FIAP

DESENVOLVIMENTO AVANÇADO MOBILE

GOOGLE ANDROID

FIAP

DESENVOLVIMENTO ANDROID

PERSISTÊNCIA

SQLITE

O Android possui um banco de dados integrado denominado SQLite;

Ele aceita comandos SQL para criação de tabelas, consultas, etc... (vide **http://www.sqlite.org/lang.html**)

Contudo, apresenta algumas limitações:

- Suporta apenas os tipos de dados TEXT, INTEGER e REAL;
- Não oferece suporte à chaves estrangeiras;
- Não mantém a integridade entre os tipos de dados (exemplo: inserir String em campo INTEGER);

Apesar disso é extremamente leve em termos de consumo de memória;

SQLITE

FIAP

O SQLite separa instruções DDL de DML;

As instruções DDL para a criação da estrutura do banco devem ficar em uma classe que estende **SQLiteOpenHelper**;

Nela, os métodos **onCreate** (que cria o banco e suas tabelas) e **onUpdate** (atualiza a estrutura do banco) devem ser implementados;

Além disso, a classe **SQLiteOpenHelper** possui o seguinte construtor:

SQLiteOpenHelper(Context P1, String P2, CursorFactory P3, int P4)
Onde:

- P1 - Contexto associado ao banco (aplicação);
- P2 - Nome do banco de dados;
- P3 - Cursor padronizado para a pesquisa de dados (pode ser null);
- P4 - Versão do banco de dados (todas as vezes que a versão é atualizada, o método OnUpdate é executado).

SQLITE

FIAP

Exemplo:

```
public class MeuDb extends SQLiteOpenHelper {

   public MeuDb(Context context) {
      super(context, "MeuDb", null, 1);
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
       final String sql = "CREATE TABLE TAB_CLIENTE (COD_CLIENTE
INTEGER
       PRIMARY KEY AUTOINCREMENT, NOM_CLIENTE
       db.execSQL(sql);
   }

   @Override
   public void onUpgrade(SQLiteDatabase db, int vAntiga, int vNova)
{
       db.execSQL("DROP TABLE IF EXISTS TA
       onCreate(db);
   }
}
```

Versão do banco.

O Métodos execSQL executa instruções DDL.

Recria o banco todas as vezes que a versão é atualizada.

SQLITE

FIAP

Para executar instruções DML é necessário abrir o banco de dados por meio do método **getWritableDatabase** que retorna um objeto do tipo **SQLiteDatabase**;

A classe **SQLiteDatabase** oferece os métodos **insert**, **delete** e **update**:

insert (String P1, String P2, ContentValues P3), onde:
P1 é o nome da tabela, **P2** a forma de tratamento de valores nulos (opcional) e **P3** os valores a serem inseridos. O insert retorna um long com o ID gerado para o registro inserido;

update(String P1, ContentValues P2, String P3, String[] P4), onde **P1** é o nome da tabela, **P2** os valores que serão atualizados, **P3** a cláusula where e **P4** os parâmetros da cláusula where;

delete(String P1, String P2, String[] P3), onde **P1** é o nome da tabela, **P2** a cláusula where e **P3** os parâmetros da cláusula where;

SQLITE

Os valores a serem utilizados nas instruções insert e update são encapsulados em objetos da classe **ContentValues**;

A classe ContentValues possui um método put que permite informarmos pares contendo nome da coluna e valor;

Exemplo:

```
ContentValues cv = new ContentValues();
cv.put("NOM_CLIENTE", "JOAO DA SILVA");
db.insert("TAB_CLIENTE",  null, cv);


ContentValues cv2 = new ContentValues();
cv2.put("NOM_CLIENTE", "BATISTA ANTONIO");
// Atualiza o registro onde COD_CLIENTE = 1
db.update("TAB_CLIENTE", cv2, "COD_CLIENTE = ?", new String[]{"1"});


// Apaga o registro onde COD_CLIENTE = 2
db.delete("TAB_CLIENTE", "COD_CLIENTE = ?", new String[]{"2"});
```

SQLITE

Consultas podem ser realizadas na base de dados por meio do método **query** da classe **SQLiteDatabase** que retorna um objeto **Cursor**, utilizado para manipular o conjunto de resultados obtidos;

Cursor query (String P1, String[] P2, String P3, String[] P4, String P5, String P6, String P7)

Onde:
P1 - nome da tabela;
P2 - colunas desejadas;
P3 - cláusula **where;**
P4 - parâmetros da cláusula **where**;
P5 - cláusula **group by**;
P6 - cláusula **having**;
P7 - cláusula **order by**;

```
Cursor c = db.query(
    "TAB_CLIENTE",
    new String[]{"NOM_CLIENTE"},
    "COD_CLIENTE = ?",
    new String[]{"3"},
    null,
    null,
    "NOM_CLIENTE");
```

SQLITE

A classe Cursor oferece mecanismos de manipulação dos dados por meio de métodos:

- **getCount()** → total de registros recuperados;
- **moveToFirst()** → posiciona o cursor no primeiro registro do conjunto de dados;
- **moveToNext()** → move para o próximo registro. Retorna false caso não existam mais registros no conjunto de dados;
- **getInt(int P1), getDouble(int P1), getFloat(int P1), getString(int P1)** → retorna o valor de uma coluna informada no parâmetro **P1** conforme o tipo de dado especificado no método;
- **close()** → fecha o cursor;

Exemplo:

```
c.moveToFirst();
Log.i("PersistenciaActivity", "Total: " + c.getCount());
do {
      Log.i("PersistenciaActivity",
      "COD: " + c.getInt(0) +
      " NOME: " + c.getString(1));
} while (c.moveToNext());
c.close();
```

FIAP