SQLite

CodeXP – Mobile

Helena Strada/Wilson Santana

BANCO DE DADOS

- Maneira estruturada de armazenar dados de forma persistente;
- Formato de tabelas (linhas e colunas);

ID	Nome	Fabricante
1	God of War	Sony

COMO

- Comandos para manipular os dados que serão inseridos, removidos, atualizados ou selecionados da nossa tabela;
- Além disso, para que os nossos dados sejam manipulados, precisamos criar as nossas tabelas com comandos específicos;
  - É importante saber que para cada dado a ser inserido da nossa tabela, nós temos um tipo de dado correspondente, bem como no Java:
    - `private String nome;`

COMANDOS

- INSERT: inserir um novo registro;
- SELECT: selecionar um registro;
- DELETE: deletar um registro;
- UPDATE: atualizar um registro existente.

SQLITE

- Biblioteca de software;
- Alta confiabilidade;
- Embedded (embutido);
- Sem configuração.

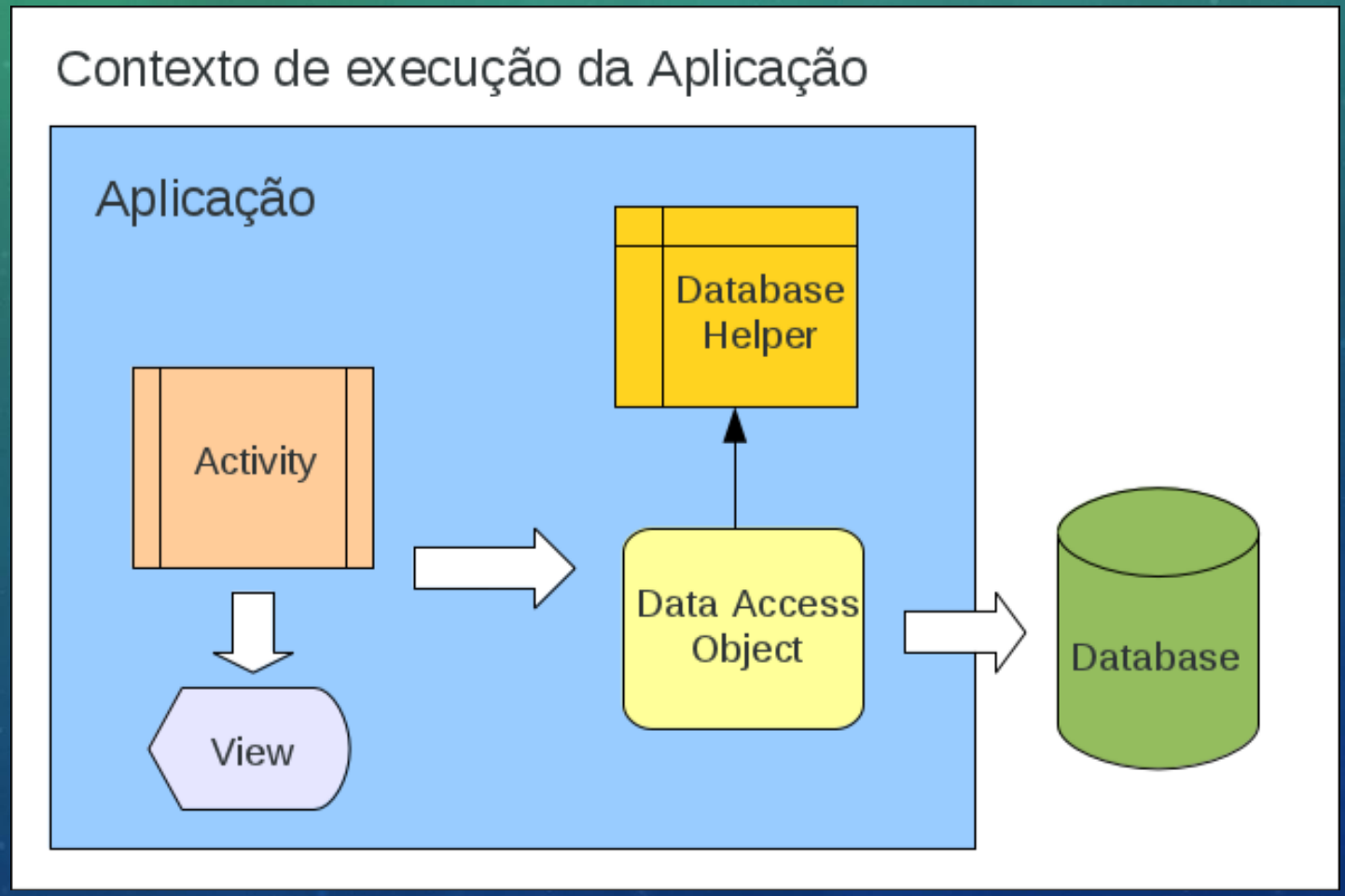
ARMAZENAMENTO

- DATA/data/<Nome-Aplicacao>/databases/<Nome-BD>
- DATA: caminho que o método Environment.getDataDirectory() retorna;
- Nome-Aplicacao: é o nome do seu aplicativo;
- Nome-BD: nome especificado para o seu banco de dados.

