

Banco de dados

Revisão de modelagem e introdução aos comandos SQL

Wagner Cesar Vieira
wagner.vieira@sp.senai.br



Agenda

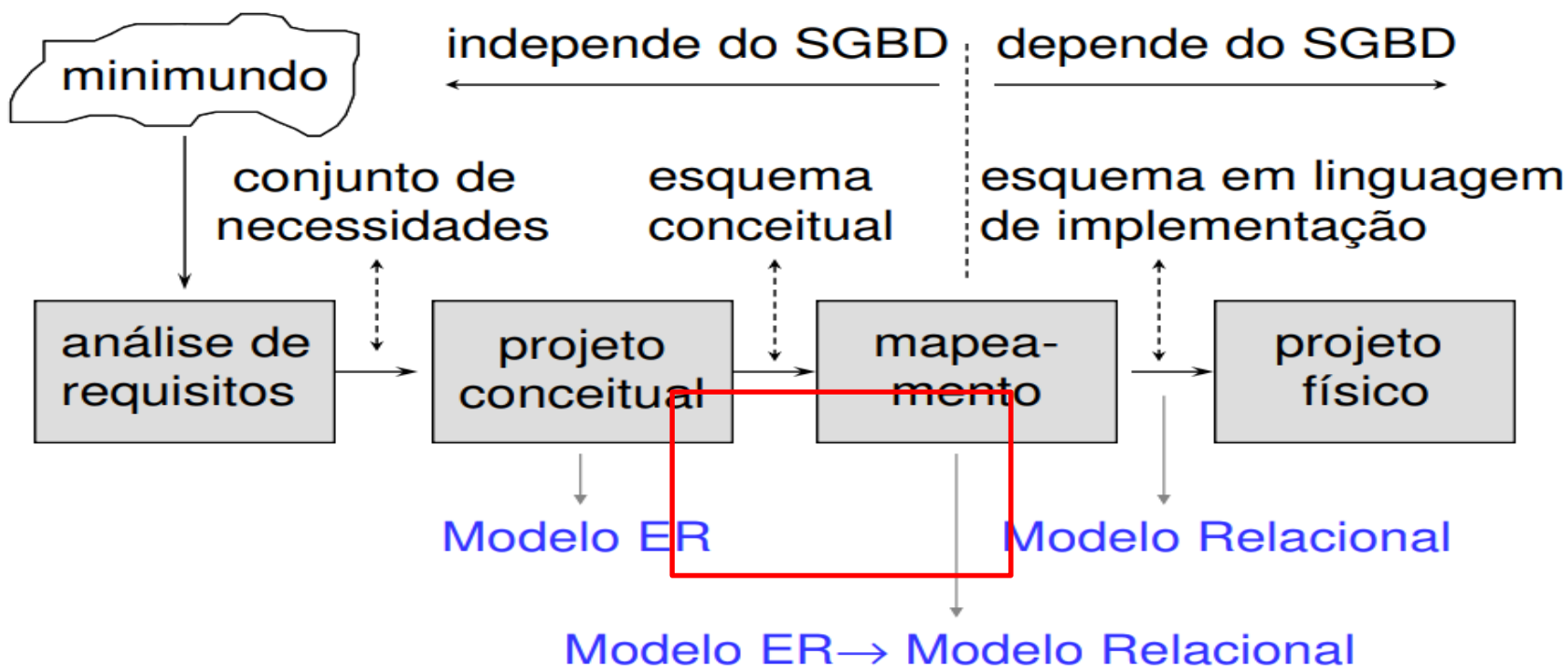


- ❑ Introdução
- ❑ Mapeamento relacional
- ❑ Resolução dos exercícios
 - ❑ Projeto conceitual
 - ❑ Projeto lógico
 - ❑ Script em linguagem SQL



