Android SQLite Fundamentals

Muhammad Jamil
Senior Software Developer

Course Overview

- Introduction to SQLite
- Using SQLiteOpenHelper Class and Contract Class
- Using Cursor to read or write data into the SQLite Database
- Using SQLite methods like Insert, Update, Delete etc. to work with SQLite Database
- Why should we use SQLite over other storage options
- At the end of SQLite Week, you'll know all the fundamentals of SQLite Database

Familiarizing with SQLite

- SQLite vs Shared Preferences vs File System
- ACID properties
- SQLite vs SQL

Prerequisites

- Basics of Android
- RecyclerView
- How to write SQL Queries
- Database Schema
- Tables , Fields, Records, DB Transactions etc.

Features of SQLite

- Open Source Database and support relational database features
- Very small in size
- Android has built in SQLite implementation
  - *No additional dependencies*
  - *android.database.sqlite (package for required APIs)*
- Use SQLiteOpenHelper and SQLite database classes

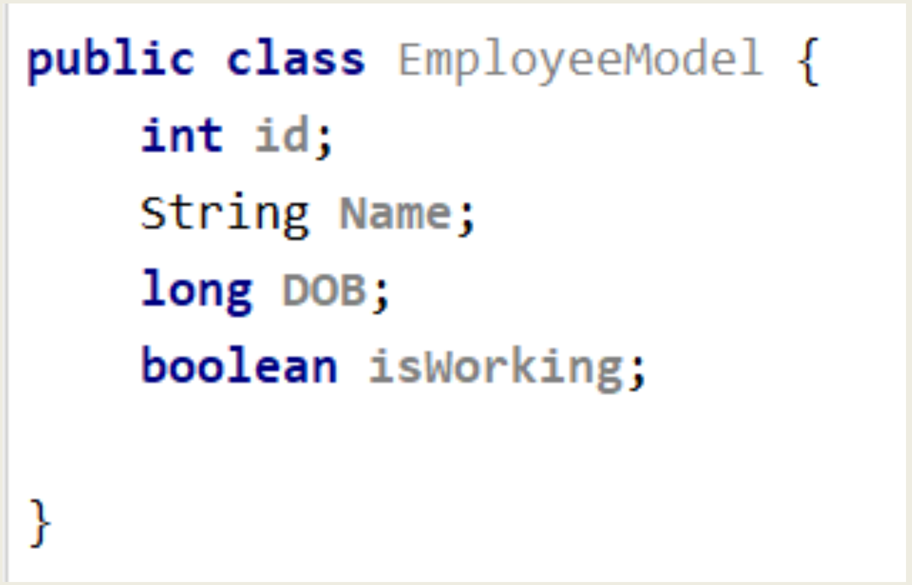
Available Storage Classes

Storage Classes

- *integer*
- *text*
- *real (floating values)*
- *null*
- *blob (for images and files)*

Model Class

- Integer
- Text
- Integer
- Integer (0,1)



```
public class EmployeeModel {
    int id;
    String Name;
    long DOB;
    boolean isWorking;

}
```

When SQLite?

Shared Preferences	File System	SQLite
• Key-value form of storage in xml files • For small and simple data like username, password • Store only primitive type of data • Key required to retrieve data	• Storing files is easy and efficient • Complex data like audio, video etc. • No ACID	• For storing complex and structured data like contact information • Easy to retrieve data using SQL Queries • ACID properties

