




OCTOBER 12, 2017

**T-CODE-BURNER**

PHP INTERVIEW QUESTIONS FOR 2 YEARS' EXPERIENCE

VISHAL TURI  
T-CODE-BURNER  
INDIA



## Before interview questions you need to clear all of the below things first:

- *Make clear resume with original skills you know and have minimum 70% of knowledge like (CorePHP, Yii2, CodeIgniter). If you have less than 50% of knowledge in any skills please mention there or don't write in resume.*
- *List all projects which you have done and performed well, you should prepare all your challenging tasks of listed projects.*
- *Be confident with your answers, if you don't know answer than feel free to say no.*
- *Be clear with previous company leaving reason.*
- *Mostly don't negotiate your salary expectation, and be honest with to show your original/current salary.*
- *Don't show you really need this job on your face in HR round.*

## What is interpreter and compiler, difference between interpreter and compiler?

We generally write a computer program using a high-level language. A high-level language is one which is understandable by us humans. It contains words and phrases from the English (or other) language. But a computer does not understand high-level language. It only understands program written in 0's and 1's in binary, called the machine code. A program written in high-level language is called a source code. We need to convert the source code into machine code and this is accomplished by compilers and interpreters. Hence, a compiler or an interpreter is a program that converts program written in high-level language into machine code understood by the computer.

The difference between an interpreter and a compiler is given below:

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.



**Figure: Compiler**



**Figure: Interpreter**

## PHP script execution process.

Reference from below link:

<https://www.scribd.com/document/18171982/PHP-Compiler-Internals>

<https://www.phpclasses.org/blog/post/117-PHP-compiler-performance.html>

<https://stackoverflow.com/questions/1514676/is-php-compiled-or-interpreted>

A compiler is a program that transforms source code into another representation of the code. The target representation is often machine code, but it may as well be source code in another language or even in the same language.

PHP became a compiled language in the year 2000, when PHP 4 was released for the first time. Until version 3, PHP source code was parsed and executed right away by the PHP interpreter.

PHP 4 introduced the the [Zend engine](#). This engine splits the processing of PHP code into several phases. The first phase parses PHP source code and generates a binary representation of the PHP code known as **Zend opcodes**. Opcodes are sets of instructions similar to Java bytecodes. These opcodes are stored in memory. The second phase of Zend engine processing consists in executing the generated opcodes.

The Zend engine was built in such way that right after the first phase, the opcodes may be stored in the server shared memory space. This is done by special PHP extensions known as **opcode caching extensions**. There are several PHP caching extensions also known as [PHP accelerator extensions](#).

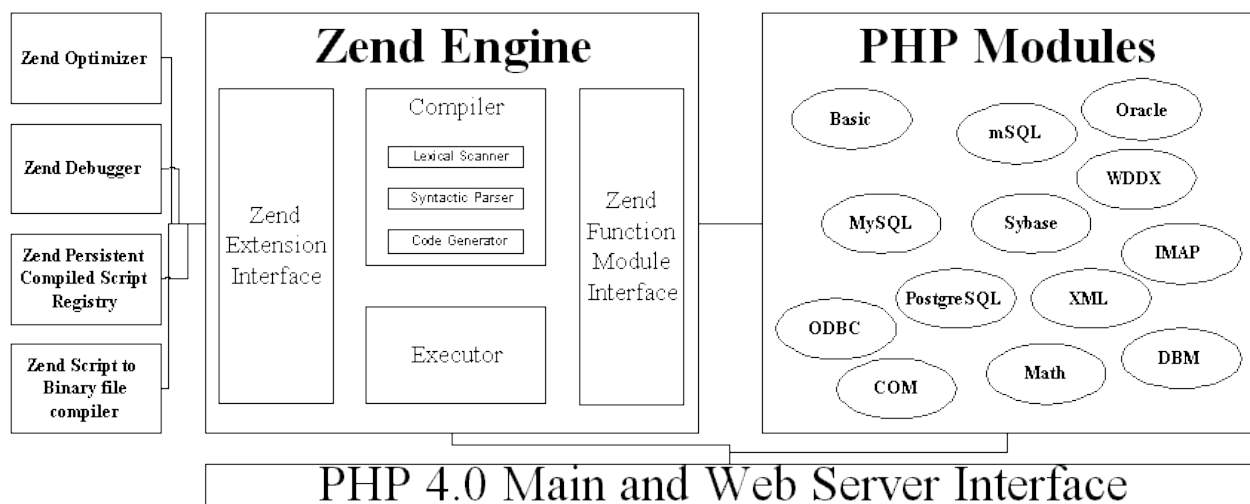
The purpose of these extensions is to skip the initial compilation step. If a PHP script was previously compiled and stored in shared memory, next time the same script is executed, the caching extension just loads the compiled opcodes from the shared memory very quickly. This way PHP gains a lot of time by skipping the

