Chapter 3

Data Definition Language (DDL)

"Database: the information you lose when your memory crashes."- Dave Barry

3.1. Creating Table with Foreign Key (Foreign key on Create Table)

The foreign key is used to link one or more than one tables together. It is also known as the referencing key. A foreign key matches the ***primary key*** field of another table. It means a foreign key field in one table refers to the primary key field of the other table. It identifies each row of another table uniquely that maintains the referential integrity in MySQL.

A foreign key makes it possible to create a parent-child relationship with the tables. In this relationship, the parent table holds the initial column values, and the column values of the child table reference the parent column values. MySQL allows us to define a foreign key constraint on the child table. The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column because it must be one of the values contained in the parent table.

Create these tables in the **Hospital** database:

```
MariaDB [hospital]> show tables;
+-------------------+
| Tables_in_hospital |
+-------------------+
| checkup           |
| doctor            |
| medicine          |
| patient           |
| prescription      |
+-------------------+
5 rows in set (0.001 sec)
```

The structures of each table are:

a. Patient

```
MariaDB [hospital]> desc patient;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| patient_code    | int(10)     | NO   | PRI | NULL    |       |
| patient_name    | varchar(25) | YES  |     | NULL    |       |
| patient_address | varchar(30) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

b. Doctor

```
MariaDB [hospital]> desc doctor;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| doc_code       | int(11)     | NO   | PRI | NULL    |       |
| doc_name       | varchar(20) | NO   |     | NULL    |       |
| specialization | varchar(20) | NO   |     | NULL    |       |
| gender         | char(10)    | YES  |     | NULL    |       |
| doc_adrress    | varchar(25) | YES  |     | NULL    |       |
| email          | varchar(15) | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
```

c. Medicine

The ENUMERATION / ENUM data type in MySQL is a string object. It allows us to limit the value chosen from a list of permitted values in the column specification at the time of table creation. It is short for enumeration, meaning each column may have one of the specified possible values. It uses numeric indexes (1, 2, 3…) to represent string values.

MySQL ENUM data type contains the following advantages:

1. Compact data storage where the column may have a limited set of specified possible values. Here, the string values are automatically used as a numeric index.
2. It allows readable queries and output because the numbers can be translated again to the corresponding string.
3. It can accept many data types like integer, floating-point, decimal, and string.

Enum syntax for columns :

```
CREATE TABLE table_name (
  col...
  col ENUM ('value_1','value_2','value_3', ....),
  col...
);
```

```
MariaDB [hospital]> create table medicine(
    -> med_code char(5) primary key,
    -> med_name varchar(20) not null,
    -> med_type enum('tablet', 'powder', 'pill', 'capsule', 'syrup') not null,
    -> stock int(4) not null,
    -> expired_date date not null);
Query OK, 0 rows affected (0.360 sec)
```

```
MariaDB [hospital]> desc medicine;
+-------------+-----------------------------------------------+------+-----+---------+-------+
| Field       | Type                                          | Null | Key | Default | Extra |
+-------------+-----------------------------------------------+------+-----+---------+-------+
| med_code    | char(5)                                       | NO   | PRI | NULL    |       |
| med_name    | varchar(20)                                   | NO   |     | NULL    |       |
| med_type    | enum('tablet','powder','pill','capsule','syrup') | NO   |     | NULL    |       |
| stock       | int(4)                                        | NO   |     | NULL    |       |
| expired_date | date                                          | NO   |     | NULL    |       |
+-------------+-----------------------------------------------+------+-----+---------+-------+
```

d. Prescription

```
MariaDB [hospital]> desc prescription;
+-----------+------------+------+-----+---------+-------+
| Field     | Type       | Null | Key | Default | Extra |
+-----------+------------+------+-----+---------+-------+
| pres_code | char(5)    | NO   | PRI | NULL    |       |
| pres_date | date       | NO   |     | NULL    |       |
| med_rules | varchar(5) | NO   |     | NULL    |       |
+-----------+------------+------+-----+---------+-------+
```

e. Checkup

Since in the checkup table, we have two foreign keys to connect with the patient and doctor table. The syntax for creating the foreign key is as follows:

FOREIGN KEY (column_name, ...) REFERENCES parent_tbl_name (column_name,...);

```
MariaDB [hospital]> create table checkup(
    -> checkup_code char(5) primary key,
    -> doc_code int(11) not null,
    -> patient_code int(10) not null,
    -> diagnosis varchar(30) not null,
    -> action varchar(30) not null,
    -> foreign key (doc_code)references doctor (doc_code),
    -> foreign key (patient_code) references patient (patient_code));
Query OK, 0 rows affected (0.402 sec)
```