Understanding ACID Properties

Atomicity

- *Execute all transaction or nothing*

Consistency

- *Maintain the consistency of the database before and after transaction*

Isolation

- *Modification in Midway of transaction is not visible to anyone*

Durability

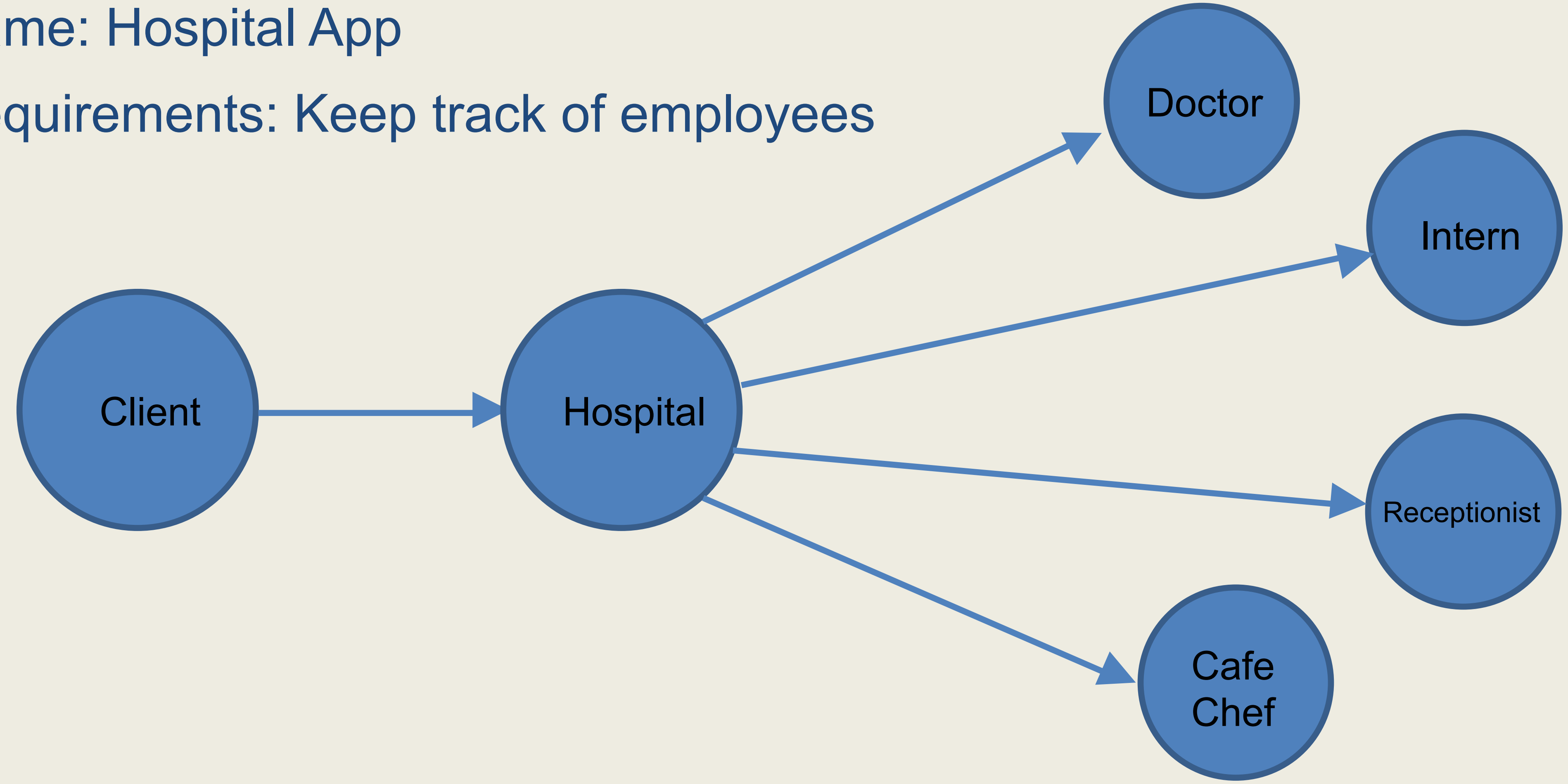
- *Once done, it is persistent, no data loss after rebooting*

