

E-R modelling is termed as Entity – Relationship modelling. It is a graphical representation of entities and their relationships to each other. It defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

The building blocks of an E-R Model are:

1. **Entity**
2. **Attribute**
3. **Relationship**

1. Entity: It is a basic object that the ER model represents and it is a thing in the physical world with independent existence. The existence can be physical (car, house, employee, etc.) or conceptual existence (company, university, job, etc.).

2. Attributes: They are the properties that describe an entity. For instance, an entity “Student” can have attributes like “Name”, “Date of birth”, “Address”, etc. Attributes can be further divided into various sub-categories:

- **Composite vs. Simple attributes:** Composite attributes can be further divided into smaller sub-components, each having its own independent meaning. For example, an attribute “address” can have individual components like “house number”, “locality name”, etc. Simple attributes cannot be further divided.
- **Single-valued vs. Multivalued attributes:** Attributes having only one value for a particular entity are known as single-valued attributes. Example: the attribute “age” for a person can have only one value as a person cannot have more than one age value. On the other hand, a multivalued attribute can take multiple values for a single entity. Example: an entity “student” can have multiple “degrees”.
- **Stored vs. derived attributes:** When two or more attributes are related, often one is dependent on the other. For instance, if the “date of birth” of a person becomes the stored attribute, then the “age” of that person becomes the derived attribute.

If an entity has a unique identifier attribute, such that all the values of that attribute are distinct for each individual entity in the entity set, then that attribute is known as the **key attribute**. For Example: For Student entity, “Roll no.” is a key attribute. An entity which has a key attribute of its own is known as a **Strong entity**.

Some entity types can have more than one key attribute. In case an entity type has no key, it is known as the weak entity type. A **weak entity** type normally has a partial key, which is the attribute that can uniquely identify weak entities that are related to the same owner entity, and is represented in the model by a dotted line under the particular attribute.

Each attribute has a value set or **domain of values**, which provides the set of values that can be assigned to that attribute for each individual entity in the entity set. For example, an entity “employee” having the attribute “age” can only take values between 18 and 60, if retirement age is 60.

3. Relationships: A relationship identifies how entities are related to each other and amongst themselves.

TYPE/DEGREE OF RELATIONSHIPS:

1. Unary or recursive relationship: It is a relationship in which an entity is related to itself or when both participants in the relationship are the same entity.

For example: Student is related to other student through “Classmate” relationship.

2. Binary relationship: A binary relationship exists when two entities participate and are related to each other.

For example: Relationship between Teacher and Student entities.

3. Ternary Relationship: A ternary relationship exists when three entities participate in the relationship.

For example: Relationship between Teacher, Student and Course entities.

4. N-ary Relationship: It is a relationship in which N-entities are related.

CARDINALITY: Cardinality is the number of relationship instances in which entities can participate.

It may be a **one to one (1:1)**, **one to many (1: M)**, **many to one (M: 1)** or **many to many (M: N)**.

- **ONE-TO-ONE (1:1):** Relationship between Head and Department.

A Head manages one department.

Each department is managed by one Head.

- **ONE-TO-MANY (1: M):** Relationship between Department and Course.

A Department offers many Courses.

- **MANY-TO-MANY (M:N) :** Relationship between Teacher and Students.
A teacher teaches many students.
Each student is taught by many teachers

Weak Entity sets and Strong Entity sets and Identifying relationships


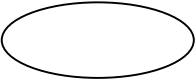
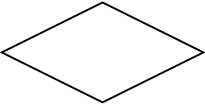

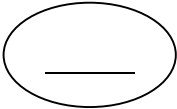
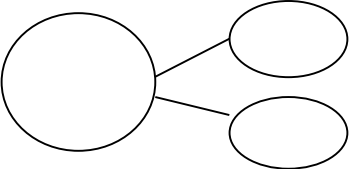
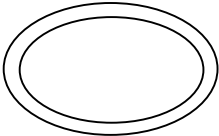

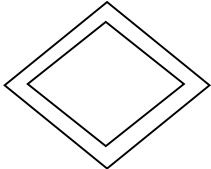
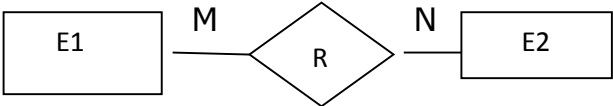
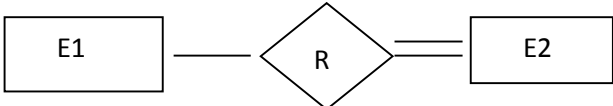
Entity types that do not have a key attribute of their own are called weak entity types. In contrast the regular entities that do have a key attribute are called strong entity types.

The existence of a weak entity set depends on the existence of a strong entity set. Weak entity set is represented by double-line rectangle.

Identifying relationships is the relationship between weak entity sets and strong entity sets. It is represented by double-line diamond.

