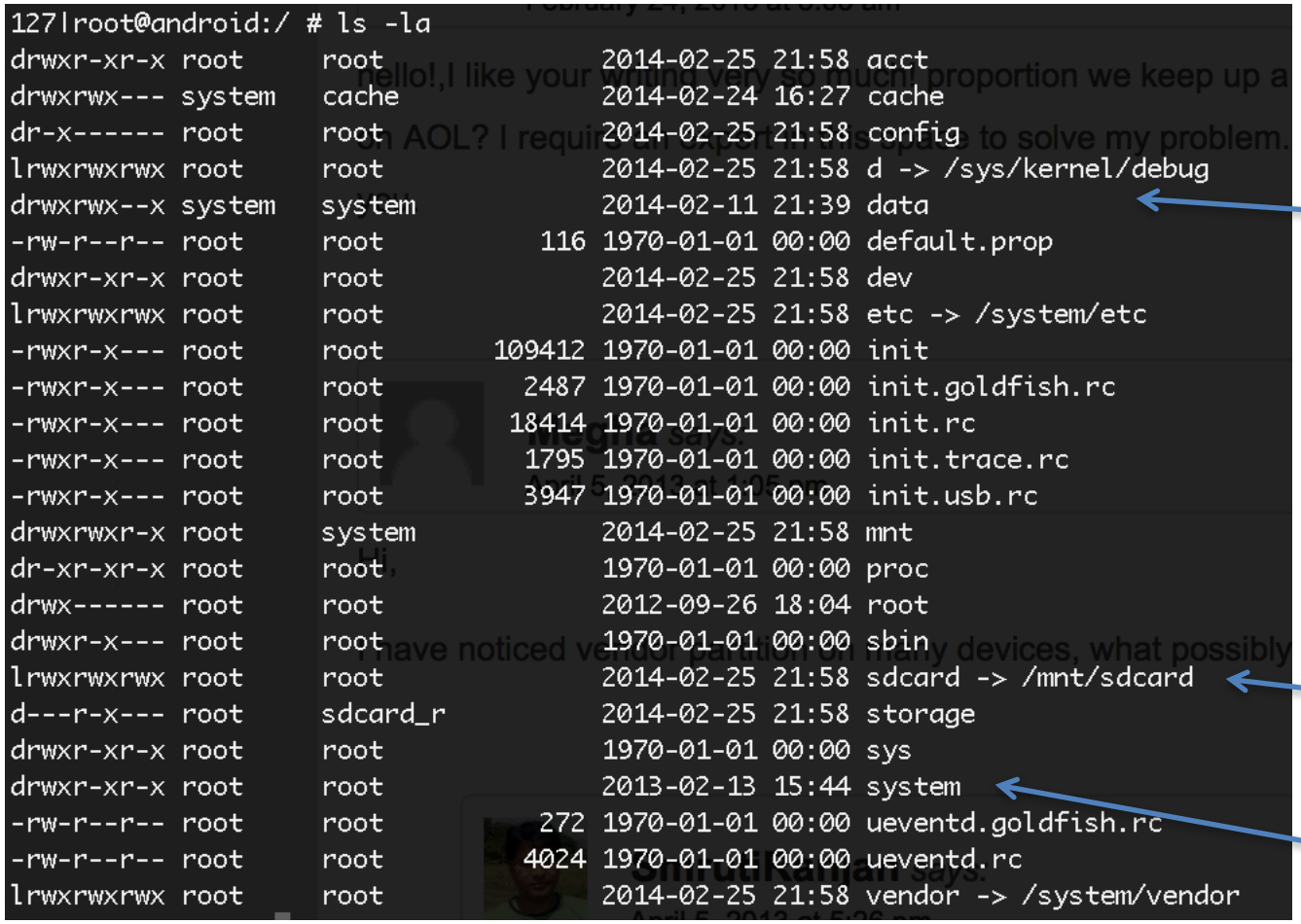
G53MDP
Mobile Device Programming

Storage

- What sort of data is stored “on” your phone?

