

DATABASES - LAB

2020-2021

الدراسات الأولية / البكالوريوس / الفصل الاول

الكادر التدريسي

م.د. مهدي دعيمي كزار

م.د. سرمد مكي محمد غريب

م.م. فراس محمد كاظم

ر. مبرمجين اقدم بيداء سلمان قنديل

2020-
2021

Databases

Third Class – First Semester – (Practical)
Morning & Evening - Undergraduate Studies

قواعد البيانات

المرحلة الثالثة – الفصل الدراسي الاول – (العملي)
الدراسات الاولى – الدراسة الصباحية & المسائية

Assist. Lecturer: Firas M. Kadhum

fmk@sc.uobaghdad.edu.iq

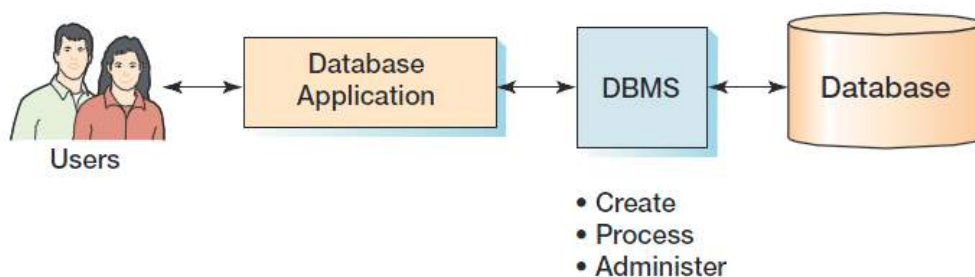
Google-Class Code : f7g25cu

www.sites.google.com/site/firasmkadhum

Brief Introduction to Database Concepts

- The Components of a Database System:

- ❖ As in Figures below, the **Database** is a collection of related tables and other structures. It can be regarded as an electronic filing cabinet.

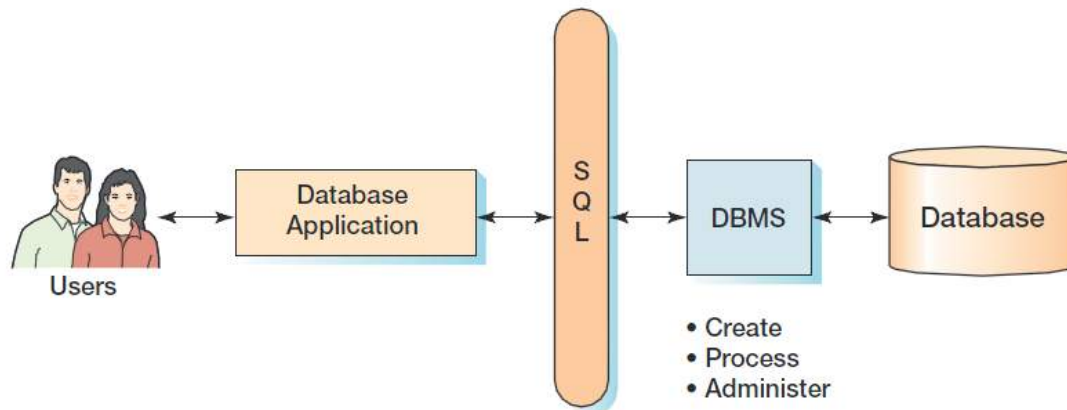


- ❖ In a computerized database, data appears in a table that looks very similar to spreadsheets. The column headings are called **field names**, and the columns are called **fields**. The rows of data are **records**.

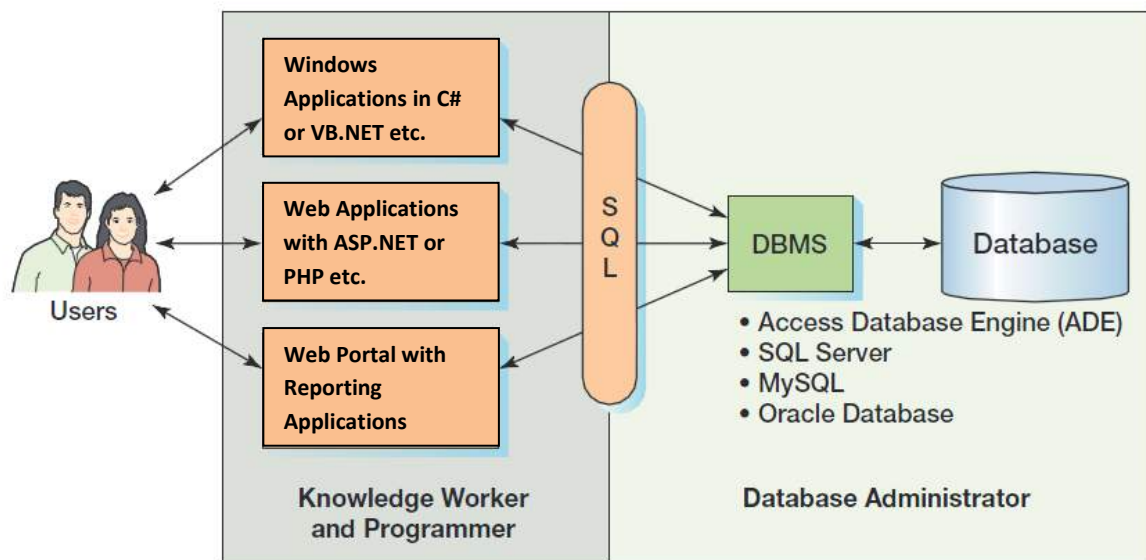
EmpNum	EmpName	DeptNum	DeptName
100	Jones	10	Accounting
150	Lau	20	Marketing
200	McCauley	10	Accounting
300	Griffin	10	Accounting

- ❖ The **Database Management System (DBMS)** is a computer program used to create, process, and administer the database.

- ❖ The **DBMS** receives requests encoded in SQL (**S**tructured **Q**uery **L**anguage) and translates those requests into actions on the database, as you can see in the figure.



- ❖ A **Database Application** is a set of one or more computer programs that serves as an intermediary between the **User(s)** and the **DBMS**.



- ❖ Application programs read or modify database data by sending SQL statements to the **DBMS**. Application programs also present data to users in the format of *forms* and *reports*.
- ❖ **Users**, the fourth component of a database system, employ a database application to keep track of things. They use *forms* to read, enter, and query data, and they produce *reports* to convey information.

- The Two Types of Database Management Systems:

- 1) **File Management System** – Sometimes called **Flat File Database** – this file system stores data in files without indexing. File management systems lack flexibility in data manipulation.
- 2) **Relational Database Management System (RDBMS)** – enables users to manipulate data in more sophisticated ways. **RDBMS** avoids redundancy in data and defines the relationships between sets of data. The **relationship** is a common element between tables such as **DeptNum**. The data stored in each table can be retrieved and updated based on data in another table.

EmpNum	EmpName	DeptNum	DeptName
100	Jones	10	Accounting
150	Lau	20	Marketing
200	McCauley	10	Accounting
300	Griffin	10	Accounting

(a) One-Table Design

OR?

DeptNum	DeptName
10	Accounting
20	Marketing

EmpNum	EmpName	DeptNum
100	Jones	10
150	Lau	20
200	McCauley	10
300	Griffin	10

(b) Two-Table Design

Brief Introduction to Microsoft Office Access 2010

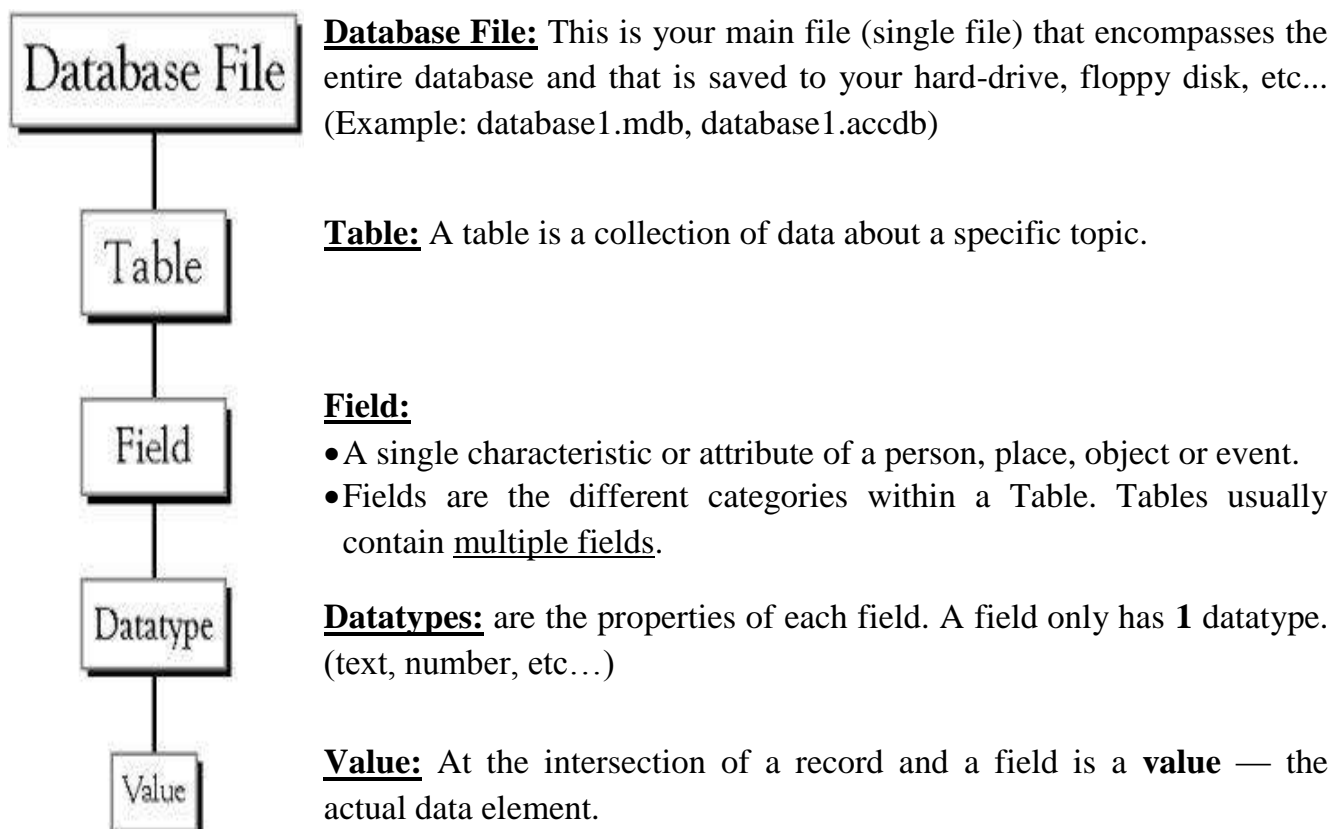
- What is Microsoft Office Access?

- ❖ **Microsoft Access**, also known as **MS Office Access**, is a Database Management System (**DBMS**) from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools.
- ❖ It is a member of the **Microsoft Office** suite of applications.

- Other examples of DBMS applications include:

- Oracle Database,
- SQL Server,
- MySQL,
- FoxPro etc.

Now, we need to understand how **MS Access** breaks down a database. Some keywords involved in this process are: **Database File**, **Table**, **Field**, **Data-type** and **Value**.

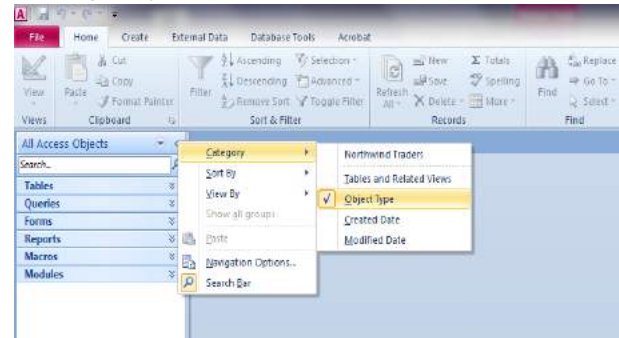


- Main MS Office Access Objects:

In MS Office Access, a database file consists of the following objects:

1- Tables:

- ☐ Basic container for data, arranged as a grid of **rows** and **columns** (i.e. stores all data).
- ☐ Each **row** contains a single record.
- ☐ Each **column** represents a field within the record.
- ☐ Fundamental data objects in Access, **Forms**, **Queries** and **Reports** are all based on **Tables**.



2- Queries:

- ☐ Used to extract specific information from database and you can use a **Query** for adding, updating or deleting data in a database.
- ☐ Queries are composed of **Structured Query Language (SQL)** statements.

Example:

```
SELECT S.SNAME, S.STATUS, S.CITY, P.PNAME, P.COLOR, SP.QTY  
FROM S, P, SP  
WHERE S.CITY = "Baghdad" or P.CITY = "Babel";
```

- ☐ MS Access allows queries to be created **graphically** (graphical tools: **Query By Example**) for hiding complexity of SQL language.

3- Forms : Display and print data from a table(s) or a query based on a user defined custom format. Forms enable you to view, edit and print data.

4- Reports: Display and print data from a table(s) or a query based on a user defined custom format. You cannot edit data in a report.

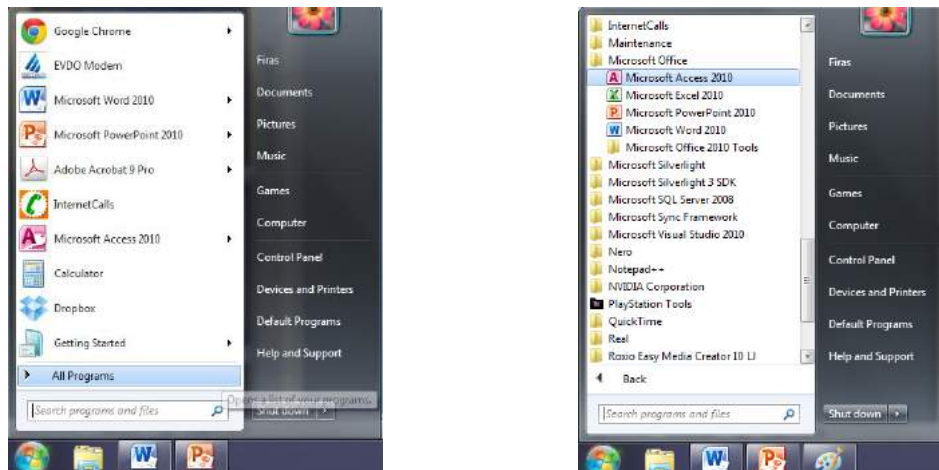
5- Macros : Automate common database actions based on user specified commands and events.

6- Modules: Automate complex operations and give a programmer more control than macros. Modules are procedures written in **Visual Basic for Applications (VBA)** Programming Language.

- Starting MS Office Access:

➤ **To start** MS Office Access 2010:

- 1) Click the Start button, and then click All Programs.
- 2) Click Microsoft Office, and then click Microsoft Office Access 2010.



As you open Access 2010, the default startup screen, called the **Backstage View**, is revealed (see Figure 1-1 below).

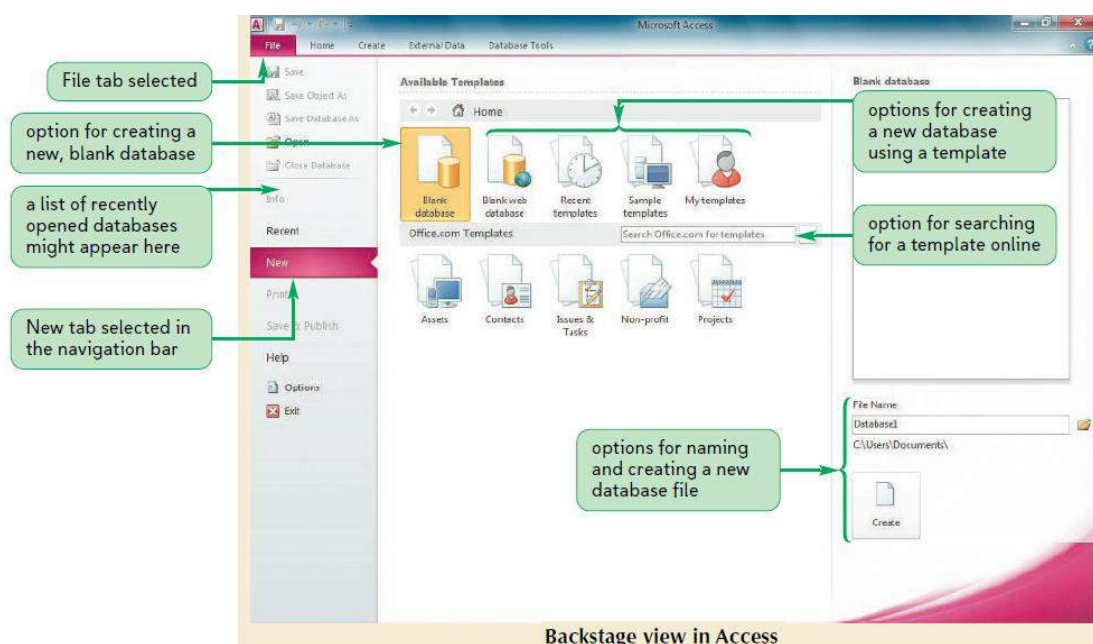

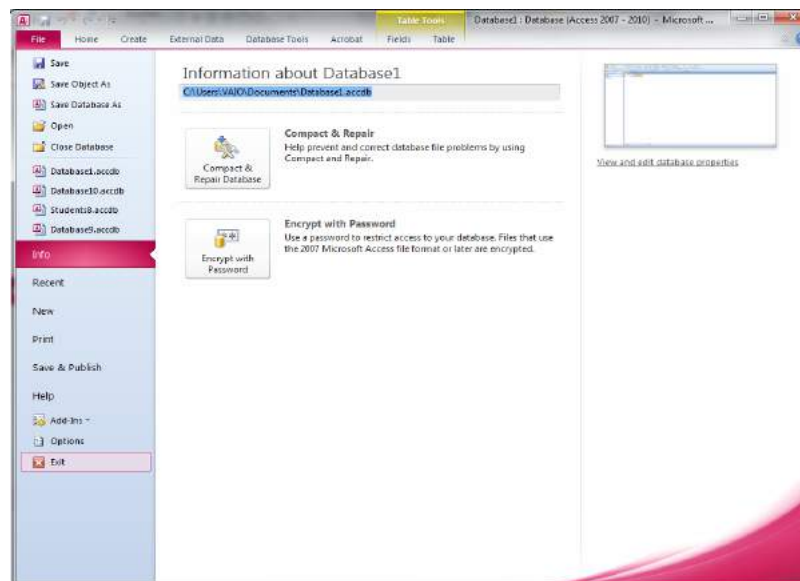


Figure 1-1: The Microsoft Access 2010 Backstage View – New Tab

➤ **To exit** from MS Office Access 2010:

Save all applications and select the **Exit** option from the **File** menu or click the  button on the extreme top-right corner of the window.



- **Planning a Database Structure:**

Before you create *tables*, *forms* and *report* etc., you should take time to plan the database. The time you invest in the planning will yield productivity gains as you create and maintain the database.

The following are some **key** issues to address as you design or plan your database:

- 1- **Determine the purpose of your database**, for example keeping track of all students' identity and their grades.
- 2- **Determine the tables you need in the database**, for example **STUDENT** table which keeps track of the identity of each student and **GRADE** table which keeps the track of all earned grades for each student.
- 3- **Determine the fields you need in the tables**, for example StudentNumber, LastName, FirstName and EmailAddress in **STUDENT** table and so on.
- 4- **Identify fields with unique values**, for example, StudentNumber within the **Student-Class-Grade** Database as each student should have a unique identity number.
- 5- **Determine the relationships between tables**, for example, a relationship between the **STUDENT** and the **GRADE** tables on the basis of key field StudentNumber to display the contents from the both tables.
- 6- **Add data and create other database objects.**

- Creating a Microsoft Access Database (Database File):

After you plan your database design, you are ready to create the database. MS Access 2010 provides you with a wide variety of **templates** (e.g. Faculty, Students, etc.) that you can use to speed up your database creation process.

You can either use these Database Templates to create an entire database **or** create a Blank Database.

1 - Creating a Database by using a Template:

- ❖ A **template** is a **ready-to-use** database that contains all of the *tables*, *queries*, *forms*, and *reports* needed for performing a specific task.
- ❖ Some **templates** contain a few sample records to help demonstrate their use.
- ❖ Template databases can be used as is, or you can **customize** them to better fit your needs. Figure 1-2 shows the sample templates in Microsoft Access 2010 Backstage View.

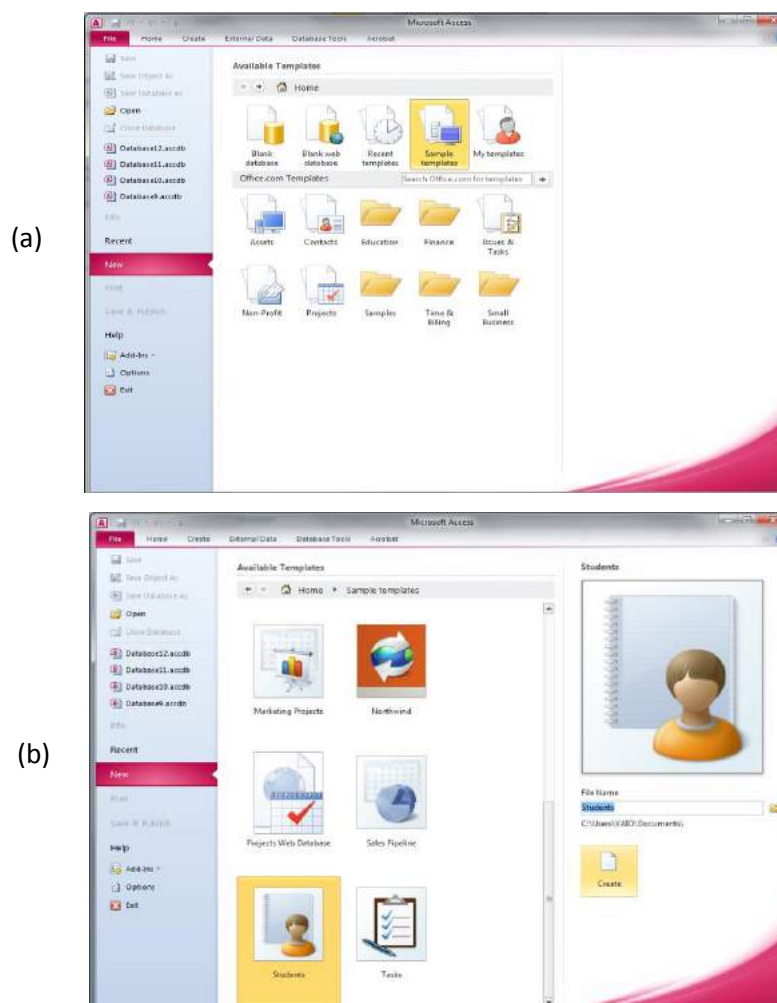


Figure 1-2: Sample Templates Snapshot

Assignment (I-1)

Please, try to open sample template (**Faculty**), enter any data and then open **Faculty Table**, see Figure 1-3 below for further details.

NOTE: Save all, in your own folder in (Lab) partition.

