G53MDP
Mobile Device Programming

Lecture 12 – Databases and Content Providers

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);
```

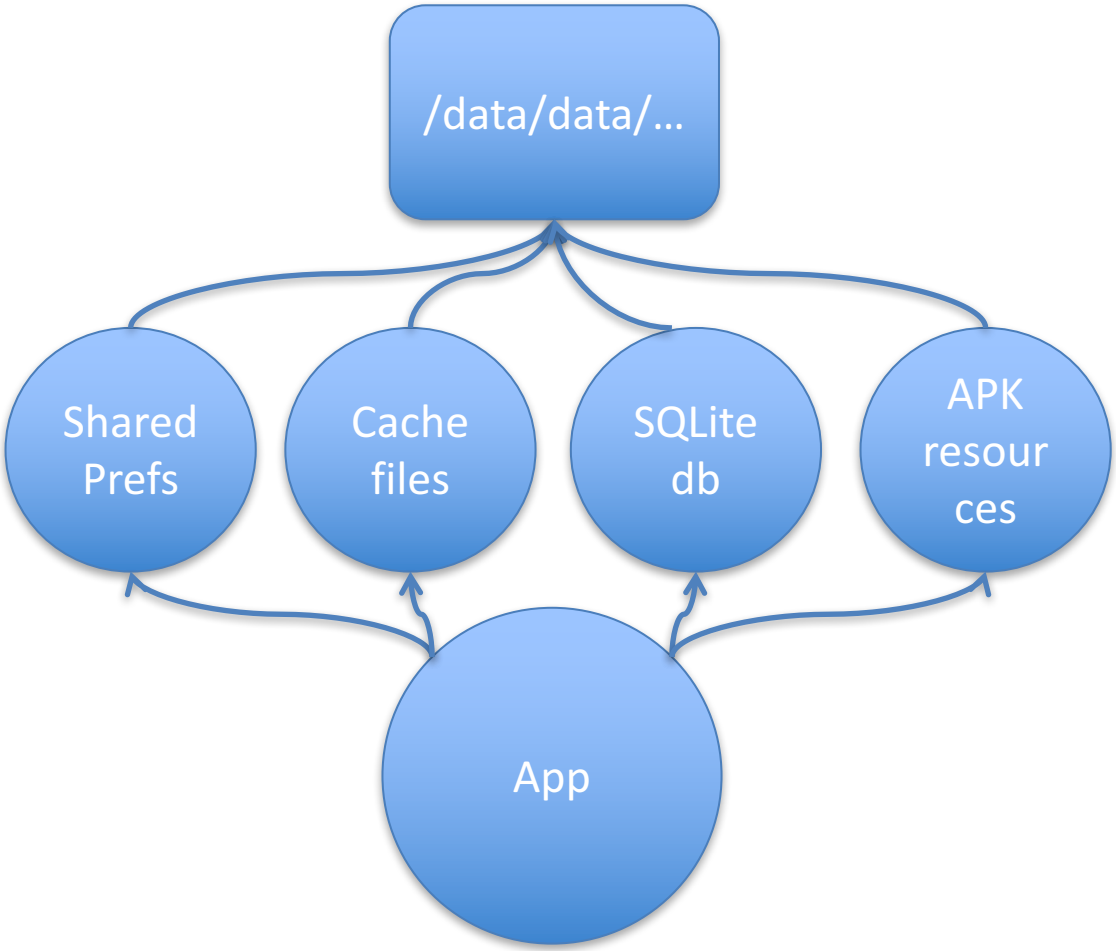
  - Relative of AsyncTask, returns to main thread to interact with UI element