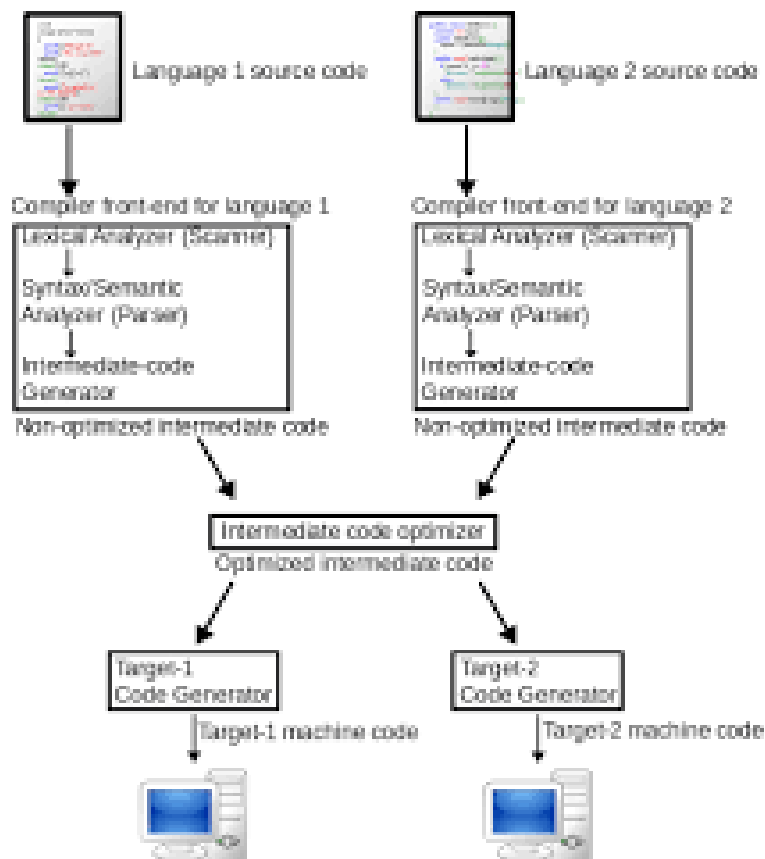
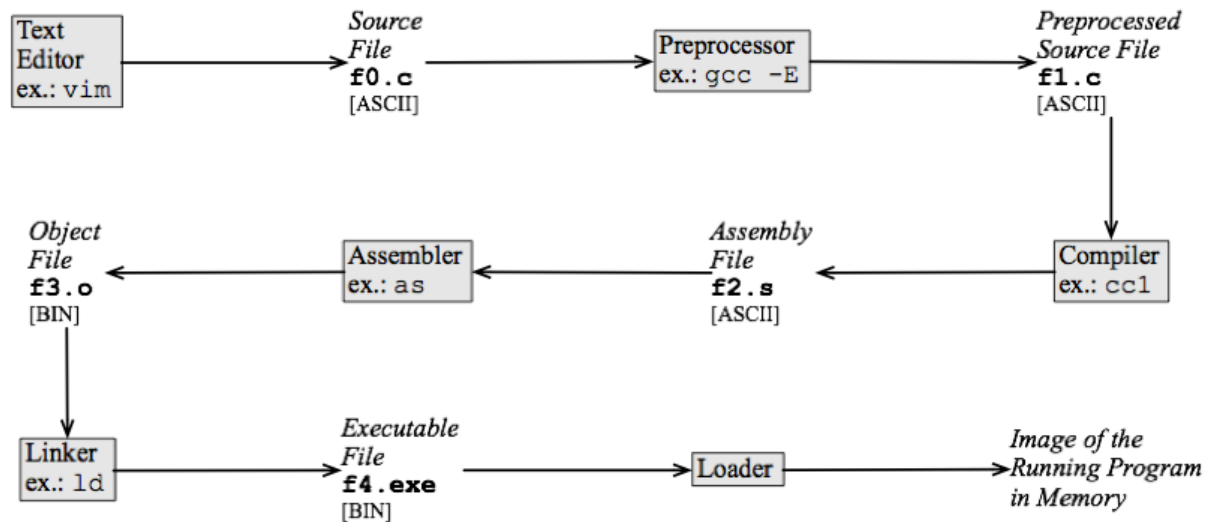
initial opcode compilation step.

