FAESA CENTRO UNIVERSITÁRIO

Linguagem SQL com junção

Apresentação

Ao longo do tempo, o uso de tecnologias, tanto nas organizações quanto na vida das pessoas, passou por grandes evoluções, como a formulação e o aprimoramento de sistemas e até o surgimento de dispositivos cada vez menores com acesso a grandes quantidades de dados. Algumas ferramentas de manipulação de dados merecem destaque, como é o caso da *Structured Query Language* (SQL). Ela consiste em uma linguagem de consulta estruturada utilizada nos sistemas de gerenciamento de banco de dados (SGBD). Por meio dela, é possível realizar diferentes consultas, atualizações, exclusões de informações em um banco de dados.

Nesta Unidade de Aprendizagem, você vai estudar algumas das funcionalidades da SQL, como consultas simples, manipulação de dados, operadores relacionais, operadores lógicos, operadores de conjuntos e junções de tabelas.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Enumerar as instruções DML e os operadores relacionais em SQL.
- Desenvolver consultas e manipulações de dados com SQL.
- Produzir consultas utilizando operadores de conjuntos e junções de tabelas.

Infográfico

A SQL possibilita o uso de três construtores de manipulação de conjunto principais, que são os operadores em conjuntos: *UNION* (união), *INTERSECT* (interseção) e *EXCEPT* (exceção ou diferença). Para combinar informações de duas ou mais relações na SQL também é possível utilizar junções. Entre elas estão: *INNER JOIN*, *OUTER JOIN*, *LEFT OUTER JOIN*, *RIGHT OUTER JOIN*.

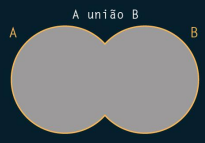
No Infográfico a seguir, você vai ver uma representação gráfica dessas manipulações de conjunto e junções para facilitar sua compreensão.

CONJUNTOS E JUNÇÕES EM SQL

Veja as representações gráficas das principais manipulações de conjuntos e junções de relações entre tabelas de banco de dados relacionadas ao seu código.

UNION

Neste caso, a consulta resulta em dados presentes nas tabelas A e B, formando a união.



INTERSECT

A consulta resulta somente em dados ou condições que estejam representadas nas tabelas A e B concomitantemente. Seu uso remove duplicatas, mas seu resultado pode ser nulo, caso as condições das cláusulas da consulta não sejam atendidas.

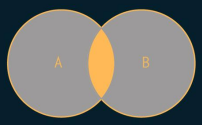
EXCEPT

Aqui, somente a exceção que interessa para a consulta. Assim, somente os dados ou condições presentes na tabela A que não estejam também B formam o resultado.

INNER JOIN

A consulta resulta somente em dados, condições ou atributos que estejam representadas nas tabelas A e B concomitantemente, possibilitando a ligação das tabelas por chaves primárias e secundárias.

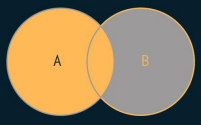
Seu uso mantém duplicatas caso haja coincidência de identificadores entre as tabelas. Não retorna resultados nulos.



LEFT OUTER JOIN

A consulta retorna todos os registros da tabela esquerda (tabela A) e as correspondências que existirem com a tabela direita (tabela B).

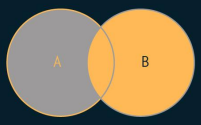
Se a consulta for por nomes (A) e telefones (B), por exemplo, resultará em todos os nomes da tabela A juntados aos telefones disponíveis na tabela B, portanto podem resultar nomes sem um telefone (nulo) correspondente.



RIGHT OUTER JOIN

A consulta retorna todos os registros da tabela direita (tabela B) e as correspondências que existirem com a tabela esquerda (tabela A).

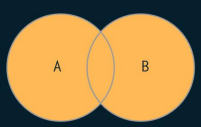
Se a consulta for por nomes (A) e telefones (B), por exemplo, resultará em todos os telefones da tabela B juntados aos nomes disponíveis na tabela A, portanto podem resultar telefones sem um nome (nulo) correspondente.



OUTER JOIN

Nesse caso, a consulta retorna todos os registros de ambas as tabelas.

Se a consulta for por nomes (A) e telefones (B), por exemplo, resultará em todos os nomes e telefones das duas tabelas, portanto podem resultar tanto telefones sem um nome (nulo) correspondente quanto nomes sem um telefone (nulo) correspondente.



Essas são apenas algumas das possibilidades de utilização das manipulações de conjuntos e das junções, existem ainda muitas outras definições possíveis.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

Bancos de dados estão presentes nas vidas das pessoas que utilizam algum tipo de aplicativo ou *software* com diferentes funções. Isso quer dizer que muita gente se relaciona com bancos de dados sem saber que eles existem, ou mesmo como eles podem ser organizados. Diferentemente dessas pessoas, você está aqui para conhecer algumas informações importantes sobre a aplicação e manipulação de bancos de dados que utilizam SQL. Para isso, irá estudar sobre inclusão e exclusão de dados e informações, assim como a realização de consultas do que já está armazenado. Você sabia que diferentes consultas de dados podem ser relacionadas e assim construir uma nova tabela, chamada tabela de junção? Entre as vantagens da utilização das junções, está a facilidade de encontrar resultados, sem a necessidade de misturar diferentes condições de seleção nas cláusulas de pesquisa.

No capítulo Linguagem SQL com junção, da obra *Banco de dados*, base teórica desta Unidade de Aprendizagem, você vai conhecer as instruções DML e os operadores relacionais em SQL, ver as consultas e manipulação de dados com SQL e aprender a aplicar os operadores de conjuntos e junções de tabelas.

Boa leitura.

BANCO DE DADOS

Roni Francisco Pichetti

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Linguagem SQL com junção

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Enumerar as instruções DML e os operadores relacionais em SQL.
- Desenvolver consultas e manipulações de dados com SQL.
- Produzir consultas utilizando operadores de conjuntos e junções de tabelas.

Introdução

Os bancos de dados podem ser descritos, de forma simplificada, como uma coleção de dados organizados e armazenados logicamente, possibilitando que os dados sejam acessados de acordo com sua localização, como livros em uma biblioteca. E, assim como livros, esses dados periodicamente precisam ser atualizados ou, até mesmo, descartados. Por esse motivo, os bancos de dados necessitam de sistemas de gerenciamento de bancos de dados (SGBDs), que trabalham como um bibliotecário, tendo o controle sobre tudo o que acontece em seu acervo. Para isso, esses sistemas utilizam linguagens de consulta, como a linguagem de consulta estruturada (SQL, do inglês *structured query language*).

Neste capítulo, você vai estudar os conceitos básicos e a utilização da linguagem de manipulação de dados (DML, do inglês *data manipulation language*) e de operadores relacionais em SQL. Além disso, você vai verificar exemplos da criação de consultas em SQL e vai conferir as instruções utilizadas para a manipulação de dados, a criação de conjuntos e a junção de tabelas.

1 Instruções DML e operadores relacionais em SQL

Quanto mais fácil for o acesso a uma informação em um banco de dados, maior a utilidade e a importância dele para seus usuários. Os SGBDs possuem uma grande variedade de relações disponíveis para que os dados possam ser encontrados, o que influencia na sua popularidade. Para isso, utiliza-se de consultas, elaboradas por linguagens específicas. Cabe lembrar que a eficiência da realização de uma consulta tem relação direta com a maneira como os dados estão armazenados. O SGBD permite que os usuários criem, modifiquem e consultem dados por meio de uma DML. Dessa forma, a DML dispõe das opções de consulta, inserção, exclusão e modificação de dados, conforme lecionam Ramarkrishnan e Gehrke (2011).

Existem dois tipos principais de DMLs: as de alto nível (ou não procedurais) e as de nível baixo (ou procedurais). As **DMLs de alto nível** são utilizadas no banco de dados na especificação de operações complexas de maneira sucinta. Elas podem ser empregadas de forma interativa, por meio de um monitor ou terminal, ou ainda embutidas em uma linguagem de programação de propósito geral. Já as **DMLs de baixo nível** recuperam objetos de bancos de dados e realizam seu processamento separadamente. Para isso, precisam necessariamente ser embutidas em uma linguagem de programação de propósito geral. Por esse motivo, as DMLs de nível baixo também podem ser chamadas de *record-at-a-time* (um registro por vez), e as de alto nível, de *set-at-a-time* (um conjunto por vez), segundo Elmasri e Navathe (2011).

Quando os comandos DML, sejam de alto ou baixo nível, são embutidos em um programa de linguagem de propósito geral, a linguagem é chamada de **linguagem hospedeira** (*host language*), e a DML, de **sublinguagem de dados**. Já nos casos em que a linguagem de alto nível é utilizada de forma interativa, ela é chamada de **linguagem de consulta** (*query language*). Comandos como "recuperar" e "atualizar" da DML de alto nível podem ser aplicados interativamente; nesses casos, são considerados parte da linguagem de consulta. Usuários finais casuais geralmente fazem uso de linguagem de consulta de alto nível na especificação de seus pedidos, enquanto usuários iniciantes necessitam de interfaces amigáveis para interagir com um banco de dados (ELMASRI; NAVATHE, 2011).

Ao pensar em linguagens de consulta para SGBD, a SQL é considerada a linguagem padrão. Entre as principais instruções DML em SQL, estão: SELECT, INSERT, UPDATE e DELETE. A instrução SELECT seleciona registros e campos específicos em um banco de dados. Já a instrução INSERT é utilizada para a inclusão de novos valores a atributos do banco de dados. Já por meio da instrução UPDATE os valores de atributos são atualizados, e por meio de DELETE, são apagados do banco de dados (ELMASRI; NAVATHE, 2011).

Para Ramarkrishnan e Gerhrke (2011), a forma básica de uma consulta SQL é a seguinte:

```
SELECT [DISTINCT] <lista-seleção>
FROM <lista-from>
WHERE <qualificação>
```

Assim, a lista-from se trata de uma lista com os nomes de tabela, e a lista-seleção contém os nomes de colunas das tabelas presentes na lista-from. Já a qualificação da cláusula FROM é uma expressão que se utiliza dos conectivos ou operadores lógicos NULL, AND, OR e NOT (“nulo”, “e”, “ou’ e “não”, respectivamente) para realizar uma seleção dos dados presentes na tabela consultada. Nesse caso, NULL é utilizado para representar um valor desconhecido, um valor que não está disponível ou, ainda, um valor não aplicável, enquanto a palavra reservada DISTINCT é opcional, pois indica que a consulta não deve apresentar duplicatas (RAMARKRISHNAN; GEHRKE, 2011).

Em toda consulta, é necessário que se tenha uma cláusula SELECT, com a especificação das colunas que devem ser mantidas no resultado, bem como uma cláusula FROM, para indicar o produto cartesiano das tabelas. Nesse caso, a cláusula WHERE pode ser opcional, pois é utilizada para refinar a consulta.

Exemplo 1

Veja o Quadro 1, que apresenta dados sobre marinheiros, e o Quadro 2, que traz dados sobre reservas de barcos.

Quadro 1. Marinheiros

id-marin	nome-marin	idade
22	Dustin	45
29	Brutus	33
32	Lubber	55
58	Andy	25
64	Brutus	33
71	Bob	35
74	Art	23

Fonte: Adaptado de Ramarkrishnan e Gehrke (2011).

Quadro 2. Reservas

id-marin	id-barco	dia
22	101	10/10/2019
32	102	10/08/2019
32	103	10/07/2019
58	104	11/10/2019
71	102	10/11/2019
71	103	10/12/2019
74	101	10/09/2019

Fonte: Adaptado de Ramarkrishnan e Gehrke (2011).

Tendo como referência os Quadros 1 e 2, é possível realizar a seguinte consulta:

```
SELECT DISTINCT MARINHEIROS.nome-marin, MARINHEIROS.idade
FROM MARINHEIROS
```

O resultado da consulta sobre o nome de todos os marinheiros e as suas idades será um conjunto de linhas com as informações solicitadas. Nesse caso, a palavra reservada `DISTINCT` auxilia para que o resultado obtido não tenha linhas com nomes e idades iguais. Como você pode visualizar nos Quadros 3 e 4, a diferença do uso ou não da palavra reservada `DISTINCT` é que o nome do marinheiro Brutus aparecerá duas vezes, pois há dois marinheiros com o mesmo nome e a mesma idade.

Quadro 3. Consulta com `DISTINCT`

nome-marin	idade
Dustin	45
Brutus	33
Lubber	55
Andy	25
Brutus	33
Bob	35
Art	23

Fonte: Adaptado de Ramarkrishnan e Gehrke (2011).

Quadro 4. Consulta sem `DISTINCT`

nome-marin	idade
Dustin	45
Brutus	33
Lubber	55
Andy	25
Bob	35
Art	23

Fonte: Adaptado de Ramarkrishnan e Gehrke (2011).

Além das instruções DML, existem outras ferramentas importantes que auxiliam na manipulação de bancos de dados em SQL, que são os **operadores relacionais**. De acordo com Manzano (2017), os operadores relacionais são responsáveis por estabelecer ações de comparação entre dados. Eles são empregados para definir possíveis condições entre dois valores, para, assim, gerar diferentes consultas em uma tabela de dados. Da mesma forma, Elmasri e Navathe (2011) afirmam que, quando o usuário realiza uma consulta de um dado específico de seu interesse, uma nova lista ou relação de dados é criada utilizando operadores relacionais. Veja os operadores no Quadro 5.

Quadro 5. Operadores relacionais

Operador	Descrição
>	Maior que
<	Menor que
=	Igual a
< >	Diferente de
>=	Maior ou igual a
<=	Menor ou igual a

Fonte: Adaptado de Manzano (2017).

Geralmente, quando uma consulta é feita a um banco de dados, sabe-se o que se quer e o que se precisa. Por esse motivo, os operadores relacionais auxiliam a selecionar os resultados a serem obtidos com condições previamente determinadas.

Exemplo 2

Este exemplo de consulta (RAMARKRISHNAN; GEHRKE, 2011) utiliza o `SELECT` e o operador relacional =. São pesquisados os nomes dos marinheiros que realizaram reserva do barco 103:

```
SELECT MARINHEIROS.nome-marin
FROM MARINHEIROS, RESERVAS
WHERE MARINHEIROS.id-marin = RESERVAS.id-marin AND RESERVAS.
id-barco = '103'
```

O resultado da consulta será a lista dos marinheiros com identificadores 32 e 71, que são respectivamente Lubber e Bob, conforme os dados dos Quadros 1 e 2. Isso porque estava registrado no banco de dados que os dois haviam reservado o barco com a identificação igual a 103.

2 Consultas e manipulação de dados com SQL

Ao realizar consultas em um banco de dados SQL, você pode encontrar um mesmo nome sendo utilizado por mais de um atributo em tabelas diferentes. Caso seja necessário ter acesso a esses atributos com o mesmo nome na mesma consulta, é possível qualificar o nome do atributo com o nome da relação, para prevenir possíveis erros ou ambiguidade. Para tanto, inclui-se um prefixo com o nome da relação ao nome do atributo, separados por um ponto. Por exemplo, suponha uma tabela `ALUNOS` com o atributo `NOME` e uma tabela `PROFESSORES` também com um atributo `NOME`. Em uma consulta que necessite de informações das duas tabelas, deve-se utilizar `ALUNOS.NOME` e `PROFESSORES.NOME` (ELMASRI; NAVATHE, 2011).

> Podemos usar esse mecanismo de nomeação de apelidos em qualquer consulta SQL para especificar variáveis de tupla para cada tabela na cláusula `WHERE`, não importando se a mesma relação precisa ser referenciada mais de uma vez. De fato, essa prática é recomendada, pois resulta em consultas mais fáceis de compreender (ELMASRI; NAVATHE, 2011, p. 67, grifo nosso).

Ambiguidades também podem ocorrer em casos nos quais as consultas tratam duas vezes da mesma relação. Pense novamente nas tabelas ALUNOS e PROFESSORES, em uma consulta com o primeiro nome e o sobrenome do aluno e o primeiro nome e o sobrenome do professor. Em vez de prefixar o nome das tabelas em todos os atributos da consulta, é possível criar um *alias* ou pseudônimo para cada tabela. Assim, pode-se utilizar A.NOME, A.SOBRENOME, P.NOME e P.SOBRENOME. Ao empregar o *alias*, ele deve ser citado na consulta na cláusula FROM, conforme orientam Elmasri e Navathe (2011).

Verifique os seguintes exemplos de *alias* (ELMASRI; NAVATHE, 2011):

```
SELECT A.NOME, A.SOBRENOME, P.NOME, P.SOBRENOME
FROM ALUNOS AS A, PROFESSORES AS P
```

```
SELECT A.NOME, A.SOBRENOME, P.NOME, P.SOBRENOME
FROM ALUNOS A, PROFESSORES P
```

Os *alias* ou pseudônimos também podem ser empregados nos atributos de uma consulta SQL. Por esse motivo, quando existir uma relação com mais de um *alias* especificado, é possível utilizá-los em diferentes referências da relação. Isso significa, portanto, que podem existir referências múltiplas de uma mesma relação em uma consulta. A atribuição dos pseudônimos pode ser feita em qualquer consulta SQL, até mesmo na especificação de variáveis na cláusula WHERE. O emprego do *alias* facilita até mesmo a compreensão de uma consulta (ELMASRI; NAVATHE, 2011).

Outro fator importante na realização de consultas SQL é o resultado da presença e da ausência da cláusula WHERE. No caso da ausência do WHERE, entende-se que não existe nenhuma condição para a seleção de tuplas (de forma sucinta, linhas ou colunas com dados) da tabela de dados; então, todas as tuplas que tiverem relação com o que foi solicitado na cláusula FROM serão apresentadas no resultado da consulta. Por esse motivo, é extremamente importante incluir todas as condições de seleção na cláusula WHERE, para evitar que se obtenham resultados incorretos ou muito grandes. Já quando o que se quer são resultados grandes nas consultas, com todos os atributos das tuplas selecionadas, pode ser utilizado apenas um asterisco (*) após a cláusula SELECT, não sendo necessário incluir todos os seus nomes (ELMASRI; NAVATHE, 2011).

Como visto anteriormente, a cláusula FROM, bem como a cláusula WHERE, utilizam-se dos operadores lógicos ou booleanos NULL, AND, OR e NOT para realizar uma seleção dos dados. É necessário compreender que esses operadores podem ser empregados de maneira conjunta, a fim de tornar menor e mais preciso o resultado de uma consulta. Por exemplo, suponha que, em um cadastro de funcionários de uma empresa, é necessário consultar os nomes dos colaboradores que trabalham no setor de produção e que recebem mais de R$ 2.000,00, ou os colaboradores que exercem suas atividades no setor de expedição, com salários até R$ 2.000,00, conforme exemplo abaixo.

```
SELECT F.NOME, F.SALARIO, F.SETOR
FROM FUNCIONARIOS AS F
WHERE F.SETOR = PRODUCAO AND SALARIO > 2000 OR F.SETOR = 'EX-
PEDICAO' AND SALARIO <= 2000
```

Assim, o resultado se torna mais preciso utilizando-se dois operadores lógicos na mesma consulta. Nesse sentido, cabe deixar claro que:

> NULL é usado para representar um valor que está faltando, mas que em geral tem uma de três interpretações diferentes – valor desconhecido (existe, mas não é conhecido), valor não disponível (existe, mas é propositadamente retido) ou valor não aplicável (o atributo é indefinido para essa tupla). Considere os seguintes exemplos para ilustrar cada um dos significados de NULL.
> 1. Valor desconhecido. A data de nascimento de uma pessoa não é conhecida, e por isso é representada por NULL no banco de dados.
> 2. Valor indisponível ou retido. Uma pessoa tem um telefone residencial, mas não deseja que ele seja listado, por isso ele é retido e representado como NULL no banco de dados.
> 3. Atributo não aplicável. Um atributo Conjuge seria NULL para uma pessoa que não fosse casada, pois ele não se aplica a essa pessoa (ELMASRI; NAVATHE, 2011, p. 76. grifo nosso).

Além dos operadores lógicos, a SQL dispõe de operadores de comparação, entre eles: LIKE, BETWEEN e IN. O LIKE permite comparar apenas parte dos caracteres do banco de dados, por meio de caracteres reservados. Nesse caso, o símbolo % substitui um número qualquer de caracteres, enquanto o símbolo de sublinhado (_) substitui apenas um caractere na consulta. O operador BETWEEN é utilizado para pesquisar uma faixa de valores entre um e outro valor; já o operador de comparação IN verifica que determinado valor *v* está entre um conjunto de *n* valores (ELMASRI; NAVATHE, 2011).

Exemplo 3

Verifique os seguintes exemplos de operadores de comparação, adaptados de Elmasri e Navathe (2011). Para consultar todos os nomes dos funcionários com endereço em São Paulo/SP, aplica-se:

```
SELECT F.NOME
FROM FUNCIONARIOS AS F
WHERE ENDERECO LIKE '%SAOPAULO%'
```

Já para consultar todos os dados dos funcionários com salários entre R$ 2.000,00 e R$ 10.000,00, utiliza-se:

```
SELECT *
FROM FUNCIONARIOS
WHERE (SALARIO BETWEEN 2000 AND 10000)
```

3 Operadores de conjuntos e junções de tabelas

Nesta seção, você vai estudar a aplicação dos operadores de conjuntos e das junções.

Operadores de conjuntos

Estão disponíveis em SQL três construtores de manipulação de conjunto, que complementam o formato de consulta básico. Tendo em vista que o resultado de uma consulta em banco de dados é um conjunto de linhas, é lógico pensar no uso de operações como união, interseção e diferença, por exemplo. Nesse sentido, a SQL suporta operações com os seguintes **operadores em conjuntos**: `UNION` (união), `INTERSECT` (interseção) e `EXCEPT` (exceção ou diferença). Já os operadores `IN`, `op ANY` e `op ALL` são utilizados para comparar um valor com os elementos de um conjunto, e o comparador `EXISTS` é utilizado para verificar se o conjunto é vazio (RAMARKRISHNAN; GEHRKE, 2011).

Exemplo 4

Verifique os seguintes exemplos de operadores de conjuntos, adaptados de Ramarkrishnan e Gerhrke (2011). Para uma consulta de nomes de funcionários que reservaram um caminhão vermelho ou um caminhão verde, aplica-se, utilizando `UNION`:

```
SELECT F.NOME
FROM FUNCIONARIOS AS F, RESERVAS AS R, CAMINHOES AS C
WHERE F.ID-FUNCIONARIO = R.ID-FUNCIONARIO AND R.ID-CAMINHAO =
C.ID-CAMINHAO AND C.COR = 'VERMELHO'
UNION
SELECT F.NOME
FROM FUNCIONARIOS AS F, RESERVAS AS R, CAMINHOES AS C
WHERE F.ID-FUNCIONARIO = R.ID-FUNCIONARIO AND R.ID-CAMINHAO =
C.ID-CAMINHAO AND C.COR = 'VERDE'
```

Já para a consulta de nomes de funcionários que reservaram um caminhão vermelho e um caminhão verde, aplica-se o seguinte código, dessa vez utilizando `INTERSECT`:

```
SELECT F.NOME
FROM FUNCIONARIOS AS F, RESERVAS AS R, CAMINHOES AS C
WHERE F.ID-FUNCIONARIO = R.ID-FUNCIONARIO AND R.ID-CAMINHAO =
C.ID-CAMINHAO AND C.COR = 'VERMELHO'
INTERSECT
SELECT F.NOME
FROM FUNCIONARIOS AS F, RESERVAS AS R, CAMINHOES AS C
WHERE F.ID-FUNCIONARIO = R.ID-FUNCIONARIO AND R.ID-CAMINHAO =
C.ID-CAMINHAO AND C.COR = 'VERDE'
```

Por fim, para a consulta de nomes de funcionários que reservaram um caminhão vermelho, mas não um caminhão verde, utiliza-se `EXCEPT`:

```
SELECT F.NOME
FROM FUNCIONARIOS AS F, RESERVAS AS R, CAMINHOES AS C
WHERE F.ID-FUNCIONARIO = R.ID-FUNCIONARIO AND R.ID-CAMINHAO =
C.ID-CAMINHAO AND C.COR = 'VERMELHO'
EXCEPT
```

```
SELECT F.NOME
FROM FUNCIONARIOS AS F, RESERVAS AS R, CAMINHOES AS C
WHERE F.ID-FUNCIONARIO = R.ID-FUNCIONARIO AND R.ID-CAMINHAO =
C.ID-CAMINHAO AND C.COR = 'VERDE'
```

Cabe ressaltar que, no caso de consultas com os operadores de conjuntos `UNION`, `INTERSECT` e `EXCEPT`, os resultados duplicados são eliminados mesmo sem o uso do `DISTINCT`. Para que resultados duplicados permaneçam nas consultas, é necessário acrescentar a palavra `ALL` após esses operadores. Assim, `UNION ALL` possibilita que o número de linhas iguais em cada parte da união seja somado. Da mesma forma, com os operadores `INTERSECT ALL` e `EXCEPT ALL`, as linhas duplicadas são mantidas (RAMARKRISHNAN; GEHRKE, 2011).

Junções

Outra operação presente em SQL que auxilia a combinar informações de duas ou mais relações é a **junção**. De acordo com Elmasri e Navathe (2011), a tabela ou relação de junção foi incorporada na SQL para possibilitar que os usuários juntem em uma tabela os resultados de uma consulta, utilizando a operação junção na cláusula `FROM`. A construção dessa tabela de junção pode ser mais fácil do que misturar diferentes condições de seleção e junção em outra cláusula.

Exemplo 5

A seguir, verifique um exemplo de operadores de junção, adaptado de Elmasri e Navathe (2011). Para uma consulta de nomes e endereços de funcionários que trabalham no departamento de produção, utiliza-se:

```
SELECT NOME, ENDERECO
FROM (FUNCIONARIOS JOIN DEPARTAMENTOS ON DNR = DNUMERO)
WHERE DNOME = 'PRODUCAO';
```

Nesse exemplo, a cláusula `FROM` contém uma única tabela de junção.

Com a tabela de junção, também é possível realizar junções de outras formas, como a `NATURAL JOIN` (junção natural) e a `OUTER JOIN` (junção externa). Por meio da `NATURAL JOIN` sobre duas relações distintas, cada par de atributos é incluído apenas uma vez na relação resultante. Caso os nomes

dos atributos de junção não sejam os mesmos nas relações iniciais, é possível alterar seus nomes para que combinem e aplicar a NATURAL JOIN somente depois. Essa alteração pode ser feita no texto da própria consulta, utilizando-se a construção AS na cláusula FROM (ELMASRI; NAVATHE, 2011).

> As operações JUNÇÃO [...] combinam com tuplas que satisfazem a condição de junção. Por exemplo, para uma operação JUNÇÃO NATURAL *R* * *S*, somente tuplas de *R* possuem tuplas combinando em *S* – e vice-versa – aparecem no resultado. Logo, as tuplas sem uma tupla correspondente (ou relacionada) são eliminadas do resultado de JUNÇÃO. As tuplas com valores NULL nos atributos também são eliminadas, Esse tipo de junção, em que as tuplas sem correspondência são eliminadas, é conhecido como junção interna (*inner join*) [...]. Um conjunto de operações, chamadas de junções externas (*outer joins*), foi desenvolvido para o caso em que o usuário deseja manter todas as tuplas em *R*, ou todas em *S*, ou todas aquelas nas duas relações no resultado da JUNÇÃO, independentemente de elas possuírem ou não tuplas correspondentes na outra relação. (ELMASRI; NAVATHE, 2011, p. 113, grifo nosso).

O tipo padrão de junção se chama INNER JOIN (o mesmo que JOIN), pois uma tupla somente é incluída no resultado se combinar com a outra relação. Se o resultado for nulo para uma das relações, a tupla é excluída da tabela de junção. Porém, quando o usuário necessitar que todas as tuplas sejam incluídas, uma OUTER JOIN precisa ser utilizada. Existem diferentes opções disponíveis para distinguir tabelas de junção; entre elas, estão: LEFT OUTER JOIN, na qual toda tupla da tabela da esquerda precisa estar presente no resultado, caso contrário, é preenchida com valores nulos; RIGHT OUTER JOIN, em que, ao contrário da anterior, toda tupla da tabela da direita precisa estar presente no resultado, caso contrário, é preenchida com valores nulos; e FULL OUTER JOIN (ELMASRI; NAVATHE, 2011).

Link

Você viu ao longo deste capítulo diferentes conceitos e exemplos em SQL de consultas e manipulação de dados, além do trabalho com conjuntos. Acessando o *link* a seguir, você pode verificar conceitos adicionais sobre junções, que podem ser empregados em diferentes versões de bancos de dados em SQL.

https://qrgo.page.link/Zhtf1

Referências

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. 6. ed. São Paulo: Pearson, 2011.

MANZANO, J. A. N. G. *Microsoft SQL Server 2016*: express edition interativo. São Paulo: Érica, 2017.

RAMARKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de banco de dados*. 3. ed. Porto Alegre: AMGH, 2011.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

Atualmente os bancos de dados representam um componente essencial para inúmeros sistemas de informação. Em um banco de dados relacional é possível realizar junções de tabelas com o intuito de combinar as informações destes sistemas presentes em duas ou mais relações.

No vídeo a seguir, você verá as diferenças entre as cláusulas *INNER JOIN*, *RIGHT JOIN* e *LEFT JOIN*, que permitem operações de junções em banco de dados relacional a partir de alguns exemplos práticos de consultas em SQL.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

A manipulação de um banco de dados é importante para a manutenção e atualização das informações armazenadas. Por isso, é necessário conhecer e praticar as funções básicas da manipulação de dados, que compreendem inserir, atualizar e excluir registros.

Nesta Dica do Professor, você verificará alguns exemplos de aplicação dos recursos de DML em SQL.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **As linguagens de manipulação de dados (DML) possibilitam criação, consulta e manipulação de dados, dentre outras funções. Entre as principais instruções DML na SQL, está:**

A) *SELECT*, que trata de seleções referentes ao banco de dados.

B) *LIKE*, que trata de pesquisar uma faixa de valores no banco de dados.

C) *BETWEEN*, que trata da comparação entre dados com caracteres reservados.

D) *INTERSECT*, que trata de comparar uma faixa de dados até outra.

E) *EXCEPT*, que trata da inserção de dados em um banco de dados.

2) **Os operadores relacionais, utilizados nas DML, são responsáveis por estabelecer ações de comparação entre dados, definindo possíveis condições entre dois valores, para assim gerar diferentes consultas. Nesse contexto, assinale a opção onde é apresentado o operador relacional que é utilizado para comparar um dado diferente do outro:**

A) !=.

B) <>.

C) <=.

D) ≠.

E) <

3) **Ao realizar consultas em um banco de dados SQL, você pode encontrar um mesmo nome sendo utilizado por mais de um atributo em tabelas diferentes. Caso seja necessário ter acesso a esses atributos com o mesmo nome na mesma consulta, é possível qualificar o nome do atributo com o nome da relação com um *alias*, para prevenir possíveis erros ou ambiguidade. Entre as opções a seguir, assinale a que utiliza corretamente o recurso do *alias* em uma consulta SQL:**

A) SELECT P.NOME, P.IDADE, P.ALTURA, M.NOME

FROM PACIENTE ALIAS P, MEDICO ALIAS M

B) SELECT P.NOME, P.IDADE, P.ALTURA, M.NOME

FROM PACIENTE A, MEDICO B

C) SELECT P.IDADE, M.NOME

FROM PACIENTE IDADE, MEDICO NOME

D) SELECT P.NOME, P.IDADE, P.ALTURA, M.NOME

FROM PACIENTE AS P

E) SELECT P.NOME, P.IDADE, P.ALTURA, M.NOME

FROM PACIENTE AS P, MEDICO AS M

4) A SQL suporta diferentes operadores em conjuntos, já que o resultado de uma consulta em banco de dados é um conjunto de linhas, é lógico pensar no uso de operações como união, interseção e diferença, por exemplo. Nesse sentido, verifique as tabelas a seguir:

FUNCIONÁRIOS				
#CÓDIGO	NOME	SEXO	NASCIMENTO	SALÁRIO
1	Ana	Feminino	13/07/1972	R$8.000,00
2	Bia	Feminino	22/08/1989	R$8.000,00
3	Cláudio	Masculino	11/09/1987	R$2.500,00
4	Diego	Masculino	26/10/1988	R$4.000,00
5	Fabiana	Feminino	30/11/1993	R$6.500,00

TERCEIRIZADOS				
#CÓDIGO	NOME	SEXO	NASCIMENTO	SALÁRIO
1	Glória	Feminino	10/08/1973	R$2.000,00
2	Hilda	Feminino	23/09/1985	R$1.500,00
3	Ivo	Masculino	12/10/1986	R$1.500,00
4	João	Masculino	27/11/1982	R$1.700,00
5	Kelly	Feminino	15/12/1983	R$2.500,00

Com base nas tabelas apresentadas, assinale a alternativa na qual é descrita corretamente uma consulta com o operador de conjuntos que realiza uma união das tabelas de dados:

A) SELECT F.NOME, F.SALARIO

FROM FUNCIONARIOS F

INTERSECT

SELECT T.NOME, T.SALARIO

FROM TERCEIRIZADOS T

WHERE F.SALARIO = T.SALARIO

B) SELECT F.NOME, F.SALARIO

FROM FUNCIONARIOS F

EXCEPT

SELECT T.NOME, T.SALARIO

FROM TERCEIRIZADOS T

WHERE F.SALARIO = T.SALARIO

C) SELECT F.NOME, F.SALARIO

FROM FUNCIONARIOS F

JOIN

SELECT T.NOME, T.SALARIO

FROM TERCEIRIZADOS T

WHERE F.SALARIO = T.SALARIO

D) SELECT F.NOME, F.SALARIO

FROM FUNCIONARIOS F

UNION

SELECT T.NOME, T.SALARIO

FROM TERCEIRIZADOS T

WHERE F.SALARIO = T.SALARIO

E) SELECT F.NOME, F.SALARIO

FROM FUNCIONARIOS F

AND

SELECT T.NOME, T.SALARIO

FROM TERCEIRIZADOS T

WHERE F.SALARIO = T.SALARIO

5) As relações de junção na SQL possibilitam que os usuários juntem em uma tabela os resultados de uma consulta utilizando a operação junção na cláusula *FROM*. Um dos operadores de junção, quando aplicado sobre duas relações distintas, resulta na inclusão de somente uma tupla no resultado, portanto se o resultado da consulta for nulo para uma das relações a tupla é excluída da tabela de junção. Assinale o operador de junção que conta com essa característica:

A) *INNER JOIN*.

B) *OUTER JOIN*.

C) *LEFT OUTER JOIN*.

D) *RIGHT OUTER JOIN*.

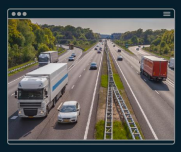
E) *FULL OUTER JOIN*.

Na prática

A tecnologia dos bancos de dados influencia diretamente o uso de computadores e de sistemas de informações cada vez mais robustos. Cabe ressaltar a importância dos bancos de dados para diferentes organizações e diferentes áreas, como engenharia, direito, educação ou segurança, por exemplo. Um banco de dados relaciona fatos que podem ser guardados, os quais possuem significados implícitos.

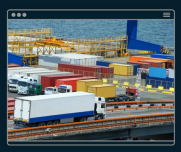
Neste Na Prática, você vai conhecer o caso de uma empresa que faz segurança de caminhões por meio de rastreamento via satélite, acompanhando os processos de criação de um novo sistema de rastreamento.

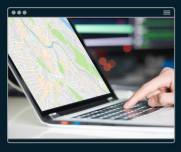
CRIAÇÃO DE SISTEMA DE RASTREAMENTO UTILIZANDO BANCO DE DADOS



A empresa OpenFix trabalha com rastreamento de caminhões que transportam cargas de alto valor financeiro.

O trabalho consiste em acompanhar a localização dos veículos e controlar se estão executando a rota previamente planejada, a quantidade de vezes que param no meio do caminho ou ainda realizar estimativas de chegada nos destinos das cargas.

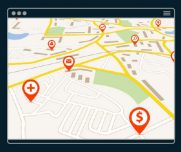




Esse trabalho é possível pela comunicação entre antenas instaladas nos caminhões e os sinais enviados (comunicação por satélite ou por GPS) à central de monitoramento. As informações recebidas são armazenadas em SGBD e sistemas específicos para isso.

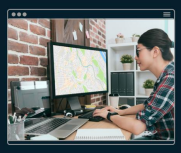
A empresa OpenFix percebeu que o mercado de sistemas para rastreamento e logística de transportes estava em crescimento, por isso decidiu criar o seu próprio sistema, para comercialização.

Como uma das funções esperadas em um sistema de rastreamento é o controle de rotas (rodovias utilizadas entre a origem e o destino), para criar o sistema foi necessário, inicialmente, criar um banco de dados com as rodovias mais utilizadas no país.

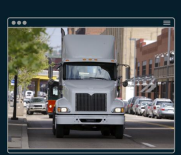


Para isso, o banco de dados foi alimentado com localizações (latitudes e longitudes) de veículos ao longo de seis meses. Para que, depois disso, os dados pudessem ser importados para o novo sistema.

A equipe de análise e desenvolvimento de sistemas contratada pela OpenFix necessitou refinar os dados de localização armazenados, utilizando SQL.



Assim, com as latitudes e longitudes captadas pelos veículos em cada localidade ou rodovia, foi possível organizá-las em mapas para posterior correção de pontos falhos onde havia menos registros. Para isso, foi necessária a inserção e atualização do banco de dados por meio do uso de operadores lógicos, relacionais e de comparação.



Ao final da etapa de construção de rotas, foi possível criar mapas próprios a serem utilizados no sistema da OpenFix. Além disso, se incluíram outras informações úteis, como locais de parada seguros, rodovias com maior índice de roubo de cargas, postos de combustíveis credenciados, entre outras.

Essa inclusão pode ser operacionalizada utilizando operadores de junção de tabelas, por meio da criação de junções entre as tabelas com as latitudes e longitudes dos caminhões com as demais tabelas, como:

- Pontos de parada seguros
- Postos fiscais
- Postos de polícia rodoviária

Uma das consultas utilizadas foi essa, para juntar todos os pontos de parada com as localizações de rotas:

```
SELECT P.Latitude, P.Longitude, P.Nome_ponto,   L.Latitude,
L.Longitude, L.Nome_ponto,
FROM Pontos_parada as P
FULL OUTER JOIN Localizacao as L
on P.Nome_ponto = L.Nome_ponto
```

Assim, é possível optar pela OUTER JOIN, já que todos os dados presentes nas tabelas de pontos seguros de parada e localização da rota necessitavam estar no resultado das consultas para a construção do banco de dados a ser utilizado no sistema.

Posterior à conclusão de todas as etapas de programação e testes, a empresa iniciou a comercialização de diferentes sistemas de rastreamento e logística de caminhões, tendo como base os mapas construídos e manipulados com SQL, o que permite sua constante atualização.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Exemplos e conceitos SQL

Confira no *link* a seguir a documentação oficial do MySQL sobre *SELECT*, *JOIN* e *UNION*, que traz exemplos do sua aplicação em banco de dados.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conjuntos e junções em SQL

Para aprofundar seus conhecimentos sobre a criação de conjuntos e de junções de tabelas em SQL para manipulação de dados, assista ao vídeo a seguir, que aborda operadores de conjuntos e exemplos de uso de *JOIN*.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Consultas em SQL

Para aprofundar seus conhecimentos sobre as consultas utilizadas em SQL utilizando a cláusula *WHERE*, assista ao vídeo a seguir, com exemplos de consultas de dados em tabelas.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.