CC BY

How to access your database from the development environment

Marco Ronchetti
Università degli Studi di Trento

Find your DB

1) Look into data/data/YOURPACKAGE/databases/YOURDATABASE.db

2) Pull the file on your PC

3) Use sqlite on your PC (in your_sdk_dir/tools)

Access your DB

Use the following script, and

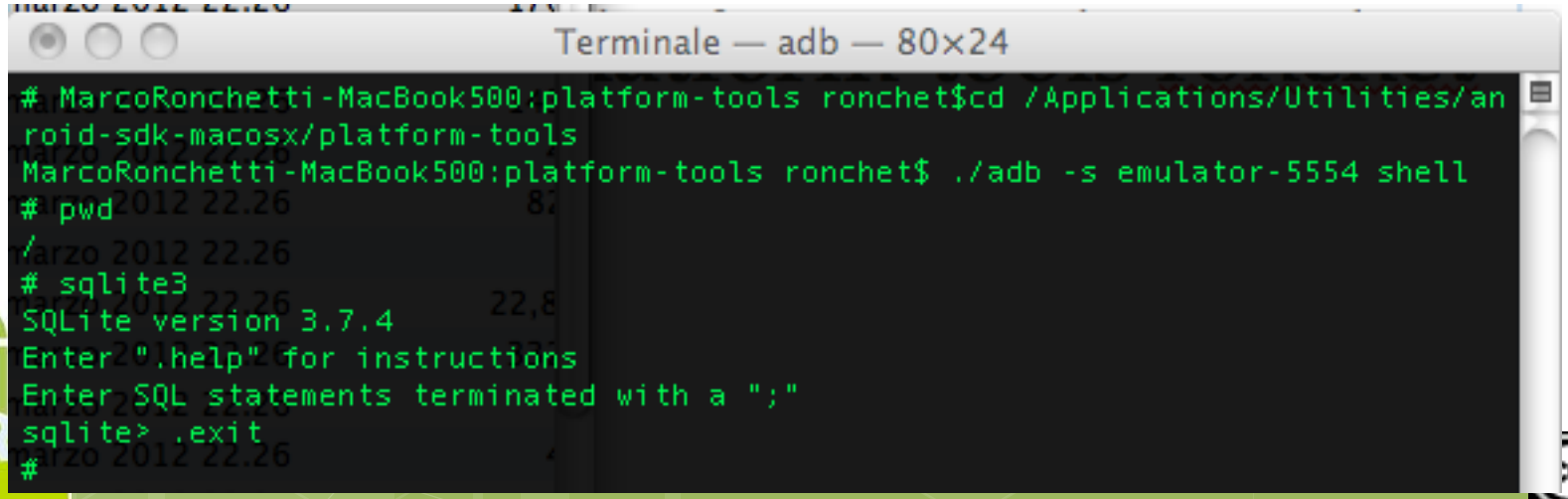
```
#!sh
adb shell "chmod 777 /data/data/com.mypackage/databases/store.db"
adb pull /data/data/com.mypackage/databases/store.db
```

OR

Run remote shell

```
$ adb –s emulator-5554 shell
$ cd /data/data/com.yourpackage/databases
$ sqlite3 your-db-file.db
> .help
```

adb -s <serialNumber> <command> to access a device



adb

adb is in your android-sdk/platform-tools directory

It allows you to:

- Run shell commands on an emulator or device
- Copy files to/from an emulator or device
- Manage the state of an emulator or device
- Manage port forwarding on an emulator or device

It is a client-server program that includes three components:

