**PROJECT NAME: INTERNATIONAL LANGAUAGE SCHOOL**

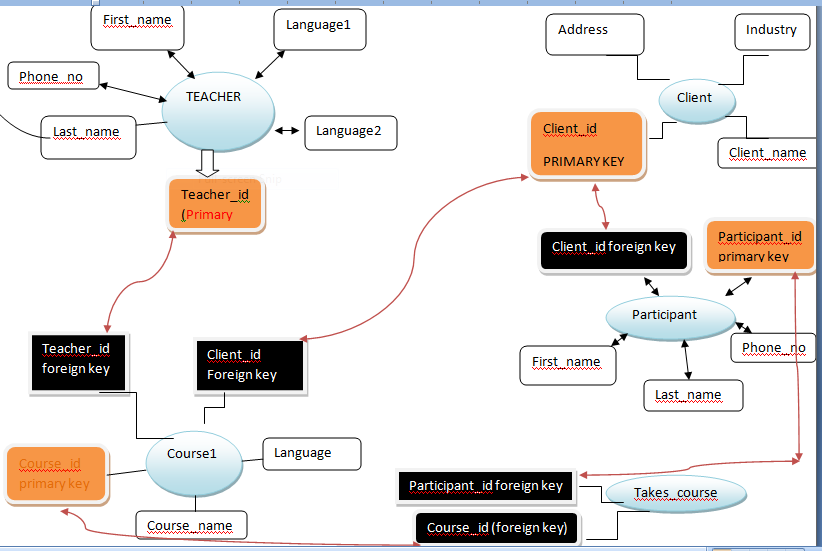
**INTRODUCTION:**

A International language study encourages and builds mental flexibility,superiority in concept formation,those who have studied these language retain these cognitive benefits well into adulthood and old age.

One of the immediate [advantages of learning a new language](https://www.studyfrenchspanish.com/language-benefits/) is that it inspires creativity in the student. Many companies worldwide, especially those from developed nations, break into new markets every day to expand their business and achieve a global reach.

Hence, organizations are always lookout for qualified people who can speak one or more international languages other than English

**VIEW DIAGRAM: ENTITY RELATIONSHIP DIAGRAM**



## \* Finally, some code!

Once we have our environment set up, we need to create a database on our MySQL server. This is not too complicated, we just have to use the [CREATE DATABASE](https://dev.mysql.com/doc/refman/8.0/en/create-database.html" \t "_blank) statement. Note that all SQL statements are closed with a semicolon ‘;’. This lets the client know that this is the end of our statement, which is especially useful when writing longer and more complex statements, like [nested queries,](https://www.w3resource.com/sql/subqueries/nested-subqueries.php" \t "_blank) but is essential for all statements (except the USE command, which we will see next).

CREATE DATABASE school;

choose which database we want to use. To do this we use the [use](https://dev.mysql.com/doc/refman/8.0/en/database-use.html" \t "_blank) command:

USE school

MySQL supports a wide range of [data types,](https://dev.mysql.com/doc/refman/8.0/en/data-types.html" \t "_blank) from simple [integers](https://dev.mysql.com/doc/refman/8.0/en/integer-types.html" \t "_blank) and [strings](https://dev.mysql.com/doc/refman/8.0/en/string-types.html" \t "_blank) to [BLOBs](https://dev.mysql.com/doc/refman/8.0/en/blob.html" \t "_blank) . We will just be using a small subset of these in our database.

We’re going to use the following;

* **[INT](https://dev.mysql.com/doc/refman/8.0/en/integer-types.html" \t "_blank)**— this is an integer, a whole number. We’re mainly going to use this for our ID fields (which will be our primary keys).
* **[VARCHAR](https://dev.mysql.com/doc/refman/8.0/en/char.html" \t "_blank)**— this is a string field of varying length, which we will use for storing text. We need to tell the RDBMS how long the VARCHAR will be, so we can define VARCHAR(20),

**Coding and Implementing a Relational Database using MySQL**

***1.***Create table TEACHER:

**create table teacher(teacher\_id int primary key,**

**first\_name varchar(20),last\_name varchar(20),language1 varchar(20),language2 varchar(30),phone\_no int);**

**Query OK, 0 rows affected (0.46 sec)**

We have also added some [constraints](https://www.w3schools.com/sql/sql_constraints.asp" \t "_blank) to some of the attributes in our table. Contraints specify rules for data in a table, and will constrain what the RDBMS will allow us to do to that particular attribute.

We have added [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp" \t "_blank) to first\_name, last\_name and language\_1 — this means that the table will not accept a record where any of those attributes are set to NULL. Those attributes require a non-NULL value for every record. This makes sense, as our teachers will require a first name and last name, and to teach in a language school they need to be able to offer at least one language.