The discriminator or partial key of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set. The primary key of a weak entity set is formed by the primary key of strong entity sets on which the weak entity set is existence dependent plus the weak entity set's discriminator.

E-R Modelling Notations

<u>Symbol</u>	<u>Meaning</u>
	Entity
	Attribute
	Relationship
	Derived Attribute
	Key attribute
	Composite attribute
	Multi valued attribute
	Weak entity
	Identifying Relationship
	Cardinality ratio M:N
	Total Participation of E2 in R

Steps involves in developing an ER diagram:

Step 1: Specify the Assumptions.

Step 2: Identify the Entities involved.

Step 3 Identify attributes of all the entities and classify them accordingly.

Step 4: Identify the relationships among various entities.

Step 5: identify the cardinalities.

Step 6: Draw complete ER diagram

EER Model: The **Enhanced entity–relationship (EER) model** (or extended entity–relationship model) in computer science is a high-level or conceptual data model incorporating extensions to the original entity–relationship (ER) model, used in the design of databases.

Additional Concepts included in EER are explained below:

SUBCLASSES AND SUPERCLASSES

- An entity type may have additional meaningful subgroupings of its entities. Entity type A is a subclass of an entity type B if and only if every A is necessarily a B.
- Subclass inherits all the attributes and relationship of its superclass entity called attribute and relationship inheritance.
- A subclass may have its own attributes and relationships (called specific attributes and relationships) along with the attributes and relationships inherited from the superclass.
- Example: EMPLOYEE may be further grouped into:
SECRETARY, ENGINEER, TECHNICIAN (Based on the EMPLOYEE's Job)
SALARIED_EMPLOYEE, HOURLY_EMPLOYEE (Based on the EMPLOYEE's method of pay)
- Each of these subgroupings is a subset of EMPLOYEE entity and are called a subclasses of EMPLOYEE
- EMPLOYEE is the superclass for each of these subclasses
- Superclass/subclass relationships:
EMPLOYEE/SECRETARY
EMPLOYEE/TECHNICIAN
EMPLOYEE/MANAGER
- These are also called IS-A relationships (SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE...etc.).

Attribute Inheritance in Superclass / Subclass Relationships

An entity that is member of a subclass inherits

- All attributes of the entity as a member of the superclass
- All relationships of the entity as a member of the superclass

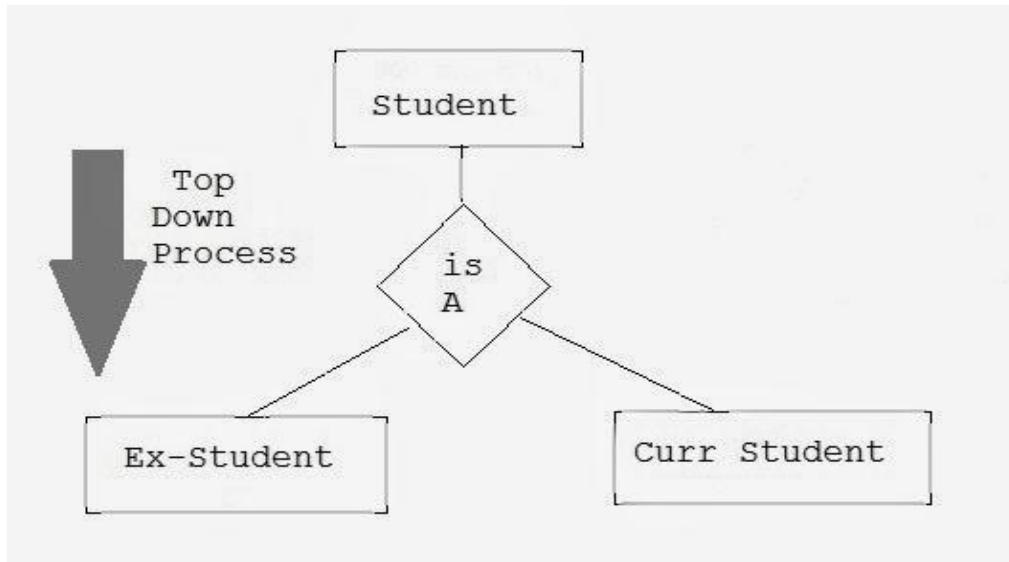
In the above example, SECRETARY (as well as TECHNICIAN and ENGINEER) inherit the attributes Name, SSN, ..., from EMPLOYEE.

Specialization

Specialization is the abstracting process of introducing new characteristics to an existing class of objects to create one or more new classes of objects.

In simple terms, when a higher level entity is broken down into some lower level entities then we call this process specialization.