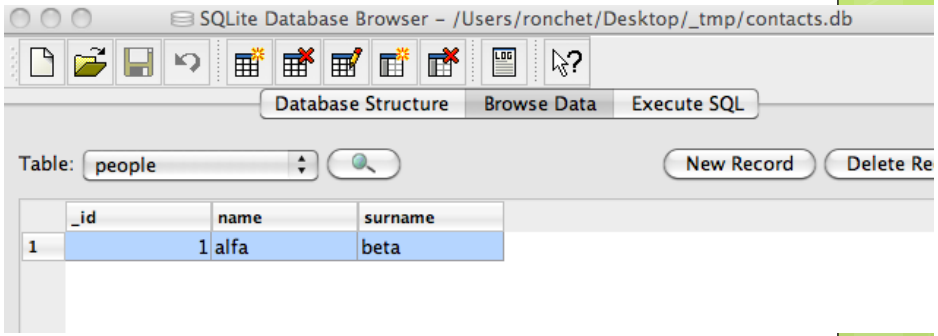
- A client, which runs on your development machine.
- A daemon, which runs as a background process on each emulator or device instance.
- A server, which runs as a background process on your development machine and manages communication between the client and the daemon.

See https://developer.android.com/studio/command-line/adb

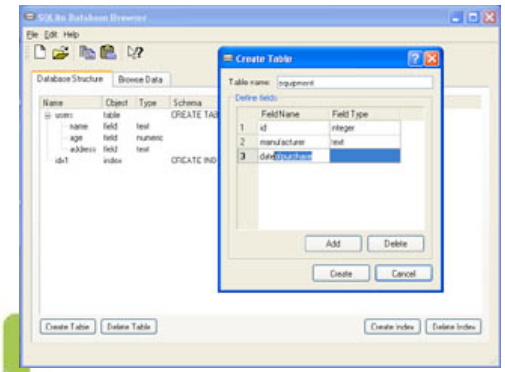
cc BY

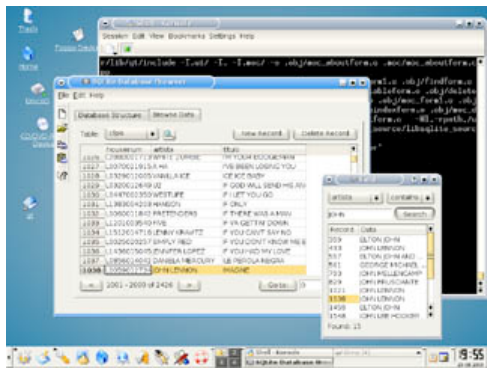
A graphical sqlite browser

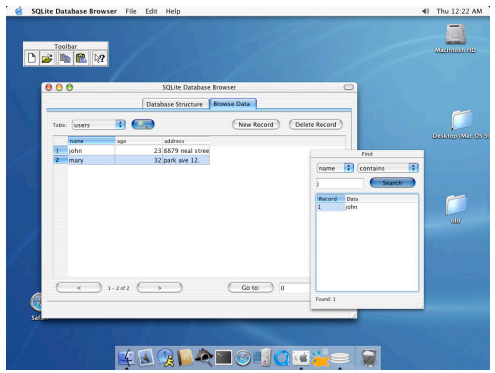
http://sqlitebrowser.org/



- Create and compact database files
- Create, define, modify and delete tables
- Create, define and delete indexes
- Browse, edit, add and delete records
- Search records
- Import and export records as text
- Import and export tables from/to CSV files
- Import and export databases from/to SQL dump files
- Issue SQL queries and inspect the results
- Examine a log of all SQL commands issued by the application







cc BY

6

CC BY

Testing and deploying on your device

Marco Ronchetti
Università degli Studi di Trento

Simple way to deploy

e.g. to give your app to your friends

Get Dropbox both on PC and Android device
Copy your apk from bin/res into dropbox (on PC)
Open dropbox on Android device, and open your apk

By sharing your dropbox with others you can easily pass your app.

