RELACIONAMENTO ENTRE TABELAS:

 Chave estrangeira

○ Definição.

○ Quando é gerada e quando é utilizada.

Chave Estrangeira (Foreign Key):

Definição: Uma chave estrangeira é um campo ou conjunto de campos em uma tabela que faz referência a uma chave primária ou única em outra tabela. Isso estabelece um vínculo entre as duas tabelas, permitindo a criação de relacionamentos entre elas.

Quando é Gerada: A chave estrangeira é definida quando você cria a tabela que a contém, especificando o campo que será a chave estrangeira e a tabela e campo que ele referencia como chave primária ou única.

Quando é Utilizada: A chave estrangeira é utilizada durante operações de consulta e manipulação de dados para garantir a integridade referencial. Ela ajuda a manter a consistência dos dados, garantindo que não haja referências a dados inexistentes na tabela relacionada.

 Tipos de relacionamentos entre tabelas

○ Relacionamento Um para Um (1:1)

■ Como identificar este relacionamento.

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

■ Exemplos

Relacionamento Um para Um (1:1):

Como Identificar Este Relacionamento:

Um relacionamento Um para Um é identificado quando duas entidades têm uma relação direta e exclusiva entre si, ou seja, cada instância de uma entidade está associada a no máximo uma instância da outra entidade, e vice-versa.

Como Definir o Local da Chave Estrangeira:

A chave estrangeira em um relacionamento 1:1 é definida em uma das tabelas que representa a entidade secundária ou dependente. Geralmente, você pode escolher qual tabela terá a chave estrangeira com base na lógica de negócios e na estruturação dos dados. Por exemplo, em um relacionamento entre Cliente e Detalhes do Cliente (onde cada cliente tem um conjunto específico de detalhes), a chave estrangeira poderia ser definida na tabela Detalhes do Cliente, referenciando a chave primária da tabela Cliente.

Exemplos:

Relacionamento Cliente e Detalhes do Cliente:

Tabela Cliente: id\_cliente (chave primária), nome, sobrenome

Tabela Detalhes do Cliente: id\_detalhes\_cliente (chave primária), id\_cliente (chave estrangeira referenciando Cliente), endereço, telefone

Neste exemplo, o relacionamento é 1:1, onde cada cliente tem um conjunto único de detalhes armazenados na tabela Detalhes do Cliente. A chave estrangeira id\_cliente na tabela Detalhes do Cliente faz referência à chave primária id\_cliente na tabela Cliente.

Relacionamento Produto e Detalhes do Produto:

Tabela Produto: id\_produto (chave primária), nome\_produto, preço

Tabela Detalhes do Produto: id\_detalhes\_produto (chave primária), id\_produto (chave estrangeira referenciando Produto), descrição, especificações

Neste caso, cada produto tem um conjunto específico de detalhes na tabela Detalhes do Produto, e a chave estrangeira id\_produto na tabela Detalhes do Produto faz referência à chave primária id\_produto na tabela Produto.

Esses são exemplos simples de como definir e identificar um relacionamento Um para Um (1:1) em um banco de dados relacional, juntamente com a localização da chave estrangeira para estabelecer a relação entre as tabelas.

○ Relacionamento Um para Vários (1:N)

■ Como identificar este relacionamento.

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

■ Exemplos

Relacionamento Um para Vários (1:N):

Como Identificar Este Relacionamento:

Um relacionamento Um para Vários (1:N) é identificado quando uma linha em uma tabela está associada a várias linhas em outra tabela, mas cada linha na segunda tabela está associada a apenas uma linha na primeira tabela.

Como Definir o Local da Chave Estrangeira:

A chave estrangeira em um relacionamento 1:N é definida na tabela que representa o "N" da relação, ou seja, a tabela que tem várias linhas associadas à única linha na outra tabela. A chave estrangeira é um campo na tabela "N" que faz referência à chave primária ou única na tabela "1".

Exemplos:

Relacionamento Cliente e Pedidos:

Tabela Cliente: id\_cliente (chave primária), nome, sobrenome

Tabela Pedidos: id\_pedido (chave primária), id\_cliente (chave estrangeira referenciando Cliente), data\_pedido, valor

Neste exemplo, cada cliente pode ter vários pedidos associados a ele na tabela Pedidos. A chave estrangeira id\_cliente na tabela Pedidos faz referência à chave primária id\_cliente na tabela Cliente.

Relacionamento Departamento e Funcionários:

Tabela Departamento: id\_departamento (chave primária), nome\_departamento

Tabela Funcionários: id\_funcionario (chave primária), nome\_funcionario, id\_departamento (chave estrangeira referenciando Departamento), cargo

Aqui, um departamento pode ter vários funcionários trabalhando nele na tabela Funcionários. A chave estrangeira id\_departamento na tabela Funcionários faz referência à chave primária id\_departamento na tabela Departamento.

Relacionamento Post e Comentários:

Tabela Post: id\_post (chave primária), título\_post, conteúdo\_post

Tabela Comentários: id\_comentario (chave primária), id\_post (chave estrangeira referenciando Post), texto\_comentario, data\_comentario

Neste exemplo, um post pode ter vários comentários associados a ele na tabela Comentários. A chave estrangeira id\_post na tabela Comentários faz referência à chave primária id\_post na tabela Post.

Estes são exemplos práticos de como identificar, definir a localização da chave estrangeira e implementar um relacionamento Um para Vários (1:N) em um banco de dados relacional.

○ Relacionamento Vários para Vários (N:N)

■ Como identificar este relacionamento.

■ Qual a regra para definir estes tipo de relacionamento.

