SQLite Database

SQLite Database

- In many occasions you will need a way to maintain your application data, which can be later accessed or altered.
- In that situation you possibly can make use of SQLiteDatabase in your application.

SQLite Database

- Android SQLite Database :
- Why SQLite for Android?

- SQLite is really a quick and compact android database technology which incorporates SQL syntax to create queries and also handle data.

- Android SDK by itself provides the SQLite support which without doubt making the setup as well as utilization process within our applications with no trouble

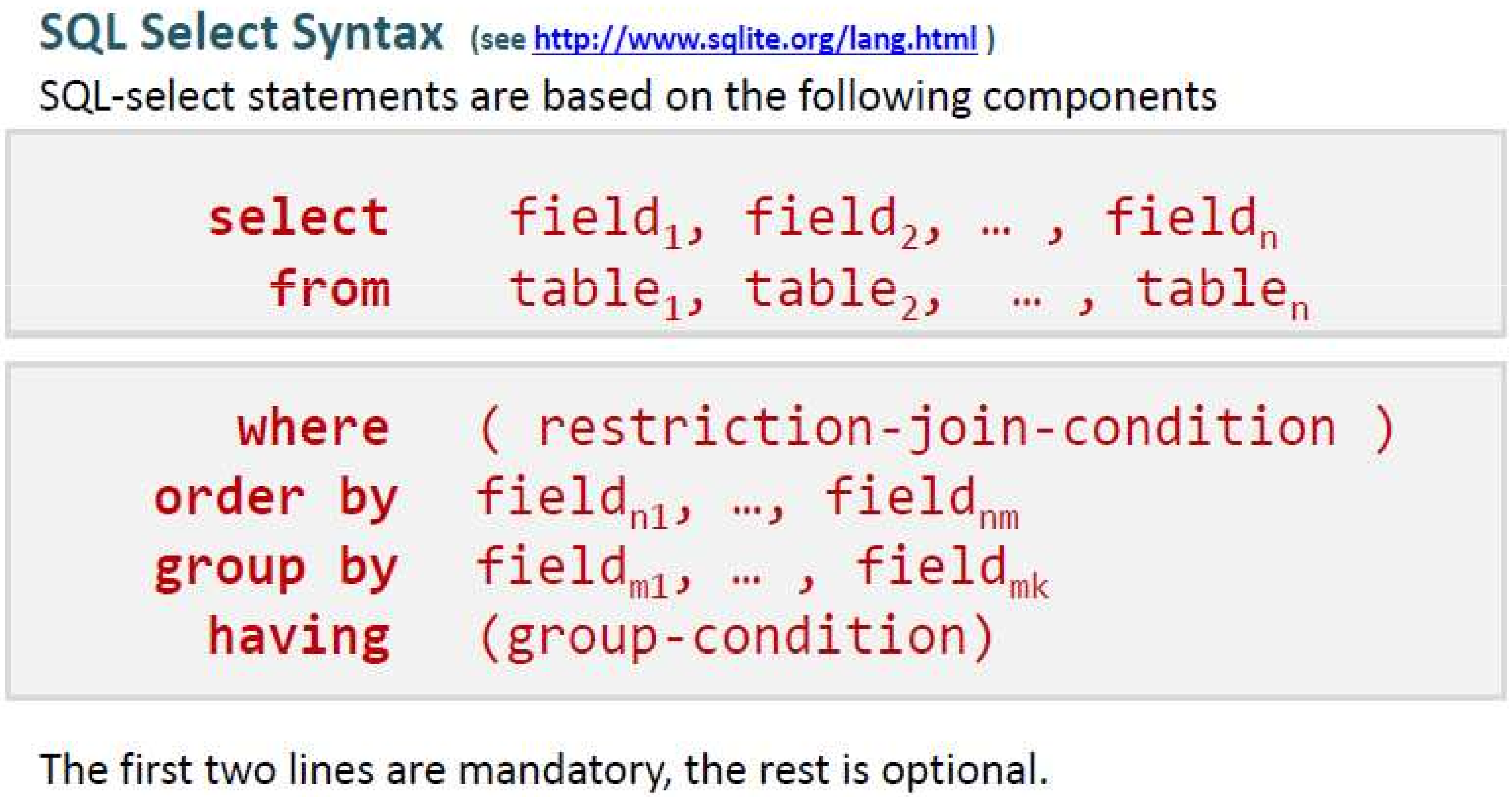
SQLite Database

- **Using SQL databases in Android.**
- Android (as well as iPhoneOS) uses an embedded standalone program called **sqlite3 which can be used to:**
- create a database,
- define SQL tables,
- queries,
- views,
- triggers
- Insert rows,
- delete rows,
- change rows,
- run queries and
- administer a SQLitedatabase file.

SQLite Database

- **Using SQLite**
- 1.SQLiteimplements most of the SQL-92standard for SQL.
- 2.It allows most complex queries
- 3.SQLITE *does not implement referential integrity constraints through the foreign keyconstraint model.*