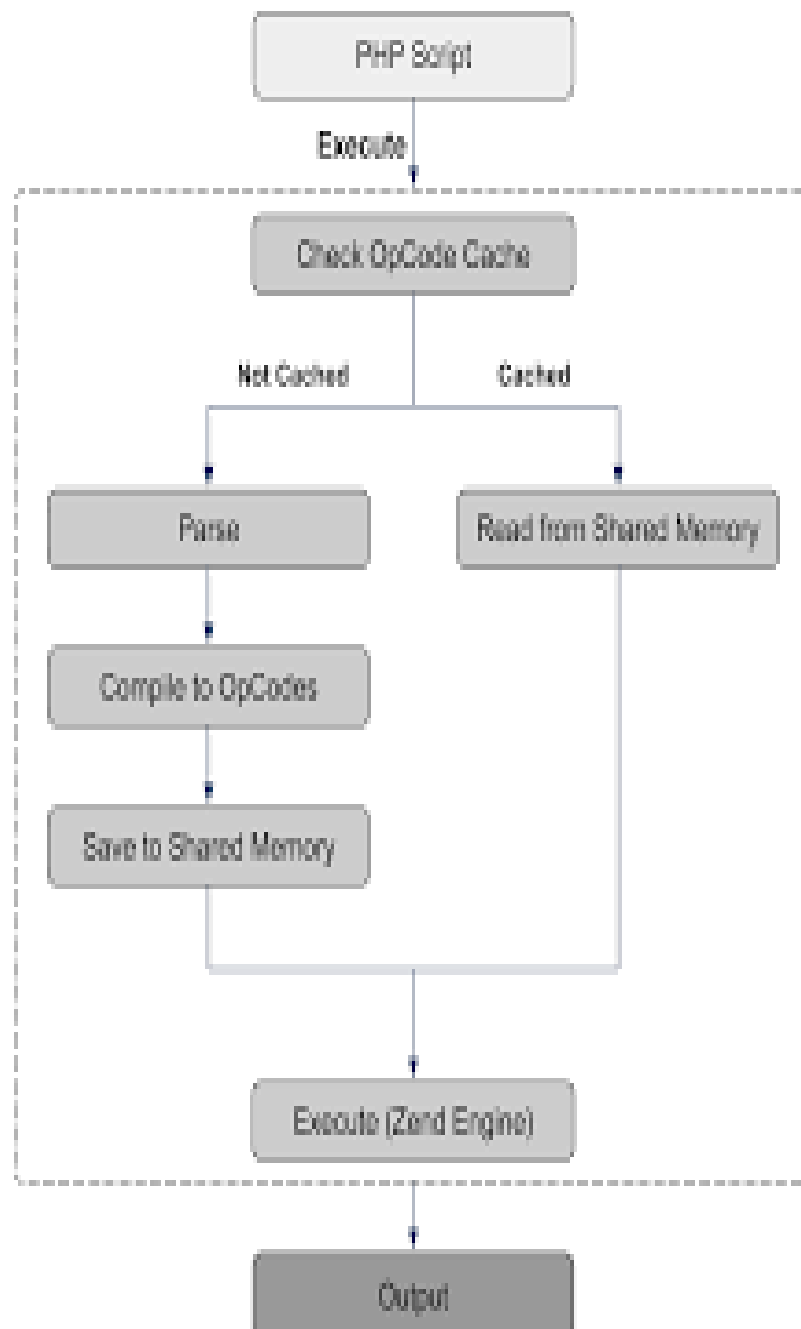
Keep in mind that despite the name, accelerator extensions do not make compiled PHP opcodes actually execute faster. Eventual acceleration of the execution of PHP opcodes may be achieved with optimizer extensions. These are special extensions that analyze the compiled opcodes and rearrange them in order that they may execute the same code faster.

This diagram shows the flow of PHP code compilation steps that happen when using the Zend Engine based PHP distributions.

## PHP 4.0 Layout









## What are the differences between public, private, protected, Static, transient, final and volatile?

- **Public:** scope to make that variable/function available from anywhere, other classes and instances of the object.
- **Protected:** scope when you want to make your variable/function visible in all classes that extend current class including the parent class.
- **Private:** scope when you want your variable/function to be visible in its own class only.
- **Static:** variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope.
- **Final:** keyword prevents child classes from overriding a method by prefixing the definition with final. If the class itself is being defined final then it cannot be extended.
- **Transient:** A transient variable is a variable that may not be serialized.
- **Volatile:** a variable that might be concurrently modified by multiple threads should be declared volatile. Variables declared to be volatile will not be optimized by the compiler because their value can change at any time.
- Example added in sololearn

## Types of error.

Basically there are four types of errors in PHP, which are as follows:

- Parse Error (Syntax Error)
- Fatal Error
- Warning Error
- Notice Error

### 1. Parse Errors (syntax errors)

The parse error occurs if there is a syntax mistake in the script; the output is Parse errors. A parse error stops the execution of the script. There are many reasons for the occurrence of parse errors in PHP. The common reasons for parse errors are as follows:

Common reason of syntax errors are:

- Unclosed quotes
- Missing or Extra parentheses
- Unclosed braces
- Missing semicolon

### 2. Fatal Errors

Fatal errors are caused when PHP understands what you've written, however what you're asking it to do can't be done. Fatal errors stop the execution of the script. If you are trying to access the undefined functions, then the output is a fatal error.