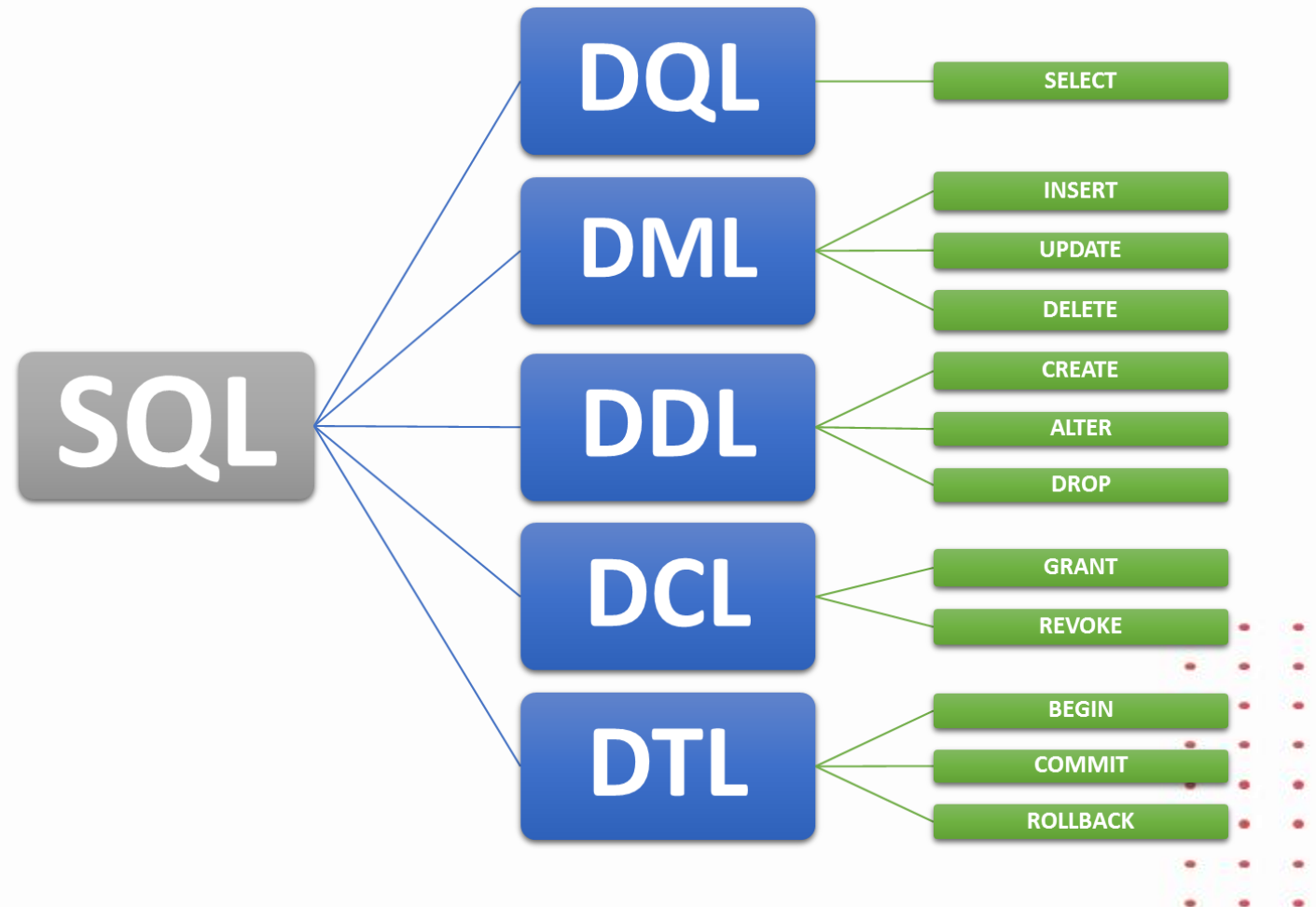
Introdução

Depois de criado o modelo Conceitual, o próximo passo é a criação do modelo lógico. Existe um processo conhecido como mapeamento que nos auxilia na passagem do modelo conceitual para o modelo lógico relacional de dados.



