ANDROID

SQL

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;

Do Java para o SQL e vice-versa, nós temos um tipo de dado correspondente.

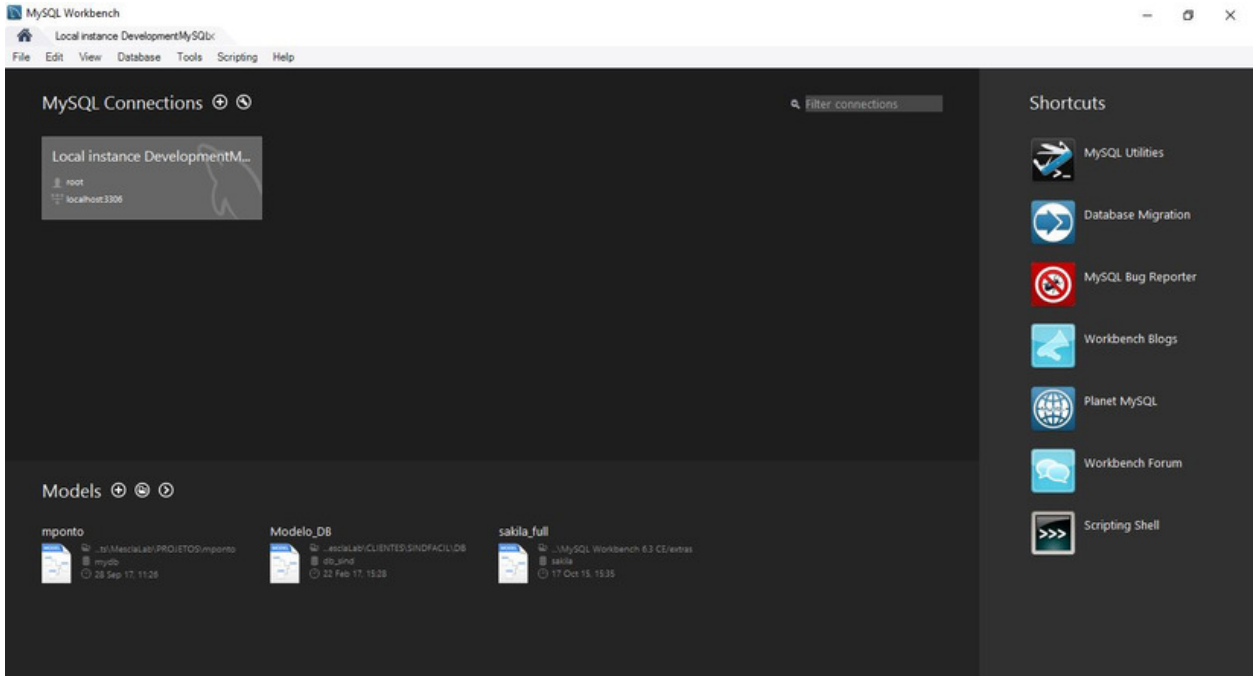
MYSQL

INTROUÇÃO

Sistema de Gerenciamento de Banco de Dados (SGDB);
Utiliza SQL (Structure Query Language).

MYSQL

INTERFACE



MySQL Workbench

CODEXP - MOBILE

MYSQL

ARMAZENAMENTO

Antes de falarmos sobre os tipos de dados e sobre as tabelas, precisamos criar o nosso banco.

BANCO DE DADOS

COMANDOS

Comandos DDL -
Data Definition Language

CREATE
criar (ex.: tabela ou schema)

ALTER
alterar (ex.: estrutura da tabela);

DROP
excluir (ex.: tabela);

BANCO DE DADOS

COMANDOS

Comandos DML -
Data Manipulation Language

INSERT
inserir um novo registro;

SELECT
selecionar um registro;

DELETE
deletar um registro;

UPDATE
atualizar um registro.

