LinkedIn Learning

Anotações:
SQL Essential Training

PARTE ÚNICA

—

Competência abordada: SQL
Duração: 3 horas
Ministrante: Bill Weinman

Pontos importantes

Para finalizar uma operação ponto e vírgula pode não ser necessário, mas é uma boa prática.

"Statement" é a instrução que executa chamadas.

"Clause" especifica detalhes da chamada. A clause FROM especifica de qual tabela será chamada; A clause WHERE especifica qual row será selecionada.

Uma "Logical Expression" Valida um valor em uma expressão. A logical expression " = " faz uma comparação de igualdade.

Na chamada:
SELECT * FROM Countries WHERE Continent = 'Europe';
A chamada inteira é a operação
SELECT é o statement
FROM é a clause
WHERE é outra clause
= é a logical expression
Countries é a tabela
Continent é a row
Europe é o valor da row Continent
O retorno dessa operação seria todas as rows que que tem o valor Continent = 'Europe'

CRUD
As quatro funções fundamentais de um sistema de database são chamadas de CRUD: Create, Read, Update, Delete

Tabelas tem Row, que são as linhas, também chamadas de **record**, e colunas, que também são chamadas de **field**.

Quando uma coluna é usada como identificador único ela é chamada de Primary Key. Geralmente a coluna primary key é chamada de Id.

As identificações são usadas para criar relacionamentos entre tabelas. Elas usam "foreign keys" ou chaves estrangeiras para se referir às chaves primárias das outras tabelas. Na prática a tabela de relacionamentos tem algo como coluna 'id' que se refere a chave primária da própria tabela, e outras duas colunas de ID: 'id_tabela1', 'id_tabela2' onde estão as chaves estrangeiras que se referem às chaves primárias das suas respectivas tabelas.

' ' ou single quotes são usadas para strings

" " ou double quotes são usados para se referir a um identificador que tem espaço, como SELECT Name AS "Nome do usuário"

AS é uma clause usada para mostrar um resultado com outro nome no **Grid View**. AS pode ser oculto declarando o resultado e o nome a ser mostrado em seguida como SELECT Name "Nome do usuário"

* significa retornar todas as colunas ou todos os dados.

ORDER BY é uma clause para ordenação, pode ser seguida de uma coluna para ordenar os resultados por ordem dos dados da coluna.

ORDER BY pode ser usado com combinações diferentes de valores.

```
SELECT coluna, coluna2 FROM Tabela
        ORDER BY coluna DESC, coluna2;
```

Quer dizer que vai ordenar primeiro por coluna1 do maior para menor e depois ordenar por coluna2 crescente.

, ou **comma** pode ser usada delimitar resultados como
SELECT Name, Age

OFFSET 5 é usada após o ORDER BY para ignorar os 5 primeiros resultados

SELECT é um statement que busca dados a partir de colunas em tabelas

COUNT é uma clause que conta a quantidade de rows em uma tabela.

```
SELECT COUNT (*) FROM Country
        WHERE Population > 200000000 AND Continent = 'America';
```

Outro exemplo é SELECT COUNT (Name) para contar quantas rows tem efetivamente valor na coluna Name

INSERT INTO é um statement para inserir rows no banco.
INSERT INTO tabela (coluna, coluna) VALUES ('valor', 'valor')

UPDATE é um statement para atualizar dados no banco.
UPDATE tabela SET coluna = 'valor', coluna = 'valor'
    WHERE chavePrimaria = valordaChave;

NULL pode ser usado como valor para significar a ausência de valor.

DELETE é um statement para deletar rows, deve ser usado a clause WHERE para comparar o valor da chave da row a ser deletada.
DELETE FROM tabela WHERE chave = valor;

Database Schema ou Table Schema é como as vezes é chamado a configuração das colunas no banco

CREATE TABLE é um statement para criar tabela.
CREATE TABLE tabela (variavel INTEGER, variavel TEXT);
Os dados não tem valor nesse momento

IF EXISTS é uma logical expression.
DROP TABLE IF EXISTS tabela;

DEFAULT é uma clause que pode ser pode ser usada para estabelecer valores NULL.
INSERT INTO test DEFAULT VALUES;

Dados de um SELECT podem ser usados para inserir valores em uma tabela.
INSERT INTO tabela2 (coluna3, coluna4)
    SELECT coluna1, coluna2 FROM tabela1;
Nesse caso todos as rows da tabela1 seriam inseridos pois não foi declarado WHERE no select. Nesse comando poderiam ser adicionados novas colunas e então o nome da coluna adicionada seria NULL

NULL é um state de ausência de um valor. WHERE coluna = NULL não funciona pois NULL não é um valor, é só a ausência. WHERE valor IS NULL funciona. IS NOT NULL funciona também

Constraint é uma regra para valores. Mais comuns:
NOT NULL (não poder inserir dado null na coluna)
UNIQUE (não pode registrar o mesmo valor em duas rows). Talvez possa usar NULL pois NULL não é um valor.
PRIMARY KEY (usado para gerar id sequencial automático. Não precisa declarar o id quando estiver populando o banco)
FOREIGN KEY
CHECK (ver se o valor em uma coluna satisfaz alguma condição)
DEFAULT ou DEFAULT 'valor' para a coluna assumir o valor indicado quando for registrado como NULL)

Sintaxe Constraint:
```
CREATE TABLE tabela (
        coluna INTEGER UNIQUE NOT NULL, coluna2 UNIQUE);
```
usei duas constraints para a coluna 1 sem usar vírgula

ALTER é um statement usado para alterar schemas, por exemplo adicionar uma coluna.
```
ALTER TABLE tabela ADD coluna TEXT
```
Aqui pode entrar a constraint DEFAULT 'valor' para setar todos os valores da nova coluna;

REGEXP selecionar dados que tem série de características

LIKE é logical operation para comparar valores.
```
WHERE coluna LIKE '%valor%';
```
Acima usado para encontrar resultados em que o valor da coluna LIKE tenha valor escrito em algum lugar no valor.

___ ou underline é um caractere curinga, o % significa qualquer quantidade de algarismos com qualquer valor..
Names LIKE '_at%' encontraria o nome 'mateus'

IN é um operador para selecionar resultados que estejam em uma lista.
WHERE coluna IN ('valor1', 'valor2');
Usado acima para retornar as rows que o valor da coluna coluna seja valor1 ou valor2

DISTINCT é usado com SELECT para retornar apenas os resultados diferentes.
SELECT DISTINCT coluna FROM tabela.
Também pode ser usado com SELECT DISTINCT coluna1, coluna2 e vai retornar as rows em que a combinação de coluna1, coluna2 é única

Em SQL 0 é falso e qualquer coisa que não seja 0 é verdadeiro

CASE é usado para iniciar uma expressão condicional e termina com **END**. Nesse exemplo vai olhar se o valor da coluna 'a' é verdadeiro ou falso mas poderia ser WHEN (a=1) ou 'mateus'.

```
SELECT
        CASE
                WHEN a THEN 'true' ELSE 'false'
        END
AS boolA FROM booltest;
```

Uma Joined query relaciona tabelas.
INNER JOIN é a forma mais simples e comum de join, ela inclui rows de ambas tabelas que estejam sob alguma condição.
INNER JOIN é referente à intersecção se pensar em um diagrama de venn.

SELECT coluna1, coluna2
FROM tabela (que diz respeito a coluna1)
JOIN tabela2 ON coluna1.id = coluna2.id; (depois do ON é a regra da intersecção)

OUTTER JOIN é menos comum, um LEFT OUTTER JOIN inclui todas as rows da tabela que não seguem a condição + as rows da tabela da direita que seguem a condição. Um RIGHT OUTTER JOIN inclui todas as rows da direita que não cumprem a condição + rows da esquerda que cumprem. FULL OUTTER JOIN combina ambos.

Junction tables são usadas nos relacionamentos many to many

SUBSTR (string, start, length) captura uma substring utilizando os argumentos string, inicio da sub e tamanho da sub. O primeiro caracter é contado como 1. M seria o start 1 de Mateus

TRIM (string) ou SELECT TRIM (nome, ',.:;') é case sensitive, remove os espaços antes e depois da string ou outros caracteres especificados que estejam antes ou depois

Precision = número de dígitos na esquerda
Scale = número de dígitos na direita

Para dinheiro é melhor usar um tipo de data que tenha mais scale para melhorar a acurácia

HAVING é uma clause que parece WHERE mas serve pra agregações, como GROUP BY
WHERE vem antes do GROUP BY e HAVING vem depois do GROUP BY

AVG é uma clause para calcular média

Transaction é um grupo de operações. Começa com BEGIN TRANSACTION, vem as operações, e termina com END TRANSACTION ou **COMMIT**
Se uma operação da transação falhar a transação não vai funcionar e o DB vai voltar para um estado válido que tinha antes da transição
A transação pode voltar para o estado anterior usando ROLLBACK dentro da transaction
Transações podem melhorar o desempenho.

ROLLBACK é usado para invalidar a TRANSACTION

Trigger é uma operação que é executada quando algo específico ocorre no banco de dados
O trigger também pode conter BEFORE UPDATE ON tabela para prevenir algo

```
CREATE TRIGGER newTriggerVenda AFTER INSERT ON venda
        BEGIN
                pode usar NEW para se referir a nova row do evento como
                UPDATE cliente SET ultimo_pedido_id = NEW.id
                WHERE cliente.id = NEW.cliente_id;
        END
;

CREATE TRIGGER updateTriggerTabela BEFORE UPDATE ON tabela
        BEGIN
                SELECT RAISE(ROLLBACK, 'erro: caiu no trigger')
                FROM tabela WHERE id = NEW.id AND
        colunaCancelarUpdate =1;
        END
;
```

Timestamp é criado AFTER INSERT

Subselect é um select dentro de select entre parêntesis para manter dados organizados ou filtrados como dar um SELECT aliasTabela.Coluna FROM (SELECT ... AS aliasTabela...));

SELECT * FROM album WHERE id IN (SELECT DISTINCT album_id FROM track WHERE duration <= 90);
Acima usado para encontrar o ID dos álbuns que tem duração menor que 90 e utilizar no SELECT de fora

VIEW é uma query salva que pode ser usada como uma tabela
```
CREATE VIEW novaView AS
    SELECT coluna1,coluna2 FROM tabela;
```