www.dropbox.com

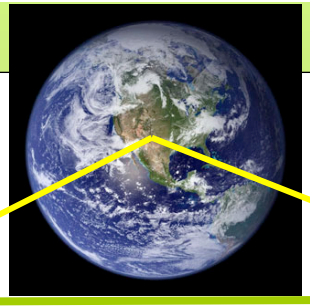
8

CC BY

DAO Implementation File System

Marco Ronchetti
Università degli Studi di Trento

ORM - DAO



WORLD

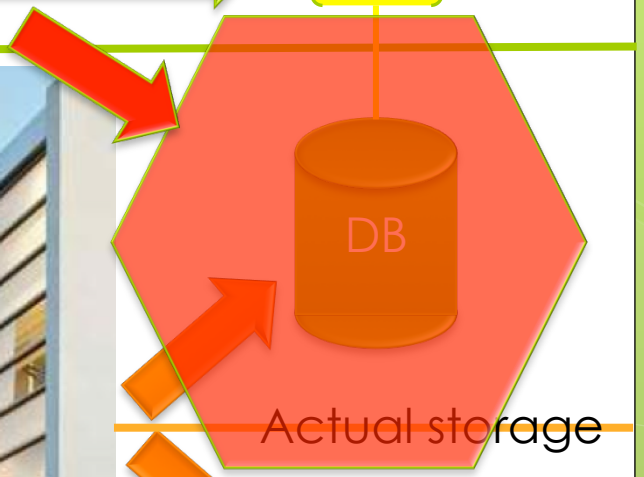
MODEL

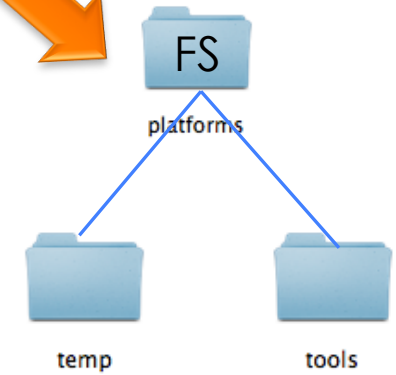
UML

ORM

ERA

ARCHITECTURE





cc BY

The Java-IO philosophy

1) Get a (raw) source

File f; … ; InputStream s = new FileInputStream(f);
Socket s; … ; InputStream s=s.getInputStream();
StringBuffer b; … ; InputStream s = new StringBufferInputStream(f);

2) Add functionality

Reader r=new InputStringReader(s); //bridge class
DataInputString dis=new DataInputString(s); //primitive data
ObjectInputString ois=new ObjectInputString(s); //serialized objects

3) Compose multiple functionalities

```
InputStream es=new FilteredInputStream(
                    new BufferedInputStream(
                    new PushBackIputStream(s)));
```

Choose the type of source!

You can choose among four types of basic sources:

	BYTE		CHARACTER	
SOURCE	InputStream	OutputStream	Reader	Writer

Both file and directory information is available via the File class, or the classes (like Path) in the nio package.

cc BY

I/O Table

	Byte Based		Character Based	
	Input	*Output*	*Input*	*Output*
Basic	InputStream	OutputStream	Reader InputStreamReader	Writer OutputStreamWriter
Arrays	ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArrayWriter
Files	FileInputStream RandomAccessFile	FileOutputStream RandomAccessFile	FileReader	FileWriter
Pipes	PipedInputStream	PipedOutputStream	PipedReader	PipedWriter
Buffering	BufferedInputStream	BufferedOutputStream	BufferedReader	BufferedWriter
Filtering	FilterInputStream	FilterOutputStream	FilterReader	FilterWriter
Parsing	PushbackInputStream StreamTokenizer		PushbackReader LineNumberReader	
Strings			StringReader	StringWriter
Data	DataInputStream	DataOutputStream		
Data - Formatted		PrintStream		PrintWriter
Objects	ObjectInputStream	ObjectOutputStream		
Utilities	SequenceInputStream			

cc BY

Android internal file I/O

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME,
        Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

NOTA: MODE_WORLD_READABLE I s deprecated!
See https://developer.android.com/reference/android/content/Context

Using temporary files

```
File file = new File(getCacheDir(), "temp.txt");
 try {
   file.createNewFile();
   FileWriter fw = new FileWriter(file);
   BufferedWriter bw = new BufferedWriter(fw);
   bw.write("Hello World\n");
   bw.close();
} catch (IOException e) {
        Toast.makeText(this,
             "Error creating a file!"
             ,Toast.LENGTH_SHORT).show();
}
```

When the device is low on internal storage space, Android may delete these cache files to recover space.

