G54MDP
Mobile Device Programming

Lecture 11 – Content Providers

git

- http://git-scm.com/

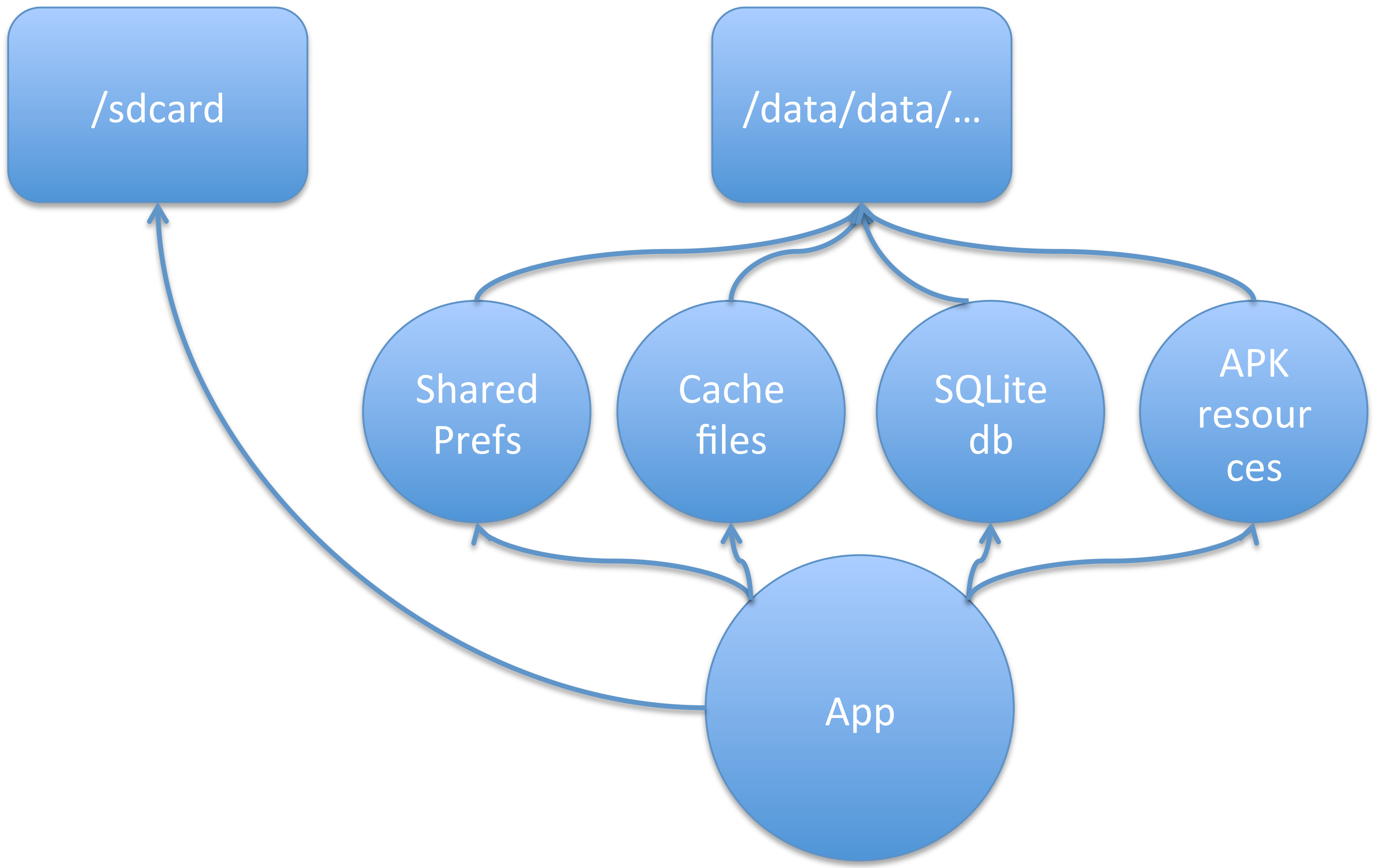
- https://github.com/mdf/g54mdp

- git clone https://github.com/mdf/g54mdp.git
- git pull

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
  - Shared contact lists, backups
  - SyncAdapter

