

Programação Para Dispositivos Móveis I

DATABASES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Índice

- Base de Dados em Android;
- SQLiteOpenHelper;
- Ligação à base de dados;
- Execução de SQL;
- Pesquisas SQL;
- Transações;
- Notas Finais;
- Leitura Adicional.

Base de Dados em Android

O Android fornece, nativamente, suporte para base de dados SQLite. Estas bases de dados apenas podem ser acedidas pela aplicação que as cria, nunca por aplicações terceiras;

- Para criar/gerir a base de dados deve-se criar uma subclasse de `SQLiteOpenHelper` com override aos métodos `onCreate()` e `onUpgrade()`.

Utiliza tipos de atributos simples

- `text`, `varchar`, `integer`, `float`, `numeric`, `date`, `time`, `timestamp`, `blob`, `boolean`, ...

Base de Dados em Android

Considerações

Não armazenar ficheiros (imagens, áudio ou vídeo)

- Em alternativa armazenar os caminhos relativos num atributo do tipo string;

Funcionalidades:

- Criar bases de dados e respetivas tabelas;
- Índices;
- Queries;
- Vistas;
- Triggers (parcialmente);
- Inserir, eliminar e atualizar registos;
- Transações;
- Chaves estrangeiras (primeiro tem de se ativar)
 - `db.execSQL("PRAGMA foreign_keys = ON;")`.

SQLiteOpenHelper

Esta classe vai auxiliar os processos de criação e manutenção da base de dados. Pode ser vista como um “assistente pessoal” que trata dos processos gerais de gestão da base de dados, nomeadamente:

- Criação da base de dados: O SQLiteHelper vai garantir que o ficheiro de base de dados é criado, com o nome correto, e correta estrutura de tabelas;
- Acesso à base de dados: A aplicação não precisa de saber todos os detalhes sobre a localização da base de dados, o SQLiteHelper fornece um objeto para obtermos acesso à base de dados sempre que necessário;
- Coerência a base de dados: É provável que a estrutura da base de dados sofra alterações ao longo do tempo, com o SQLiteHelper podemos converter uma versão antiga da base de dados para a versão mais atual.

SQLiteOpenHelper

Herança

onCreate()

- Executado quando a base de dados é criada (ficheiro com o nome definido em DATABASE_NAME);
- Permite a inicialização da base de dados (por ex.: criação das tabelas).

onUpgrade()

- Executado quando a versão da base de dados (DATABASE_VERSION) é alterada para um valor superior;
- Permite efetuar operações de atualização (por ex.: apagar e criar de novo as tabelas).

SQLiteOpenHelper

Estrutura [MyDBHelper.java]

```
public class MyDBHelper extends SQLiteOpenHelper {  
  
    private static final String DATABASE_NAME = "mydatabase.db";  
    private static final int DATABASE_VERSION = 1;  
  
    public MyDBHelper (Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) { ... }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { ... }  
}
```

Extende SQLiteOpenHelper

Nome da Base de Dados

Versão atual

Este parâmetro está relacionado com cursors (a abordar futuramente)

SQLiteOpenHelper

Criação Tabelas [MyDBHelper.java]

Criação da tabela Person em SQLite

```
CREATE TABLE PERSON (_id INTEGER PRIMARY KEY AUTOINCREMENT,  
NAME TEXT,  
AGE INTEGER)
```

Sabemos que este é um valor calculável, mas vamos utilizar para efeitos de exemplo

Criação da tabela Person com SQLiteHelper

```
@Override  
public void onCreate(SQLiteDatabase db) {
```

Este método executa instruções de SQL

```
    db.execSQL("CREATE TABLE PERSON(" +  
        + "_id INTEGER PRIMARY KEY AUTOINCREMENT," +  
        + "NAME TEXT," +  
        + "AGE INTEGER);");  
}
```

Em Android é convenção identificar os atributos que são chaves primárias como _id

SQLiteOpenHelper

Inserção de Dados [MyDBHelper.java]

- Para inserir dados numa tabela podem ser utilizados dois métodos. Podemos criar uma string com o método `execSQL()` ou utilizar a operação de `insert()` presente no `SQLiteOpenHelper`.

```
private static void insertPerson(SQLiteDatabase db, String name, int age){
```

```
    ContentValues personValues = new ContentValues();  
    personValues.put("NAME", name);  
    personValues.put("AGE", age);
```

```
    db.insert("PERSON", null, personValues);
```

```
}
```

Retorna o id da
linha inserida, ou
-1 em caso de
erro

Nome da tabela

Conjunto de valores
(par chave - valor)