The screenshot shows a web-based form titled "Faculty Details" for a user named "Nada Ahmed". The form is divided into several sections: General, Employment Information, and Emergency Information. The General section includes fields for First Name (Nada), Last Name (Ahmed), E-mail Address (Nada_Ahmed@yahoo.com), and Web Page (www.Nada-Ahmed.com). There is also a section for Phone Numbers with fields for Business Phone (078888888), Home Phone (078888880), Mobile Phone (078888881), and Fax Number (078222222). An Address section includes fields for Address (New Baghdad), City (Baghdad), State/Province (Baghdad), Zip/Postal Code (11112), and Country/Region (Iraq). On the right side, there are dropdown menus for Faculty ID (1234), Faculty Type (Lecturer), Department (Science), and Office (English, Physics, Mathematics, Science, Physical Education). A Notes section is also present. The bottom of the form shows a status bar with "Record: 1 of 1", "No Filter", and a search field.

Field	Value
First Name	Nada
Last Name	Ahmed
E-mail Address	Nada_Ahmed@yahoo.com
Web Page	www.Nada-Ahmed.com
Business Phone	078888888
Home Phone	078888880
Mobile Phone	078888881
Fax Number	078222222
Address	New Baghdad
City	Baghdad
State/Province	Baghdad
Zip/Postal Code	11112
Country/Region	Iraq
Faculty ID	1234
Faculty Type	Lecturer
Department	Science
Office	English, Physics, Mathematics, Science, Physical Education

Figure 1-3: Faculty Details Form

2 - Creating a Database by using a Blank Database:

- ❖ If the Database templates do not meet your database design needs, create a **blank database**. The blank database is empty and does not contain any objects, relationships or sample data.
- ❖ To create a **blank database**, you will write the database name, for example **Student-Class-Grade.accdb**, into the **File Name** textbox and then click the **Create** button as shown in Figure 2-1. It is the first step to create a new **Microsoft Access 2010 Database File**.

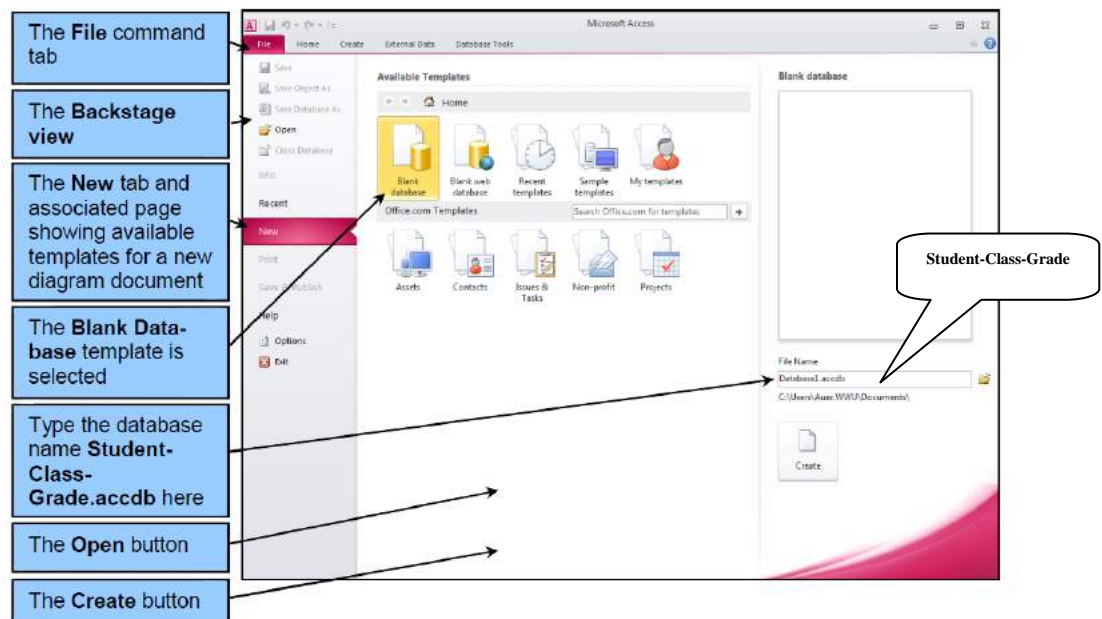


Figure 2-1: The Microsoft Access 2010 Backstage View – New Tab

NOTES:

- 1- By default, the database will be created in **your Documents** folder in the Documents library folder.
- 2- If you clicked the **Open** button to browse to a different file location, use the **File New Database** dialog box to create the new database file. Once you have browsed to the correct folder, type the database name in the **File Name** text box of the **File New Database** dialog box, and then click the **OK** button to create the new database.

- ❖ Because this is a new database, **Microsoft Access 2010** has assumed that you will want to immediately create a new table. Therefore, a new table named **Table1** is displayed in **Datasheet view** in the document window. We do not want this table open at this time, so click the **Close** document button shown in Figure 2-2.

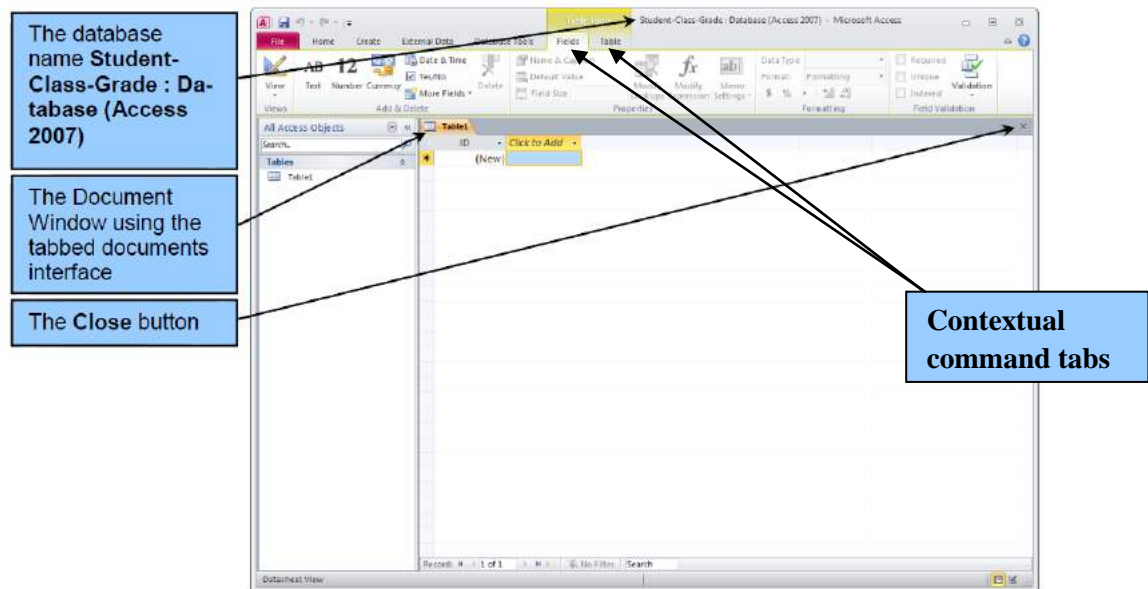


Figure 2-2: The New Microsoft Access Database

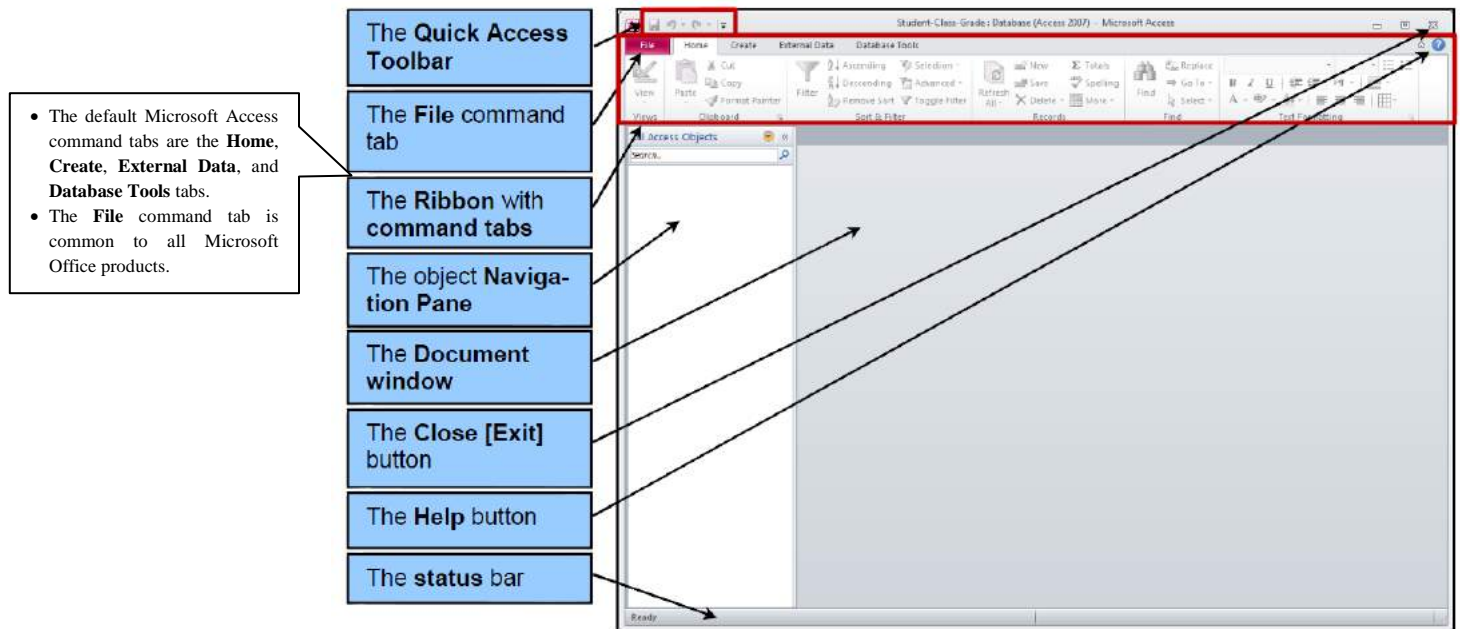


Figure 2-3: The Microsoft Office Fluent User Interface

- ❖ The **Microsoft Access 2010** window with the new database appears, as depicted in figure above. You can see most of the features of the **Microsoft Office Fluent User Interface** in this window.

- Closing a Database and Exiting Microsoft Access:

Click the **Close** button, the database closes, and then you can exit the Microsoft Access program by clicking on **Exit** button, as mentioned in *Lab (1)*.

- Creating Main MS Office Access Objects:

1- Tables:

Tables are the foundation of a database. All other database objects depend on the data in the tables.

➤ Creating Tables:

Microsoft Access 2010 provides a number of ways to create a table in;

- 1- **Datasheet View,**
- 2- **Design View,**
- 3- **SQL View.**

1– Create a table in Datasheet View:

- **Datasheet View** lets you create a table by entering data in a blank datasheet, similar to entering data in spreadsheet, when you save the datasheet, MS Access analyzes the data entered and automatically assigns the appropriate field types and formats to your fields.
- **Datasheet View** provides a visual way to create a table. To create a new table by entering data into a datasheet, follow these steps as depicted in Figure 2-4 below:

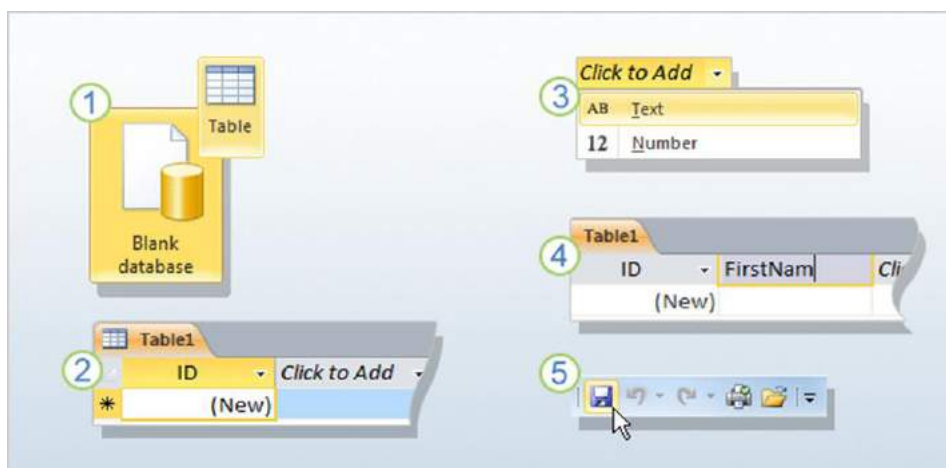
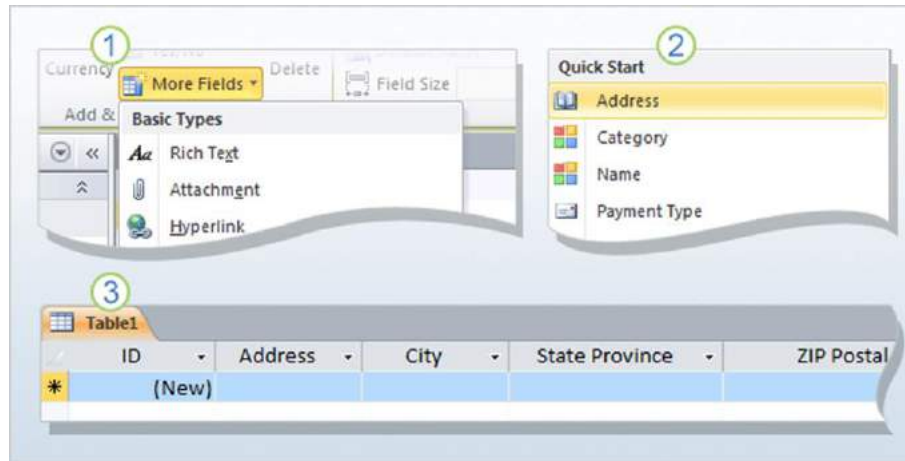


Figure 2-4: The Process of Creating a Table in Datasheet View

Go faster with Quick Start fields:

Quick Start fields are a faster way to build parts of a new table. The fields capture data for common business needs, and all field names and data types are set for you.



- ① With a table open in **Datasheet view**, click the **Fields** tab, and in the **Add & Delete** group, click **More Fields**. A list appears.
- ② Scroll down the list until you see the **Quick Start** section, click the type of fields you want to use, such as **Address**, or **Name**, and...
- ③ MS Access adds the fields for you, with field names data types already set.

You can use the new fields right away — just start entering data — or you can rename them, and remove fields you don't need.

2 – Create a table in Design View:

- **Design View** allows you to create a table from the scratch. In **Table Design View**, you can enter the field names, set field types and set field properties on your own.
- **Design View** allows you to build a table from scratch and set or change every available property for each field.
- You can also open existing tables in **Design view** and **add**, **remove**, or **change fields**, for doing that you may follow these steps as depicted in Figure 2-5 below:

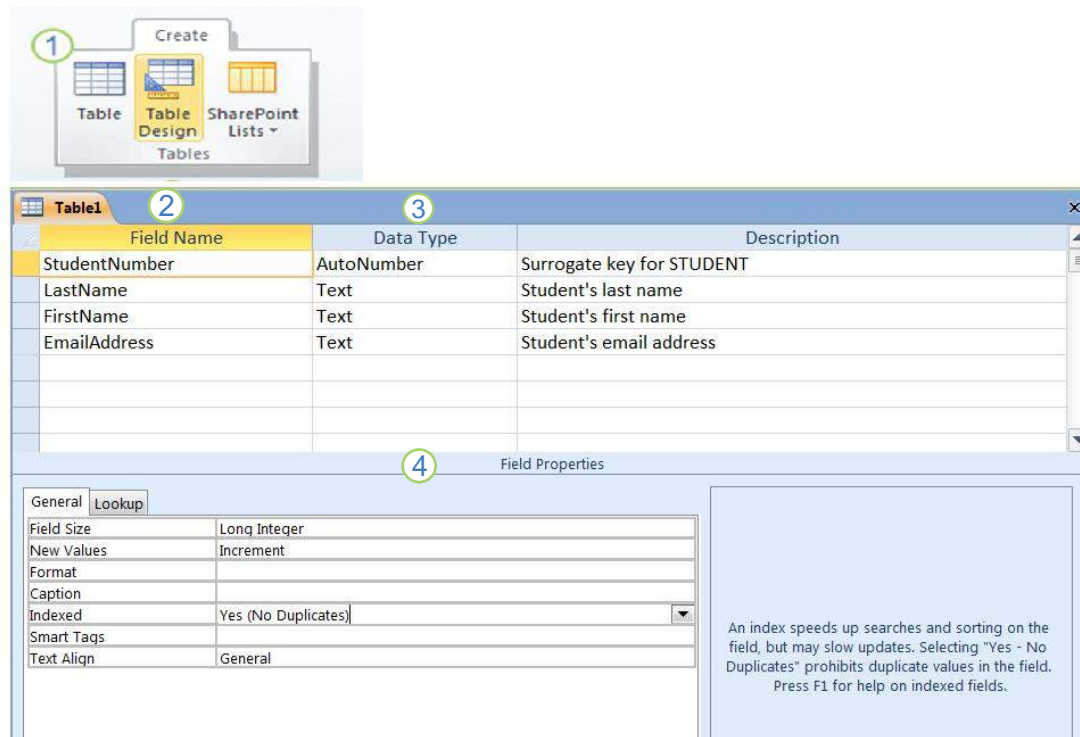


Figure 2-5: The Process of Creating a Table in Design View

Tip: As always, save your changes and give your new table a name that describes the data it contains.

The Components of Design View:

As illustrated in Figure 2-6, **Design View** contains the following components:

- ① **Field Grid Pane** contains columns that enable you to define **field names**, **data types** and **description**.
- ② **The Design Command Tab** and its command groups that help you design and work with your table.
- ③ **Field Properties Pane** contains various **properties** such as **Caption**, **Validations Rules** and **default setting** for each **field**.

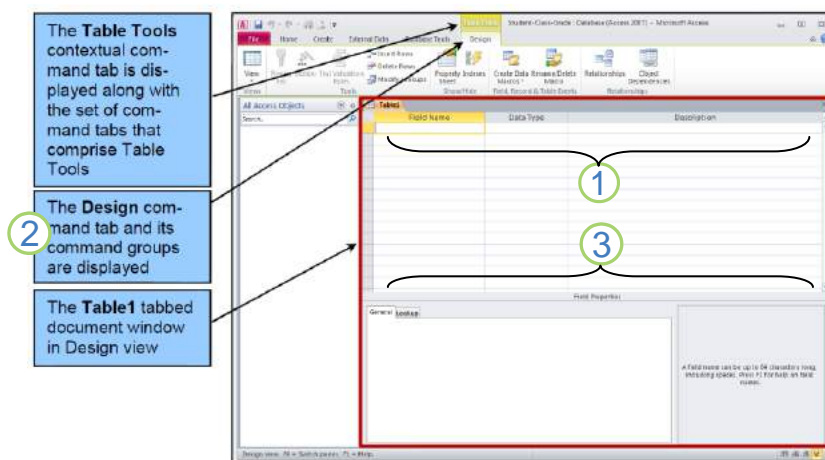


Figure 2-6: The Components of Design View

The next few sections explain how to work with those components;

1) Working in Field Grid Pane:

The **Field Grid Pane** enables you to define **field names**, **data types** and **descriptions**.

➤ Naming Fields:

- ✓ As is true of most objects in an Access database, field names contain up to **64 characters**.
- ✓ Field names must be **unique** within the table.

➤ Determining the Data Type:

- ✓ A **Data Type** specifies the kind of information you can store in a field. For example MS Access does not permit you to enter text in the Date/Time type field.
- ✓ Access also specifies the storage space used by each data type. A Date/Time value required eight bytes of storage space whereas text requires one byte for each character.

MS Access 2010 provides the following **basic data types**, as listed in Table 2-1:

Data Type	Type of Data Stored	Storage Size
Text	Alphanumeric characters	255 characters or less
Memo	Alphanumeric characters	65,536 characters or less
Number	Numeric values	1, 2, 4, or 8 bytes, 16 bytes for Replication ID (GUID)
Date/Time	Date and time data	8 bytes
Currency	Monetary data	8 bytes
AutoNumber	Automatic number increments	4 bytes, 16 bytes for Replication ID (GUID)
Yes/No	Logical values: Yes/No, True/False	1 bit (0 or –1)
OLE Object	Pictures, graphs, sound, video	Up to 1GB (disk space limitation)
Hyperlink	Link to an Internet resource	64,000 characters or less
Attachment	A special field that enables you to attach external files to an Access database.	Varies by attachment
Lookup Wizard	Displays data from another table	Generally 4 bytes

Table 2-1: Data Types Available in Microsoft Access

The exact type of numeric data stored in a **number field** is determined by the **Field Size** property. Table 2-2 lists the various numeric data types.

Field Size Setting	Range	Decimal Places	Storage Size
Byte	0 to 255	None	1 byte
Integer	–32,768 to 32,767	None	2 bytes
Long Integer	–2,147,483,648 to 2,147,483,647	None	4 bytes
Double	–1.797 × 10 ³⁰⁸ to 1.797 × 10 ³⁰⁸	15	8 bytes
Single	–3.4 × 10 ³⁸ to 3.4 × 10 ³⁸	7	4 bytes
Replication ID	N/A	N/A	16 bytes
Decimal	1–28 precision	15	2 bytes

Table 2-2: Numeric Field Settings

➤ Describing Fields:

- ✓ You use the **Description** column to provide further information (**hint**) about a field.
- ✓ The **Description** is optional. It is displayed in status bar when the insertion point is in the field in **Datasheet** or **Form View**.

2) Working in Design Command Tab:

In Table **Design View** window, the Design command tab contains several active buttons such as **Primary key**, **Relationships** and **Indexes** etc...

❑ Setting Primary Keys:

What is Primary key (P.K)?

- The **Primary Key** can be defined as a **unique identifier for the table**. That is, an attribute (or set of attributes) with the property that, at any given time, no two rows of the table contain the same value in that attribute (or set of attributes).
- i.e. **duplications** and **null value** not allowed. It is usually used to establish the **relationships**.
- So, the **Primary Key** identifies a record as being unique. In **Student-Class-Grade** database, for example, each student has a unique identify number. This **StudentNumber** field would be the primary key of its table.

The **benefits** of setting a primary key include the following:

- 1) **Speed** – MS Access creates an **index** based on the primary key, which enables MS Access to improve processing of queries and other functions.
- 2) **Order** – MS Access automatically sorts and displays database recodes in primary key order.
- 3) **No Duplicates** – MS Access does not permit users to enter data with the same primary key as an existing record.
- 4) **Connections** – MS Access maintains **relationships** between linked tables based on relating the **primary key (P.K.)** to **foreign key (F.K.)** in another table with the same information as the primary key.

To set a primary key, see figure 2-7.

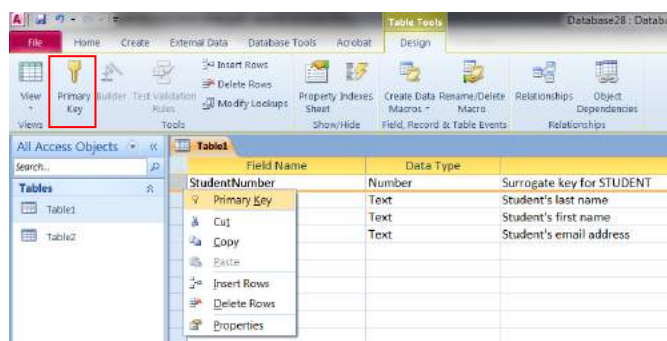


Figure 2-7: Primary Key Setting

Assignment (I-2)

Create a blank database, you will name your database **Student-Class-Grade** (**Student-Class-Grade.accdb**). The **tables** and **data** are shown in Figure 2-8 below.