```
MariaDB [hospital]> desc checkup;
+--------------+------------+------+-----+---------+-------+
| Field        | Type       | Null | Key | Default | Extra |
+--------------+------------+------+-----+---------+-------+
| checkup_code | char(5)    | NO   | PRI | NULL    |       |
| doc_code     | int(11)    | NO   | MUL | NULL    |       |
| patient_code | int(10)    | NO   | MUL | NULL    |       |
| diagnosis    | varchar(30) | NO   |     | NULL    |       |
| action       | varchar(30) | NO   |     | NULL    |       |
+--------------+------------+------+-----+---------+-------+
```

3.2. Renaming Table

Sometimes our table name is non-meaningful, so it is required to rename or change the table's name. MySQL provides a useful syntax that can rename one or more tables in the current database.

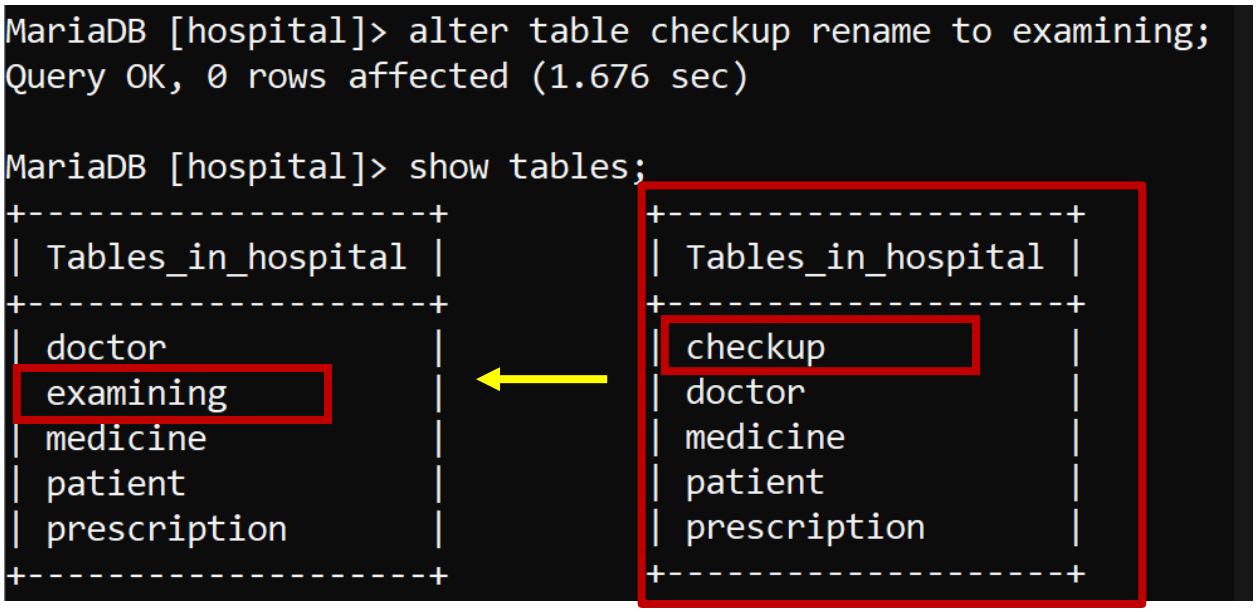
MySQL provides an ALTER…RENAME TO query with a query format:

```
ALTER TABLE old_table_name RENAME TO new_table_name;
```

In addition to the ALTER…RENAME TO query, there is also a RENAME command to change the table name and query format:

```
RENAME TABLE old table_name TO new table_name;
```

Example: We want to rename the checkup table to examining



3.3. Adding Columns to the Table

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name ADD column_name datatype;
```

Add a birthplace column in the patient table.

```
MariaDB [hospital]> alter table patient
    -> add birthplace varchar(20)
    -> ;
Query OK, 0 rows affected (0.393 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc patient;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| patient_code    | int(10)     | NO   | PRI | NULL    |       |
| patient_name    | varchar(25) | YES  |     | NULL    |       |
| patient_address | varchar(30) | YES  |     | NULL    |       |
| birthplace      | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

Add a gender column after the patient_address column in the patient table.

```
MariaDB [hospital]> alter table patient
    -> add gender varchar (10)
    -> after patient_address;
Query OK, 0 rows affected (0.261 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc patient;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| patient_code    | int(10)     | NO   | PRI | NULL    |       |
| patient_name    | varchar(25) | YES  |     | NULL    |       |
| patient_address | varchar(30) | YES  |     | NULL    |       |
| gender          | varchar(10) | YES  |     | NULL    |       |
| birthplace      | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

Exercise:

1. Add a phone_number (varchar (15)) column after the email column in the doctor table
2. Add a marital_status (varchar (10)) column after the specialization column in the doctor table
3. Add a cost column (int (15)) after the action column in the examining table
4. Add a price (int (15)) column after the expired_date column in the medicine table
5. Add a med_amount (int (3)) column after the med_rules in the prescription table

3.4. Changing Table Column

To change the name of a column in an existing table, you can use the ALTER…CHANGE command, with the following query format:

ALTER TABLE table_name CHANGE column_name_new_column_name data_type;

Rename the column patient_address to patient_addr in the patient table

```
MariaDB [hospital]> alter table patient
    -> change patient_address patient_addr varchar(30);
Query OK, 0 rows affected (0.169 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc patient;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| patient_code | int(10)     | NO   | PRI | NULL    |       |
| patient_name | varchar(25) | YES  |     | NULL    |       |
| patient_addr | varchar(30) | YES  |     | NULL    |       |
| gender       | varchar(10) | YES  |     | NULL    |       |
| birthplace   | varchar(20) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

Changing the data type of the gender column into an enum ('Male' and 'Female') in the doctor table

```
MariaDB [hospital]> alter table doctor
    -> change gender gender enum('Male','Female');
Query OK, 0 rows affected (1.705 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc doctor;
+----------------+----------------------+------+-----+---------+-------+
| Field          | Type                 | Null | Key | Default | Extra |
+----------------+----------------------+------+-----+---------+-------+
| doc_code       | int(11)              | NO   | PRI | NULL    |       |
| doc_name       | varchar(20)          | NO   |     | NULL    |       |
| specialization | varchar(20)          | NO   |     | NULL    |       |
| marital_status | varchar(10)          | YES  |     | NULL    |       |
| gender         | enum('Male','Female') | YES  |     | NULL    |       |
| doc_adrress    | varchar(25)          | YES  |     | NULL    |       |
| email          | varchar(15)          | NO   |     | NULL    |       |
+----------------+----------------------+------+-----+---------+-------+
```

Exercise:

1. Changing the data type of the gender column into an enum (‘Male’ and ‘Female’) in the patient table
2. Rename the column email to email_addr in the doctor table
3. Changing the data type of the marital_status column into an enum (‘Single’, ‘Married’ and ‘Divorce’) in the doctor table
4. Rename the med_rules to med_usage in the prescription table
5. Rename the price to med_price in the medicine table

3.5. Deleting Columns in Table

Sometimes, we want to remove single or multiple columns from the table. MySQL allows the ALTER TABLE DROP COLUMN statement to delete the column from the table. The following are the syntax to do this:

```
ALTER TABLE table_name DROP column_name;
```

```
MariaDB [hospital]> desc patient;
+--------------+----------------------+------+-----+---------+-------+
| Field        | Type                 | Null | Key | Default | Extra |
+--------------+----------------------+------+-----+---------+-------+
| patient_code | int(10)              | NO   | PRI | NULL    |       |
| patient_name | varchar(25)          | YES  |     | NULL    |       |
| patient_addr | varchar(30)          | YES  |     | NULL    |       |
| gender       | enum('Male','Female') | YES  |     | NULL    |       |
| birthplace   | varchar(20)          | YES  |     | NULL    |       |
+--------------+----------------------+------+-----+---------+-------+
```

*Above is the table structure of the patient table, and we want to delete the **birthplace** column. The command for deleting the column can be seen in the image below.*

```
MariaDB [hospital]> alter table patient
    -> drop birthplace;
Query OK, 0 rows affected (0.155 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc patient;
+--------------+----------------------+------+-----+---------+-------+
| Field        | Type                 | Null | Key | Default | Extra |
+--------------+----------------------+------+-----+---------+-------+
| patient_code | int(10)              | NO   | PRI | NULL    |       |
| patient_name | varchar(25)          | YES  |     | NULL    |       |
| patient_addr | varchar(30)          | YES  |     | NULL    |       |
| gender       | enum('Male','Female') | YES  |     | NULL    |       |
+--------------+----------------------+------+-----+---------+-------+
```

3.6. Default Constraints

The DEFAULT constraint is used to set a default value for a column. The default value will be added to all new records if no other value is specified. To create a DEFAULT constraint on the column when the table is already created, use the following SQL:

```
ALTER TABLE table_name ALTER column_name SET DEFAULT 'value';
```

Adding the Default value for the gender is Male.