You should not rely on the system to clean up these files for you.

Clean the cache files yourself

stay within a reasonable limit of space consumed, such as 1MB.

Other useful methods

getFilesDir()

Get the absolute path where internal files are saved.

getDir()

Creates (or opens an existing) directory within your internal storage space.

deleteFile()

Deletes a file saved on the internal storage.

fileList()

Returns an array of files currently saved by your application.

The DAO interface

```
package it.unitn.science.latemar;

import java.util.List;

public interface PersonDAO {
        public void open();
        public void close();

        public Person insertPerson(Person person) ;
        public void deletePerson(Person person) ;
        public List<Person> getAllPerson() ;
}
```

package it.unitn.science.latemar;
import ...

The DAO implementation - FS

```
public class PersonDAO_FS_impl implements PersonDAO {
        DataOutputStream fos;
        DataInputStream fis;
        Context context=MyApplication.getAppContext();
        final String FILENAME="contacts";

@Override
        public void open() {
          try {
              fos=new DataOutputStream(
                    context.openFileOutput(FILENAME, Context.MODE_APPEND)
              );
          } catch (FileNotFoundException e) {e.printStackTrace();}

        }
@Override
        public void close() {
          try {
              fos.close();
           } catch (IOException e) {e.printStackTrace();}
        }
```

This should never happen

cc BY

The DAO impl. – data access 2

```
@Override
  public Person insertPerson(Person person) {
      try {
          fos.writeUTF(person.getName());
          fos.writeUTF(person.getSurname());
      } catch (IOException e) { e.printStackTrace(); }
      return person;
  }

@Override
   public void deletePerson(Person person) {
       Log.d("trace","deletePerson DAO_FS – UNIMPLEMENTED!");
   }
```

