Sistemas de Banco de Dados

Fundamentos em Bancos de Dados Relacionais

Wladmir Cardoso Brandão
www.wladmirbrandao.com

Fevereiro, 2020

Seção 04

SQL (Structured Query Language)

SQL

- ***Structured Query Language***: Linguagem de Consulta Estruturada.
- Oferece uma interface de linguagem *declarativa* de nível mais alto.
- O usuário apenas especifica *qual* deve ser o resultado, deixando a otimização real e as decisões sobre como executar a consulta para o SGBD.
- Linguagem padrão para SGBDs relacionais comerciais.

SQL

Linguagem abrangente de banco de dados:

- Contém instruções para definição de dados, consultas e atualizações.

Facilidades oferecidas:

- Definição de visões sobre o banco de dados;
- Especificação de segurança e autorizações;
- Definição de restrições de integridade;
- Especificação de controles de transações.

Definições e tipos de dados em SQL

Termos equivalentes ao modelo relacional:

- Relação → Tabela
- Tupla → Linha
- Atributo → Coluna

O principal comando para definição de dados é o **CREATE**, que pode ser usado para criar **esquemas, tabelas** e **domínios**.

Definições e tipos de dados em SQL

ESQUEMA

- Agrupa tabelas e outras construções que pertencem a mesma aplicação de banco de dados.

Identificado por:

- **Nome do esquema**;
- **Identificador de autorização** (indica o usuário proprietário do esquema);
- Descritores para cada **elemento** (*tabelas, restrições, views, domínios e outras construções*).

Criação de esquema EMPRESA que pertence ao usuário identificado por 'Jsilva':

```
CREATE SCHEMA EMPRESA AUTHORIZATION 'Jsilva';
```

Definições e tipos de dados em SQL

CATÁLOGO

- Coleção nomeada de esquemas em um ambiente SQL.
- Sempre contém o esquema **INFORMATION_SCHEMA** (*padrão*): oferece informações sobre todos os esquemas no catálogo e os descritores dos elementos.

Definições e tipos de dados em SQL

CREATE TABLE

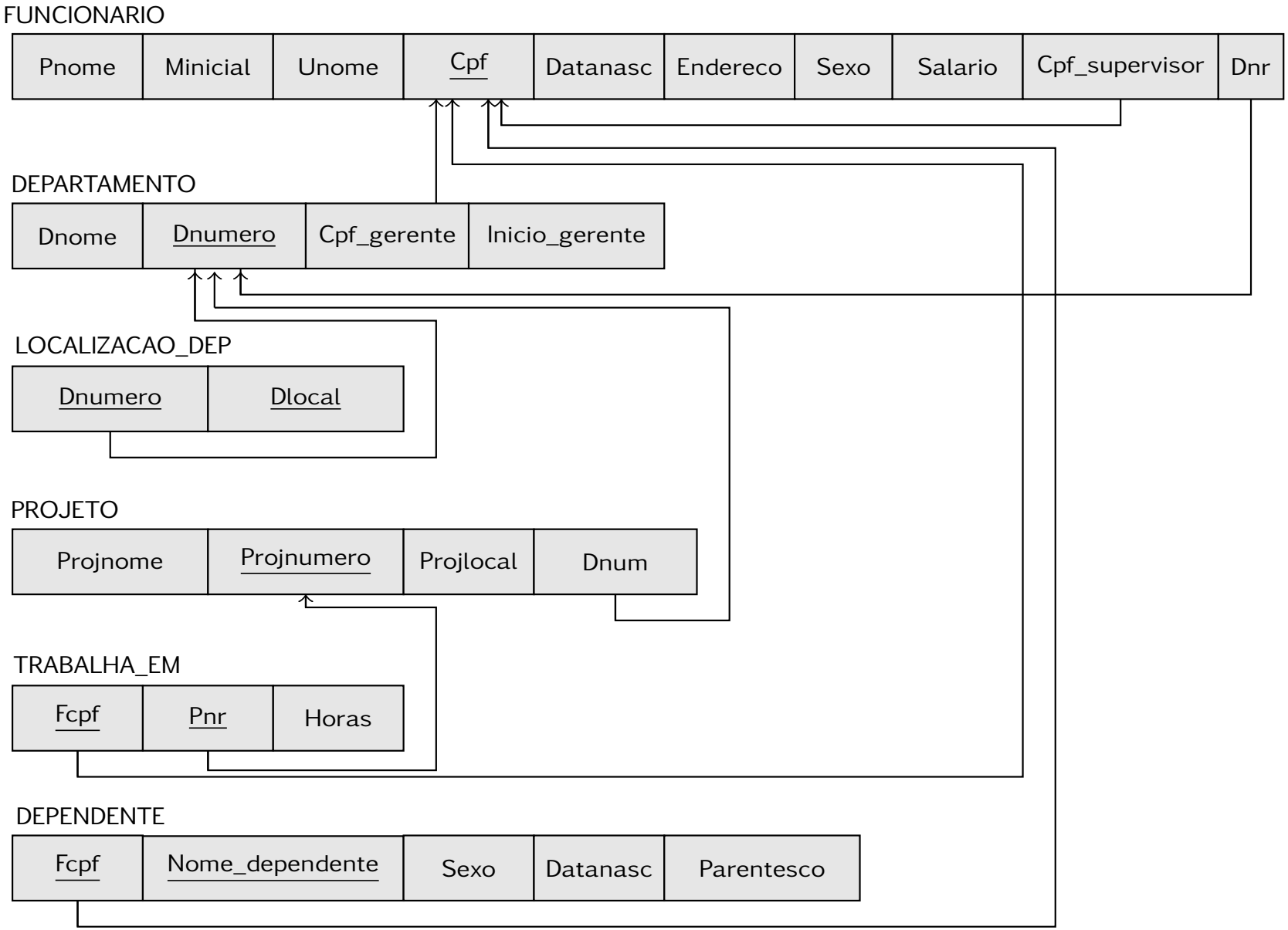
- Especifica uma nova *relação*, dando-lhe um *nome* e especificando seus *atributos e relações iniciais*.
- As restrições de **chave, integridade de entidade** ou **referencial** podem ser especificados com esta cláusula. Depois que os atributos forem criados, são definidos a partir do **ALTER TABLE.**