SQLite and SQL

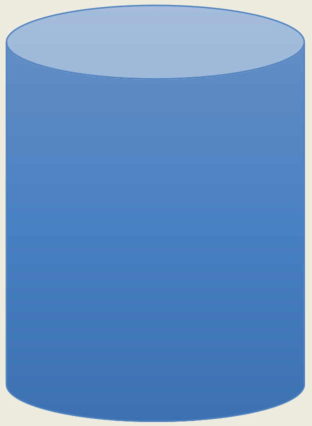
- SQL is a query language
- SQLite is a relational database management system, which uses SQL
- It is a lightweight version of MySQL
- We use SQLite because
  - *It stores structured data*
  - *Small in size*
  - *Persistent local storage*
  - *Built-in Android (No other dependency is required)*

A Simple Case Study

- Name: Hospital App
- Requirements: Keep track of employees



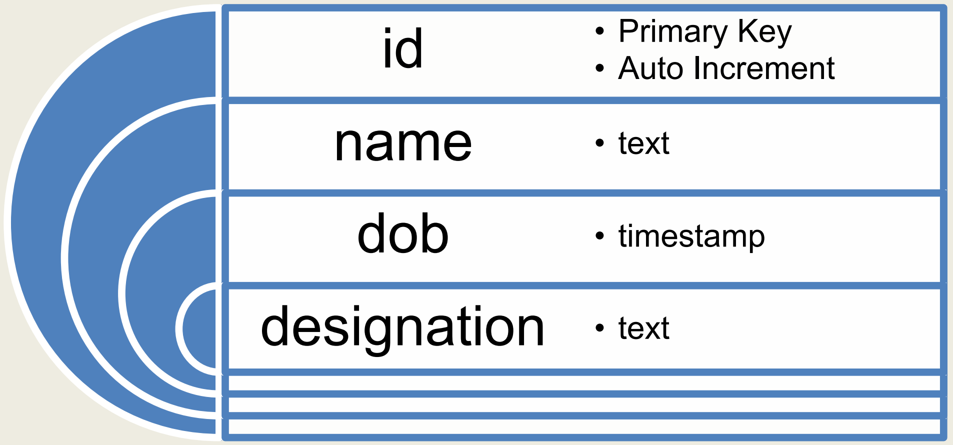
Hospital App



Hospital_db

id	name	dob	designation
1	Abdullah	601430400	Physician
2	Hamza	601330400	Intern
3	Usman	601230400	Cafeteria Chef
4	Z. Huma	601130400	Receptionist

tbl_employees



id	• Primary Key • Auto Increment
name	• text
dob	• timestamp
designation	• text

Classes for SQLite

Contract Class

Model Data Class

DB Helper Class

Activity Class

Database Schema

- Declaration of Database Design
  - *No. of tables in DB*
  - *No. of columns (fields) in a particular table*
  - *All the details regarding DB*
- Contract Class
  - *Contains database schema*
  - *Constants for table name and fields names*

SQLite

- Schema Class
  - *Defining Constants for schema*
- Database SQLiteOpenHelper
  - *Callback methods*
- Create Table for Hospital App
- SQL Open Helper Class
  - *SQLiteOpenHelper*
  - *Database Access Methods like onCreate(), onUpgrade() etc.*
  - *.getReadableDatabase , .getWriteable Database*
  - *Helps creating and versioning Database*

