Lecture – Part A Persistent Data in Android

DT228/3

Dr. Susan McKeever

Storing Data in Android

- A typical desktop operating system provides a common file system that any application can use to store files that can *be read by any other applications* (allowing for access settings)

- Android uses a different system: On Android, all application data(including files) are private to that application

Android Data storage

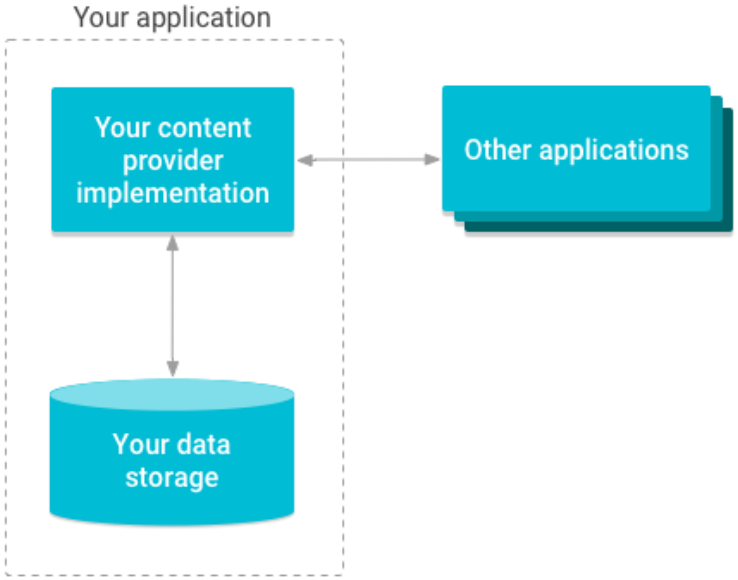
Android also provides a standard way for an application
to expose its private data to other applications — through
content providers.

Android supplies a number of content providers for
standard data types, such as
image,
audio,
video files and
personal contact information.

Android Data storage

A **content provider** manages access to a central repository of data.

Example: a Contacts Application .. For other other apps to use the contacts data, need a Content Provider for the contacts data



Use content providers if you plan to share data. If you don't plan to share data, you may still use them because they provide a nice abstraction, but you don't have to (`extend ContentProvider etc`)

Persistent Data

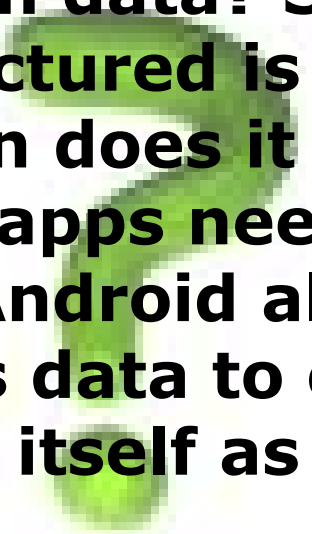
- Most apps (or any application) need a way to store persistent data
  - E.g. on your phone..
    - Contacts, SMS messages…
    - If no persistent data
      - Contacts list empty
      - No SMS msgs saved when you reopen

Persistent Data in Android apps

- Several ways : (1) Local database on the device, (2) local file, (3) cloud database

- **Choice depends on various factors -**

  - **How much data? Space?**
  - **How structured is it?**
  - **How often does it change?**
  - **Do other apps need to access it?**
  - **(BTW - Android allows an app to supply its data to others by declaring itself as *a Content Provider*)**



Choices for persistent data

1. Local device database (SQLLite)

   - App has its own private database
   - Good for structured data
   - Good for regular queries/ updates etc

2. Files
   - Can store and update information on files
   - Can be on internal or external (e.g. SD card) storage
   - Faster than databases
   - Doesn't know about data *structure*
     e.g. Age = 10 bytes, integer etc
   - To update data, may need to read through whole file

Choices for persistent data

3. Remote cloud database

- Your app might dump data onto some sort of central storage..
- Or it might retrieve data from a central store.
- E.g. Some sort of survey app for collecting data about…
- News app that stores news on remote server and updates the local app...

- API has packages to support connectivity side -

  java.net.*
  android.net.*

Choices for persistent data

For an app, when choosing how to store its data persistently..