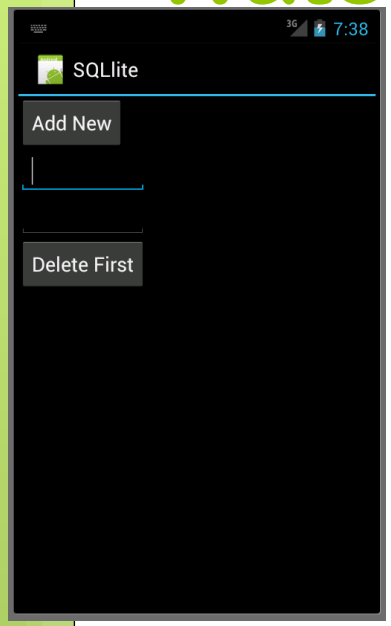
write as Unicode

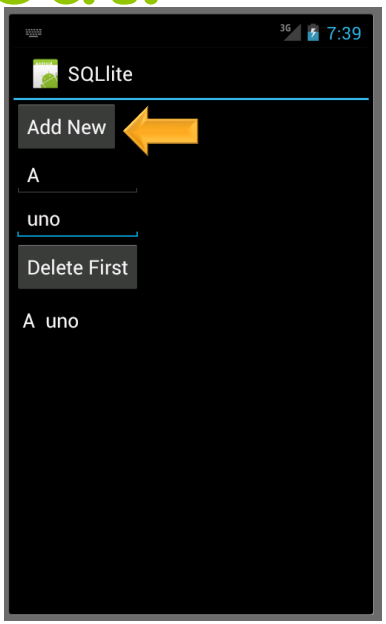
cc BY

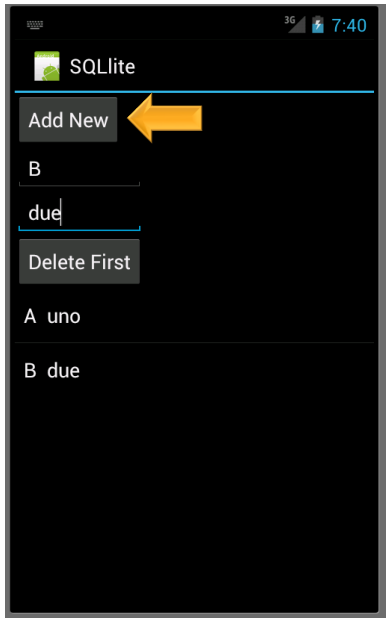
The DAO impl. – data access 3

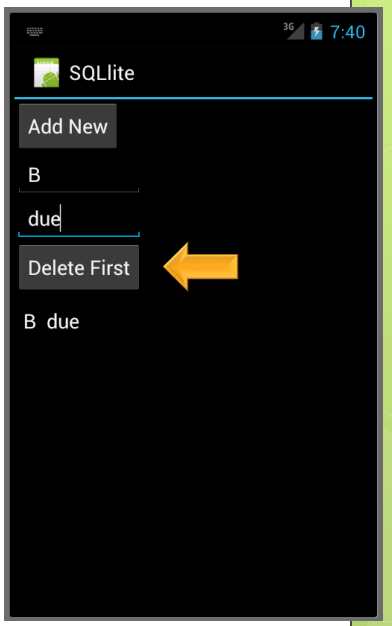
```
@Override
  public List<Person> getAllPersons() {
      List<Person> list=new ArrayList<Person>();
      try { fis=new DataInputStream( context.openFileInput(FILENAME) );
       } catch (FileNotFoundException e) {
           e.printStackTrace();    return list;
       }
       while (true) {
           try {
                   String name=fis.readUTF();
                   String surname=fis.readUTF();
                   Person p=new Person(name, surname);
                   list.add(p);
            } catch (EOFException e) {  break;
            } catch (IOException e) { e.printStackTrace(); break; }
     }
     try { fis.close(); } catch (IOException e) { e.printStackTrace(); }
     return list;
  }
```

Watch out!