SQLITE

TIPOS DE DADOS

https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-reference-type-conversions.html

BIT(1) (new in MySQL-5.0)	BIT	java.lang.Boolean
BIT(> 1) (new in MySQL-5.0)	BIT	byte[]
TINYINT	TINYINT	java.lang.Boolean if the configuration property tinyInt1isBit is set to true (the default) and the storage size is 1, or java.lang.Integer if not.
BOOL, BOOLEAN	TINYINT	See TINYINT, above as these are aliases for TINYINT(1), currently.
SMALLINT[(M)] [UNSIGNED]	SMALLINT [UNSIGNED]	java.lang.Integer (regardless of whether it is UNSIGNED or not)
MEDIUMINT[(M)] [UNSIGNED]	MEDIUMINT [UNSIGNED]	java.lang.Integer (regardless of whether it is UNSIGNED or not)
INT, INTEGER[(M)] [UNSIGNED]	INTEGER [UNSIGNED]	java.lang.Integer, if UNSIGNED java.lang.Long
BIGINT[(M)] [UNSIGNED]	BIGINT [UNSIGNED]	java.lang.Long, if UNSIGNED java.math.BigInteger
FLOAT[(M,D)]	FLOAT	java.lang.Float
DOUBLE[(M,B)]	DOUBLE	java.lang.Double
DECIMAL[(M[,D])]	DECIMAL	java.math.BigDecimal
DATE	DATE	java.sql.Date
DATETIME	DATETIME	java.sql.Timestamp
TIMESTAMP[(M)]	TIMESTAMP	java.sql.Timestamp
TIME	TIME	java.sql.Time
YEAR[(2|4)]	YEAR	If yearIsDateType configuration property is set to false, then the returned object type is java.sql.Short. If set to true (the default), then the returned object is of type java.sql.Date with the date set to January 1st, at midnight.
CHAR(M)	CHAR	java.lang.String (unless the character set for the column is BINARY, then byte[] is returned.
VARCHAR(M) [BINARY]	VARCHAR	java.lang.String (unless the character set for the column is BINARY, then byte[] is returned.
BINARY(M)	BINARY	byte[]
VARBINARY(M)	VARBINARY	byte[]
TINYBLOB	TINYBLOB	byte[]
TINYTEXT	VARCHAR	java.lang.String
BLOB	BLOB	byte[]
TEXT	VARCHAR	java.lang.String
MEDIUMBLOB	MEDIUMBLOB	byte[]
MEDIUMTEXT	VARCHAR	java.lang.String
LONGBLOB	LONGBLOB	byte[]
LONGTEXT	VARCHAR	java.lang.String

1. CRIAR O BANCO DE DADOS (DATABASE/SCHEMA)

create schema ´senai_codexp_mobile´;

use ´senai_codexp_mobile´;

1.1. DEFINIÇÕES

Tables - manter os nossos registros (formato de linhas e colunas);
Views - é um objeto que pertence ao banco aonde podemos retornar registros à partir de uma determinada query (apenas select's);
Stored Procedures - similar a view, porém pode ou não retornar um registro; Aceita parâmetros;
Functions - retorna apenas um valor;

2. CRIAR TABELA

```
create table filmes
(
    id              int primary key auto_increment not null
    ,nome       varchar(255) not null
    ,genero     varchar(255)
);
```

3. INSERIR REGISTROS

```
insert into filmes (nome, genero) values ('Rambo - First Blood', 'Ação');
insert into filmes (nome, genero) values ('Rambo - First Blood Part II', 'Ação');
insert into filmes (nome, genero) values ('Rambo III', 'Ação');
insert into filmes (nome, genero) values ('Rambo', 'Ação');
insert into filmes (nome, genero) values ('Como eu era antes de você', 'Romance');
```

4. SELECIONAR REGISTROS

Para selecionar todas as colunas:
select * from filmes;

	id	nome	genero
▶	1	Rambo - First Blood	Ação
	2	Rambo - First Blood Part II	Ação
	3	Rambo III	Ação
	4	Rambo	Ação
	5	Como eu era antes de você	Romance
*	NULL	NULL	NULL

Selecionar apenas algumas colunas:
select nome, genero from filmes;

	nome	genero
▶	Rambo - First Blood	Ação
	Rambo - First Blood Part II	Ação
	Rambo III	Ação
	Rambo	Ação
	Como eu era antes de você	Romance

