Temari

1) Introducció
2) Framework Android
3) Projectes Android i Android SDK
4) Activity
5) Fragments, Views i ListViews
6) Intents
7) Layouts i Custom Views
8) Resources i Themes
9) Dialogs, Menus i WebView
10) Persistència de dades
11) Tasques en Background i internet
12) SQLite i content providers
13) Notificacions

12 – SQLite i content Providers

- SQLite
- Cursors
- Content Providers
- Native android Content Providers
- Custom Content Provider

12 – SQLite i Content Providers

SQLite

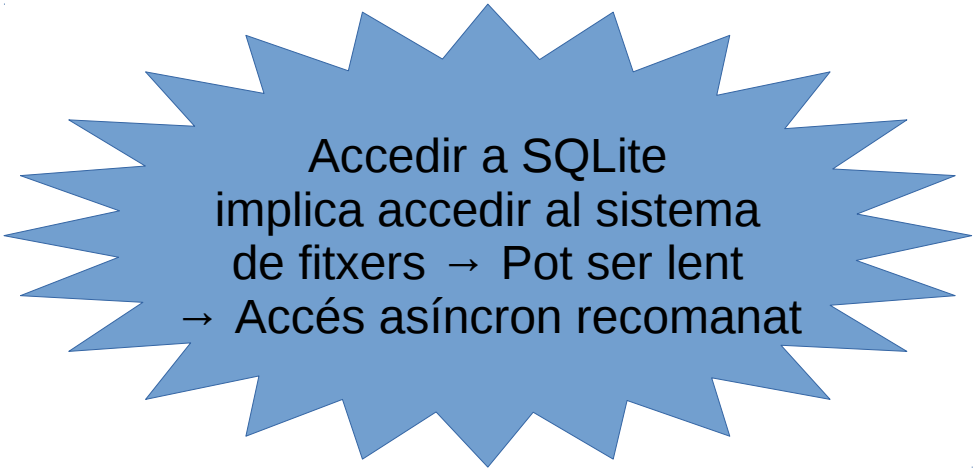
Base de dades SQL, OpenSource, i molt lleugera (250kB). Permet **transaccions**, **consultes sql** i **prepared statements**.

Inclosa i gestionada per Android

A **android.database** i **android.database.sqlite** hi ha les classes

Les BDD creades per a l'aplicació es desen a

/data/data/<app-name>/databases/<database>



SQLite: Crear i actualitzar la BDD

Cal extendre la classe **SQLiteOpenHelper**:

- Crear un cosntructor que cridi al **super()** especificant el context, la BDD i la versio.
- Sobreescriure el mètode **onCreate()** → executar les consultes per crear les taules
- Sobreescriure el mètode **onUpgrade()** → especificar les consultes per actualitzar les taules de la BDD

La convenció és que la Clau primària sigui una Columna anomenada **_id**

Si la BDD té vàries taules, fer una classe que actualitzi i crei cadascuna de les taules, Per mantenir el codi llegible

SQLite: Crear i actualitzar la BDD

```
public MySQLiteHelper(Context context) {
  super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase database) {
  database.execSQL(DATABASE_CREATE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
  Log.w(MySQLiteHelper.class.getName(),
      "Upgrading database from version " + oldVersion + " to "
          + newVersion + ", which will destroy all old data");
  db.execSQL("DROP TABLE IF EXISTS " + TABLE_COMMENTS);
  onCreate(db);
}
```


SQLite: Executar Queries

La classe **SQLiteOpenHelper** ofereix els mètodes:

- getReadableDatabase()
- getWritableDatabase()

Que retornen un **SQLiteDatabase**. Sobre aquest podem executar:

- insert() → per insertar una fila a la BDD
- update() → per actualitzar files a la BDD
- delete() → per eliminar files de la BDD
- rawQuery() → consultes amb SQL directament (no recomanat)
- query() → consultes de forma estructurada

Addicionalment la classe **SQLiteQueryBuilder** facilita construir queries

SQLite: Executar Queries

Exemples:

rawQuery()

```
Cursor cursor = getReadableDatabase().
  rawQuery("select * from todo where _id = ?", new String[] { id });
```

insert()

```
SQLiteDatabase db = this.getWritableDatabase();
ContentValues contentValues = new ContentValues();
contentValues.put("name", name);
contentValues.put("phone", phone);
contentValues.put("email", email);
contentValues.put("street", street);
contentValues.put("place", place);
db.insert("contacts", null, contentValues);
return true;
```

query() → molts paràmetres, mirar la documentació!