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

■ Exemplos

Relacionamento Vários para Vários (N:N):

Como Identificar Este Relacionamento:

Um relacionamento Vários para Vários (N:N) ocorre quando várias linhas em uma tabela estão associadas a várias linhas em outra tabela. Esse tipo de relacionamento é identificado quando há uma associação muitos-para-muitos entre as entidades representadas pelas tabelas.

Regra para Definir Este Tipo de Relacionamento:

A regra para definir um relacionamento N:N em um banco de dados relacional é utilizar uma tabela de junção (também conhecida como tabela associativa ou de ligação) que conecta as duas tabelas principais envolvidas no relacionamento. Essa tabela de junção contém chaves estrangeiras que fazem referência às chaves primárias das tabelas principais.

O Que Acontece com as Tabelas Quando Este Relacionamento É Criado:

Quando um relacionamento N:N é criado, uma tabela de junção é introduzida para representar as associações entre as entidades das tabelas principais. Isso significa que as tabelas principais não têm diretamente uma chave estrangeira que aponta diretamente para a outra tabela, mas sim através da tabela de junção.

Exemplos:

Relacionamento Alunos e Disciplinas:

Tabela Alunos: id\_aluno (chave primária), nome\_aluno

Tabela Disciplinas: id\_disciplina (chave primária), nome\_disciplina

Tabela Matrícula: id\_matricula (chave primária), id\_aluno (chave estrangeira referenciando Alunos), id\_disciplina (chave estrangeira referenciando Disciplinas), nota\_final

Neste exemplo, um aluno pode estar matriculado em várias disciplinas, e uma disciplina pode ter vários alunos matriculados. A tabela de junção Matrícula representa essa relação muitos-para-muitos entre Alunos e Disciplinas.

Relacionamento Autores e Livros:

Tabela Autores: id\_autor (chave primária), nome\_autor

Tabela Livros: id\_livro (chave primária), título\_livro

Tabela Autoria: id\_autoria (chave primária), id\_autor (chave estrangeira referenciando Autores), id\_livro (chave estrangeira referenciando Livros)

Aqui, um autor pode escrever vários livros, e um livro pode ter vários autores. A tabela de junção Autoria representa o relacionamento muitos-para-muitos entre Autores e Livros.

Em resumo, um relacionamento Vários para Vários (N:N) é resolvido através de uma tabela de junção que conecta as tabelas principais envolvidas no relacionamento, permitindo a representação adequada das associações muitos-para-muitos entre as entidades.

Agora que você entendeu um pouco mais sobre o relacionamento entre tabelas, vamos pesquisar

agora sobre as formas de manipulação de dados, outras formas de consultas de informações através

das instruções SQL específicas para cada tipo de informação que deseja obter . Segue o

desenvolvimento do fichamento.

 JOIN (SQL)

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

○ Quais são os tipos de Join e suas características.

○ Apresente exemplos de código utilizando essas diferentes junções.

JOIN (SQL):

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

O JOIN é uma operação em SQL que combina linhas de duas ou mais tabelas com base em uma condição relacionada entre elas. Sua função em um banco de dados relacional é permitir a obtenção de informações relacionadas de diferentes tabelas em uma única consulta, facilitando a análise e manipulação dos dados.

Tipos de Join e suas características:

Existem diferentes tipos de JOIN em SQL, cada um com suas características específicas:

INNER JOIN: Retorna apenas as linhas que têm correspondências nas duas tabelas sendo unidas. Ou seja, se não houver correspondência entre as tabelas, as linhas não serão incluídas no resultado.

LEFT JOIN (ou LEFT OUTER JOIN): Retorna todas as linhas da tabela à esquerda (tabela A) e as linhas correspondentes da tabela à direita (tabela B). Se não houver correspondência na tabela B, as colunas da tabela B terão valores nulos.

RIGHT JOIN (ou RIGHT OUTER JOIN): Retorna todas as linhas da tabela à direita (tabela B) e as linhas correspondentes da tabela à esquerda (tabela A). Se não houver correspondência na tabela A, as colunas da tabela A terão valores nulos.

FULL JOIN (ou FULL OUTER JOIN): Retorna todas as linhas das duas tabelas, combinando-as onde houver correspondência e preenchendo com valores nulos onde não houver correspondência.

CROSS JOIN: Retorna o produto cartesiano das duas tabelas, ou seja, combina cada linha da primeira tabela com cada linha da segunda tabela, resultando em todas as combinações possíveis.

Exemplos de código utilizando essas diferentes junções:

INNER JOIN:

sql

Copy code

SELECT clientes.nome, pedidos.numero\_pedido

FROM clientes

INNER JOIN pedidos ON clientes.id\_cliente = pedidos.id\_cliente;

LEFT JOIN:

sql

Copy code

SELECT clientes.nome, pedidos.numero\_pedido

FROM clientes

LEFT JOIN pedidos ON clientes.id\_cliente = pedidos.id\_cliente;

RIGHT JOIN:

sql

Copy code

SELECT clientes.nome, pedidos.numero\_pedido

FROM clientes

RIGHT JOIN pedidos ON clientes.id\_cliente = pedidos.id\_cliente;

FULL JOIN:

sql

Copy code

SELECT clientes.nome, pedidos.numero\_pedido

FROM clientes

FULL JOIN pedidos ON clientes.id\_cliente = pedidos.id\_cliente;

CROSS JOIN:

sql

Copy code

SELECT clientes.nome, produtos.nome\_produto

FROM clientes

CROSS JOIN produtos;

Estes são exemplos básicos de como usar diferentes tipos de JOIN em SQL para combinar informações de várias tabelas em consultas. As características e resultados de cada tipo de JOIN podem variar dependendo da estrutura das tabelas e das condições de junção especificadas.