Logical Data Storage on Android



Database Lifecycle

- class DBHelper extends SQLiteOpenHelper
- onCreate(SQLiteDatabase db)
  - Called **once** when the application is **first** run
  - CREATE TABLE
- dbHelper = new DBHelper(this, "martinDB", null, 1);
- onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
  - Called when the version number is changed
  - I.e. if the application is updated
    - Opportunity to DROP and re-CREATE a table with an updated schema

Querying a Database

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

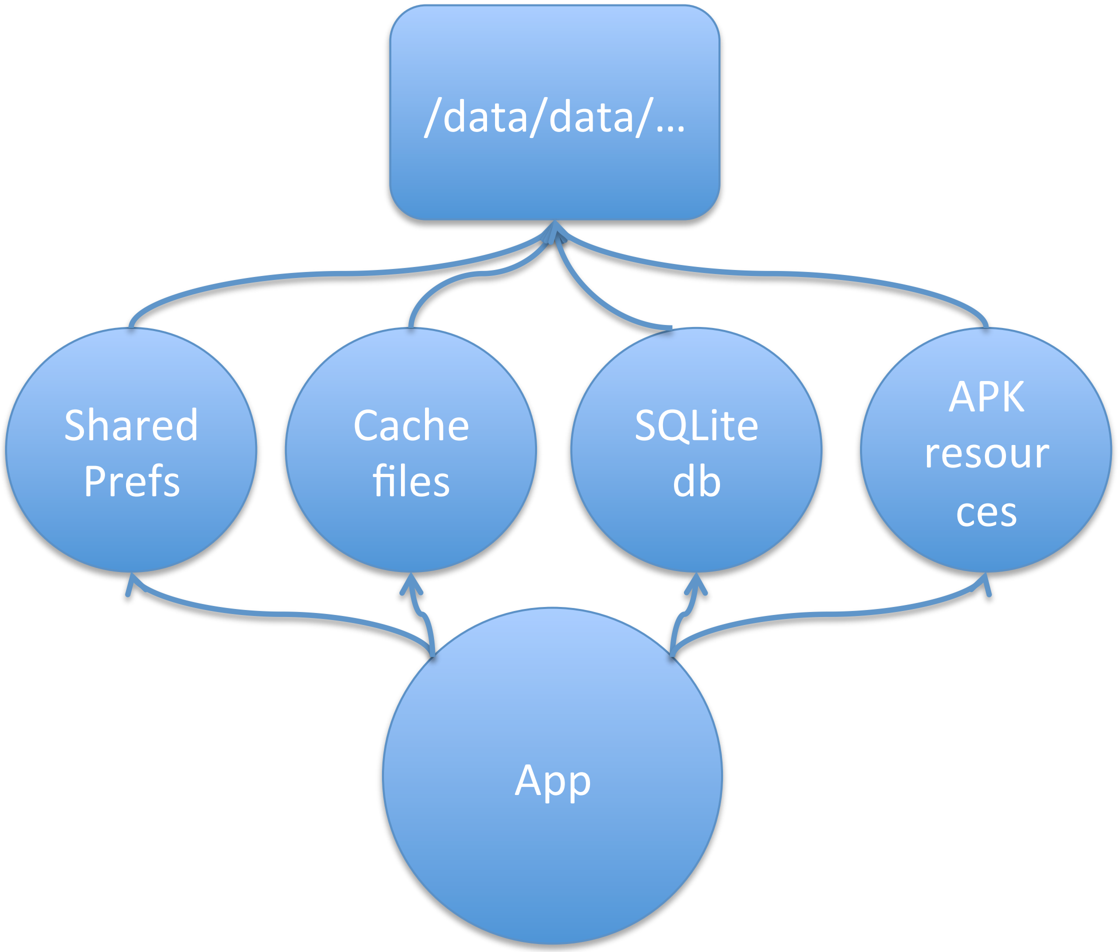
Querying a Database

```
Cursor c = db.query("myList", new String[] { "id",
"name" }, null, null, null, null, null);
if(c.moveToFirst())
{
    int id = c.getInt(0);
    String name = c.getString(1);
```

