**Q.1 What is SQLite Database? Explain CRUD operations.**

An SQL database is SQLite. Tables are used in SQL databases to hold data. Data is stored in tables, which are made up of rows and columns. An Android database is a type of permanent data storage used by apps for Android-powered devices. In order for the app to remain accessible even if the device loses connectivity, it frequently comprises of on-device, local storage. The most popular database technology used with Android applications is SQLite, an open-source relational database, as a result of its inclusion in the Android Software Development Kit (SDK). The most popular combination of SQLite with Room, a framework for controlling the lifespan of objects, is for Android

**CRUD Operations**

The four fundamental actions that each software programme should be able to carry out are referred to as CRUD:

1. Create
2. Read
3. Update
4. Delete

Users must be able to create data, access it in the UI by reading it, update or edit it, and delete it in these apps.

**Create a SQLiteOpenHelper class**

**DBHandler** class is our database handler class which extends **SQLiteOpenHelper**class to manage database operations. First time on constructor (DBHandler) call we will create our database. In onCreate() method we will create any table e.g. Student.

**Creating db in sqlite**

public DBHandler( Context context, String name, SQLiteDatabase.CursorFactory factory, int version)

{  
 super(context, "TEST.db", factory, version);  
}

**Create Table in SQLite**

**E.g. Students**

@Override  
public void onCreate(SQLiteDatabase db) {  
 db.execSQL("CREATE TABLE STUDENTS(ID INTEGER PRIMARY KEY AUTOINCREMENT, FIRSTNAME TEXT UNIQUE, LASTNAME TEXT);");  
  
  
}

**Insert New Record**

public void InsertStudent(String SName, String SAddress, String ContactNo, String Emailid) {

        // on below line we are creating a variable for

        // our sqlite database and calling writable method

        // as we are writing data in our database.

        SQLiteDatabase db = this.getWritableDatabase();

        // on below line we are creating a

        // variable for content values.

        ContentValues values = new ContentValues();

        // on below line we are passing all values

        // along with its key and value pair.

        values.put(NAME\_COL, SName);

        values.put(ADDRESS\_COL, SAddress);

        values.put(CONTACTNUMBER\_COL, ContactNo);

        values.put(EMAIL\_COL, Emailid);

        // after adding all values we are passing

        // content values to our table.

        db.insert(TABLE\_NAME, null, values);

        // at last we are closing our

        // database after adding database.

        db.close();

    }

**Content Values** creates an empty set of values using the given initial size. We’ll discuss the other instance values when we jump into the coding part.

**When you need to modify a subset of your database values, use the**[**update()**](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase#update(java.lang.String,%20android.content.ContentValues,%20java.lang.String,%20java.lang.String[]))**method. Updating the table combines the [ContentValues](https://developer.android.com/reference/android/content/ContentValues) syntax of**[**insert()**](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase#insert(java.lang.String,%20java.lang.String,%20android.content.ContentValues))**with the WHERE syntax of**[**delete()**](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase#delete(java.lang.String,%20java.lang.String,%20java.lang.String[]))**.**

**Updating Table**

public int update(long\_id, String Sname, String SAddress) {

ContentValues contentValues = new ContentValues();

contentValues.put(DatabaseHelper.Sudent, name);

contentValues.put(DatabaseHelper.SName, SAddress);

int i = database.update(DatabaseHelper.TABLE\_NAME, contentValues, DatabaseHelper.\_ID + " = " + \_id, null);

return i;

}

**Delete Record**

 Just need to pass the id of the record to be deleted as shown below.

public void delete(long \_id) {

database.delete(DatabaseHelper.TABLE\_NAME, DatabaseHelper.\_ID + "=" + \_id, null);

}

**Read Records**

A Cursor represents the entire result set of the query. Once the query is fetched a call to cursor.moveToFirst() is made. Calling moveToFirst() does two things:

* It allows us to test whether the query returned an empty set (by testing the return value)
* It moves the cursor to the first result (when the set is not empty)

The following code is used to fetch all records:

public Cursor fetch() {

String[] columns = new String[] { DatabaseHelper.\_ID, DatabaseHelper.Sname, DatabaseHelper.SAddress };

Cursor cursor = database.query(DatabaseHelper.TABLE\_NAME, columns, null, null, null, null, null);

if (cursor != null) {

cursor.moveToFirst();

}

return cursor;

**}**

Another way to use a Cursor is to wrap it in a CursorAdapter. Just as ArrayAdapter adapts arrays, CursorAdapter adapts Cursor objects, making their data available to an AdapterView like a ListView. Let’s jump to our project that uses SQLite to store some meaningful data.