**UNIVERSIDADE SENAI JOINVILLE**

**Felipe Rafael Rothbarth**

**OT2-BD Linguagem SQL JOINS**

**JOINVILLE**

**Fichamento**

Relacionamento Entre Tabelas

**Chave Estrangeira**

Definição: A chave estrangeira (foreign key) é uma coluna ou conjunto de colunas em uma tabela que estabelece um vínculo com a chave primária de outra tabela, permitindo a criação de relacionamentos entre elas. Isso garante a integridade referencial, ou seja, a consistência dos dados entre as tabelas.

Quando é Gerada e Utilizada: Uma chave estrangeira é criada ao definir a estrutura de uma tabela para relacioná-la a outra. Ela é utilizada sempre que se deseja estabelecer um vínculo entre tabelas, como ao relacionar registros de clientes com suas respectivas ordens.

Exemplo:

CREATE TABLE clientes (

cliente\_id INT PRIMARY KEY,

nome VARCHAR(50)

);

CREATE TABLE pedidos (

pedido\_id INT PRIMARY KEY,

cliente\_id INT,

FOREIGN KEY (cliente\_id) REFERENCES clientes(cliente\_id)

);

**Tipos de Relacionamentos Entre Tabelas**

**Relacionamento Um para Um (1:1)**

Como Identificar: Este tipo de relacionamento ocorre quando um registro de uma tabela está relacionado a um único registro em outra tabela. Em um banco de dados, isso pode ser útil para separar informações que precisam ser armazenadas de forma modular.

Como Definir o Local da Chave Estrangeira: A chave estrangeira pode ser colocada em qualquer uma das tabelas, mas normalmente é na tabela que contém os detalhes da relação.

Exemplo:

CREATE TABLE pessoa (

pessoa\_id INT PRIMARY KEY,

nome VARCHAR(50)

);

CREATE TABLE endereco (

endereco\_id INT PRIMARY KEY,

pessoa\_id INT UNIQUE,

endereco\_completo VARCHAR(100),

FOREIGN KEY (pessoa\_id) REFERENCES pessoa(pessoa\_id)

);

**Relacionamento Um para Vários (1)**

Como Identificar: Este relacionamento é usado quando um registro de uma tabela pode estar relacionado a múltiplos registros em outra tabela. Por exemplo, um cliente pode ter vários pedidos, mas cada pedido pertence a um único cliente.

Como Definir o Local da Chave Estrangeira: A chave estrangeira é colocada na tabela que representa os "muitos" registros do relacionamento.

Exemplo:

CREATE TABLE clientes (

cliente\_id INT PRIMARY KEY,

nome VARCHAR(50)

);

CREATE TABLE pedidos (

pedido\_id INT PRIMARY KEY,

cliente\_id INT,

FOREIGN KEY (cliente\_id) REFERENCES clientes(cliente\_id)

);

**Relacionamento Vários para Vários (N)**

Como Identificar: Este relacionamento ocorre quando múltiplos registros em uma tabela podem estar relacionados a múltiplos registros em outra tabela. Como o MySQL não permite relações diretas N, é necessária uma tabela intermediária para realizar essa ligação.

Regra para Definir este Tipo de Relacionamento: Cria-se uma tabela associativa com chaves estrangeiras que apontam para as tabelas relacionadas.

Impacto nas Tabelas: Ao criar este relacionamento, a tabela intermediária armazena as chaves estrangeiras, possibilitando a relação entre as duas tabelas originais.

Exemplo:

CREATE TABLE alunos (

aluno\_id INT PRIMARY KEY,

nome VARCHAR(50)

);

CREATE TABLE cursos (

curso\_id INT PRIMARY KEY,

nome VARCHAR(50)

);

CREATE TABLE aluno\_curso (

aluno\_id INT,

curso\_id INT,

PRIMARY KEY (aluno\_id, curso\_id),

FOREIGN KEY (aluno\_id) REFERENCES alunos(aluno\_id),

FOREIGN KEY (curso\_id) REFERENCES cursos(curso\_id)

);

**JOIN (SQL)**

**O Que é o JOIN e Sua Função em um Banco de Dados Relacional?**

O JOIN é uma operação SQL que permite combinar registros de duas ou mais tabelas com base em uma condição. Ele é utilizado para consultar e exibir dados relacionados de tabelas diferentes de maneira consolidada, essencial em bancos de dados relacionais para obter informações complexas.

**Tipos de JOIN e Suas Características**

INNER JOIN: Retorna registros que possuem correspondência em ambas as tabelas.

SELECT clientes.nome, pedidos.pedido\_id

FROM clientes

INNER JOIN pedidos ON clientes.cliente\_id = pedidos.cliente\_id;

LEFT JOIN (ou LEFT OUTER JOIN): Retorna todos os registros da tabela à esquerda e os registros correspondentes da tabela à direita. Se não houver correspondência, retorna NULL para as colunas da tabela à direita.

SELECT clientes.nome, pedidos.pedido\_id

FROM clientes

LEFT JOIN pedidos ON clientes.cliente\_id = pedidos.cliente\_id;

RIGHT JOIN (ou RIGHT OUTER JOIN): Retorna todos os registros da tabela à direita e os registros correspondentes da tabela à esquerda. Se não houver correspondência, retorna NULL para as colunas da tabela à esquerda.

SELECT clientes.nome, pedidos.pedido\_id

FROM clientes

RIGHT JOIN pedidos ON clientes.cliente\_id = pedidos.cliente\_id;

FULL JOIN (ou FULL OUTER JOIN): Retorna todos os registros das duas tabelas e preenche com NULL onde não houver correspondência. O MySQL não possui um comando FULL JOIN nativo, mas podemos simulá-lo com a união de LEFT JOIN e RIGHT JOIN.

SELECT clientes.nome, pedidos.pedido\_id

FROM clientes

LEFT JOIN pedidos ON clientes.cliente\_id = pedidos.cliente\_id

UNION

SELECT clientes.nome, pedidos.pedido\_id

FROM clientes

RIGHT JOIN pedidos ON clientes.cliente\_id = pedidos.cliente\_id;

**REFERENCIAS BIBLIOGRAFICAS**  
HEUSER, Carlos A. Projeto de Banco de Dados. 5. ed. Porto Alegre: Bookman, 2009.

DATE, C. J. Introdução a Sistemas de Banco de Dados. 8. ed. São Paulo: Pearson Addison Wesley, 2004.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistemas de Banco de Dados. 6. ed. São Paulo: McGraw-Hill, 2011.