Example:

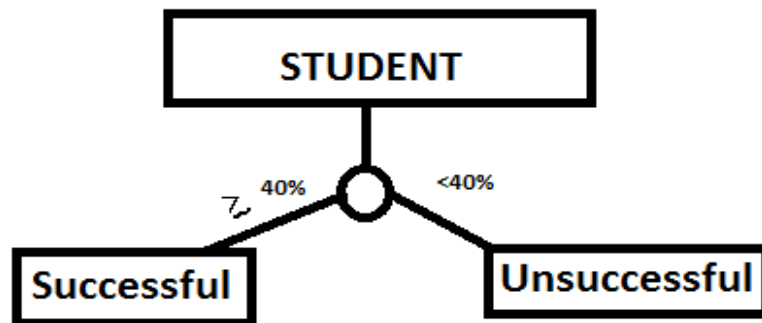


Types of Specialization:

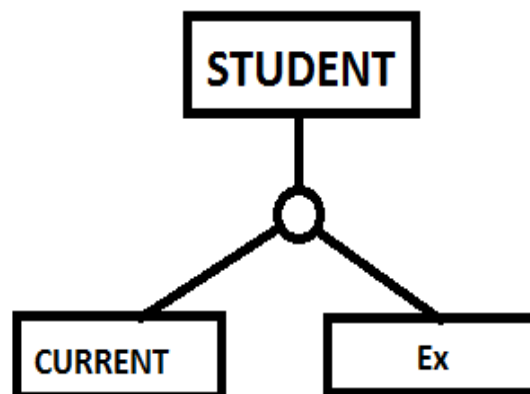
1. Predicate Defined or Condition defined specialization:

In a predicate-defined (or condition-defined) subclass, the subclass membership of an entity can be determined from its attribute values in the superclass.

e.g.



2. **Attribute Defined specialization:** A specialization having single common defining attribute and is known as Attribute defined specialization. E.g. { Current Student, Ex Student } specialization of Student entity based on type of student as follows.



3. **User Defined specialization:**

A specialization that is neither predicate defined nor attribute defined and solely based on users decision is user defined specialization.

GENERALISATION

Generalization is the reverse of the specialization process.

It is a bottom up approach where several classes with common features are generalized into a superclass. Example: CAR, TRUCK generalized into VEHICLE

Both CAR, TRUCK become subclasses of the superclass VEHICLE.

We can view {CAR, TRUCK} as a specialization of VEHICLE

Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK.

Diagrammatic notation are sometimes used to distinguish between generalization and specialization. Arrow pointing to the generalized superclass represents a generalization. Arrows pointing to the specialized subclasses represent a specialization. We do not use this notation because it is often subjective as to which process is more appropriate for a particular situation.

Specialization and Generalization Constraints:

1. Disjointness and overlapping constraint

Disjointness constraint specify that the subclasses of super class are disjoint i.e. an entity which belongs to the super class belongs to at most one subclass of the specialization. Otherwise, it is an overlapping constraint.

Represented by:

Disjointness -- 

Overlapping- 

2. Completeness(Total / Partial) constraint

A specialization is **total** in nature if an entity belonging to the super class belongs to at least one of the subclass.

Represented by-



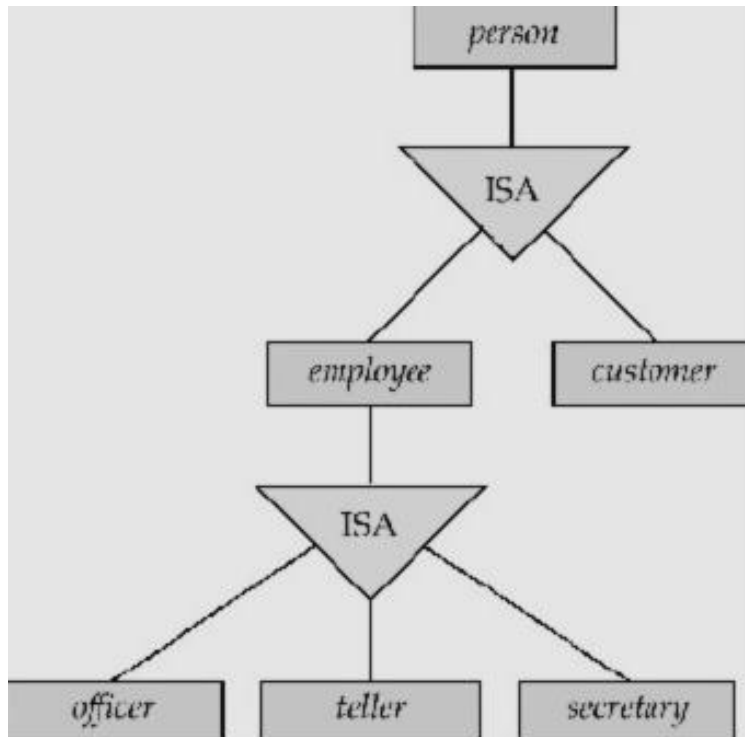
If we take some entity of super class and they do not belong to any of the subclasses then this is **partial** specialization.