- 4.Instead of assigning a type to an entire column, types are assigned to individual values. This is similar to the *Varian ttype in Visual Basic*.
- 5.Therefore it is possible to insert a string into numeric column and so on.
- 6.Documentation on SQLITE available at http://www.sqlite.org/sqlite.htmlGood GUI tool for SQLITE available at: http://sqliteadmin.orbmu2k.de/

Syntax



SQL Select Syntax (see http://www.sqlite.org/lang.html)

SQL-select statements are based on the following components

```
select    field_1, field_2, … , field_n
  from    table_1, table_2,  … , table_n
```

```
   where  ( restriction-join-condition )
order by  field_n1, …, field_nm
group by  field_m1, … , field_mk
  having  (group-condition)
```

The first two lines are mandatory, the rest is optional.

Cont..

SQL Select Syntax (see http://www.sqlite.org/lang.html)

Examples

```
   select    LastName, cellPhone
     from    ClientTable
    where    state = ‘Ohio’
 order by    LastName
```

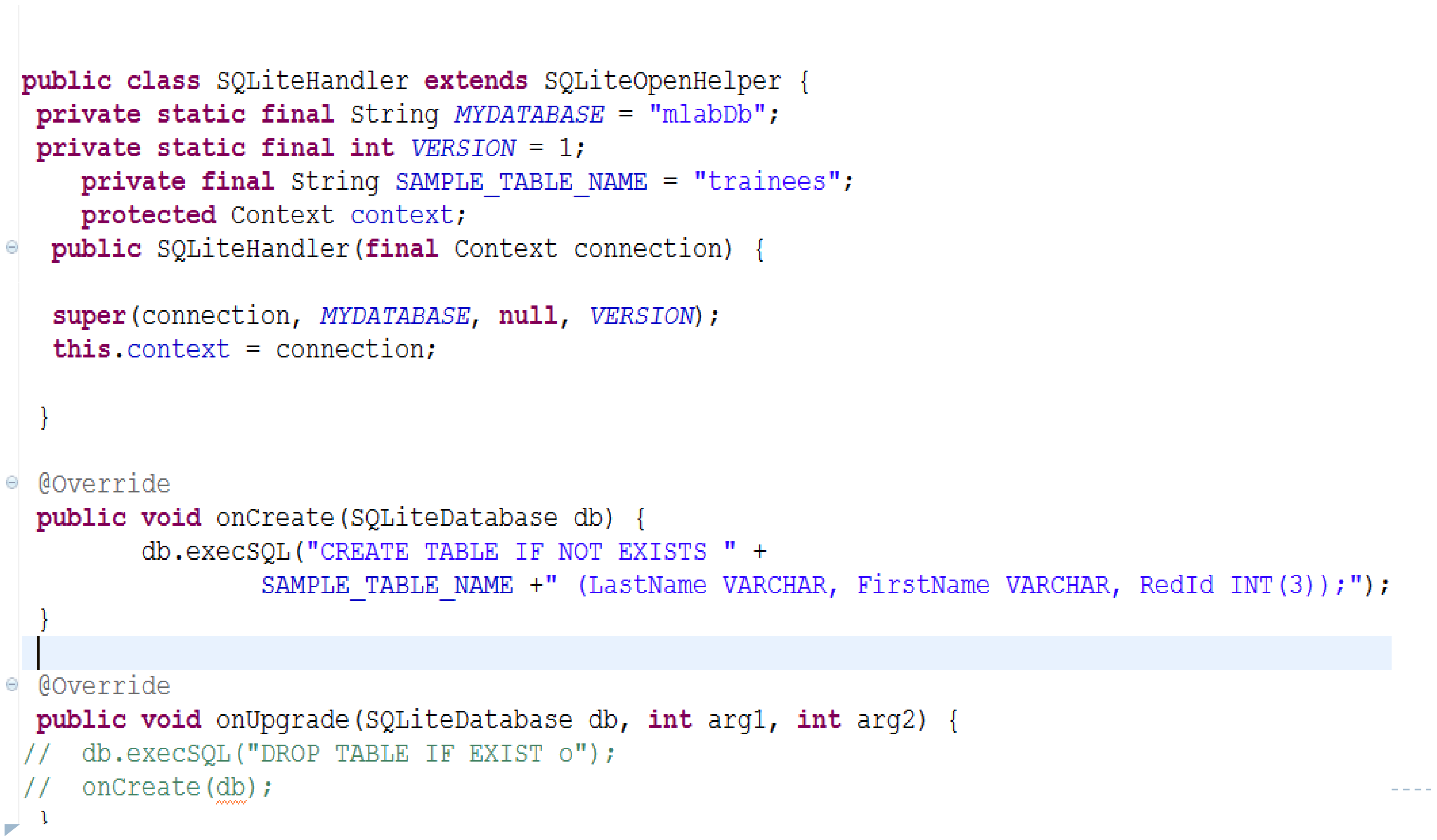
```
   select    city, count(*) as TotalClients
     from    ClientTable
 group by    city
```

