SQL

Banco de Dados

Prof. Igor Avila Pereira
igor.pereira@riogrande.ifrs.edu.br

Divisão de Computação
Instituto Federal do Rio Grande do Sul - IFRS - Câmpus Rio Grande

Definição

Definição

SQL (*Structured Query Language*) é um conjunto de comandos de manipulação para bancos de dados.

- É utilizado para criar a estrutura do banco de dados
- Incluir
- Modificar
- Pesquisar informações

Divisão

- A linguagem SQL é dividida nos seguintes componentes:
  - **Data Definition Language (DDL):** permite a criação dos componentes do banco de dados, como tabelas, índices e etc.
  - **Data Manipulation Language (DML):** permite a manipulação dos dados armazenados no banco de dados.
  - **Data Query Language (DQL):** permite extrair dados do banco de dados.
  - **Data Control Language (DCL):** provê a segurança interna do banco de dados.

Divisão

DDL	CREATE TABLE; ALTER TABLE; DROP TABLE....
DML	INSERT; DELETE; UPDATE;
DQL	SELECT
DCL	CREATE USER; ALTER USER...

Tabela : Principais comandos de cada componente

Banco de Dados

Criação:

```
CREATE DATABASE banco;
```

Exclusão:

```
DROP DATABASE banco;
```

Tabelas - Criação e Exclusão

Criação:

```
CREATE TABLE products (
        product_no integer,
        name text DEFAULT 'Igor',
        price numeric NOT NULL CHECK (price > 0)
);
```

```
CREATE TABLE products (
        product_no integer,
        name text,
        price numeric,
        CHECK (price > 0),
        discounted_price numeric,
        CHECK (discounted_price > 0),
        CHECK (price > discounted_price)
);
```

Tabelas - Criação e Exclusão

Exclusão:

```
DROP TABLE products;
```

Dica:

Se você remover uma tabela inexistente, PostgreSQL emite um erro.

Para evitar essa situação, você pode usar o IF EXISTS.

Exclusão (testando se existe a tabela - PostgreSQL):

```
DROP TABLE IF EXISTS products;
```

Tabelas - Criação e Exclusão

A opção CASCADE permite que você remova tabelas dependentes automaticamente.

Exclusão (em cascata):

```
DROP TABLE IF EXISTS products CASCADE;
```

A opção RESTRICT é padrão (atribuída por omissão).

Se houver qualquer tabela ou objeto dependente a tabela, a tabela não é excluída.

Dicas

Você pode colocar uma lista de tabelas (separadas por vírgula) após o comando DROP TABLE.

Assim, você poderá remover várias tabelas ao mesmo tempo.

Colunas

Tabela : Colunas - Tipos

Nome	Descrição
boolean	logical Boolean (true/false)
character [(n)]	fixed-length character string
character varying [(n)]	variable-length character string
date	calendar date (year, month, day)
double precision	double precision floating-point number (8 bytes)
integer	signed four-byte integer
interval [fields] [(p)]	time span
money	currency amount
numeric [(p, s)]	exact numeric of selectable precision

Colunas

Tabela : Colunas - Tipos

Nome	Descrição
real	single precision float-number (4 bytes)
serial	autoincrementing four-byte integer
text	variable-length character string
time	time of day
time with time zone	time of day, including time zone
timestamp	date and time (no time zone)
timestamp with time zone	date and time, including time zone

Modificando Tabelas

Adicionando uma nova coluna:

```
ALTER TABLE products ADD COLUMN description text;
```

Adicionando uma nova coluna (com validação):

```
ALTER TABLE products ADD COLUMN description text
       CHECK (description <> '');
```

Modificando Tabelas

Removendo uma coluna:

```
ALTER TABLE products DROP COLUMN description;
```

Adicionando uma restrição:

```
ALTER TABLE products ADD CHECK (name <> '');
ALTER TABLE products ALTER COLUMN
       product_no SET NOT NULL;
ALTER TABLE products ADD FOREIGN KEY
       (product_group_id) REFERENCES product_groups;
```

