**FICHAMENTO**

RELACIONAMENTO ENTRE TABELAS:

 Chave estrangeira

○ Definição.

Para Asakura (2022), uma chave estrangeira é composta por uma ou mais colunas de uma tabela que fazem referência à chave primária de outra tabela, estabelecendo uma relação entre ambas. Não é necessária a aplicação de configurações específicas ou códigos adicionais para formalizar a designação de uma chave estrangeira; basta que a coluna contenha valores compatíveis com a chave primária referenciada, garantindo a integridade relacional dos dados.

○ Quando é gerada e quando é utilizada.

O autor afirma que uma chave estrangeira é definida durante a criação ou modificação da estrutura de uma tabela, a partir do comando o comando FOREIGN KEY.

As chaves estrangeiras são utilizadas sempre que for necessário garantir a integridade referencial, assegurando que os valores em uma ou mais colunas de uma tabela dependente (ou filha) correspondam a valores existentes na chave primária de uma tabela relacionada (ou pai).

 Tipos de relacionamentos entre tabelas

○ Relacionamento Um para Um (1:1)

■ Como identificar este relacionamento.

Asakura (2010) continua, afirmando que para se identificar este tipo de relacionamento, ambas as tabelas envolvidas devem conter uma coluna que funcione como chave primária. Em seguida, é criada uma terceira tabela para relacionar essas duas. Nessa terceira tabela, as colunas que são chaves primárias nas duas tabelas originais tornam-se chaves estrangeiras. Essas duas chaves estrangeiras formam uma chave primária composta na terceira tabela, estabelecendo assim o relacionamento 1:1.

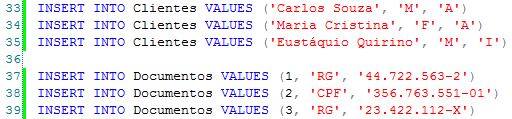
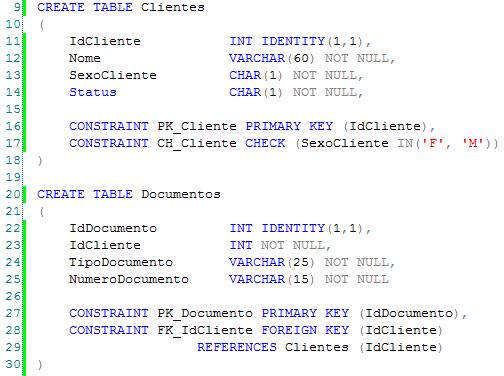
■ Como definir o local da chave estrangeira (em qual tabela é identificada).

Para Asakura (2010), a chave estrangeira é definida na terceira tabela, responsável por relacionar as duas tabelas originais. Nessa terceira tabela, as colunas que são chaves primárias nas duas primeiras tabelas tornam-se chaves estrangeiras.

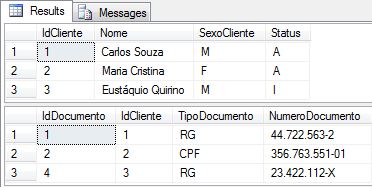
■ Exemplos

Considere-se as tabelas *Clientes* e *Documentos*, em que cada um dos clientes pode ter apenas um documento (considerando que nesta tabela um mesmo cliente não pode ter mais de um documento), assim como cada um dos documentos só pode pertencer a apenas um cliente.

Verificar nas imagens abaixo a criação das tabelas, das **constraints** e inserção de alguns registros:



Na tabela *Documentos* foi criada uma *Foreign Key* (chave estrangeira) se relacionando com a *Primary Key* da tabela *Clientes*. Desta forma, as tabelas ficarão assim:



Assim obtém-se o relacionamento de **um-para-um.**

○ Relacionamento Um para Vários (1:N)

■ Como identificar este relacionamento.

Asakura estabelece ainda que, neste relacionamento que é um dos mais comuns hoje em dia, cada elemento da entidade “A” pode ter um relacionamento com vários elementos da entidade “B”. Em contrapartida, cada um dos elementos da entidade B pode estar relacionado a apenas um elemento da A.

■ Como definir o local da chave estrangeira (em qual tabela é identificada).

De acordo com o autor, a chave estrangeira é definida na tabela cuja entidade pode ter múltiplos registros relacionados a um único elemento da primeira tabela. Assim, essa segunda tabela conterá uma coluna com chave estrangeira que referencia a chave primária da primeira tabela, garantindo que cada registro esteja associado a um único cliente.

■ Exemplos

Considerando a tabela *Clientes* criada no exemplo acima, criemos a tabela *Telefones* com o código a seguir, já que neste caso cada cliente pode ter vários telefones, mas cada telefone pode pertencer a um, e somente um cliente:

CREATE TABLE Telefones

(

IdTelefone INT IDENTITY(1,1),

IdCliente INT NULL,

TipoTelefone VARCHAR(15) NOT NULL,

NumeroTelefone VARCHAR(15) NOT NULL

CONSTRAINT PK\_Telefone PRIMARY KEY (IdTelefone),

CONSTRAINT FK\_Telefones\_IdCliente FOREIGN KEY (IdCliente)

REFERENCES Clientes (IdCliente)

)

--Insiro alguns registros na tabela Telefones

INSERT INTO Telefones VALUES (1, 'Residencial', '3243-2015')

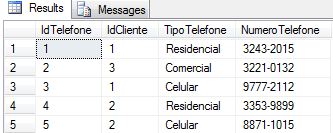
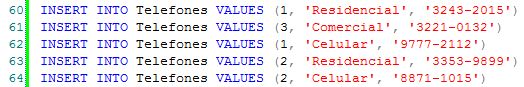
INSERT INTO Telefones VALUES (3, 'Comercial', '3221-0132')

INSERT INTO Telefones VALUES (1, 'Celular', '9777-2112')

INSERT INTO Telefones VALUES (2, 'Residencial', '3353-9899')

INSERT INTO Telefones VALUES (2, 'Celular', '8871-1015')

Nas imagens a seguir, são inseridos alguns registros e o resultado da consulta a ela:



Assim temos o relacionamento de **um-para-muitos.**

○ Relacionamento Vários para Vários (N:N)

■ Como identificar este relacionamento.