SQLiteHelper

- A helper class to manage database creation and version management.
- You create a subclass implementing onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) .
- This class takes care of opening the database if it exists, creating it if it does not, and upgrading it as necessary.
- Transactions are used to make sure the database is always in a sensible state.

Get Started

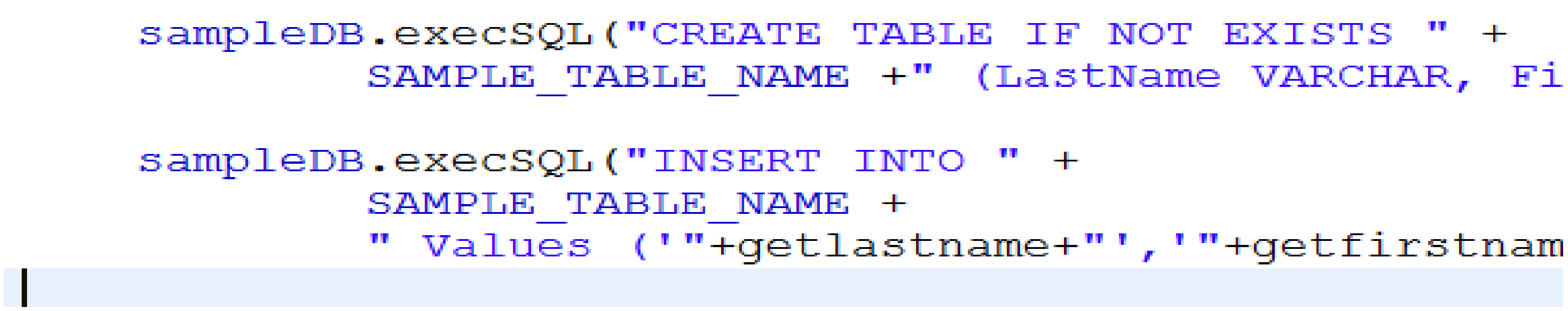
- SQLiteHelper



```java
public class SQLiteHandler extends SQLiteOpenHelper {
 private static final String MYDATABASE = "mlabDb";
 private static final int VERSION = 1;
    private final String SAMPLE_TABLE_NAME = "trainees";
    protected Context context;
  public SQLiteHandler(final Context connection) {

  super(connection, MYDATABASE, null, VERSION);
  this.context = connection;

 }

 @Override
 public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE IF NOT EXISTS " +
                SAMPLE_TABLE_NAME +" (LastName VARCHAR, FirstName VARCHAR, RedId INT(3));");
 }

 @Override
 public void onUpgrade(SQLiteDatabase db, int arg1, int arg2) {
//   db.execSQL("DROP TABLE IF EXIST o");
//   onCreate(db);
 }
```

INSERT

- Before inserting data in a table, you have to create the table first.