Logical Data Storage on Android

- File-based abstractions
  - Shared Preferences
    - Simple key value pairs
  - File-based storage
    - Internal Data Storage
      - Soldered RAM
      - Internal APK resources, temporary files
    - External Data Storage
      - SD Card
      - Large media files
  - SQLite Database
    - Structured data, small binary files
- Network
  - SyncAdapters
    - Shared contact lists, backups



"User" data – application data

"External" storage

Android OS / libraries

Internal File Storage

- Internal Data storage is private to the app
  - Other apps (and the user) cannot access it
    - Kernel enforced user permissions
  - Removed on uninstall
    - Why?
  - Data is stored in Files
    - /data/data/com.example.martinstorage/files/
    - openRawResource
      - Can be used to read our own packaged resources
- Android provides a standard place to store (small) cache files
  - FingerPainterView?
  - /data/data/com.example.martinstorage/cache
  - Use getCacheDir() to get a File for the directory
  - Still need to manage the files yourself
    - **May** be deleted when internal storage becomes full / contested
    - **Will** be deleted when the application is uninstalled
    - A “well behaved” application will delete them when no longer in use
    - Recommended to use less than 1MB

Shared Preferences

- Internal Storage
- Stored on a per-*application* basis
  - I.e. all components in an application may access the same Shared Preferences
    - But should not be used for data transfer (instead of Intents, Binder etc)
- Primitive data in key-value pairs
  - Primitives
    - Strings, integers etc
    - Not Bundles
- Can have multiple *preference files* per application