Exemplo:

```
CREATE TABLE FUNCIONARIO;
```

O esquema em que as relações são declaradas é especificado implicitamente no ambiente em que as instruções CREATE TABLE são executadas.

BD Relacional Exemplo



Definições e tipos de dados em SQL

TABELAS BASE

- Relações declaradas por meio das instruções CREATE TABLE.
- A relação e suas tuplas são realmente criadas e armazenadas como um arquivo pelo SGBD.

RELAÇÕES VIRTUAIS

- Relações declaradas por meio das instruções CREATE VIEW.
- A relação e suas tuplas podem ou não corresponder a um arquivo físico real.

Tipos de dados de atributos

NUMÉRICO

- Incluem números inteiros de vários tamanhos (INTEGER e SMALLINT) e números de ponto flutuante (reais) de várias posições (FLOAT/REAL e DOUBLE PRECISION).

CADEIA DE CARACTERES

- **Tamanho fixo** - CHAR(n), onde n é a quantidade de caracteres.
- **Tamanho variável** - VARCHAR(n), onde n é a quantidade máxima de caracteres.
- Para especificar um valor literal de cadeia de caracteres, este é colocado entre aspas simples e é *case sensitive*.

Tipos de dados de atributos

CADEIA DE BITS

- **Tamanho fixo** - BIT(n), onde n é a quantidade de bits.
- **Tamanho variável** - BIT VARYING(n), onde n é a quantidade máxima de bits.
- O valor padrão para n é 1.
- Os literais de cadeia de bits literais são colocados entre apóstrofos, mas precedidos por um B para distingui-los das cadeias de caracteres. Por exemplo: **B'10101'**.

BOOLEANO

- Valores tradicionais TRUE (verdadeiro) ou FALSE (falso).

Tipos de dados de atributos

DATE

- Possui dez posições e seus componentes são DAY (dia), MONTH (mês) e YEAR (ano) na forma **DD-MM-YYYY**.
- **Tamanho variável** - BIT VARYING(n), onde n é a quantidade máxima de bits.
- O tipo **TIME** tem ao menos oito posições, com os componentes HOUR (hora), MINUTE (minuto) e SECOND (segundo) na forma **HH:MM:SS**.
- Somente datas e horas válidas devem ser permitidas pela implementação SQL.

Tipos de dados de atributos

TIMESTAMP

- Inclui os campos DATE e TIME, mais um mínimo de seis posições para frações decimais de segundos e um qualificador opcional WITH TIME ZONE.