4.1. WHERE

Para selecionar registros com condição
select * from filmes where genero = 'Romance';
select * from filmes where id > 2;

	id	nome	genero
▶	5	Como eu era antes de você	Romance
*	NULL	NULL	NULL

	id	nome	genero
▶	3	Rambo III	Ação
	4	Rambo	Ação
	5	Como eu era antes de você	Romance
*	NULL	NULL	NULL

5. DELETE

delete from filmes where id = 2;
select * from filmes;

	id	nome	genero
▶	1	Rambo - First Blood	Ação
	3	Rambo III	Ação
	4	Rambo	Ação
	5	Como eu era antes de você	Romance
*	NULL	NULL	NULL

6. UPDATE

update filmes set nome = 'Como eu sou depois de você' where id = 5;
select * from filmes;

id	nome	genero
1	Rambo - First Blood	Ação
3	Rambo III	Ação
4	Rambo	Ação
5	Como eu sou depois de você	Romance
NULL	NULL	NULL

PROBLEMA

Quando começamos a inserir os registros, percebemos que na coluna Gênero, por exemplo, os valores estão sendo repetidos. Mas imagina que, um membro da equipe escreveu o nome nesta coluna errado.

insert into filmes (nome, genero) values ('Rambo - O retorno', 'Aço');
select * from filmes where genero = 'Ação';

id	nome	genero
1	Rambo - First Blood	Ação
3	Rambo III	Ação
4	Rambo	Ação
NULL	NULL	NULL

Além disso, se o banco de dados for configurado para case sensitive, o valor de Ação e ação serão considerados distintos, não trazendo também o resultado da sua query.

PROBLEMA

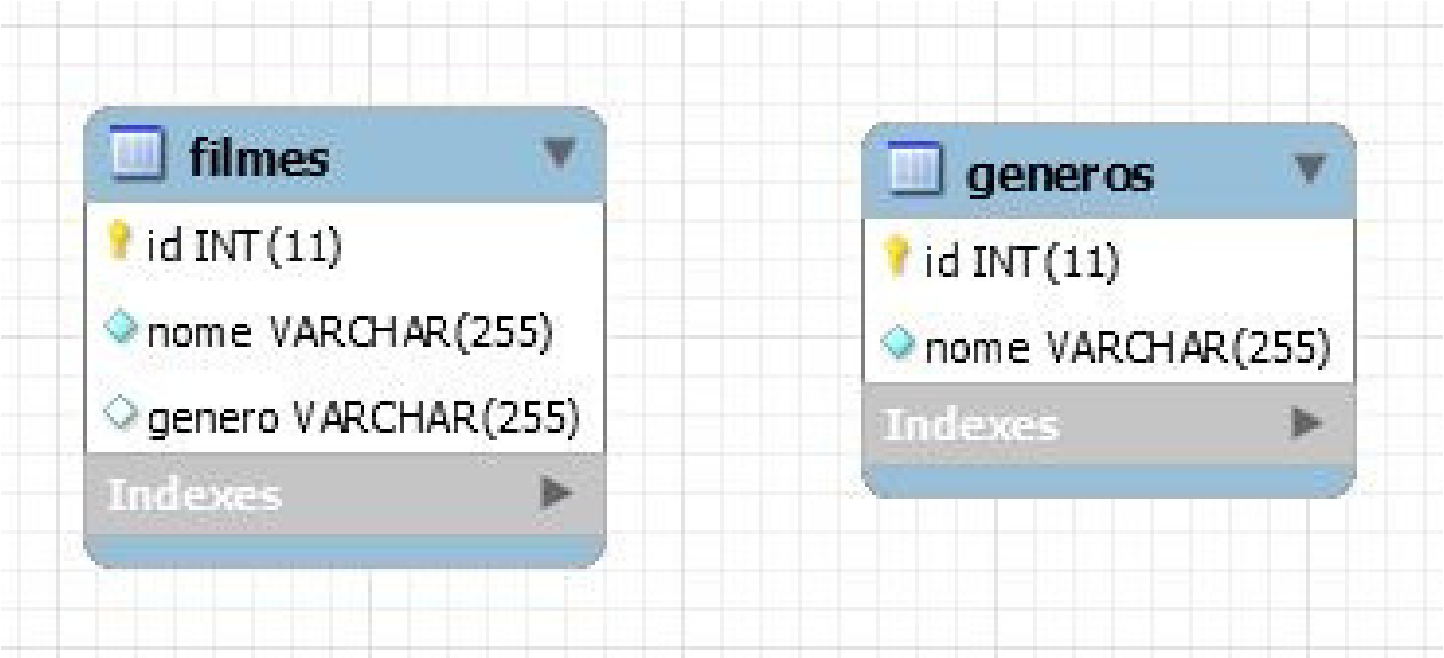
Qual a outra negativa em relação a isso? Imagina se precisássemos alterar o gênero de Ação para outro, teríamos que atualizar todos os nossos registros da nossa tabela dos quais o gênero dos filmes fosse igual ao que desejamos alterar.
Como podemos resolver isso?