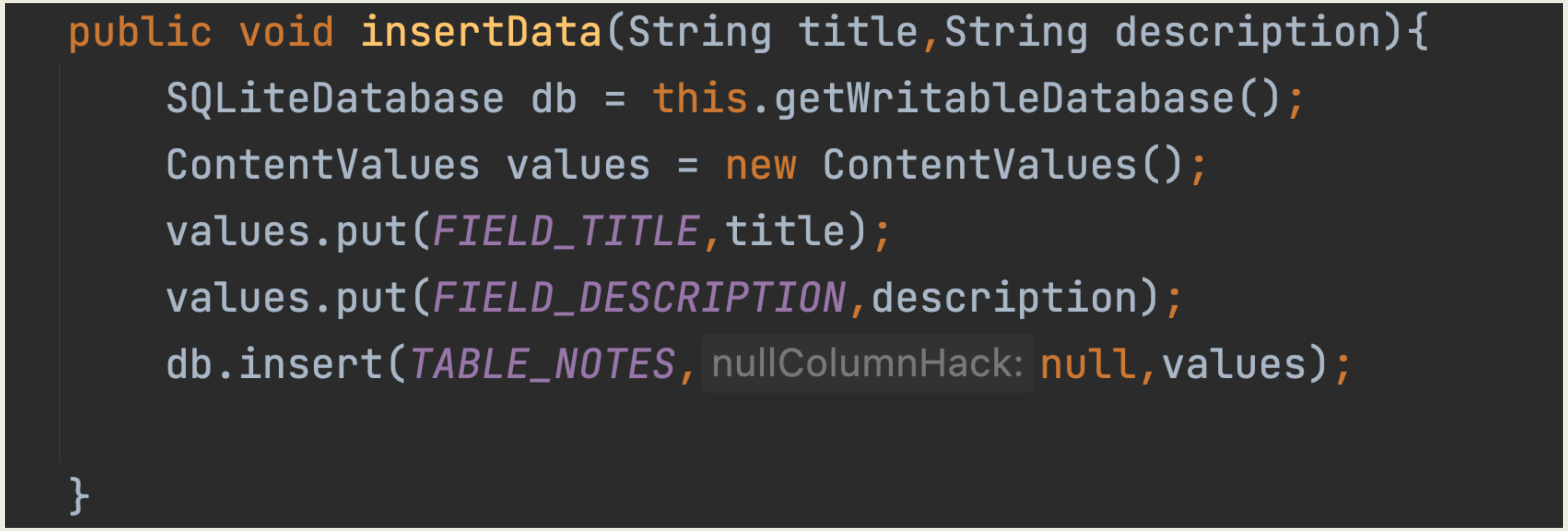
Creating Model Class

- To map the table fields as class members

Insert Data (cont.)

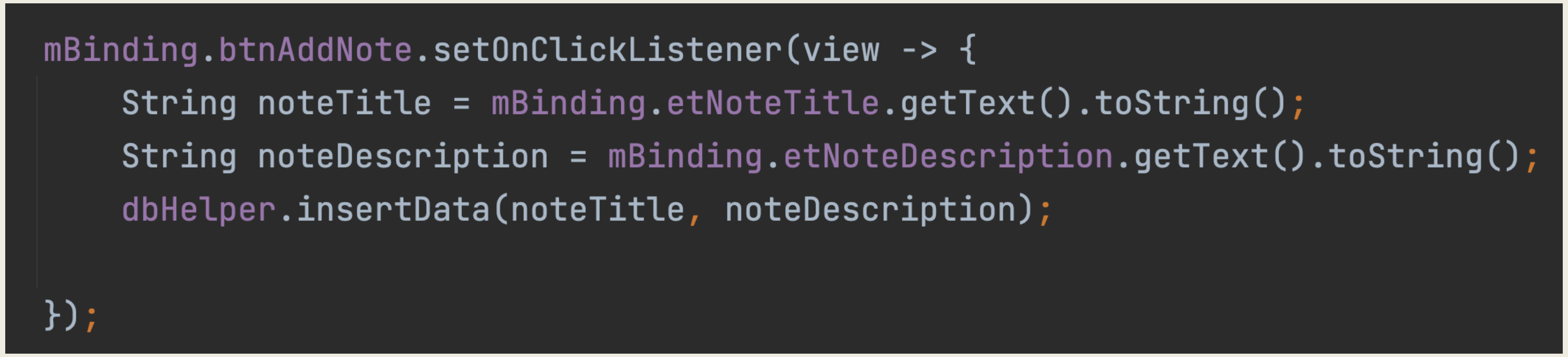
- Add a new row to the table
- We need to access the geWriteableDatabase() method to insert records in the database.
- Use ContentValues Class
  - *Contains a list of Columns Names and values*
  - *Use put() method to add values to an instance of ContentValues*
  - *Use insert(Table Name, values) method to insert values in database by passing two parameters i.e. table name and instance of ContentValues*