- Valores literais são representados por cadeias entre apóstrofos precedidos pela palavra-chave TIMESTAMP na forma: **TIMESTAMP '27-09-2008 09:12:47.648302'**.

Restrições em SQL

- Restrições básicas podem ser especificadas em SQL como parte da criação de tabela.
- Incluem **restrições de chave e integridade referencial, restrições sobre domínios de atributos e NULLs e restrições sobre tuplas individuais dentro de uma relação.**

Restrições em SQL

Restrição NOT NULL

- Pode ser especificada se o valor NULL não for permitido para determinado atributo.
- Sempre especificado de maneira implícita para os atributos que fazem parte da chave primária.

Exemplo:

```
CREATE TABLE FUNCIONARIO
(Pnome VARCHAR(15) NOT NULL,
 Unome VARCHAR(15) NOT NULL,
 Cpf CHAR(11), NOT NULL,
 Cpf_supervisor CHAR(11), NOT NULL,
PRIMARY KEY (Cpf));
```

Restrição DEFAULT

- Define um valor padrão para um atributo em sua definição.
- O valor padrão será incluído em qualquer nova tupla se um valor explícito não for fornecido para esse atributo.
- Se nenhuma cláusula default for especificada, o *valor padrão* será NULL para atributos *que não possuem* a restrição NOT NULL.

Restrições em SQL

Restrição CHECK

- Limita os valores de atributo ou domínio após sua definição.

Exemplo:

Suponha que os números de departamento sejam restritos a números inteiros entre 1 e 20; então, podemos mudar a declaração de atributo de **Dnumero** na tabela DEPARTAMENTO para o seguinte:

```
DNUMERO INT NOT NULL CKECK
(DNUMERO > 0 AND Dnumero < 21);
```

Restrições em SQL

Restrição PRIMARY KEY

- Especifica um ou mais atributos que compõem a chave primária de uma relação.
- Se a chave primária tiver apenas um atributo, a cláusula pode acompanhar o atributo diretamente.

Exemplo:

```
CREATE TABLE FUNCIONARIO
(Cpf CHAR(11), NOT NULL,
PRIMARY KEY (Cpf));
```

Restrição UNIQUE

- Especifica chaves alternativas (secundárias).
- Pode ser especificada diretamente para uma chave secundária se esta for um único atributo, como exemplo:

```
CREATE TABLE DEPARTAMENTO
(Dnome VARCHAR(15) NOT NULL,
 Dnumero INT NOT NULL,
PRIMARY KEY (Dnumero),
UNIQUE (Dnome));
```

Restrições em SQL

Restrição FOREIGN KEY

- Integridade referencial - chave estrangeira.

Exemplo:

```
CREATE TABLE LOCALIZACAO_DEP
(Dnumero INT NOT NULL,
 Dlocal VARCHAR(15) NOT NULL,
PRIMARY KEY (Dnumero, Dlocal),
FOREIGN KEY (Dnumero) REFERENCES DEPARTAMENTO(Dnumero));
```

- Uma restrição de integridade pode ser violada quando tuplas são inseridas ou excluídas, ou quando um valor de atributo de chave estrangeira ou chave primária é modificado.
- A ação default para evitar a violação da integridade é **rejeitar** a operação de atualização - opção RESTRICT.

Restrição FOREIGN KEY

- Ação de disparo referencial - definido pelo projetista de banco de dados. Especifica uma ação alternativa para os casos de violação de integridade:

Opções: **SET NULL, CASCADE** e **SET DEFAULT**

As opções acima devem ser escolhidas com **ON DELETE** (remoção) ou **ON UPDATE** (atualização).

Restrição FOREIGN KEY - Exemplo

- Se o projetista escolhe ON DELETE SET NULL e ON UPDATE CASCADE para a chave estrangeira Cpf_supervisor de FUNCIONARIO:

```
FOREIGN KEY (Cpf_supervisor) REFERENCES FUNCIONARIO(Cpf)
ON DELETE SET NULL ON UPDTE CASCADE);
```

*Se a tupla para um **funcionário superior é excluída**, o valor de Cpf_supervisor será automaticamente definido como NULL para todas as tuplas de funcionários que estavam referenciando a tupla do funcionário excluído.*

*Se o valor de Cpf para um funcionário supervisor é **atualizado**, o novo valor será **propagado em cascata** de Cpf_superior para todas as tuplas de funcionário que referencia a tupla de funcionário atualizada.*

Definição de nome a restrições

- Uma restrição pode receber um **nome de restrição**, seguindo a palavra-chave **CONSTRAINT**;
- Os nomes de todas as restrições de um esquema precisam ser exclusivos/únicos.