NOTES:

1- Save all in your own folder in (Lab) partition.

2- To check the names of fields, see database schema format as:

STUDENT (StudentNumber, LastName, FirstName, EmailAddress)

CLASS (ClassNumber, ClassName, Term, Section)

GRADE (StudentNumber, ClassNumber, Grade)

The STUDENT table

StudentNumber	LastName	FirstName	EmailAddress
1	Cooke	Sam	Sam.Cooke@OurU.edu
2	Lau	Marcia	Marcia.Lau@OurU.edu
3	Harris	Lou	Lou.Harris@OurU.edu
4	Greene	Grace	Grace.Green@OurU.edu
(New)			

The CLASS table

ClassNumber	ClassName	Term	Section
10	CHEM 101	2010-Fall	1
20	CHEM 101	2010-Fall	2
30	CHEM 101	2011-Spring	1
40	ACCT 101	2010-Fall	1
50	ACCT 101	2011-Spring	1

The GRADE table with foreign keys—now each grade is linked back to the STUDENT and CLASS tables

StudentNumber	ClassNumber	Grade
1	10	3.7
1	40	3.5
2	20	3.7
3	30	3.1
4	40	3.0
4	50	3.5
		0.0

Figure 2-8 : The Student-Class-Grade Database

❑ Setting Relationships between Tables:

- The **relationship** is an association among entities, where an **entity** can be a **person**, an **object**, or **concept**, etc., about which some information needs to be stored in a database.
- There are **Three** types of relationships:
 - (1) **One-to-One**
 - (2) **One-to-Many (or Many-to-One)**
 - (3) **Many-to-Many**
- To create a **relationship** between **tables**, follow these steps:

- 1) The most common way of creating a relationship between tables is to add the tables to the **Relationships** window displayed when you click the **Relationships** button in the **Relationships** group on the **Database Tools** tab or **Design** tab as in Figure 3-1 below.



Figure 3-1: The Database Tools Command Tab

- 2) Add the tables to **Relationships** window, as shown in Figure 3-2, and then drag a field in one table to the common field in the other table and complete the relationship definition in the **Edit Relationships** dialog box as illustrated in Figure 3-3 below.

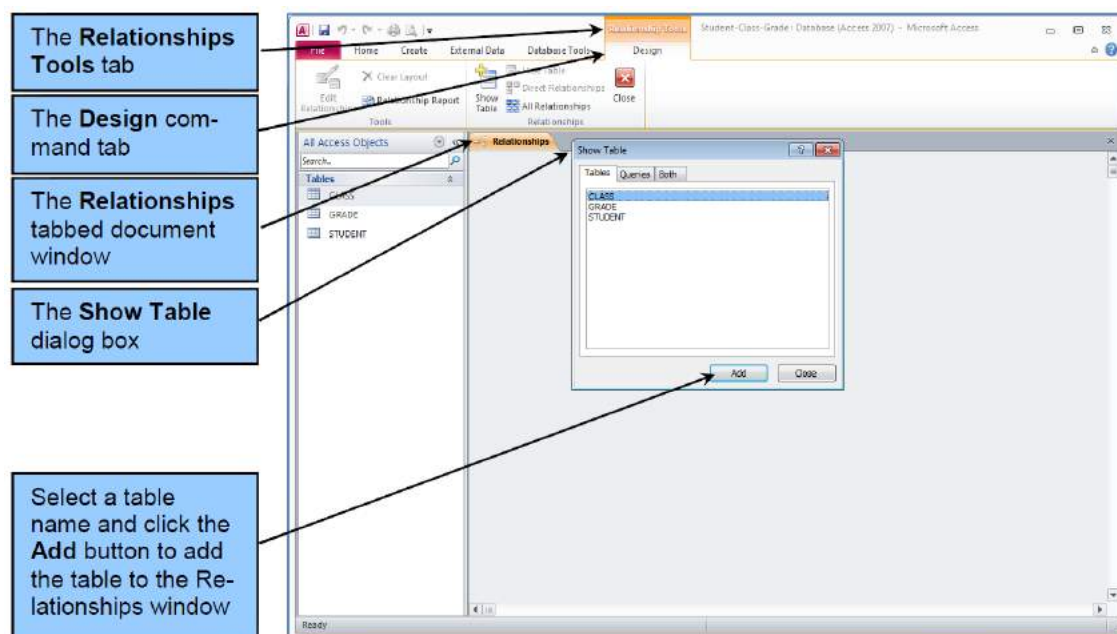


Figure 3-2: The Relationships Window

- 3) In this dialog box, you are given the opportunity to impose a restriction called **referential integrity** on the data, which means that an entry will not be allowed in one table unless it already exists in the other table.

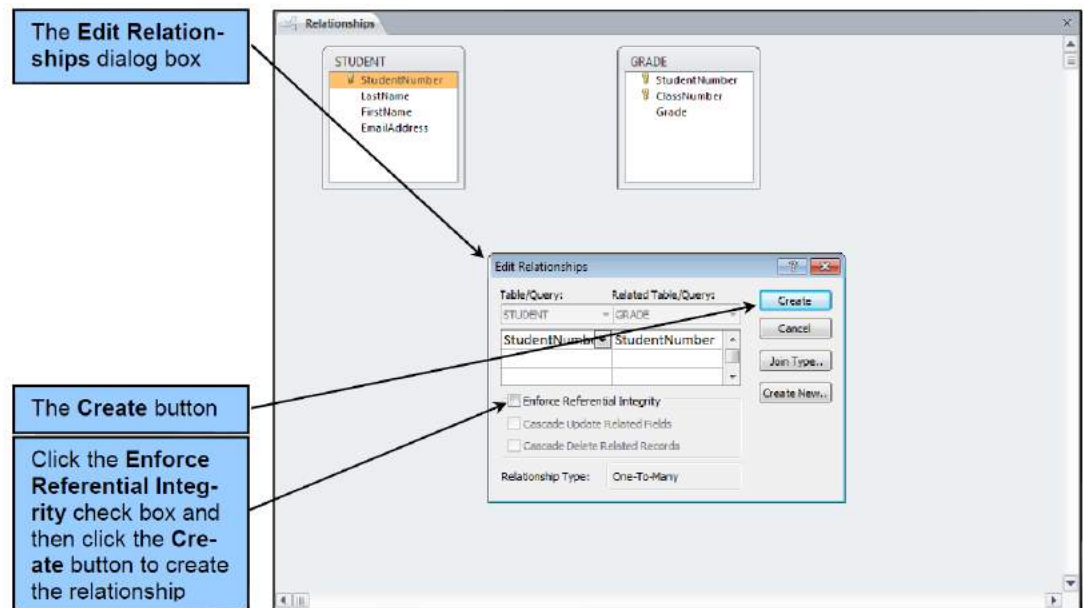


Figure 3-3: The Edit Relationships Dialog Box

- 4) Figure 3-4 shows the completed relationships for the **Student-Class-Grade** database. After you have created a relationship, you can delete it by deleting the line connecting the tables on the **Relationships** window. You can clear all the boxes from the window by clicking the **Clear Layout** button in the **Tools** group on the **Relationship Tools Design** contextual tab.

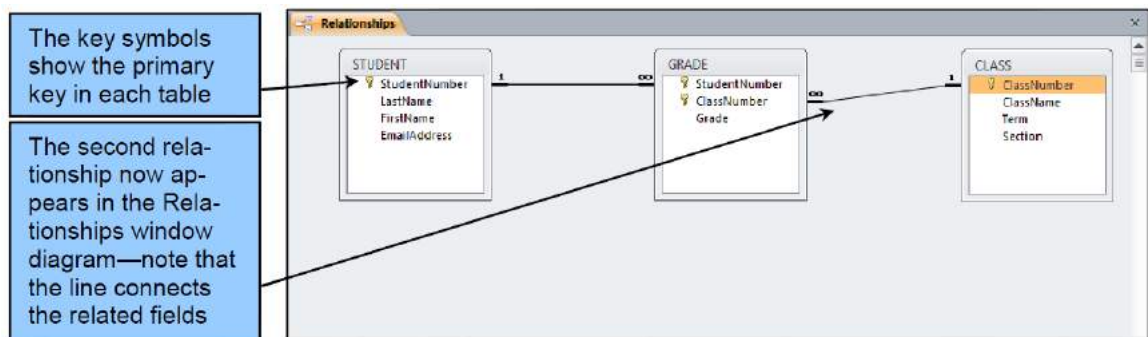


Figure 3-4: The Completed Relationships for the Student-Class-Grade Database

• What is Foreign Key (F.K)?

The **Foreign Key** can be defined as an attribute in relation and a primary key (or unique value) in other relation. It is usually used to establish the relationships.

❑ Setting Index Properties:

- In MS Access, an index is like a list of field values that appear in a table. Each entry in the list also shows the locations for the records that contain the field values. If you want to find a particular field value, an index makes it much faster than reading through the whole table.

NOTES:

- 1) The **Primary Key** of a table is **automatically** indexed.
- 2) You cannot index a field whose data type is **OLE Object**, **Attachment**, or **Calculated**.
- 3) You can create an index that is based on a **single field** or on **multiple fields**.
- 4) You should consider indexing fields that **you search frequently**, **fields that you sort**, and **fields that you join to fields in other tables in queries**.
- 5) Indexes can help speed up searches and select queries, but they can **slow down performance** when you add or update data. When you enter data in a table that contains one or more indexed fields, MS Access must update the indexes every time a record is added or changed
- 6) To see all the **indexes** stored on each table; On the **Design** tab, in the **Show/Hide** group, click **Indexes**.



- To create an index, you first decide whether you want to create a **single-field index** or a **multiple-field index**.
- You create an **index** on a **single field** by setting the **Indexed** property. The following table lists the possible settings for the **Indexed** property.

INDEXED PROPERTY SETTING	MEANING
No	Don't create an index on this field (or delete the existing index)
Yes (Duplicates OK)	Create an index on this field
Yes (No Duplicates)	Create a unique index on this field

Table 3-1: Indexed Property Setting

- To create a **single-field** index, follow these steps:
 - 1) In the **Navigation Pane**, right-click the name of the table that you want to create the index in, and then click **Design View** on the shortcut menu.
 - 2) Click the **Field Name** for the field that you want to index.
 - 3) Under **Field Properties**, click the **General** tab.
 - 4) In the **Indexed** property, click **Yes (Duplicates OK)** if you want to enable duplicates, or **Yes (No Duplicates)** to create a unique index.
 - 5) To save your changes, click **Save** on the **Quick Access Toolbar**, or press **CTRL+S**, see Figure 3-5.

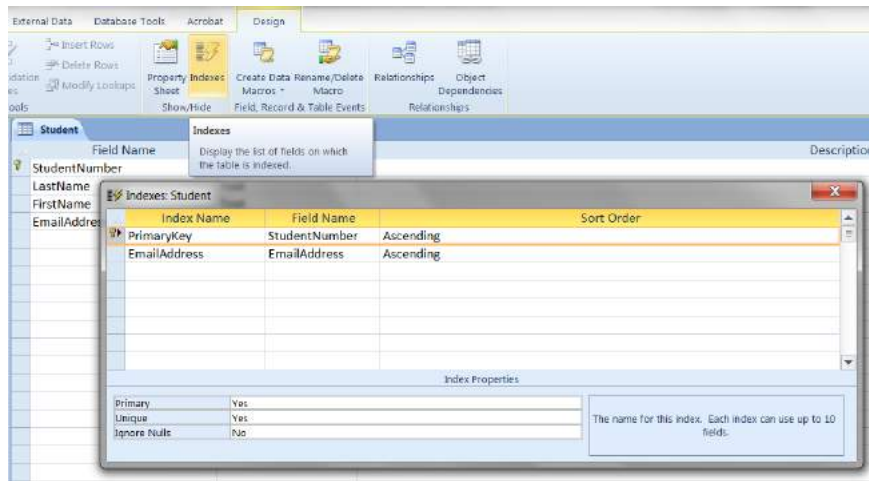


Figure 3-5: Indexes Window

□ Setting Table Properties:

- You can set properties that apply to an entire table or to entire records.
 - 1) Select the table whose properties you want to set.
 - 2) On the **Home** tab, in the **Views** group, click **View**, and then click **Design View**.
 - 3) On the **Design** tab, in the **Show/Hide** group, click **Property Sheet**. The table property sheet is shown.
 - 4) On the property sheet, click the **General** tab.
 - 5) Click the box to the left of the property that you want to set, and then enter a setting for the property.
 - 6) To save your changes, press **CTRL+S**.



Available Table Properties:

USE THIS TABLE PROPERTY	TO
Read Only When Disconnected	This is for databases using SharePoint/web database . The "Read Only When Disconnected" property by default is set to No , which means you can still update or add new records to a web table that is linked to a Microsoft SharePoint Services site when you are offline .
Subdatasheet Expanded	Expand all subdatasheets when you open the table.
Subdatasheet Height	Do one of the following: <ul style="list-style-type: none"> • If you want the subdatasheet window to expand to display all rows, leave this property set at 0". • If you want to control the height of the subdatasheet, enter the desired height in inches.
Orientation	Set the view orientation, according to whether your language is read left-to-right, or right-to-left.
Description	Provide a description of the table. This description will appear in tooltips for the table.
Default View	Set Datasheet , PivotTable , or PivotChart as the default view when you open the table.
Validation Rule	Enter an expression that must be true whenever you add or change a record.
Validation Text	Enter a message that is displayed when a record violates the expression in the Validation Rule property.
Filter	Define criteria to display only matching rows in Datasheet view.
Order By	Select one or more fields to specify the default sort order of rows in Datasheet view.
Subdatasheet Name	Specify whether a subdatasheet should appear in Datasheet view, and if so, which table or query should supply the rows in the subdatasheet.
Link Child Fields	List the fields in the table or query that are used for the subdatasheet that match the Link Master Fields property that is specified for the table.
Link Master Fields	List the fields in the table that match the Link Child Fields property that is specified for the table.
Filter On Load	Automatically apply the filter criteria in the Filter property (by setting to Yes) when the table is opened in Datasheet view.
Order By On Load	Automatically apply the sort criteria in the Order By property (by setting to Yes) when the table is opened in Datasheet view.

Table 3-2 : Available Table Properties

3) Working in Field Properties pane:

- After you create a field, you can set **Field properties** to control its **appearance** and **behavior**. For example, by setting **Field properties**, you can:
 - ✓ **Control the appearance of data in a field,**
 - ✓ **Help prevent incorrect data entry in a field,**
 - ✓ **Specify default values for a field...**
- You can set any field property while you work with a table in **Design view**. In **Design view**, you set a field's data type in the table design grid, and you set other properties in the **Field Properties** pane.
- The **Field properties** are organized on **Two** tabs, **General** and **Lookup**.

❖ General Tab:

To set the general properties, follow these steps:

- 1) In the table design grid, select the field for which you want to set properties. MS Access displays the properties for this field in the **Field Properties** pane.
- 2) In the **Field Properties** pane\ **General** tab, enter the settings that you want for each property, or press **F6** and then use the arrow keys to select a property.

Tip: 1- You can set some of the available field properties while you work in **Datasheet view**. To have access to and set the complete list of field properties; however, you must use **Design view**.
 2- **Not all properties** are available for every field. A **field's data type** determines which properties it

Some of available Field Properties:

FIELD PROPERTY	DESCRIPTION
Field Size	Set the maximum size for data stored as a Text, Number, or AutoNumber data type. <small>TIP For best performance, always specify the smallest sufficient Field Size.</small>
Format	Customize the way that the field appears by default when displayed or printed.
Decimal Places	Specify the number of decimal places to use when displaying numbers.
New Values	Specify whether an AutoNumber field is incremented or assigned a random value when a new record is added.
Input Mask	Display characters that help to guide data entry.
Caption	Set the text displayed by default in labels for forms, reports, and queries.
Default Value	Automatically assign a default value to a field when new records are added.
Validation Rule	Supply an expression that must be true whenever you add or change the value in this field.
Validation Text	Enter a message to display when a value violates the expression in the Validation Rule property.
Required	Require that data be entered in a field.
Text Align	Specify the default alignment of text within a control.
Indexed	Speed up access to data in this field by creating and using an index.
Unicode Compression	Compress text stored in this field when a small amount of text is stored (< 4,096 characters).
Action Tags	Attach an action tag to this field.
Append Only	Track the history of field values (by setting the property's value to Yes).

Table 3-3: Available Field Properties

❖ **Lookup Tab:**

- A **lookup field** displays a list of values from which the user can choose. This can make data entry **quicker** and more **accurate**.
- The **Two** types of **lookup fields** that you can create are a **lookup list** and a **value list**.

1) Use a lookup list:

- ✓ When a lookup field is based on a lookup list, the field gets its data from an existing **table** or **query** in the database.
- ✓ In this type of lookup, the tables are related and when the values in the data source changes, the current data are available in the lookup field.

(2) Use a lookup value list:

- ✓ When a lookup field is based on a lookup value list, the lookup field gets its data from a **list of values** that you type in when you create the field.
- ✓ This type of lookup field is ideal when you have a limited set of values that do not change often.

To set or change the lookup field properties, follow these steps:

- 1) In the table design grid, select the field for which you want to set properties. MS Access displays the properties for this field in the **Field Properties** pane.
- 2) In the **Field Properties** pane\ **Lookup** tab, enter the settings that you want for each property, or press **F6** and then use the arrow keys to select a property.

Tip:

- 1) Setting the **Display Control** property to either **Text Box** or **Check Box** disables lookups.
- 2) When you use the **Lookup Wizard**, the lookup field properties are automatically set. However, you can modify these properties to change the behavior of a lookup field.
- 3) The correct syntax for a value list is:

“Option 1”; “Option 2”; “Option 3”

Place each option between **double quotes** and separate each option with a **semicolon**.

Assignment (I-3)

- 1) Using the **Student-Class-Grade** database as depicted in Figure 3-6, do the following:
- Establish the **Relationships** between the tables.
 - Create a unique (No Duplicates) **index** on **EmailAddress** field.
 - Make a unique (No Duplicates) **multiple-field index** on fields **ClassName**, **term** and **section**.
 - Set up at least **Five** different properties that are discussed earlier in this lab.

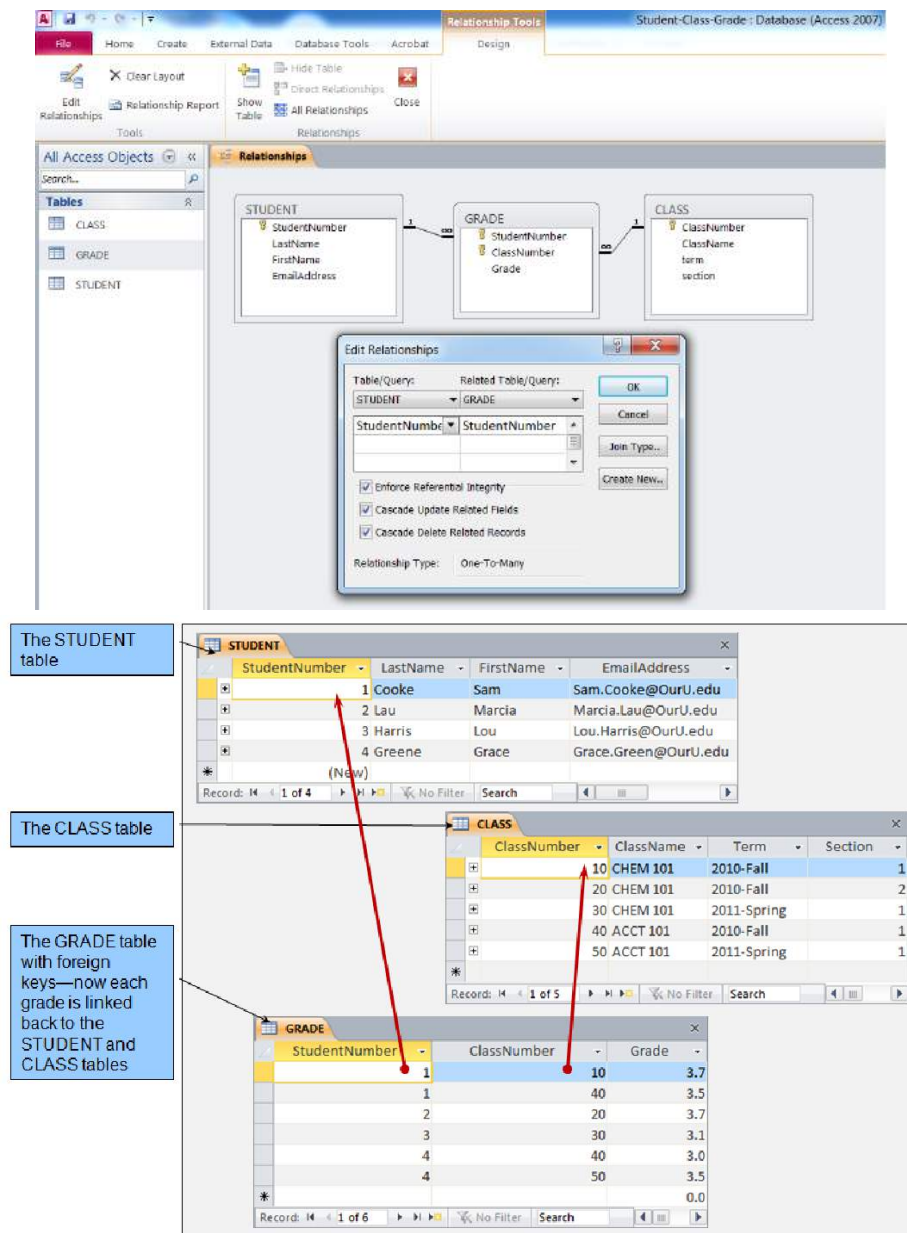


Figure 3-6: The Relationships Window of Student-Class-Grade Database

2) Create the **Suppliers-Parts-Shipments** (**Suppliers-Parts-Shipments.acddb**) database described in the following schema:

SUPPLIERS (**SID**, Sname, Status, City)

PARTS (**PID**, Pname, Color, Weight, City)

SHIPMENTS (**SID**, ***PID***, Qty)

Notes:

- 1- In the database schema, **Primary Key** columns in a table are underlined. Thus, the **Primary Key** of **SUPPLIERS** table is **SID**.
- 2- In the schema, **Foreign Key** columns in a table are shown in *italic*.

(a) **USER_ TABLES** Table:

TableName	NumberColumns	PrimaryKey
SUPPLIERS	4	SID
PARTS	5	PID
SHIPMENTS	3	(SID,PID)

(b) **USER_ COLUMNS** Table:

ColumnName	TableName	DataType	Length(bytes)
SID	SUPPLIERS	Text	5
Sname	SUPPLIERS	Text	20
Status	SUPPLIERS	Number (Integer)	4
City	SUPPLIERS	Text (Lookup List)	20
PID	PARTS	Text	6
Pname	PARTS	Text	20
Color	PARTS	Text (Lookup Value List)	15
Weight	PARTS	Number (Decimal)	(3,2)
City	PARTS	Text (Lookup List)	20
SID	SHIPMENTS	Text	5
PID	SHIPMENTS	Text	6
Qty	SHIPMENTS	Number (Integer)	4

Table 3-4 (a) & (b): The Metadata of Suppliers-Parts-Shipments Database

SUPPLIERS

SID	Sname	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

PARTS

PID	Pname	Color	Weight	City
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

SHIPMENTS

SID	PID	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Figure 3-7: The Suppliers-Parts-Shipments Database

Then, **do** the following:

- Create the **Relationships** between the tables in **Suppliers-Parts-Shipments** database.
- Create an **index** on **Sname** field.
- Set up at least **Five** different properties that are discussed earlier in this lab.

