C207 - BD

Arthur Openheimer

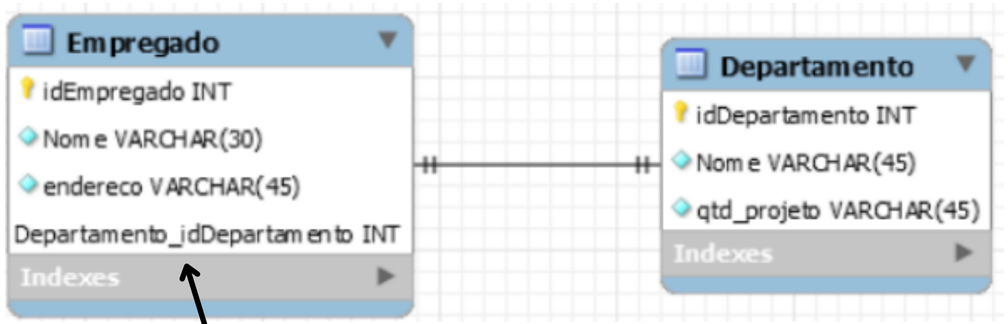
Inatel

Informações Gerais

- Atendimento → Terça-feira 17:30-19:30, prédio 1, sala 19
- Email → arthur.openheimer@ges.inatel.br
- Github → https://github.com/ArthurOpenheimer/C207-Monitoria

Inatel

Criando tabelas com chaves estrangeiras




```sql
CREATE TABLE Departamento(
    idDepartamento INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(20),
    qtd_projeto int
);

CREATE TABLE Empregado(
    idEmpregado INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(20),
    endereco VARCHAR(20)
);
```

Mas e a chave estrangeira?

Inatel

Criando tabelas com chaves estrangeiras

CONSTRAINT fk_tabela FOREIGN KEY (atributo1) REFERENCES Nome_Tabela2 (atributo2);

- atributo1 é da tabela que RECEBE a chave primária da outra tabela através do atributo2
- Esse padrão é usado para qualquer tipo de relacionamento

Inatel

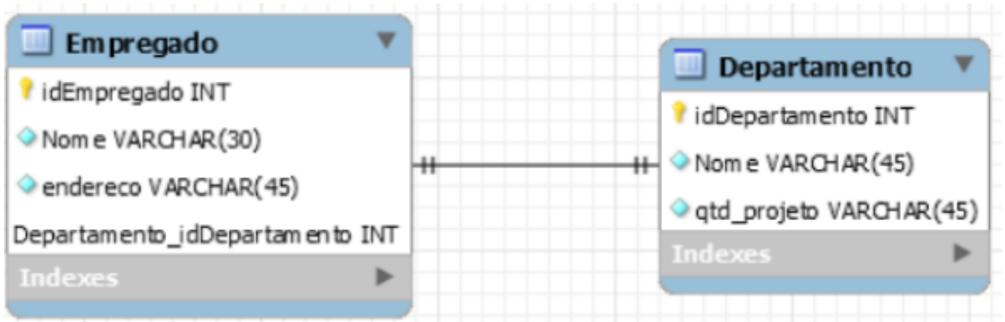
Criando tabelas com chaves estrangeiras

CONSTRAINT fk_tabela FOREIGN KEY (atributo1) REFERENCES Nome_Tabela2 (atributo2);

- **Constraint**
  - Define o que deve ser feito nas tabelas e atributos relacionados quando uma restrição for quebrada (define regras para os dados de uma tabela)
- **On Delete/On Update**
  - São regras opcionais que fazem com que a tabela original do atributo2 receba alterações quando uma determinada ação é executada:
  - “**No Action**“ e “**Restrict**“ - Não alteram ou deletam nada quando uma restrição é quebrada
  - “**Cascade**“ - Altera ou deleta todos os atributos que estão envolvidos no relacionamento
  - “**Set Null**“ - Seta como null os atributos envolvidos na restrição