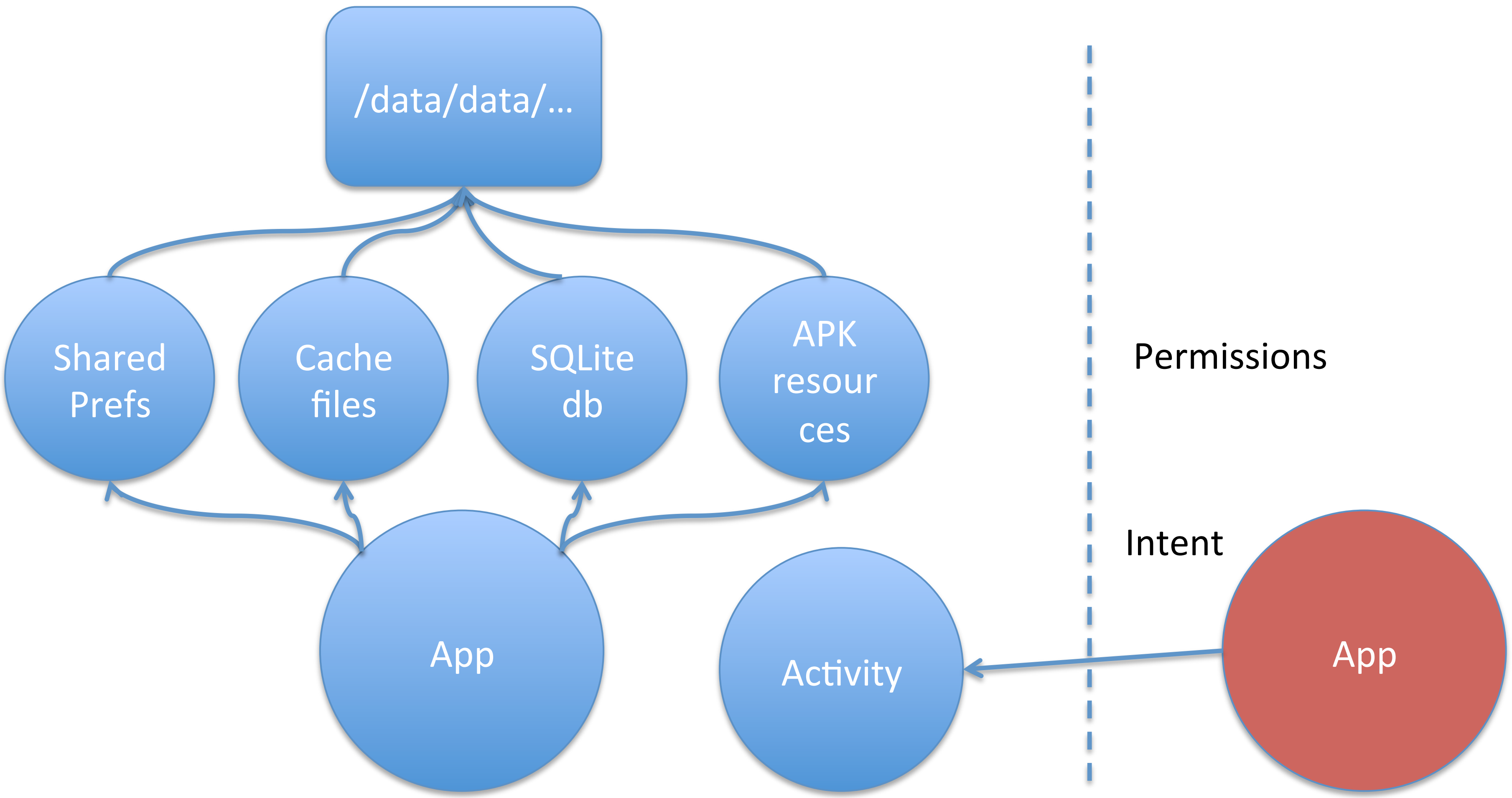
Database Abstraction

- 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

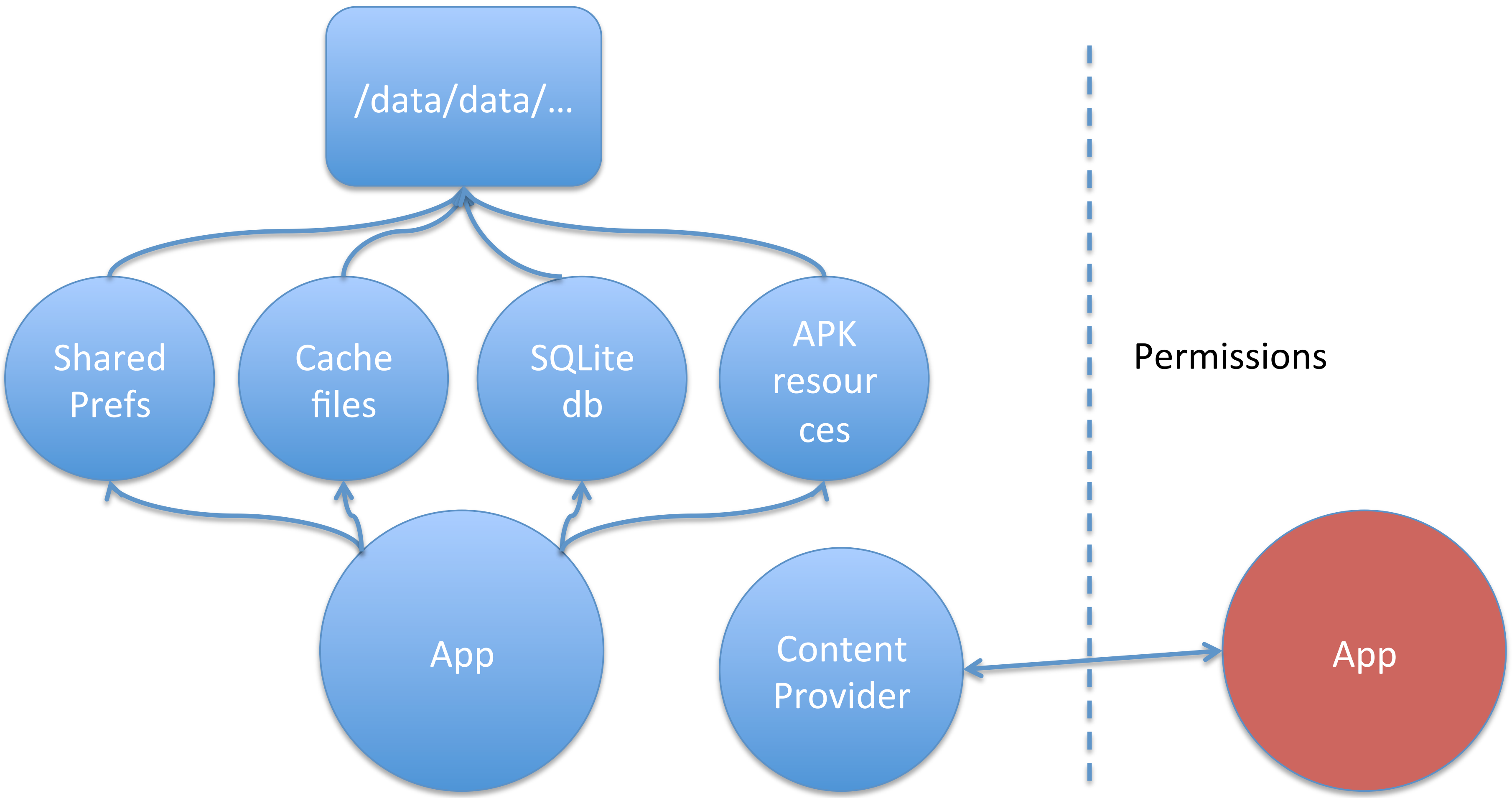
Sharing Data



Sharing Data – is this good enough?



Sharing Data – if not



ContentProvider

- Access to data is restricted to the app that owns it
  - Remember where the database file is?
  - If we want other apps to access our data, or we want to access other apps' data
  - …we need to provide or make use of a ContentProvider
    - Component number **3**
    - Exposes data / content to other applications in a structured manner

