

Mobile Application Design and Development

CS 5520

Adrienne Slaughter

Summer 2019



Today's Schedule

- Finish up with Networking
- Permissions
- MySQL/Data Storage
- Content Providers



Topics

- Storing Data on Android (locally)
 - Structured data
- Content Providers
- And review:
 - adb



DATABASES



Overview

- Storing data on Android
- SQL & SQLite overview/review
- Relevant SQLite classes in Android
- An example of creating a SQLite database in Android
- Encapsulating a SQLite database with a ContentProvider
- Using a Loader to access data from a ContentProvider
- SyncManager



Ways to Store Data

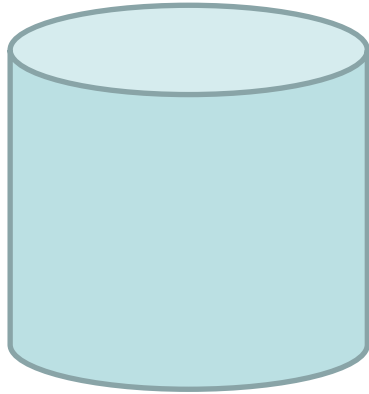
- **Shared Preferences:**
- **Raw data files:**
- **Raw data files**
- **Content from over the network:**
- **Structured DB on device:**



Relational Databases, SQL,

- Submit SQL statements, get results
- Database consists of tables, each with rows with columns
- Setup DB with Data Definition Language (DDL) statements





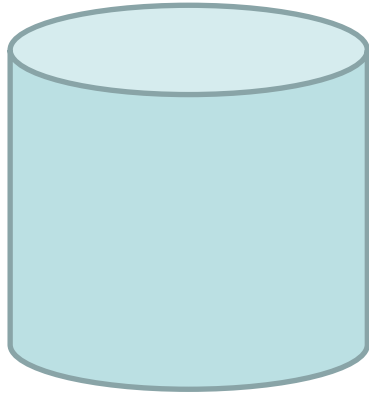
_id	age
1	54

2	_id	name	phone
3	1	Ben Bitdiddle	555 1212
	2	Alyssa Abernathy	555 9876
	3	Charlie Carpenter	555 1029

} rows

} columns

tables



_id	age
1	54

2	_id	name	phone
3	1	Ben Bitdiddle	555 1212
	2	Alyssa Abernathy	555 9876
	3	Charlie Carpenter	555 1029

} **rows**

{ **columns**

tables

DDL: Create DB

Need _id column to
implement a
ContentProvider

```
create table mytable (  
  _id integer primary key autoincrement,  
  name text,  
  phone text );
```

Column types are
hints – not enforced

Modification statements

```
insert into mytable values(null, 'Steven King', '555-1212');  
insert into mytable values(null, 'John Smith', '555-2345');  
insert into mytable values(null, 'Fred Smitheizen', '555-4321');
```

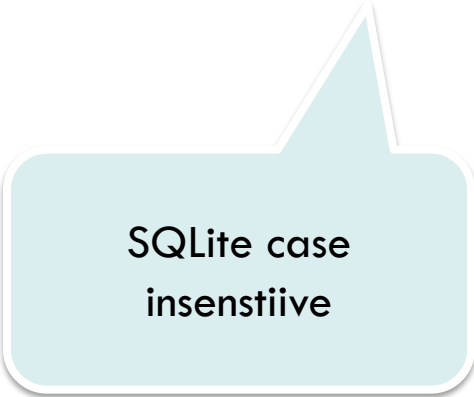
SQLite figures out
_id value
automatically

_id	name	phone
1	Ben Bitdiddle	555 1212
2	Alyssa Abernathy	555 9876
3	Charlie Carpenter	555 1029

Query statements

```
select * from mytable where(_id=3);
```

```
select name, phone from mytable where(name like "%smith%");
```



SQLite case
insensitive

SQLite Pros

- Open source
- Most widely deployed SQL DB anywhere
- Standards-compliant
- Lightweight (150kb)
- No setup or administration
- Just a file in `/data/data/packagename/databases`
- Robust (hopefully)



SQLite Cons

- Cons
 - Loosely type columns
 - Can get corrupted
 - No user management (permissions!)
 - Can't tweak performance
- Compact C library part of the Android software stack

SQLite Types

- **NULL:** NULL value.
- **INTEGER:** Signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
- **REAL:** Floating point value, stored as an 8-byte IEEE floating point number.
- **TEXT:** Text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).
- **BLOB:** A blob (binary object) of data, stored exactly as it was input.