2- Queries:

- ✓ A database's primary purpose is to **store** and **extract** information. Information can be obtained from a database immediately after **the data is added**, or **days**, **weeks**, or **even years later**.
- ✓ A **MS Access Query** is a question that you ask about the data stored in **tables** or **queries**.
- ✓ You can use a **Query** for adding, updating or deleting data in a database. Also, you can use a **Query** for defining the structure (**schema**) of the database.
- ✓ MS Access provides **Three** ways to create the queries, as follows:
 - 1- Using Query Wizard**
 - 2- Using Query Design**
 - 3- Using SQL View**
- ✓ **MS Access** supports many different **Types of Queries**, as follows:
 - **Data Definition Query (SQL Specific Query)**
 - **Select Query**
 - **Total Query**
 - **Crosstab Query**
 - **Pass-Through Query (SQL Specific Query)**
 - **Union Query (SQL Specific Query)**
 - **Join Query**
 - **Action Query (Make-Table, Append, Update or Delete)**
 - **Subquery, etc...**

We'll discuss each type, in the next section and later.

❖ Data Definition Query:

- ✓ An **SQL Specific Query** that can **create**, **delete** or **alter** a **table**, or **create** or **delete** an **index**.

Tip: An **SQL Specific Query** is one that can be created **only** by writing an **SQL statement** in **SQL View**.
Union, **Pass-Through**, and **Data Definition** queries are **SQL Specific Queries**.

✓ To switch to **SQL View**;

- 1- Open the query in **Design** or **Datasheet View**.
- 2- Then click on the drop-down **View** list and then select the **SQL > SQL View**.
- 3- When you finish writing or changing the SQL statements as needed, go to the **Design** tab, and in the **Results** group, click **Run**.

✓ The Data Definition Queries:

- 1) **CREATE TABLE** is used to create database tables, defining their structure in the terms of fields. In other words; use the **CREATE TABLE** statement to define a new table and its fields and field constraints.
- 2) **DROP TABLE** is used for removing tables.
- 3) **ALTER TABLE** is used to change the structure of a table, adding, removing or changing the length or type of fields.
- 4) **CREATE INDEX** is used to create indexes one or more specified fields in a table.
- 5) **DROP INDEX** is used for removing an index.

1) **CREATE TABLE statement**: Creates a new table.**Syntax:**

```
CREATE TABLE table ( field1 type [(size)] [NOT NULL] [ CONSTRAINT index1]

[ , field2 type [(size)] [NOT NULL] [ CONSTRAINT index2] [, ...] ]

[ , CONSTRAINT multifieldindex [, ...] ] )
```

The **CREATE TABLE** statement has these parts:

Part	Description
<i>table</i>	The name of the table to be created.
<i>field1, field2</i>	The name of field or fields to be created in the new table. You must create at least one field.
<i>type</i>	The data type of <i>field</i> in the new table.
<i>size</i>	The field size in characters (Text and Binary fields only).
<i>index1, index2</i>	A CONSTRAINT clause defining a single-field index. For more information on how to create this index, see CONSTRAINT Clause .
<i>multifieldindex</i>	A CONSTRAINT clause defining a multiple-field index. For more information on how to create this index, see CONSTRAINT Clause .

Tip: If **NOT NULL** is specified for a field, then new records are required to have valid data in that field.

CONSTRAINT Clause:

- ✓ A **constraint** is similar to an **index**, although it can also be used to establish a relationship with another table.
- ✓ You use the **CONSTRAINT Clause** in **ALTER TABLE** and **CREATE TABLE** statements to create or delete constraints.
- ✓ There are **Two** types of **CONSTRAINT Clause**: one for creating a constraint on a **single field** and one for creating a constraint on **more than one field**.

Syntax:

Single-field constraint:

```
CONSTRAINT constraint_name { PRIMARY KEY |  
  

UNIQUE |  
  

NOT NULL |  
  

REFERENCES foreigntable [( foreignfield ) ] }
```

Multiple-field constraint:

```
CONSTRAINT constraint_name { PRIMARY KEY (primary1 [, primary2 [, ...]]) |  
  

UNIQUE (unique1 [, unique2 [, ...]]) |  
  

NOT NULL (notnull1 [, notnull2 [, ...]]) |  
  

FOREIGN KEY (ref1 [, ref2 [, ...]]) REFERENCES foreigntable [( foreignfield1 [, foreignfield2 [, ...]]) ] }
```

The **CONSTRAINT clause** has these parts:

Part	Description
<i>constraint_name</i>	The name of the constraint to be created.
<i>primary1</i> , <i>primary2</i>	The name of the field or fields to be designated the primary key.
<i>unique1</i> , <i>unique2</i>	The name of the field or fields to be designated as a unique key.
<i>notnull1</i> , <i>notnull2</i>	The name of the field or fields that are restricted to non-Null values.
<i>ref1</i> , <i>ref2</i>	The name of a foreign key field or fields that refer to fields in another table.
<i>foreigntable</i>	The name of the foreign table containing the field or fields specified by <i>foreignfield</i> .
<i>foreignfield1</i> , <i>foreignfield2</i>	The name of the field or fields in <i>foreigntable</i> specified by <i>ref1</i> , <i>ref2</i> . You can omit this clause if the referenced field is the primary key of <i>foreigntable</i> .

Tip: 1) You can use the **PRIMARY KEY** reserved words to designate one field or set of fields in a table as a **primary key**. All values in the primary key must be **unique** and **not Null**, and there can be only one **primary key** for a table.

2) You can use the **UNIQUE** reserved word to designate a field as a **unique key**. This means that no two records in the table can have the same value in this field. You can constrain any field or list of fields as unique. If a multiple-field constraint is designated as a unique key, the combined values of all fields in the index must be unique, even if two or more records have the same value in just one of the fields.

- 3) You can use the **FOREIGN KEY** reserved words to designate a field as a **foreign key**. If the foreign table's primary key consists of more than one field, you must use a multiple-field constraint definition, listing all of the referencing fields, the name of the foreign table, and the names of the referenced fields in the foreign table in the same order that the referencing fields are listed.

2) **DROP TABLE** statement: Deletes an existing table from a database.

Syntax:

DROP TABLE *table*

The **DROP TABLE** statement has this part:

Part	Description
<i>table</i>	The name of the table to be deleted

Notes: 1- You must close the table before you can delete it.

2- If the table is involved in a relationship with another table, you will have to delete the relationship first.

Remember that!

Microsoft Access data type	SQL Data Types
Yes/No	BIT
AutoNumber	COUNTER
CURRENCY	CURRENCY
DATE/TIME	DATETIME
MEMO	LONGTEXT
NUMBER (FieldSize= SINGLE)	SINGLE
NUMBER (FieldSize= DOUBLE)	DOUBLE
NUMBER (FieldSize= BYTE)	BYTE
NUMBER (FieldSize= INTEGER)	SHORT
NUMBER (FieldSize= LONG INTEGER)	LONG
OLE	LONGBINARY
TEXT	TEXT

Table 4-1: Mapping Data Types

Assignment (I-4)

1) In new database (Call it **Customers-Payments**), create and run the following **queries**, **after correct them**:

- Create table **CUSTOMER**

(Cno short **Constraint** CPKC primary key,

Title text (5) , FirstName text not null, Street text (15) , Post text (10) , Balance currency)

- Create table **INVOICE**

(Invno short **Constraint** CPKI primary key,

Cno short **Constraint** CFKI references CUSTOMER (Cno),

Invdate, Amount currency)

- Create table **PAYMENT**

(Pmtno short, Invno long , Pmtdate datetime, Amount currency,

Constraint CPKP primary key (Invno , Pmtno),

Constraint CFKP foreign key(Invno) references INVOICE (Invno))

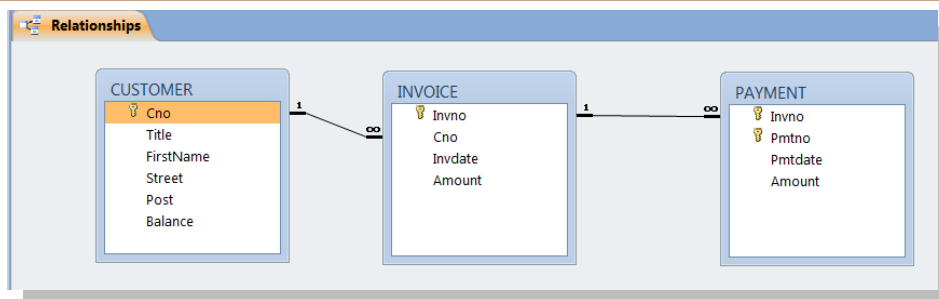


Figure 4-1: Customers-Payments database

2) Remove the **INVOICE** table.

3) Delete the tables; **SUPPLIERS**, **PARTS** and **SHIPMENTS** from **Suppliers-Parts-Shipments** database.

4) Create the tables and relationships as depicted in figure below:

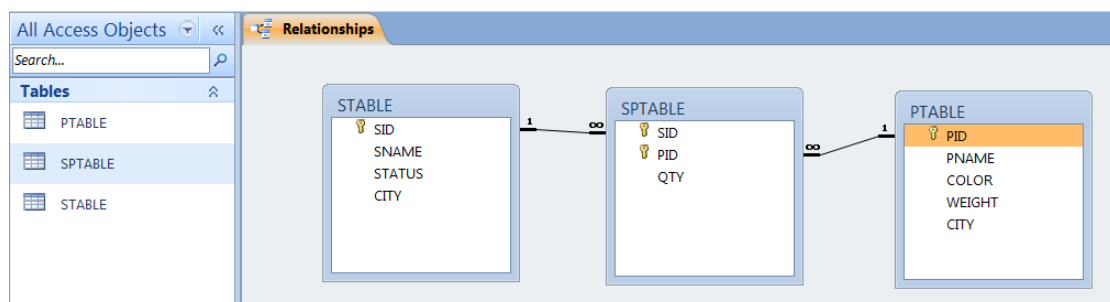


Figure 4-2: Suppliers-Parts-Shipments database

- 5) Create a database file (Call it *e-Voting*) and then construct the tables with all possible constraints as depicted in figure below:

Note: Give a suitable data type and a size for each field (if any).

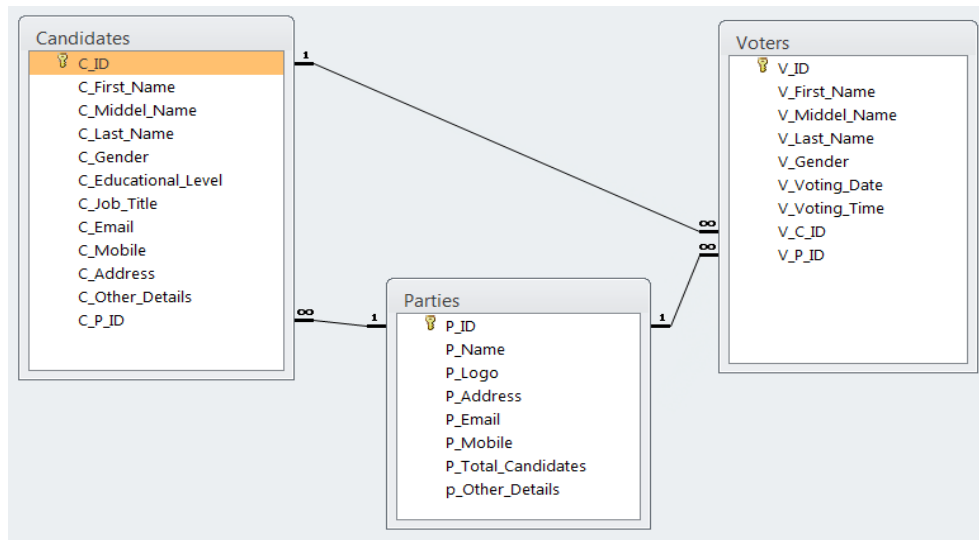


Figure 4-3: *e-Voting* database

- 6) In new database file (Call it *Shopping*), create all the tables with suitable data type and size for each field (if any), also create all possible constraints as depicted in figure below:

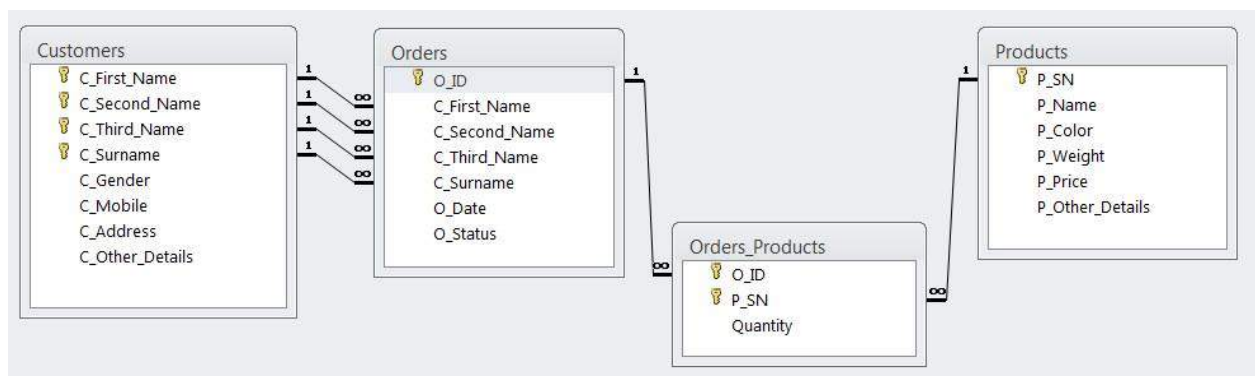


Figure 4-4: *Shopping* database

3) ALTER TABLE Statement:

- ✓ To maintain a **Table**, you use an **ALTER TABLE** command.
- ✓ You can use an **ALTER TABLE** command to **Add**, **Alter**, or **Drop Columns** | **Constraints**.
- ✓ An **ALTER TABLE** command has the following syntax:

Syntax:

```
ALTER TABLE table_name  
Predicate
```

Where *Predicate* can be any one of the following:

```
• ADD COLUMN field type[(size)] [NOT NULL] [CONSTRAINT index ]
```

```
• ADD CONSTRAINT multifieldindex
```

```
• ALTER COLUMN field type[(size)]
```

```
• DROP COLUMN field
```

```
• DROP CONSTRAINT constraint_name
```

4) CREATE INDEX Statement:

- ✓ To create an index on an existing table, you use a **CREATE INDEX** command.
- ✓ A **CREATE INDEX** command has the following syntax:

Syntax:

```
CREATE [UNIQUE] INDEX index_name  
ON table_name (field1 [ASC|DESC] [ , field2 [ASC|DESC] , ... ])  
[ WITH { PRIMARY | DISALLOW NULL | IGNORE NULL } ]
```

Note:

1- To prohibit duplicate values in the indexed field or fields of different records, use the **UNIQUE** reserved word.

2- In the optional **WITH** clause you can enforce data validation rules. You can:

- Designate the indexed field or fields as the primary key by using the **PRIMARY** reserved word. This implies that the key is **unique**, so you can omit the **UNIQUE** reserved word.
- Prohibit **Null** entries in the indexed field or fields of new records by using the **DISALLOW NULL** option.
- Prevent records with Null values in the indexed field or fields from being included in the index by using the **IGNORE NULL** option.

Tip:

- Suppose that you have a table named **Cars** with fields that store the **Make**, **Model**, **Year**, **Price**, and **Condition** of used cars for sale.
- Also suppose this table has become large and that you frequently include the **Model** field in queries. You can create an index on the **Model** field to help your queries return results more quickly by using the following **SQL** statement:

```
CREATE INDEX IdxModel
ON Cars ( Model )
```

5) DROP INDEX Statement:

- ✓ Is used for removing an index. That is, deletes an existing index from a table.
- ✓ A **DROP INDEX** command has the following syntax:

Syntax:

```
DROP INDEX index_name
ON table_name
```

Note: You must close the table before you can remove an index from it.

Remember that!

DDL KEYWORDS

Keyword	Use
CREATE	Create an index or table that does not already exist.
ALTER	Modify an existing table or column.
DROP	Delete an existing table, column, or constraint.
ADD	Add a column or a constraint to a table.
COLUMN	Use with ADD, ALTER, or DROP
CONSTRAINT	Use with ADD, ALTER, or DROP
INDEX	Use with CREATE
TABLE	Use with ALTER, CREATE, or DROP

Table 5-1: DDL Keywords

Assignment (I-5)

- 1) Assume you have a database contains **three** tables. The database **schema** format as:

STUDENTS (StudentID, FirstName, SecondName, LastName, BOD, Email)

SUBJECTS (SubjectCode, SubjectName, Semester)

DEGREES (StudentID, SubjectCode, Degree)

A- Create this database (Call it **Students-Info**) without **Foreign Key** constraints.

B- Use **Alter Table** command to construct a relationship (Call it **CFKDSID**) between **STUDENTS** and **DEGREES**, and another relationship (Call it **CFKDSCCode**) between **SUBJECTS** and **DEGREES**.

Note:

- 1- In the database schema, **Primary Key** columns in a table are underlined, and also **Foreign Key** columns in a table are shown in *italic*.
- 2- Please, decide on a suitable data type and a size for each field (if any).

- 2) Add **Mobile** field to the **STUDENTS** table.
- 3) Remove **Email** field from the **STUDENTS** table.
- 4) Append **Notes** field to the **SUBJECTS** table with data type **Text** and size more than (450) characters.
- 5) Create an index on **Degree** field (Call it **IdxDegree**).
- 6) Make a unique index on **Mobile** field (Call it **IdxUniqueMobile**), this index prohibits **Null** values as entries.
- 7) Delete the index (**IdxUniqueMobile**) that has been created in the previous question.
- 8) Create a unique index on fields (**SubjectName**, **Semester**) in **SUBJECTS** table (Call it **IdxUniqueSNS**).
- 9) Firstly, append **Photo** field to the **STUDENTS** table, then make it as indexed field.
(Hint: you will encounter an error!) Why?
- 10) Remove the relationship between **SUBJECTS** and **DEGREES**.



2021 /9/ 2

م.م. فراس محمد كاظم

1. Relational Algebra

The relational algebra is a theoretical language with operations that work on one or more relations. Thus, both the operands and the results are relations; hence the output from one operation can become the input to another operation. Relational algebra is an abstract language, which means that the queries formulated in relational algebra are not intended to be executed on a computer.

Relational algebra consists of group of relational operators that can be used to manipulate relations to obtain a desired result.

1.1 Role of Relational Algebra in DBMS

Knowledge about relational algebra allows us to understand query execution and optimization in relational database management system. The role of relational algebra in DBMS is shown in Fig. 1. From the figure it is evident that when a SQL query has to be converted into an executable code, first it has to be parsed to a valid relational algebraic expression, then there should be a proper query execution plan to speed up the data retrieval. The query execution plan is given by query optimizer.

1.2 Relational Algebra Operations

Operations in relational algebra can be broadly classified into set operation and database operations.

Unary and Binary Operations

Unary operation involves one operand, whereas binary operation involves two operands. The selection and projection are unary operations. Union, difference, Cartesian product, and Join operations are binary operations:

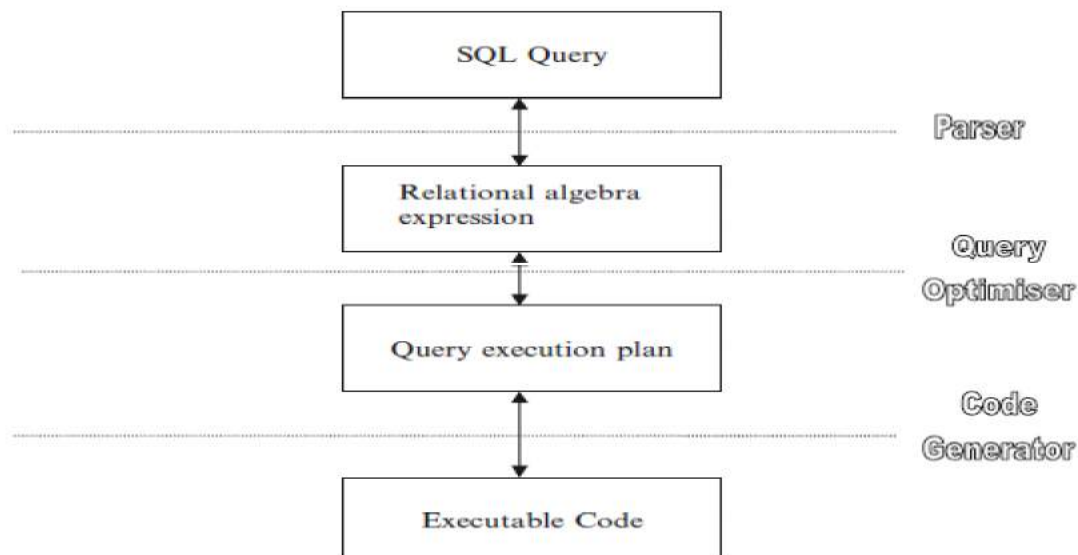
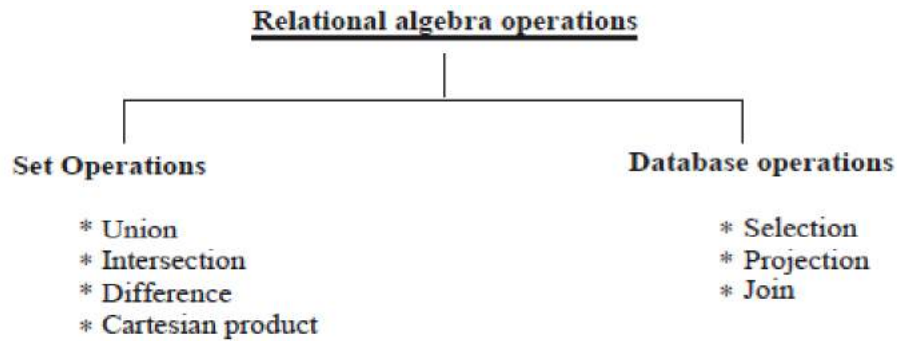


Fig. 1. Relational algebra in DBMS

- ☞ Unary operation operate on one relation
- ☞ Binary operation operate on more than one relation



Three main database operations are SELECTION, PROJECTION, and JOIN. Selection Operation The selection operation works on a single relation R and defines a relation that contains only those tuples of R that satisfy the specified condition (Predicate). Selection operation can be considered as row wise filtering. This is pictorially represented in Fig. 2.

2. Syntax of Selection Operation

The syntax of selection operation is: $\sigma_{\text{Predicate}}(R)$. Here R refers to relation and predicate refers to condition.



Fig. 2. Pictorial representation of SELECTION operation

Illustration of Selection Operation

To illustrate the SELECTION operation, consider the STUDENT relation with the attributes Roll number, Name, and GPA (Grade Point Average).

Example

Consider the relation STUDENT shown later:

STUDENT

Student Roll. No	Name	GPA
001	Aravind	7.2
002	Anand	7.5
003	Balu	8.2
004	Chitra	8.0
005	Deepa	8.5
006	Govind	7.2
007	Hari	6.5

Query 1: List the Roll. No, Name, and GPA of those students who are having GPA of above 8.0

Query expressed in relational algebra as: $\sigma_{\text{GPA} > 8} (\text{Student})$.

The result of the earlier query is:

Student Roll. No	Name	GPA
003	Balu	8.2
005	Deepa	8.5

Query 2: Give the details of first four students in the class.