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

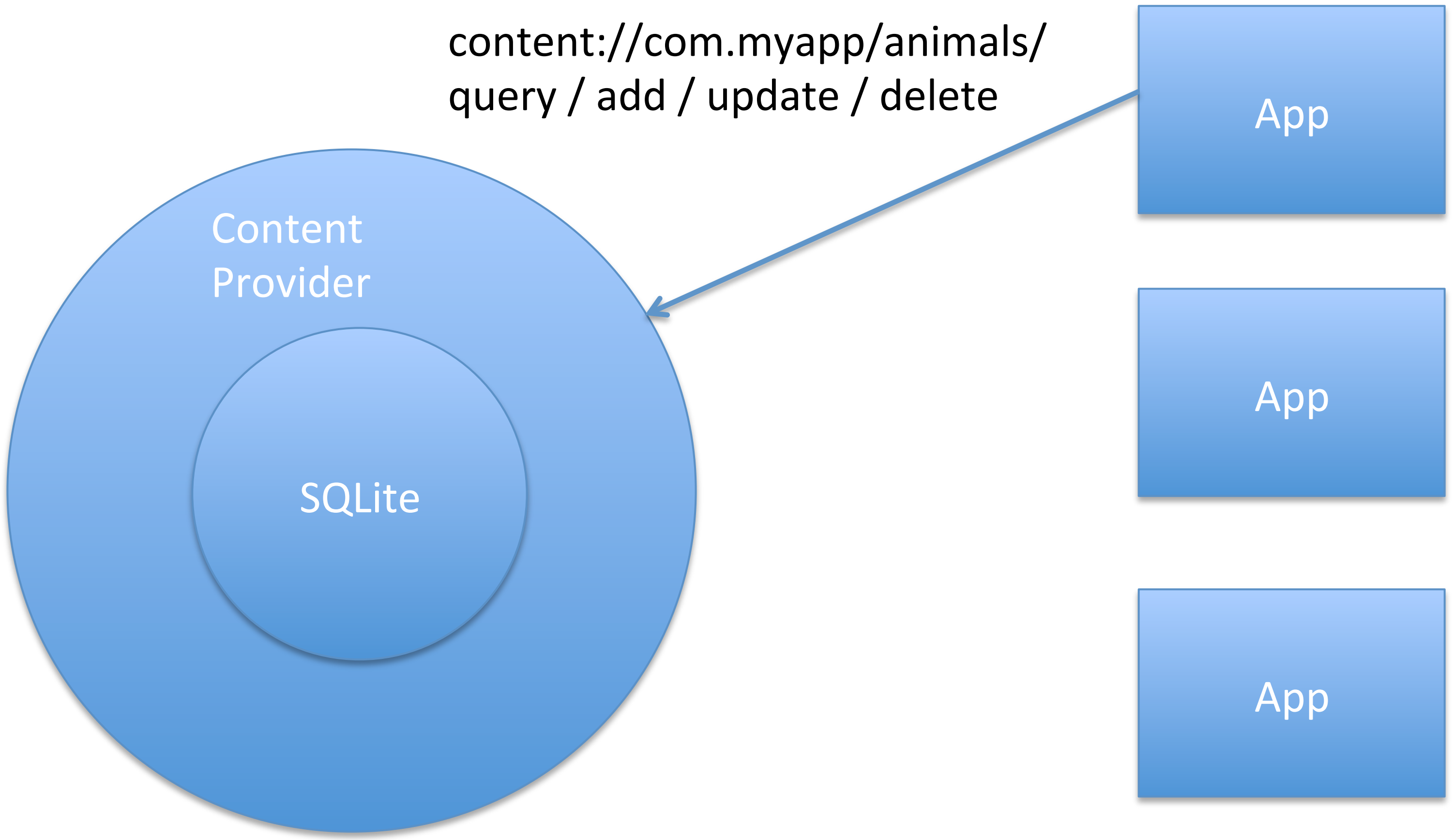
Content Providers

- 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

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
- 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
  - Most of the “work” is specifying the abstraction

Data Model


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

Querying a ContentProvider

- ContentResolver
  - Manages and supports ContentProvider access
  - 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
- 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
- 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 query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)
- Cursor c =

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

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 to a person

Modifying a ContentProvider

- Uri insert(Uri url, ContentValues values)
- int update(Uri uri, ContentValues values, String where, String[] selectionArgs)
- Uri
  - The table that we wish to update / insert
- ContentValues
  - Values for the new row
  - Key/value pairs
    - Key is the column name
- where
  - SQL WHERE clause

Creating a Content Provider

- Implement a storage system for the data
  - Structured data / SQLite
    - Values, binary blobs up to 64k
    - Database
  - Large binary blobs
    - Files
  - Photos / media manager
- Implement a ContentProvider
  - Implement required methods
  - query, add, update, insert etc
  - onCreate
  - getType
    - What type of data are we providing?
  - ParcelFileDescripter openFile()
- Tell Android we are a provider
  - Declare in the AndroidManifest

Contract

- Defines metadata pertaining to the provider
- Constant definitions that are exposed to developers via a compiled .jar file
  – Authority
    - Which app is responsible for this data
