SCIENCES SORBONNE UNIVERSITÉ

Content Providers

Renaultᵃlrde.epita.fr

Inter-processes communication

How to exchange information between processes?

Send Intents

- fill the extra with a lot of informations
  - **Management can be hard**

Share information through database

- Not available for inter-process communication

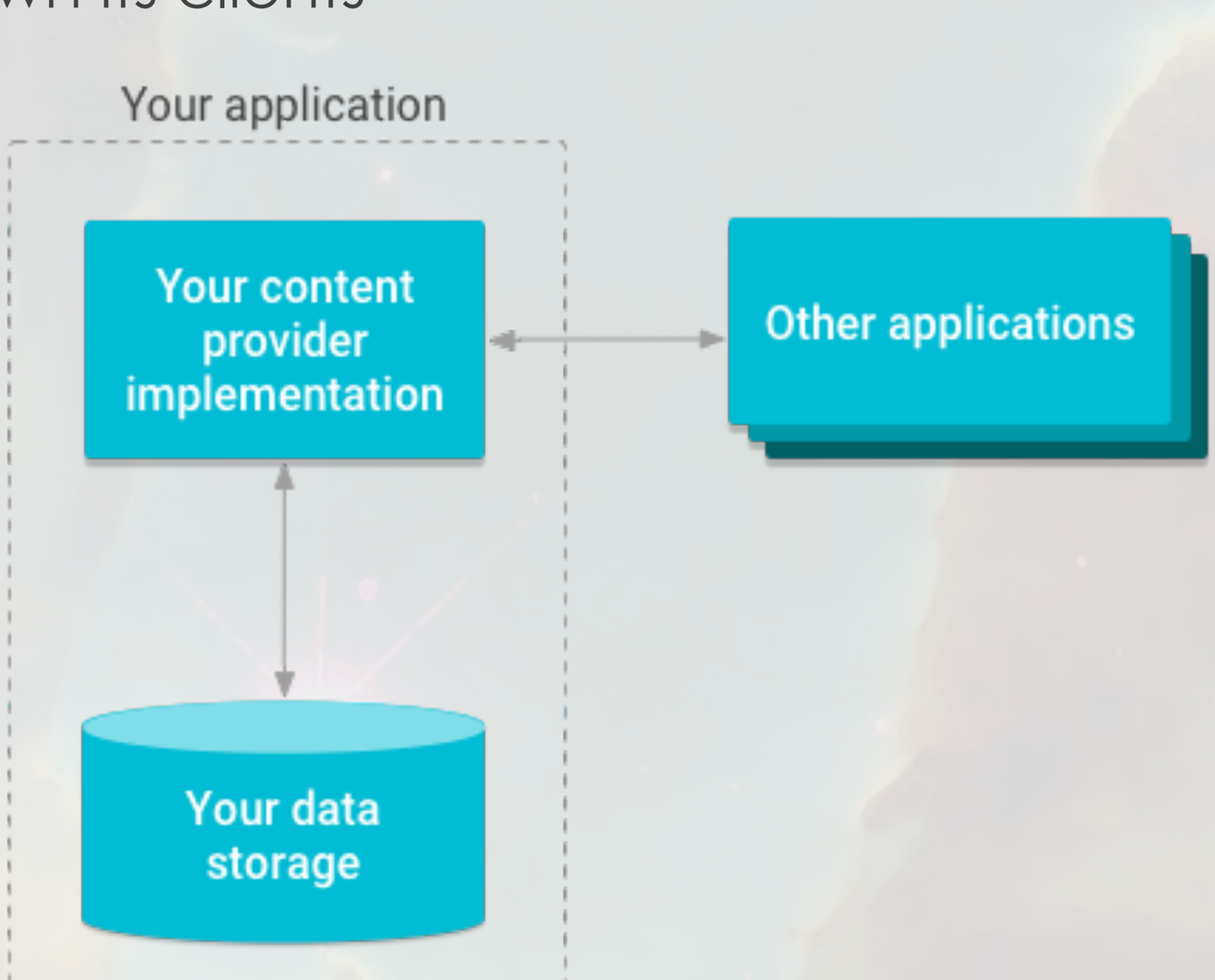
Use files

- Read / Write must be synchronized
- The format must be fixed
- cannot restrict to a specific application