Exemplo:

```
CONSTRAINT CHPFUNC
  PRIMARY KEY (Cpf);

CONSTRAINT CHESUPERFUNC
  FOREIGN KEY (Cpf_supervisor) REFERENCES FUNCIONARIO(Cpf)
  ON DELETE SET NULL ON UPDTE CASCADE);
```

Recuperação de dados em SQL

- O **SELECT** é uma instrução básica para recuperar informações de banco de dados.
- A seguir, apresentaremos os recursos da SQL para *consultas de recuperação simples*.

Estrutura SELECT-FROM-WHERE

- A forma básica do comando SELECT, também chamado do **mapeamento** ou **bloco select-from-where**, é composta por três cláusulas, no seguinte formato:

SELECT <lista atributos>

FROM <lista tabelas>

WHERE <condição>;

onde:

- **<lista atributos>** é uma lista de nomes de atributo cujos valores devem ser recuperados pela consulta.
- **<lista tabelas>** é uma lista dos nomes de relação exigidos para processar a consulta.
- **<condição>** é uma expressão condicional (booleana) que identifica as tuplas a serem recuperadas pela consulta.

Estrutura SELECT-FROM-WHERE

- Os operadores básicos de comparação lógicos para comparar valores de atributo entre si e com constantes literais são =, <, <=, >, >= e <>.

Estrutura SELECT-FROM-WHERE

EXEMPLO 01

- Recuperar a data de nascimento e o endereço do(s) funcionário(s) cujo nome seja 'João B. Silva':

```
SELECT Datanasc, Endereco
FROM   Funcionario
WHERE  Pnome = 'Joao'
       AND Minicial = 'B'
       AND Unome = 'Silva';
```

Resultado da consulta:

Datanasc	Endereco
09-01-1965	Rua das Flores, 751, São Paulo, SP

Estrutura SELECT-FROM-WHERE

EXEMPLO 02

- Recuperar o nome e o endereço de todos os funcionários que trabalham para o departamento 'Pesquisa':

```
SELECT Pnome, Unome, Endereco
FROM   Funcionario, Departamento
WHERE  Dnome = 'Pesquisa' AND Dnumero = Dnr;
```

Resultado da consulta:

Pnome	Unome	Endereco
João	Silva	Rua das Flores, 751, São Paulo, SP
Fernando	Wong	Rua da Lapa, 34, São Paulo, SP
Ronaldo	Lima	Rua Rebouças, 65, Piracicaba, SP
Joice	Leite	Av. Lucas Obes, 74, São Paulo, SP

Estrutura SELECT-FROM-WHERE

EXEMPLO 03

- Para cada projeto localizado em 'Mauá', liste o número do projeto, o número do departamento que o controla e o sobrenome, endereço e data de nascimento do gerente do departamento:

```
SELECT Projnumero, Dnum, Unome, Endereco, Datanasc
FROM   Projeto, Departamento, Funcionario
WHERE  Dnum = Dnumero
       AND Cpf_gerente = Cpf
       AND Projlocal = 'Maua'
```

Resultado da consulta:

Projnumero	Dnum	Unome	Endereco	Datanasc
10	4	Souza	Av. Artur de Lima, 54, Santo André, SP	20-06-1941
30	4	Souza	Av. Artur de Lima, 54, Santo André, SP	20-06-1941

Ambiguidade de nomes de atributos

- O mesmo nome pode ser usado para dois (ou mais) atributos, desde que estes estejam em *relações diferentes*;

- É preciso *qualificar* o nome do atributo com o nome da relação para evitar ambiguidade - isso é feito prefixando o nome da relação (ou apelido) ao nome do atributo.