  – URI
  – Meta-data types
  – Column names
    - Abstraction of database architecture

URI Matching

- All of these methods (except onCreate()) take a URI as the first parameter
  - The object will need to parse it to some extent to know what to return, insert or update
  - Android provides android.content.UriMatcher to simplify this
    - Provides mapping between abstraction of contract class to concrete db implementation
  - Does the calling application want all data from a table, or just a row, or a specific table?

Permissions

- By default our new provider requires no permissions
  - Can be accessed read/write by all other applications
  - Specify required permissions in the manifest entry
    - Can specify URI path-level permissions for fine grained access control
      - You can read the names of my contacts, but not see their email addresses
    - Can grant temporary permission to access certain URIs via code

"Access to the mail should be protected by permissions, since this is sensitive user data. However, if a URI to an image attachment is given to an image viewer, that image viewer will not have permission to open the attachment since it has no reason to hold a permission to access all e-mail."

Let’s have a look…

Network

- One last type of data storage
  - Get it off the phone, and into the cloud
- Implement a SyncAdapter
  - Appears in the “Accounts and Sync” menu in the OS
  - Synchronizes a local database / content provider with a remote server
  - Make use of a Service to push data in the background
- http://www.youtube.com/watch?feature=player_embedded&v=xHXn3Kg2IQE

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
- http://developer.android.com/guide/topics/providers/content-provider-basics.html
- http://developer.android.com/training/sync-adapters/creating-sync-adapter.html