Relevant Android packages/classes

- `android.database:`
 - Classes to explore data from a content provider.
 - `Cursor`
- `android.database.sqlite`
 - Classes to create & modify the underlying database
 - `SQLiteOpenHelper`
 - Used to create and/or open the database
 - `SQLiteQuery`
 - `SQLiteCursor`



SQLite in Android

SQLiteOpenHelper

Cursor

ContentValues

DEMO: Creating & Inserting data into a SQLite db

- **MyFriendsData: extends SQLiteOpenHelper**
 - Override onCreate(), onUpgrade()
 - Use this class or another to manage column names/types
 - VERSION is important
 - To access the database via code:
 - Create new MyFriendData, passing in the current Application Context
 - Call getWritableDatabase() or getReadableDatabase()
 - Use ContentValues to package up a new insert row
 - Pass the contentValues object to db.insertOrThrow()
 - Database is created if it doesn't already exist



DEMO: Reading from SQLite DB

- ReadDataActivity
 - Create a new MyFriendData object, passing in the Application Context
 - Get a readable/writeable database instance db
 - Get a cursor by executing a query against db
 - Call startManagingCursor
 - The Activity manages the Cursor
 - Use the cursor to either bind data to a UI component or iterate through the data.

About “managingCursor”

- Once upon a time, there were 2 ways to get data:
 - startManagingCursor
 - Manages the cursor based on the activity’s life cycle (calling activate/deactivate/requery as appropriate)
 - managedQuery
 - A wrapper; starts managing the cursor, then runs the query.
- Both methods put data access & query on the UI thread
 - ➔NO! Results in sluggish performance.

What's new: Loader & LoaderManager

- Ensures all cursor operations are done asynchronously
- LoaderManager manages Loaders associated with Activitys & Fragments
- Activities & Fragments communicate with LoaderManager about lifecycle events
- LoaderManager “does the right thing” to manage loading of data and notifying the Activity/Fragment when data updates
- Problem: a Loader requires a Content URI, not a db handle

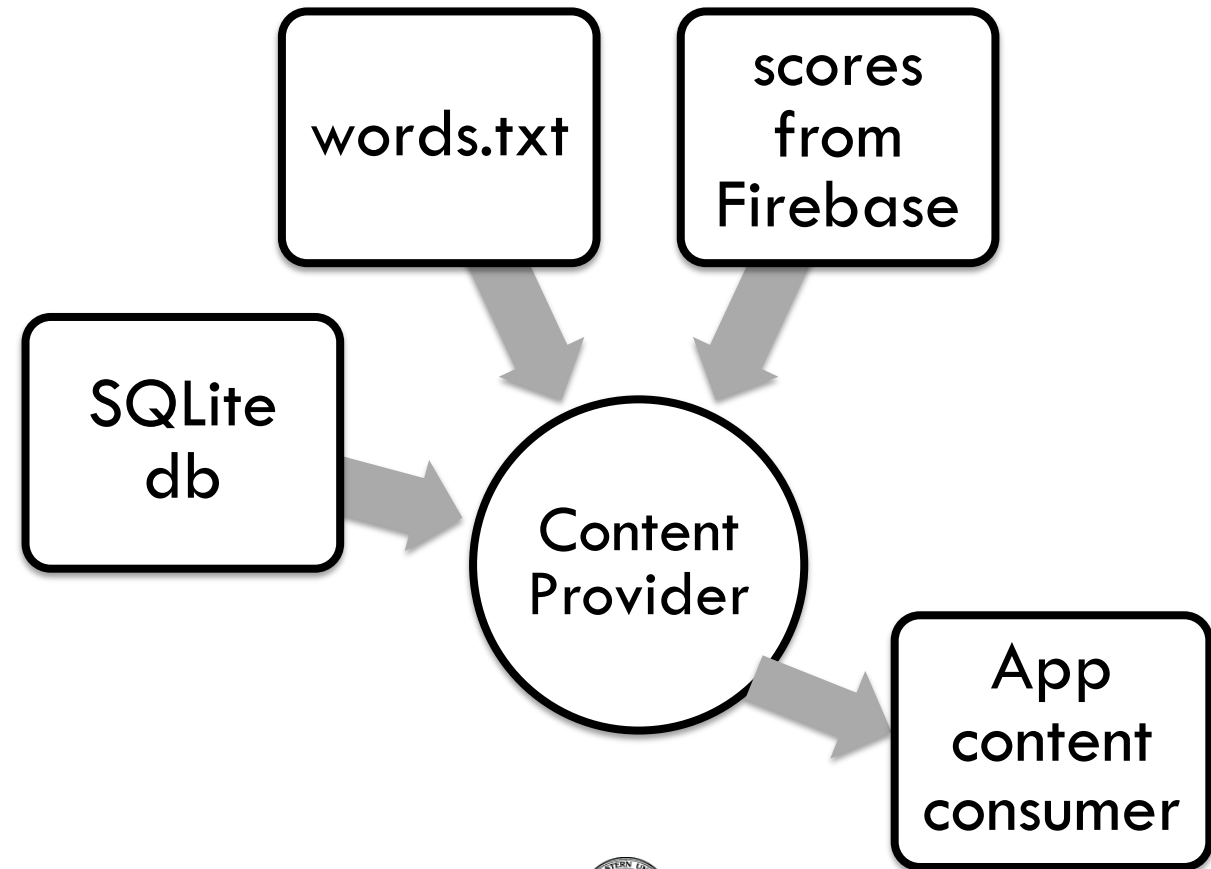
Quick Review: Android Application Components