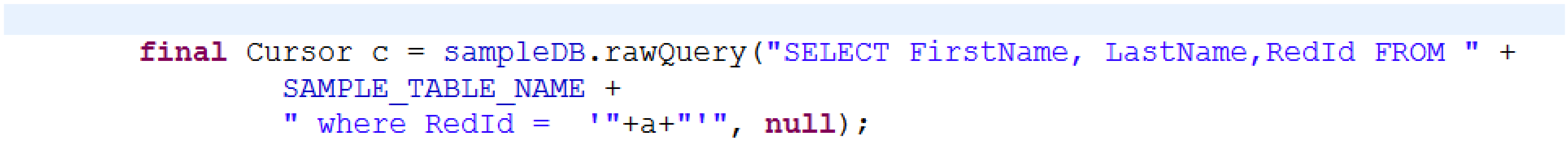
```
sampleDB.execSQL("CREATE TABLE IF NOT EXISTS " +
        SAMPLE_TABLE_NAME +" (LastName VARCHAR, Fi

sampleDB.execSQL("INSERT INTO " +
        SAMPLE_TABLE_NAME +
        " Values ('"+getlastname+"','"+getfirstnam
```

Comments

- The field **recID is defined as PRIMARY KEY of the table.**
- The database data types are very simple, for instance we will use: ***text,varchar, integer, float, numeric, date, time, timestamp, blob, boolean, and so on.***
- In general, any well-formed SQL action command (insert, delete, update, create, drop, alter, etc.) could be framed inside an **execSQL(…) method.**
- You *should make the call to execSQL inside of a try-catch-finally block. Be aware of potential* ***SQLiteExceptionsituations thrown by the method.***

Selecting Data

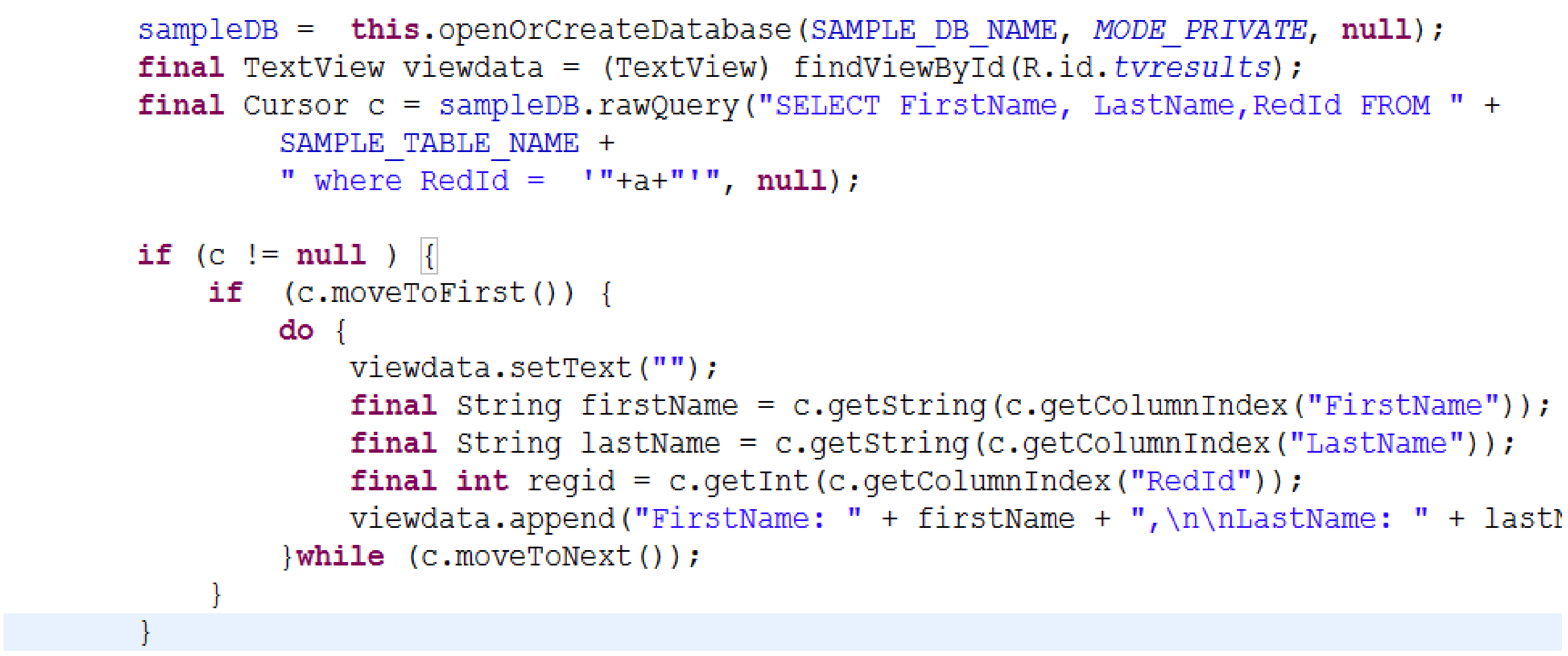
- This follows the SELECT SQL syntax,



```
    final Cursor c = sampleDB.rawQuery("SELECT FirstName, LastName,RedId FROM " +
            SAMPLE_TABLE_NAME +
            " where RedId =  '"+a+"'", null);
```

