TYPES OF KEYS,FOREIGN KEY

Candidate Key:

A candidate key is a column or a set of columns that can qualify as a primary key in the database. There can be multiple SQL candidate keys in a database relation and each candidate can work as a primary key for the table.

student_table

s_id	name	age	grade	email id	ssn
1	JOHN	21	72	john@gmail.com	123478
2	MIKE	20	68	mike@yahoo.com	124563
3	KATE	22	86	kate@gmail.com	145678
4	ANDY	21	94	andy@gmail.com	167982

If you consider the table above here **s_Id, email id, ssn(social security number)** all are unique identifiers for the table so all these columns represent **candidate keys**.

Candidate key is classified into

- **Primary key**
- **Alternate key or Secondary key**

Here s_id is represented as the primary key so email id and ssn are considered as Alternate Key or secondary key.

Now primary key is further classified as

- Simple primary key
- Composite primary key

Simple primary key: A simple primary key is made up of a single field only, where only one column is a unique identifier for the table. If you considered the table below here s_id is the unique identifier and is represented as simple primary key

Student table

s_id	name	age	grade
1	JOHN	21	72
2	MIKE	20	68
3	KATE	22	86
4	ANDY	21	94

Composite primary key:

A composite primary key combines more than one field to make a unique value.

Student table

s_id	name	age	grade
1	JOHN	21	72
2	MIKE	20	68
3	KATE	22	86
4	ANDY	21	94

4	KATE	23	65

If you observe the table above here there are two students with id 4 in this table. In this situation it is difficult to identify the row uniquely with only student id but with the combination of student id and name it will be easy to identify the row uniquely this is only called as composite key

You can create a MySQL composite Primary Key in the following two ways:

- During table creation using **CREATE TABLE statements**.
- After creating tables using the ALTER **TABLE statement**.

- **Creating MySQL Composite Primary Key while Table Creation**

To create a MySQL composite primary key during table creation you can follow the steps.

Suppose you want to create a **student** with the fields (s_id, name, age, grade) table with a MySQL Composite primary key (s_id, name).

```
CREATE TABLE student (
  s_id INT,
  name VARCHAR(45),
  age INT,
  grade INT,
  PRIMARY KEY (s_id, name)
);
```

In the above, a MySQL Composite primary is created with the column names **s_id** and **name**.

You can verify the same using the command as below:

```
DESCRIBE student;
```

Output:

Field	Type	Null	Key
s_id	int	NO	PRI
name	varchar(45)	NO	PRI
age	int	YES	
grade	int	YES	

After the successful execution of the above query, you can see that the Key column has two **PRI**. It means you have successfully added the MySQL Composite Primary key on **s_id** and **name** columns.

Next, you need to insert the values into this table as given below:

```
INSERT INTO student (s_id, name, age, grade) VALUES (1,
'JOHN', 21, 72);
INSERT INTO student (s_id, name, age, grade) VALUES (2,
'MIKE', 20, 68);
INSERT INTO student (s_id, name, age, grade) VALUES (3,
'KATE', 22, 86);
INSERT INTO student (s_id, name, age, grade) VALUES (4,
'ANDY', 21, 94);
INSERT INTO student (s_id, name, age, grade) VALUES (4,
'KATE', 23, 65);
```

Next, you can execute the following command to display the data in the table:

```
SELECT * FROM student;
```

Output:

Student table

s_id	name	age	grade
1	JOHN	21	72
2	MIKE	20	68
3	KATE	22	86
4	ANDY	21	94
4	KATE	23	65

To understand Composite key with better clarity
If you try to execute below query

```
INSERT INTO student (s_id, name, age, grade) VALUES (1,
'JOHN', 21, 72);
```

Output:

```
Error Code: 1062. Duplicate entry '1-JOHN' for key
'student.PRIMARY'
```

You will get the above error because you are trying to add the same student id and name into the student table.

If you try to execute below query

```
INSERT INTO student (s_id, name, age, grade) VALUES (1,
'ANDY', 21, 72);
```

The data will be inserted successfully as the combination of name and id is different.

- **Adding MySQL Composite Primary Key in Existing Table**

Now to do this let's delete the constraint primary key from the above table. This can be achieved using the query below

```
ALTER TABLE student DROP primary key;
```

To just check whether primary key constraint is delete let's describe the table

```
DESCRIBE student;
```

Output:

Field	Type	Null	key
s_id	int	NO	
name	varchar(45)	NO	
age	int	YES	
grade	int	YES	

As you can clearly see from the above table, s_id and name are not the primary key in the key column.