Ambiguidade de nomes de atributos

EXEMPLO 04

- Para cada funcionário, recupere o primeiro e o último nome do funcionário e o primeiro e último nome de seu supervisor imediato:

```
SELECT F.Pnome, F.Unome, S.Pnome, S.Unome
FROM   Funcionario AS F,
       Funcionario AS S
WHERE  F.Cpf_supervisor = S.Cpf;
```

Cláusula WHERE não especificada

- A falta de uma cláusula WHERE indica que **não há condições sobre a seleção das tuplas**.
- Todas as tuplas da relação especificada na cláusula FROM são selecionadas para o resultado da consulta.
- Se mais de uma relação for especificada e não houver uma cláusula WHERE, serão selecionadas todas as combinações possíveis entre as tuplas.

Uso de asteriscos

- Usado para recuperar todos os atributos de todas as tuplas selecionadas, não precisando listar seus nomes explicitamente.

EXEMPLO 05

- Recuperar todos os valores de atributo de qualquer FUNCIONARIO que trabalha no DEPARTAMENTO número 5:

```
SELECT *
FROM   Funcionario
WHERE  Dnr = 5
```

Tabelas como conjuntos

- As tabelas não são tratadas como um conjunto, mas como um **multiconjunto**.
- *Tuplas duplicadas podem aparecer mais de uma vez* em uma tabela e no resultado de uma consulta. A SQL não elimina automaticamente tuplas duplicadas nos resultados das consultas.
- Para eliminar as tuplas duplicadas no resultado da consulta, utiliza-se a palavra-chave **DISTINCT** na cláusula **SELECT**.

Apenas as tuplas distintas deverão permanecer no resultado.

Tabelas como conjuntos

EXEMPLO 06

- Recuperar o valor do salário de cada funcionário (sem tratamento de valores duplicados):

```
SELECT Salario
FROM   Funcionario
```

- Resultado da consulta:

Salário
30.000
40.000
25.000
43.000
38.000
25.000
25.000
55.000

Tabelas como conjuntos

EXEMPLO 06

- Recuperar o valor do salário de cada funcionário (com tratamento de valores duplicados):

```
SELECT DISTINCT Salario
FROM   Funcionario
```

- Resultado da consulta:

Salário
30.000
40.000
25.000
43.000
38.000
55.000

Comparação de subcadeias

Recursos adicionais SQL:

1. **LIKE**
   - Condição de comparação apenas sobre partes de uma cadeia de caracteres;
   - Cadeias parciais são especificadas usando dois caracteres reservados: **%** substitui um número qualquer, de zero ou mais caracteres, e o sublinhado (_) substitui um único caracter.

EXEMPLO 07

- Recuperar todos os funcionários cujo endereço esteja em São Paulo, SP:

```
SELECT Pnome, Unome
FROM   Funcionario
WHERE  Endereco LIKE '%SaoPaulo,SP%';
```

Comparação de subcadeias

Recursos adicionais SQL:

2 **BETWEEN** *(entre)*

EXEMPLO 08

- Recuperar todos os funcionários no departamento 5 cujo salário esteja entre R$30.000 e R$40.000:

```
SELECT *
FROM   Funcionario
WHERE  (Salario BETWEEN 30.000 AND 40.000)
       AND Dnr = 5;
```

Ordenação de resultados

- A SQL permite que o usuário ordene as tuplas do resultado de uma consulta pelos valores de um ou mais atributos que aparecem.
- Cláusula **ORDER BY**.
- A ordem default é a crescente de valores.
- Pode-se especificar a palavra-chave **DESC** caso queira exibir o resultado em ordem descrescente de valores.

Ordenação de resultados

EXEMPLO 09

- Recuperar uma lista dos funcionários e dos projetos em que estão trabalhando, ordenada por departamento e, dentro de cada departamento, ordenada alfabeticamente pelo sobrenome, depois pelo nome.

```
SELECT   D.nome, F.Unome, F.Pnome, P.Projnome
FROM     Departamento D, Funcionario F,
         Trabalha_em T, Projeto P
WHERE    D.numero = F.Dnr
         AND F.Cpf = T.Fcpf
         AND T.Pnr = P.Projnumero
ORDER BY D.Dnome, F.Unome, F.Pnome;
```

Comparações com NULL

Significados de **NULL**:

- Valor desconhecido;
- Valor indisponível ou retido;
- Atributo não aplicável.

A SQL permite consultas que verificam se o valor de um atributo é NULL.

- **IS** ou **IS NOT NULL**

Comparações com NULL

Exemplo: Recuperar os nomes de todos os funcionários que não possuem supervisores.

```
SELECT Pnome, Unome
FROM FUNCIONARIO
WHERE Cpf_supervisor IS NULL;
```

Consultas aninhadas

- São blocos **SELECT-FROM-WHERE** completos dentro da cláusula WHERE de outra consulta.
- Consulta externa.