### 3. Warning Errors

Warning errors will not stop execution of the script. The main reason for warning errors are to include a missing file or using the incorrect number of parameters in a function.

### 4. Notice Errors

Notice that an error is the same as a warning error i.e. in the notice error execution of the script does not stop. Notice that the error occurs when you try to access the undefined variable, then produce a notice error.

#### Comprehensive List of Errors in PHP:

To better deal with the errors or facilitate debugging, it is necessary to understand the type of error and its cause. In this article, we discussed the major types of errors and their usual causes. Below is a more comprehensive list of PHP errors:

E_PARSE	It shows parse error occurred during compilation time.
E_NOTICE	It is a run time error which is caused due to error in script.
E_ERROR	This error terminates script execution. It is a fatal error
E_ALL	They catch all types of warnings and errors.
RECOVERABLE_ERROR	They are serious errors which are catchable.
E_STRICT	They are Run time notices.
E_USER_NOTICE	It indicates notice message generated by the user.
E_USER_WARNING	It indicates warning message generated by the user.
E_USER_ERROR	It indicates a user generated error message.
E_COMPILE_ERROR	It indicates compile time error problem.
E_CORE_WARNING	This is a type of warning which occurs at the time of initial startup of PHP
E_CORE_ERROR	This is a fatal error which usually appears during installation process.
E_WARNING	It doesn't halt script execution. It is a run time warning.

## Difference between warnings and notices.

### Notice

- A notice is an advisory message like "You probably shouldn't be doing what you're doing"
- Execution of the script is not halted
- Example

```
echo $undefinedVariable;
```

### Warning

- A warning is a message like "You are doing something wrong and it is very likely to cause errors in the future, so please fix it." Execution of the script is not halted;
- Example

```
echo 1/0;
```

## What are JOINS?

Joins help retrieving data from two or more database tables.  
The tables are mutually related using primary and foreign keys.

Note: JOIN is the most misunderstood topic amongst SQL learners. For sake of simplicity and ease of understanding, we will be using a new Database to practice sample. As shown below

<b>id</b>	<b>first_name</b>	<b>last_name</b>	<b>movie_id</b>
1	Adam	Smith	1
2	Ravi	Kumar	2
3	Susan	Davidson	5
4	Jenny	Adrianna	8
6	Lee	Pong	10

<b>id</b>	<b>title</b>	<b>category</b>
1	ASSASSIN'S CREED: EMBERS	Animations
2	Real Steel(2012)	Animations
3	Alvin and the Chipmunks	Animations
4	The Adventures of Tin Tin	Animations
5	Safe (2012)	Action
6	Safe House(2012)	Action
7	GIA	18+
8	Deadline 2009	18+
9	The Dirty Picture	18+
10	Marley and me	Romance

There are main 3 types of joins:

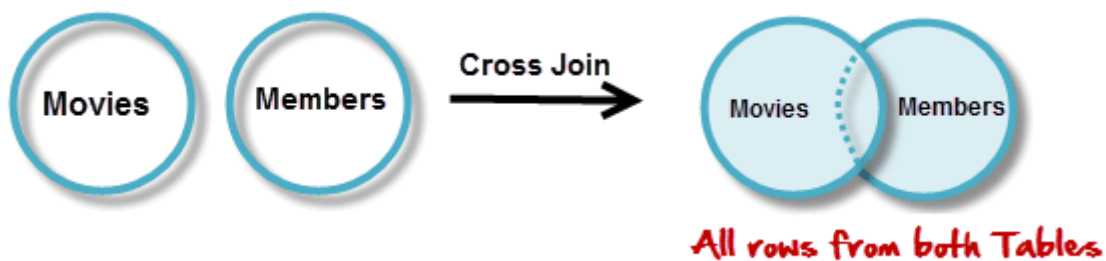
- Cross Join
- Inner Join
- Outer Join(left, right, full)

## CROSS JOIN:

Cross JOIN is a simplest form of JOINS which matches each row from one database table to all rows of another.

In other words it gives us combinations of each row of first table with all records in second table.

Suppose we want to get all member records against all the movie records, we can use the script shown below to get our desired results.

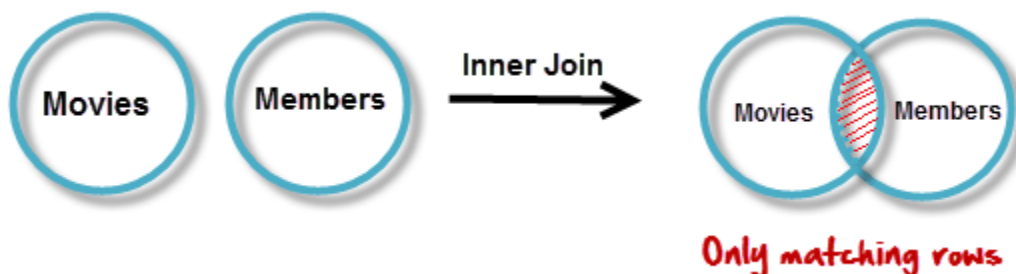


```
SELECT * FROM `movies` CROSS JOIN `members`
```

## INNER JOIN:

The inner JOIN is used to return rows from both tables that satisfy the given condition.