Removendo uma restrição NOT NULL:

```
ALTER TABLE products
       ALTER COLUMN product_no DROP NOT NULL;
```

Modificando Tabelas

Alterando o valor padrão de uma coluna:

```
ALTER TABLE products
        ALTER COLUMN price SET DEFAULT 7.77;
```

Removendo o valor padrão de uma coluna

```
ALTER TABLE products
        ALTER COLUMN price DROP DEFAULT;
```

Modificando Tabelas

Mudando o nome de uma coluna:

```
ALTER TABLE products
        RENAME COLUMN product_no TO product_number;
```

Mudando o nome de uma tabela:

```
ALTER TABLE products RENAME TO items;
```

Modificando Tabelas

Restrição - UNIQUE

Assegura que os dados contidos em uma coluna (ou um grupo de colunas) é única no que diz respeito a todas as linhas da tabela.

Exemplo 1:

```
CREATE TABLE products (
        product_no integer UNIQUE,
        name text,
        price numeric
);
```

Modificando Tabelas

Continuação - Restrição UNIQUE....

Exemplo 2:

```
CREATE TABLE example (
        a integer,
        b integer,
        c integer,
        UNIQUE (a, c)
);
```

Atenção

- Dois valores nulos não são considerados iguais nesta comparação.
- Isso significa que , mesmo na presença de uma única restrição é possível armazenar linhas duplicadas que contêm um valor nulo.

Chave Primária

Conceito

Tecnicamente, é simplesmente a combinação da restrição de unicidade (UNIQUE) com a restrição de não-nulo.

```
CREATE TABLE products (
        product_no serial primary key,
        name text DEFAULT 'Igor',
        price numeric NOT NULL CHECK (price > 0)
);
```

Chave Estrangeira

Exemplo 1:

```
CREATE TABLE orders (
        order_id serial PRIMARY KEY,
        product_no integer REFERENCES products,
        quantity integer
);
```

Exemplo 2:

```
CREATE TABLE orders (
        order_id integer PRIMARY KEY,
        product_no integer
                REFERENCES products (product_no),
        quantity integer
);
```

Chave Estrangeira - Relacionamentos N:M

```
CREATE TABLE products (
        product_no serial PRIMARY KEY,
        name text,
        price numeric
);
CREATE TABLE orders (
        order_id integer PRIMARY KEY,
        shipping_address text,
);
CREATE TABLE order_items (
        product_no integer REFERENCES products,
        order_id integer REFERENCES orders,
        quantity integer,
        PRIMARY KEY (product_no, order_id)
);
```

Chave Estrangeira - Relacionamentos N:M

```
CREATE TABLE products (
        product_no serial PRIMARY KEY,
        name text,
        price numeric
); CREATE TABLE orders (
        order_id serial PRIMARY KEY,
        shipping_address text
);
CREATE TABLE order_items (
        product_no integer
                REFERENCES products ON DELETE RESTRICT,
        order_id integer
                REFERENCES orders ON DELETE CASCADE,
        quantity integer,
        PRIMARY KEY (product_no, order_id)
);
```

Chaves Estrangeiras

- **NO ACTION** significa que, se ainda existem quaisquer linhas que fazem referência quando a restrição está marcada, será gerado um erro;
  - Este é o comportamento padrão se você não especificar nada.
- **CASCADE** especifica que quando uma linha referenciada é excluída, linha(s) que fazem referência deve(m) ser automaticamente excluída(s) também.

Há também a opção **RESTRICT**.

- A diferença essencial entre estas duas opções é **NO ACTION** permite que a verificação seja adiada até mais tarde na transação, enquanto **RESTRICT** não.

INSERT

Comando utilizado para inserção de dados em uma tabela.