Relational algebra expression is: $\sigma_{\text{Roll. No} \leq 4} (\text{student})$.

Table as a result of query 2 is:

Student Roll. No	Name	GPA
001	Aravind	7.2
002	Anand	7.5
003	Balu	8.2
004	Chitra	8.0

3. Projection Operation

The projection operation works on a single relation R and defines a relation that contains a vertical subject of R, extracting the values of specified attributes and elimination duplicates. The projection operation can be considered as column wise filtering. The projection operation is pictorially represented in Fig. 3.

Syntax of Projection Operation

The syntax of projection operation is given by: $\prod_{a_1, a_2, \dots, a_n} (R)$. Where a_1, a_2, \dots, a_n are attributes and R stands for relation.

STAFF

Staff No	Name	Gender	Date of birth	Salary
SL21	Raghavan	M	1-5-76	15,000
SL22	Raghu	M	1-5-77	12,000
SL55	Babu	M	1-6-76	12,500
SL66	Kingsly	M	1-8-78	10,000



Fig. 3. Pictorial representation of Projection operation

Illustration of Projection Operation

To illustrate projection operation consider the relation STAFF, with the attributes Staff number, Name, Gender, Date of birth, and Salary.

Query 1: Produce the list of salaries for all staff showing only the Name and salary detail.

Relational algebra expression:

$$\prod_{\text{Name.salary}} (\text{staff})$$

Output for the Query 1

Name	Salary
Raghavan	15,000
Raghu	12,000
Babu	12,500
Kingsly	10,000

Query 2: Give the name and Date of birth of the all the staff in the STAFF relation.

Relational algebra expression for query 2: $\Pi_{\text{Name, date of birth}}(\text{staff})$

Name	Date of birth
Raghavan	1-5-76
Raghu	1-5-77
Babu	1-6-76
Kingsly	1-8-78

4. Union Compatibility

In order to perform the Union, Intersection, and the Difference operations on two relations, the two relations should be union compatible. Two relations are union compatible if they have same number of attributes and belong to the same domain. Mathematically UNION COMPATIBILITY it is given as:

Let $R(A_1, A_2, \dots, A_n)$ and $S(B_1, B_2, \dots, B_n)$ be the two relations. The relation R has the attributes A_1, A_2, \dots, A_n and the relation S has the attributes B_1, B_2, \dots, B_n . The two relations R and S are union compatible if $\text{dom}(A_i) = \text{dom}(B_i)$ for $i = 1$ to n .

Union Operation

The union of two relations R and S defines a relation that contains all the tuples of R or S or both R and S , duplicate tuples being eliminated.

Relational Algebra Expression

The union of two relations R and S are denoted by $R \cup S$. It is pictorially represented in the Fig. 4.

Illustration of UNION Operation

To illustrate the UNION operation consider the two relations Customer 1 and Customer 2 with the attributes Name and city.

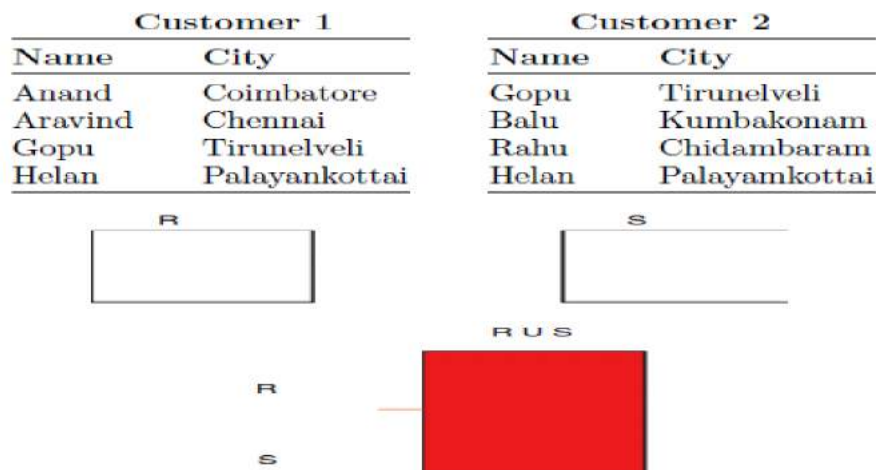


Fig. 4 Union of two relations R and S

Query: Determine Customer 1 \cup Customer 2

Result of Customer 1 \cup Customer 2

Customer 1 \cup Customer 2	
Name	City
Anand	Coimbatore
Aravind	Chennai
Balu	Kumbakonam
Gopu	Tirunelveli
Rahu	Chidambaram
Helan	Palayamkottai

5. Intersection Operation

The intersection operation defines a relation consisting of the set of all tuples that are in both R and S.

Relational Algebra Expression

The intersection of two relations R and S is denoted by $R \cap S$.

Illustration of Intersection Operation

The intersection between the two relations R and S is pictorially shown in Fig. 5.



Fig. 5. Intersection of two relations R and S

Example

Find the intersection of Customer 1 with Customer 2 in the following table.

Customer 1 \cap Customer 2	
Name	City
Gopu	Tirunelveli
Helan	Palayamkottai

6. Difference Operation

The set difference operation defines a relation consisting of the tuples that are in relation R but not in S.

Relational Algebra Expression

The difference between two relations R and S is denoted by $R - S$.

Illustration of Difference Operation

The difference between two relations R and S is pictorially shown in Fig. 6.

Example

Compute $R - S$ for the relation shown in the following table.



Fig. 6. Difference between two relations R and S

Customer 1 – Customer 2	
Name	City
Anand	Coimbatore
Aravind	Chennai

7. Cartesian Product Operation

The Cartesian product operation defines a relation that is the concatenation of every tuples of relation R with every tuples of relation S. The result of Cartesian product contains all attributes from both relations R and S.

Relational Algebra Symbol for Cartesian Product:

The Cartesian product between the two relations R and S is denoted by $R \times S$.

Note: If there are n_1 tuples in relation R and n_2 tuples in S, then the number of tuples in $R \times S$ is $n_1 \times n_2$.

Example

If there are 5 tuples in relation “R” and 2 tuples in relation “S” then the number of tuples in $R \times S$ is $5 \times 2 = 10$.

Illustration of Cartesian Product:

To illustrate Cartesian product operation, consider two relations R and S as given later:

R	S
a	1
b	2
	3

Determine $R \times S$:

R	S
a	1
a	2
a	3
b	1
b	2
b	3

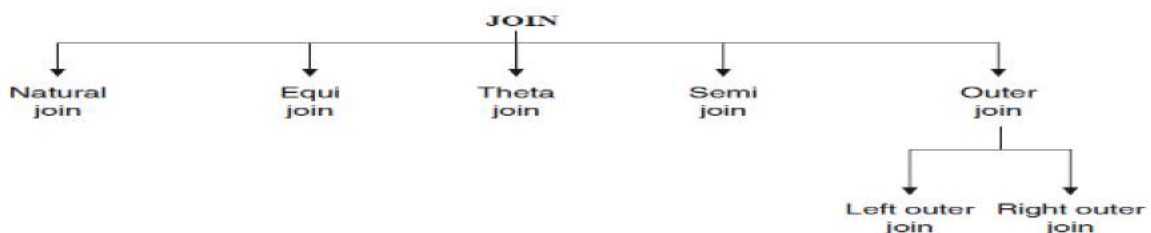
Note: No. of tuples in $R \times S = 2 \times 3 = 6$

No. of attributes in $R \times S = 2$

8. Join Operations

Join operation combines two relations to form a new relation. The tables should be joined based on a common column. The common column should be compatible in terms of domain.

Types of Join Operation



8.1 Natural Join

The natural join performs an equi join of the two relations R and S over all common attributes. One occurrence of each common attribute is eliminated from the result. In other words a natural join will remove duplicate attribute. In most systems a natural join will require that the attributes have the same name to identity the attributes to be used in the join. This may require a renaming mechanism. Even if the attributes do not have same name, we can perform the natural join provided that the attributes should be of same domain.

. For example a JOIN operation can be specified as a CARTESIAN PRODUCT followed by a SELECT operation, as we discussed:

$$R \bowtie_{\langle \text{condition} \rangle} S = \sigma_{\langle \text{condition} \rangle} (R \times S)$$

Example of Natural Join Operation

Consider two relations EMPLOYEE and DEPARTMENT. Let the common attribute to the two relations be DEPTNUMBER. The two relations are shown later:

It is worth to note that Natural join operation is associative. i.e., If R, S, and T are three relations then:

$$R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$$

Employee				Department	
Employee ID	Designation	Dept Number		Dept name	Dept Number
C100	Lecturer	E1	\bowtie	Electrical	E1
C101	Assistant Professor	E2		Computer	C1
C102	Professor	C1			

Employee			\bowtie	Department	
Employee ID	Designation	Dept Number		Dept name	
C100	Lecturer	E1		Electrical	
C102	Professor	C1		Computer	

8.2 Equi Join

A special case of condition joins where the condition C contains only equality.

Example of Equi Join

Given the two relations STAFF and DEPT, produce a list of staff and the departments they work in.

STAFF			DEPT	
Staff No	Job	Dept	Dept	Name
1	salesman	100	100	marketing
2	draftsman	101	101	civil

Answer for the earlier query is equi-join of STAFF and DEPT:

STAFF EQUI JOIN DEPARTMENT				
Staff No	Job	dept	dept	Name
1	salesman	100	100	marketing
2	draftsman	101	101	civil

8.3 Theta Join

A conditional join in which we impose condition other than equality condition. If equality condition is imposed then theta join become equi join. The symbol θ stands for the comparison operator which could be $>$, $<$, $>=$, $<=$.

Expression of Theta Join:

$$\sigma_{\theta}(R \times S)$$

Illustration of Theta Join

To illustrate theta join consider two relations FRIENDS and OTHERS with the attributes Name and age.

FRIENDS	
Name	Age
Joe	4
Sam	9
Sue	10

OTHERS	
Alias	Size
Bob	8
Gim	10

Result of theta join			
Name	Age	Alias	Size
Joe	4	Bob	8
Sam	9	Gim	10
Sue	10		

8.4 Outer Join

In outer join, matched pairs are retained unmatched values in other tables are left null.

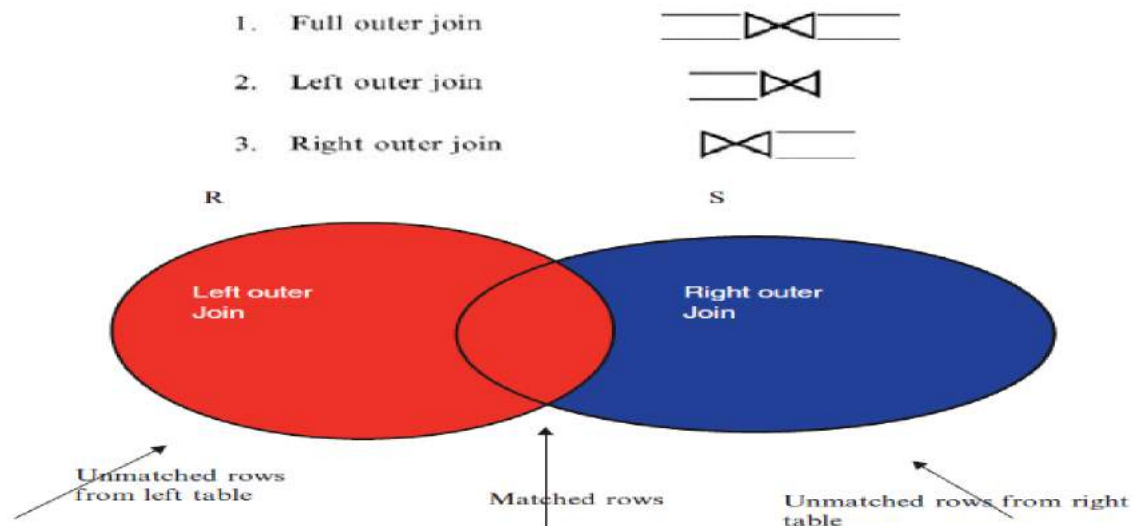


Fig. 7. Representation of left and right outer join

Types of Outer Join

The pictorial representation of the left and the right outer join of two relations R and S are shown in Fig. 7:

1. **Left Outer Join.** Left outer joins is a join in which tuples from R that do not have matching values in the common column of S are also included in the result relation.
2. **Right Outer Join.** Right outer join is a join in which tuples from S that do not have matching values in the common column of R are also included in the result relation.

3. Full Outer Join. Full outer join is a join in which tuples from R that do not have matching values in the common columns of S still appear and tuples in S that do not have matching values in the common columns of R still appear in the resulting relation.

9. Examples of Relational Algebra Queries

We will consider a number of example queries using the following schema:

Sailors(sid: integer, sname: string, rating: integer, age: real)

Boats(bid: integer, bname: string, color: string)

Reserves(sid: integer, bid: integer, day: dates)

```
CREATE TABLE Sailors ( sid INTEGER,  
                        sname CHAR(10),  
                        rating INTEGER,  
                        age REAL,  
                        PRIMARY KEY (sid))
```

```
CREATE TABLE Boats ( bid INTEGER,  
                      bname CHAR(10),  
                      color CHAR(10),  
                      PRIMARY KEY (bid))
```

```
CREATE TABLE Reserves ( sid INTEGER,  
                         bid INTEGER,  
                         day DATE,  
                         FOREIGN KEY (sid) REFERENCES Sailors  
                         FOREIGN KEY (bid) REFERENCES Boats )
```

We illustrate queries using the instances of tables: Sailors, Reserves, and Boats shown in Fig. 8:

Reserves			Sailors				Boats		
sid	bid	day	sid	sname	rating	age	bid	bname	color
22	101	10/10/98	22	Dustin	7	45.0	101	Interlake	blue
22	102	10/10/98	29	Brutus	1	33.0	102	Interlake	red
22	103	10/8/98	31	Lubber	8	55.5	103	Clipper	green
22	104	10/7/98	32	Andy	8	25.5	104	Marine	red
31	102	11/10/98	58	Rusty	10	35.0			
31	103	11/6/98	64	Horatio	7	35.0			
31	104	11/12/98	71	Zorba	10	16.0			
64	101	9/5/98	74	Horatio	9	35.0			
64	102	9/8/98	85	Art	3	25.5			
74	103	9/8/98	95	Bob	3	63.5			

Fig. 4 Sailors, Reserves, Boats tables

(Q1) Find the names of sailors who have reserved boat 103.

This query can be written as follows:

$$\pi_{sname}((\sigma_{bid=103}Reserves) \bowtie Sailors)$$

However, there are indeed several distinct ways to write a query in relational algebra. Here is another way to write this query:

$$\pi_{sname}(\sigma_{bid=103}(Reserves \bowtie Sailors))$$

In this version we first compute the natural join of Reserves and Sailors and then apply the selection and the projection.

(Q2) Find the names of sailors who have reserved a red boat.

$$\pi_{sname}((\sigma_{color='red'}Boats) \bowtie Reserves \bowtie Sailors)$$

An equivalent expression is:

$$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color='red'}Boats) \bowtie Reserves) \bowtie Sailors)$$

(Q3) Find the colors of boats reserved by Lubber.

$$\pi_{color}((\sigma_{sname='Lubber'}Sailors) \bowtie Reserves \bowtie Boats)$$

This query is very similar to the query we used to compute sailors who reserved red boats.

(Q4) Find the names of sailors who have reserved at least one boat.

$$\pi_{sname}(Sailors \bowtie Reserves)$$

The join of Sailors and Reserves creates an intermediate relation in which tuples consist of a Sailors tuple 'attached to' a Reserves tuple. A Sailors tuple appears in (some tuple of) this intermediate relation only if at least one Reserves tuple has the same sid value, that is, the sailor has made some reservation.

**References:**

1. S. Sumathi, S. Esakkirajan, "Fundamentals of Relational Database Management Systems", Springer, 2007.
2. Ramez Elmasri, Shamkant B. Navathe, "Fundamentals of Database Systems", 4th Edition, Addison Wesley, 2003.
3. C. J. Date, "An Introduction to Database Systems", 8th Edition, Addison Wesley, 2004.
4. Raghu Ramakrishnan, Johannes Gehrke, "Database Management Systems", 3rd Edition, McGraw Hill, 2003.

Introduction to Database Systems

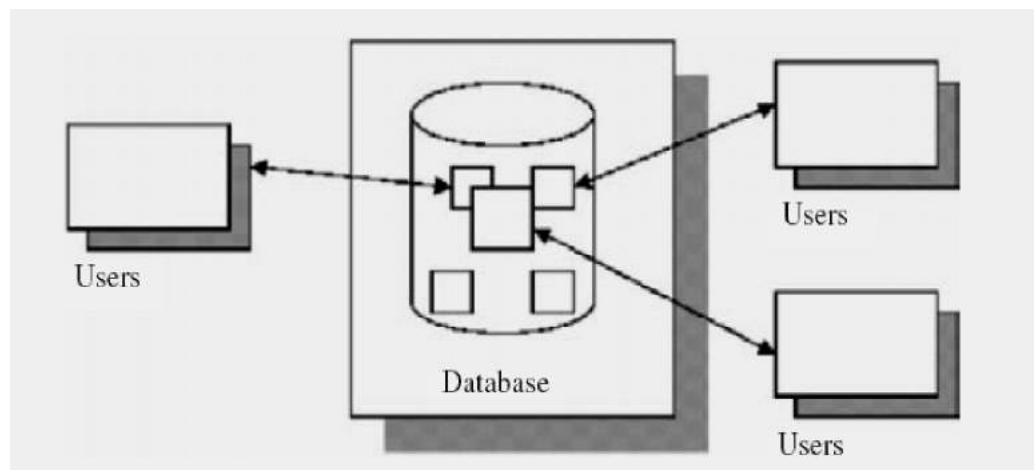
Information is a valuable resource to an organization. Computer software provides an efficient means of processing information, and database systems are becoming increasingly common means by which it is possible to store and retrieve information in an effective manner. This course provides comprehensive coverage of fundamentals of database management system. Relational databases are the most popular database management systems in the world and are supported by a variety of vendor implementations.

1. Data and Information

The data in DBMS can be broadly classified into two types, one is the collection of information needed by the organization and the other is “metadata” which is the information about the database.

2. Database

A database is a well-organized collection of data that are related in a meaningful way. The simplified view of database system is shown in Fig. (1). From this figure, it is clear that several users can access the data in the database.



3. Database Management System

The database management system (DBMS) is a software component that manages the database and shields users from low-level details (in particular, details of how the database is physically stored and accessed). A DBMS consists of collection of interrelated data and a set of programs to access that data. DBMS components are:

1. A collection of interrelated and persistent data. This part of DBMS is referred to as database (DB).
2. A set of application programs used to access, update, and manage data. This part constitutes data management system (MS).

A DBMS is general-purpose software i.e., not application specific. The same DBMS (e.g., Oracle, Sybase, etc.) can be used in railway reservation system, library management, university, etc.

A DBMS takes care of storing and accessing data, leaving only application specific tasks to application programs.

4. Objectives of DBMS

The main objectives of database management system are data availability, data integrity, data security, and data independence.

4.1 Data Availability

Data availability refers to the fact that the data are made available to wide variety of users in a meaningful format at reasonable cost so that the users can easily access the data.

4.2 Data Integrity

Data integrity refers to the correctness of the data in the database. In other words, the data available in the database is a reliable data.

4.3 Data Security

Data security refers to the fact that only authorized users can access the data. Data security can be enforced by passwords. If two separate users are accessing a particular data at the same time, the DBMS must not allow them to make conflicting changes.

4.4 Data Independence

DBMS allows the user to store, update, and retrieve data in an efficient manner. DBMS provides an “abstract view” of how the data is stored in the database. In order to store the information efficiently, complex data structures are used to represent the data. The system hides certain details of how the data are stored and maintained.

5. File-Based System

Prior to DBMS, file system provided by OS was used to store information. In a file-based system, we have collection of application programs that perform services for the end users. Each program defines and manages its own data. Consider University database, the University database contains details about student, faculty, lists of courses offered, and duration of course, etc.

In File-based processing for each database there is separate application program which is shown in Fig. (2).

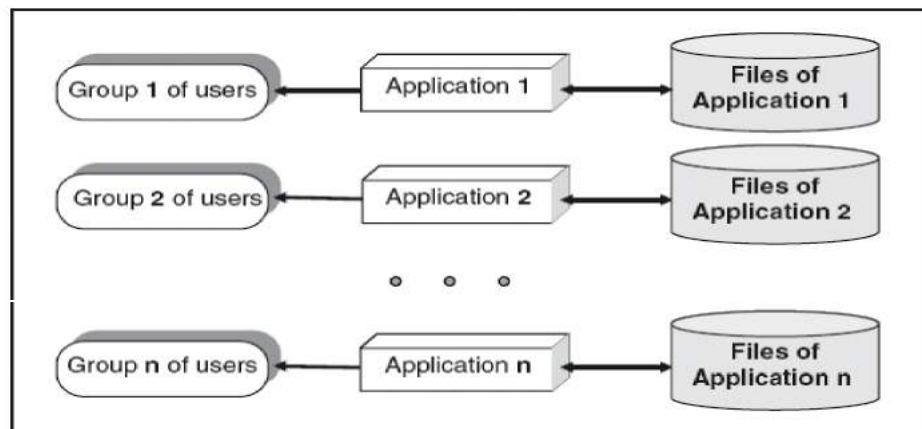


Fig. (2). File-based System

The information is stored in separate files and separate applications programs are written.

Drawbacks of File-Based System

The limitations of file-based approach are duplication of data, data dependence, incompatible file formats, separation, and isolation of data.

5.1 Duplication of Data

Duplication of data means same data being stored more than once. Duplication of data leads to wastage of storage space. Duplication of data can lead to loss of data integrity.

5.2 Data Dependence

Data dependence means the application program depends on the data. If some modifications have to be made in the data, then the application program has to be rewritten.

5.3 Incompatible File Formats

As file-based system lacks program data independence, the structure of the file depends on the application programming language. For example, the structure of the file generated by FORTRAN program may be different from the structure of a file generated by “C” program.

5.4 Separation and Isolation of Data

In file-based approach, data are isolated in separate files. Hence it is difficult to access data.

6. DBMS Approach

DBMS is software that provides a set of primitives for defining, accessing, and manipulating data. In DBMS approach, the same data are being shared by different application programs; DBMS approach of data access is shown in Fig. (3).

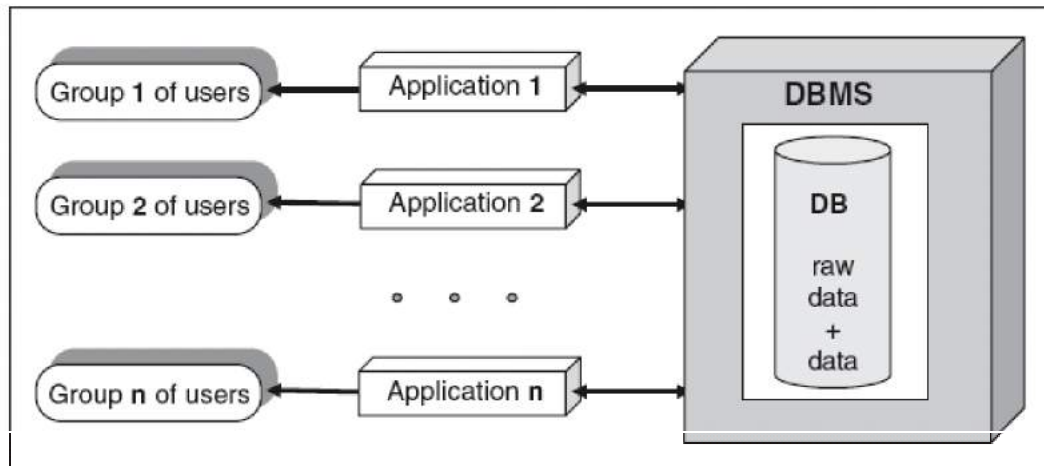


Fig. (3). Data access through DBMS

Advantages of DBMS

There are many advantages of database management system. Some of the advantages are listed later:

1. Centralized data management.
2. Data Independence.
3. Data Consistency.
4. Data integrity and security
5. The data can be shared.

6.1 Centralized Data Management (reducing redundancies)

In DBMS all files are integrated into one system thus reducing redundancies and making data management more efficient.

6.2 Data Independence

Data independence means that programs are isolated from changes in the way the data are structured and stored. In a database system, the database management system provides the interface between the application programs and the data. *Data independence is the immunity of application programs to changes in storage structures and access techniques.*

Physical data independence is the ability to modify physical schema without causing the conceptual schema or application programs to be rewritten. Logical data independence is the ability to modify the conceptual schema without having to change the external schemas or application programs.

6.3 Data Consistency

Data inconsistency means different copies of the same data will have different values. For example, consider a person working in a branch of an organization. The details of the person should be stored and maintained both in the branch office as well as in the main office in order to ensure data consistency.

6.4 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.

6.5 The data can be shared.

Sharing means not only that existing applications can share the data in the database. But also that new application can be developed to operate against that same data. In other words: it might be, possible to satisfy the data requirements of new applications without having to add any new data to the database.

7. Database Architecture

The different levels of database architecture are: Fig. (4)

1. The **Physical (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 **Logical (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. A high-level data model or an implementation data model can be used at this level.
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. A high-level data model or an implementation data model can be used at this level.

Notice that the three schemas are only *descriptions* of data; the only data that *actually* exists is at the physical level. In a DBMS based on the three-schema architecture, each user group refers only to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are called **mappings**.

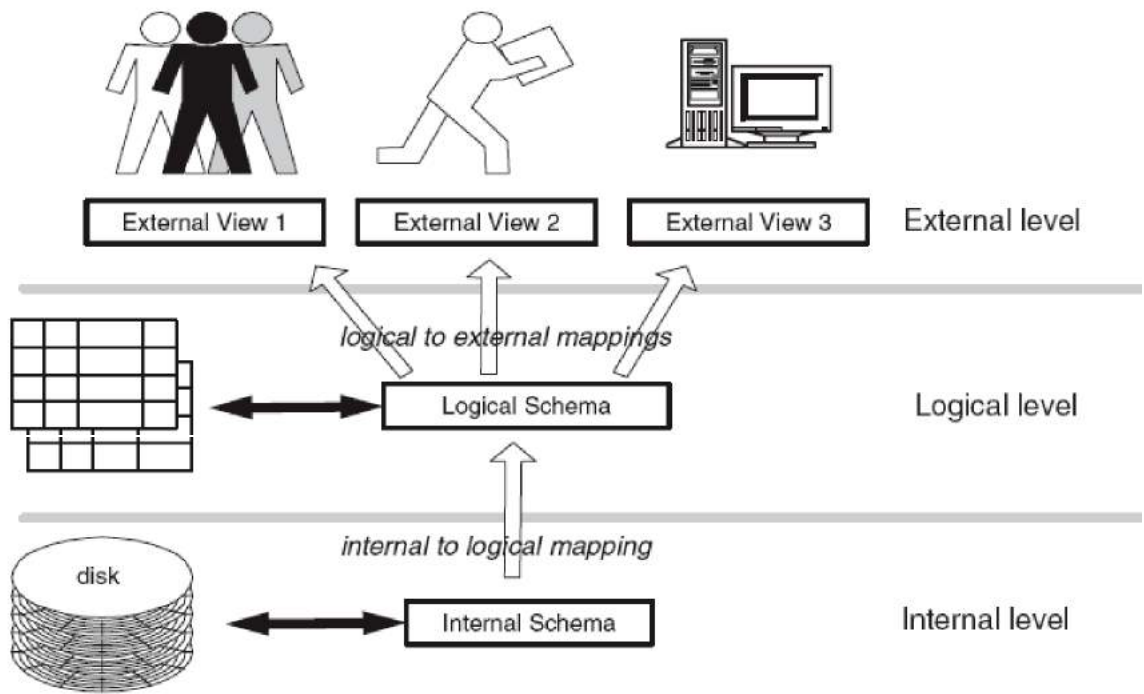


Fig. (4) Three levels of database architecture

The **conceptual/internal mapping** defines the correspondence between the conceptual view and the stored database.

An **external/conceptual mapping** defines the correspondence between an external view and the conceptual view.

The three-schema architecture can be used to explain the concept of **data independence**.

Data independence is accomplished because, when the schema is changed at some level, the schema at the next higher level remains unchanged; only the *mapping* between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.

Database Instances

Database change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an instance of the database.

Database Schema

The overall design of the database is called the database schema. A schema is a collection of named objects. Schemas provide a logical classification of objects in the database. A schema can contain tables, views, triggers, functions, packages, and other objects.

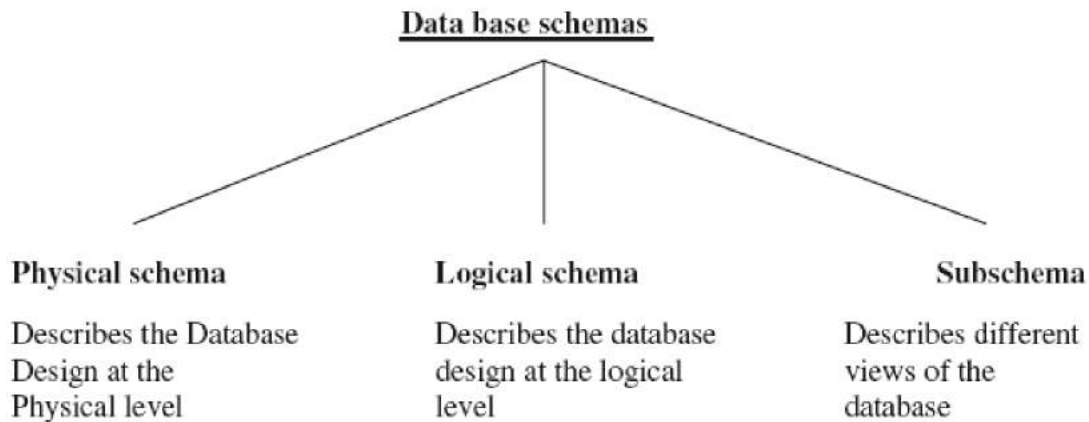


Figure 5 The three schema kinds (levels)

Data administrator is the person who decides what data should be kept in the database and (b) establishing the necessary policies for maintaining and dealing with that data.

The database administrator (DBA) is the person whose job it is to create the actual database and to implement the technical controls needed to enforce the various policy decisions made by the data administrator.

Responsibilities of Database Administrator (DBA)

The responsibility of the database administrator is to maintain the integrity, security, and availability of data. The responsibilities of the database administrator are summarized as follows:

1. Authorizing access to the database.
2. Coordinating and monitoring its use.
3. Acquiring hardware and software resources as needed.
4. Database Backup and recovery

People interacting with database

1. **Database designers** are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. These tasks are mostly undertaken before the database is actually implemented and populated with data.
2. **System analysts** determine the requirements of end users, especially naive and parametric end users, and develop specifications for canned transactions that meet these requirements.
3. **Application programmers** implement these specifications as programs; then they test, debug, document, and maintain these canned transactions. Such analysts and programmers (nowadays called **software engineers**) should be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks.

4. **Casual end users** occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests
5. **Naive or parametric end users:** Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates
6. **Sophisticated end users** include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS so as to meet their complex requirements.

Data Models

Data model is collection of conceptual tools for describing data, relationship between data, and consistency constraints. Data model describe the structure of the database. Three historically important data models are the hierarchical, network, and relational models. Together they are often referred to as the “basic” data models.

The constructs of the data model may be defined at many levels of abstraction.

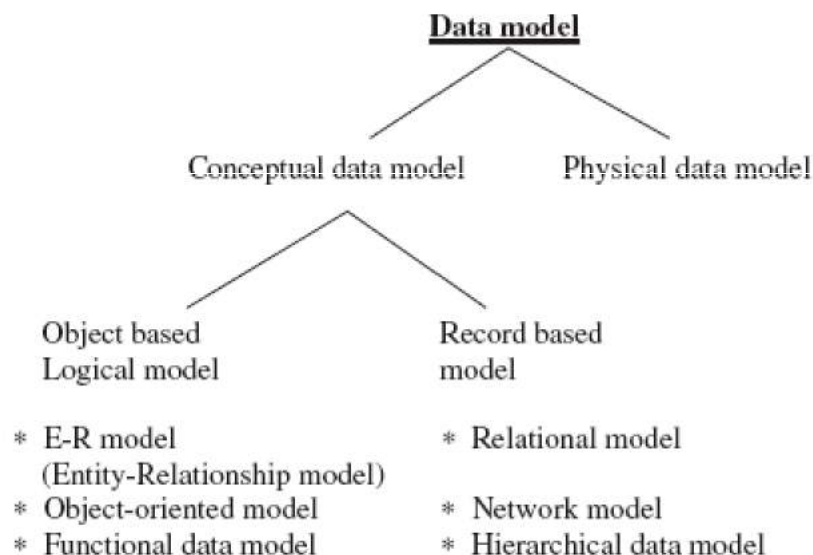


Fig. 6 Data models types

An implementation of a given data model is a physical realization on a real machine of the components of that model.

Data Dictionary

A data dictionary, also known as a “system catalog,” is a centralized store of information about the database. It contains information about the tables, the fields of tables, data types, primary keys, indexes, and so on. The information stored in the data dictionary is called the “Metadata.” Thus a data dictionary can be considered as a file that stores Metadata.

Metadata: The information (data) about the data in a database is called Metadata. The Metadata are available for query and manipulation, just as other data in the database.

1. OVERVIEW OF DATABASE DESIGN

The database design process can be divided into five steps. The ER model is most relevant to the first three steps:

- (1) **Requirements Analysis:** The very first step in designing a database application is to understand what data is to be stored in the database, we must find out what the users want from the database. Several methodologies have been proposed for organizing and presenting the information gathered in this step, and some automated tools have been developed to support this process.
- (2) **Conceptual Database Design:** This step is often carried out using the ER model. The information gathered in the requirements analysis step is used to develop a high-level description of the data to be stored in the database.
- (3) **Logical Database Design:** The task in the logical design step is to convert an ER schema into a relational database schema. We must choose a DBMS to implement our database design.

The remaining three steps of database design are briefly described below:

- (4) **Schema Refinement:** The fourth step in database design is to analyze the collection of relations in our relational database schema to identify potential problems, and to refine it. We discuss the theory of normalizing relations to ensure some desirable properties later.
- (5) **Physical Database Design:** In this step we must consider typical expected workloads that our database must support and further refine the database design to ensure that it meets desired performance criteria.

The entity-relationship (ER) data model allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design. The ER model is used in a phase called conceptual database design.

2. Data Modeling Using the Entity-Relationship Model

Conceptual modeling is an important phase in designing a successful database application. The entity-relationship (ER) data model allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design. The ER model is used in a phase called conceptual database design.

The main motivation for defining the ER model is to provide a high level model for conceptual database design. The ER model achieves a high degree of data independence which means that the database designer does not have to worry about the physical structure of the database.

2.1 The Building Blocks of an Entity-Relationship Diagram

The ER modeling can be carried out with the help of pictorial representation of entities, attributes, and relationships.

Entity

An entity is an object that exists and is distinguishable from other objects. The examples of entities are: a particular person, a particular department, a particular place.

Entity Type: An entity type or entity set is a collection of similar entities. Some examples of entity types are: students, courses, departments.

Relationship

A relationship is an association of entities where relationship type is a meaningful association between entity types. The examples of relationship types are:

- ⊙ Teaches is the relationship type between LECTURER and STUDENT.
- ⊕ Buying is the relationship between VENDOR and CUSTOMER.
- ⊕ Treatment is the relationship between DOCTOR and PATIENT.

Attributes

Entities are described in a database by a set of attributes. The following are example of attributes:

- ⊕ Brand, cost, and weight are the attributes of CELLPHONE.
- ⊕ Roll number, name, and grade are the attributes of STUDENT.
- ⊙ Data bus width, address bus width, and clock speed are the attributes of MICROPROCESSOR.

2.2 ER Diagram

The ER diagram is used to represent database schema. The elements in ER diagram are Entity, Attribute, and Relationship. See Fig. (1)

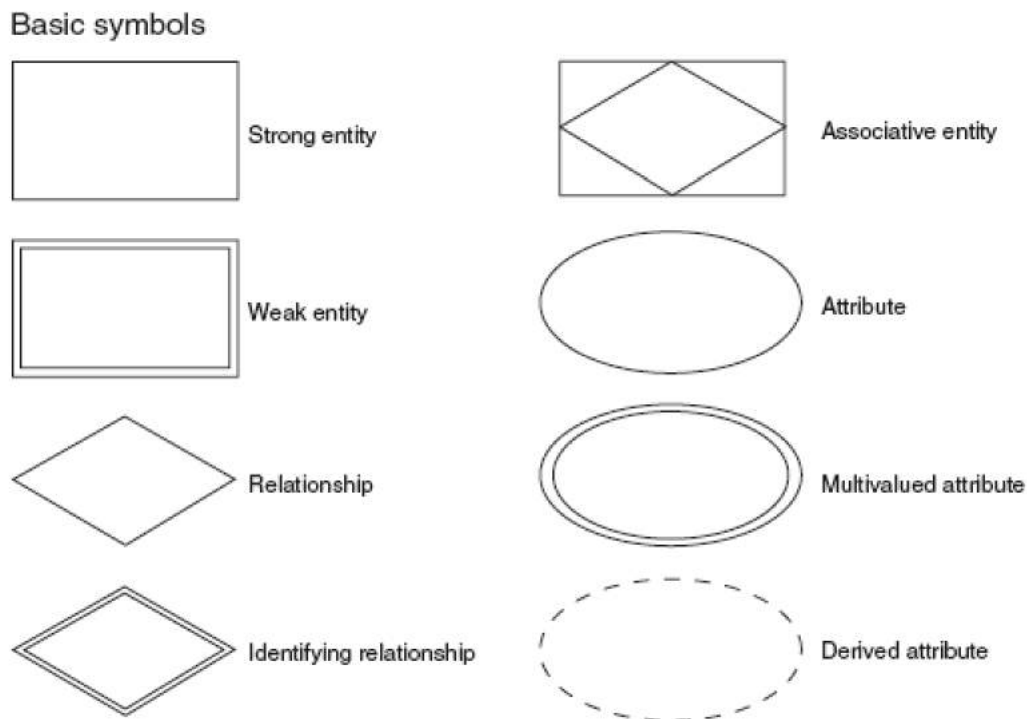


Fig. (1) ER diagram Symbols

- ❖ A rectangle represents an entity set.
- ❖ An ellipse represents an attribute.
- ❖ A diamond represents a relationship.
- ❖ Lines represent linking of attributes to entity sets and of entity sets to relationship sets.

Example of ER diagram

Let us consider a simple ER diagram as shown in Fig. (2). In the ER diagram the two entities are STUDENT and CLASS. Two simple attributes which are associated with the STUDENT are Roll number and the name. The attributes associated with the entity CLASS are Subject Name and Hall Number. The relationship between the two entities STUDENT and CLASS is Attends.

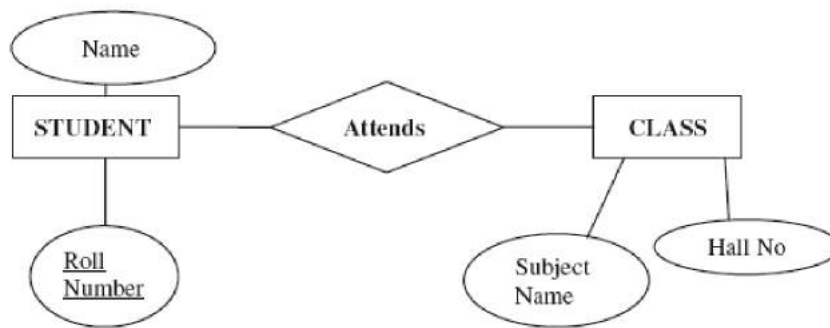


Fig. (2). ER diagram

3. Classification of Entity Sets

Entity sets can be broadly classified into: see Fig. (3).

1. Strong entity.
2. Weak entity.
3. Associative entity.

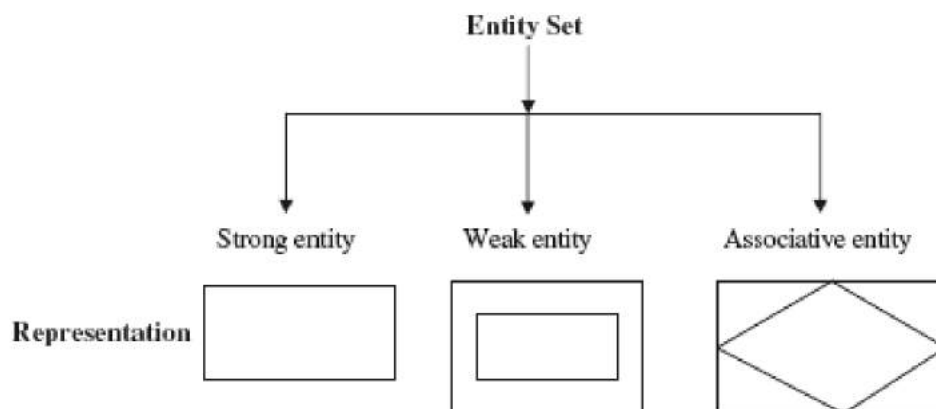
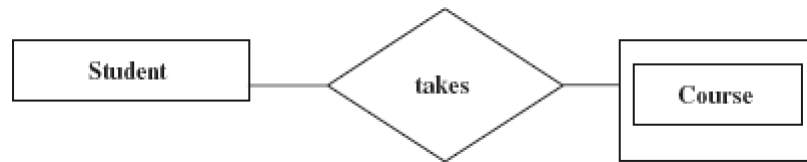


Fig. (3) Entity sets

3.1 Strong Entity

Strong entity is one whose existence does not depend on other entity.

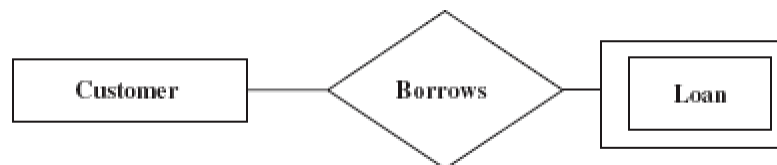
Example: Consider the example, student takes course. Here student is a strong entity. In this example, course is considered as weak entity because, if there are no students to take a particular course, then that



3.2 Weak Entity

Weak entity is one whose existence depends on other entity. In many cases, weak entity does not have primary key.

Example: Consider the example, customer borrows loan. Here loan is a weak entity. For every loan, there should be at least one customer. Here the entity loan depends on the entity customer hence loan is a weak entity.



4. Attribute Classification

Attribute is used to describe the properties of the entity. This attribute can be broadly classified based on value and structure. See Fig. (4)

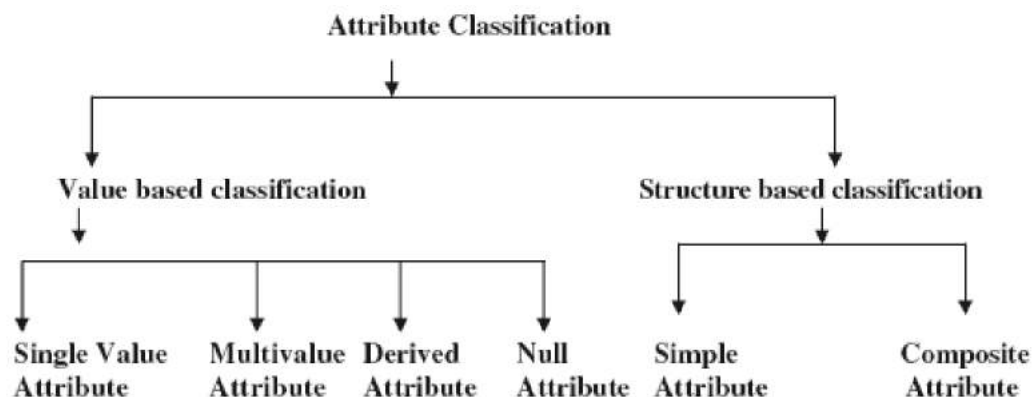


Fig. (4) Attribute Classification

4.1 Single value attribute

Single value attribute means, there is only one value associated with that attribute. One example of single value attribute is age of a person.

4.2 Multivalued attribute

In the case of *multivalued* attribute; Fig. (5), more than one value will be associated with that attribute. Consider an entity EMPLOYEE. An Employee can have many skills; hence skills associated to an employee are a multivalued attribute.

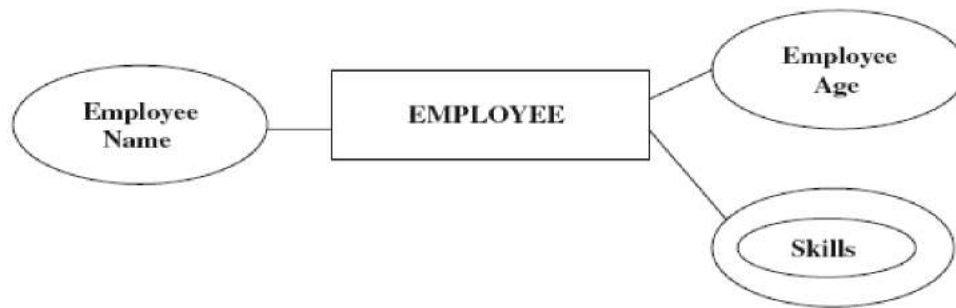
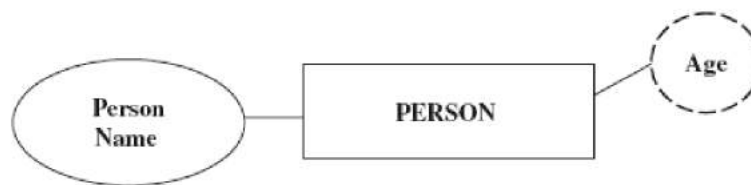


Fig. (5) A multivalued attribute (skills)

4.3 Derived Attribute

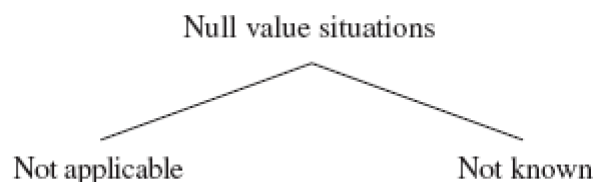
The value of the derived attribute can be derived from the values of other related attributes or entities.

Example: Age of a person can be derived from the date of birth of the person. In this example, age is the derived attribute.



4.4 Null Value Attribute

In some cases, a particular entity may not have any applicable value for an attribute. For such situation, a special value called null value is created.