SQLiteOpenHelper

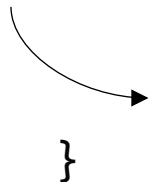
Inserção de Dados [MyDBHelper.java]

```
@Override
public void onCreate(SQLiteDatabase db) {

    db.execSQL("CREATE TABLE PERSON("
        + "_id INTEGER PRIMARY KEY AUTOINCREMENT,"
        + "NAME TEXT,"
        + "AGE INTEGER);");

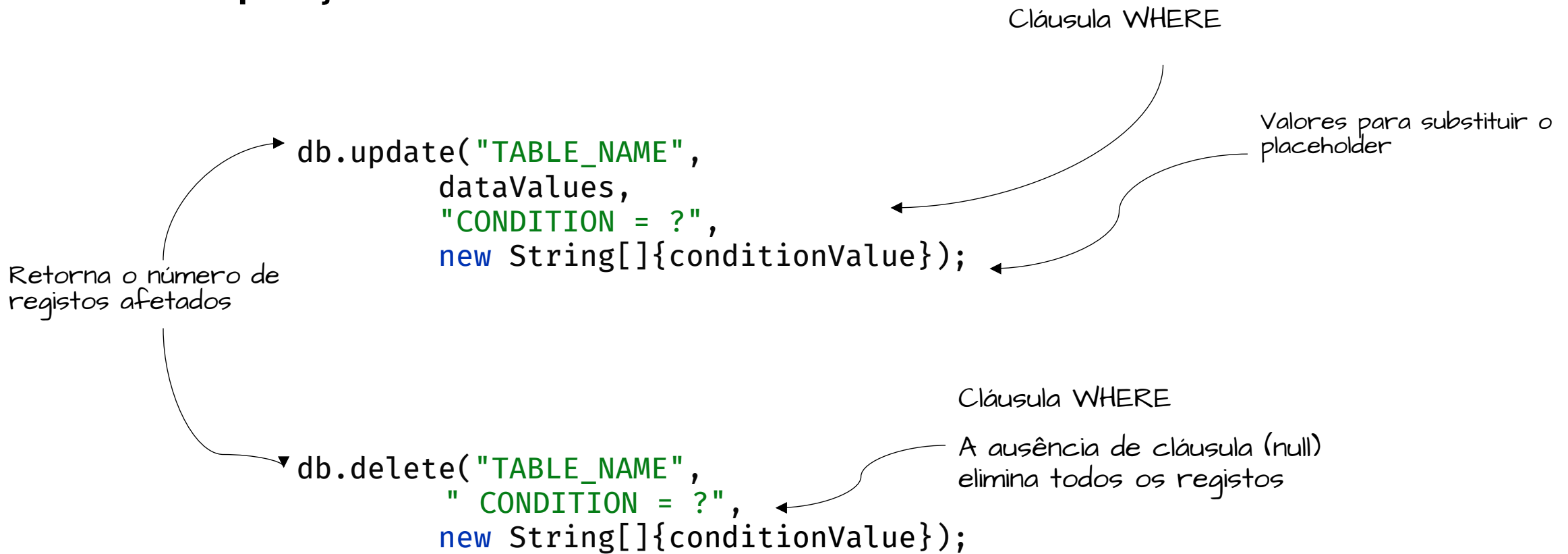
    insertPerson(db, "John", 35);
    insertPerson(db, "Chris", 18);
}
```

Inicializar a BD com
dados



SQLiteOpenHelper

Outras operações de dados



Base de Dados

Execução de SQL -> `execSQL()`

Cria a tabela PESSOA e respectivos campos

```
db.execSQL("CREATE TABLE PESSOA (_id INTEGER PRIMARY KEY AUTOINCREMENT,  
    name TEXT,  
    age INTEGER);");
```

Insere uma linha de valores na tabela PESSOA

```
db.execSQL("INSERT INTO PESSOA(name, age) VALUES ('John', 35)");
```

```
db.execSQL("UPDATE PESSOA SET name = 'Dan' WHERE _id=1");
```

Atualiza dados na linha com o atributo id = 1

```
db.execSQL("DELETE FROM PESSOA WHERE _id=1");
```

Remove o registo com o atributo id = 1

NOTA - A invocação do método `execSQL` deve ser feito dentro de um **bloco try-catch-finally**.