```
MariaDB [hospital]> Alter table patient
    -> alter gender set default 'Male';
Query OK, 0 rows affected (0.132 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc patient;
+--------------+----------------------+------+-----+---------+-------+
| Field        | Type                 | Null | Key | Default | Extra |
+--------------+----------------------+------+-----+---------+-------+
| patient_code | int(10)              | NO   | PRI | NULL    |       |
| patient_name | varchar(25)          | YES  |     | NULL    |       |
| patient_addr | varchar(30)          | YES  |     | NULL    |       |
| gender       | enum('Male','Female') | YES  |     | Male    |       |
+--------------+----------------------+------+-----+---------+-------+
```

3.7. Adding Primary and Foreign Key (on ALTER TABLE)

a. Adding Primary Key (on ALTER TABLE)

Adding the primary key to a column in a table that has already been created. The syntax is as follows:

```
ALTER TABLE table_name ADD PRIMARY KEY (column_name);
```

For example, let's use prescription table, with the structure shown below.

```
MariaDB [hospital]> desc prescription;
+-----------+------------+------+-----+---------+-------+
| Field     | Type       | Null | Key | Default | Extra |
+-----------+------------+------+-----+---------+-------+
| pres_code | char(5)    | NO   | PRI | NULL    |       |
| pres_date | date       | NO   |     | NULL    |       |
| med_rules | varchar(5) | NO   |     | NULL    |       |
+-----------+------------+------+-----+---------+-------+
```

Next, delete the primary key.

```
MariaDB [hospital]> alter table prescription
    -> drop primary key;
Query OK, 0 rows affected (1.013 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc prescription;
+-----------+------------+------+-----+---------+-------+
| Field     | Type       | Null | Key | Default | Extra |
+-----------+------------+------+-----+---------+-------+
| pres_code | char(5)    | NO   |     | NULL    |       |
| pres_date | date       | NO   |     | NULL    |       |
| med_rules | varchar(5) | NO   |     | NULL    |       |
+-----------+------------+------+-----+---------+-------+
```

And then, we create the primary key using ALTER command.

```
MariaDB [hospital]> alter table prescription
    -> add primary key (pres_code);
Query OK, 0 rows affected (0.723 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc prescription;
+-----------+------------+------+-----+---------+-------+
| Field     | Type       | Null | Key | Default | Extra |
+-----------+------------+------+-----+---------+-------+
| pres_code | char(5)    | NO   | PRI | NULL    |       |
| pres_date | date       | NO   |     | NULL    |       |
| med_rules | varchar(5) | NO   |     | NULL    |       |
+-----------+------------+------+-----+---------+-------+
```

b. Adding Foreign Key (on ALTER TABLE)

The following is a method if we want to add a foreign key to a column in a table.

The syntax is as follows:

ALTER TABLE table_name ADD FOREIGN KEY (column name) REFERENCES table_name reference (PK reference column name) ON DELETE CASCADE ON UPDATE CASCADE;

CASCADE: It is used when we delete or update any row from the parent table; the values of the matching rows in the child table will be deleted or updated automatically. (*Optional*)

For example, add the med_code column and the checkup_code column in the prescription table.

```
MariaDB [hospital]> alter table prescription
    -> add med_code char(5) not null
    -> after pres_code,
    -> add checkup_code char(5) not null
    -> after med_code;
Query OK, 0 rows affected (0.255 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [hospital]> desc prescription;
+--------------+------------+------+-----+---------+-------+
| Field        | Type       | Null | Key | Default | Extra |
+--------------+------------+------+-----+---------+-------+
| pres_code    | char(5)    | NO   | PRI | NULL    |       |
| med_code     | char(5)    | NO   |     | NULL    |       |
| checkup_code | char(5)    | NO   |     | NULL    |       |
| pres_date    | date       | NO   |     | NULL    |       |
| med_usage    | varchar(5) | NO   |     | NULL    |       |
| med_amount   | int(3)     | YES  |     | NULL    |       |
+--------------+------------+------+-----+---------+-------+
```

Add the foreign key for med_code (medicine table) and checkup_code (examining table)

```
MariaDB [hospital]> alter table prescription
    -> add foreign key (med_code) references medicine(med_code),
    -> add foreign key (checkup_code) references examining (checkup_code);
Query OK, 0 rows affected (1.355 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc prescription;
+--------------+------------+------+-----+---------+-------+
| Field        | Type       | Null | Key | Default | Extra |
+--------------+------------+------+-----+---------+-------+
| pres_code    | char(5)    | NO   | PRI | NULL    |       |
| med_code     | char(5)    | NO   | MUL | NULL    |       |
| checkup_code | char(5)    | NO   | MUL | NULL    |       |
| pres_date    | date       | NO   |     | NULL    |       |
| med_usage    | varchar(5) | NO   |     | NULL    |       |
| med_amount   | int(3)     | YES  |     | NULL    |       |
+--------------+------------+------+-----+---------+-------+
```

c. Delete Foreign Key

For example, let's delete the med_code column.