We have also set the teacher\_id field as our PRIMARY KEY. In practice this is really just a combination of NOT NULL and UNIQUE, but it is important to define one primary key (which again, can be a single attribute or a combination of different attributes) for each table.

If we now try to use a SELECT \* FROM table statement to see what data is in our table , we should receive ‘Empty set’ as our response This shows us our table exists (otherwise we would receive an error), but is still empty. Exactly as we would expect!

MariaDB [final]> desc teacher;

+------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+------------+-------------+------+-----+---------+-------+

| teacher\_id | int(11) | NO | PRI | NULL | |

| first\_name | varchar(20) | YES | | NULL | |

| last\_name | varchar(20) | YES | | NULL | |

| language1 | varchar(20) | YES | | NULL | |

| language2 | varchar(30) | YES | | NULL | |

| phone\_no | int(11) | YES | | NULL | |

+------------+-------------+------+-----+---------+-------+

6 rows in set (0.01 sec)



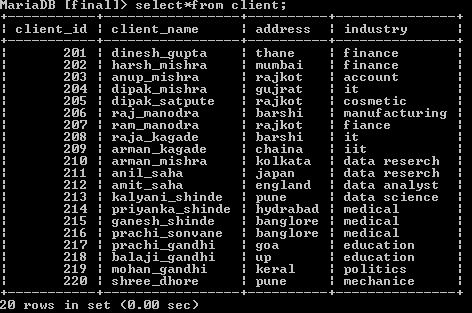
**2.Create a table client:**

**CREATE TABLE client ( client\_id INT PRIMARY KEY,**

**client\_name VARCHAR(40) NOT NULL,**

**address VARCHAR(60) NOT NULL,**

**industry VARCHAR(20));**



**3.Create table course1:**

**MariaDB [final]> create table course1(course\_id int auto\_increment primary key,**

**course\_name varchar(20),**

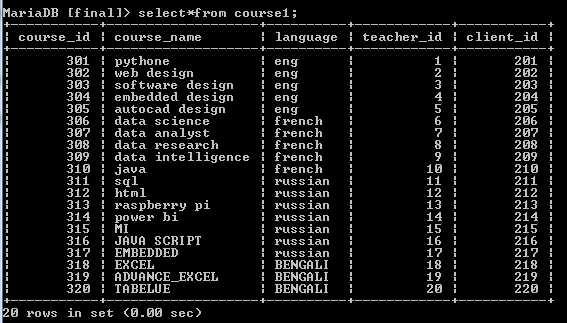
**language varchar(20),teacher\_id int,**

**constraint fhp foreign key(teacher\_id) references teacher(teacher\_id) on delete set null,client\_id int,constraint fk foreign key(client\_id) references client(client\_id)on delete set null);**

Query OK, 0 rows affected (0.39 sec)

The next step is to establish the relationships between these by setting up our foreign keys,This means that any course can only be taught by one teacher, but that a teacher may teach many courses. The foreign key perfectly enables us to capture this relationship in our database.

We also have a new constraint here — [ON DELETE SET NULL](https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html" \l "foreign-key-referential-actions" \t "_blank). This tells MySQL what we want to do when a record in the client table is deleted — in this case the value of the client attribute for that participant will be set to NULL, but the record will remain in the participants table. The other option is [ON DELETE CASCADE](https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html" \l "foreign-key-referential-actions" \t "_blank). If we used this here then when a record is deleted from the client table all participants linked to that client via this foreign key relationship would also be deleted.

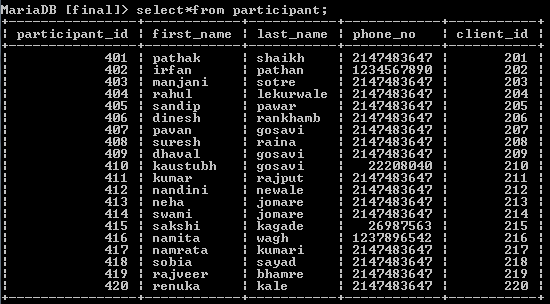


**4.Create table PARTICIPANT:**

**MariaDB [final]> create table participant(participant\_id int primary key,**

**first\_name varchar(20) not null,last\_name varchar(20) not null,**

**phone\_no int,client\_id int,constraint fkp foreign key(client\_id) references client(client\_id) on delete**

**set null);Query OK, 0 rows affected (0.25 sec)**

**Create tableTakes\_course:**

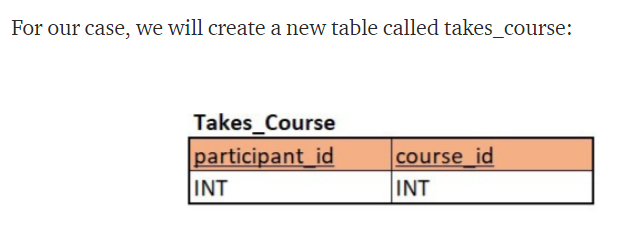
**MariaDB [final]> create table takes\_course(participant\_id int,**