Inatel

Relacionamento 1:1

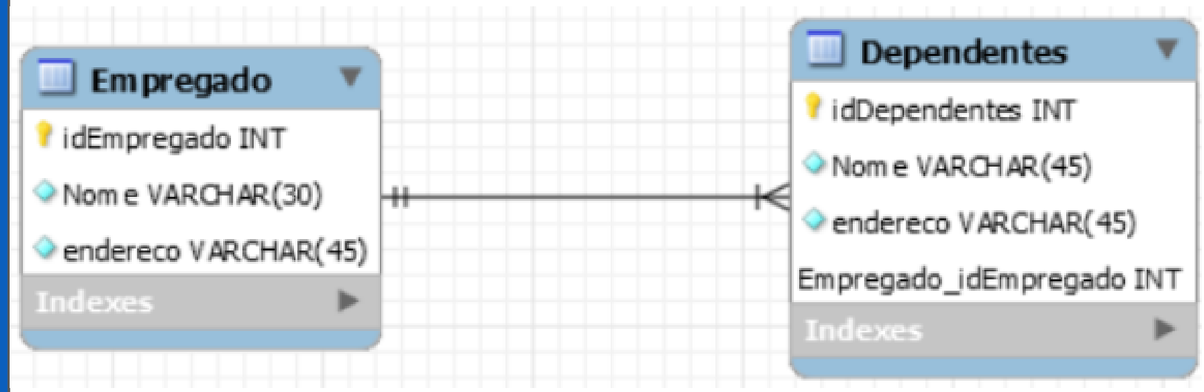



```sql
CREATE TABLE Departamento(
    idDepartamento INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(20),
    qtd_projeto int
);
```

```sql
CREATE TABLE Empregado(
    idEmpregado INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(20),
    endereco VARCHAR(20),
    empregado_departamento INT,
    CONSTRAINT fk_empregado FOREIGN KEY (empregado_departamento)
        REFERENCES Departamento (idDepartamento) ON UPDATE CASCADE
);
```

Inatel

Relacionamento 1:N

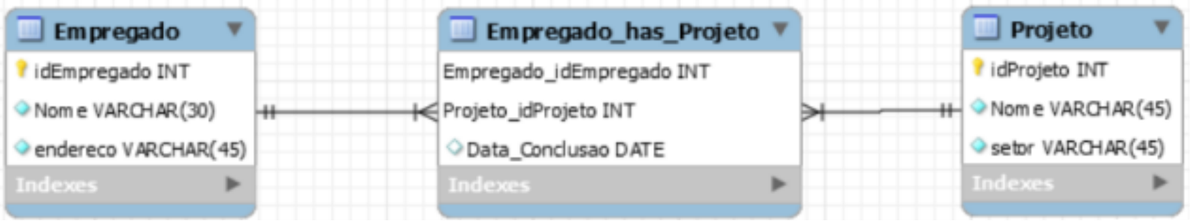



```sql
CREATE TABLE Empregado(
    idEmpregado INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(20),
    endereco VARCHAR(20)
);
```

```sql
CREATE TABLE Dependentes(
    idDependentes INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(45),
    endereco VARCHAR(45),
    empregado_idEmpregado INT,
    CONSTRAINT fk_empregado FOREIGN KEY (empregado_idEmpregado)
        REFERENCES Empregado (idEmpregado) ON DELETE CASCADE
);
```

Inatel

Relacionamento N:N




```sql
CREATE TABLE Empregado(
    idEmpregado INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(20),
    endereco VARCHAR(20)
);

CREATE TABLE Projeto(
    idProjeto INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(45),
    setor VARCHAR(45)
);

CREATE TABLE Empregado_has_Projeto(
    empregado_id INT,
    projeto_id INT,
    data_conclusao DATE,
    PRIMARY KEY(empregado_id, projeto_id),
    CONSTRAINT fk_empregado FOREIGN KEY (empregado_id) REFERENCES Empregado (idEmpregado),
    CONSTRAINT fk_projeto FOREIGN KEY (projeto_id) REFERENCES Projeto (idProjeto)
);
```

