ITW202: Mobile Application

Unit IV: Developing for Android

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology
Royal University of Bhutan

May 30, 2021

SQLite Database

Features of SQLite

Open source database and supports standard relational database features

Very small in size

Android has built-in SQLite implementation

- No additional dependencies
- android.database.sqlite

Use SQLiteOpenHelper and SQLite database classes

Available storage classes

Storage Classes:

- integer
- text
- real
- null
- blob

When SQLite?

Shared Preference

- Key-value form of storage in XML files
- For small and simple data like username, password
- Stores only primitive type data
- Key required to retrieve the data

File System

- Storing files is easy and efficient
- No ACID

SQLite

- For complex and structured data like contact information
- Easy to retrieve data using queries
- ACID properties

ACID

Mobile Application

A — **Atomicity**
Execute all or nothing

C — **Consistency**
Maintain the consistency of the database before and after transaction

I — **Isolation**
Modification in midway of transaction is not visible to anyone

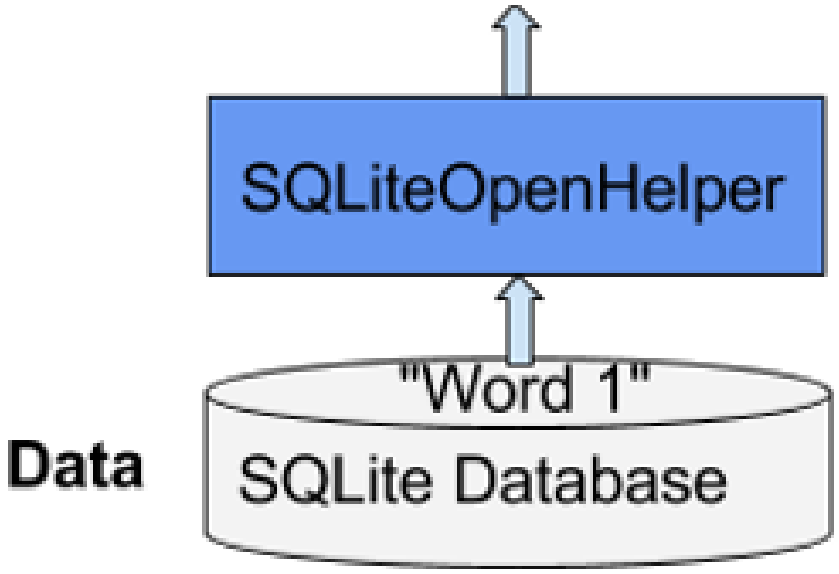
D — **Durability**
Once done, it is persistent

SQL and SQLite

SQL is a query language

SQLite is a relational database management system, which uses SQL

Components of SQLite database



SQLiteOpenHelper

- SQLite database represented as an SQLiteDatabase object and all interactions with database are done through SQLiteOpenHelper
- Executes your requests
- Manages your database
- Separates data and interaction from app

Cursor

- Provides read-write access to result set
- Provides row-by-row access
- Initial cursor position is -1
- To read the data, use cursor functions to move it to desired row
- SQLiteDatabase always presents results as Cursor

Cursor Functions

- moveToFirst() - Move the cursor to the first row
- moveToLast() - Move the cursor to the last row
- moveToNext() - Move the cursor to the next row
- moveToPosition(int position) - Move the cursor to an absolute position
- moveToPrevious() - Move the cursor to the previous row

Accessing Cursor Results

- Fetch column positions - getColumnIndex()
- Fetch column data - getString() or getLong()
- Close cursor after use to release resources - close()

Processing Cursors

```
// Store results of query in a cursor
Cursor cursor = db.rawQuery(...);
try {
    while (cursor.moveToNext()) {
        // Do something with data
    }
} finally {
    cursor.close();
}
```

Content Values

- Represents one table row
- Stores data as key-value pairs
- Key is the name of the column
- Value is the value for the field
- Used to pass row data between methods

Content Values

```
ContentValues values = new ContentValues();

// Inserts one row.
// Use a loop to insert multiple rows.
values.put(KEY_WORD, "Android");
values.put(KEY_DEFINITION, "Mobile operating system.");

db.insert(WORD_LIST_TABLE, null, values);
```

Implementing SQLite

You always need to ...