Exemplo:

Tabela:

```
CREATE TABLE products (
        product_no integer,
        name text,
        price numeric
);
```

Comando INSERT:

```
INSERT INTO products (product_no, name, price)
        VALUES (1, 'Cheese', 9.99);
```

INSERT

Exemplo:

Comando INSERT com SELECT:

```
INSERT INTO fita (filme_id) SELECT id FROM filme;
```

UPDATE

Comando utilizado para alterar qualquer valor de uma tupla

```
UPDATE products SET price = 10 WHERE product_no = 1;
```

DELETE

Você usa o comando DELETE para remover linhas de uma tabela.

Todas:

```
DELETE FROM products;
```

Somente uma tupla específica:

```
DELETE FROM products WHERE product_no = 10;
```

SELECT

O processo de recuperação ou o comando para recuperar dados de um banco de dados é chamada de consulta.

Em SQL o comando SELECT é usada para especificar consultas.

A sintaxe geral do comando SELECT é:

```
SELECT * FROM table1;
```

A sintaxe geral para selecionar apenas algumas colunas:

```
SELECT column1, column2 FROM table1;
```

Selecionando uma tupla específica (com id $= 23$):

```
SELECT * FROM table1 where id = 23;
```

SELECT - AS

É possível apelidar colunas retornadas de um SELECT.

```
SELECT a AS value, b + c AS sum FROM ...
```

SELECT - ORDER BY

É possível ordenar os resultados de um SELECT

```
SELECT a, b FROM table1 ORDER BY a ASC, B DESC;
```

SELECT - LIMIT

É possível limitar a quantidade de resultados retornados de um SELECT

```
SELECT * FROM table ORDER BY ... LIMIT 10
```

SELECT - OFFSET

Além de limitar a quantidade de resultados de um SELECT é possível determinar a quantidade de resultados que serão "pulados"

```
SELECT * FROM table ORDER BY ... LIMIT 10 OFFSET 2
```

SELECT - Operadores Lógicos

Tabela : Operadores Lógicos

a	b	a AND b	a or b
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	NULL	NULL	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL
NULL	NULL	NULL	NULL

a	NOT a
FALSE	TRUE
FALSE	TRUE
NULL	NULL

SELECT - Operadores de Comparação

Tabela : Operadores de Comparação

Operador	Descrição
<	Menor que
>	Maior que
<=	Menor e igual
>=	Maior e igual
=	Igual
<> ou ! =	Diferente

SELECT - Operadores de Comparação

Para verificar se um valor é ou não é nulo, use as construções:

```
expression IS NULL
expression IS NOT NULL
```

Valores booleanos podem ser testados desta forma:

```
expression IS TRUE
expression IS NOT TRUE
expression IS FALSE
expression IS NOT FALSE
expression IS UNKNOWN
expression IS NOT UNKNOWN
```

SELECT - Operadores e Funções Matemáticas

Tabela : Operadores e Funções Matemáticas

Operador	Descrição	Exemplo	Resultado
+	adição	2+3	5
−	subtração	2-3	-1
/	divisão	4/2	2
%	módulo ou resto	5%4	1
∧	exponenciação	2.0 ∧ 3.0	8.0
|/	raiz quadrada	|/25.0	5
||/	raiz cúbica	||/27.0	3
!	fatorial	5!	120
!!	fatorial prefixado	!!5	120
@	valor absoluto	@ -5.0	5

SELECT - Operadores e Funções Matemáticas

Tabela : Operadores e Funções Matemáticas

Exemplo	Resultado
$abs(-17.4)$	17.4
$cbrt(27.0)$	3
$ceil(-42.8)$	-42
$ceiling(-95.3)$	-95
$degrees(0.5)$	28.6478897565142
$div(9,4)$	2
$exp(1.0)$	2.718281828445905
$floor(-42.8)$	-43
$ln(2.0)$	0.693147180559945
$log(100.0)$	2
$ln(2.0, 64.0)$	6
$mod(9,4)$	1
$pi()$	3.14159265358979