Insert Data



```java
public void insertData(String title,String description){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(FIELD_TITLE,title);
    values.put(FIELD_DESCRIPTION,description);
    db.insert(TABLE_NOTES, nullColumnHack: null,values);

}
```



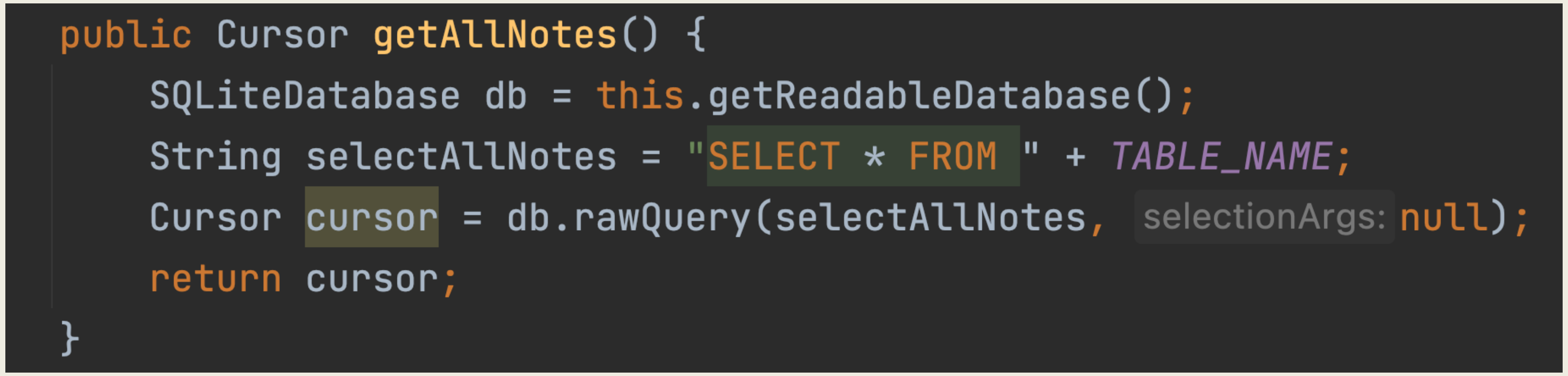
```java
mBinding.btnAddNote.setOnClickListener(view -> {
    String noteTitle = mBinding.etNoteTitle.getText().toString();
    String noteDescription = mBinding.etNoteDescription.getText().toString();
    dbHelper.insertData(noteTitle, noteDescription);

});
```