Linguagem SQL (Structured Query Language)

A linguagem SQL é o recurso mais conhecido por DBAs e programadores para a execução de comandos em bancos de dados relacionais. É por meio dela que criamos tabelas, colunas, índices, atribuímos permissões a usuários, bem como realizamos consultas a dados. Enfim, é utilizando a SQL que “conversamos” com o banco de dados.



SQL DDL - Data Definition Language



Uma tarefa indispensável, porém realizada com menos frequência, é criar o banco de dados. Em seguida, é natural a criação das tabelas a ele relacionadas. Para isso, lidamos com os comandos **CREATE DATABASE** e **CREATE TABLE** da linguagem SQL, os quais serão apresentados a seguir utilizando o SGBD **MySQL** com a ferramenta **Query Browser**:

❑ Criando o banco de dados academia:

❖ **CREATE DATABASE** csi_sesi; [IF NOT EXISTS]

❑ Para selecionar o banco de dados criado, utilizamos:

❖ **USE** csi_sesi;

❑ Para excluir o banco de dados com todos os objetos, utilizamos:

❖ **DROP DATABASE** csi_sesi; [IF EXISTS]



SQL DDL – Criando tabelas (tables)




A criação de uma base de dados com tabelas e tudo mais depende de muito planejamento. Não existe uma fórmula única, muito pelo contrário. Desenhar a base de dados é como programar, há sempre mais de um caminho para a mesma solução.

❑ Criando uma tabela para o banco de dados **csi_sesi**:

CREATE TABLE vitima

(idVitima **INT** NOT NULL ,
nome **VARCHAR**(80) NOT NULL,
cpf **NUMERIC**(11,0) NOT NULL,
rg **VARCHAR**(20) NOT NULL,
telefone **VARCHAR**(20) NOT NULL,
email **VARCHAR**(60) NOT NULL,
datanasc **DATE**,
PRIMARY KEY (idVitima)
);

vitima			
	Column Name	Data Type	Allow Nulls
	idVitima	int	<input type="checkbox"/>
	nome	varchar(80)	<input type="checkbox"/>
	cpf	numeric(11, 0)	<input type="checkbox"/>
	rg	varchar(20)	<input type="checkbox"/>
	telefone	varchar(20)	<input type="checkbox"/>
	email	varchar(60)	<input type="checkbox"/>
	datanasc	date	<input type="checkbox"/>
			<input type="checkbox"/>

Comandos DDL: Alter Table



ALTER TABLE – Comando usado para adicionar, excluir ou modificar as colunas de uma tabela existente.

Adiciona coluna

❑ **ALTER TABLE** table_name **ADD** column_name datatype

Exemplo: **ALTER TABLE** vitima **ADD** bairro **VARCHAR**(60);

Exclui uma coluna

❑ **ALTER TABLE** table_name **DROP COLUMN** column_name

Exemplo: **ALTER TABLE** vitima **DROP COLUMN** bairro;

Modifica uma coluna

ALTER TABLE vitima

ALTER COLUMN nome varchar(120);



Comandos DDL: Rename, truncate e drop



❑ **RENAME** – Comando utilizado para renomear tabelas.

Exemplo: **RENAME TABLE** vitimas **TO** vitima;

❑ **TRUNCATE** – Comando para excluir os dados de uma tabela, e não a tabela em si.

Exemplo: **TRUNCATE TABLE** vitima;

❑ **DROP** – Comando utilizado para excluir os dados de tabelas ou bancos:

DROP TABLE – Excluir uma tabela

Exemplo: **DROP TABLE** vitima;

DROP DATABASE – Excluir um banco de dados

Exemplo: **DROP DATABASE** csi_sesi;