- Activities
- Services
- Broadcast Receivers
- Content Providers



Content Provider

- Encapsulates access to a datasource provided by your app
- Provides a “public” interface
- Can be used to get data from other apps
 - Examples:
 - Contacts
 - Calendar



Writing a Content Provider

- Write your provider class, extending `ContentProvider`
- Register your provider with the manifest:
 - Specify which class services the provider
 - What the authority (namespace) is
 - Whether the provider is enabled
 - Whether the provider is exported (available for other apps to use)
- In our case, the `ContentProvider` opens the database and executes queries on it, returning the appropriate cursor
- The cursor can then be managed by the `LoaderManager`

Contract classes

- Provide the URIs, tables, columns, etc.
- A developer can provide these classes packaged into a .jar to third parties so data from the content provider can be used



Demo: Adding a ContentProvider to our Friends db



DEMO: Creating a new app using ContentProvider



sqlite3 tool

- Can connect to a device/emulator with adb to handle the database
- The DB is in `/data/data/[PACKAGE_NAME]/database/db.name`
- Manipulate, dump, restore, and work with your databases using the sqlite3 tool
- Most devices lock down the data directory and will not allow you to browse their content using standalone tools
(use emulator instead)

sqlite3 tool

- Usage

- `cd [ANDROID_HOME]/tools`
 - `adb shell` (might have to run `adb root` first)
 - `sqlite3 /data/data/com.example.program/databases/mydb.db`

- Docs: <http://www.sqlite.org/sqlite.html>

sqlite3 useful commands

- `.tables` : list tables
- `.quit` : quit
- `.databases`
- `.schema`



SQLite INSERT

```
INSERT INTO table1 (  
    column1,  
    column2 , .. )  
VALUES  
(  
    value1,  
    value2 , ... );
```

SQLite SELECT

```
SELECT * FROM table LIMIT 10;
```

```
SELECT * FROM table WHERE _id=5;
```



SQLite SELECT



DEMO: sqlite3 tool via adb



- S



Check your knowledge

- Which of the following are reasons why SQLite is so common in mobile settings?
 1. The license makes it cheap
 2. The database is many files, but very small
 3. The administration or setup is very easy



Answer:

- 3.
- SQLite is popular in mobile settings because it's small, free, and stores each database in a single file that can be moved around. Further, no administration or setup is required, which is different than many SQL databases that require someone to create users and such.



Check your knowledge

- Why is managingCursor() problematic?
 1. As a developer, you have to manage the connection to the db when the activity lifecycle changes
 2. It loads data on the UI thread
 3. Cursors are slow to get data from a database
 4. It loads all the data from the database into memory, whether or not you will use it



Answer

- 2



Review

- Storing data on Android
- SQL & SQLite overview/review
- Relevant SQLite classes in Android
- An example of creating a SQLite database in Android
- Encapsulating a SQLite database with a ContentProvider
- Using a Loader to access data from a ContentProvider