Now, you can execute the **ALTER TABLE statement** to add a **MySQL Composite Primary key** as follows:

```
ALTER TABLE sql_notes.student ADD PRIMARY KEY(s_id, name);
```

You can verify the MySQL Composite Primary key is added into a table or not using the following command:

You can use the following command to verify if the MySQL Composite Primary key has been added to the table.

```
DESCRIBE sql_notes.student;
```

Output:

Field	Type	Null	key
s_id	int	NO	PRI
name	varchar(45)	NO	PRI
age	int	YES	
grade	int	YES	

In the output, we can see that the key column has **2 PRI**, which means we have successfully added the composite primary key to **s_id** and **name** columns.

Foreign Key:

Foreign Key helps establish database relationships and maintain referential integrity. They help link one or more columns in one table to another table. Here's how to add foreign key in MySQL.

If an attribute is a **primary key** in one table but was not used as a primary key in another table then the attribute which is not a primary key in the other table is called a foreign key.

You can add foreign key to the table in two ways:

- During Table Creation using **CREATE Command**
- After Table creation using **ALTER Command**

- **Adding foreign key during the creation of table**

If you consider the table student and course below, here c_id is the fireign key which is referring to the primary key c_id in the courses table.

The **course table is the parent table** which contains primary key which is acting like foriegn key is student table

The **student table is the child table** which contains foriegn key

Student

s_id	name	age	grade	c_id
1	JOHN	21	72	1J
2	MIKE	20	68	1J
3	KATE	22	86	2P
4	ANDY	21	94	4PFS

Course

c_id	name	rating
1J	Java	4.6
2P	Python	4.8
3JFS	Java Full Stack	4.8
4PFS	Python Full Stack	4.9

- First child table that is course table need to be created as shown below

```
CREATE TABLE sql_notes.courses(
c_id VARCHAR(5) PRIMARY KEY,
name VARCHAR(10),
rating decimal
);
```

- Create parent table with foreign key

```
CREATE TABLE sql_notes.student(
s_id tinyint PRIMARY KEY,
name VARCHAR(10),
age tinyint,
grade decimal(4,2),
c_id VARCHAR(5),
FOREIGN KEY(c_id) REFERENCES sql_notes.courses(c_id)
);
```

You can verify the MySQL Composite Primary key is added into a table or not using the following command:

You can use the following command to verify if the MySQL foreign key has been added to the table.

```
DESCRIBE sql_notes.student;
```

Output:

Field	Type	Null	Key
s_id	tinyint	NO	PRI
name	varchar(10)	YES	
age	tinyint	YES	
grade	decimal(4,2)	YES	
c_id	varchar(5)	YES	MUL

Here If you observe the key of c_id is MUL. MUL is basically an index that is neither a primary key nor a unique key. The name comes from “multiple” because multiple occurences of the same value are allowed. IT indicates that c_id is foreign key.

Note: If you try to add any c_id in the student table which is not there in the course table that time you will get an error.

Now let's insert the values in the table

Course table

```
INSERT INTO courses (c_id, name, rating) VALUES ('1J',
'Java', 4.6);
INSERT INTO courses (c_id, name, rating) VALUES ('2P',
'Python', 4.8);
INSERT INTO courses (c_id, name, rating) VALUES ('3JFS',
'Java Full Stack', 4.8);
INSERT INTO courses(c_id, name, rating) VALUES ('4PFS',
'Python Full Stack', 4.9);
```

Student table

```
INSERT INTO student (s_id, name, age, grade, c_id) VALUES
(1, 'JOHN', 21, 72, '1J');
INSERT INTO student (s_id, name, age, grade, c_id) VALUES
(2, 'MIKE', 20, 68, '1J');
INSERT INTO student (s_id, name, age, grade, c_id) VALUES
(3, 'KATE', 22, 86, '2P');
INSERT INTO student (s_id, name, age, grade, c_id) VALUES
(4, 'ANDY', 21, 94, '4PFS');
```

- After Table creation using **ALTER Command**

To understand this first let's drop the foreign key constraint available in the student table using the query below

```
ALTER TABLE student DROP FOREIGN KEY c_id;
```

Output:

0	**3 5**	**18:53:47**	**ALTER TABLE sql_notes.student DROP FOREIGN KEY c_id**	**Error Code: 1091. Can't DROP 'c_id'; check that column/key exists**	**0.000 sec**

You will get the above error because you have not defined a constraint for foreign key.

Now let's try to add constraint for the foreign key using alter command

```
ALTER TABLE sql_notes.student
ADD CONSTRAINT cIdFk FOREIGN KEY (c_id)
REFERENCES sql_notes.courses(c_id);
```

Now by executing the below query you can drop a foreign key constraint

```
ALTER TABLE sql_notes.student DROP FOREIGN KEY cIdFk ;
```

Now there is no foreign key constraint in the table let's add foreign key using alter command

To create a **FOREIGN KEY** constraint on the "c_id" column when the "student" table is already created, use the following MySQL:

```
ALTER TABLE student ADD FOREIGN KEY (c_id) REFERENCES courses(c_id);
```