**course\_id int,primary key(participant\_id,course\_id),**

**foreign key(participant\_id) references partici**

**pant(participant\_id) on delete cascade,foreign key(course\_id) references course1**

**(course\_id) on delete cascade);**

****

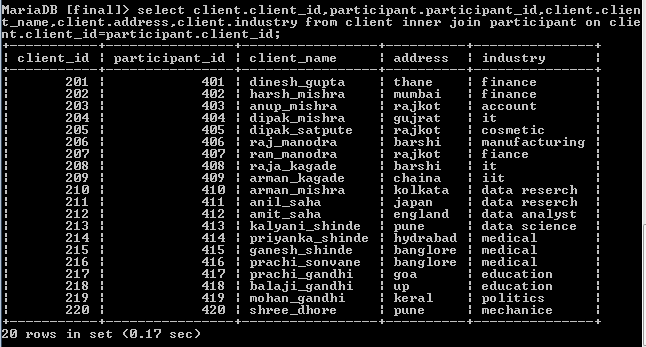
. The other option is [ON DELETE CASCADE](https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html" \l "foreign-key-referential-actions" \t "_blank). If we used this here then when a record is deleted from the client table all participants linked to that client via this foreign key relationship would also be deleted.

# SQL INNER JOIN TABLES Working with Queries in SQL Server,

# select data from two or more related tables. n a relational database, multiple tables are connected to each other via foreign key constraints.

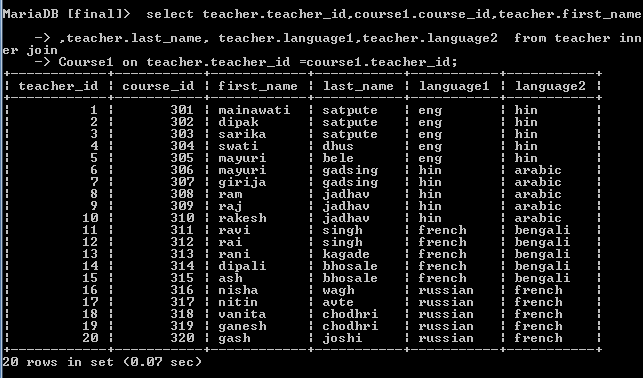
# If you want to retrieve data from related tables simultaneously, SQL JOIN tables queries will be useful.

**select client.client\_id,participant.participant\_id,client.client\_name,client.address,client.industry from client inner join participant on client.client\_id=participant.client\_id;**



**INNER JOIN TABLES TEACHER AND COURSE1.**

**select teacher.teacher\_id,course1.course\_id,teacher.first\_name,teacher.last\_name, teacher.language1,teacher.language2 from teacher inner joinCourse1 on teacher.teacher\_id =course1.teacher\_id;**

****

## Performing Multiple-Table Retrievals with Subqueries

A subquery is a SELECT statement written within parentheses and nested inside another statement.

Subqueries can return different types of information:

* A scalar subquery returns a single value.
* A column subquery returns a single column of one or more values.
* A row subquery returns a single row of one or more values.
* A table subquery returns a table of one or more rows of one or more columns

**MariaDB [final]> SELECT CLIENT\_ID,CLIENT\_NAME,ADDRESS FROM CLIENT WHERE ADDRES**

**S IN(SELECT ADDRESS FROM CLIENT WHERE CLIENT\_NAME='DINESH\_GUPTA');**

+-----------+--------------+---------+

| CLIENT\_ID | CLIENT\_NAME | ADDRESS |

+-----------+--------------+---------+

| 201 | dinesh\_gupta | thane |

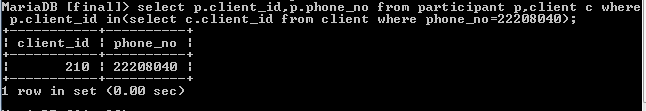
+-----------+--------------+---------+

1 row in set (0.00 sec)

**Multiple subqueries from two table:**

MariaDB [final]> select p.client\_id,p.phone\_no from participant p,client c where

p.client\_id in(select c.client\_id from client where phone\_no=22208040);



MariaDB [final]> select p.client\_id,p.phone\_no,p.first\_name,p.last\_name from par

ticipant p,client c where p.client\_id in(select c.client\_id from client where ph

one\_no=22208040 or phone\_no=26987563);

+-----------+----------+------------+-----------+

| client\_id | phone\_no | first\_name | last\_name |

+-----------+----------+------------+-----------+

| 210 | 22208040 | kaustubh | gosavi |

| 215 | 26987563 | sakshi | kagade |

+-----------+----------+------------+-----------+

