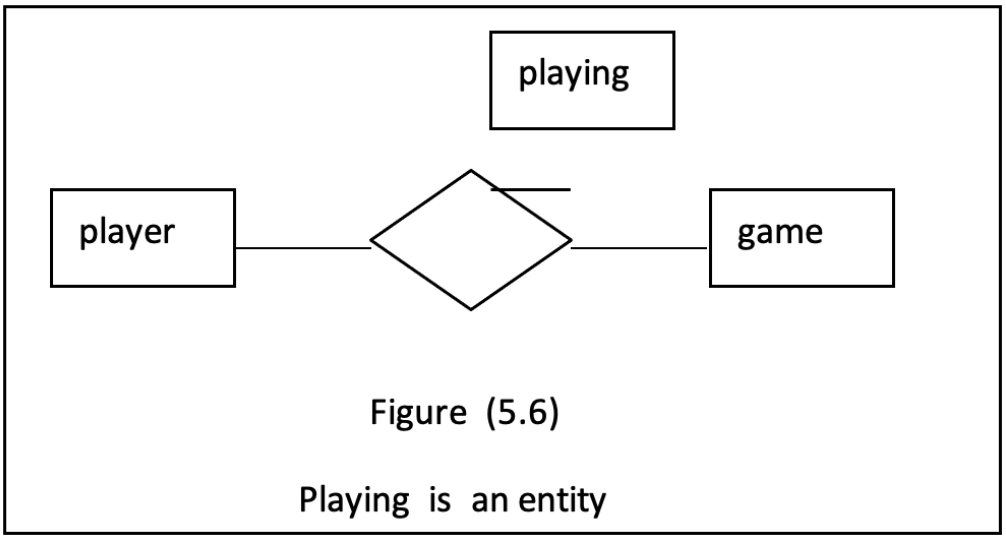


Figure (5.6)

Playing is an entity