- Files
- Local SQLLite Database
- Preferences
- Remote storage

Pros and cons for each

Simplicity/ overhead/ complexity of data to be stored/ability to query data/ structure of data/ storage capacity/reqmt for internet connection/ security of data/ frequency of data of data/ etc

Of course may use a combination in a single app

How would you store persistent Data for..

- An **emergency first aid app** on the phone – allowing user to view procedures in an emergency
- A **Mountain rescue** app that monitors where you are (GPS), monitors other user locations nearby and sends a distress signal if you request one.

SQLLite – Android's local dB soln

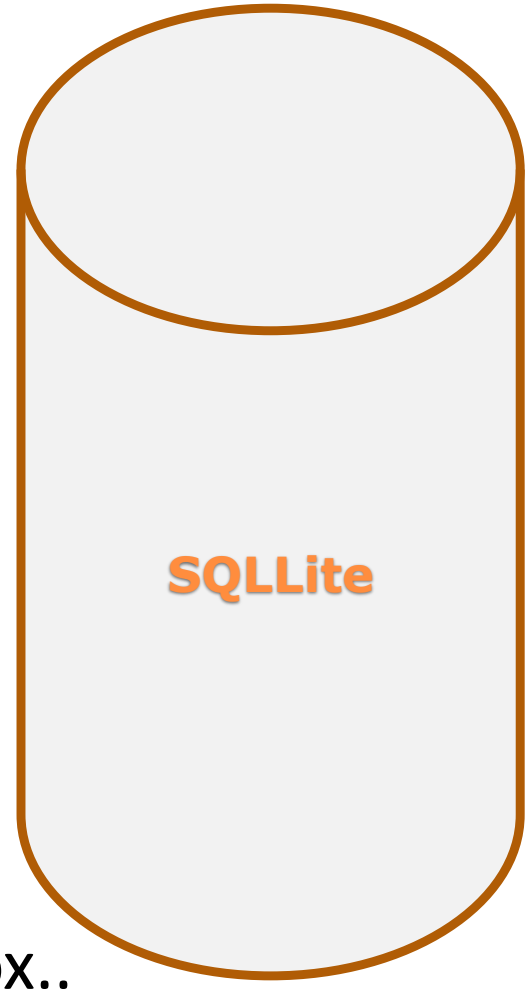
SQLLite database is well suited to mobile development

ANSI SQL compliant (i.e. standard implementation)

Runs well in memory

Android ships with the capability

Used by other s e.g. iPhone, Firefox..



Two types of dB things for an app

Manage the dB *structure*

e.g.

Create table

SQLLiteOPenHelper

***Use* the dB data in your app**

Eg.
Select data
Insert new data

SQLLiteDatabase

To use a dB in an App, you need to do both..

