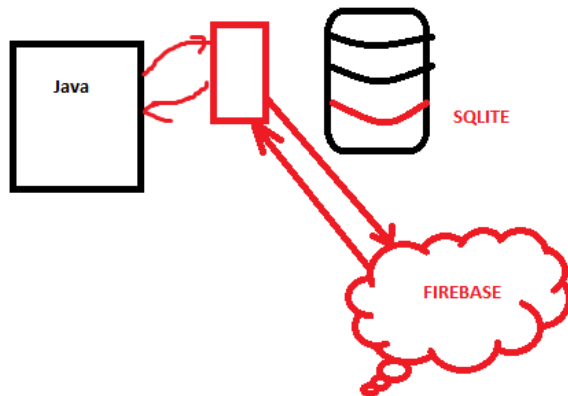


Persistência

- Arquivo (Pode dar ruim se houver vários usuários) (SQLite)
- SGB (Sistema gerenciador de banco de dados (MySQL, SQL Oracle)

Java é totalmente orientada a objeto, que é muito importante dividir tarefa, para mudar uma implementação sem mudar o código inteiro. Quando estivermos trabalhando com bancos com Java:



Conceito Básico de Banco de Dados

Uma das formas de manipular esses dados é utilizando banco de dados

MYSQL: Bastante difundido pois você não paga pra usar.

Fora o SQL, tem o DB2 da IBM, SQL Server da Microsoft (quando trabalhando com o Dot.net)

O SQLite permite utilizando comandos SQL em um arquivo de texto (base de dados).

Para conectar a um banco de dados SQLite ou qualquer outro utilizando JAVA, é necessário trazer o conector da JDBC ao projeto.

Mais sobre conectores ainda nessa aula.

Conector SQLite:

<https://bitbucket.org/xerial/sqlite-jdbc/downloads/>

Jar é como se fosse um arquivo jar contendo toda a sua aplicação Java.

- A criação das tabelas que serão utilizadas em um banco vem no momento anterior ao seu projeto de uso.

- As tabelas podem ser criadas em código, mas em geral, elas são criadas utilizando alguma ferramenta que permita a modelagem de suas interações

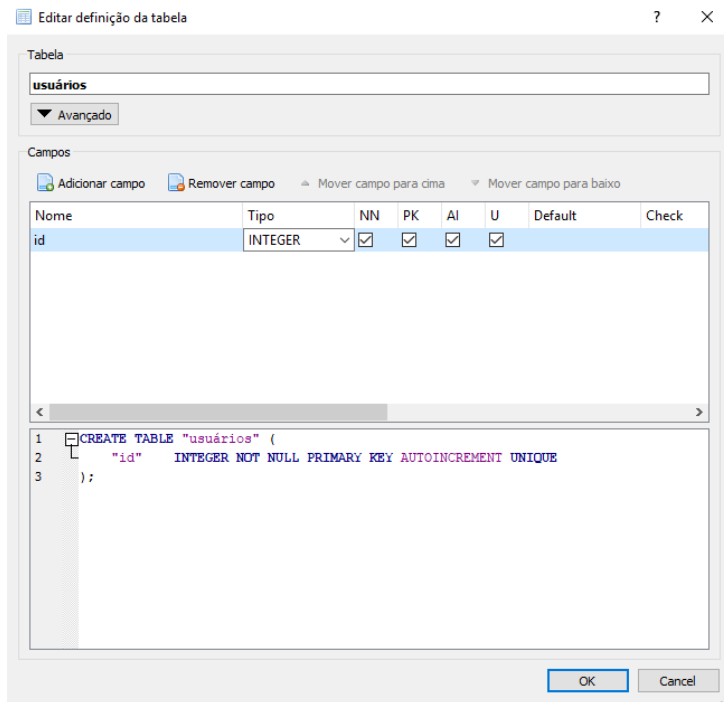
SQLite é nativo do Android e IOS

(Você consegue ver os arquivos que usam .db)

O banco é como se fosse um arquivo de excel, cada tabela que eu crio funciona como se fosse uma planilha. O excel é uma forma de extrair isso de forma visual.

Dentro do nosso banco vão ser criadas várias tabelas, cada uma com seus dados próprios. Cada coluna vai ser o que a gente chama de “campo”. Cada linha corresponde a um dado diferente.

Para o que serve um campo id numa tabela? Para você conseguir separar as linhas. Não é bom separar pelo nome, pois podem existir aplicações que linhas possuem o mesmo nome.



Editar definição da tabela

Tabela: usuários

Avançado

Campos

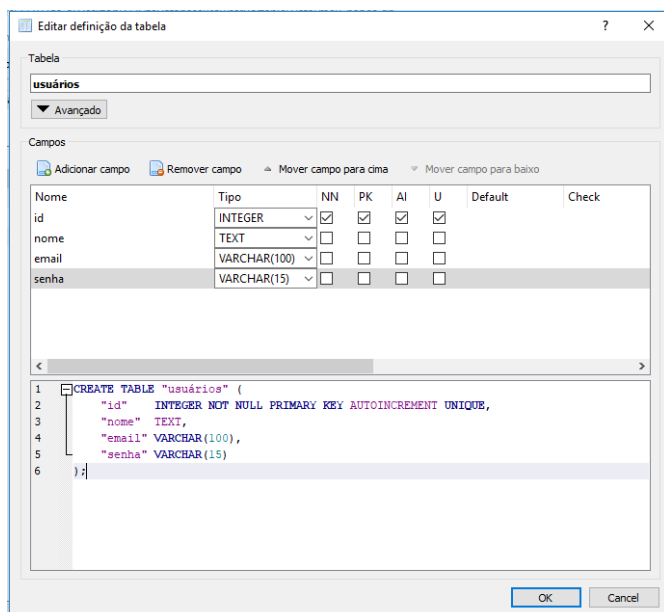
Adicionar campo Remover campo Mover campo para cima Mover campo para baixo

Nome	Tipo	NN	PK	AI	U	Default	Check
id	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		

```
1 CREATE TABLE "usuários" (  
2   "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE  
3 );
```

OK Cancel

VARCHAR(n): Pega uma lista de char, pegando até n caracteres.