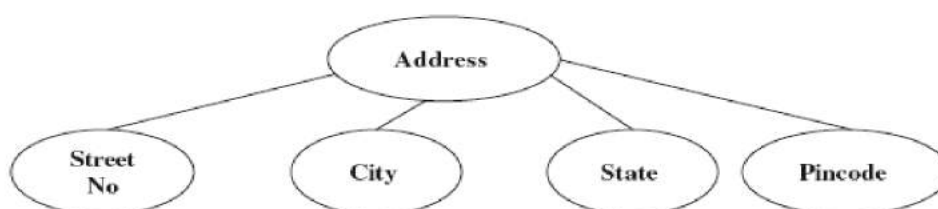
Example

In application forms, there is one column called phone no. if a person does not have a phone then a null value is entered in that column.

4.5 Composite Attribute

Composite attribute is one which can be further subdivided into simple attributes.

Example: Consider the attribute “address” which can be further subdivided into Street name, City, and State.

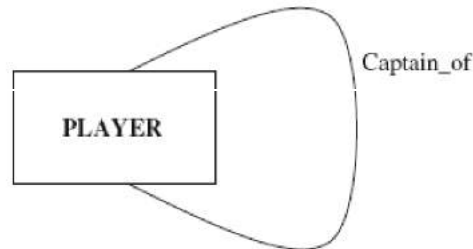


5. Relationship Degree

Relationship degree refers to the number of associated entities. The relationship degree can be broadly classified into unary, binary, and ternary relationship.

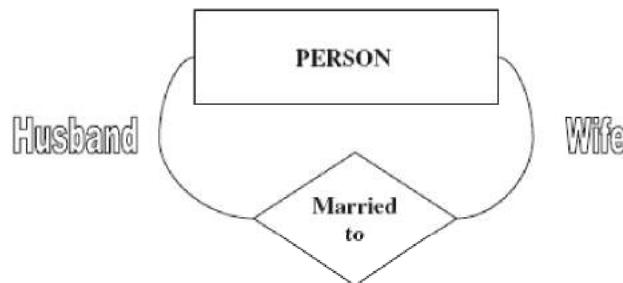
5.1 Unary Relationship

The unary relationship is otherwise known as recursive relationship. In the unary relationship the number of associated entity is one. An entity related to itself is known as recursive relationship.



Example:

In this example, Husband and wife are referred as roles.



5.2 Binary Relationship

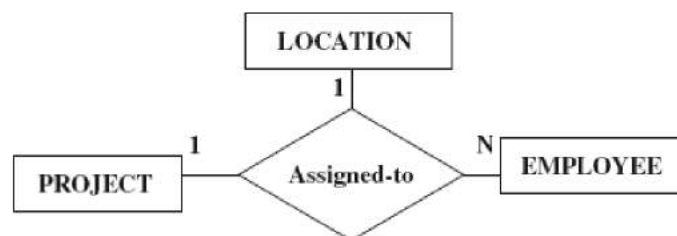
In a binary relationship, two entities are involved. Consider the example; each staff will be assigned to a particular department. Here the two entities are STAFF and DEPARTMENT.



5.3 Ternary Relationship

In a ternary relationship, three entities are simultaneously involved. Ternary relationships are required when binary relationships are not sufficient.

Example: Consider the example of employee assigned a project. Here we are considering three entities EMPLOYEE, PROJECT, and LOCATION. The relationship is “assigned-to.” Many employees will be assigned to one project hence it is an example of one-to-many relationship.



6. Relationship Cardinality

Relationship is an association among one or more entities. This relationship can be broadly classified into one-to-one relation, one-to-many relation, many-to-many relation and recursive relation.

6.1 One-to-One Relationship Type

One-to-one relationship is a special case of one-to-many relationship. True one-to-one relationship is rare. The relationship between the President and the country is an example of one-to-one relationship. For a particular country there will be only one President.

6.2 One-to-Many Relationship Type

The relationship that associates one entity to more than one entity is called one-to-many relationship. Example of one-to-many relationship is Country having states. For one country there can be more than one state.

6.3 Many-to-One Relationship Type


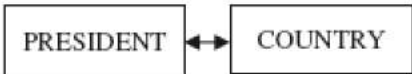

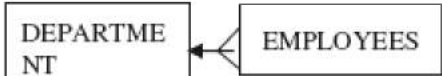

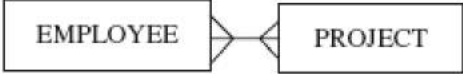

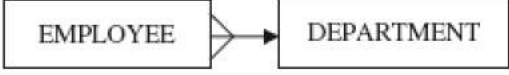
The relationship between EMPLOYEE and DEPARTMENT is an example of many-to-one relationship. There may be many EMPLOYEES working in one DEPARTMENT.

6.4 Many-to-Many Relationship Type

The relationship between EMPLOYEE entity and PROJECT entity is an example of many-to-many relationship. Many employees will be working in many projects hence the relationship between employee and project is many-to-many relationship.

The four relationship types are summarized and shown in Table (1).

Table (1). Relationship types

Relationship type	Representation	Example
One-to-one		
One-to-many		
Many-to-many		
Many-to-one		

7. An Example Relation Schema

In this section we describe an example database application, called COMPANY, which serves to illustrate the ER model concepts and their use in schema design. We list the data requirements for the database here, and then we create its conceptual schema step-by-step as we introduce the modeling concepts of the ER model. The COMPANY database keeps track of a company's employees, departments, and projects.

1. The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. A department may have several locations.
2. A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
3. We store each employee's name, social security number, address, salary, sex, and birth date. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department.
4. We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date, and relationship to the employee.

7.1 Entities and attributes of the COMPANY Database

1. An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager StartDate.
2. An entity type PROJECT with attributes Name, Number, Location, and Controlling Department.
3. An entity type EMPLOYEE with attributes Name, SSN (for social security number), Sex, Address, Salary, BirthDate, Department, and Supervisor.
4. An entity type DEPENDENT with attributes Employee, Dependent Name, Sex, BirthDate, and Relationship (to the employee).

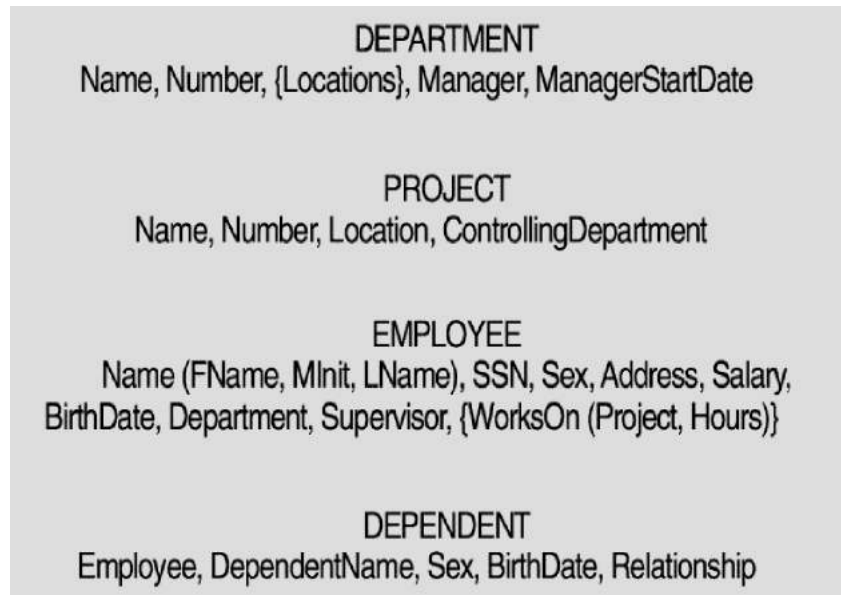


Figure (6) Initial Conceptual Design of the COMPANY Database

7.2 ER Design for the COMPANY Database

1. **MANAGES**, a 1:1 relationship type between **EMPLOYEE** and **DEPARTMENT**.
2. **WORKS_FOR**, a 1:N relationship type between **DEPARTMENT** and **EMPLOYEE**.
3. **CONTROLS**, a 1:N relationship type between **DEPARTMENT** and **PROJECT**.
4. **SUPERVISION**, a 1:N relationship type between **EMPLOYEE** (in the supervisor role) and **EMPLOYEE** (in the supervisee role).
5. **WORKS_ON**, determined to be an M:N relationship type between **EMPLOYEE** and **PROJECT**. The rule is that an employee can work on several projects and a project can have several employees. This relation can be resolved using attribute Hours.
6. **DEPENDENTS_OF**, a 1:N relationship type between **EMPLOYEE** and **DEPENDENT**, which is also the identifying relationship for the weak entity type **DEPENDENT**.

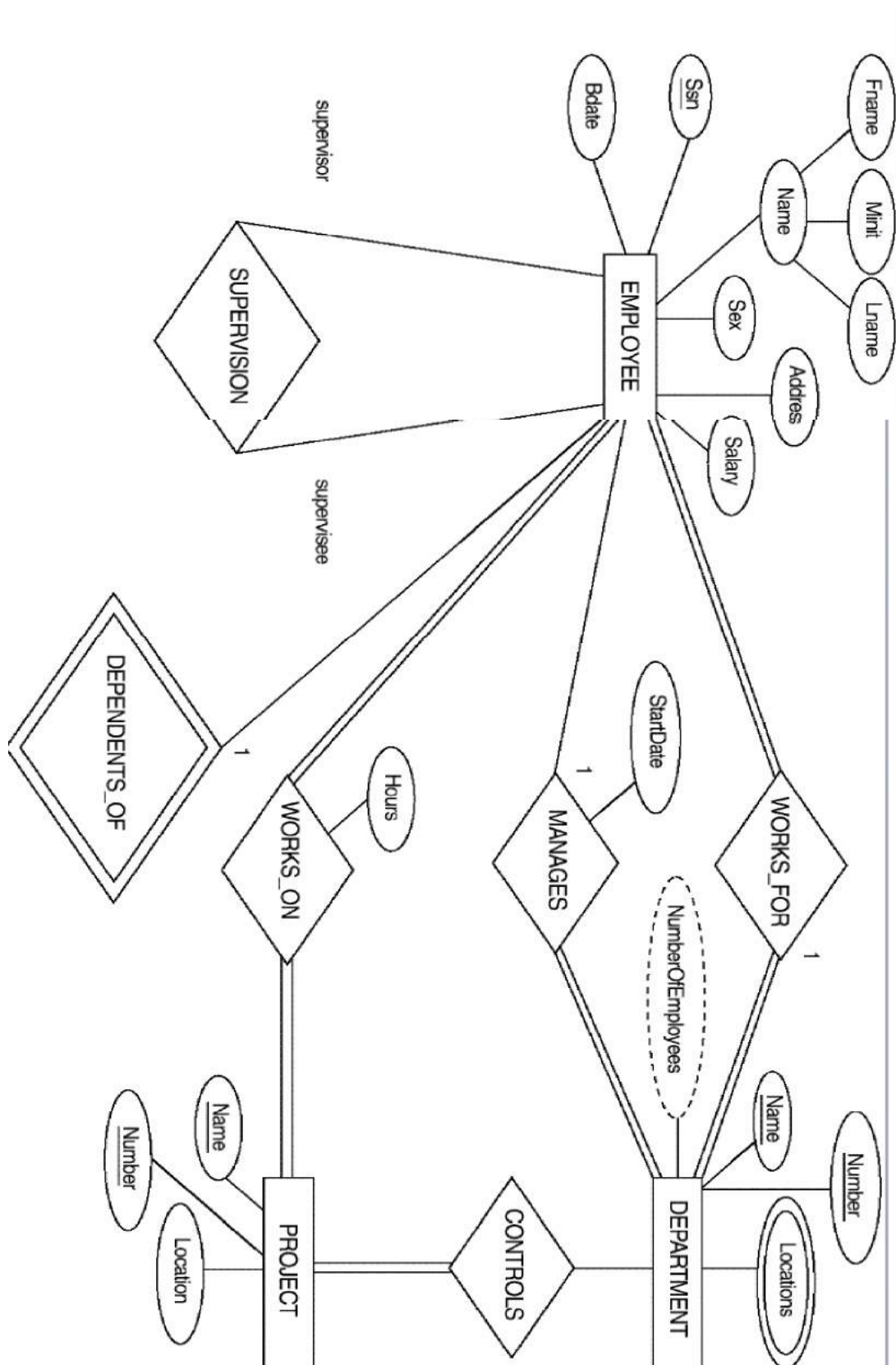


Figure (7) ER schema diagram for the COMPANY database.

1. Relational Model

E.F. Codd had written an article “A relational model for large shared data banks” in June 1970. His work triggered people to work in relational model. It gave rise to major developments such as a structured query language called SQL which has since become an ISO standard. Various commercial relational DBMS products were developed during the 1980s such as DB2, and Oracle. In relational data model the data are stored in the form of tables.

In 1985, Codd published a list of rules that became a standard way of evaluating a relational system. After publishing the original article, Codd stated that there are no systems that will satisfy every rule. Nevertheless the rules represent relational ideal and remain a goal for relational database designers.

1.1 Relational Model Properties

The relational model uses a collection of tables to represent both data and the relationships among those data. Tables are logical structures maintained by the database manager. The relational model is a combination of Structural, Integrity, and Manipulative components.

The relational algebra and relational calculus are the tools used to manipulate data in the database. Thus relational model has a strong mathematical background. The key features of relational data model are as follows:

- ⊕ Each row in the table is called tuple.
- Each column in the table is called attribute.
- The intersection of row with the column will have data value.
- ⊕ In relational model rows can be in any order.
- ⊕ In relational model attributes can be in any order.
- By definition, all rows in a relation are distinct. No two rows can be exactly the same.
- Relations must have a key. Keys can be a set of attributes.
- ⊕ For each column of a table there is a set of possible values called its domain. The domain contains all possible values that can appear under that column.
- *Domain* is the set of valid values for an attribute.
- **Degree** of the relation is the number of attributes (columns) in the relation.
- ⊕ **Cardinality** of the relation is the number of tuples (rows) in the relation.

The terms commonly used by user, model, and programmers are given here.

USER	MODEL	PROGRAMMER
Row	Tuple	Record
Column	Attribute	Field
Table	Relation	File

1.2 Relation Schema and Relation Instance

For a table to be relation, the following rules hold. The intersection row with the column should contain single value (atomic value).

- All entries in a column are of same type.
- ⊕ Each column has a unique name (column order not significant).
- ⊕ No two rows are identical (row order not significant).

The main construct for representing data in the relational model is a relation. A relation consists of a **relation schema** and a **relation instance**. The relation instance is a table, and the relation schema describes the column heads for the table. The schema specifies the relation's name, the name of each field (or column, or attribute), and the domain of each field.

We use the example of student information in a university database to illustrate the parts of a relation schema:

`Students(sid: string, name: string, login: string, age: integer, gpa: real)`

An instance of a relation is a set of tuples, also called records, in which each tuple has the same number of fields as the relation schema. A relation instance can be thought of as a table in which each tuple is a row, and all rows have the same number of fields. An instance of the Students relation appears in Figure (1). The instance S1 contains six tuples and has, as we expect from the schema, five fields.

The diagram shows a table representing an instance of the Students relation. Above the table, the text 'FIELDS (ATTRIBUTES, COLUMNS)' has arrows pointing to each of the five column headers. To the left of the table, the text 'Field names' has a wavy arrow pointing to the column headers. To the left of the table, the text 'TUPLES (RECORDS, ROWS)' has arrows pointing to each of the six data rows.

FIELDS (ATTRIBUTES, COLUMNS)				
<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure (1) An Instance S1 of the Students Relation

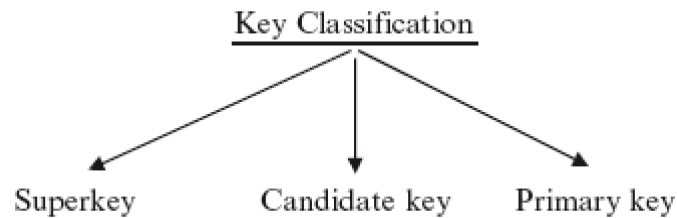
In this earlier relation:

- ⊕ The degree of the relation (i.e., is the number of column in the relation) = 5.
- ⊕ The cardinality of the relation (i.e., the number of rows in the relation) = 6.

A relational database is a collection of relations with distinct relation names. The relational database schema is the collection of schemas for the relations in the database.

2. Concept of Key

Key is an attribute or group of attributes, which is used to identify a row in a relation. Key can be broadly classified into (1) Superkey (2) Candidate key, and (3) Primary key



Superkey

A superkey is a subset of attributes of an entity-set that uniquely identifies the entities. Superkeys represent a constraint that prevents two entities from ever having the same value for those attributes.

Candidate Key

Candidate key is a minimal superkey. A candidate key for a relation schema is a minimal set of attributes whose values uniquely identify tuples in the corresponding relation.

Primary Key

The primary key is a designated candidate key. It is to be noted that the primary key should not be null.

Example:

Consider the employee relation, which is characterized by the attributes, employee ID, employee name, employee age, employee experience, employee salary, etc. In this employee relation:

- ⊙ Superkeys can be [employee ID, employee name], [employee ID], [employee name], etc.
- ⊕ Candidate keys can be [employee ID], [employee name].
- ⊖ Primary key is [employee ID].

Note: If we declare a particular attribute as the primary key, then that attribute value cannot be NULL. Also it has to be distinct.

Foreign Key

Foreign key is set of fields or attributes in one relation that is used to “refer” to a tuple in another relation.

3. Relational Integrity

Data integrity constraints refer to the accuracy and correctness of data in the database. Data integrity provides a mechanism to maintain data consistency for operations. The different types of data integrity constraints are Entity, NULL, Domain, and Referential integrity.

3.1 Entity Integrity

Entity integrity implies that a primary key cannot accept null value. Consider the entity PLAYER; the attributes of the entity PLAYER are: Id, Name, Age, Nation, and Rank. In this example, let us consider PLAYER's Id as the primary key. We cannot insert any data in the relation PLAYER without entering the Id of the player. This implies that primary key cannot be null.

3.2 Null Integrity

Null implies that the data value is not known temporarily. Consider the relation PERSON. The attributes of the relation PERSON are name, age, and salary. The age of the person cannot be NULL (according to the rules of the company).

3.3 Domain Integrity Constraint

Domain refers to the set of all possible values that attribute can take. The domain integrity constraints are used to specify the valid values that a column defined over the domain can take. We can define the valid values by listing a set of values, a range of values, or an expression.

Example:

The age of the person cannot have any letter from the alphabet. The age should be a numerical value.

Example:

Consider the relation APPLICANT. Here APPLICANT refers to the person who is applying for job. The sex of the applicant should be either male (M) or female (F). Any entry other than M or F violates the domain constraint.

3.4 Referential Integrity

In the relational data model, associations between tables are defined through the use of foreign keys. A referential integrity is a rule that states that either each foreign key value must match a primary key value in another relation or the foreign key value must be null. There can be situations where a relationship does not exist for a particular instance, in which case the foreign key is null.

Referential Integrity Constraints

According to referential integrity constraint, when a foreign key in one relation references primary key in another relation, the foreign key value must match with the primary key value. In other words, the referential integrity says "pointed to" information must exist.

Example

In order to understand referential constraint, consider two relation DEPARTMENT and EMPLOYEE. Here the DEPARTMENT relation forms the parent table. The meaning is the DEPARTMENT table contains the primary key. The relation EMPLOYEE forms the child table. The meaning is the relation EMPLOYEE has foreign key which references to primary key in DEPARTMENT table. The following Figure shows parent-child relationship.

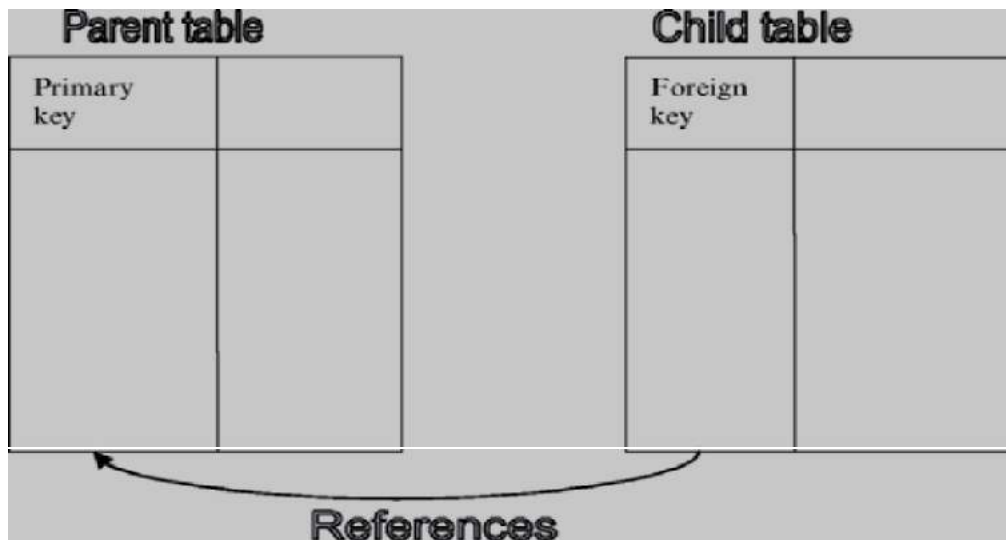


Fig. Primary key and foreign key relationship

In our example, the relation DEPARTMENT is the parent table which holds the parent table, and the relation EMPLOYEE forms the child table which has foreign key which references primary key in DEPARTMENT table. It is to be noted that the parent table should be created first, then the child table.

DEPARTMENT			EMPLOYEE		
DeptID	Dname	Location	EID	DID	Ename
D100	electrical	B	E201	D100	Raman
D101	civil	A	E202	D101	Ravi
D102	computer	C	E203	D101	Krishnan

Fig. Department and Employee tables' data

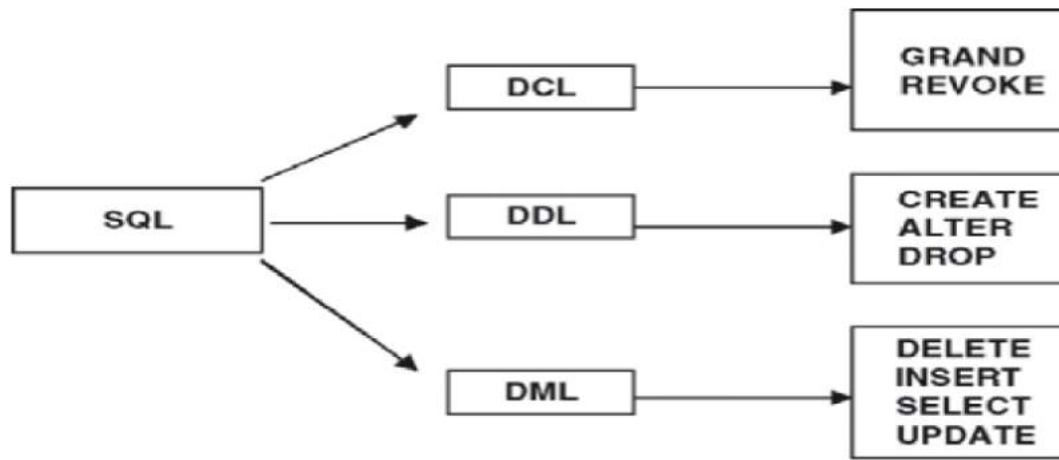
Structured Query Language

The Structured Query Language (SQL) is a relational database language. SQL commands consist of English-like statements which are used to query, insert, update, and delete data. SQL commands can be classified in to three types:

1. Data Definition Language commands (DDL): DDL commands are used to define a database, including creating, altering, and dropping tables and establishing constraints.
2. Data Manipulation Language commands (DML): DML commands are used to maintain and query a database, including updating, inserting, modifying, and querying data.