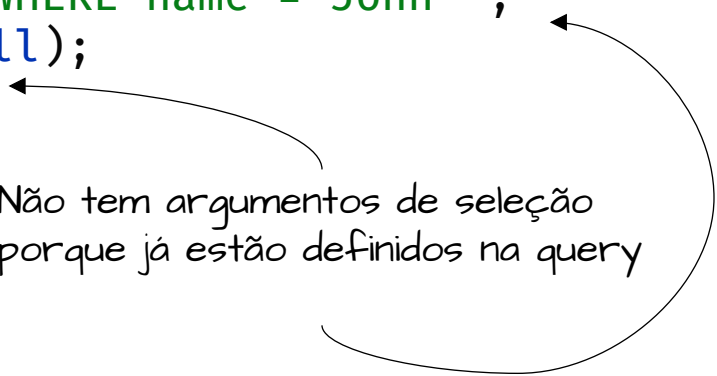
É necessário ter em atenção potenciais situações onde uma exceção `SQLException` possa ser lançada

SQLiteOpenHelper

Obtenção de Dados (Pesquisa) -> `rawQuery()`

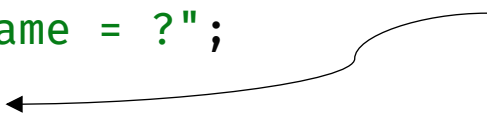
```
String sql = "SELECT * FROM PESSOA WHERE name = 'John'";  
Cursor cursor = db.rawQuery(sql, null);
```

Não tem argumentos de seleção
porque já estão definidos na query



```
String[] params = {"John"};  
String sql = "SELECT * FROM PESSOA WHERE name = ?";  
Cursor cursor = db.rawQuery(sql, params);
```

Os parâmetros substituem os
placeholders definidos na query



SQLiteOpenHelper

Obtenção de Dados (Pesquisa) -> query()

```
Cursor cursor = db.query(  
    Distinct false,  
    Nome da tabela "TABLE_NAME",  
    Nome das colunas seleccionadas new String[]{"COLUMN 1", "COLUMN 2"},  
    Cláusula WHERE "CONDITION = ?",  
    Parametros do placeholder conditionParameters,  
    Group By null,  
    Having null,  
    Order by null,  
    Limit null);
```

SQLiteOpenHelper

Obtenção de Dados (Pesquisa) -> Cursor

O cursor permite aceder aos dados produzidos pelas queries. Incluem métodos para aceder aos dados:

- **Métodos de posicionamento**

`isFirst()`, `isLast()`, `isBeforeFirst()`, `isAfterLast()`

- **Métodos de navegação**

`moveToFirst()`, `moveToLast()`, `moveToNext()`, `moveToPrevious()`, `move(n)`

- **Métodos de extração**

`getInt()`, `getString()`, `getFloat()`, `getBlob()`, `getDate()`, etc.

- **Métodos de análise da estrutura**

`getColumnName()`, `getColumnNames()`, `getColumnIndex()`, `getColumnCount()`,
`getCount()`, `getPosition()`

SQLiteOpenHelper

Obtenção de Dados (Pesquisa)

Verifica se o cursor
não está vazio

```
public Person getPerson(SQLiteDatabase db, String personName){  
    Cursor cursor = db.query("PERSON",  
        new String[]{"_id", "name", "age"},  
        "name = ?",  
        new String[]{personName},  
        null, null, null);  
  
    if(cursor != null && cursor.moveToFirst()){  
        int id = cursor.getInt(0);  
        String name = cursor.getString(1);  
        String age = cursor.getInt(2);  
        return new Person(id, name, age);  
    }  
    cursor.close();  
    return null;  
}
```

Fechar sempre o
cursor

SQLiteOpenHelper

Obtenção de Dados (Pesquisa)

```
public ArrayList<Person> getAllPersons(SQLiteDatabase db){
    ArrayList<Person> persons = new ArrayList<>();
    Cursor cursor = db.query("PERSON",
        new String[]{"_id", "name", "age"},
        null, null, null, null, null);

    if(cursor != null && cursor.moveToFirst()){
        do{
            int id = cursor.getInt(0);
            String name = cursor.getString(1);
            String age = cursor.getInt(2);
            persons.add(new Person(id, name, age));
        }while (cursor.moveToNext());
    }
    cursor.close();
    return persons;
}
```

Percorre o cursor
enquanto existirem
resultados

Não temos condição
porque queremos
obter todos os
registos

Ligação à Base de Dados

Estrutura [ex. MainActivity.java]

```
MyDBHelper dbHelper = new MyDBHelper(context);
SQLiteDatabase db = dbHelper.getWritableDatabase();

...

db.close();
```

Utilizamos o SQLiteOpenHelper para ligar à Base de Dados

Enviamos sempre o context como argumento

Liga à base de dados invocando o método onCreate() se ela ainda não tiver sido criada.