E se criássemos uma tabela para armazenar os gêneros dos nossos filmes?

```
create table generos
(
    id int auto_increment primary key not null
    ,nome varchar(255) not null
);
```

PROBLEMA

Como está nossa estrutura até agora?

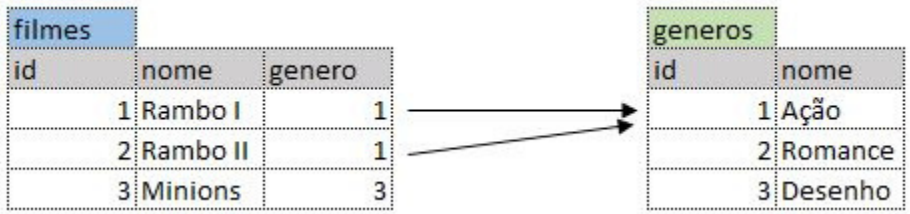


PROBLEMA

Ainda não estamos fazendo referência de uma tabela para outra. Precisamos dizer para a nossa tabela de filmes que o gênero que desejamos utilizar, é o gênero que inserimos na outra tabela. Como faremos isso?

Vamos começar do zero novamente:

Vamos começar o nosso problema. Precisamos inserir filmes, estes filmes possuem gêneros dos quais temos valores iguais.
Vamos criar duas tabelas. Filmes e gêneros. Apenas precisamos informar à nossa tabela de filmes que o gênero que desejamos trabalhar é o gênero da outra tabela. Vamos ver em uma tabela do excel.



filmes

id	nome	genero
1	Rambo I	1
2	Rambo II	1
3	Minions	3

generos

id	nome
1	Ação
2	Romance
3	Desenho

SOLUÇÃO

Precisamos apenas passar isto para o MySQL. Vamos começar novamente o nosso exercício.

```
drop schema senai_codexp_mobile;
create schema senai_codexp_mobile;
use senai_codexp_mobile;
```

SOLUÇÃO

Porém, faremos diferente para criarmos a nossa tabela. Qual tabela sabemos que não tem nenhuma "dependência"? A nossa tabela de gêneros. Eles irão existir antes de criarmos a nossa tabela de filmes. Podemos definir, neste nosso exemplo, a tabela de filmes sem que exista a tabela de gêneros?

```
create table generos
(
	id		int auto_increment primary key not null
	,nome	varchar(255) not null
);
```

