Android Developer Fundamentals V2

Storing Data with Room

Lesson 10

10.0 SQLite Primer

Contents

- SQLite Database
- Queries

This is only a refresher

This course assumes that you are familiar with

- Databases in general
- SQL databases in particular
- SQL query language

This chapter is a refresher and quick reference

SQLite Database

SQL Databases

- Store data in tables of rows and columns (spreadsheet…)
- Field = intersection of a row and column
- Fields contain data, references to other fields, or references to other tables
- Rows are identified by unique IDs
- Column names are unique per table

Tables

WORD_LIST_TABLE		
_id	**word**	**definition**
1	"alpha"	"first letter"
2	"beta"	"second letter"
3	"alpha"	"particle"

SQLite software library

Implements SQL database engine that is

- self-contained (requires no other components)
- serverless (requires no server backend)
- zero-configuration (does not need to be configured for your application)
- transactional (changes within a single transaction in SQLite either occur completely or not at all)

What is a transaction?

A transaction is a sequence of operations performed as a single logical unit of work.

A logical unit of work must have four properties

- atomicity
- consistency
- isolation
- durability

All or nothing

All changes within a single transaction in SQLite either occur completely or not at all, even if the act of writing the change out to the disk is interrupted by

- program crash
- operating system crash
- power failure.

ACID

- **Atomicity**—All or no modifications are performed
- **Consistency**—When transaction has completed, all data is in a consistent state
- **Isolation**—Modifications made by concurrent transactions must be isolated from the modifications made by any other concurrent transactions
- **Durability**—After a transaction has completed, its effects are permanently in place in the system

Queries

SQL basic operations

- Insert rows
- Delete rows
- Update values in rows
- Retrieve rows that meet given criteria

SQL Query

- SELECT word, description
  FROM WORD_LIST_TABLE
  WHERE word="alpha"

Generic

- SELECT columns
  FROM table
  WHERE column="value"

SELECT columns FROM table

- **SELECT columns**
  - Select the columns to return
  - Use * to return all columns

- **FROM table**—specify the table from which to get results

WHERE column="value"

- **WHERE**—keyword for conditions that have to be met

- **column="value"**—the condition that has to be met
  - common operators: =, LIKE, <, >

AND, ORDER BY, LIMIT

SELECT _id FROM WORD_LIST_TABLE WHERE word="alpha" **AND** definition LIKE "%art%" **ORDER BY word DESC LIMIT 1**

- **AND, OR**—connect multiple conditions with logic operators
- **ORDER BY**—omit for default order, or ASC for ascending, DESC for descending
- **LIMIT**—get a limited number of results

Sample queries

1	SELECT * FROM WORD_LIST_TABLE	Get the whole table
2	SELECT word, definition FROM WORD_LIST_TABLE WHERE _id > 2	Returns [["alpha", "particle"]]

More sample queries

3	SELECT _id FROM WORD_LIST_TABLE WHERE word="alpha" AND definition LIKE "%art%"	Return id of word alpha with substring "art" in definition `[["3"]]`
4	SELECT * FROM WORD_LIST_TABLE ORDER BY word DESC LIMIT 1	Sort in reverse and get first item. Sorting is by the first column (_id) `[["3","alpha","particle"]]`

Last sample query

5	SELECT * FROM WORD_LIST_TABLE LIMIT 2,1	Returns 1 item starting at position 2. Position counting starts at 1 (not zero!). Returns `[["2","beta","second letter"]]`

rawQuery()

```
String query = "SELECT * FROM WORD_LIST_TABLE";
rawQuery(query, null);

query = "SELECT word, definition FROM
WORD_LIST_TABLE WHERE _id> ? ";

String[] selectionArgs = new String[]{"2"}
rawQuery(query, selectionArgs);
```

query()

`SELECT * FROM WORD_LIST_TABLE WHERE word="alpha" ORDER BY word ASC LIMIT 2,1;`
 	
Returns:
 	
`[["alpha", "particle"]]`
	`String table = "WORD_LIST_TABLE"`
	`String[] columns = new String[]{"*"};`
	`String selection = "word = ?"`
	`String[] selectionArgs = new String[]{"alpha"};`
	`String groupBy = null;`
	`String having = null;`
	`String orderBy = "word ASC"`
	`String limit = "2,1"`
	
	`query(table, columns, selection, selectionArgs, groupBy, having, orderBy, limit);`

Cursors

Queries always return a Cursor object

Cursor is an object interface that provides random read-write access to the result set returned by a database query

⇒ Think of it as a pointer to table rows

You will learn more about cursors in the following chapters

Learn more

- SQLite website
- Full description of the Query Language
- SQLite class
- Cursor class

What's Next?

- Concept Chapter: 10.0 SQLite Primer
- No Practical

END