Base de dados em modo de escrita.

Para apenas modo de leitura utilizar getReadableDatabase()

É importante fechar sempre as ligações à base de dados

SQLiteOpenHelper

onUpgrade()

Este método tem 3 parâmetros: a base de dados, a versão atual da base de dados que o utilizador possui, a nova versão existente da base de dados.

@Override

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { }
```

Com o número de versões podemos controlar que alterações devem ser realizadas à base de dados.

SQLiteOpenHelper

onUpgrade()

@Override

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    if(oldVersion == 1){  
        //TODO: Implementar código caso a base de dados anterior esteja na versão 1  
    }  
    if(oldVersion < 3){  
        //TODO: Código a correr caso a versão da base de dados seja a 1 ou a 2  
    }  
}
```

SQLiteOpenHelper

onDowngrade()

Tal como o onUpgrade() este método têm 3 parâmetros: a base de dados, a versão atual da base de dados que o utilizador possui, a nova versão existente da base de dados.

```
@Override  
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    super.onDowngrade(db, oldVersion, newVersion);  
}
```

Transações

```
db.beginTransaction();
try {
    ...
    db.setTransactionSuccessful();
} catch (SQLException sqLiteException) {
    Log.e("SQL", sqLiteException.getMessage());
} finally {
    db.endTransaction();
}
```

Transações

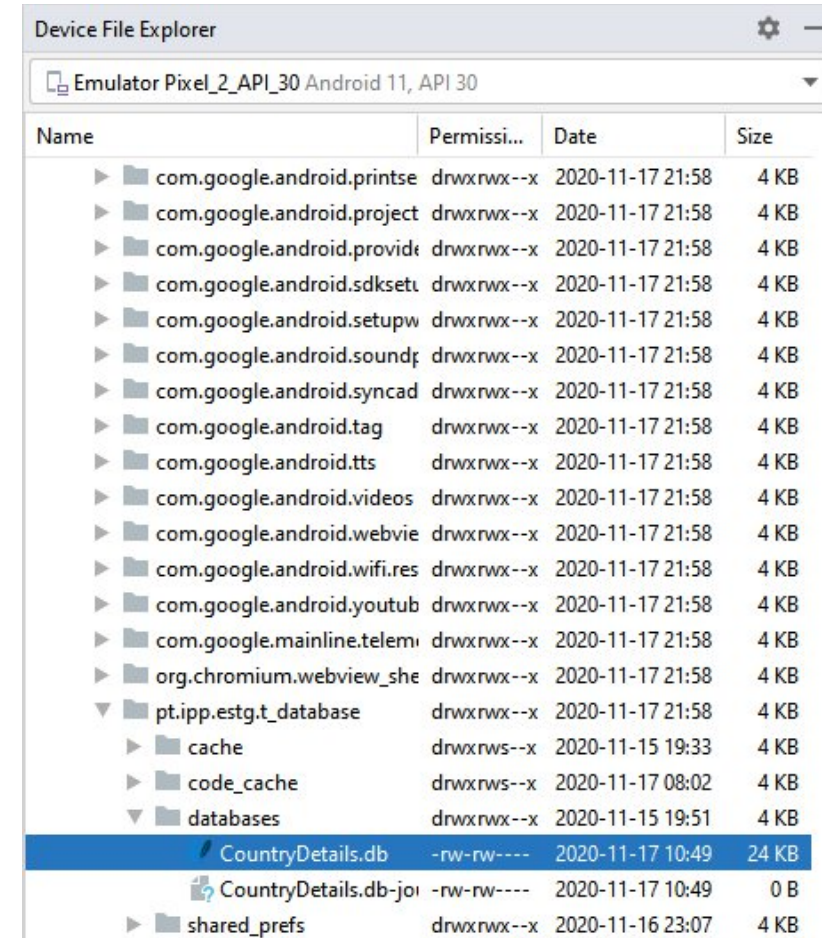
A transação é definida entre os métodos `beginTransaction()` e `endTransaction()`

- É necessário invocar o método `setTransactionSuccessful()` para submeter qualquer alteração à base de dados;
- A ausência do método `setTransactionSuccessful()` provoca um rollback automático para o estado da base de dados antes do início da transação.