Represented by-



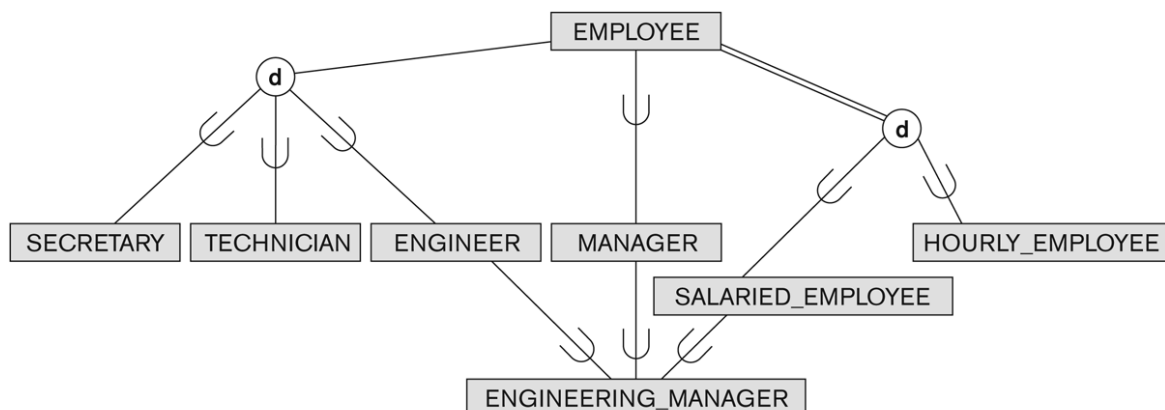
Specialization hierarchy:

Specialization Hierarchy has the constraint that every subclass participates as a subclass in only one class/subclass relationship, i.e. that each subclass has only one parent. This results in a tree structure. There is single inheritance.



Specialization lattice:

Specialization Lattice has the constraint that a subclass can be a subclass of more than one class/subclass relationship i.e. a subclass has more than one superclass, and such a subclass is called a shared subclass. This leads to multiple inheritance.



A specialization lattice with shared subclass ENGINEERING_MANAGER.

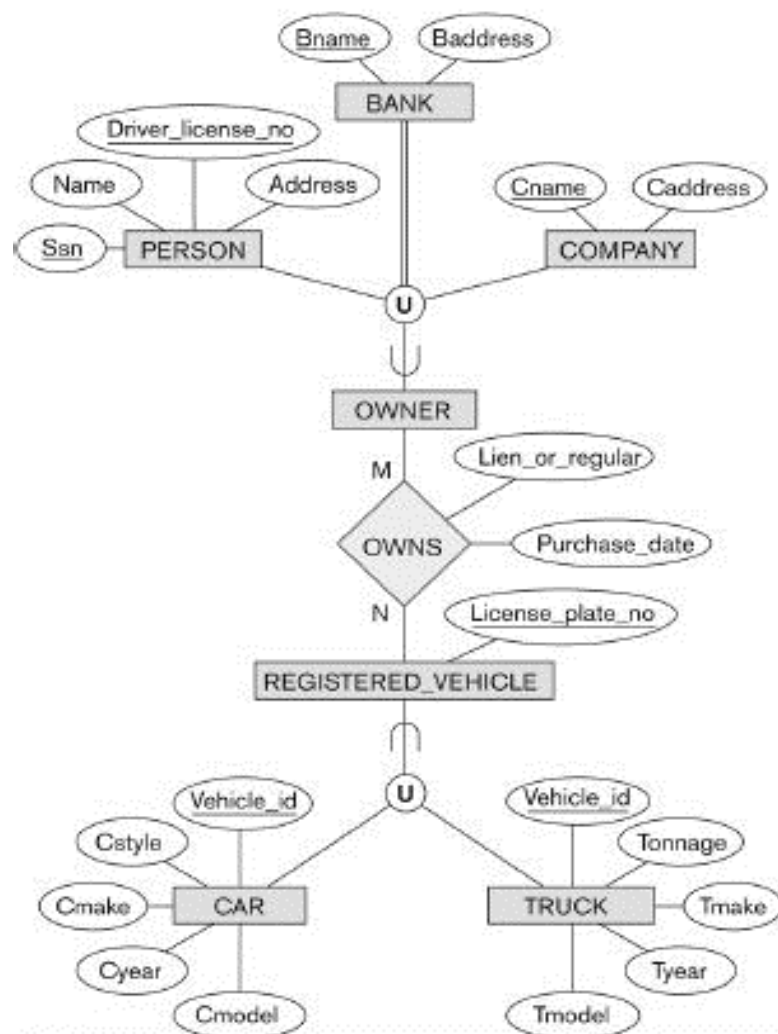
Categories (UNION TYPES)

So far we have seen that all the superclass/ subclass relationships have a single superclass.

But in some cases, we have a subclass participating in multiple superclass/subclass relationships with more than one superclass. Such a subclass is called a category or UNION TYPE.

Example: In a Database for vehicle registration, vehicle owner can be a person, a bank (holding a lien on a vehicle) or a company.