TIPOS DE DADOS

- Null;
- Integer;
- Real;
- Text;
- Blob.

CONTEXTO


Contexto de execução da Aplicação
Aplicação
Database Helper
Activity
Data Access Object
Database
View

ESQUEMA E CONTRATO

- Esquema:
  - Definição de como o nosso banco de dados será organizado.
- Contrato:
  - Definirá a estrutura do esquema, por exemplo, a estrutura da nossa tabela.

SQLiteOpenHelper

Criando o nosso banco de dados.

```java
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by helena.strada on 11/01/18.
 */

public class JogoDbHelper extends SQLiteOpenHelper {

    private static final String NOME_BANCO = "dbjogos.db";
    public static final String TABELA = "jogos";
    public static final String ID = "_id";
    public static final String NOME = "nome";
    public static final String FABRICANTE = "fabricante";
    private static final int VERSAO = 1;

    public JogoDbHelper(Context context) { super(context, NOME_BANCO,  factory: null, VERSAO); }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        String criarBD = "CREATE TABLE "+TABELA+" ("
                + ID + " integer primary key autoincrement,"
                + NOME + " text,"
                + FABRICANTE + " text)";
        sqLiteDatabase.execSQL(criarBD);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABELA);
        onCreate(sqLiteDatabase);

    }
}
```

REFERÊNCIA

DAO

```
private SQLiteDatabase db;
private JogoDbHelper dbo;

public JogoDao (Context context) {
    dbo = new JogoDbHelper(context);
}
```

COMANDOS

insert

```
public void salvar(Jogo jogo) {

    SQLiteDatabase db = dbo.getWritableDatabase();

    String inserir = "insert into "
            + JogoDbHelper.TABELA
            + " (nome, fabricante) values (?, ?)";
    db.execSQL(inserir, new Object[]{jogo.getNome(), jogo.getFabricante()});
    db.close();

}
```

COMANDOS

select

```java
public List<Jogo> getLista() {
    List<Jogo> jogos = new LinkedList<>();
    String rawQuery = "SELECT _id, nome, fabricante FROM " +
            JogoDbHelper.TABELA;
    SQLiteDatabase db = dbo.getReadableDatabase();
    Cursor cursor = db.rawQuery(rawQuery, null);
    Jogo jogo = null;
    if (cursor.moveToFirst()) {
        do {
            jogo = new Jogo();
            jogo.setId(cursor.getLong(0));
            jogo.setNome(cursor.getString(1));
            jogo.setFabricante(cursor.getString(2));
            jogos.add(jogo);
        } while (cursor.moveToNext());
    }
    return jogos;
}
```

```java
public Jogo localizar(Long id) {
    SQLiteDatabase db = dbo.getWritableDatabase();
    String query = "SELECT _id, nome, fabricante FROM " + JogoDbHelper.TABELA + " WHERE _id = ?";
    Cursor cursor = db.rawQuery(query, new String[]{String.valueOf(id)});
    cursor.moveToFirst();
    Jogo jogoA = new Jogo();
    jogoA.setId(cursor.getLong(0));
    jogoA.setNome(cursor.getString(1));
    jogoA.setFabricante(cursor.getString(2));
    db.close();
    return jogoA;
}
```

COMANDOS

delete

```java
public void remover(Jogo jogo) {
    SQLiteDatabase db = dbo.getWritableDatabase();

    String deletar = "delete from " + JogoDbHelper.TABELA + " where _id = ?";
    db.execSQL(deletar, new Object[]{jogo.getId()});
    db.close();

}
```

COMANDOS

update

```
public void atualizar(Jogo jogo) {

    SQLiteDatabase db = dbo.getWritableDatabase();

    String update = "update " + JogoDbHelper.TABELA + " set nome = ?, fabricante = ? where _id = ?";
    db.execSQL(update, new Object[]{jogo.getNome(), jogo.getFabricante(), jogo.getId()});
    Log.d("sql: ", update);
    db.close();
}
```