Inatel

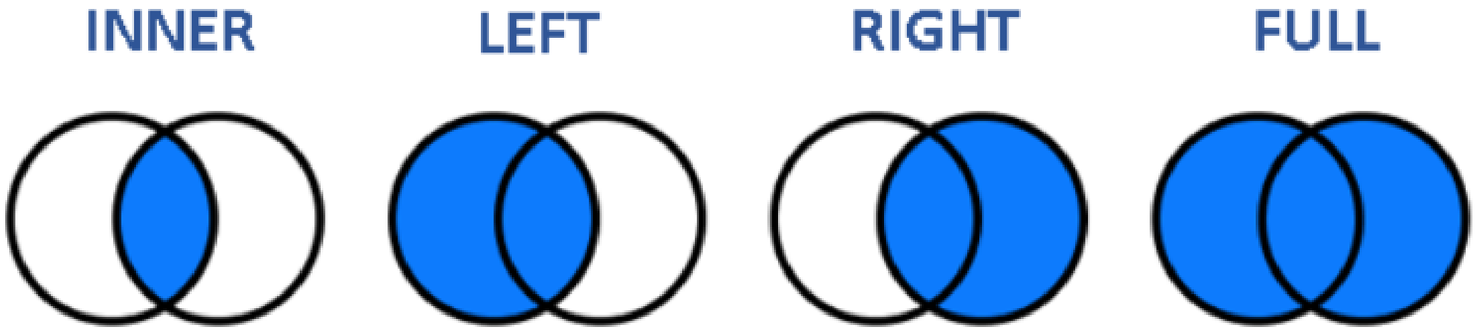
SELECT

SELECT <lista de atributos> FROM <lista de tabelas> WHERE <condição>;

- Regras opcionais do select:
  - **GROUP BY** - Para agrupar registros selecionados em grupos
  - **HAVING** - Para expressar a condição que deve satisfazer cada grupo
  - **ORDER BY** - Para ordenar os registros selecionados numa ordem específica
- Regras do Where:
  - **AND**, **NOT** e **OR**, juntamente com os operadores <,>,=,<>,<= e >=
  - (**NOT**) **BETWEEN** - Para especificar intervalos de valores
  - **LIKE** - Para comparar textos
  - **IN** - Para buscar dados de valores específicos dentro do WHERE
- Funções de agregação:
  - AVG - Para calcular a média dos valores de um campo determinado
  - COUNT - Para contar o número de registros da seleção
  - SUM - Para somar todos os valores de um atributo
  - MAX - Para devolver o valor mais alto de um atributo especificado
  - MIN - Para devolver o valor mais baixo de um atributo especificado

SELECT com JOIN

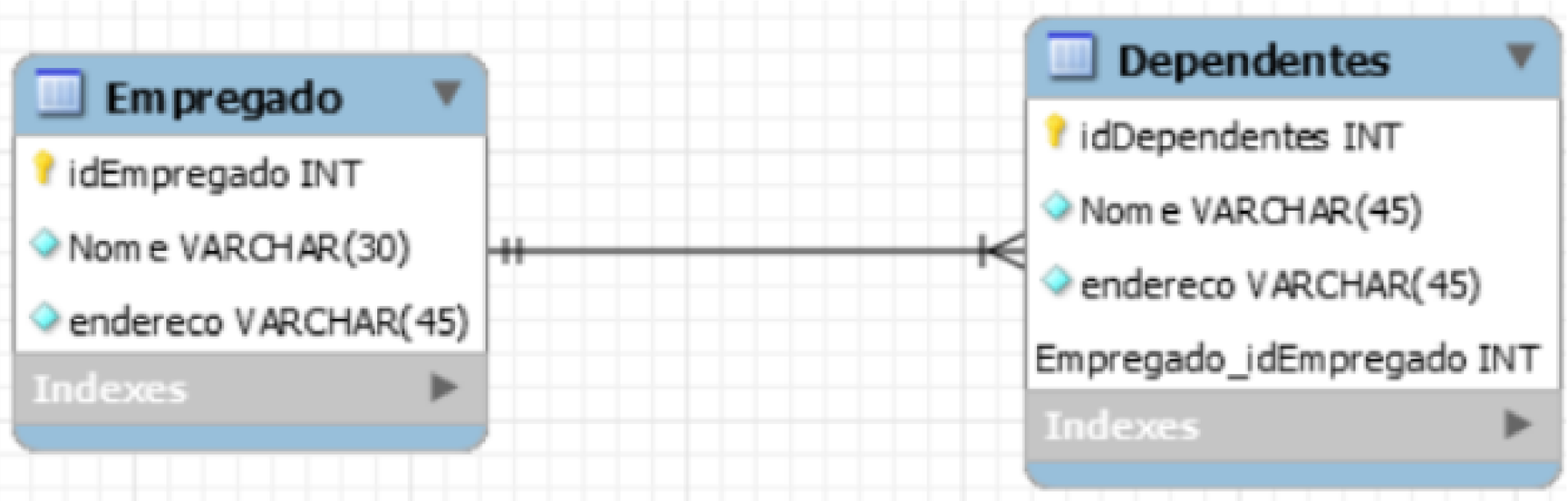
O JOIN é usado para fazer junções no SQL, ele é usado dentro do comando SELECT para juntar colunas de diferentes tabelas em um relacionamento, trazendo informações mais completas sobre algo



Inatel

SELECT com JOIN

Para relacionamentos 1:1 e 1:N



Inatel

SELECT com JOIN

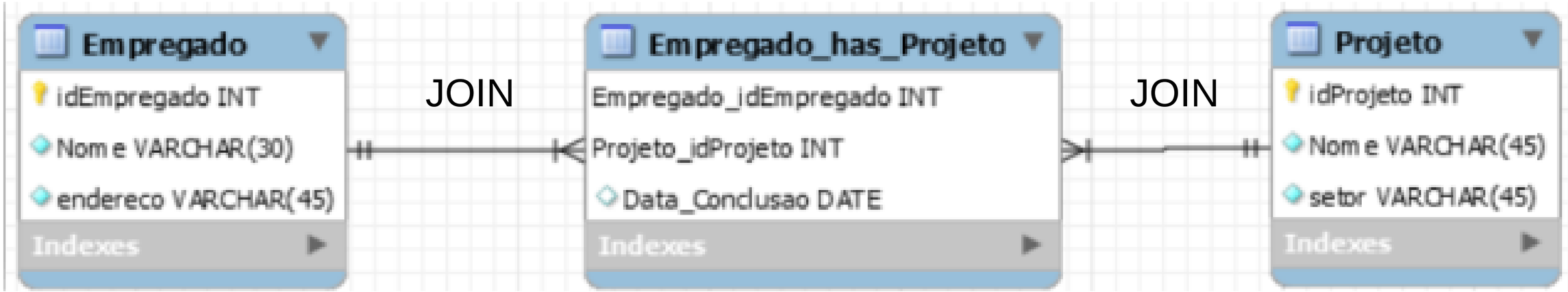
Para relacionamentos 1:1 e 1:N

```
SELECT Empregado.nome, Dependentes.nome FROM Empregado JOIN Dependentes;

SELECT E.nome, D.nome FROM Empregado AS E INNER JOIN Dependentes AS D;

SELECT E.nome, D.nome FROM Empregado AS E JOIN Dependentes AS D ON
D.empregado_idEmpregado = E.idEmpregado;

SELECT E.nome, D.nome FROM Empregado AS E JOIN Dependentes AS D ON
D.empregado_idEmpregado = E.idEmpregado WHERE E.idEmpregado = 1 ORDER BY E.nome;
```

Inatel

SELECT com JOIN

Para relacionamentos N:N



Inatel

SELECT com JOIN

Para relacionamentos N:N

```
SELECT E.nome, P.nome, EP.data_conclusao FROM Empregado AS E JOIN
Empregado_has_Projeto AS EP ON E.idEmpregado = EP.empregado_id JOIN Projeto AS P
ON P.idProjeto = EP.Projeto_id ORDER BY EP.data_conclusao;
```

Inatel