So in this example, a category (UNION Type) called OWNER is created to represent a subset of the union of the three Superclasses: COMPANY, BANK & PERSON.



Two categories (union types): OWNER and REGISTERED_VEHICLE.

DATABASE ANOMALIES: Database anomalies refer to errors and inconsistencies in the database. Anomalies creates problems while performing the operations like updation, insertion and deletion. The anomalies can be eliminated by redefining a relation into two or more relations.

There are three types of anomalies:

- 1) Update Anomaly
- 2) Insert anomaly
- 3) Delete Anomaly

Update anomaly: An update anomaly is a data inconsistency that results from data redundancy and a partial update. If in table 1, the address of student with student ID 101 is changed, then it has to be changed in both rows: 1 and 5, otherwise update anomaly occurs.

Insert anomaly: Inability to add data to the database due to absence of other data. Suppose in the table 1, a new student is to be added who is yet to get admission in some department and has not been assigned the department so far. In such cases, insert anomaly occurs.

Delete anomaly: Unintended loss of data due to deletion of other data. If we try to delete department “Statistics” from the database, then details of student “S2” will also get deleted even though we do want to keep student details.

Student ID	Student name	Address	Department
101	S1	Rohini	Operational Research
102	S2	Dwarka	Statistics
103	S3	Narela	Computer Science
104	S4	Palam	Mathematics
101	S1	Rohini	Operational Research

NORMALIZATION:

Normalization is a systematic approach of decomposing tables to *eliminate data redundancy* and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process that puts data into tabular form by *removing duplicated data* from the relation tables.

Main purpose of normalization:

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e. data is logically stored.

Types of normalisation:

- First Normal Form
- Second Normal Form
- Third Normal Form
- Boyce-Codd Normal Form

FIRST NORMAL FORM (1NF): A table is said to be in 1NF, if it satisfies the following properties:

- An attribute (column) of a table cannot hold multiple values.
- Any row must not have a column in which more than one value is saved, like separated with commas.

Example: Consider the following table:

Student_id	Student_name	Student_address	Student_Mobile No.
101	S1	New Delhi	9891989191
102	S2	Mumbai	9911121212, 9999112222
103	S3	Kolkata	9998881212
104	S4	Bangalore	9911000123, 9871987142

Two students (S2 & S4) are having two mobile numbers as seen in the table above.

This table is **not in 1NF** as the rule says “each attribute of a table must have atomic (single) values”, the Student_Mobile No. values for students S2 & S4 violate that rule. To make the table compatible with 1NF, we modify the above table as follows:

Student_id	Student_name	Student_address	Student_Mobile No.
101	S1	New Delhi	9891989191
102	S2	Mumbai	9911121212
102	S2	Mumbai	9999112222
103	S3	Kolkata	9998881212
104	S4	Bangalore	9911000123
104	S4	Bangalore	9871987142

This table satisfies the following:

- It is in First Normal Form (1NF).
- It contains redundant data.
- There is no column which uniquely identifies each and every row.
- Primary key = {Student_id, Student_Mobile No.}

SECOND NORMAL FORM (2NF): A table is said to be in 2NF, if it satisfies the following properties:

- Table should be in first normal form.
- All non-key or non-prime attributes are fully functional dependent on the primary key.

Note: An attribute, which is a part of the primary key, is known as a prime attribute. An attribute, which is not a part of the primary key, is said to be a non-prime attribute.

Example:

Student_id	Subject	Age
101	Mathematical Programming	23
101	Inventory Management	23
102	Mathematical Programming	22

- **Primary Key:** {Student_id, Subject}. **Non-prime attribute:** Age.
- The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non-prime attribute age is dependent on Student_id alone which is a proper subset of Primary key. This violates the rule for 2NF.
- To make the table compatible with 2NF, we can break it in two tables as follows:

Student_details table:

Student_id	Age
101	23
102	22

Student_Subject table:

Student_id	Subject
101	Mathematical Programming
101	Inventory Management
102	Mathematical Programming

THIRD NORMAL FORM (3NF): A table is said to be in 3NF, if it satisfies the following properties:

- Table is in second normal form.
- Every non-prime attribute of table must be dependent on primary key and there should be no transitive dependency.

Example:

Student_id	City	Pin Code
101	New Delhi	110001
102	Faridabad	121001

In this table, Student_id is the Primary key, and therefore non-prime attribute “City” is dependent on “Student_id” whereas other non-prime attribute “Pin Code” can be determined by “City” alone. Thus, there is **transitive dependency** in this table. Now, to make the table compatible with **3NF**, we need to break it into two new tables as follows:

Student_Detail Table:

Student_id	Pin Code
101	110001
102	121001

Address Table:

City	Pin Code
New Delhi	110001
Faridabad	121001

BOYCE AND CODD NORMAL FORM (BCNF): A table is said to be in BCNF, if it satisfies the following properties:

- R must be in 3rd Normal Form
- For each functional dependency (**X -> Y**), X should be a super Key.

Example:

Student_id	Student_name	Course_name	Course_code
101	S1	Mathematical Programming	C001
101	S1	Inventory Management	C002
102	S2	Mathematical Programming	C001

- **Functional dependencies in the table above:**

Student_id -> Student_name

Course_name -> Course_code

- **Primary key:** {Student_id, Course_code}
- The table is not in BCNF as neither Student_id nor Course_code alone is a key.

To make this table compatible with BCNF, we can break the table in the following manner:

Table 1:

Student_id	Student_name
101	S1
102	S2

Table 2:

Course_name	Course_code
Mathematical Programming	C001
Inventory Management	C002

Table 3:

Student_id	Course_code
101	C001
101	C002
102	C001

Primary Keys:

Table 1: Student_id

Table 2: Course_code

Table 3: {Student_id, Course_code}

1. What do you understand by event driven programming? List and explain about some of the events supported by Visual Basic Objects.

Event Driven Programming :-

In conventional programming, the sequence of operations for an application is determined by a central controlling program (e.g., a main procedure). In event-driven programming, the sequence of operations for an application is determined by the user's interaction with the application's interface (forms, menus, buttons, etc.).

In an event-driven application, the code doesn't follow a predetermined path rather it executes different code sections in response to events. Events can be triggered by the user's actions, by messages from the system or other applications, or even from the application itself. The sequence of these events determines the sequence in which the code executes, thus the path through the application's code or the sequence of execution differs each time the program runs.

Visual Basic is an event driven programming language.
Events common to most VB controls are described in the table below.

Event	Occurs When ...
Change	The user modifies text in a combo box or text box.
Click	The user clicks the primary mouse button on an object.
DbClick	The user double-clicks the primary mouse button on an object.
DragDrop	The user drags an object to another location.
DragOver	The user drags an object over another control.
GotFocus	An object receives focus.
KeyDown	The user presses a keyboard key while an object has focus.
KeyPress	The user presses and releases a keyboard key while an object has focus.
KeyUp	The user releases a keyboard key while an object has focus.
LostFocus	An object loses focus.
MouseDown	The user presses any mouse button while the mouse pointer is over an object.
MouseMove	The user moves the mouse pointer over an object.
MouseUp	The user releases any mouse button while the mouse pointer is over an object.

2. What are the different data types supported by Visual Basic ? How they can be declared ? Also mention their uses

Visual Basic Data Types

Visual Basic classifies the data types in two major categories:-

1). Numeric Data Types and 2). Non-Numeric data types.

1. Numeric Data Types

Numeric data types are types of data that consist of numbers, which can be computed mathematically with various standard operators such as add, minus, multiply, divide and more. Examples of numeric data types are examination marks, height, weight, the number of students in a class, share values, price of goods, monthly bills, fees and others. In Visual Basic, numeric data are divided into 7 types, depending on the range of values they can store. Calculations that only involve round figures or data that does not need precision can use Integer or Long integer in the computation. Programs that require high precision calculation need to use Single and Double decision data types, they are also called floating point numbers. For currency calculation, you can use the currency data types. Lastly, if even more precision is required to perform calculations that involve a many decimal points, we can use the decimal data types. These data types summarized in Table given below:-

Numeric Data Types

Type	Storage	Range of Values
Byte	1 byte	0 to 255
Integer	2 bytes	-32,768 to 32,767
Long	4 bytes	-2,147,483,648 to 2,147,483,648
Single	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E+38 for positive values.
Double	8 bytes	-1.79769313486232e+308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232e+308 for positive values.
Currency	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	12 bytes	+/- 79,228,162,514,264,337,593,543,950,335 if no decimal is use +/- 7.9228162514264337593543950335 (28 decimal places).

2. Non-numeric Data Types

Nonnumeric data types are data that cannot be manipulated mathematically using standard arithmetic operators. The non-numeric data comprises text or string data types, the Date data types, the Boolean data types that store only two values (true or false), Object data type and Variant data type. They are summarized in Table given below:-

Non-Numeric Data Types

Data Type	Storage	Range
String(fixed length)	Length of string	1 to 65,400 characters
String(variable length)	Length + 10 bytes	0 to 2 billion characters
Date	8 bytes	January 1, 100 to December 31, 9999
Boolean	2 bytes	True or False
Object	4 bytes	Any embedded object
Variant(numeric)	16 bytes	Any value as large as Double
Variant(text)	Length+22 bytes	Same as variable-length string

3. Give difference between arrays and dynamic arrays. How they can be created in Visual Basic ? Give Syntax

Arrays :-

An array is a collection of values of the same data type. The values in an array are called array elements. Array elements are accessed using a single name and an index number representing the position of the element within the array.

Arrays are used in a database application to handle data for processing.

Declaring Arrays

Unlike simple variables, arrays must be declared with the Dim (or Public, or Private) statement followed by the name of the array and the index of the last element in the array in parentheses for example:

```
Dim Ages(19) As Integer
```

Ages is the name of an array that holds 20 values (the ages of the 20 employees), with indices ranging from 0 to 19. Ages(0) is the first person's age, Ages(1) the second person's age, and so on. All we have to do is remember who corresponds to each age, but even this data can be handled by another array. To do this, we declare another array of 19 elements as follows:

```
Dim Names(19) As String
```

and then assign values to the elements of both arrays:

```
Names(0) = "John"  
Ages(0) = 34  
Names(1) = "Sam"  
Ages(1) = 38  
Names(19) = "Hedric"  
Ages(19) = 45
```

Dynamic Arrays

Sometimes we will not know how large an array to create. The earlier approach was to make it large enough to hold the maximum number of data. This will result in on an average, most of the array will be empty. To avoid this we can declare a dynamic array. The size of a dynamic array can vary during the execution of the software program.

With a dynamic array, we can discard the data and return the resources it occupied to the system.

To create a dynamic array, declare it as usual with the Dim statement, Public or Private but don't specify its dimensions:

```
Dim DynArray As Integer
```

Later in the program, when we know how many elements we want to store in the array, use the ReDim statement to redimension the array, this time to its actual size. In the following example, UserCount is a user-entered value:

```
ReDim DynArray(UserCount)
```

The ReDim statement can appear only in a procedure. Unlike the Dim statement, ReDim is executable, it forces the application to carry out an action at runtime. Dim statements aren't

executable, and they can appear outside procedures.

A dynamic array also can be redimensioned to multiple dimensions. Declare it with the Dim statement outside any procedure as follows

```
Dim Matrix() As Double
```

and then use the ReDim statement in a procedure to declare a three-dimensional array:

```
ReDim Matrix(9, 9, 9)
```

Note that the ReDim statement can't change the type of the array that's why the As clause is missing from the ReDim statement. Moreover, subsequent ReDim statements can change the bounds of the array Matrix but not the number of its dimensions. For example, we can't use the statement ReDim Matrix(99, 99) later in your code. Once an array has been redimensioned once, its number of dimensions can't change. In the preceding example, the Matrix array will remain three-dimensional through the course of the application.

The ReDim statement can be issued only from within a procedure. In addition, the array to be redimensioned must be visible from within the procedure that calls the ReDim statement.

Sr No.	Arrays	Dynamic Arrays
1.	An array is a collection of values of the same data type. The values in an array are called array elements. Array elements are accessed using a single name and an index number representing the position of the element within the array.	Sometimes we will not know how large an array to create. The earlier approach was to make it large enough to hold the maximum number of data. This will result in on an average, most of the array will be empty. To avoid this we can declare a dynamic array. The size of a dynamic array can vary during the execution of the software program. With a dynamic array, we can discard the data and return the resources it occupied to the system.
2.	The size of an array can't vary during the execution of the program.	The size of a dynamic array can vary during the execution of the software program
3.	Arrays must be declared with the Dim (or Public, or Private) statement followed by the name of the array and the index of the last element in the array in parentheses for example: Dim Ages(19) As Integer.	Dynamic array is also declared with the Dim statement, Public or Private but don't specify its dimensions: Dim DynArray As Integer Later in the program, when we know how many elements we want to store in the array, use the ReDim statement to redimension the array, this time to its actual size. In the following example, UserCount is a user-entered value: ReDim DynArray(UserCount)

5. Write a program with a good interface in visual Basic to print first 20 Fibonacci numbers.


```

Private Sub Command1_Click()
Dim x, g, n, i, sum As Integer
n = 20
x = 0
y = 1
Print x
Print y
For i = 3 To n
sum = x + y
Print sum
x = y
y = sum
y = sum
Next i
End Sub

```

6. a) *Difference b/w List Box and Combo Box*

Sr No.	List Box	Combo Box
1.	Occupies more space but shows more than one value.	Occupies less space but shows only one value for visibility .
2.	We can select multiple items.	Multiple select is not possible
3.	we can use checkboxes with in the list box.	can't use checkboxes within combo boxes
4.		

b) Diffence between picture box and image box :-

1. A Picture Box can act as a container , An image control can't.
2. An Image control has Stretch property, a Picture Box control does not.
3. Picture Box control has an AutoSize property, an Image control does not.
4. A Picture Box control is a container control, an Image control is not.
5. A Picture Box control also has a whole bunch of properties that an Image Control does not - BackColor, FillColor, FillStyle, etc.

c) Diffence between Check Box and Option Button :-

1.

Option Button	Checkbox
It is an element of a form	It is also an element of a form
It is used for selecting options	It is also used for selecting options
It is a graphical user interface widget	It is also a graphical user interface widget
Only one option can be selected	One or more options can be selected
Example:Selecting of Gender	Example:Selecting the games known
Select your Highest Educational qualification:	Select the games you can play:

A message box is not a control but it is a dialog box that appears when needed to get information from the user .