```
MariaDB [hospital]> desc prescription;
+--------------+-----------+------+-----+---------+-------+
| Field        | Type      | Null | Key | Default | Extra |
+--------------+-----------+------+-----+---------+-------+
| pres_code    | char(5)   | NO   | PRI | NULL    |       |
| med_code     | char(5)   | NO   | MUL | NULL    |       |
| checkup_code | char(5)   | NO   | MUL | NULL    |       |
| pres_date    | date      | NO   |     | NULL    |       |
| med_usage    | varchar(5)| NO   |     | NULL    |       |
| med_amount   | int(3)    | YES  |     | NULL    |       |
+--------------+-----------+------+-----+---------+-------+
```

When the "ALTER TABLE prescription DROP FOREIGN KEY med_code" command is not working, then you need to find out the constraint's name, using "SHOW CREATE TABLE prescription" command. Then delete the constraint's name using "ALTER TABLE prescription DROP FOREIGN KEY prescription_IBfk_2;

```
prescription | CREATE TABLE `prescription` (
`pres_code` char(5) NOT NULL,
`med_code` char(5) NOT NULL,
`checkup_code` char(5) NOT NULL,
`pres_date` date NOT NULL,
`med_usage` varchar(5) NOT NULL,
`med_amount` int(3) DEFAULT NULL,
PRIMARY KEY (`pres_code`),
KEY `med_code` (`med_code`),
KEY `checkup_code` (`checkup_code`),
CONSTRAINT `prescription_ibfk_2` FOREIGN KEY
```

```
MariaDB [hospital]> desc prescription;
+--------------+-----------+------+-----+---------+-------+
| Field        | Type      | Null | Key | Default | Extra |
+--------------+-----------+------+-----+---------+-------+
| pres_code    | char(5)   | NO   | PRI | NULL    |       |
| med_code     | char(5)   | NO   | MUL | NULL    |       |
| checkup_code | char(5)   | NO   | MUL | NULL    |       |
| pres_date    | date      | NO   |     | NULL    |       |
| med_usage    | varchar(5)| NO   |     | NULL    |       |
| med_amount   | int(3)    | YES  |     | NULL    |       |
+--------------+-----------+------+-----+---------+-------+
```

```
MariaDB [hospital]> alter table prescription
    -> drop med_code;
Query OK, 0 rows affected (0.191 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [hospital]> desc prescription;
+--------------+------------+------+-----+---------+-------+
| Field        | Type       | Null | Key | Default | Extra |
+--------------+------------+------+-----+---------+-------+
| pres_code    | char(5)    | NO   | PRI | NULL    |       |
| checkup_code | char(5)    | NO   | MUL | NULL    |       |
| pres_date    | date       | NO   |     | NULL    |       |
| med_usage    | varchar(5) | NO   |     | NULL    |       |
| med_amount   | int(3)     | YES  |     | NULL    |       |
+--------------+------------+------+-----+---------+-------+
```

TASK 3:

1. Design a database (ACADEMIC) consisting of 4 tables:
    a. Students
        - NPM char (10) primary key
        - students_name varchar (30) not null
        - date of birth date not null
        - gender enum (Male, Female) default Male
        - dept varchar (20) not null
        - students_address varchar (30) not null

    b. Subjects
        - subject_code char (5) primary key
        - subject_name varchar (30) not null
        - semester int (1) not null
        - sks int (1) not null
        - dept varchar (20) not null

    c. Lecturer
        - lecturer_code char (10) primary key
        - lecturer_name varchar (30) not null
        - date of birth date not null
        - gender varchar (2) not null

- lecturer_address varchar (30) not null

d. Final Score
    - NPM char (10) foreign key
    - lecturer_code char (10) foreign key
    - task_score int (3)
    - UTS_score int (3)
    - UAS_score int (3)

2. Show table structure: (a) Subjects, (b) Lecturer, (c) Students, (d) Final Score
3. Change the student_address field in the student table to student_add
4. Change the data type for the gender_field field in the lecturer table to enum(Male, Female)
5. Delete the lecturer_code from the Final Score table
6. Add a city field varchar (20) to the student table
7. Add a foreign key (subject_code) in the Final Score table related to the Subject table.
8. Delete the UAS_score field from the Final Score table
9. Give the default value Male in the gender field in the lecturer table
10. Add a foreign key (lecturer_code) in the Final Score table related to the Lecturer table.
11. Show all the table structure