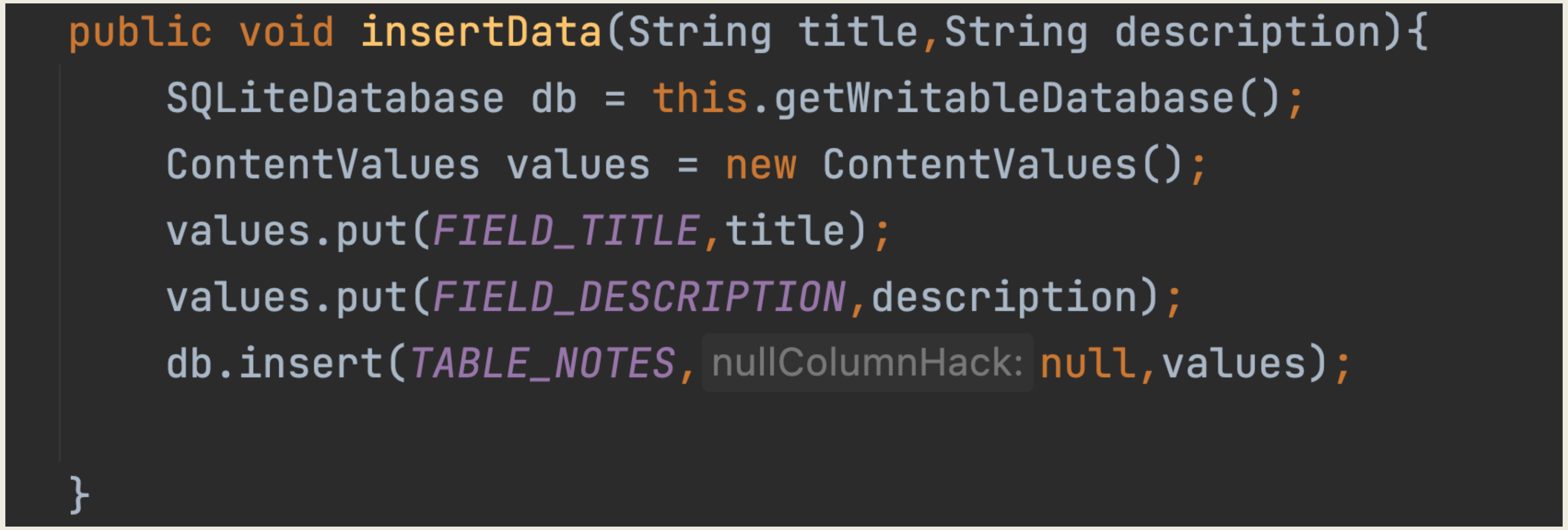
Show Data

- Show data from a table using the **SELECT** query
- We need to access the getReadableDatabase() method to fetch records from the database.
- The SQLiteDatabase class always presents the results as a Cursor in a table format that resembles that of a SQL database.
- A cursor is a pointer to one row of that structured data.
- The Cursor class provides methods for moving the cursor through the data structure and methods to get the data from the fields in each row.

Update Data



```
public Cursor getAllNotes() {
    SQLiteDatabase db = this.getReadableDatabase();
    String selectAllNotes = "SELECT * FROM " + TABLE_NAME;
    Cursor cursor = db.rawQuery(selectAllNotes,  selectionArgs: null);
    return cursor;
}
```

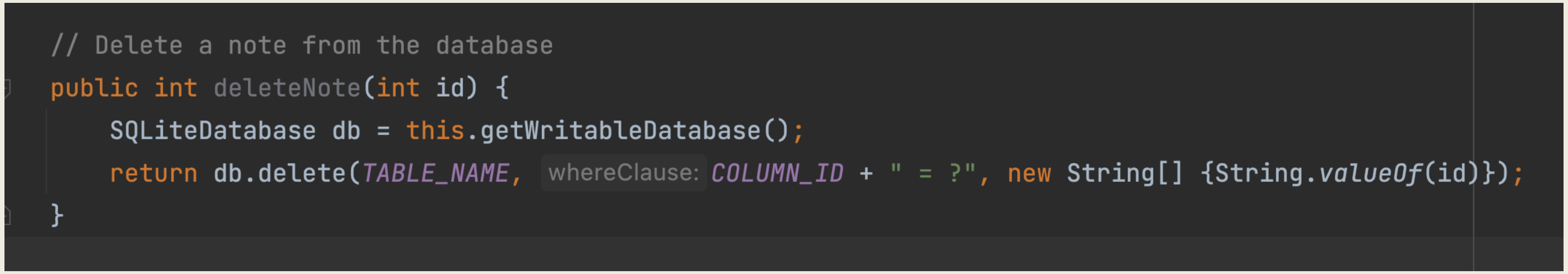


```
public void insertData(String title,String description){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(FIELD_TITLE,title);
    values.put(FIELD_DESCRIPTION,description);
    db.insert(TABLE_NOTES, nullColumnHack: null,values);

}
```