Editar definição da tabela

Tabela: usuários

Avançado

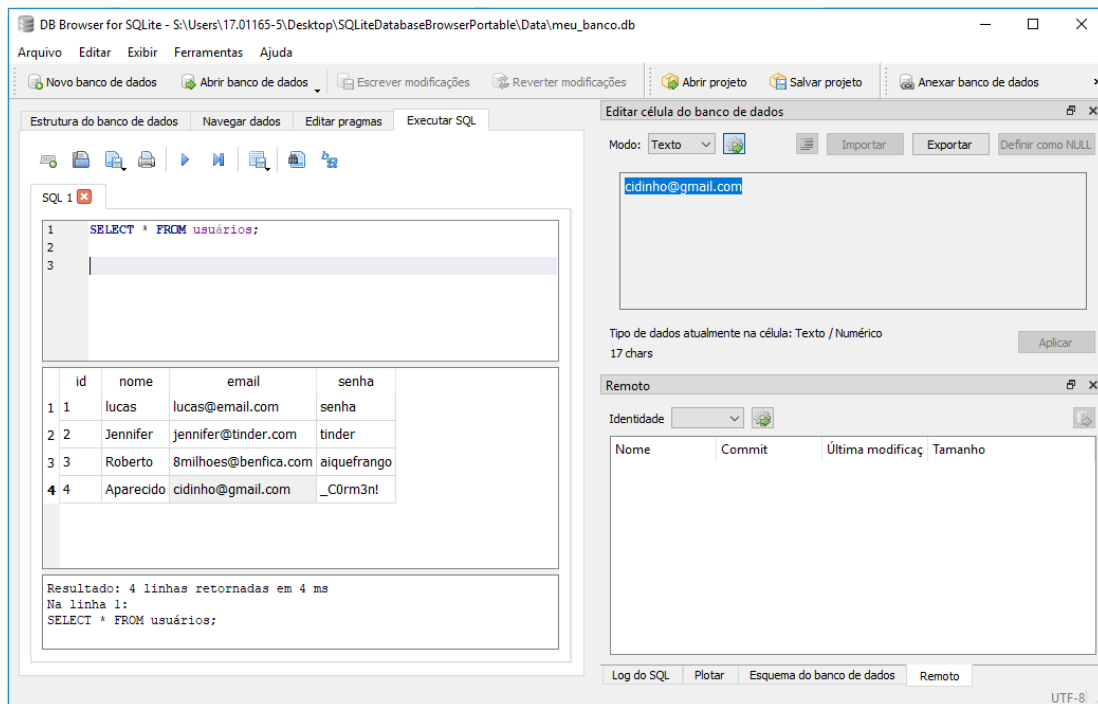
Campos

Adicionar campo Remover campo Mover campo para cima Mover campo para baixo

Nome	Tipo	NN	PK	AI	U	Default	Check
id	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
nome	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
email	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
senha	VARCHAR(15)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

```
1 CREATE TABLE "usuários" (  
2   "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,  
3   "nome" TEXT,  
4   "email" VARCHAR(100),  
5   "senha" VARCHAR(15)  
6 );
```

OK Cancel



<https://www.oracle.com/technetwork/java/overview-141217.html>

<https://docs.oracle.com/javase/tutorial/jdbc/basics/connecting.html>

- **DriverManager:** This fully implemented class connects an application to a data source, which is specified by a database URL. When this class first attempts to establish a connection, it automatically loads any JDBC 4.0 drivers found within the class path. Note that your application must manually load any JDBC drivers prior to version 4.0.
- **DataSource:** This interface is preferred over DriverManager because it allows details about the underlying data source to be transparent to your application. A DataSource object's properties are set so that it represents a particular data source. See [Connecting with DataSource Objects](#) for more information. For more information about developing applications with the DataSource class, see the latest [The Java EE Tutorial](#).

No nosso caso, ele vai se conectar ao arquivo de dados que a gente pediu. Se o banco não existe, ele lança uma exceção, caso contrário, ocorreria uma exceção.

```
Connection conn = null;

Properties connectionProps = new Properties();

connectionProps.put("user", this.userName);

connectionProps.put("password", this.password);

if (this.dbms.equals("mysql")) {

    conn = DriverManager.getConnection(

        "jdbc:" + this.dbms + "://" +

        this.serverName +
```

```
        ":" + this.portNumber + "/",  
        connectionProps);  
    } else if (this.dbms.equals("derby")) {  
        conn = DriverManager.getConnection(  
            "jdbc:" + this.dbms + ":@" +  
            this.dbName +  
            ";create=true",  
            connectionProps);  
    }  
    System.out.println("Connected to database");  
    return conn;  
}
```

CRUD: Operações mais comuns de serem realizadas com um banco de dados (CREATE, READ, UPDATE, DELETE).