Suppose, you want to get list of members who have rented movies together with titles of movies rented by them. You can simply use an INNER JOIN for that, which returns rows from both tables that satisfy with given conditions.



```
SELECT members.`first_name`, members.`last_name`, movies.`title`  
FROM members ,movies  
WHERE movies.`id` = members.`movie_id`
```

Note the above results script can also be written as follows to achieve the same results.

```
SELECT A.`first_name`, A.`last_name`, B.`title`  
FROM `members` AS A  
INNER JOIN `movies` AS B  
ON B.`id` = A.`movie_id`
```

## OUTER JOINS:

MySQL Outer JOINS return all records matching from both tables.

It can detect records having no match in joined table. It returns **NULL** values for records of joined table if no match is found.

Sounds Confusing? Let's look into an example -



The LEFT JOIN returns all the rows from the table on the left even if no matching rows have been found in the table on the right. **Where no matches have been found in the table on the right, NULL is returned.**

```
SELECT A.`title`, B.`first_name`, B.`last_name`  
FROM `movies` AS A  
LEFT JOIN `members` AS B  
ON B.`movie_id` = A.`id`
```

## RIGHT JOIN:

RIGHT JOIN is obviously the opposite of LEFT JOIN. The RIGHT JOIN returns all the columns from the table on the right even if no matching rows have been found in the table on the left. Where no matches have been found in the table on the left, NULL is returned.

In our example, let's assume that you need to get names of members and movies rented by them. Now we have a new member who has not rented any movie yet



```
SELECT B.`title`, A.`first_name`, A.`last_name`  
FROM `members` AS A  
RIGHT JOIN `movies` AS B  
ON B.`id` = A.`movie_id`
```

### "ON" and "USING" clauses

In above JOIN query examples, we have used ON clause to match the records between table.

USING clause can also be used for the same purpose. The difference with **USING** is it **needs to have identical names for matched columns in both tables**.

In "movies" table so far we used its primary key with the name "id". We referred to same in "members" table with the name "movie\_id".

Let's rename "movies" table's "id" field to have the name "movie\_id". We do this in order to have identical matched field names.



```
ALTER TABLE `movies` CHANGE `id` `movie_id` INT( 11 ) NOT NULL  
AUTO_INCREMENT;
```

Next let's use USING with above LEFT JOIN example.

```
SELECT A.`title`, B.`first_name`, B.`last_name`  
FROM `movies` AS A  
LEFT JOIN `members` AS B  
USING ( `movie_id` )
```

Apart from using **ON** and **USING with JOINS** you can use many other MySQL clauses like **GROUP BY**, **WHERE** and even functions like **SUM**, **AVG**, etc.

## Summary

- JOINS allow us to combine data from more than one table into a single result set.
- JOINS have better performance compared to sub queries
- INNER JOINS only return rows that meet the given criteria.
- OUTER JOINS can also return rows where no matches have been found. The unmatched rows are returned with the NULL keyword.
- The major JOIN types include Inner, Left Outer, Right Outer, Cross JOINS etc.
- The frequently used clause in JOIN operations is "ON". "USING" clause requires that matching columns be of the same name.
- JOINS can also be used in other clauses such as GROUP BY, WHERE, SUB QUERIES, AGGREGATE FUNCTIONS etc.

## What is mysql database storage engines?

Storage engines are MySQL components that handle the SQL operations for different table types. InnoDB is the default (for v5.7) and most general-purpose storage engine. MySQL storage engines include both those that handle **transaction-safe** tables and those that handle **nontransaction-safe** tables.

Different storage engines available, there are few reasons not to use either the MyISAM or InnoDB engine types. MyISAM will do in most situations, but if you have a high number of updates or inserts compared to your searches and selects then you will get better performance out of the InnoDB engine. To get the best performance out of InnoDB you need to tweak the parameters for your server, otherwise there is no reason not to use it.

The MERGE engine is an exceedingly effective way of querying data from multiple, identically defined, tables. The MEMORY engine is the best way to perform a large number of complex queries on data that would be inefficient to search on a disk based engine. The CSV engine is a great way to export data that could be used in other applications. BDB is excellent for data that has a unique key that is frequently accessed.

```
mysql> SHOW ENGINES;
```

```
+-----+-----+-----+-----+-----+
---+-----+
| Engine   | Support | Comment                                     | Transactions |
XA | Savepoints |
+-----+-----+-----+-----+-----+
---+-----+
| InnoDB   | YES    | Supports transactions, row-level locking, and foreign keys
| YES      | YES    | YES    |
| MRG_MYISAM | YES    | Collection of identical MyISAM tables
NO      | NO     | NO     |
| BLACKHOLE | YES    | /dev/null storage engine (anything you write to it
disappears) | NO     | NO     | NO     |
```

```
| CSV      | YES    | CSV storage engine                                | NO      | NO
| NO      |
| MEMORY   | YES    | Hash based, stored in memory, useful for temporary
tables    | NO      | NO      | NO      |
| FEDERATED | NO     | Federated MySQL storage engine                    |
NULL      | NULL   | NULL      |
| ARCHIVE  | YES    | Archive storage engine                            | NO      |
NO | NO      |
| MyISAM   | DEFAULT | Default engine as of MySQL 3.23 with great
performance | NO      | NO      | NO      |
+-----+-----+-----+-----+-----+
---+-----+-----+
```

Feature	MyISAM	Memory	InnoDB	Archive	NDB
Storage limits	256TB	RAM	64TB	None	384EB
Transactions	No	No	Yes	No	Yes
Locking granularity	Table	Table	Row	Row	Row
MVCC	No	No	Yes	No	No
Geospatial datatype support	Yes	No	Yes	Yes	Yes
Geospatial indexing support	Yes	No	No	No	No
B-tree indexes	Yes	Yes	Yes	No	Yes
Hash indexes	No	Yes	No	No	Yes
Full-text search indexes	Yes	No	No	No	No
Clustered indexes	No	No	Yes	No	No
Data caches	No	N/A	Yes	No	Yes
Index caches	Yes	N/A	Yes	No	Yes
Compressed data	Yes <sup>[a]</sup>	No	Yes <sup>[b]</sup>	Yes	No
Encrypted data <sup>[c]</sup>	Yes	Yes	Yes	Yes	Yes
Cluster database support	No	No	No	No	Yes
Replication support <sup>[d]</sup>	Yes	Yes	Yes	Yes	Yes
Foreign key support	No	No	Yes	No	No
Backup / point-in-time recovery <sup>[e]</sup>	Yes	Yes	Yes	Yes	Yes
Query cache support	Yes	Yes	Yes	Yes	Yes
Update statistics for data dictionary	Yes	Yes	Yes	Yes	Yes

<sup>[a]</sup> Compressed MyISAM tables are supported only when using the compressed row format. Tables using the compressed row format with MyISAM are read only.

<sup>[b]</sup> Compressed InnoDB tables require the InnoDB Barracuda file format.

<sup>[c]</sup> Implemented in the server (via encryption functions), rather than in the storage engine.

<sup>[d]</sup> Implemented in the server, rather than in the storage product

<sup>[e]</sup> Implemented in the server, rather than in the storage product

## MYSQL functions.

### 1.FIND\_IN\_SET():

MySQL FIND\_IN\_SET() returns the position of a string if it is present (as a substring) within a list of strings. The string list itself is a string contains substrings separated by ',' (comma) character.

This function returns 0 when search string does not exist in the string list and returns NULL if either of the arguments is NULL.

#### Syntax:

FIND\_IN\_SET (search string, string list)

#### Arguments

Name	Description
search string	A string which is to be looked for in following a list of arguments.
string list	List of strings to be searched if they contain the search string.

### 2.GROUP\_CONCAT():

MySQL GROUP\_CONCAT() function returns a string with concatenated non-NULL value from a group.

Returns NULL when there are no non-NULL values.

#### Syntax:

GROUP\_CONCAT(expr);

Where expr is an expression.

### 3.DISTINCT:

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

The SELECT DISTINCT statement is used to return only distinct (different) values.

#### SELECT DISTINCT Syntax

```
SELECT DISTINCT column1, column2, ...  
FROM table_name;
```

## Select students who have maximum marks.

Id | name | marks

1	Vishal	100.00
2	Vijay	101.00
3	Govind	101.00
4	Akash	90.00

```
SELECT * FROM `students` WHERE marks = (SELECT MAX(marks) FROM students)
```

## Select students who have second highest marks.

```
SELECT * FROM `students` WHERE marks = (SELECT MAX(marks) FROM students  
WHERE marks < (SELECT MAX(marks) FROM students))
```

## Display rank of students on the base of marks.

```
SELECT *, @i:=@i+1 AS rank FROM students ORDER BY marks DESC
```

Real way:

```
SELECT *, FIND_IN_SET( marks, ( SELECT GROUP_CONCAT( DISTINCT marks ORDER  
BY marks DESC ) FROM students ) ) AS rank FROM students
```

## What is database normalization? Types of database normalization.

Reference from below links:

<https://www.guru99.com/database-normalization.html>

<https://www.w3schools.in/dbms/database-normalization/>

Database normalization is a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data.

Normalization split a large table into smaller tables and define relationships between them to increases the clarity in organizing data.

### Some facts about database normalization

- The words normalization and normal form refers to the structure of database.
- Normalization was developed by IBM researcher E.F. Codd In the 1970s.
- Normalization increases the clarity in organizing data in Database.

Normalization of a Database is achieved by following a set of rules called '**forms**' in creating the database.

### Database normalization rules

Database normalization process are divided into following normal form:

- [First Normal Form \(1NF\)](#)
- [Second Normal Form \(2NF\)](#)
- [Third Normal Form \(3NF\)](#)
- [Boyce-Codd Normal Form \(BCNF\)](#)
- [Fourth Normal Form \(4NF\)](#)
- [Fifth Normal Form \(5NF\)](#)

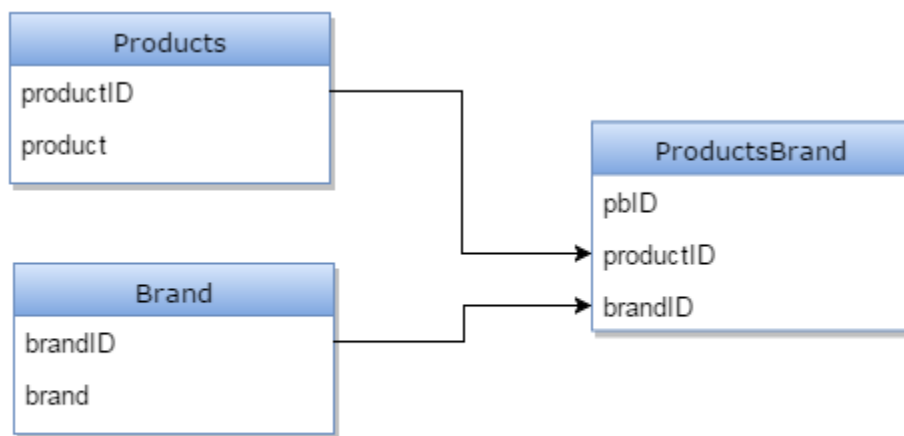


## **First Normal Form (1NF):**

Each column is unique in 1NF.

## **Second Normal Form (2NF):**

The entity should be considered already in 1NF and all attributes within the entity should depend solely on the unique identifier of the entity.



Second Normal Form (2NF)

## **Third Normal Form (3NF):**

The entity should be considered already in 2NF and no column entry should be dependent on any other entry (value) other than the key for the table.

If such an entity exists, move it outside into a new table.

3NF is achieved are considered as the database is normalized.

## **Boyce-Codd Normal Form (BCNF):**

3NF and all tables in the database should be only one primary key.

**Fourth Normal Form (4NF):**

Tables cannot have multi-valued dependencies on a Primary Key.

**Fifth Normal Form (5NF):**

Composite key shouldn't have any cyclic dependencies.

Well this is a highly simplified explanation for Database Normalization. One can study this process extensively though. After working with databases for some time you'll automatically create Normalized databases. As, it's logical and practical.

## What is indexing?

Reference from below links:

<https://www.tutorialspoint.com/mysql/mysql-indexes.htm>

<https://stackoverflow.com/questions/3567981/how-do-mysql-indexes-work>

A database index is a data structure that improves the speed of operations in a table. Indexes can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records.

While creating index, it should be taken into consideration which all columns will be used to make SQL queries and create one or more indexes on those columns.

Practically, indexes are also a type of tables, which keep primary key or index field and a pointer to each record into the actual table.

The users cannot see the indexes, they are just used to speed up queries and will be used by the Database Search Engine to locate records very fast.

The INSERT and UPDATE statements take more time on tables having indexes, whereas the SELECT statements become fast on those tables. The reason is that while doing insert or update, a database needs to insert or update the index values as well.

### Simple and Unique Index

You can create a unique index on a table. A unique index means that two rows cannot have the same index value. Here is the syntax to create an Index on a table.

```
CREATE UNIQUE INDEX index_name  
ON table_name ( column1, column2,...);
```

You can use one or more columns to create an index.

For example, we can create an index on **tutorials\_tbl** using **tutorial\_author**.

```
CREATE UNIQUE INDEX AUTHOR_INDEX  
ON tutorials_tbl (tutorial_author)
```

You can create a simple index on a table. Just omit the **UNIQUE** keyword from the query to create a simple index. A Simple index allows duplicate values in a table.