1. Create data model
2. Subclass SQLiteOpenHelper
   - Create constants for tables
   - onCreate()—create SQLiteDatabase with tables
   - onUpgrade(), and optional methods
   - Implement query(), insert(), delete(), update(), count()
3. In MainActivity, create instance of SQLiteOpenHelper
4. Call methods of SQLiteOpenHelper to work with database

Data model

- Class with getters and setters
- One "item" of data (for database, one record or one row)

```
public class WordItem {
    private int mId;
    private String mWord;
    private String mDefinition;
   ...
}
```

Subclass SQLiteOpenHelper

```
public class WordListOpenHelper extends SQLiteOpenHelper {

    public WordListOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
        Log.d(TAG, "Construct WordListOpenHelper");
    }
}
```

Declare constants for tables

```
private static final int DATABASE_VERSION = 1;
// Has to be 1 first time or app will crash.
private static final String DATABASE_NAME = "wordlist";
private static final String WORD_LIST_TABLE = "word_entries";

// Column names...
public static final String KEY_ID = "_id";
public static final String KEY_WORD = "word";

// ... and a string array of columns.
private static final String[] COLUMNS = {KEY_ID, KEY_WORD};
```

Define query for creating database

- *You need a query to create the database*
- *Customarily defined as a string constant*

```
private static final String WORD_LIST_TABLE_CREATE =
            "CREATE TABLE " + WORD_LIST_TABLE + " (" +
                    KEY_ID + " INTEGER PRIMARY KEY, " +
                    // will auto-increment if no value passed
                    KEY_WORD + " TEXT );";
```

onCreate()

```
@Override
public void onCreate(SQLiteDatabase db) { // Creates new database
  // Create the tables
  db.execSQL(WORD_LIST_TABLE_CREATE);
  // Add initial data
  ...
}
```

onUpgrade()

```java
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
    onCreate(db);
}
```

Database Operations

Database operations

- query()
- insert()
- update()
- delete()

Executing queries

- implement query() method in open helper class
- query() can take and return any data type that UI needs
- Only support queries that your app needs
- Use database convenience methods for insert, delete, and update

Database methods for executing queries

- SQLiteDatabase.rawQuery(): Use when data is under your control and supplied only by your app
- SQLiteDatabase.query(): Use for all other queries

SQLiteDatabase.rawQuery() format

```
rawQuery(String sql, String[] selectionArgs)
```

- ***First parameter is SQLite query string***
- ***Second parameter contains the arguments***

SQLiteDatabase.Query() format

```
query(
    TABLE_NAME,         // Table to query
    projection,         // Array of columns to return
    selection,          // Columns for the WHERE clause
    selectionArgs,      // Values for the WHERE clause
    null,               // Don't group the rows
    null,               // Don't filter by row groups
    sortOrder           // Sort order
)
```

Insert, Delete, Update, Count

insert() format

```
long insert(String table, String nullColumnHack,
            ContentValues values)
```

- ***First argument is the table name.***
- ***Second argument is a*** `String nullColumnHack`.
  - ***Workaround that allows you to insert empty rows***
  - ***Use*** `null`
- ***Third argument must be a ContentValues with values for the row***

delete() format

```
int delete (String table,
            String whereClause, String[] whereArgs)
```

- First argument is table name
- Second argument is WHERE clause
- Third argument are arguments to WHERE clause

update() format

```
int update(String table, ContentValues values,
        String whereClause, String[] whereArgs)
```

- ***First argument is table name***
- ***Second argument must be ContentValues with new values for the row***
- ***Third argument is WHERE clause***
- ***Fourth argument are the arguments to the WHERE clause***

Always!

- Always put database operations in try-catch blocks
- Always validate user input and SQL queries

Instantiate OpenHelper

Create an instance of your OpenHelper

- In MainActivity onCreate()
  mDB = new WordListOpenHelper(this);

THANK YOU