ContentProviders

A provider

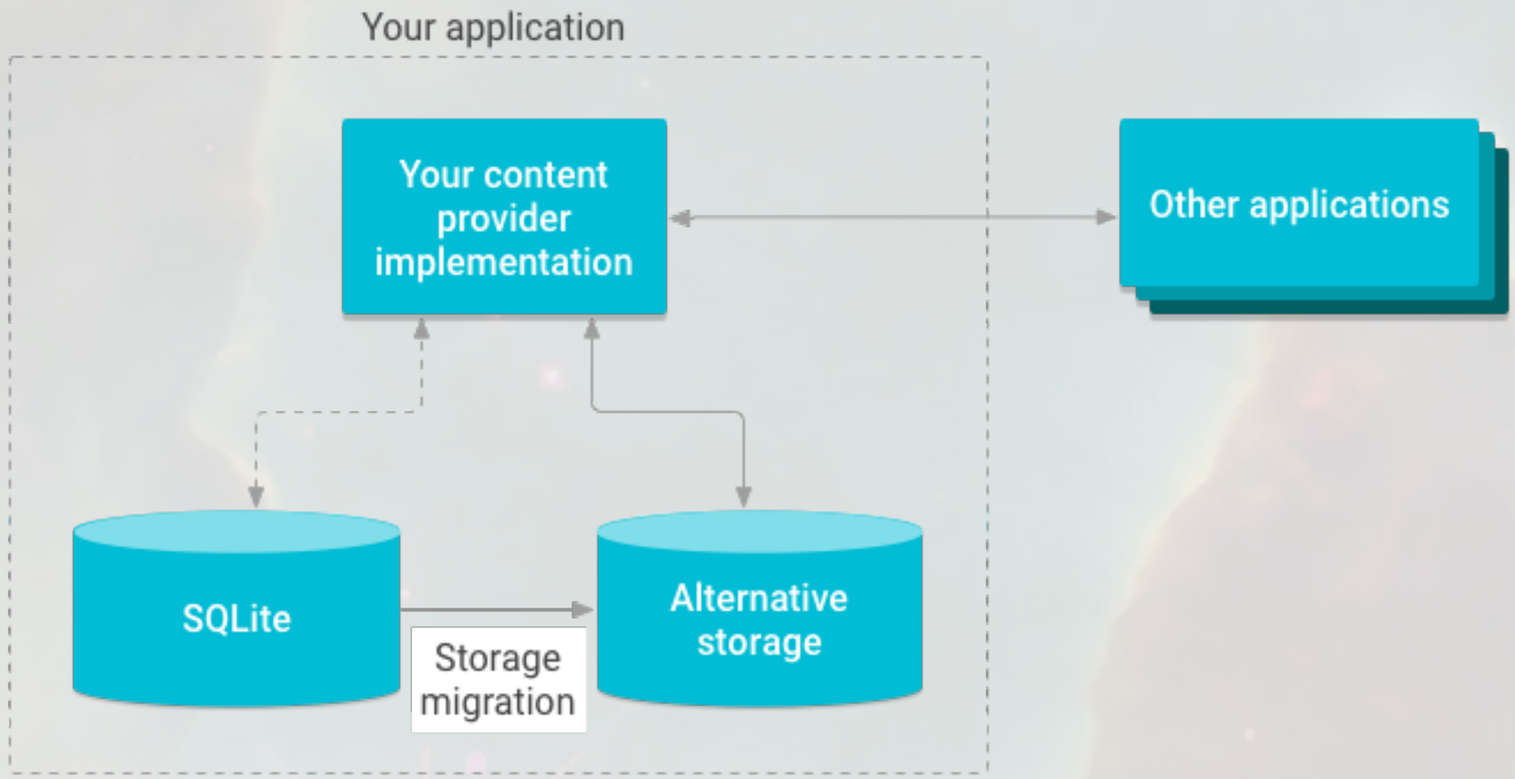
- Handle some content (data)
- Manage this content through a database (for instance)
- Must ensure the validity of its database
- May not known its clients