Database Abstraction

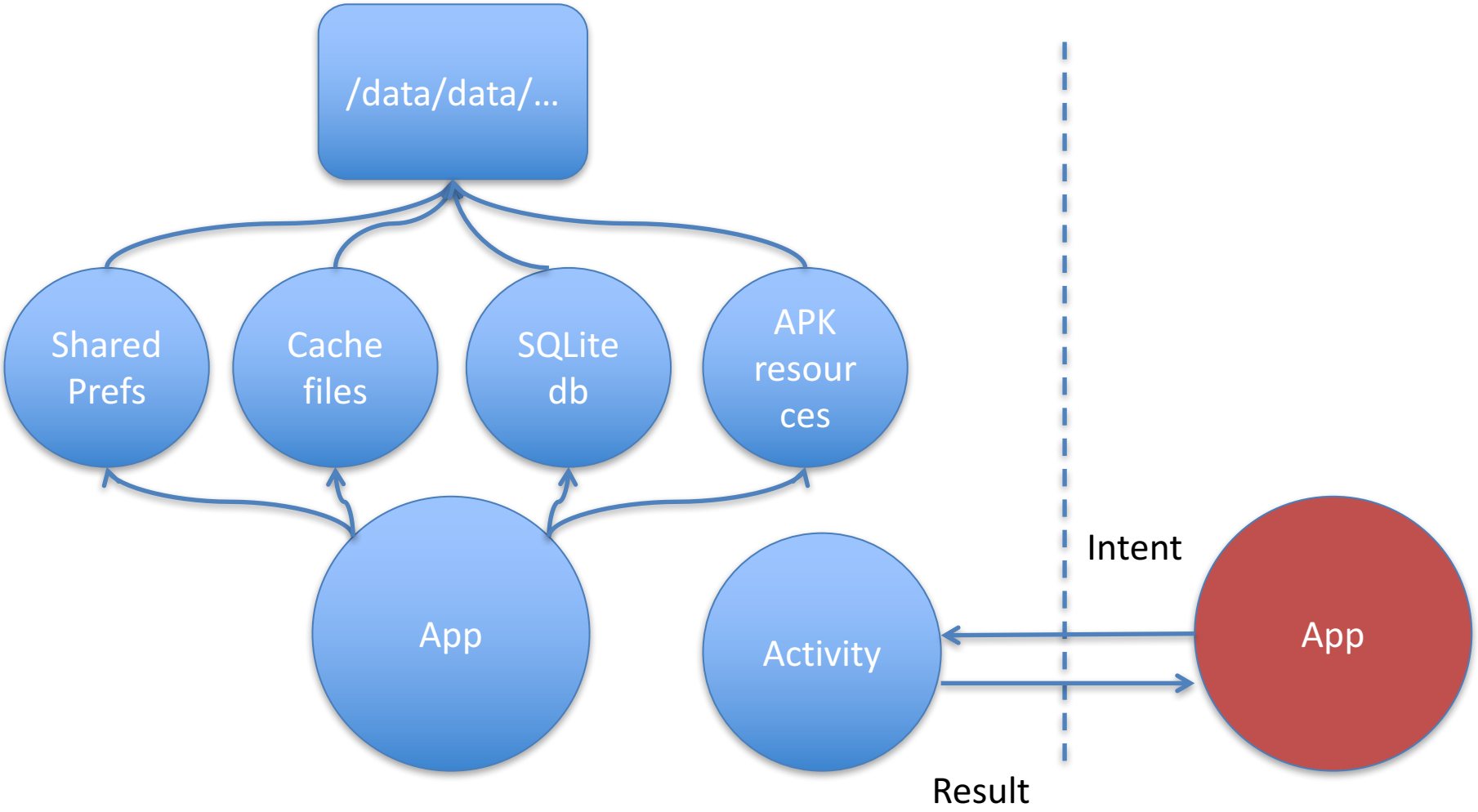
- Good software architecture
  - Separation of data model from presentation / views
- Abstraction of database architecture
  - Easier to update storage code
  - Expose column indices as static class variables