Notas Finais

Localização da Base de Dados (Device File Explorer)

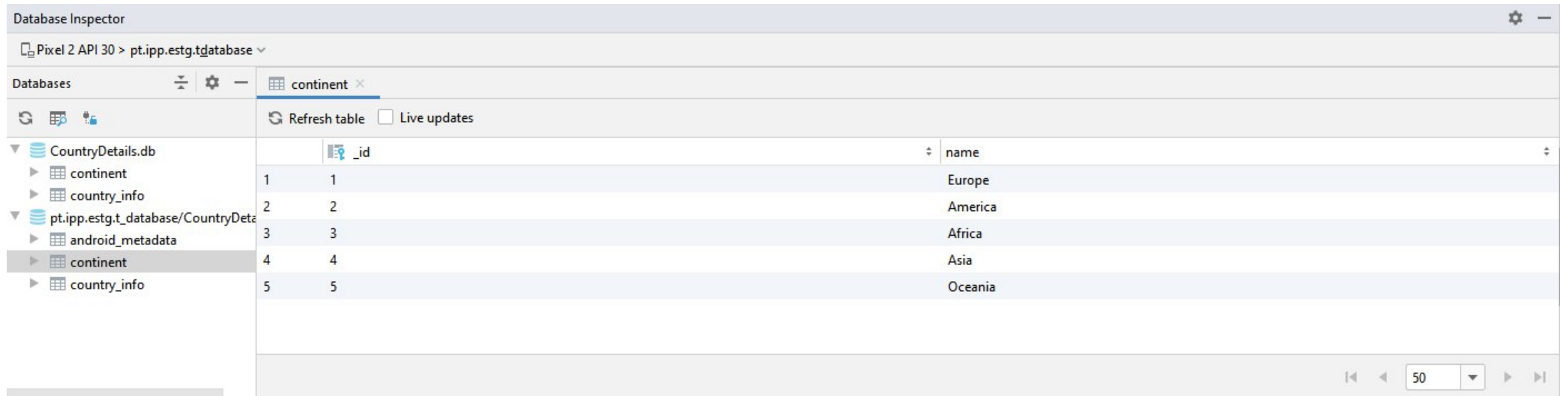
/data/data/[myAppPackage]/databases/[myDatabaseFile]



Name	Permissi...	Date	Size
▶ com.google.android.printse	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.project	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.provide	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.sdksetl	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.setupw	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.soundf	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.syncad	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.tag	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.tts	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.videos	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.webvie	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.wifi.res	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.android.youtub	drwxrwx--x	2020-11-17 21:58	4 KB
▶ com.google.mainline.telem	drwxrwx--x	2020-11-17 21:58	4 KB
▶ org.chromium.webview_she	drwxrwx--x	2020-11-17 21:58	4 KB
▼ pt.ipp.estg.t_database	drwxrwx--x	2020-11-17 21:58	4 KB
▶ cache	drwxrws--x	2020-11-15 19:33	4 KB
▶ code_cache	drwxrws--x	2020-11-17 08:02	4 KB
▼ databases	drwxrwx--x	2020-11-15 19:51	4 KB
CountryDetails.db	-rw-rw----	2020-11-17 10:49	24 KB
CountryDetails.db-journal	-rw-rw----	2020-11-17 10:49	0 B
▶ shared_prefs	drwxrwx--x	2020-11-16 23:07	4 KB

Notas Finais

Database Inspector



The screenshot shows the Database Inspector application interface. The title bar reads 'Database Inspector'. Below it, the connection path is 'Pixel 2 API 30 > pt.ipp.estg.tdatabase'. The left sidebar shows a tree view of databases, with 'pt.ipp.estg.t_database/CountryDetails.db' expanded and 'continent' selected. The main area displays the 'continent' table with columns '_id' and 'name'. The table contains 5 rows of data. At the bottom right, there are navigation controls and a page number '50'.

_id	name
1	Europe
2	America
3	Africa
4	Asia
5	Oceania

Notas Finais

Acesso

- Os conteúdos da pasta `data/data/myAppPackage` são sempre acessíveis através da aplicação identificada pelo `myAppPackage`;
- Do exterior (outras aplicações, ADB, ...) não é possível aceder a esta pasta, com a exceção de:
 - Emuladores
 - Dispositivos com Acesso Root

Leitura Adicional

SQLite on Android:

<https://developer.android.com/training/data-storage/sqlite>

SQLite API:

<https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase>

AndroidX e SQLite:

<https://developer.android.com/jetpack/androidx/releases/sqlite>

Programação Para Dispositivos Móveis I

DATABASES

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt