**INNER JOIN:** The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same.  
**Syntax**:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

INNER JOIN table2

ON table1.matching\_column = table2.matching\_column;

**table1**: First table.

**table2**: Second table

**matching\_column**: Column common to both the tables.

SELECT StudentCourse.COURSE\_ID, Student.NAME, Student.AGE FROM Student

INNER JOIN StudentCourse

ON Student.ROLL\_NO = StudentCourse.ROLL\_NO;

**LEFT JOIN**: This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.**Syntax:**

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

LEFT JOIN table2

ON table1.matching\_column = table2.matching\_column;

table1: First table.

table2: Second table

matching\_column: Column common to both the tables.

SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

LEFT JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

**RIGHT JOIN**: RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join. The rows for which there is no matching row on left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.**Syntax:**

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

RIGHT JOIN table2

ON table1.matching\_column = table2.matching\_column;

table1: First table.

table2: Second table

matching\_column: Column common to both the tables.

SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

RIGHT JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

**FULL JOIN:** FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain *NULL* values.**Syntax:**

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

FULL JOIN table2

ON table1.matching\_column = table2.matching\_column;

table1: First table.

table2: Second table

matching\_column: Column common to both the tables.

SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

FULL JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | **ALL**  The ALL operator is used to compare a value to all values in another value set. |
| 2 | **AND**  The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause. |
| 3 | **ANY**  The ANY operator is used to compare a value to any applicable value in the list as per the condition. |
| 4 | **BETWEEN**  The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value. |
| 5 | **EXISTS**  The EXISTS operator is used to search for the presence of a row in a specified table that meets a certain criterion. |
| 6 | **IN**  The IN operator is used to compare a value to a list of literal values that have been specified. |
| 7 | **LIKE**  The LIKE operator is used to compare a value to similar values using wildcard operators. |
| 8 | **NOT**  The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator.** |
| 9 | **OR**  The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause. |
| 10 | **IS NULL**  The NULL operator is used to compare a value with a NULL value. |
| 11 | **UNIQUE**  The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates). |

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password);  
// Check connection  
if ($conn->connect\_error) {  
  die("Connection failed: " . $conn->connect\_error);  
}  
  
// Create database  
$sql = "CREATE DATABASE myDB";  
if ($conn->query($sql) === TRUE) {  
  echo "Database created successfully";  
} else {  
  echo "Error creating database: " . $conn->error;  
}  
  
$conn->close();  
?>

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

SELECT \* FROM Customers  
ORDER BY Country ASC, CustomerName DESC;

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NULL;

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

SELECT MAX(Price) AS LargestPrice  
FROM Products;

SELECT AVG(Price)  
FROM Products;

SELECT SUM(Quantity)  
FROM OrderDetails;

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

* The percent sign (%) represents zero, one, or multiple characters
* The underscore sign (\_) represents one, single character

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a%';

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the [LIKE](https://www.w3schools.com/sql/sql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

* SELECT \* FROM Customers  
  WHERE City LIKE 'ber%'

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

SELECT \* FROM Customers  
WHERE Country IN ('Germany', 'France', 'UK');

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

SELECT CustomerName AS Customer, ContactName AS [Contact Person]  
FROM Customers;

The UNION operator is used to combine the result-set of two or more SELECT statements.

* Every SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in every SELECT statement must also be in the same order

SELECT City FROM Customers  
UNION  
SELECT City FROM Suppliers  
ORDER BY City;

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5;

SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders  
FROM Orders  
INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
WHERE LastName = 'Davolio' OR LastName = 'Fuller'  
GROUP BY LastName  
HAVING COUNT(Orders.OrderID) > 25;

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns TRUE if the subquery returns one or more records.

SELECT SupplierName  
FROM Suppliers  
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price = 22);

The ANY and ALL operators allow you to perform a comparison between a single column value and a range of other values.

The ANY operator:

* returns a boolean value as a result
* returns TRUE if ANY of the subquery values meet the condition

ANY means that the condition will be true if the operation is true for any of the values in the range.

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name operator ANY  
  (SELECT column\_name  FROM table\_name  WHERE condition);

The ALL operator:

* returns a boolean value as a result
* returns TRUE if ALL of the subquery values meet the condition
* is used with SELECT, WHERE and HAVING statements

ALL means that the condition will be true only if the operation is true for all values in the range.

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name operator ALL  
  (SELECT column\_name  FROM table\_name  WHERE condition);

There are mainly seven different types of Keys in DBMS and each key has it’s different functionality:

* **Super Key -** A super key is a group of single or multiple keys which identifies rows in a table.
* **Primary Key -** is a column or group of columns in a table that uniquely identify every row in that table.
* **Candidate Key -** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
* **Alternate Key -** is a column or group of columns in a table that uniquely identify every row in that table.
* **Foreign Key -** is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
* **Compound Key -** has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.
* **Composite Key -** An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.
* **Surrogate Key -** An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

## What is the Super key?

A superkey is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification.

**Example:**

|  |  |  |
| --- | --- | --- |
| **EmpSSN** | **EmpNum** | **Empname** |
| 9812345098 | AB05 | Shown |
| 9876512345 | AB06 | Roslyn |
| 199937890 | AB07 | James |

In the above-given example, EmpSSN and EmpNum name are superkeys.

## What is a Primary Key?

**PRIMARY KEY** is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. A table cannot have more than one primary key.

### Rules for defining Primary key:

* Two rows can't have the same primary key value
* It must for every row to have a primary key value.
* The primary key field cannot be null.
* The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

**Example:**

In the following example, <code>StudID</code> is a Primary Key.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **StudID** | Roll No | First Name | LastName | Email |
| 1 | 11 | Tom | Price | [abc@gmail.com](mailto:abc@gmail.com) |
| 2 | 12 | Nick | Wright | [xyz@gmail.com](mailto:xyz@gmail.com) |
| 3 | 13 | Dana | Natan | [mno@yahoo.com](mailto:mno@yahoo.com) |

## What is the Alternate key?

**ALTERNATE KEYS** is a column or group of columns in a table that uniquely identify every row in that table. A table can have multiple choices for a primary key but only one can be set as the primary key. All the keys which are not primary key are called an Alternate Key.

**Example:**

In this table, StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **StudID** | Roll No | First Name | LastName | Email |
| 1 | 11 | Tom | Price | [abc@gmail.com](mailto:abc@gmail.com) |
| 2 | 12 | Nick | Wright | [xyz@gmail.com](mailto:xyz@gmail.com) |
| 3 | 13 | Dana | Natan | [mno@yahoo.com](mailto:mno@yahoo.com) |

## What is a Candidate Key?

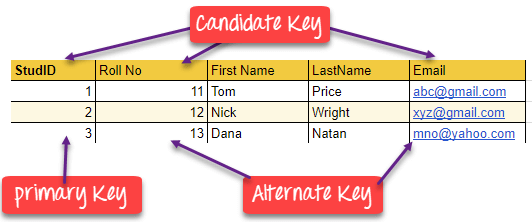
**CANDIDATE KEY** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes. The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key. A table can have multiple candidate keys but only a single primary key.

**Properties of Candidate key:**

* It must contain unique values
* Candidate key may have multiple attributes
* Must not contain null values
* It should contain minimum fields to ensure uniqueness
* Uniquely identify each record in a table

Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **StudID** | Roll No | First Name | LastName | Email |
| 1 | 11 | Tom | Price | [abc@gmail.com](mailto:abc@gmail.com) |
| 2 | 12 | Nick | Wright | [xyz@gmail.com](mailto:xyz@gmail.com) |
| 3 | 13 | Dana | Natan | [mno@yahoo.com](mailto:mno@yahoo.com) |

[](https://www.guru99.com/images/1/100518_0517_DBMSKeysPri1.png)

## What is the Foreign key?

**FOREIGN KEY** is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table.

**Example:**

|  |  |
| --- | --- |
| **DeptCode** | **DeptName** |
| 001 | Science |
| 002 | English |
| 005 | Computer |

|  |  |  |
| --- | --- | --- |
| **Teacher ID** | **Fname** | **Lname** |
| B002 | David | Warner |
| B017 | Sara | Joseph |
| B009 | Mike | Brunton |

In this key in dbms example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

|  |  |  |  |
| --- | --- | --- | --- |
| **Teacher ID** | **DeptCode** | **Fname** | **Lname** |
| B002 | 002 | David | Warner |
| B017 | 002 | Sara | Joseph |
| B009 | 001 | Mike | Brunton |

This concept is also known as Referential Integrity.

## What is the Compound key?

**COMPOUND KEY** has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database. However, when combined with the other column or columns the combination of composite keys become unique. The purpose of the compound key in database is to uniquely identify each record in the table.

**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| **OrderNo** | **PorductID** | **Product Name** | **Quantity** |
| B005 | JAP102459 | Mouse | 5 |
| B005 | DKT321573 | USB | 10 |
| B005 | OMG446789 | LCD Monitor | 20 |
| B004 | DKT321573 | USB | 15 |
| B002 | OMG446789 | Laser Printer | 3 |

In this example, OrderNo and ProductID can't be a primary key as it does not uniquely identify a record. However, a compound key of Order ID and Product ID could be used as it uniquely identified each record.

## What is the Composite key?

**COMPOSITE KEY** is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individually uniqueness is not guaranteed. Hence, they are combined to uniquely identify records in a table.

The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key.

## What is a Surrogate key?

**SURROGATE KEYS** is An artificial key which aims to uniquely identify each record is called a surrogate key. This kind of partial key in dbms is unique because it is created when you don't have any natural primary key. They do not lend any meaning to the data in the table. Surrogate key is usually an integer. A surrogate key is a value generated right before the record is inserted into a table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fname** | **Lastname** | **Start Time** | **End Time** |
| Anne | Smith | 09:00 | 18:00 |
| Jack | Francis | 08:00 | 17:00 |
| Anna | McLean | 11:00 | 20:00 |
| Shown | Willam | 14:00 | 23:00 |

Above, given example, shown shift timings of the different employee. In this example, a surrogate key is needed to uniquely identify each employee.

Surrogate keys in sql are allowed when

* No property has the parameter of the primary key.
* In the table when the primary key is too big or complicated.

## Difference Between Primary key & Foreign key

|  |  |
| --- | --- |
| **Primary Key** | **Foreign Key** |
| Helps you to uniquely identify a record in the table. | It is a field in the table that is the primary key of another table. |
| Primary Key never accept null values. | A foreign key may accept multiple null values. |
| Primary key is a clustered index and data in the DBMS table are physically organized in the sequence of the clustered index. | A foreign key cannot automatically create an index, clustered or non-clustered. However, you can manually create an index on the foreign key. |
| You can have the single Primary key in a table. | You can have multiple foreign keys in a table. |

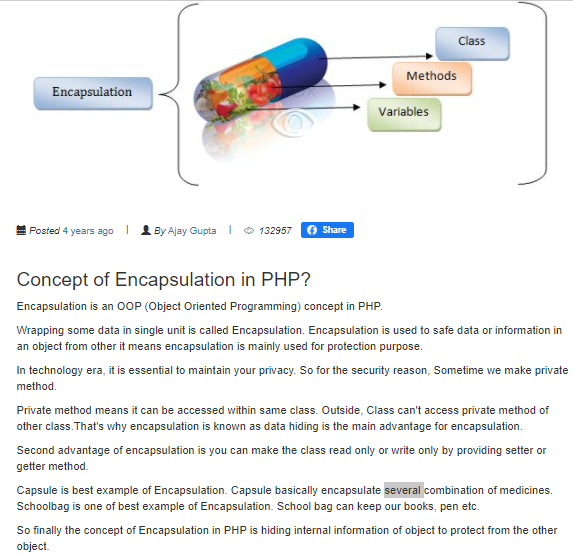
## Summary

* A key in SQL is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table)
* DBMS keys allow you to establish a relationship between and identify the relation between tables
* Seven Types of DBMS keys are Super, Primary, Candidate, Alternate, Foreign, Compound, Composite, and Surrogate Key.
* A super key is a group of single or multiple keys which identifies rows in a table.
* A column or group of columns in a table which helps us to uniquely identifies every row in that table is called a primary key
* All the keys which are not primary key are called an alternate key
* A super key with no repeated attribute is called candidate key
* A compound key is a key which has many fields which allow you to uniquely recognize a specific record
* A key which has multiple attributes to uniquely identify rows in a table is called a composite key
* An artificial key which aims to uniquely identify each record is called a surrogate key
* Primary Key never accept null values while a foreign key may accept multiple null values.

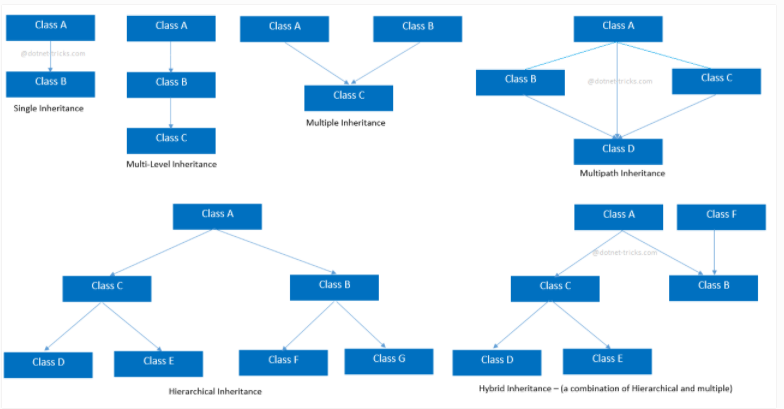
JavaScript is used to make the web page dynamic by adding interactivity to it which means with JavaScript we can add some code for mouse click, mouse over and other events on the web page and lot more.

The three major principles of OOP are;

* **Encapsulation** – this is concerned with hiding the implementation details and only exposing the methods. The main purpose of encapsulation is to;
  + Reduce software development complexity – by hiding the implementation details and only exposing the operations, using a class becomes easy.
  + Protect the internal state of an object – access to the class variables is via methods such as get and set, this makes the class flexible and easy to maintain.
  + The internal implementation of the class can be changed without worrying about breaking the code that uses the class.



* **Inheritance**– Inheritance is a mechanism of acquiring the features and behaviors of a class by another class.
* this is concerned with the relationship between classes. The relationship takes the form of a parent and child. The child uses the methods defined in the parent class. The main purpose of inheritance is;
  + Re-usability– a number of children, can inherit from the same parent. This is very useful when we have to provide common functionality such as adding, updating and deleting data from the database.

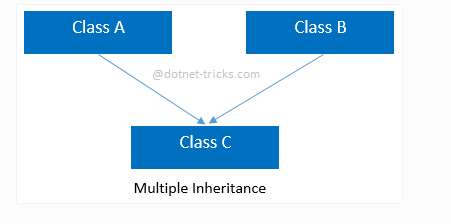


### 2) Multiple Inheritance

“**Multiple Inheritance**” refers to the concept of one class extending (Or inherits) more than one base class. The inheritance we learnt earlier had the concept of one base class or parent. The problem with “multiple inheritance” is that the derived class will have to manage the dependency on two base classes.

Note 1: Multiple Inheritance is very rarely used in software projects. Using Multiple inheritance often leads to problems in the hierarchy. This results in unwanted complexity when further extending the class.

Note 2: Most of the new OO languages like **Small Talk, Java, C# do not support Multiple inheritance**. Multiple Inheritance is supported in C++.



## PHP - What are Traits?

PHP only supports single inheritance: a child class can inherit only from one single parent.

So, what if a class needs to inherit multiple behaviors? OOP traits solve this problem.

Traits are used to declare methods that can be used in multiple classes. Traits can have methods and abstract methods that can be used in multiple classes, and the methods can have any access modifier (public, private, or protected).

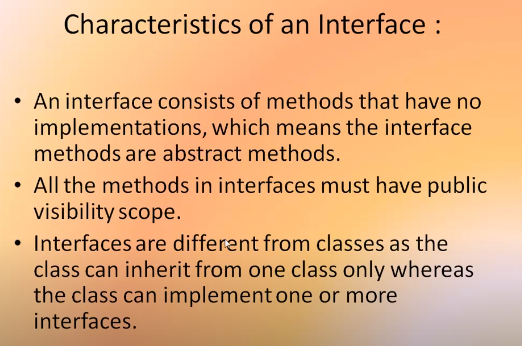
Traits are declared with the trait keyword:

<?php  
trait message1 {  
public function msg1() {  
    echo "OOP is fun! ";  
  }  
}  
  
class Welcome {  
  use message1;  
}  
  
$obj = new Welcome();  
$obj->msg1();  
?>

## PHP - What are Interfaces?

Interfaces allow you to specify what methods a class should implement.

* **Interface** – it is similar to a class. It only defines the methods and parameters.

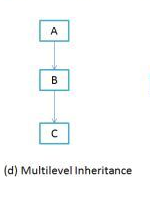


Interfaces make it easy to use a variety of different classes in the same way. When one or more classes use the same interface, it is referred to as "polymorphism".

Interfaces are declared with the interface keyword:

### 3) Multilevel Inheritance

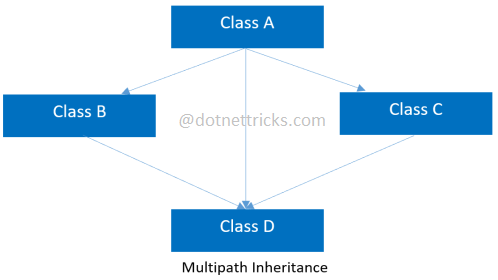
**Multilevel inheritance** refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class. As you can see in below flow diagram C is subclass or child class of B and B is a child class of A. For more details and example refer – [Multilevel inheritance in Java](https://beginnersbook.com/2013/12/multilevel-inheritance-in-java-with-example/).



### **Multipath inheritance**

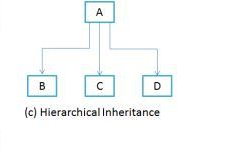
In this inheritance, a derived class is created from another derived classes and the same base class of another derived classes. This inheritance is not supported by .NET Languages like C#, F# etc.

In the given example, class D inherits the properties and behavior of class C and class B as well as Class A. Both class C and class B inherits the Class A. So, Class A is the parent for Class B and Class C as well as Class D. So it's making it Multipath inheritance.



### 4) Hierarchical Inheritance

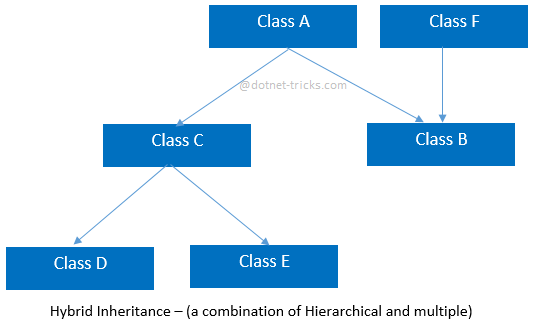
In such kind of inheritance one class is inherited by many**sub classes**. In below example class B,C and D **inherits** the same class A. A is **parent class (or base class)** of B,C & D. Read More at –



### **Hybrid inheritance**

This is combination of more than one inheritance. Hence, it may be a combination of Multilevel and Multiple inheritance or Hierarchical and Multilevel inheritance or Hierarchical and Multipath inheritance or Hierarchical, Multilevel and Multiple inheritance.

Since .NET Languages like[C#](https://www.dotnettricks.com/learn/csharp), F# etc. does not support multiple and multipath inheritance. Hence hybrid inheritance with a combination of multiple or multipath inheritances is not supported by .NET Languages.



* **Polymorphism** – this is concerned with having a single form but many different implementation ways. The main purpose of polymorphism is;
  + Simplify maintaining applications and making them more extendable.

*Polymorphism describes a pattern in object oriented programming in which classes have different functionality while sharing a common interface.*

* + But **PHP** "does not **support**" compile time **polymorphism**, which means function overloading and operator overloading.
* interface MyInterface {
* // methods
* }

class MyClass implements MyInterface {

    // methods

}

interface MyInterface {

    public function doThis();

    public function doThat();

    public function setName($name);

}

// VALID

class MyClass implements MyInterface {

    protected $name;

    public function doThis() {

        // code that does this

    }

    public function doThat() {

        // code that does that

    }

    public function setName($name) {

        $this->name = $name;

    }

}

// INVALID

class MyClass implements MyInterface {

    // missing doThis()!

    private function doThat() {

        // this should be public!

    }

    public function setName() {

        // missing the name argument!

    }

}

* **Encapsulation** - via the use of “get” and “set” methods etc.
* **Inheritance** - via the use of extends keyword
* **Polymorphism** - via the use of implements keyword
* “private $family, $food” means the variables cannot be accessed directly outside the class (Encapsulation).
* “public function \_\_construct($family…)” is the php constructor method. This function is called whenever an instance of the class has been created. In this case, we are setting the family and food.
* “public function get…()” is the method used to access the family or food value (Encapsulation)
* “public function set…()” is the method used to set the family or food value (Encapsulation)
* **Interface** – it is similar to a class. It only defines the methods and parameters.
* **Abstract class**– it is a class that cannot be used to create an object directly. Its purpose is to provide partial or whole implementations of common methods.

**Method overloading** occurs when two or more **methods** with same **method** name but different number of parameters in single class. **PHP** does not support **method overloading**.

**Method overriding** means two **methods** with same **method** name and same number of parameters in two different classes means parent class and child class.

Encapsulation

The meaning of **Encapsulation**, is to make sure that "sensitive" data is hidden from users. To achieve this, you must:

* declare class variables/attributes as private
* provide public **get** and **set** methods to access and update the value of a private variable

## Get and Set

You learned from the previous chapter that private variables can only be accessed within the same class (an outside class has no access to it). However, it is possible to access them if we provide public **get** and **set** methods.

The get method returns the variable value, and the set method sets the value.

Syntax for both is that they start with either get or set, followed by the name of the variable, with the first letter in upper case:

Java Packages & API

A package in Java is used to group related classes. Think of it as **a folder in a file directory**. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:

* Built-in Packages (packages from the Java API)
* User-defined Packages (create your own packages)

# **What is an API? (Application Programming Interface)**

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you’re using an API.

## What Is an Example of an API?

When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API.

To explain this better, let us take a familiar example.

Imagine you’re sitting at a table in a restaurant with a menu of choices to order from. The kitchen is the part of the “system” that will prepare your order. What is missing is the critical link to communicate your order to the kitchen and deliver your food back to your table. That’s where the waiter or API comes in. The waiter is the messenger – or API – that takes your request or order and tells the kitchen – the system – what to do. Then the waiter delivers the response back to you; in this case, it is the food.

Here is a real-life API example. You may be familiar with the process of searching flights online. Just like the restaurant, you have a variety of options to choose from, including different cities, departure and return dates, and more. Let us imagine that you’re booking you are flight on an airline website. You choose a departure city and date, a return city and date, cabin class, as well as other variables. In order to book your flight, you interact with the airline’s website to access their database and see if any seats are available on those dates and what the costs might be.

PHP - Access Modifiers

Properties and methods can have access modifiers which control where they can be accessed.

There are three access modifiers:

* public - the property or method can be accessed from everywhere. This is default
* protected - the property or method can be accessed within the class and by classes derived from that class
* private - the property or method can ONLY be accessed within the class

## PHP - What is Inheritance?

Inheritance in OOP = When a class derives from another class.

The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.

An inherited class is defined by using the extends keyword.

<?php  
class Fruit {  
  public $name;  
  public $color;  
  public function \_\_construct($name, $color) {  
    $this->name = $name;  
    $this->color = $color;  
  }  
  public function intro() {  
    echo "The fruit is {$this->name} and the color is {$this->color}.";  
  }  
}  
  
class Strawberry extends Fruit {  
  public $weight;  
  public function \_\_construct($name, $color, $weight) {  
    $this->name = $name;  
    $this->color = $color;  
    $this->weight = $weight;  
  }  
  public function intro() {  
    echo "The fruit is {$this->name}, the color is {$this->color}, and the weight is {$this->weight} gram.";  
  }  
}  
  
$strawberry = new Strawberry("Strawberry", "red", 50);  
$strawberry->intro();  
?>

## PHP - The final Keyword

The final keyword can be used to prevent class inheritance or to prevent method overriding.

The following example shows how to prevent class inheritance:

<?php  
final class Fruit {  
  // some code  
}  
  
// will result in error  
class Strawberry extends Fruit {  
  // some code  
}  
?>

## PHP - Class Constants

Constants cannot be changed once it is declared.

Class constants can be useful if you need to define some constant data within a class.

A class constant is declared inside a class with the const keyword.

Class constants are case-sensitive. However, it is recommended to name the constants in all uppercase letters.

We can access a constant from outside the class by using the class name followed by the scope resolution operator (::) followed by the constant name, like here:

<?php  
class Goodbye {  
  const LEAVING\_MESSAGE = "Thank you for visiting W3Schools.com!";  
}  
  
echo Goodbye::LEAVING\_MESSAGE;  
?>

## PHP - What are Abstract Classes and Methods?

Abstract classes and methods are when the parent class has a named method, but need its child class(es) to fill out the tasks.

An abstract class is a class that contains at least one abstract method. An abstract method is a method that is declared, but not implemented in the code.

An abstract class or method is defined with the abstract keyword:

When inheriting from an abstract class, the child class method must be defined with the same name, and the same or a less restricted access modifier. So, if the abstract method is defined as protected, the child class method must be defined as either protected or public, but not private. Also, the type and number of required arguments must be the same. However, the child classes may have optional arguments in addition.

So, when a child class is inherited from an abstract class, we have the following rules:

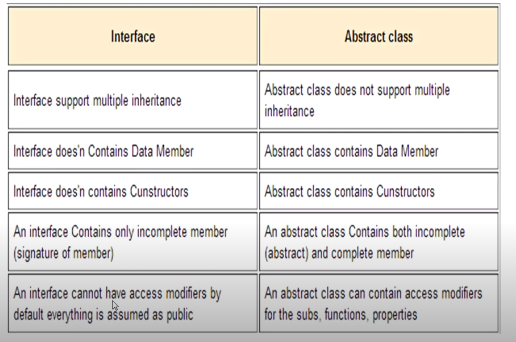
* The child class method must be defined with the same name and it redeclares the parent abstract method
* The child class method must be defined with the same or a less restricted access modifier
* The number of required arguments must be the same. However, the child class may have optional arguments in addition

<?php  
abstract class ParentClass {  
  // Abstract method with an argument  
  abstract protected function prefixName($name);  
}  
  
class ChildClass extends ParentClass {  
  public function prefixName($name) {  
    if ($name == "John Doe") {  
      $prefix = "Mr.";  
    } elseif ($name == "Jane Doe") {  
      $prefix = "Mrs.";  
    } else {  
      $prefix = "";  
    }  
    return "{$prefix} {$name}";  
  }  
}  
  
$class = new ChildClass;  
echo $class->prefixName("John Doe");  
echo "<br>";  
echo $class->prefixName("Jane Doe");  
?>

PHP - Interfaces vs. Abstract Classes

Interface are similar to abstract classes. The difference between interfaces and abstract classes are:

* Interfaces cannot have properties, while abstract classes can
* All interface methods must be public, while abstract class methods is public or protected
* All methods in an interface are abstract, so they cannot be implemented in code and the abstract keyword is not necessary
* Classes can implement an interface while inheriting from another class at the same time



## PHP - Using Interfaces

To implement an interface, a class must use the implements keyword.

A class that implements an interface must implement **all** of the interface's methods.

<?php  
interface Animal {  
  public function makeSound();  
}  
  
class Cat implements Animal {  
  public function makeSound() {  
    echo "Meow";  
  }  
}  
  
$animal = new Cat();  
$animal->makeSound();  
?>

## PHP - Static Methods

Static methods can be called directly - without creating an instance of the class first.

Static methods are declared with the static keyword:

<?php  
class greeting {  
  public static function welcome() {  
    echo "Hello World!";  
  }  
}  
  
// Call static method  
greeting::welcome();  
?>

## PHP - Static Properties

Static properties can be called directly - without creating an instance of a class.

Static properties are declared with the static keyword:

<?php  
class pi {  
  public static $value = 3.14159;  
}  
  
// Get static property  
echo pi::$value;  
?>

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

* local
* global
* static

## Global

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

The global keyword is used to access a global variable from within a function.

To do this, use the global keyword before the variables (inside the function):

**Local**

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

## static

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the static keyword when you first declare the variable:

PHP Namespaces

Namespaces are qualifiers that solve two different problems:

1. They allow for better organization by grouping classes that work together to perform a task
2. They allow the same name to be used for more than one class

For example, you may have a set of classes which describe an HTML table, such as Table, Row and Cell while also having another set of classes to describe furniture, such as Table, Chair and Bed. Namespaces can be used to organize the classes into two different groups while also preventing the two classes Table and Table from being mixed up.

## Declaring a Namespace

Namespaces are declared at the beginning of a file using the namespace keyword:

## HP - What is an Iterable?

An iterable is any value which can be looped through with a foreach() loop.

The iterable pseudo-type was introduced in PHP 7.1, and it can be used as a data type for function arguments and function return values.

## PHP - Using Iterables

The iterable keyword can be used as a data type of a function argument or as the return type of a function:

## What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

## Create Cookies With PHP

A cookie is created with the setcookie() function.

### **Syntax**

setcookie(name, value, expire, path, domain, secure, httponly);

# PHP Sessions

A session is a way to store information (in variables) to be used across multiple pages.

Unlike a cookie, the information is not stored on the users computer.

**Unlink() function:** The unlink() function is an inbuilt function in PHP which is used to delete a file. The filename of the file which has to be deleted is sent as a parameter and the function returns True on success and False on failure. The unlink() function in PHP accepts two-parameter.

**Syntax:**

unlink( filename, context )

**Unset() function:** The Unset() function is an inbuilt function in PHP which is used to remove the content from the file by emptying it. It means that the function clears the content of a file rather deleting it. The unset() function not only clears the file content but also is used to unset a variable, thereby making it empty.

**Syntax:**

unset( $variable )

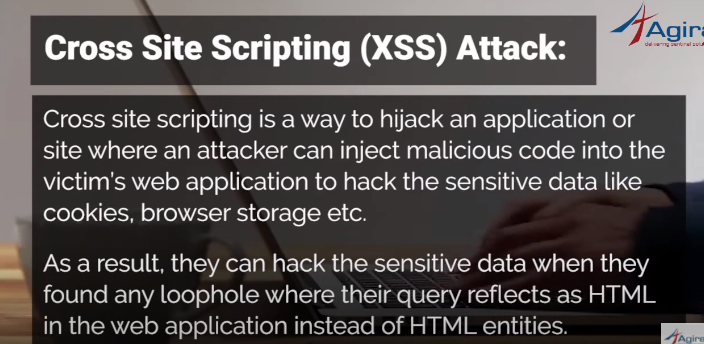
**Difference between unlink() and unset() function:**

|  |  |
| --- | --- |
| unlink() Function | unset() Function |
| It is used to delete a file within a directory completely on successful execution. | It is used to make a specific file empty by removing its content. |
| There are two parameter **filename** and the other one is **context**. | There is only one parameter **variable**. |
| Return True on success and false on failure. | This function does not return any value. |
| This is a function for file system handling. | This is a function for variable management. |
|  |  |

**KEY DIFFERENCE:**

* **SQL** is a language which is used to operate your database whereas **MySQL** was one of the first open-source database available in the market
* SQL is used in the accessing, updating, and manipulation of data in a database while MySQL is an RDBMS that allows keeping the data that exists in a database organized
* SQL is a Structured Query Language and MySQL is a RDBMS to store, retrieve, modify and administrate a database.
* SQL is a query language while MYSQL is a database software

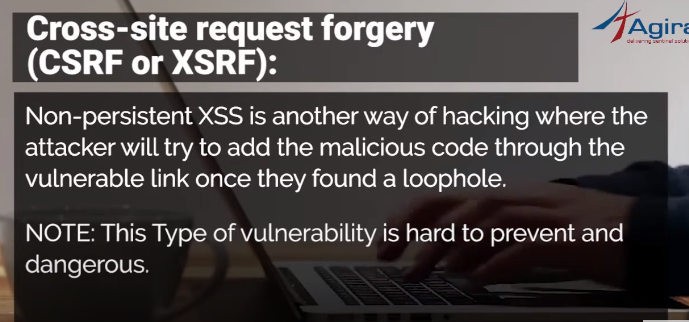
XSS



[Cross-site scripting](https://portswigger.net/web-security/cross-site-scripting) (or XSS) allows an attacker to execute arbitrary JavaScript within the browser of a victim user.

[Cross-site request forgery](https://portswigger.net/web-security/csrf) (or CSRF) allows an attacker to induce a victim user to perform actions that they do not intend to.

The consequences of XSS vulnerabilities are generally more serious than for CSRF vulnerabilities



## PHP include vs. require

**The include and require statements are identical, except upon failure:**

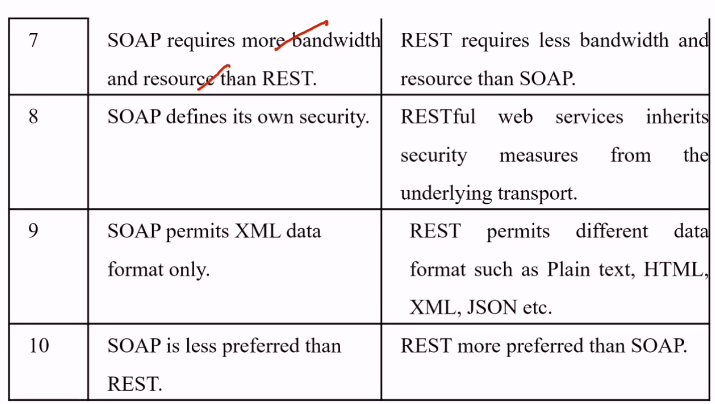
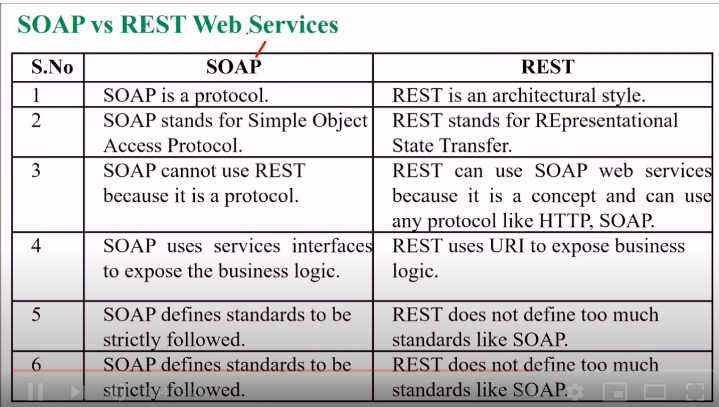
* require will produce a fatal error (E\_COMPILE\_ERROR) and stop the script
* include will only produce a warning (E\_WARNING) and the script will continue

The require statement is also used to include a file into the PHP code.

However, there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute:

So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.



The GET Method

**GET is used to request data from a specified resource.**

**GET is one of the most common HTTP methods.**

Note that the query string (name/value pairs) is sent in the URL of a GET request:

/test/demo\_form.php?name1=value1&name2=value2

**Some other notes on GET requests:**

* GET requests can be cached
* GET requests remain in the browser history
* GET requests can be bookmarked
* GET requests should never be used when dealing with sensitive data
* GET requests have length restrictions
* GET requests are only used to request data (not modify)

The POST Method

**POST is used to send data to a server to create/update a resource.**

The data sent to the server with POST is stored in the request body of the HTTP request:

POST /test/demo\_form.php HTTP/1.1  
Host: w3schools.com  
name1=value1&name2=value2

**POST is one of the most common HTTP methods.**

**Some other notes on POST requests:**

* POST requests are never cached
* POST requests do not remain in the browser history
* POST requests cannot be bookmarked
* POST requests have no restrictions on data length

The PUT Method

**PUT is used to send data to a server to create/update a resource.**

The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

## The HEAD Method

**HEAD is almost identical to GET, but without the response body.**

## The DELETE Method

**The DELETE method deletes the specified resource.**

## The OPTIONS Method

**The OPTIONS method describes the communication options for the target resource.**