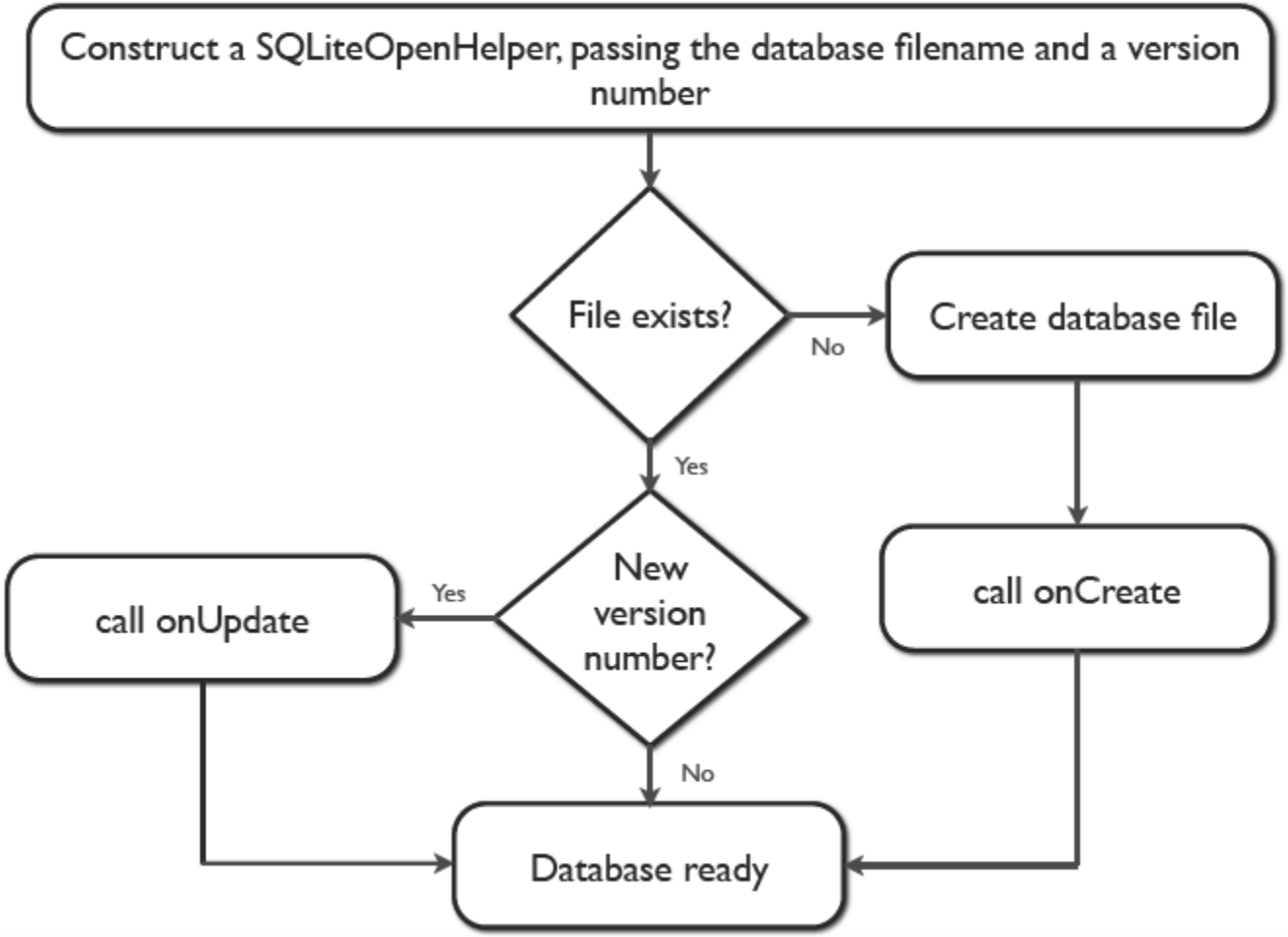
What 's in the API for SQLLite support?

- **SQLLiteDatabase** = Class containing methods to manage a local SQLLite Database file (e.g. insert rows)

- **SQLLiteOpenHelper** = utility class that makes it easy to create and maintain an SQLLite database

Once you know how to use each of these – you've mastered most of what you need to use dBs in Android

How a SQLiteOpenHelper works



SQLLiteOpenHelper

- Database and version number

- **Version number** indicates a dB structure change (i.e. if you change it.. Android will run the upgrade() method

- If dB doesn't exist yet, the SQLLiteopenhelper will create it, and call OnCreate() in *your* instance of the class.
- If it DOES exist,
  - if the version number has changed, onUpgrade () method called and you can then put in code to adjust to the new dB etc.

SQLLiteOpenHelper

Must use it for your app if you're using database *

Purpose is to

- Create the database if it doesn't exist (in the onCreate() method)
- Upgrade the database structure if it has changed (onUpgrade() method)
- Connect to the database..
  - getReadableDatabase()
  - getWritableDatabase()

*Caveat – there are some easy to use third party libraries such as Google's Room Persistence library to simplify SQLite operations)

SQLLiteOpenHelper

- Make your own helper class (by extending SQLliteopenhelper)
- Include a constructor that calls the “super” class
- dB create statement → the onCreate() method
- dB structure change → put in the the onUpgrade() method

SQLLiteOpenHelper

```
public class MyDatabaseHelper extends SQLiteOpenHelper {

// variables for setting up dB name, column names etc
    private static final String etc etc


    // constructor
    public MyDatabaseHelper(Context context) {
       super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }



    public void onCreate(SQLiteDatabase db) {
      // create dB code goes here…

    public void onUpgrade(SQLiteDatabase db, int oldVer, int newVer)
    {
        // DB structure code goes here – triggered by new version
   Etc

}
```

Remember your SQL...