Operador de comparação IN

- Compara um valor *v* com um conjunto (ou multiconjunto) de valores *V*.
- Avalia como TRUE se *v* for um dos elementos em *V*.

```
SELECT DISTINCT Fcpf
FROM TRABALHA_EM
WHERE (Pnr, Horas) IN (SELECT Pnr, Horas
                       FROM TRABALHA_EM
                       WHERE Fcpf = '12345678966');
```

Função EXISTS

Operador de comparação IN

- Verificar se o resultado de uma consulta aninhada correlacionada é vazio ou não.

Operador de comparação IN

- Costumam ser usados em conjunto com uma consulta aninhada correlacionada.

Exemplo: Recuperar os nomes de funcionários que não possuem dependentes.

```
SELECT Pnome, Unome
FROM FUNCIONARIO
WHERE NOT EXISTS (SELECT *
                  FROM DEPENDENTE
                  WHERE Cpf = Fcpf);
```

Funções de agregação

Utilizadas para resumir informações de várias tuplas em uma síntese de tupla única.

Funções de agregação embutidas:

- **COUNT**, **SUM**, **MAX**, **MIN** e **AVG**

Essas funções podem ser usadas na cláusula **SELECT** ou em uma cláusula **HAVING**.

Valores NULL são descartados quando as funções de agregação são aplicadas a um atributo.

Funções de agregação

Exemplo: Achar a soma dos salários de todos os funcionários do departamento 'Pesquisa', bem como o salário máximo, o salário mínimo e a média dos salários nesse departamento.

```
SELECT SUM (Salario), MAX (Salario),
       MIN (Salario), AVG (Salario)
FROM (FUNCIONARIO JOIN DEPARTAMENTO ON Dnr = Dnumero)
WHERE Dnome = 'Pesquisa';
```

Funções de agregação

Exemplo: Recuperar o número de funcionários no departamento 'Pesquisa'.

```
SELECT COUNT(*)
FROM FUNCIONARIO, DEPARTAMENTO
WHERE Dnr = Dnumero AND Dnome = 'Pesquisa';
```

Exemplo: Contar o número de valores de salário distintos no BD.

```
SELECT COUNT (DISTINCT Salario)
FROM FUNCIONARIO;
```

Cláusulas GROUP BY e HAVING

Particionam a relação em subconjuntos de tuplas, com base no atributo(s) de agrupamento.

Cláusula GROUP BY

- Especifica os atributos de agrupamento.
- Se há NULLs no atributo de agrupamento, um grupo separado é criado para todas as tuplas que satisfazem tal condição.

Cláusula HAVING

- Oferece uma condição sobre a informação de resumo.

Cláusulas GROUP BY e HAVING

Exemplo: Para cada departamento, recuperar o número do departamento, o número de funcionários no departamento e seu salário médio.

```
SELECT Dnr, COUNT (*), AVG (Salario)
FROM FUNCIONARIO
GROUP BY Dnr;
```

Cláusulas GROUP BY e HAVING

Exemplo: Para cada projeto em que mais de dois funcionários trabalham, recupere o número e o nome do projeto e o número de funcionários que trabalham no projeto.

```
SELECT Projnumero, Projnome, COUNT (*)
FROM PROJETO, TRABALHA_EM
WHERE Projnumero=Pnr
GROUP BY Projnumero, Projnome
HAVING COUNT (*) > 2
```

Cláusulas GROUP BY e HAVING

Em resumo:

```
SELECT
FROM
[WHERE <condicao>]
[GROUP BY <atributo(s) de agrupamento>]
[HAVING <condicao de grupo>]
[ORDER BY <lista de atributos>];
```

O comando INSERT

- É usado para acrescentar uma única tupla a uma relação.
- É necessário especificar o nome da relação e uma lista de valores para a tupla.
- Valores devem ser listados na mesma ordem em que os atributos correspondentes são especificados na relação.

O comando INSERT

- Acrescentar uma nova tupla a relação **FUNCIONARIO**.