If you want to index the values in a column in a descending order, you can add the reserved word **DESC** after the column name.

```
mysql> CREATE UNIQUE INDEX AUTHOR_INDEX  
ON tutorials_tbl (tutorial_author DESC)  
ALTER command to add and drop INDEX
```

There are four types of statements for adding indexes to a table –

- **ALTER TABLE tbl\_name ADD PRIMARY KEY (column\_list)** – This statement adds a **PRIMARY KEY**, which means that the indexed values must be unique and cannot be NULL.
- **ALTER TABLE tbl\_name ADD UNIQUE index\_name (column\_list)** – This statement creates an index for which the values must be unique (except for the NULL values, which may appear multiple times).
- **ALTER TABLE tbl\_name ADD INDEX index\_name (column\_list)** – This adds an ordinary index in which any value may appear more than once.
- **ALTER TABLE tbl\_name ADD FULLTEXT index\_name (column\_list)** – This creates a special FULLTEXT index that is used for text-searching purposes.

The following code block is an example to add index in an existing table.

```
mysql> ALTER TABLE testalter_tbl ADD INDEX (c);
```

You can drop any INDEX by using the **DROP** clause along with the ALTER command.

Try out the following example to drop the above-created index.

```
mysql> ALTER TABLE testalter_tbl DROP INDEX (c);
```

You can drop any INDEX by using the DROP clause along with the ALTER command.

### ALTER Command to add and drop the PRIMARY KEY

You can add a primary key as well in the same way. But make sure the Primary Key works on columns, which are NOT NULL.

The following code block is an example to add the primary key in an existing table. This will make a column NOT NULL first and then add it as a primary key.

```
mysql> ALTER TABLE testalter_tbl MODIFY i INT NOT NULL;  
mysql> ALTER TABLE testalter_tbl ADD PRIMARY KEY (i);
```

You can use the ALTER command to drop a primary key as follows –

```
mysql> ALTER TABLE testalter_tbl DROP PRIMARY KEY;
```

To drop an index that is not a PRIMARY KEY, you must specify the index name.

### Displaying INDEX Information

You can use the **SHOW INDEX** command to list out all the indexes associated with a table. The vertical-format output (specified by \G) often is useful with this statement, to avoid a long line wraparound –

Try out the following example –

```
mysql> SHOW INDEX FROM table_name\G
```

## What is Procedures?

Reference from below links:

<https://www.sitepoint.com/stored-procedures-mysql-php/>

<http://www.mysqltutorial.org/php-calling-mysql-stored-procedures/>

<https://www.9lessons.info/2010/07/stored-procedure-lesson.html>

## What is method overloading and overriding?

Reference from below links:

<https://www.techflirt.com/tutorials/oop-in-php/overloading-and-overriding.html>

Overloading and Overriding are forms of polymorphism in OOP. According to Object Oriented Programming (OOP) concept if a class has methods of the same name but different parameters then we say that we are overloading that method. Also if we were to create a method in the child class having the same name, same number of parameters and the same access specifier as in its parent then we can say that we are doing method overriding.

As we know that PHP is not type strict, means if we implement overloading in C++ or Java, the function will look like `add(int, float)` is different from `add(float, int)` or even `add(float, int, int)` but it is not possible in PHP. Actually polymorphism is not easy in PHP. In this post, you will learn how to handle polymorphism in PHP.

## What is polymorphism?

Polymorphism is a long word for a very simple concept.

Polymorphism is basically derived from the Greek which means 'many forms'.

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

The beauty of polymorphism is that the code working with the different classes does not need to know which class it is using since they're all used the same way.

A real world analogy for polymorphism is a button. Everyone knows how to use a button: you simply apply pressure to it. What a button "does," however, depends on what it is connected to and the context in which it is used -- but the result does not affect how it is used. If your boss tells you to press a button, you already have all the information needed to perform the task.

In the programming world, polymorphism is used to make applications more modular and extensible. Instead of messy conditional statements describing different courses of action, you create interchangeable objects that you select based on your needs. That is the basic goal of polymorphism.

Try below example:



```
<?php
interface Shape {
    public function getArea();
}

class Square implements Shape {
    private $width;
    private $height;

    public function __construct($width, $height) {
        $this->width = $width;
        $this->height = $height;
    }

    public function getArea(){
        return $this->width * $this->height;
    }
}

class Circle implements Shape {
    private $radius;

    public function __construct($radius) {
        $this->radius = $radius;
    }

    public function getArea(){
        return 3.14 * $this->radius * $this->radius;
    }
}

function calculateArea(Shape $shape) {
    return $shape->getArea();
}

$square = new Square(5, 5);
$circle = new Circle(7);

echo calculateArea($square), "<br/>";
echo calculateArea($circle);
?>
```

## Thank You

This document is prepared by me but all content are not my, and also added reference sites in answers,

The purpose of this document is just prepare interview instantly and easy to learn and link reference concepts.

If you have any query or any confusions or suggestions please feel free to ask in email: [tcodeburner@gmail.com](mailto:tcodeburner@gmail.com) I will reply as soon as possible, Thank you again...