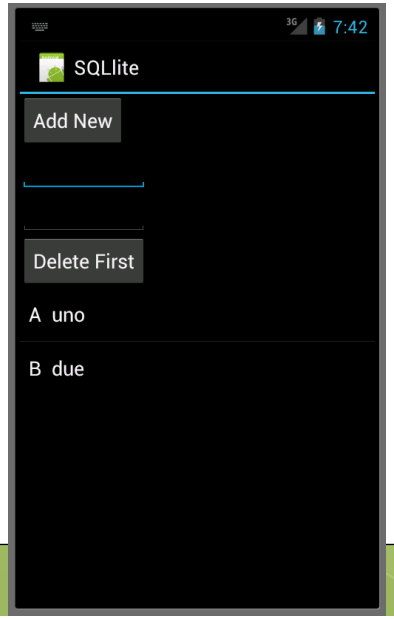








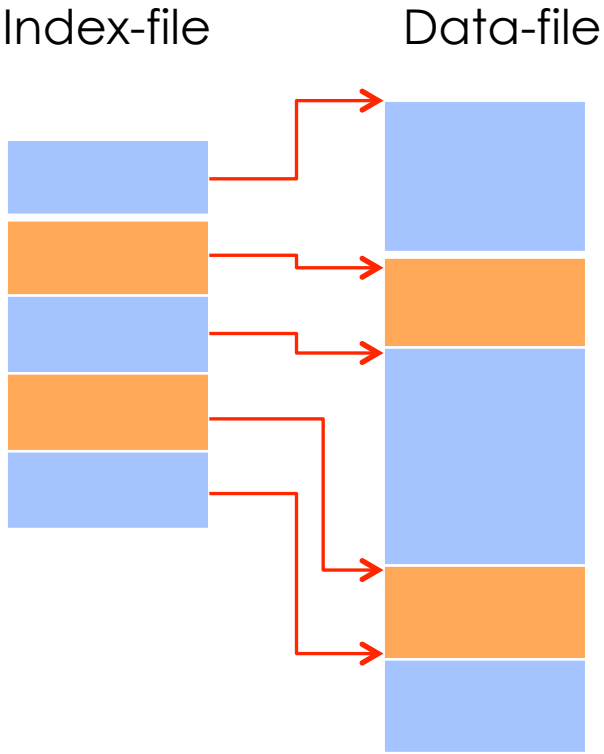
Restart…



Why so?

Serializing any-size objects to a random access file

http://photonsphere.org/posts-output/2011-06-29-serializing-any-size-objects-to-a-random-access-file/



See java.io
Class RandomAccessFile

cc BY

CC BY

External Files

Marco Ronchetti
Università degli Studi di Trento

External storage

Every Android-compatible device supports a shared "external storage" that you can use to save files.

It can be:

- a removable storage media (such as an SD card)
- an internal (non-removable) storage.

Files saved to the external storage

- are world-readable
- can be modified by the user when the USB card storage in moved on a computer!

Possible states of external media

String Environment.getExternalStorageState();

MEDIA_MOUNTED
- media is present and mounted at its mount point with read/write access.

MEDIA_MOUNTED_READ_ONLY
- media is present and mounted at its mount point with read only access.

MEDIA_NOFS
- media is present but is blank or is using an unsupported filesystem

MEDIA_CHECKING
- media is present and being disk-checked

MEDIA_UNMOUNTED
- media is present but not mounted

MEDIA_SHARED
- media is in SD card slot, unmounted, and shared as a mass storage device.

MEDIA_UNMOUNTABLE
- media is present but cannot be mounted.

MEDIA_REMOVED
- media is not present.

boolean Environment.isExternalStorageEmulated()
boolean Environment.isExternalStorageRemovable()

MEDIA_BAD_REMOVAL
- media was removed before it was unmounted.

Standard directories (constants):

DIRECTORY_DOWNLOADS

- files that have been downloaded by the user.

DIRECTORY_MOVIES

- movies that are available to the user.

DIRECTORY_PICTURES

- pictures that are available to the user.

DIRECTORY_DCIM

- The traditional location for pictures and videos when mounting the device as a camera.

Places for audio files:

- DIRECTORY_MUSIC
  - music for the user.
- DIRECTORY_ALARMS
  - alarms sounds that the user can select (not as regular music).
- DIRECTORY_NOTIFICATIONS
  - notifications sounds that the user can select (not as regular music).
- DIRECTORY_PODCASTS
  - podcasts that the user can select (not as regular music).
- DIRECTORY_RINGTONES
  - ringtones that the user can select (not as regular music).

Other Environment static methods

static File **getRootDirectory**()

- Gets the Android root directory (typically returns /system).

static File **getDataDirectory**()

- Gets the Android data directory (typically returns /data).

static File **getDownloadCacheDirectory**()

- Gets the Android Download/Cache content directory. Here go **temporary** files that **are specific to your application** If the user uninstalls your application, this directory and all its contents will be deleted. You should manage these cache files and remove those that aren't needed in order to preserve file space.

static File **getExternalStorageDirectory**()

- Gets the Android external storage directory. Here go files that **are specific to your application** If the user uninstalls your application, this directory and all its contents will be deleted.

static File **getExternalStoragePublicDirectory**(String type)

- Get a top-level public external storage directory for placing files of a particular type. This is where the user will typically place and manage their own files. Here go **files that are not specific to your application** and that should *not* be deleted when your application is uninstalled