Advantages of ContentProviders? (1/2)

Nice abstraction

- Changing the underlying representation does not affect other applications



You can use it even if you don't plan to share with other Applications

Advantages of ContentProviders? (2/2)

- **Granular control over the permissions for accessing data**
  - Restrict access
  - Grant blanket
  - Configure read / write permissions

- **Can be used to mask many sources**
  - Your data appears as a single one

- **Nice pattern that abstract your data**
  - SQL or raw files are accessed the same way

Cursors

Wrap around read / write data

- Data is manipulated without knowing the underlying structure
- Can be seen as a direct access to a database raw

Offers CRUD features

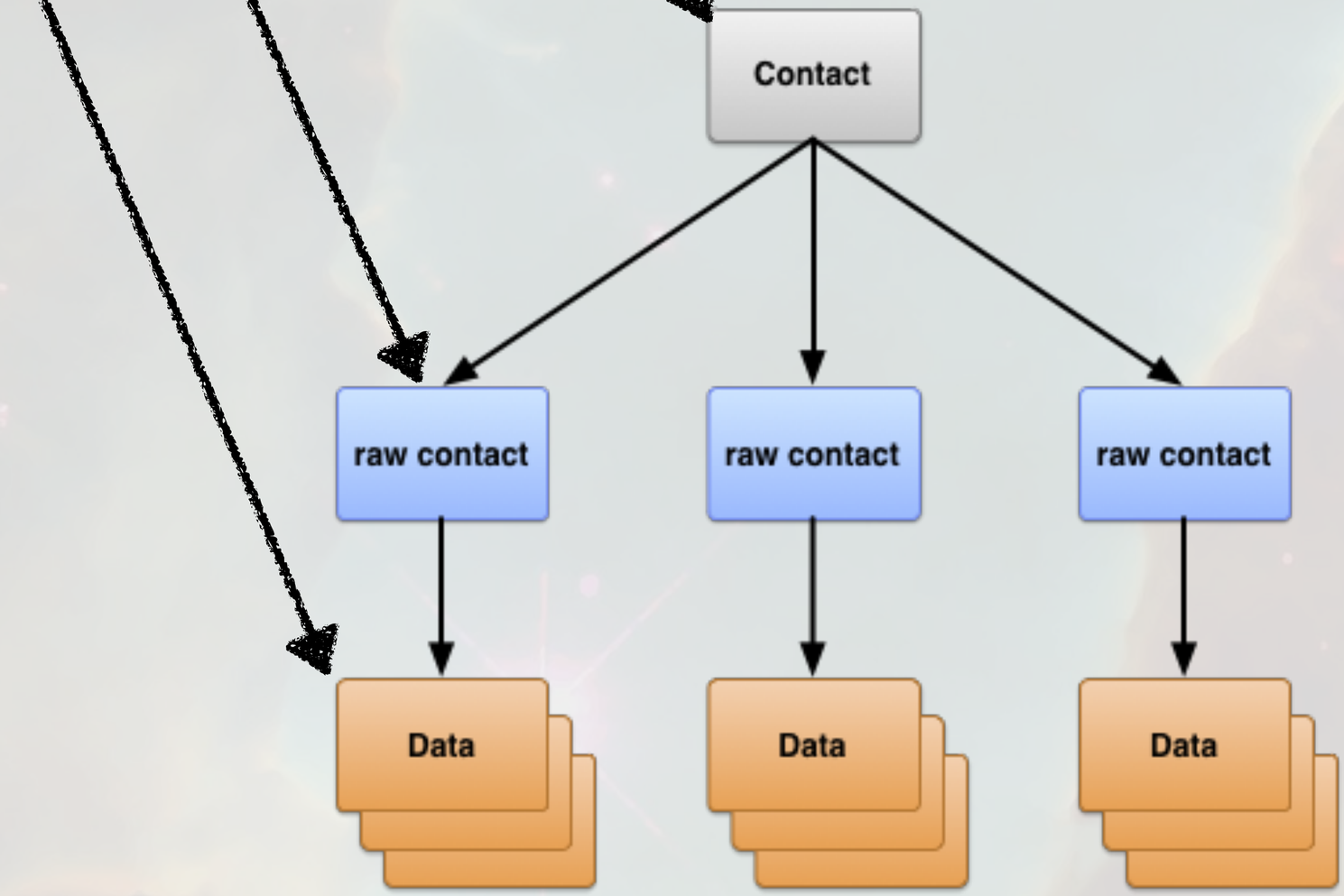
- **C**reate: create a new data
- **R**emove: remove some existing data
- **U**pdate: update existing data
- **D**elete: delete existing data

Implement android.database.Cursor

Using existing ContentProvider (Contacts)

Contacts structure

- ContactContract.Contracts: store all contacts
- ContactContract.RawContact: store summary for each contact
- ContactContract.Data: SMS, mails, etc.



Access to Contacts

Modify the AndroidManifest.xml

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.admin.mycontactapplication"
>
<uses-permission
    android:name="android.permission.READ_CONTACTS"
/>
```

Build the request

```
ContentResolver cr = getContentResolver();
Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
                      null, null, null, null);
```

Parameters for filtering etc.

Exploiting the cursor


```
if (cur.getCount() > 0) {
  while (cur.moveToNext()) {
     String name = cur.getString(
          cur.getColumnIndex(
               ContactsContract.Contacts.DISPLAY_NAME));

        Toast.makeText(getApplicationContext(),name,
                       Toast.LENGTH_SHORT).show();
     }
}
```

We can then look in the other bases if there are additional informations

Define a ContentProvider

Define your own URI

```
contents://com.example.admin.mycontactapplication
```

Implement ContentProvider

onCreate()	Prepare ContentProvider
getType(Uri)	Return the MIME type of the URI
delete(…)	Suppress an entry
insert(…)	Insert a new entry
Update(…)	Update a data

Create Database (1/2)


```
private SQLiteDatabase db;

static final String DATABASE_NAME = "mydb";

static final String TABLE_NAME = "names";

static final int DATABASE_VERSION = 1;

static final String CREATE_DB_TABLE =
          " CREATE TABLE "
        + TABLE_NAME
        + " (id INTEGER PRIMARY KEY AUTOINCREMENT, "
        + " name TEXT NOT NULL);";

private static class DatabaseHelper extends SQLiteOpenHelper {
    DatabaseHelper(Context context) {
       super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
```

Create Database (2/2)


```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_DB_TABLE);
}

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
                          int newVersion) {
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
      onCreate(db);
    }
}
```

Define Constants


```
public class MyProvider extends ContentProvider {
    static final String PROVIDER_NAME =
            "com.example.admin.mycontactapplication.MyProvider";
    static final String URL =
            "content://" + PROVIDER_NAME + "/cte";

    static final Uri CONTENT_URI = Uri.parse(URL);

    static final String id = "id";
    static final String name = "name";
    static final int uriCode = 1;

    static final UriMatcher uriMatcher;
    private static HashMap<String, String> values;

    static {
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(PROVIDER_NAME, "cte", uriCode);
        uriMatcher.addURI(PROVIDER_NAME, "cte/*", uriCode);
    }
```

Implement methods (1/4)


```
@Override
public String getType(Uri uri) {
 switch (uriMatcher.match(uri)) {
  case uriCode:
    return "vnd.android.cursor.dir/cte";
  default:
   throw new IllegalArgumentException
                ("Unsupported URI: " + uri);
  }
}

@Override
public boolean onCreate() {
   Context context = getContext();
   DatabaseHelper dbHelper = new DatabaseHelper(context);
   db = dbHelper.getWritableDatabase();
   if (db != null)
     return true;
  return false;
}
```

Implement methods (2/4)


```
@Override
public int delete(Uri uri, String selection,
                  String[] selectionArgs) {
  int count = 0;
  switch (uriMatcher.match(uri)) {
       case uriCode:
         count = db.delete(TABLE_NAME,
                           selection,
                           selectionArgs);
         break;
       default:
        throw new IllegalArgumentException
                           ("Unknown URI " + uri);
  }
  getContext().getContentResolver().notifyChange(uri, null);
  return count;
}
```

Implement methods (3/4)


```
@Override
public Uri insert(Uri uri, ContentValues values) {
   long rowID = db.insert(TABLE_NAME, "", values);

   if (rowID > 0) {
      Uri _uri =
           ContentUris.withAppendedId(CONTENT_URI, rowID);
      getContext().getContentResolver()
                  .notifyChange(_uri, null);
      return _uri;
   }

   throw new SQLException("Failed to add a record into " + uri);
}
```

Implement methods (3/4)


```
@Override
public Cursor query(Uri uri, String[] projection,
                    String selection, String[] selectionArgs,
                    String sortOrder) {
 SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
 qb.setTables(TABLE_NAME);
 switch (uriMatcher.match(uri)) {
      case uriCode:
          qb.setProjectionMap(values);
          break;
      default:
          throw new IllegalArgumentException("Unknown URI " + uri);
 }
 if (sortOrder == null || sortOrder == "") {
      sortOrder = name;
 }
  Cursor c = qb.query(db, projection, selection,
                  selectionArgs, null, null, sortOrder);
  c.setNotificationUri(getContext().getContentResolver(), uri);
  return c;
}
```

Add element to the database

New element to the database

```
public void onClickAddName(View view) {
  ContentValues values = new ContentValues();
  values.put(MyProvider.name, ((EditText)
             findViewById(R.id.txtName)).getText().toString());
  Uri uri = getContentResolver()
            .insert(MyProvider.CONTENT_URI, values);
}
```

Modify the AndroidManifest.xml

```
<provider
  android:name=".MyProvider"
  android:authorities=
        "com.example.admin.mycontactapplication.MyProvider"
  android:exported="true"
  android:multiprocess="true">
</provider>
```

Client (1/2)

Implement LoaderManager.LoaderCallbacks<Cursor>

```
CursorLoader cursorLoader;

public void onClickDisplayNames(View view) {
    getSupportLoaderManager().initLoader(1, null,  this);
}

@Override
public Loader<Cursor>
onCreateLoader(int arg0, Bundle arg1) {
   cursorLoader =
      new CursorLoader(this, Uri.parse("content://
com.example.admin.mycontactapplication.MyProvider/cte"),
                       null, null, null, null);
  return cursorLoader;
}
```

Client (2/2)

Implement LoaderManager.LoaderCallbacks<Cursor>

```
@Override
public void onLoadFinished(Loader<Cursor> arg0,
                           Cursor cursor) {
    cursor.moveToFirst();
    StringBuilder res = new StringBuilder();
    while (!cursor.isAfterLast()) {
      res.append("\n"
          + cursor.getString(cursor.getColumnIndex("id"))
          + "_"
          + cursor.getString(cursor.getColumnIndex("name")));
      cursor.moveToNext();
    }
  resultView.setText(res);
}
```

Summary

Content Providers

- Are used on databases
  - **But not necessarily**
- Efficient way to share informations between threads
  - **permissions can be fixed**
- May be used by a single application to provide an abstraction layer

Access is done through ContentResolver

- Predefined ones: CursorLoader, ...
- URI must be known

You can now build your own database