SELECT Code

- This code places data in textview defined in XML



```java
sampleDB =  this.openOrCreateDatabase(SAMPLE_DB_NAME, MODE_PRIVATE, null);
final TextView viewdata = (TextView) findViewById(R.id.tvresults);
final Cursor c = sampleDB.rawQuery("SELECT FirstName, LastName,RedId FROM " +
        SAMPLE_TABLE_NAME +
        " where RedId =  '"+a+"'", null);

if (c != null ) {
    if  (c.moveToFirst()) {
        do {
            viewdata.setText("");
            final String firstName = c.getString(c.getColumnIndex("FirstName"));
            final String lastName = c.getString(c.getColumnIndex("LastName"));
            final int regid = c.getInt(c.getColumnIndex("RedId"));
            viewdata.append("FirstName: " + firstName + ",\n\nLastName: " + last
        }while (c.moveToNext());
    }
}
```

Content Providers

- Content providers manage access to a structured set of data.
- They summarize the data, and provide mechanisms for accessing the information.
- Content providers are the standard interface that connects data in one process with code running in another process.

Content Providers

- When you want to access data in a content provider, you use the ContentResolver object in your application's Context to communicate with the provider as a client.

- The ContentResolver object communicates with the provider object, an instance of a class that implements ContentProvider.

- The provider object receives data requests from clients, performs the requested action, and returns the results.

Content Providers

- Android itself includes content providers that manage data such as audio, video, images, and personal contact information.

- You can see some of them listed in the reference documentation for the android.provider.package.

- With some restrictions, these providers are accessible to any Android application.

Content Providers

- **Content Provider Basics**
  - How to access data in a content provider when the data is organized in tables.
- **Creating a Content Provider**
  - How to create your own content provider.
- **Calendar Provider**
  - How to access the Calendar Provider that is part of the Android platform. – using SQLite
- **Contacts Provider**
  - How to access the Contacts Provider that is part of the Android platform

Content Providers

- The following are some of the content provider uris.

- content://sms/inbox URI messages from the inbox
- *content://media/internal/images* URI return the list of all internal images on the device.
- *content://contacts/people/* URI return the list of all contact names on the device.
- *content://contacts/people/45* URI return the single result row, the contact with ID=45.
- *Find Example Attached*

Assignment

- Create a content provider to view phone contacts.