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr

```
INSERT INTO FUNCIONARIO
VALUES      ( 'Ricardo' , 'K', 'Marini',
            '65329865388', '30-12-
            1962', 'Rua Itapira, 44,
            Santos, SP', 'M', 37.000,
            '65329865388', 4 );
```

O comando INSERT

- Também é possível que o usuário especifique nomes de atributos que correspondem aos valores fornecidos no comando INSERT.
- É necessário especificar o nome da relação e uma lista de valores para a tupla.
- *Inserir uma tupla para um novo **FUNCIONARIO** do qual conhecemos apenas os atributos **Pnome**, **Unome**, **Dnr** e **Cpf**.*

```
INSERT INTO  FUNCIONARIO (Pnome, Unome,
             Dnr, Cpf)
VALUES       ( 'Ricardo', 'Marini', 4,
             '65329865388' );
```

O comando INSERT

- Atributos não especificados são definidos como seu valor DEFAULT ou NULL, os valores são listados na mesma ordem que os atributos são listados no próprio comando INSERT.

O comando INSERT

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

Não existe uma tupla **DEPARTAMENTO** no banco de dados com **Dnumero = 2**.

```
INSERT INTO  FUNCIONARIO (Pnome, Unome,
             Cpf, Dnr)
VALUES       ( 'Roberto', 'Gomes',
             '98076054011', 2 );
```

Operação rejeitada

O comando INSERT

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

Nenhum valor de ***Cpf*** é fornecido e essa é a chave primária, que não pode ser NULL.

```
INSERT INTO  FUNCIONARIO (Pnome, Unome,
             Dnr)
VALUES       ( 'Roberto', 'Gomes', 5 );
```

Operação rejeitada

O comando INSERT

- *Criar uma tabela temporária que possui o sobrenome do funcionário, o nome do projeto e as horas por semana para cada funcionário que trabalha em um projeto.*

```
CREATE TABLE   TRABALHA_EM_INFO
( Func_nome VARCHAR(15),
( Proj_nome VARCHAR(15),
  Horas_semanal DECIMAL(3, 1) );
INSERT INTO    TRABALHA_EM_INFO
               ( Func_nome, Proj_nome,
               Horas_por_semana)
SELECT         F.Unome, P.Projnome, T.Horas
FROM           PROJETO P, TRABALHA_EM T,
               FUNCIONARIO F
WHERE          P.Projnumero = T.Pnr AND T.Fcpf = F.Cpf;
```

O comando DELETE

- Remove tuplas de uma relação.
- As tuplas são explicitamente excluídas de apenas uma tabela por vez.
- No entanto, a exclusão pode se propagar para as tuplas em outras relações, de acordo com as restrições de integridade.

O comando DELETE

- Uma cláusula WHERE inexistente especifica que todas as tuplas na relação deverão ser excluídas;
- A tabela permanece no banco de dados como uma tabela vazia.

O comando DELETE

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

```
DELETE FROM  FUNCIONARIO
WHERE        Unome = 'Braga';
```

Excluirá zero tuplas

O comando DELETE

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

```
DELETE FROM  FUNCIONARIO;
WHERE        Cpf = '12345678966';
```

Excluirá uma tupla

O comando DELETE

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

```
DELETE FROM  FUNCIONARIO;
WHERE        Dnr = 5;
```

Excluirá quatro tuplas

O comando DELETE

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

```
DELETE FROM  FUNCIONARIO;
```

Excluirá todas tuplas

O comando UPDATE

- Usado para modificar valores de atributo de uma ou mais tuplas selecionadas.
- Cada comando UPDATE refere-se explicitamente a apenas uma única relação.
- A atualização de uma chave primária pode ser propagada para os valores de chave estrangeira das tuplas em outras relações de acordo com as restrições de integridade.
- Uma cláusula SET adicional no comando UPDATE especifica os atributos a serem modificados e seus novos valores.

O comando UPDATE

- *Alterar o local e o número de departamento que controla o número de projeto 10 para 'Santo André' e 5, respectivamente.*

```
UPDATE  PROJETO
SET     Projlocal = 'Santo Andre', Dnum = 5
WHERE   Projnumero = 10;
```

O comando UPDATE

- *Dar a todos os funcionários no departamento 'Pesquisa' um aumento de 10 por cento no salário.*

```
UPDATE   FUNCIONARIO
SET      Salario = Salario * 1,1
WHERE    Dnr = 5;
```

OBRIGADO

Wladmir Cardoso Brandão
www.wladmirbrandao.com

"Science is more than a body of knowledge. It is a way of thinking."
Carl Sagan