    - c.getInt(0) -> c.getInt(DBHelper.NAME)
  - Helper methods keep database internals from “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…

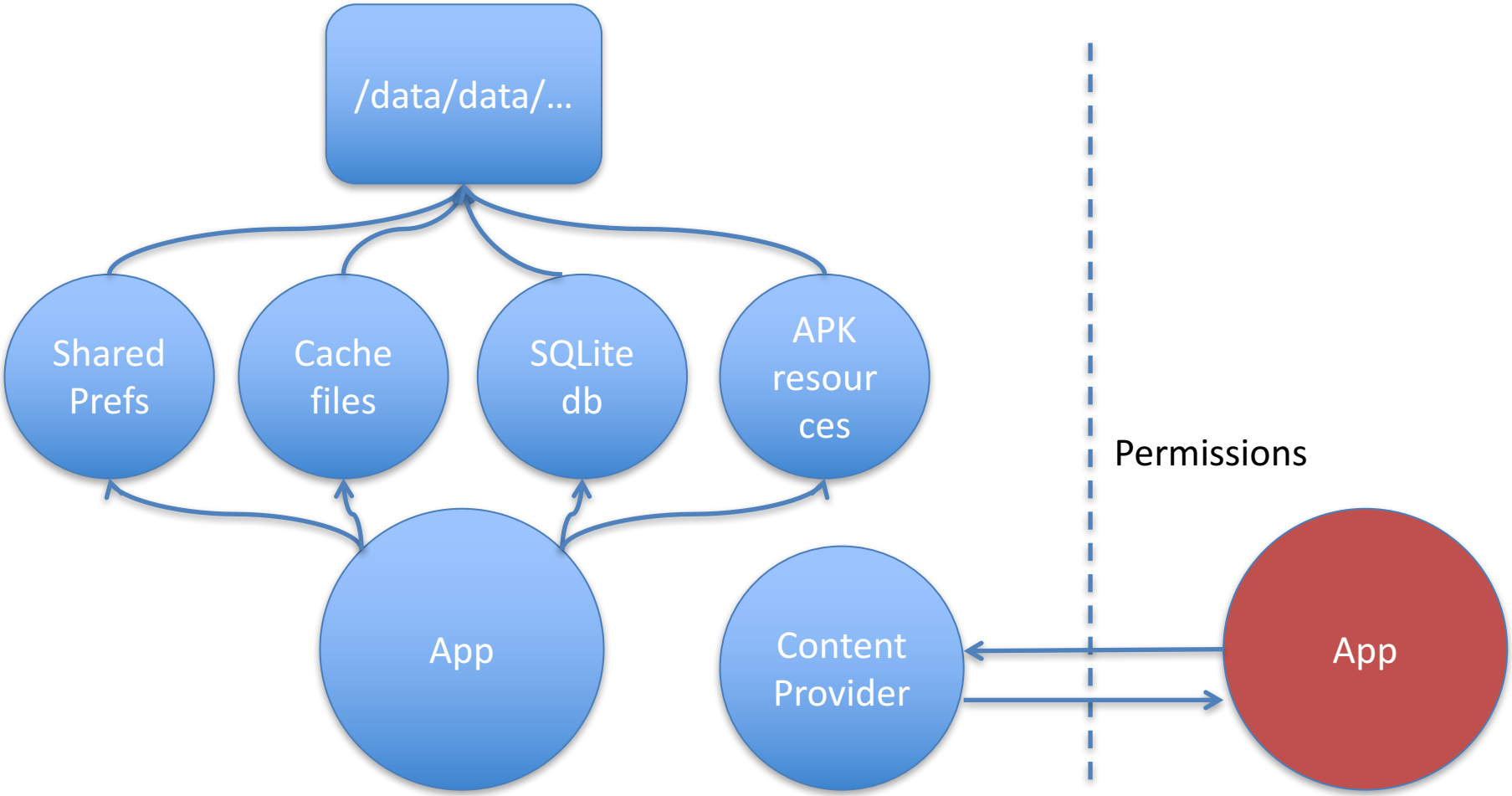
Sharing Data



Sharing Data – is this good enough?