SQLite: Cursor

Les queries retornen **un Cursor**

El cursor apunta en un determinat moment a una fila del resultat → més eficient, tot i que hi hagi molts resultats

Provem-ho!

Té una sèrie de mètodes útils:

- **getCount**() → numero total de files
- **moveToFirst**() → s'ha dexecutar abans de començar a llegir!
- **moveToNext**()
- **IsAfterLast()** → indica que ja hem passat la darrera fila
- **get*(int columnIndex)** → getLong(), getString()...
- **getColumnIndex**(**String columnName**)
- ...
- **close**()

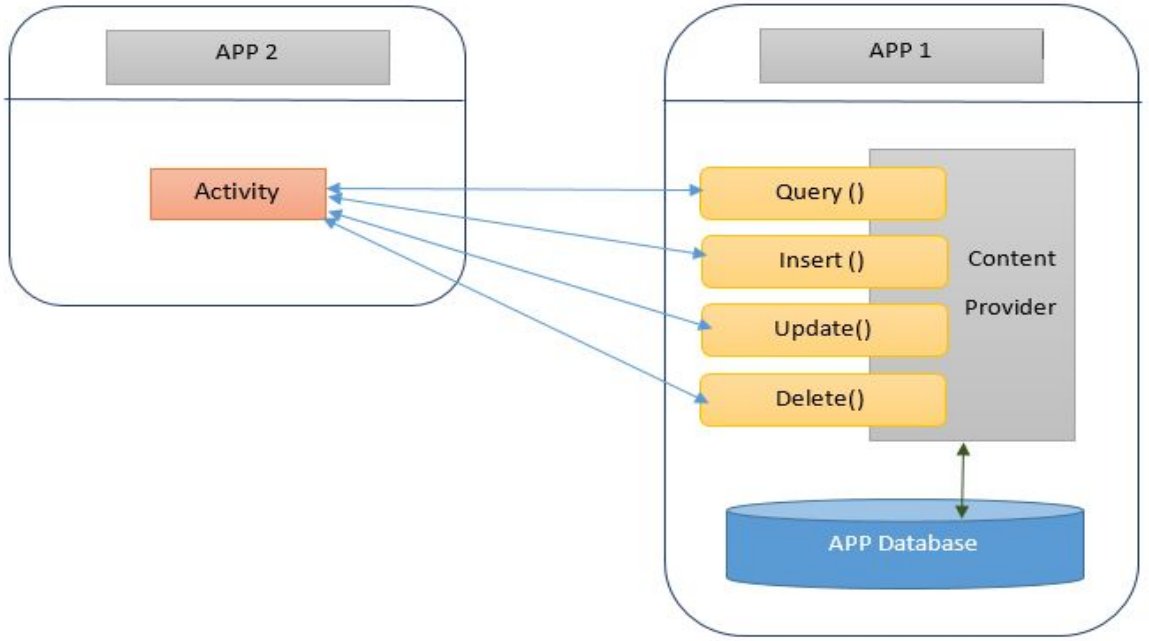
SEMPRE cal tancar-lo!

ContentProviders

Per fer accessibles dades des de vàries aplicacions, existeixen els **ContentProviders**

S'accedeix al contingut mitjançant **URIs**, que comencen per **content://**

Permeten realitzar operacions **CRUD**



Ha d'estar al manifest!

ContentProviders: Native ContentProviders

Android n'ofereix per defecte per accedir a informació trasnversal:

- Calendari
- Contactes
- Multimedia
- SMS/MMS
- ...

Android proporciona la classe ContentResolver:

```
ContentResolver cr = getContentResolver ();
Cursor cursor = cr.query (People.CONTENT_URI , null,
   PEOPLE.NAME " = '" + name + "'", null, null);
if (cursor.moveToFirst())
{
  String name = cursor.getString (cursor.getColumnIndex
       (People.NAME));
  String phone = cursor.getString (cursor.getColumnIndex
       (People.NUMBER));
}
```


ContentProviders: crear un propi

Tot va amb URIs
UriMatcher
ens ajuda!

- Extendre la classe ContentProvider
- Crear constants amb la URI, i els camps de les dades
- Sobreescriure els mètodes
  - onCreate() → aquí, si darrera hi ha una BDD, crearem el Helper
  - query(), insert(), delete() i update() → si alguna no està implementada o permesa, llençar **UnsupportedOperationException()**
  - GetType() → retorna el mimeType de la resposta, donada una URI
- Registrar-lo al AndroidManifest i tenir en compte el **android:exported** (per defecte a false des de la 4.2, abans true)
- NO és **Thread safe!!** → potser fer alguns mètodes **synchronized**