3. Data Control Language commands (DCL): DCL commands are used to control a database including administering privileges and saving of data.

The classification of commands in SQL is shown below:



Creating and Modifying Relations Using SQL:

The SQL language standard uses the word table to denote relation:

```
CREATE TABLE Students ( sid   CHAR(20),
                        name CHAR(30),
                        login CHAR(20),
                        age   INTEGER,
                        honorsCHAR(10) NOT NULL,
                        gpa   REAL)
PRIMARY KEY (sid),
FOREIGN KEY (honors) REFERENCES Courses (cid))
```

```
CREATE TABLE Courses (cid   CHAR(10),
                       cname CHAR(10),
                       creditsINTEGER,
                       grader CHAR(20) NOT NULL,
                       PRIMARY KEY (cid)
                       FOREIGN KEY (grader) REFERENCES Students (sid))
```

1. Transforming ER Diagram to Tables

To implement the database, it is necessary to use the relational model. There is a simple way of mapping from ER model to the relational model. There is almost one-to-one correspondence between ER constructs and the relational ones.

1.1 Mapping Algorithm

The mapping algorithm gives the procedure to map ER diagram to tables. The rules in mapping algorithm are given as:

1. For each strong entity type say E, create a new table. The columns of the table are the attribute of the entity type E.
2. For each weak entity W that is associated with only one 1–1 identifying owner relationship, identify the table T of the owner entity type. Include as columns of T, all the simple attributes and simple components of the composite attributes of W.
3. For each weak entity W that is associated with a 1–N or M–N identifying relationship, create a new table T and include as its columns, all the simple attributes and simple components of the composite attributes of W. Also form its primary key by including as a foreign key in R, the primary key of its owner entity.
4. For each binary 1–1 relationship type R, identify the tables S and T of the participating entity types. Choose one, say S; include as foreign key in S, the primary key of T. Include as columns of S, all the simple attributes and simple components of the composite attributes of R.
5. For each binary 1–N relationship type R, identify the table S, which is at N side and T of the participating entities. Include as a foreign key in S, the primary key of T. Also include as columns of S, all the simple attributes and simple components of composite attributes of R.
6. For each M–N relationship type R, create a new table T and include as columns of T, all the simple attributes and simple components of composite attributes of R. Include as foreign keys, the primary keys of the participating entity types. Specify as the primary key of T, the list of foreign keys.
7. For each multivalued attribute, create a new table T and include as columns of T, the simple attribute or simple components of the attribute A. Include as foreign key, the primary key of the entity or relationship type that has A. Specify as the primary key of T, the foreign key and the columns corresponding to A.

Regular Entity

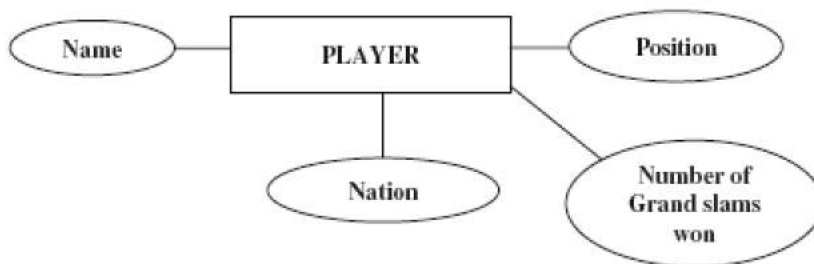
Regular entities are entities that have an independent existence and generally represent real-world objects such as persons and products. Regular entities are represented by rectangles with a single line.

1.2 Mapping Regular Entities

1. Each regular entity type in an ER diagram is transformed into a relation. The name given to the relation is generally the same as the entity type.
2. Each simple attribute of the entity type becomes an attribute of the relation.
3. The identifier of the entity type becomes the primary key of the corresponding relation.

Example:

Mapping regular entity type tennis player



This diagram is converted into corresponding table as Here,

Player Name	Nation	Position	Number of Grand slams won
Roger Federer	Switzerland	1	5
Roddick	USA	2	4

- ➡ Entity name = Name of the relation or table.
- ➡ In our example, the entity name is PLAYER which is the name of the table
- ➡ Attributes of ER diagram=Column name of the table.

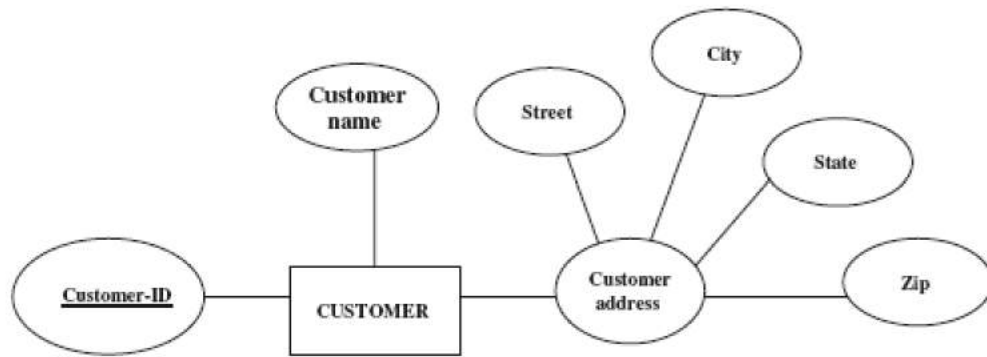
In our example the Name, Nation, Position, and Number of Grand slams won which forms the column of the table.

1.3 Converting Composite Attribute in an ER Diagram to Tables

When a regular entity type has a composite attribute, only the simple component attributes of the composite attribute are included in the relation.

Example:

In this example the composite attribute is the Customer address, which consists of Street, City, State, and Zip.



CUSTOMER

<u>Customer-ID</u>	Customer name	Street	City	State	Zip
--------------------	---------------	--------	------	-------	-----

When the regular entity type contains a multivalued attribute, two new relations are created. The first relation contains all of the attributes of the entity type except the multivalued attribute.

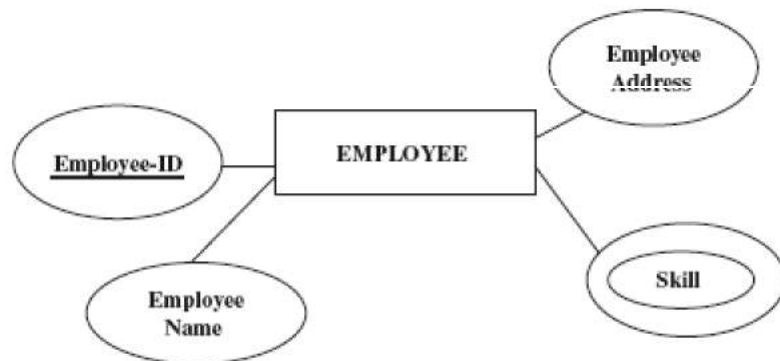
The second relation contains two attributes that form the primary key of the second relation. The first of these attributes is the primary key from the first relation, which becomes a foreign key in the second relation. The second is the multivalued attribute.

1.4 Mapping Multivalued Attributes in ER Diagram to Tables

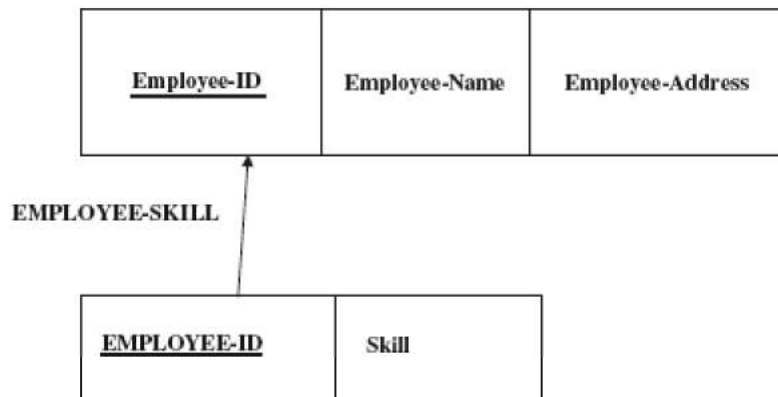
A multivalued attribute is having more than one value. One way to map a multivalued attribute is to create two tables.

Example:

In this example, the skill associated with the EMPLOYEE is a multivalued attribute, since an EMPLOYEE can have more than one skill as fitter, electrician, turner, etc.



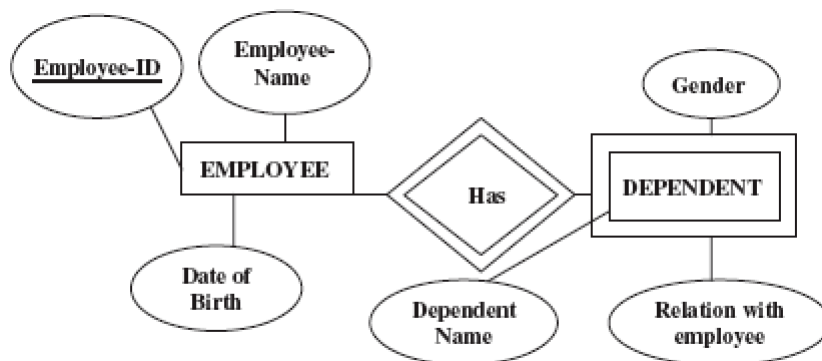
EMPLOYEE



1.5 Converting “Weak Entities” in ER Diagram to Tables

Weak entity type does not have an independent existence and it exists only through an identifying relationship with another entity type called the owner. For each weak entity type, create a new relation and include all of the simple attributes as attributes of the relation. Then include the primary key of the identifying relation as a foreign key attribute to this new relation.

The primary key of the new relation is the combination of the primary key of the identifying and the partial identifier of the weak entity type. In this example DEPENDENT is weak entity.



The corresponding table is given by

EMPLOYEE

<u>Employee-ID</u>	Employee-Name	Date of Birth
--------------------	---------------	---------------

DEPENDENT

Dependent-Name	Gender	<u>Employee-ID</u>	Relation with Employee
----------------	--------	--------------------	------------------------

1.6 Converting Binary Relationship to Table

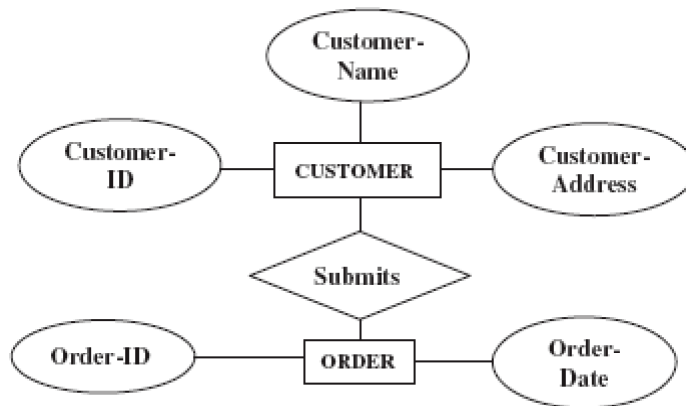
A relationship which involves two entities can be termed as binary relationship. This binary relationship can be one-to-one, one-to-many, many-to-one, and many-to-many.

Mapping one-to-Many Relationship

For each 1–M relationship, first create a relation for each of the two entity type's participation in the relationship.

Example:

One customer can give many orders. Hence the relationship between the two entities CUSTOMER and ORDER is one-to-many relationship. In one-to-many relationship, include the primary key attribute of the entity on the one-side of the relationship as a foreign key in the relation that is on the many side of the relationship.



Here we have two entities CUSTOMER and ORDER. The relationship between CUSTOMER and ORDER is one-to-many. For two entities CUSTOMER and ORDER, two tables namely CUSTOMER and ORDER are created as shown later. The primary key CUSTOMER ID in the CUSTOMER relation becomes the foreign key in the ORDER relation.

CUSTOMER

<u>Customer-ID</u>	Customer-Name	Customer-Address
--------------------	---------------	------------------

ORDER

<u>Order-ID</u>	Order-Date	Customer-ID
-----------------	------------	-------------

Binary one-to-one relationship can be viewed as a special case of one-to-many relationships. The process of mapping one-to-one relationship requires two steps. First, two relations are created, one for

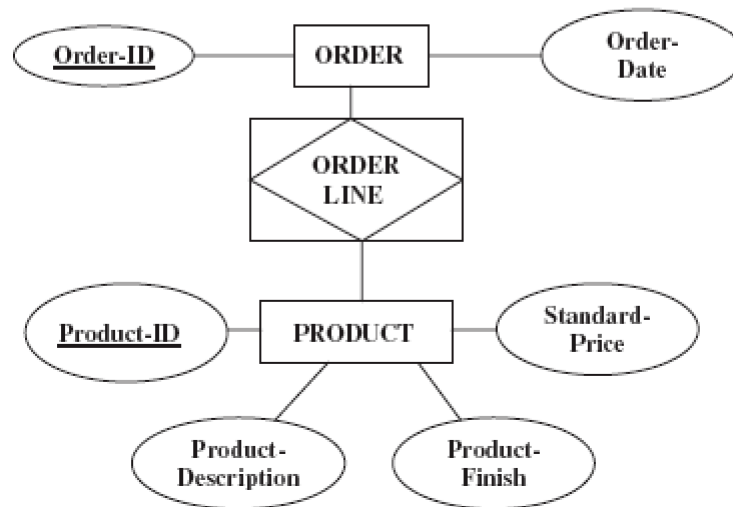
each of the participating entity types. Second, the primary key of one of the relations is included as a foreign key in the other relation.

1.7 Mapping Associative Entity to Tables

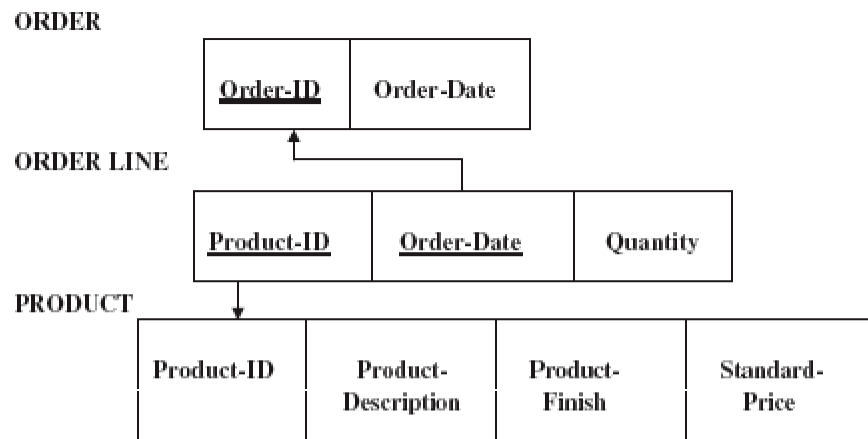
Many-to-many relationship can be modeled as an associative entity in the ER diagram.

Example: (Without Identifier)

Here the associative entity is ORDERLINE, which is without an identifier. That is the associative entity ORDERLINE is without any key attribute.



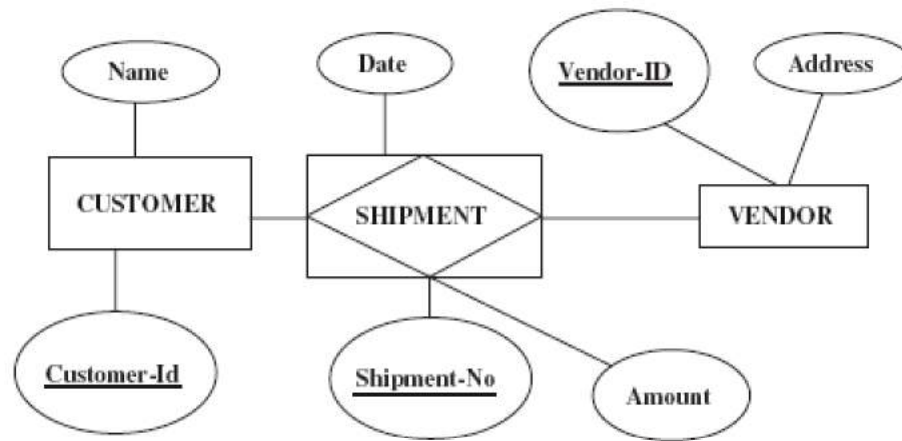
The first step is to create three relations, one for each of the two participating entity types and the third for the associative entity. The relation formed from the associative entity is associative relation.



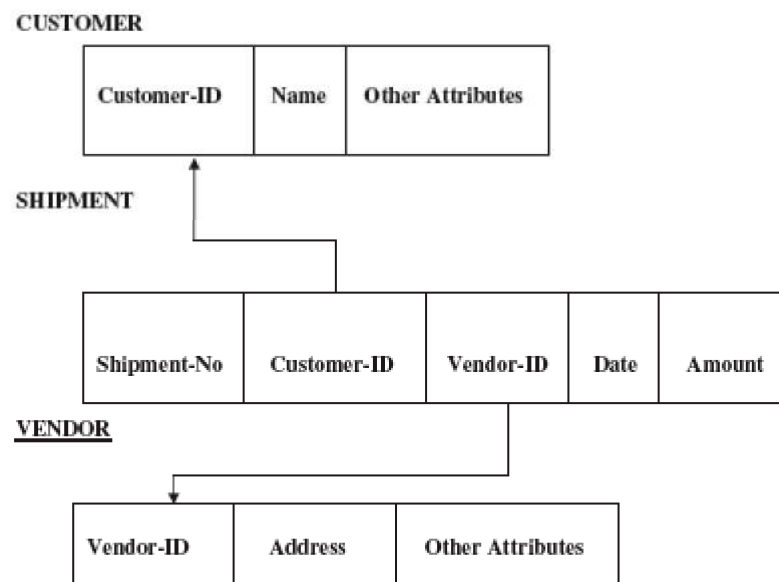
Example: (With Identifier)

Sometimes data models will assign an identifier (surrogate identifier) to the associative entity type on the ER diagram. There are two reasons to motivate this approach:

1. The associative entity type has a natural identifier that is familiar to end user.
2. The default identifier may not uniquely identify instances of the associative entity.



- a-** Shipment-No is a natural identifier to end user.
- b-** The default identifier consisting of the combination of Customer-ID and Vendor-ID does not uniquely identify the instances of SHIPMENT.



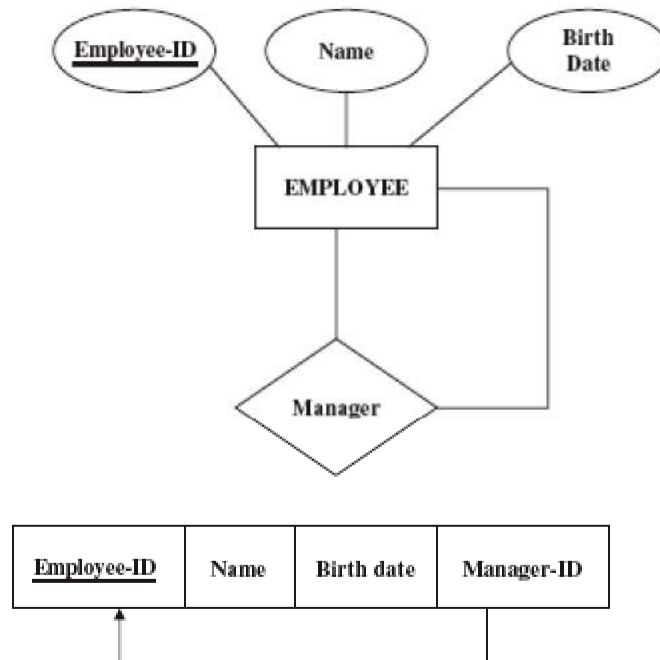
1.8 Converting Unary Relationship to Tables

Unary relationships are also called recursive relationships. The two most important cases of unary relationship are one-to-many and many-to-many.

One-to-many Unary Relationship

Each employee has exactly one manager. A given employee may manage zero to many employees. The foreign key in the relation is named Manager-ID.

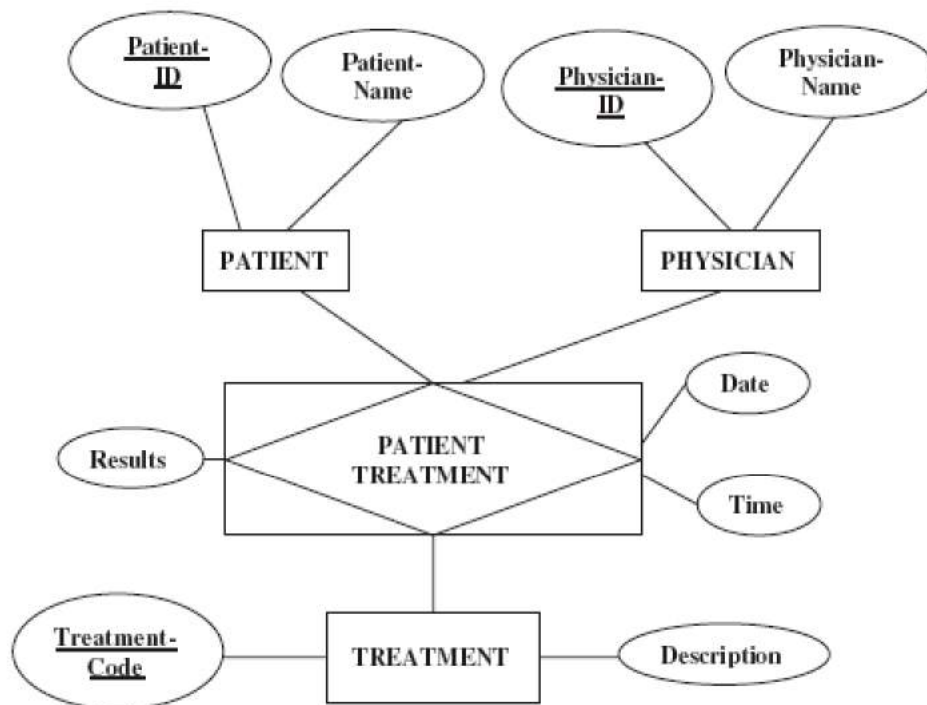
This attribute has the same domain as the primary key Employee-ID.



1.9 Converting Ternary Relationship to Tables

A ternary relationship is a relationship among three entity types. The three entities given in this example are PATIENT, PHYSICIAN, and TREATMENT.

The PATIENT–TREATMENT is an associative entity.



The primary key attributes – Patient ID, Physician ID, and Treatment Code – become foreign keys in PATIENT TREATMENT. These attributes are components of the primary key of PATIENT TREATMENT.

PATIENT TREATMENT

<u>Patient-ID</u>	Patient-Name
-------------------	--------------

PHYSICIAN

<u>Physician-ID</u>	Physician-Name
---------------------	----------------

PATIENT TREATMENT

<u>Patient-ID</u>	<u>Physician-ID</u>	<u>Treatment-Code</u>	<u>Date</u>	<u>Time</u>	Results
-------------------	---------------------	-----------------------	-------------	-------------	---------

TREATMENT

<u>Treatment-Code</u>	Description
-----------------------	-------------

3. Advantages of ER Modeling

An ER model is derived from business specifications. ER models separate the information required by a business from the activities performed within a business. Although business can change their activities, the type of information tends to remain constant. Therefore, the data structures also tend to be constant. The advantages of ER modeling are summarized later:

1. The ER modeling provides an easily understood pictorial map for the database design.
2. It is possible to represent the real world problems in a better manner in ER modeling.
3. The conversion of ER model to relational model is straightforward.
4. The enhanced ER model provides more flexibility in modeling real world problems.
5. The symbols used to represent entity and relationships between entities are simple and easy to follow.