Script SQL – Criação do banco e tabelas



```
USE master
GO
```

```
CREATE DATABASE csi_sesi3
GO
```

```
USE csi_sesi3
GO
```

```
CREATE TABLE vitima
( idVitima INT NOT NULL,
  nome VARCHAR(80) NOT NULL,
  cpf NUMERIC(11,0) NOT NULL,
  rg VARCHAR(20) NOT NULL,
  telefone VARCHAR(20) NOT NULL,
  email VARCHAR(60) NOT NULL,
  datanasc DATE NOT NULL,
  PRIMARY KEY(idVitima)
)
GO
```

```
CREATE TABLE criminoso
( codigoCriminoso INT NOT NULL,
  nome VARCHAR(50) NOT NULL,
  cpf NUMERIC(11,0) NOT NULL,
  rg VARCHAR(20) NOT NULL,
  datanasc DATE NOT NULL,
  foto NVARCHAR(100) NOT NULL,
  impressao NVARCHAR(100) NOT NULL,
  PRIMARY KEY(codigoCriminoso)
)
GO
```



Script SQL – Criação do banco e tabelas



```
CREATE TABLE policial
( idPolicia INT NOT NULL,
  nome VARCHAR(50) NOT NULL,
  cpf NUMERIC(11,0) NOT NULL,
  rg VARCHAR(20) NOT NULL,
  telefone VARCHAR(20) NOT NULL,
  cargo VARCHAR(20) NOT NULL,
  PRIMARY KEY(idPolicia)
)
GO
```

```
CREATE TABLE arma
( idArma INT NOT NULL,
  descricao VARCHAR(40) NOT NULL,
  fabricante VARCHAR(20) NOT NULL,
  tipo NVARCHAR(20) NOT NULL,
  PRIMARY KEY(idArma)
)
GO
```

```
CREATE TABLE ocorrencia
✓ ( idOcorrencia SMALLINT NOT NULL,
  descricao NVARCHAR(100) NOT NULL,
  local VARCHAR(30) NOT NULL,
  data DATE NOT NULL,
  hora TIME NOT NULL,
  vitima INT NOT NULL,
  bandidin INT NOT NULL,
  policia INT NOT NULL,
  arma INT NOT NULL,
  PRIMARY KEY(idOcorrencia),
  FOREIGN KEY(vitima) REFERENCES vitima(idVitima),
  FOREIGN KEY(bandidin) REFERENCES criminoso(codigoCriminoso),
  FOREIGN KEY(policia) REFERENCES policial(idPolicia),
  FOREIGN KEY(arma) REFERENCES arma(idArma)
)
GO
```

Comandos DML: INSERT



INSERT – Insere linhas de dados em uma coluna.

Exemplo:

INSERT INTO policial **VALUES**

(35533, 'Jose Ribas', 19987547852, '29.256.487-X', '(99) 8775-7878', 'escrivão');

INSERT INTO – Comando para inserir os dados clientes – nome da tabela

VALUES – Valores que serão inseridos

INSERT INTO policial (idPolicia, nome, cpf, rg, telefone, cargo **VALUES**

(87855, 'Maria Silva', 48878544152, '395698745', '(77) 4455-4774', 'delegada');



Comandos DML: INSERT



Inserção de vários registros

```
INSERT INTO policial VALUES
(88862, 'Pedro Tomás Alexandre Corte Real', 55086234000, '474406520', '(99) 5555-4052', 'policial'),
(88864, 'Márcio Giovanni Porto', 4507314070, '108759477', '(99) 5555-2799', 'policial'),
(88872, 'Benjamin Renato Marcos Ferreira', 17459586730, '190529507', '(99) 5555-7447', 'policial'),
(88873, 'Pedro Sérgio Duarte', 88152360740, '338783805', '(99) 5555-9548', 'policial'),
(88876, 'Mariana Joana Sales', 40288180445, '277445802', '(99) 5555-8963', 'policial'),
(88878, 'Carolina Malu Marina Silveira', 67396675278, '482404851', '(99) 5555-3002', 'policial'),
(88879, 'Agatha Sophia da Luz', 44354916134, '220810795', '(99) 5555-3998', 'policial'),
(99932, 'Luana Bárbara Rayssa Freitas', 72899983679, '155663471', '(99) 5544-0002', 'delegado'),
(99949, 'Isaac Renan Davi Fogaça', 25445237354, '400849306', '(99) 5544-0003', 'delegado'),
(99952, 'Benjamin Edson Noah das Neves', 6227851680, '245600577', '(99) 5544-0001', 'delegado')
GO
```