Sharing Data – if not



ContentProvider

- Access to data is restricted to the app that owns it
  - Database is located in *internal* app-specific storage
    - Inaccessible by other applications
  - If we want other apps to access our data, or we want to access other apps' data, or we want to be notified when data has changed
- Provide or make use of a **ContentProvider**
    - Application component number 3
    - Exposes data / content to other applications in a structured manner
    - Fundamentally IPC via Binder (again) + ashmem with a well defined (database-like) interface

System ContentProviders

- ContentProviders manage data for:
  - Browser
    - Bookmarks, history
  - Call log
    - Telephone usage
  - Contacts
    - Contact data
    - WhatsApp?
  - Media
    - Media database
  - UserDictionary
    - Database for predictive spelling
  - …
- Again, recall common mobile capabilities

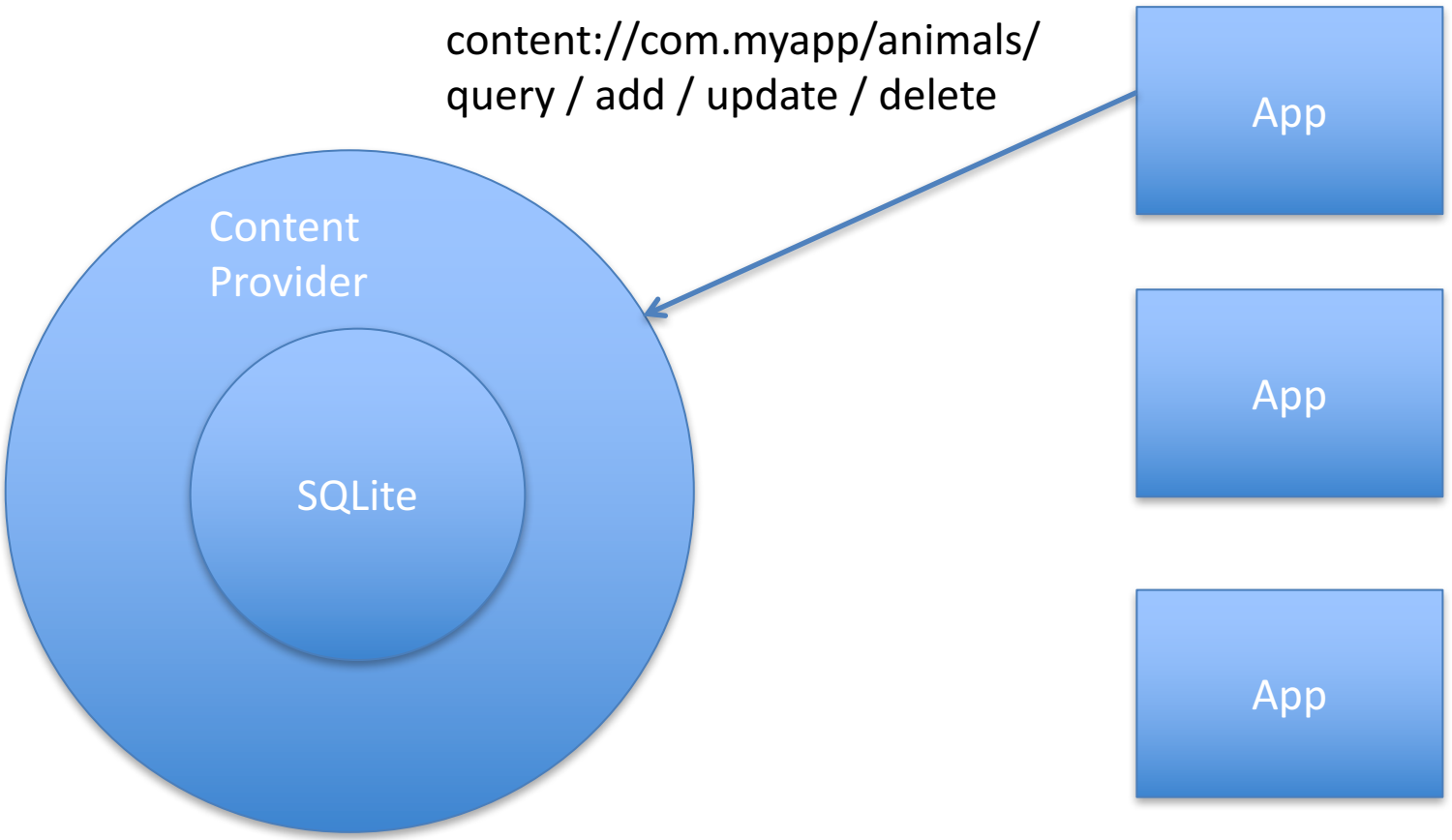
Content Providers

- Good practice even when making data available only within the application
  - Either create a new one (by sub-classing ContentProvider)
  - Or add / query data via an existing / native ContentProvider
- Assuming that spawning an Activity via Intent is not sufficient
  - Querying complex data
  - Requiring close coupling of application to data
    - c.f. Binding to Services

Data Model

- ContentProviders enforce a specific data model
- Very similar to a relational database table
  - A collection of records
  - Support for reading and writing
  - Support typical database operations
    - CRUD
- Records are stored in rows, with each column providing different data fields
  - Each record has a numeric id (in the field _ID) that uniquely identifies it
- Tables exposed via URI
  - Abstraction again
    - Can be close to or distant from underlying storage
  - Most of the “work” is specifying the abstraction / linkage

Data Model


content://com.myapp/animals/
query / add / update / delete
App
App
App
Content
Provider
SQLite

Querying a ContentProvider

- ContentResolver
  - Manages and supports ContentProvider access
    - How to service a request for content
    - Similar to ServiceManager