Creating a Database table

The Create Table statement:

```
CREATE TABLE table_name
(column_1 type,
 [ column_N type ]);
```

Example:

```
CREATE TABLE tasks (
  id integer primary key autoincrement not null,
  name text,
  complete text
);
```

Note on code structure

- **Various ways you can structure your dB functionality**
- **Two classes needed (at least)**
  - **(1) class that uses SQLLIteOpenHelper**
  - **(2) class that does the inserts, updates, selects etc**
- Good principle is to *separate* database creation/access into separate classes from your Activity classes
- **One way** is
  - Put all database functionality into a single "database manager " class but..
    - SQLLiteOpenHelper functionality in a nested class
    - Examples..

Code Problem…Sample Code

Answers the following questions by looking at DatabaseExample . java

1. What is the overall purpose of this code?
2. How many classes are in the code?
3. What methods are in the helper class?
4. What is the name of the database it is using?
5. What method is used to execute the SQL to create a table?
6. What class does this method belong to?
7. How many tables are in the dB and what columns are in the table(s)?
8. What is the onCreate() method doing and in what class is it?
9. What is the onUpgrade() method doing?
10. What do you think “Context” is?
11. What method is being used to add data?
12. What is the ContentValues() class being used for ?
13. What is the Cursor class doing?
14. Name a method of the cursor class in use in the code & guess what it does
15. Find where the word “autoincrement” and figure out what it’s doing
16. There is a method here to select a single row from the dB. What is its name, and how is the “where” clause specified
17. What super class, if any, is being used in this code?

SQLLiteOpenHelper – sample code

Note the following from the code..

- Created its own helper for the app
- Extends the SQLLiteOpenHelperclass
- Has a constructor
  - Takes a “context”
- Has an onCreate() method
- Has an onUpgrade() method

When you declare a member class with a static modifier, it becomes a nested top-level class and can be used as a normal top-level class

ContentValues – see code

- ContentValues class is a way of referring to key value pair..
- Used to build up a set of data for the INSERT statement
- Pass in the fields..”put” in against the columns and insert

```
public long insertPerson(String firstName, String surname,
String city)
    {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_FIRSTNAME, firstName);
        initialValues.put(KEY_TITLE, surname);
        initialValues.put(KEY_PUBLISHER, surname);
        return db.insert(DATABASE_TABLE, null, initialValues);
    }
```

Underneath the covers, Android deals with putting it into the INSERT statement

ContentValues –

Also can use it for **updates..**

Need to tell the system the new values and which row(s) to update

```
public boolean updatePerson(long rowId, String firstName,
    String surname, String city)
    {
        ContentValues args = new ContentValues();
        args.put(KEY_FIRSTNAME, firstName);
        args.put(KEY_SURNAME, surname);
        args.put(KEY_CITY, city);
        return db.update(DATABASE_TABLE, args,
                         KEY_ROWID + "=" + rowId, null) > 0;
    }
```

Underneath the covers, Android deals with putting it into the sql update statement

Code sample –

```
ContentValues updateModule = new ContentValues();
updateModule.put("module_name", "Algorithms");
db.update("tbl_modules", updateModules, "id=?", new
String[] {Long.toString(moduleId)});
```

What's this piece of code doing?
What is the table name?
What column and column value is being changed?
Which row will it update?
Any comments on hard coding?

Sample Code – more questions

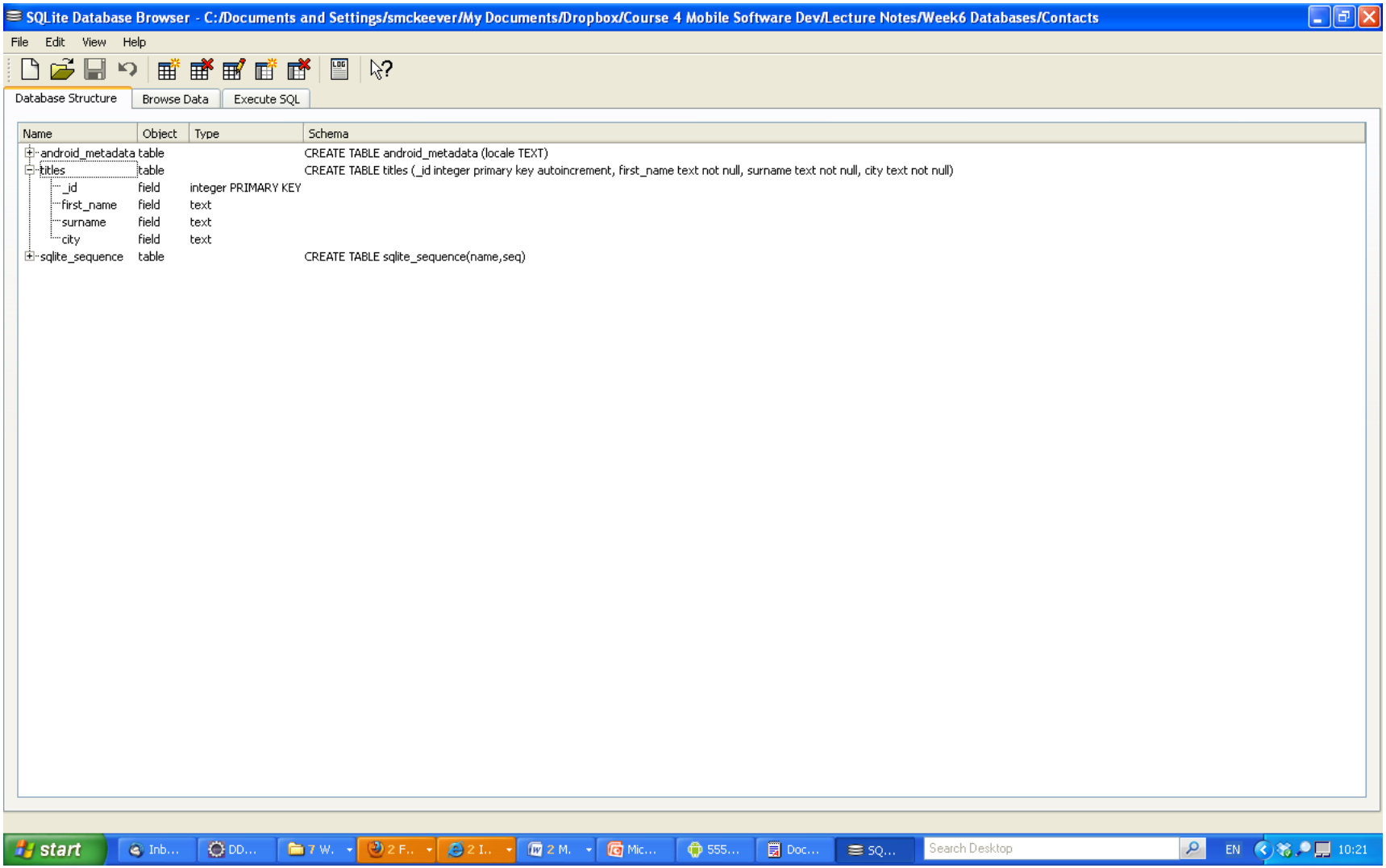
- If you wanted to use a different dB e.g. to store book titles instead of people: What code would you change? - ..….
- With existing code: How/where would you add code if you wanted to update a row on the database (e.g. person's moves to a new city)
- How would you see this code being used? Does it display a screen? What does it do?

How to "see" your database for debugging?

- First... get a copy of the dB off the device.
- If using an AVD -
  - Click "device explorer" on RHS of android studio screen
  - Browse into Data/data directories for your package
  - File save as for the dB file.
  - Use any SQLLite browser to look at contents..

- If you're using your own phone, need root access and specific steps to pull the dB out of the phone on to your hard drive. Google!

SQLLite Browsers



How to USE the database in an app

- We've created the class that sets up the database and calls the SQL methods

- To **use** the class.. Your activity(s) will need to instantiate this class and call whichever methods are needed.

- Remember to get (open) and close the database
- Closing the database should be done in appropriate lifecycle method (more in another topic)

Sample Code..PersonSearchActivity..