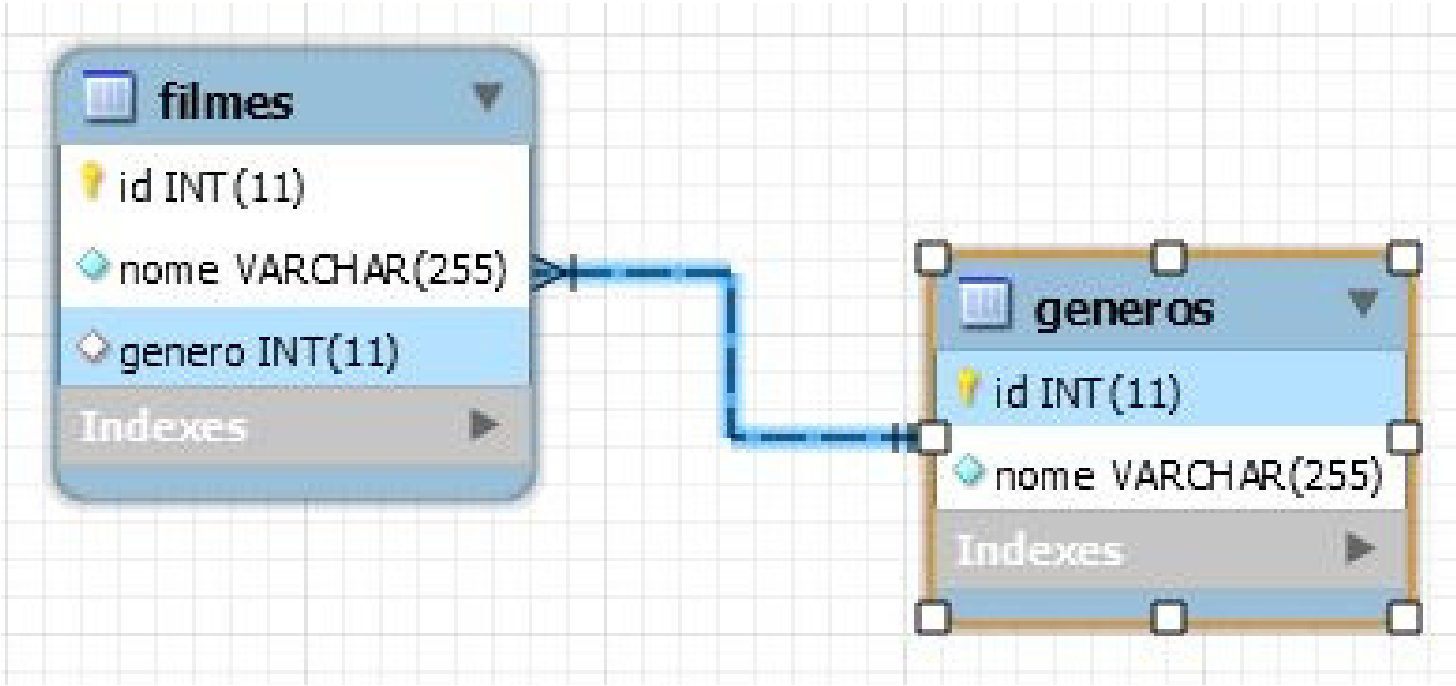
SOLUÇÃO

Agora podemos criar a nossa tabela de filmes e informarmos para ela que o gênero que desejamos incluir, fará uma referência para a nossa tabela de gêneros.

```
create table filmes
(
	id		int not null auto_increment primary key
	,nome	varchar(255) not null
	,genero	int
	,foreign key (genero) references generos(id)
);
```

SOLUÇÃO

Estrutura atual



SOLUÇÃO

Antes de inserirmos um filme, agora precisamos inserir um tipo de gênero.

insert into generos (nome) values ('Ação'), ('Romance'), ('Drama'), ('Desenho');
select * from generos;

	id	nome
▶	1	Ação
	2	Romance
	3	Drama
	4	Desenho
*	NULL	NULL

SOLUÇÃO

Vamos agora inserir um novo registro para filmes.

insert into filmes (nome, genero) values ('Rambo I', 1), ('Minions', 4), ('Rambo II', 1);
select * from filmes;

	id	nome	genero
▶	1	Rambo I	1
	2	Minions	4
	3	Rambo II	1
*	NULL	NULL	NULL

SOLUÇÃO

Trouxemos o id do nosso gênero, porém, queremos mostrar o nome do gênero.

select f.*, g.* from filmes f inner join generos g on f.genero = g.id;

	id	nome	genero	id	nome
▶	1	Rambo I	1	1	Ação
	3	Rambo II	1	1	Ação
	2	Minions	4	4	Desenho

ANDROID - SQLITE

OBRIGADO!