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
   <string name="preference 1">preference</string>
</map>
```

External File Storage

- Every Android device provides externally-accessible storage, e.g. SD card
  - Even those phones without an SD card
    - Logical representation of “external” storage
      - Single storage device partitioned into internal / external
  - “Private” application files
    - *Internal Storage* on the External partition (!)
  - “Public” general files
    - World readable
    - Other applications can read and modify these files
    - Each “user” has their own *virtual* SD card
- Can be mounted externally (and/or disconnected)
- Before accessing files need to check the state of external storage
  - It may not be there, or mounted by something else

External Data Storage

- Check state with Environment.getExternalStorageState()
  - It is a separate file system
  - Returns a String containing the details
  - Compare with the constants:
    - Environment.MEDIA_MOUNTED
    - Environment.MEDIA_MOUNTED_READ_ONLY
- Use getExternalStoragePublicDirectory(String type) to obtain a File for the directory
  - Pass a type to obtain a sub-directory for that type
    - Used to enable the Media scanner to categorize material
  - Use File object returned to createNewFile()
  - Scoped Directory access
    - “With each new release, developers have been provided with new and updated APIs to work with”
- getExternalFilesDir()
  - Provides private external storage
  - /sdcard/Android/data/com.example.pszmdf.fingerpainter/

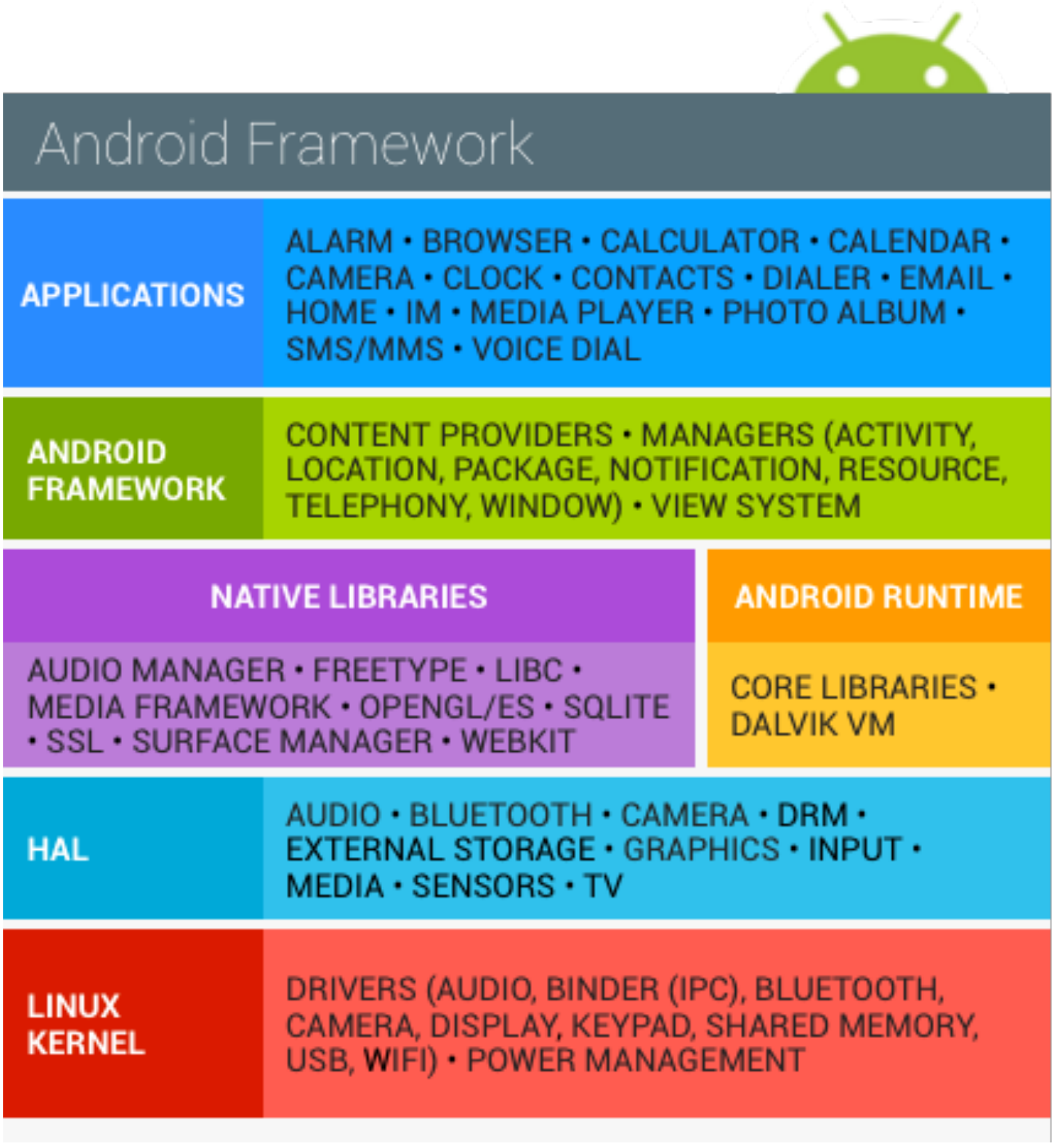
Fields		
public static String	DIRECTORY_ALARMS	Standard directory in which to the list of alarms that the user
public static String	DIRECTORY_DCIM	The traditional location for pict device as a camera.
public static String	DIRECTORY_DOWNLOADS	Standard directory in which to by the user.
public static String	DIRECTORY_MOVIES	Standard directory in which to user.
public static String	DIRECTORY_MUSIC	Standard directory in which to the regular list of music for the
public static String	DIRECTORY_NOTIFICATIONS	Standard directory in which to the list of notifications that the
public static String	DIRECTORY_PICTURES	Standard directory in which to user.
public static String	DIRECTORY_PODCASTS	Standard directory in which to the list of podcasts that the us
