ANALISING ANDROID SQLITE DATABASE

URMANSCHI Mihail, ZUBCOV Olga
Technical University of Moldova

Abstract : *This article is intended for those who were not familiar with SQL programming in android. Here it is shown how to use SQL android and its elements for a comparison to other systems, and their differences.*

Key words:: *SQLite, SQLiteOpenHelper, SQLiteDatabase, Cursor, query.*

1. Introduction

SQLite is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. The database requires limited memory at runtime (approx. 250 KByte) which makes it a good candidate from being embedded into other runtimes.

SQLite supports the data types TEXT (similar to String in Java), INTEGER (similar to long in Java) and REAL (similar to double in Java). All other types must be converted into one of these fields before getting saved in the database. SQLite itself does not validate if the types written to the columns are actually of the defined type, e.g. you can write an integer into a string column and vice versa.

2. SQLite in Android

SQLite is embedded into every Android device. Using an SQLite database in Android does not require a setup procedure or administration of the database. You only have to define the SQL statements for creating and updating the database. Afterwards the database is automatically managed for you by the Android platform.Access to an SQLite database involves accessing the file system. This can be slow. Therefore it is recommended to perform database operations asynchronously.If your application creates a database, this database is by default saved in the directory DATA/data/APP_NAME/databases/ FILENAME.The parts of the above directory are constructed based on the following rules. DATA is the path which theEnvironment .getDataDirectory() method returns. APP_NAME is your application name. FILENAME is the name you specify in your application code for the database.

3. SQLite architecture. Creating and updating database with SQLiteOpenHelper

To create and upgrade a database in your Android application you create a subclass of the SQLiteOpenHelper class. In the constructor of your subclass you call the super() method of SQLiteOpenHelper, specifying the database name and the current database version.

In this class you need to override the following methods to create and update your database.

onCreate() - is called by the framework, if the database is accessed but not yet created.

onUpgrade() - called, if the database version is increased in your application code. This method allows you to update an existing database schema or to drop the existing database and recreate it via the onCreate() method.

The SQLiteOpenHelper class provides the getReadableDatabase() and getWriteableDatabase() methods to get access to an SQLiteDatabase object; either in read or write mode.The database tables should use the identifier _id for the primary key. Several Android functions rely on this standard.

4. SQLiteDatabase

SQLiteDatabase is the base class for working with a SQLite database in Android and provides methods to open, query, update and close the database. More specifically SQLiteDatabase provides the insert(), update() and delete() methods. In addition it provides the execSQL() method, which allows to execute an SQL statement directly.

The object ContentValues allows to define key/values. The key represents the table column identifier and the value represents the content for the table record in this column. ContentValues can be used for inserts and updates of database entries. Queries can be created via the rawQuery() and query() methods or via the SQLiteQuery Builder class .rawQuery() directly accepts an SQL select statement as input. query() provides a structured interface for specifying the SQL query. SQLiteQueryBuilder is a convenience class that helps to build SQL queries.

5. Cursor

A query returns a Cursor object. A Cursor represents the result of a query and basically points to one row of the query result. This way Android can buffer the query results efficiently; as it does not have to load all data into memory.To get the number of elements of the resulting query use the getCount() method.

To move between individual data rows, you can use the moveToFirst() and moveToNext() methods. The isAfterLast()method allows to check if the end of the query result has been reached.

Cursor provides typed get*() methods, e.g. getLong(columnIndex),getString(columnIndex) to access the column data for the current position of the result. The "columnIndex" is the number of the column you are accessing.

Cursor also provides the getColumnIndex OrThrow(String) method which allows to get the column index for a column name of the table.A Cursor needs to be closed with the close() method call.

6. Conclusion

Most of us are familiar with at least some of the persistence features Core Data offers us out of the box. Unfortunately, many of those things aren't automatic on the Android platform. For instance, Core Data abstracts away most of the SQL syntax and database normalization concerns facing database engineers every day. Since Android only provides a thin client to SQLite, you'll still need to write SQL and ensure your database tables are appropriately normalized.

Bibliography

1. Lars Vogel. Android SQLite database and content provider. [Electronic resource].- Access form : http://www.vogella.com/tutorials/AndroidSQLite/article.html

2. Ravi Tamada. Android SQLite Database Tutorial. [Electronic resource].- Access form :http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/

3. Saving Data in SQL Databases. [Electronic resource].- Access form :http://developer.android.com/training/basics/data-storage/databases.html

4. Android - SQLite database Tutorial. [Electronic resource].- Access form :http://www.tutorialspoint.com/android/android_sqlite_database.htm