O relacionamento muitos-para-muitos é identificado quando uma entidade pode se associar a múltiplos registros de outra entidade, e vice-versa. Nesse tipo de relação, não é possível armazenar diretamente essas associações em apenas duas tabelas, pois cada registro de ambas as entidades pode ter vários vínculos entre si.

■ Qual a regra para definir estes tipo de relacionamento.

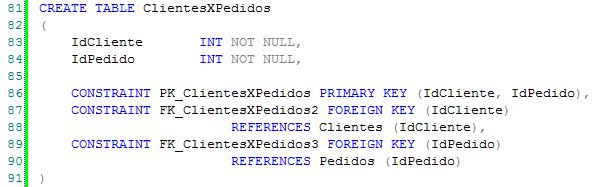
De acordo com o autor, a regra essencial para definir um relacionamento muitos-para-muitos consiste na criação de uma tabela intermediária que conecta as duas entidades principais. Esta tabela adicional armazena exclusivamente as chaves primárias das tabelas originais, que também funcionam como chaves estrangeiras. Essa estrutura garante a integridade referencial e possibilita a associação múltipla entre os registros das duas entidades.

■ O que acontece com as tabelas quando esse relacionamento é criado.

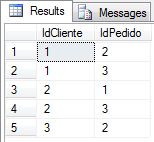
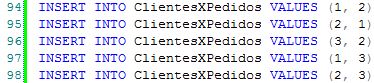
Asakura (2010) explica que quando um relacionamento muitos-para-muitos é criado, a estrutura do banco de dados se organiza em três tabelas: duas representando as entidades principais e uma tabela intermediária que conecta essas entidades.

■ Exemplos

Criando a terceira tabela, denominada ClientesXPedidos, responsável por interligá-lasNas imagens a seguir, o autor insere registros (ela conterá apenas as chaves das duas outras tabelas) e posteriormente faz a consulta:



Na imagem acima os mesmos campos, IdCliente e IdPedido, são chaves primárias e também são chaves estrangeiras, criando assim o relacionamento entre as tabelas *Cliente* e *Pedido*.



Desta forma se dá o relacionamento muitos-para-muitos.

* JOIN (SQL)

○ O que é o Join e qual sua função em um banco de dados relacional?

De acordo com W3 Schools, uma cláusula JOIN é utilizada para combinar linhas de duas ou mais tabelas com base em uma coluna relacionada entre elas.

○ Quais são os tipos de Join e suas características.

Segundo os autores, os diferentes tipos de JOINs em SQL são:

**(INNER) JOIN**: Retorna os registros que possuem valores correspondentes em ambas as tabelas.

**LEFT (OUTER) JOIN**: Retorna todos os registros da tabela à esquerda e os registros correspondentes da tabela à direita.

**RIGHT (OUTER) JOIN**: Retorna todos os registros da tabela à direita e os registros correspondentes da tabela à esquerda.

**FULL (OUTER) JOIN**: Retorna todos os registros quando há correspondência em pelo menos uma das tabelas, seja à esquerda ou à direita.

○ Apresente exemplos de código utilizando essas diferentes junções.

**INNER JOIN**:

SELECT C.NOMECARGO [CARGO], F.NOMEFUNCIONARIO AS [FUNCIONÁRIO], F.SALARIOFUNCIONARIO AS [SALÁRIO]

FROM CARGO AS C INNER JOIN FUNCIONARIO AS F ON C.IDCARGO = F.IDCARGO

**LEFT (OUTER) JOIN**

SELECT E.Nome, C.Nome\_Curso FROM Estudantes AS E LEFT JOIN Cursos AS C ON E.Curso\_ID = C.ID\_Curso;

**RIGHT (OUTER) JOIN**

SELECT E.Nome, C.Nome\_Curso FROM Estudantes AS E RIGHT JOIN Cursos AS C ON E.Curso\_ID = C.ID\_Curso;

**FULL (OUTER) JOIN**

SELECT E.Nome, C.Nome\_Curso FROM Estudantes AS E FULL JOIN Cursos AS C ON E.Curso\_ID = C.ID\_Curso;

**REFERÊNCIAS**

ASAKURA, Oscar Kenjiro N.. **Chave Estrangeira x Chave Primária**: qual é a diferença?. Qual é a Diferença?. 2022. Disponível em: https://www.linkedin.com/pulse/chave-estrangeira-x-prim%C3%A1ria-qual-%C3%A9-diferen%C3%A7a-asakura/. Acesso em: 29 out. 2024.

DEVMEDIA (Brasil). **Exemplo Prático**: regras e tipos de relacionamentos - sql server 2008 - parte 3. Regras e Tipos de Relacionamentos - SQL Server 2008 - Parte 3. 2010. Disponível em: https://www.devmedia.com.br/exemplo-pratico-regras-e-tipos-de-relacionamentos-sql-server-2008-parte-3/18409. Acesso em: 29 out. 2024.

W3 SCHOOLS. **SQL Joins**. Disponível em: https://www.w3schools.com/sql/sql\_join.asp. Acesso em: 29 out. 2024.