public static String	DIRECTORY_RINGTONES	Standard directory in which to the list of ringtones that the us

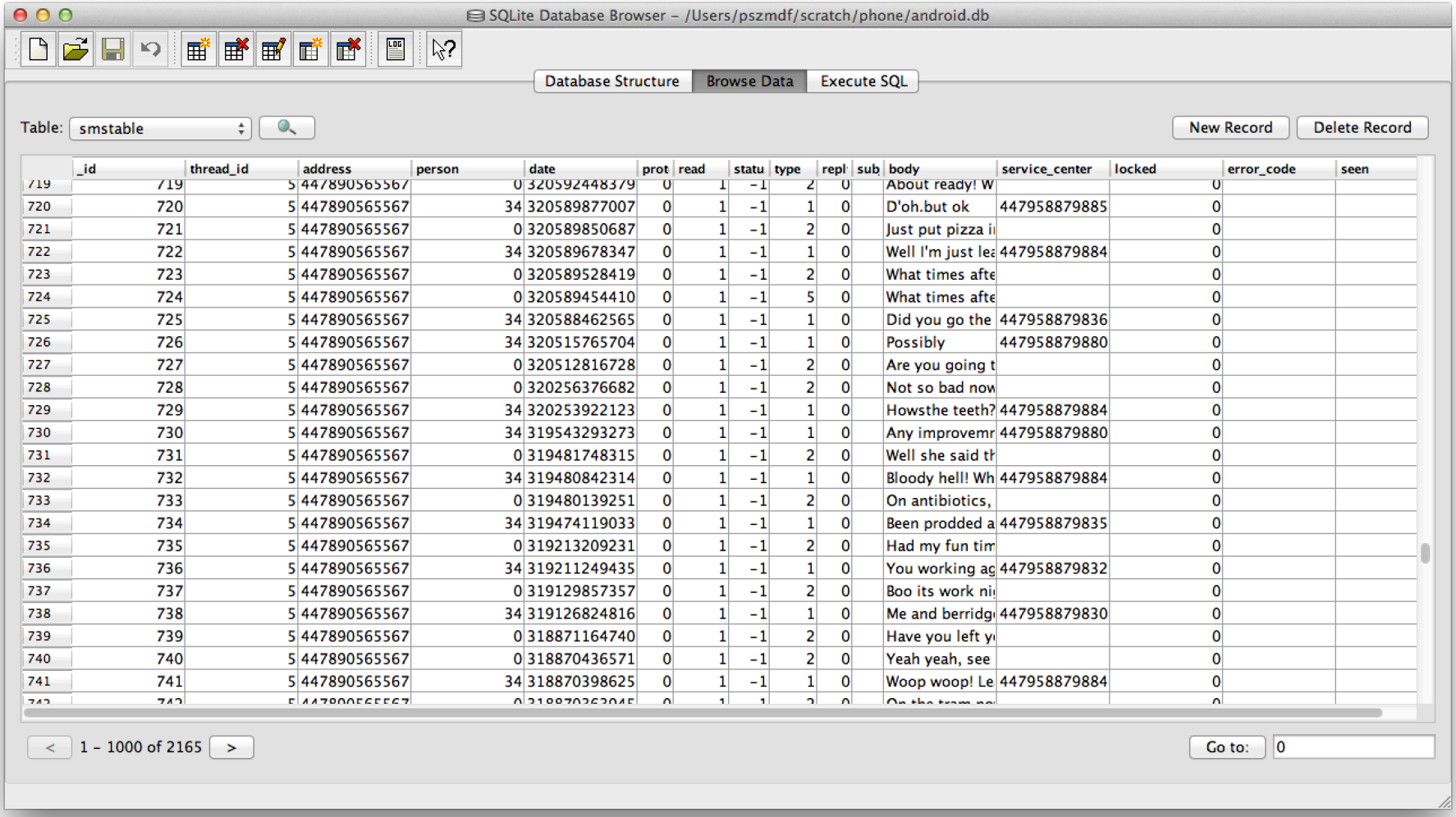
Structured Data

- Often the data we are storing is logically structured
  - Need to query it based on that structure
- Could store this in a file and write our own routines to access it
  - Obb virtual file system
    - StorageManager (wrapper for MountService *system service*)
    - Opaque Binary Blobs
- Normally, we'd use a database to store it
  - E.g. An address book, music library
  - V.s. binary "blobs"
    - Images, mp3s
  - Media gallery?

Android Databases

- Android comes with local database support
  - Complete with the ability to run full SQL queries
  - Each app's databases are local to it
    - Database.db stored in Internal Storage
- Uses SQLite
  - Public Domain software library
  - "A software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine."
    - File based
  - "Most widely deployed software engine on the planet"


Android Framework
APPLICATIONS
ALARM • BROWSER • CALCULATOR • CALENDAR • CAMERA • CLOCK • CONTACTS • DIALER • EMAIL • HOME • IM • MEDIA PLAYER • PHOTO ALBUM • SMS/MMS • VOICE DIAL
ANDROID FRAMEWORK
CONTENT PROVIDERS • MANAGERS (ACTIVITY, LOCATION, PACKAGE, NOTIFICATION, RESOURCE, TELEPHONY, WINDOW) • VIEW SYSTEM
NATIVE LIBRARIES
AUDIO MANAGER • FREETYPE • LIBC • MEDIA FRAMEWORK • OPENGL/ES • SQLITE • SSL • SURFACE MANAGER • WEBKIT
ANDROID RUNTIME
CORE LIBRARIES • DALVIK VM
HAL
AUDIO • BLUETOOTH • CAMERA • DRM • EXTERNAL STORAGE • GRAPHICS • INPUT • MEDIA • SENSORS • TV
LINUX KERNEL
DRIVERS (AUDIO, BINDER (IPC), BLUETOOTH, CAMERA, DISPLAY, KEYPAD, SHARED MEMORY, USB, WIFI) • POWER MANAGEMENT


SQLite Database Browser – /Users/pszmdf/scratch/phone/android.db
Database Structure
Browse Data
Execute SQL
Table: smstable
New Record
Delete Record
_id thread_id address person date prot read statu type repl sub body service_center locked error_code seen
719 719 5 447890565567 0 320592448379 0 1 -1 2 0 About ready! W 0
720 720 5 447890565567 34 320589877007 0 1 -1 1 0 D'oh.but ok 447958879885 0
721 721 5 447890565567 0 320589850687 0 1 -1 2 0 Just put pizza i 0
722 722 5 447890565567 34 320589678347 0 1 -1 1 0 Well I'm just lea 447958879884 0