Comandos DML: UPDATE



UPDATE – Atualiza os dados de uma tabela. Veja exemplo de utilização:

UPDATE alunos **SET** telefone = '(99) 5555-8936'

WHERE cpf=40288180445 **AND** nome='Mariana Joana Sales';

Atualize (**UPDATE**) o campo altura para (**SET**) quando (**WHERE**) o cpf for igual a 40288180445 e (**AND**) o nome for igual a Mariana Joana Sales.

UPDATE – Comando para atualizar os dados / **SET** – o que será alterado

WHERE – condição / **AND** – acrescenta outras condições



Comandos DML: DELETE



DELETE – Exclui dados de uma tabela. Veja exemplo de utilização:

DELETE FROM policial **WHERE** nome = 'Pedro Sérgio Duarte';

DELETE – comando para deletar

FROM – em, indica a tabela

WHERE – condição essencial que indica o que será excluído

Note que em **DELETE** você ainda pode usar o **AND** caso deseje acrescentar mais condições.

Observação: Nunca realize um **DELETE** sem uma cláusula **WHERE**, para evitar excluir vários registros.



Comandos DQL: SELECT simples



O comando **SELECT** é bastante simples e, serve para você, mostrar os dados da tabela. Sua sintaxe pode ser utilizado de algumas formas, como:

SELECT * FROM NOME_DA_TABELA;

Exemplo: **SELECT * FROM** vitima;

Nota 1: o asterisco indica que você deseja mostrar todos os campos da tabela.

Exemplo 2: **SELECT cpf, nome, telefone FROM** vitima;

Nota 2: neste caso, você está indicando que deseja mostrar somente cpf, nome e telefone. Isso é interessante, pois a consulta é executada de forma mais rápida, já que você está trazendo menos informações da tabela.



Comandos DML: SELECT simples / Cláusula WHERE



A cláusula Where permite ao comando SQL passar condições de filtragem. Exemplos:

```
SELECT * from vitima WHERE cpf > 22222222222;
```

```
SELECT * from vitima WHERE idVitima >= 33333;
```

**Operação
com números**

```
SELECT * from vitima  
WHERE nome LIKE '%A%' AND cpf > 90040040044;
```

```
SELECT * from vitima  
WHERE nome LIKE '%A%' AND idVitima > 55555  
ORDER BY nome;
```

**Operação
com textos**



Exercício proposto: Cinema



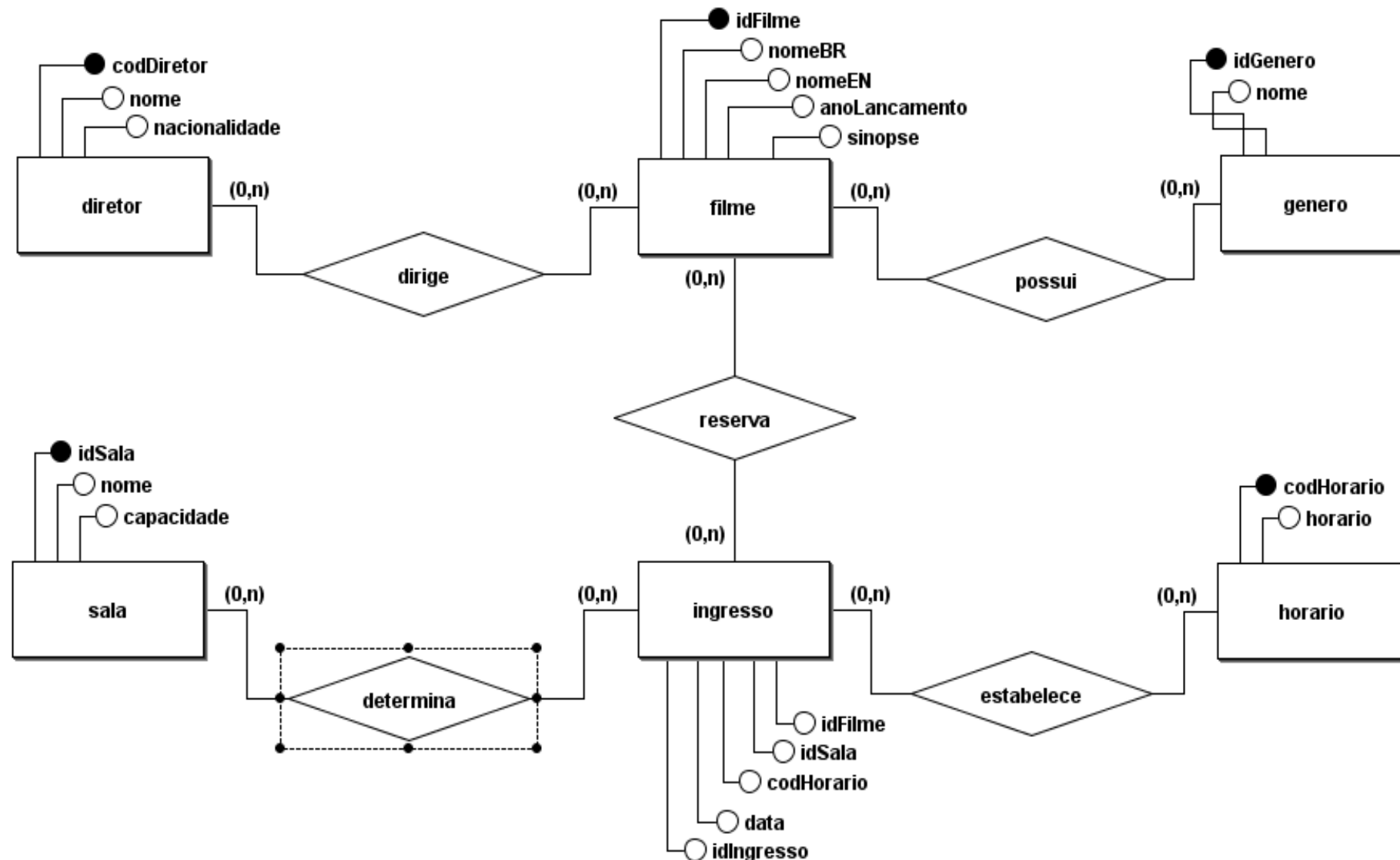
Um cinema possui várias salas de cinema, as quais exibem filmes em horários diversos. O cinema tem interesse em saber quais filmes estão atualmente em cartaz, em que salas e em que horários.

Cada sala possui um nome (único) e capacidade (número de lugares). Os filmes são caracterizados por seu nome em português, nome na língua original (se estrangeiro), diretor, ano de lançamento, tipo, e sinopse. Não existem dois filmes com o mesmo nome (em português) e ano de lançamento.