A text box is a control that resides on the form .

B.Sc	Hockey
M.Sc	Cricket
B.Tech	Basket Ball

(d) Difference Between Pop Up and Dynamic Menu :-

Pop up menu - is a floating menu that is displayed over a form independent of the menu bar.

Pop up menus are also called context menus because the items displayed on the Pop up menu depend on where the pointer is located when the right mouse button is clicked.

Val() function - converts a string to a number
OR

We can say that it returns the number contained in a string as a numeric value of appropriate type.

Syntax - Val (string)

Visual Basic Errors

In Visual Basic, errors also called *exceptions* and they fall into one of the three categories: syntax errors, run-time errors, and logical errors.

Syntax Errors

Syntax errors are those that appear while you write code. Visual Basic checks your code as you type it in the **Code Editor** window and alerts you if you make a mistake, such as misspelling a word or using a language element improperly. Syntax errors are the most common type of errors. You can fix them easily in the coding environment as soon as they occur.

Run-Time Errors

Run-time errors are those that appear only after you compile and run your code. These involve code that may appear to be correct in that it has no syntax errors, but that will not execute. For example, you might correctly write a line of code to open a file. But if the file is corrupted, the application cannot carry out the **Open** function, and it stops running. You can fix most run-time errors by rewriting the faulty code, and then recompiling and rerunning it.

Logical Errors

Logical errors are those that appear once the application is in use. They are most often unwanted or unexpected results in response to user actions. Logical errors are generally the hardest type to fix, since it is not always clear where they originate.

Dim Statement: It Declares and allocates storage space for one or more variables.

e.g. Dim marks As Integer

Arrays

A variable is a name to which we can assign a single value.

An array variable is an ordered collection of variables of the same type to which we can efficiently assign a list of values.

Creating an Array

In Visual Basic we first must declare or create an array before we try to store information in it. The command for doing this is DIM.

DIM mark(20) as Double

This statement Creates an array called **mark** with elements 0 to 20 that stores 21 values of type double.

DIM age(15) as integer

This statement Creates an array called age with elements 0 to 15 that stores 16 integers.

DIM name(10) as string

This statement creates an array called name with elements 0 to 10 that stores 11 strings.

Assigning Values to the elements in an Array

Mark(1) = 76, Stores the number 76 in the first element of the array called mark
Mark(3) = 87, Stores the number 87 in the third element of the array called mark

Displaying values in an array
txtOutput.Text = mark(1).toString

To assign 0 in every element in mark
mark(0) = 0
mark(1) = 0
mark(2) = 0 ...
mark(20) = 0

OR
for x = 0 to 20
 age(x) = 0
next x

The following two statements are equivalent. Each statement declares an array of 21 **Integer** elements. When you access the array, the index can vary from 0 through 20.

Dim totals(20) As Integer
Dim totals(0 To 20) As Integer

The following statement declares a two-dimensional array of type **Double**. The array has 4 rows (3 + 1) of 6 columns (5 + 1) each. Note that an upper bound represents the highest possible value for the index, not the length of the dimension. The length of the dimension is the upper bound plus one.

Dim matrix2(3, 5) As Double

An array can have from 1 to 32 dimensions.

Numeric Functions

1. Val Function

This function is used as a converter function. It converts the numbers contained in a string to its numeric equivalent.

Syntax:

Val(String)

Note: the string must be put in quotes

Working :-Val("124566") produces the number 124566

Val("125.0066")

will produce the number 125.0066

However, the Val function stops reading the string, when it encounters a character which has NO NUMERIC EQUIVALENCE, which is for alphabets, symbols, commas, etc are not recognized. But, Blanks and Tabs are simply removed and not counted at all.

so therefore we get something like this

Val ("4566 3442 5553 ")

gives the output 456634425553, no tabs, spaces included.

Now, for input something like this it has another output

Val ("12534ABCD2345")

gives output 12534 because when it reads "A" after four, it doesn't recognize it as a number, and the string is not read further, so even if there are numbers at the back of the string, they aren't added as they are never read.

2. Sgn Function

This function is used to determine the sign of a number.

Syntax:

Sgn(number)

the "number" should be a valid number or numeric expression

Following are the three cases for a sign of a number
If the number is
Positive (number > 0) then sign value returned is "1"
Negative (number < 0) then sign value returned is "-1"
Zero (number = 0) then sign value returned is "0"

example:
Dim N1 , N2 , N3 As Integer

N1 = 34
N2 = -29
N3 = 0

Print Sgn(N1)
Print Sgn(N2)
Print Sgn(N3)

the output is

1
-1
0

3. Int and Fix Functions

Both these functions are used to remove the numbers ^{left} to the decimal point. but these functions work differently.

the Fix() function simply removes (truncates) the fractional part.
num = Fix(134.2423)

// num is stored with the value 134

and the Int() function removes the fractional part, and rounds down to the nearest integer value
num1 = Int(134.9423)

// num1 is stored with the value 135