  - Enables ContentProviders to be used across multiple applications
  - Provides additional services such as change notification
    - Can *observe* a ContentProvider to be informed of real-time modifications
      - A new MP3 has been added to the library
      - ContentObserver
- ContentResolver cr = getContentResolver();

Querying a ContentProvider

- ContentProviders identify data sets through URIs
  - content://authority/path/id
- content
  - Data managed by a ContentProvider
- authority
  - ID for the ContentProvider (i.e. fully qualified class name, com.example.martindata)
- path
  - 0 or more segments indicating the subset of data to be requested
    - e.g. table name, or something more readable / abstracted
      - RESTful resource philosophy
- id
  - Specific record (row) being accessed

Querying a ContentProvider

- URI for searching Contacts
  - ContactsContract.Contacts.CONTENT_URI = “content://com.android.contacts/contacts/”
- ContentResolver.query(…)
  - Returns a Cursor instance for accessing results
  - Cursor is a *pointer*
    - ...to a CursorWindow
    - A read-only reference to shared memory allocated by ashmem, retrieved via Binder
      - .close()...
    - Max CursorWindow size is 2Mb
      - Is this big enough? Why?

Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)

Contacts

- To access / modify Contacts, requires a Permission
  - android.permission.READ_CONTACTS
  - android.permission.WRITE_CONTACTS
- Contacts has three components
  - Data
    - Rows (mime-typed) that can hold personal information
  - RawContacts
    - A contact for a given person from a given system
      - Gmail contact, Facebook contact etc
      - Associated with Data entries
  - Contacts
    - Aggregated RawContacts
      - Single view of a “person”

```
Cursor c =  cr.query(ContactsContract.Contacts.CONTENT_URI, new String[]
    { ContactsContract.Contacts.DISPLAY_NAME },
    null, null, null);
```

References

- http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html
- http://developer.android.com/reference/android/database/Cursor.html
- http://developer.android.com/guide/topics/providers/content-providers.html
- http://developer.android.com/guide/components/fundamentals.html