SELECT - Operadores e Funções Matemáticas

Tabela : Operadores e Funções Matemáticas

Função	Descrição
$acos(x)$	Secante
$asin(x)$	Cossecante
$atan(x)$	Cotangente
$atan2(y, x)$	Cotangente de y/x
$cos(x)$	cosseno
$sin(x)$	seno
$tan(x)$	tangente

SELECT - Funções com Strings

Tabela : Funções com Strings

Exemplo	Resultado
'Post' || 'greSQL'	PostgreSQL
'Value:' || 42	Value: 42
`char_length`('jose')	4
lower('TOM')	tom
overlay('Txxx' placing 'hom' from 2 for 4)	Thomas
position('om' in 'Thomas')	3
substring('Thomas' from 2 for 3)	hom
trim(both 'x' from 'xTomxx')	Tom
upper('tom')	TOM

SELECT - Operador LIKE

```
'abc' LIKE 'abc' true
'abc' LIKE 'a%' true
'abc' LIKE '_b_' true
'abc' LIKE 'c' false
```

SELECT - DISTINCT

- Depois da execução de um comando SELECT, podemos, opcionalmente, eliminar linhas duplicadas.
- A palavra-chave DISTINCT deve ser escrita logo após SELECT a eliminação das possíveis linhas duplicatas.

Expressões Condicionais

```
SELECT * FROM test;
a
---
1
2
3
```

```
SELECT a,
       CASE WHEN a=1 THEN 'one'
        WHEN a=2 THEN 'two'
       ELSE 'other' END FROM test;
a | case
---+-------
1 | one
2 | two
3 | other
```

Combinando Consultas

- Os resultados de duas ou mais consultas podem ser combinados utilizando as operações com conjunto:
  - **união: (union)**
  - **intersecção: (intersect)**
  - **diferença: (except)**

```
consulta1 UNION [ALL] consulta2
consulta1 INTERSECT [ALL] consulta2
consulta1 EXCEPT [ALL] consulta2
```

Combinando Consultas

- **UNION:** anexa o resultado da consulta2 ao resultado da consulta1
  - Embora não seja garantido que as linhas retornem nesta ordem.
  - Além disso, são eliminadas todas as linhas duplicadas, do mesmo modo que o DISTINCT, a não ser que UNION ALL seja utilizado.
- **INTERSECT:** retorna todas as linhas presentes tanto no resultado da consulta1 quanto no resultado da consulta2.
  - As linhas duplicadas são eliminadas, a não ser que INTERSECT ALL seja utilizado

Combinando Consultas

- **EXCEPT:** retorna todas as linhas presentes no resultado da consulta1, mas que não estão presentes no resultado da consulta2
  - Novamente, as linhas duplicadas são eliminadas a não ser que EXCEPT ALL seja utilizado.

Lembrete:

- Para ser possível calcular a união, a intersecção ou a diferença entre duas consultas, estas duas consultas precisam ser **compatíveis para união** significando que as duas devem retornar o mesmo número de colunas, e que as colunas correspondentes devem possuir tipos de dados compatíveis.

Expressões de Subconsultas

- **EXISTS**
  - A subconsulta é avaliada para determinar se ele retorna nenhuma linha.
  - Se retornar pelo menos uma linha , o resultado de EXISTS é "verdade";
  - Se a subconsulta não retorna nenhuma linha, o resultado de EXISTS é "falso"

```
SELECT col1 FROM tab1 WHERE
                EXISTS
(SELECT 1 FROM tab2 WHERE col2 = tab1.col2);
```

Expressões de Subconsultas

- **IN**
  - O resultado do IN é "verdade"se uma linha subconsulta igual for encontrada .
  - O resultado é "falso"se nenhuma linha igual for encontrada (incluindo o caso onde a subconsulta não retorna nenhuma linha).

```
expression IN (subquery)
```

Anotação

Há também a expressão **NOT IN**

Junção de Tabelas

- CROSS JOIN
- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

Junção de Tabelas

Exemplo:

Tabela t1:

```
num | name
----+------
1 | a
2 | b
3 | c
```

Tabela t2:

```
num | value
----+-------
1 | xxx
3 | yyy
5 | zzz
```

Junção de Tabelas

```
=> SELECT * FROM t1 CROSS JOIN t2;

num | name | num | value
-----+------+-----+-------
1 | a | 1 | xxx
1 | a | 3 | yyy
1 | a | 5 | zzz
2 | b | 1 | xxx
2 | b | 3 | yyy
2 | b | 5 | zzz
3 | c | 1 | xxx
3 | c | 3 | yyy
3 | c | 5 | zzz
(9 rows)
```

Junção de Tabelas

```
=> SELECT * FROM t1
        INNER JOIN t2
                ON t1.num = t2.num;

num | name | num | value
-----+------+-----+-------
1 | a | 1 | xxx
3 | c | 3 | yyy
(2 rows)
```

Junção de Tabelas

```
=> SELECT * FROM t1 INNER JOIN t2 USING (num);
num | name | value
-----+------+-------
1 | a | xxx
3 | c | yyy
(2 rows)
```

```
=> SELECT * FROM t1 NATURAL INNER JOIN t2;
num | name | value
-----+------+-------
1 | a  | xxx
3 | c | yyy
(2 rows)
```

Junção de Tabelas

```
=> SELECT * FROM t1 LEFT JOIN t2
        ON t1.num = t2.num;
num | name | num | value
-----+------+-----+-------
1 | a | 1 | xxx
2 | b | |
3 | c | 3 | yyy
(3 rows)
```

Junção de Tabelas

```
=> SELECT * FROM t1 LEFT JOIN t2 USING (num);
num | name | value
-----+------+-------
1 | a | xxx
2 | b |
3 | c | yyy
(3 rows)
```

Junção de Tabelas

```
=> SELECT * FROM t1 RIGHT JOIN t2
                ON t1.num = t2.num;

num | name | num | value
-----+------+-----+-------
1 | a | 1 | xxx
3 | c | 3 | yyy
        | | 5 | zzz
(3 rows)
```

Junção de Tabelas

```
=> SELECT * FROM t1 FULL JOIN t2
     ON t1.num = t2.num;

 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   2 | b    |     |
   3 | c    |   3 | yyy
     |      |   5 | zzz

(4 rows)
```

Junção de Tabelas

```
=> SELECT * FROM t1 LEFT JOIN t2 ON t1.num = t2.num
       AND t2.value = 'xxx';

num | name | num | value
-----+------+-----+-------
1 | a | 1 | xxx
2 | b | |
3 | c | |

(3 rows)
```

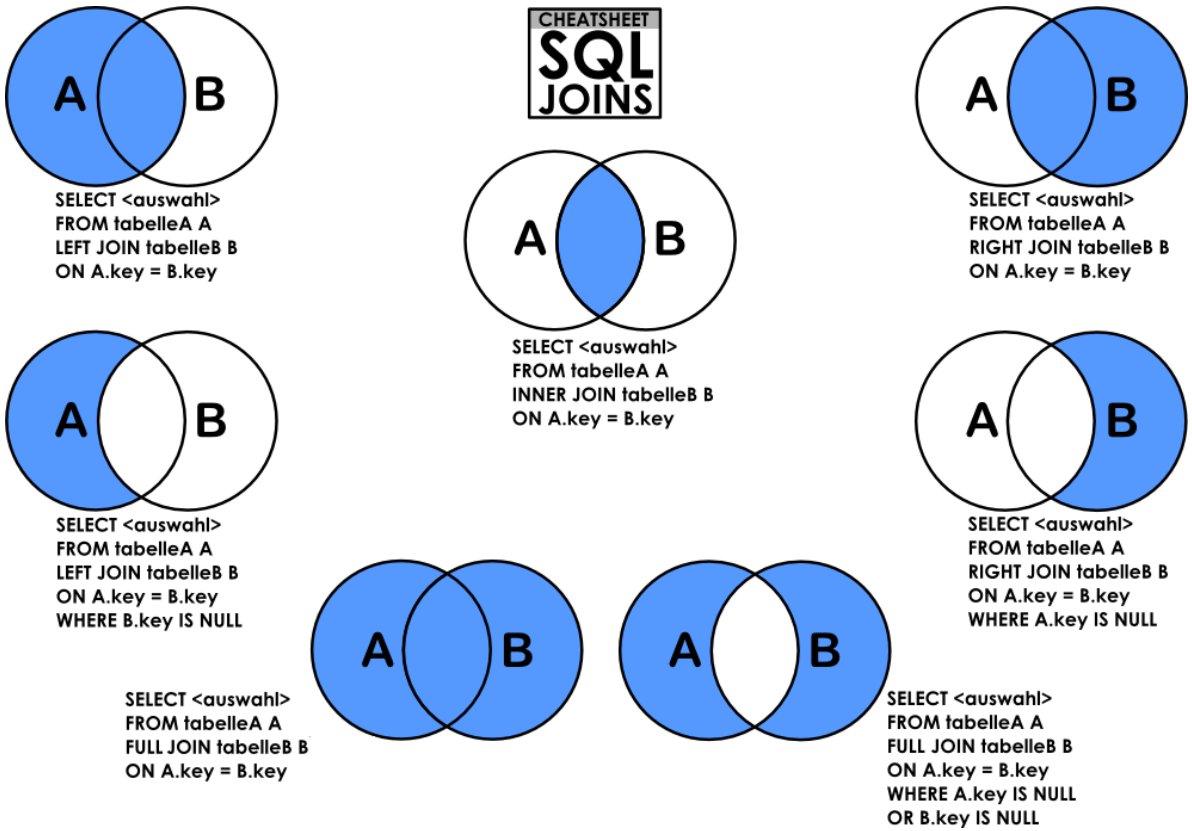
Junção de Tabelas

```
=> SELECT * FROM t1 LEFT JOIN t2 ON t1.num = t2.num
       WHERE t2.value = 'xxx';

num | name | num | value
-----+------+-----+-------
1   | a    | 1 | xxx

(1 row)
```

Junção de Tabelas



CROSS JOIN faz o produto cartesiano e não pode ser expresso em um diagrama de Venn.

Transações

- Transação é um conceito fundamental de todo SGBD.
- O ponto essencial de uma transação é que esta engloba vários passos em uma única operação de tudo ou nada
- Os estados dos passos intermediários não são visíveis para as outras transações concorrentes e, se alguma falha ocorrer que impeça a transação de chegar até o fim, então nenhum dos passos intermediários irá afetar o B.D de nenhuma forma.

```
BEGIN;
        UPDATE conta_corrente
                SET saldo = saldo - 100.00
        WHERE nome = 'Alice';
        -- etc etc
COMMIT;
```

Transações

- Se no meio da transação for decidido que está não deve ser concluída, pode ser executado o comando ROLLBACK em vez do COMMIT, fazendo com que todas as atualizações sejam canceladas.

- O PostgreSQL na verdade trata todo comando SQL como sendo executado dentro de uma transação.
- Se não for utilizado o comando BEGIN, então cada comando possui individualmente um BEGIN e um (se tudo der certo) COMMIT em torno dele.

SQL

Banco de Dados

Prof. Igor Avila Pereira
igor.pereira@riogrande.ifrs.edu.br

Divisão de Computação
Instituto Federal do Rio Grande do Sul - IFRS - Câmpus Rio Grande