Delete Data

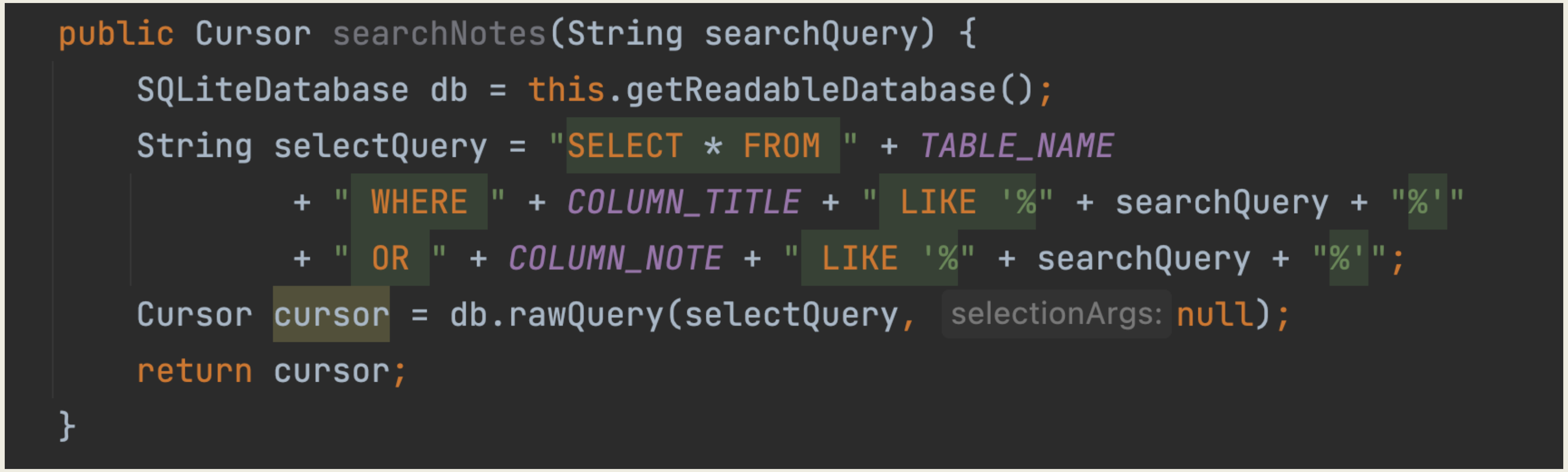
- To delete a record from the database, we again, need to access getWritableDatabase() method.
- We have to use the SQL DELETE command to perform the delete operation.



```
// Delete a note from the database
public int deleteNote(int id) {
    SQLiteDatabase db = this.getWritableDatabase();
    return db.delete(TABLE_NAME,  whereClause: COLUMN_ID + " = ?", new String[] {String.valueOf(id)});
}
```

Search Data

- To find a record from the database, we need to access the getReadableDatabase() method.
- We have to use the SQL SELECT command with the LIKE operator to perform the search operation.



```
public Cursor searchNotes(String searchQuery) {
    SQLiteDatabase db = this.getReadableDatabase();
    String selectQuery = "SELECT * FROM " + TABLE_NAME
            + " WHERE " + COLUMN_TITLE + " LIKE '%" + searchQuery + "%'"
            + " OR " + COLUMN_NOTE + " LIKE '%" + searchQuery + "%'";
    Cursor cursor = db.rawQuery(selectQuery, selectionArgs: null);
    return cursor;
}
```

Summary

- SQLite for persistence (data saving) on mobile
- Self-contained, decentralized (serverless), zero configuration and a transactional SQL database engine
- Small footprint
- Native as well as cross-platform support
- Integrate with other APIs
- Flexibility is unmatched
- Realm increases apk size

Thank You!

For Queries: jamil.57@Hotmail.com