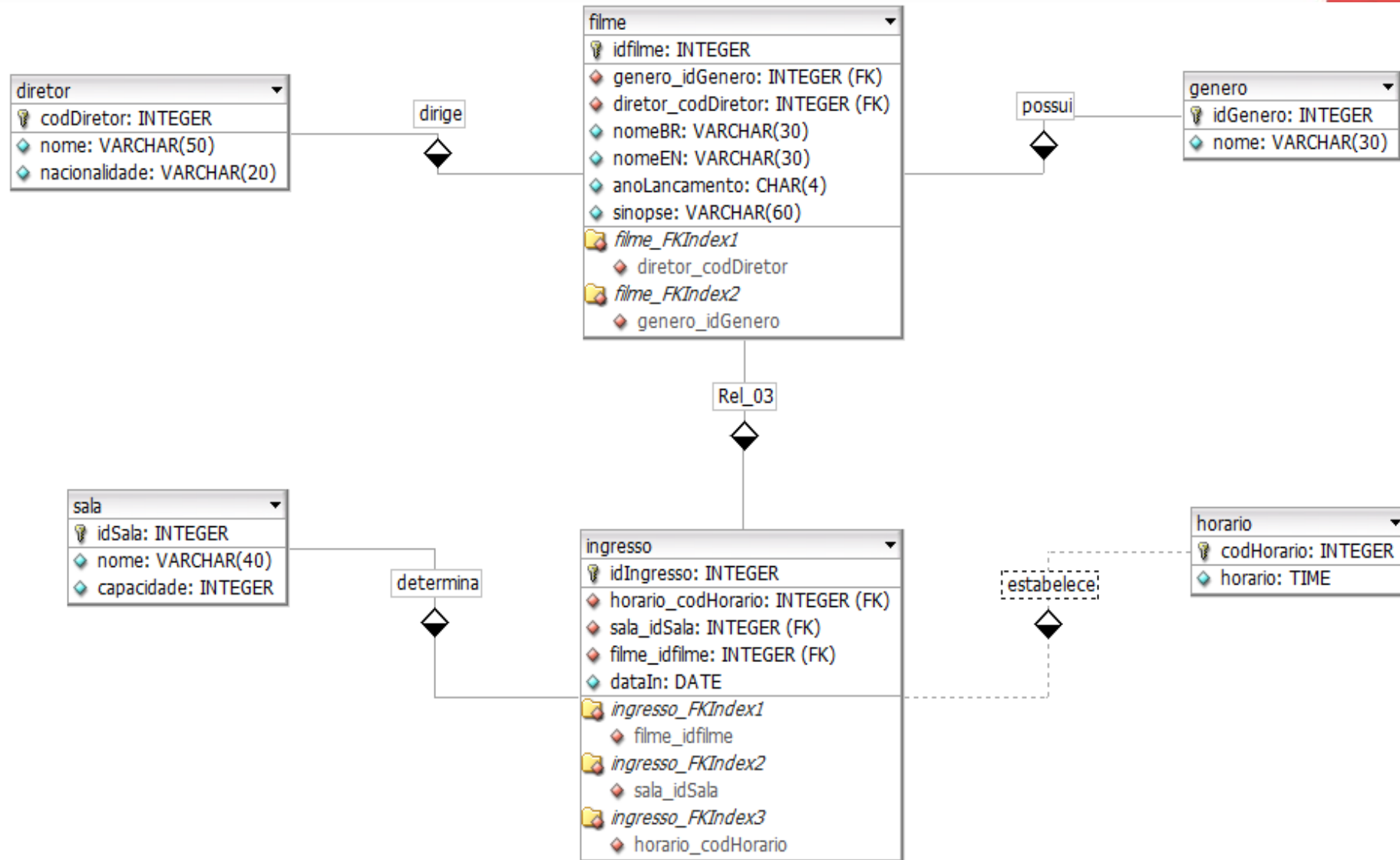
Uma exibição de filme ocorre em uma dada sala e horário. Um mesmo filme pode ser exibido na mesma sala, em vários horários. Para filmes muito procurados, o cinema pode ter exibição simultâneas em várias salas (em horários simultâneos ou não). Filmes diferentes podem passar na mesma sala, desde que obviamente não no mesmo horário.



Modelo conceitual – Cinema



Projeto lógico (mapeamento): Cinema



Exercício proposto: Biblioteca



Uma biblioteca deseja manter informações sobre seus livros. Inicialmente, quer armazenar para os livros as seguintes características: ISBN, título, ano, editora e autores deste livro. Para os autores, deseja manter: nome e nacionalidade. Cabe salientar que um autor pode ter vários livros, assim como um livro pode ser escrito por vários autores.

Cada livro da biblioteca pertence a uma categoria. A biblioteca deseja manter um cadastro de todas as categorias existentes, com informações como: código da categoria e descrição. Uma categoria pode ter vários livros associados a ela.





Wagner Cesar Vieira
wagner.vieira@sp.senai.br