723 723 5 447890565567 0 320589528419 0 1 -1 2 0 What times afte 0
724 724 5 447890565567 0 320589454410 0 1 -1 5 0 What times afte 0
725 725 5 447890565567 34 320588462565 0 1 -1 1 0 Did you go the 447958879836 0
726 726 5 447890565567 34 320515765704 0 1 -1 1 0 Possibly 447958879880 0
727 727 5 447890565567 0 320512816728 0 1 -1 2 0 Are you going t 0
728 728 5 447890565567 0 320256376682 0 1 -1 2 0 Not so bad now 0
729 729 5 447890565567 34 320253922123 0 1 -1 1 0 Howsthe teeth? 447958879884 0
730 730 5 447890565567 34 319543293273 0 1 -1 1 0 Any improvemr 447958879880 0
731 731 5 447890565567 0 319481748315 0 1 -1 2 0 Well she said th 0
732 732 5 447890565567 34 319480842314 0 1 -1 1 0 Bloody hell! Wh 447958879884 0
733 733 5 447890565567 0 319480139251 0 1 -1 2 0 On antibiotics, 0
734 734 5 447890565567 34 319474119033 0 1 -1 1 0 Been prodded a 447958879835 0
735 735 5 447890565567 0 319213209231 0 1 -1 2 0 Had my fun tim 0
736 736 5 447890565567 34 319211249435 0 1 -1 1 0 You working ag 447958879832 0
737 737 5 447890565567 0 319129857357 0 1 -1 2 0 Boo its work ni 0
738 738 5 447890565567 34 319126824816 0 1 -1 1 0 Me and berridg 447958879830 0
739 739 5 447890565567 0 318871164740 0 1 -1 2 0 Have you left y 0
740 740 5 447890565567 0 318870436571 0 1 -1 2 0 Yeah yeah, see 0
741 741 5 447890565567 34 318870398625 0 1 -1 1 0 Woop woop! Le 447958879884 0
< 1 - 1000 of 2165 >
Go to: 0

Android and SQLite

- Wrapped up in two main classes
  - Database represented by SQLiteDatabase
    - Lets us run SQL queries on the database
  - Also provides SQLiteOpenHelper to help create the database
    - *Application* lifecycle
      - SQLiteOpenHelper onCreate()
      - SQLiteOpenHelper onUpgrade(int oldVersion, int newVersion)

Using Databases

- SQLiteOpenHelper manages database creation and upgrades between versions
  - Create a subclass of it
  - Override onCreate to provide the code to create the database
  - Using SQL CREATE TABLE
  - Handled automatically
- Create an instance of our SQLiteOpenHelper subclass
- Obtain reference to SQLiteDatabase using:
  - getReadableDatabase()
  - getWriteableDatabase()
- Both return the same object, unless memory is low and can only open the DB readonly

Querying a Database

- Some abstraction supported
- void execSQL()
  - used to run SQL queries that don't return anything

```
execSQL("INSERT INTO myList (name, colour) VALUES
('banana','yellow');");
```

- query() and rawQuery()
  - These return a Cursor object pointing to the results
- `Cursor rawQuery(String sql, String[] selectionArgs)`
  - processes a raw SQL query

`rawQuery("SELECT id, name FROM people WHERE name = ? AND id = ?", new String[] {"Martin", "78"});` SQL has to be parsed so there is also query() where the SQL is exploded into separate strings

  - Simpler to construct a query programmatically
    - A projection onto / a subset of columns

```
Cursor query(String table, String[] columns, String selection,
String[] selectionArgs, String groupBy, String having, String
orderBy)
```

Cursors

- Provides random access to results of a query
- Fairly self explanatory object
  - Enables us to step over all the rows returned by a query
    - moveToFirst(), moveToNext()
    - getString(columnIndex), getInt(columnIndex)
      - Where column index is index of projection result
  - Has a close() method to close the query when finished
    - Shouldn't wait for it to be garbage collected
    - IPC implications
      - Can we pass a cursor to another process? (component number 3)
- "Connect" a cursor to a CursorAdapter and ListView
  - Data driven interfaces
  - SimpleCursorAdapter(...Cursor...)
  - Map the projection to a View layout for a single item, populate a list of views
    - Link resource IDs to projection columns
    - **Requires each row to have an "_id" field**
  - Can extend BaseAdapter for more sophisticated data->row mapping

CursorLoader

- A query may last some time
  - Database may be large
  - Database may be in a different process
    - How?
  - *Don't block the main UI thread*
- CursorLoader
  - Populates views asynchronously
  - Auto Updating
    - Monitors for notification that content has changed
      - Again, how?

```
getLoaderManager().initLoader(0, null, this);
public Loader<Cursor> onCreateLoader(int id, Bundle args)
```

  - Multiple loaders associated with an *Activity*

```
onLoadFinished(Loader<Cursor> loader, Cursor data)
    simpleCursorAdapter.swapCursor);
```

  - Relative of AsyncTask, returns to main thread to interact with UI element

Database Abstraction

- Good software architecture
  - Separation of data model from presentation / views
- Abstraction of database architecture
  - Easier to update storage code
  - Expose column indices as static class variables
    - c.getInt(0) -> c.getInt(DBHelper.NAME)
  - Helper methods keep database internals from “leaking” into other classes
    - Return a Collection of results rather than a Cursor
      - Use Cursor internally in DBHelper class
  - SQL injection
    - Sanitise user input
  - Important when thinking about the logical next step – exposing data to other applications via a Component

Let’s have a look…

References

- http://developer.android.com/guide/topics/data/data-storage.html
- http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html
- http://developer.android.com/reference/android/database/Cursor.html