SQL - Structured Query Language :
DML – Data Manipulation Language

# Comandos de Consulta em SQL
# Parte 2/2

## Junção de Relações (cont.)

- As junções externas podem ser feitas com:
  - LEFT JOIN: Retorna todas as tuplas da relação do lado esquerdo e, para apenas que têm valores iguais no campo de junção, também retorna as tuplas do dado direito;
  - RIGHT JOIN: É o inverso do LEFT JOIN;
  - FULL JOIN: Retorna todas as tuplas das relações dos dois lados e para aquelas que têm valores iguais no campo de junção, faz-se a junção.
- A palavra OUTER é opcional.




## Junção de Relações (cont.)

- Assim como para as junções internas, nas junções externas também são utilizados:
  - ON: usada quando os nomes dos campos de junção das duas relações foram diferentes;
  - USING: usado com o nomes dos campos de junção das duas relações foram iguais.

## Junção de Relações (cont.)

- Retorne todos os empregados e para aqueles que têm supervisor retorne o nome deles:

```
Q8:SELECT   E.PNOME, E.UNOME, S.PNOME, S.UNOME
   FROM     EMPREGADO E LEFT OUTER JOIN
            EMPREGADO S ON (E.SUPERSSN=S.SSN)
```




## Consultas Aninhadas

- Algumas consultas necessitam de valores presentes no BD para, então, usá-los na condição de comparação
  - Essas consultas pode ser formuladas por meio de consultas aninhadas
- Uma consulta SELECT completa,chamada consulta aninhada, pode ser especificada dentro da cláusula WHERE de outra consulta, chamada consulta externa
- Consulta 1: Recupere nome e endereço dos empregados que trabalhem no departamento de ‘Pesquisa'.

```
Q1:SELECT   PNOME, UNOME, ENDERECO
   FROM     EMPREGADO
   WHERE    DNO IN  (SELECT  DNUMERO
                     FROM     DEPARTAMENTO
                     WHERE    DNOME=‘Pesquisa' )
```

## Consultas Aninhadas (cont.)

- A consulta aninhada seleciona o número do departamento de ‘Pesquisa'
- A consulta externa seleciona uma tupla se seu valor de DNO estiver no resultado da consulta aninhada
- O operador de comparação IN compara um valor v com um conjunto (ou multiconjunto) V de valores, e avalia para verdade se v for um dos elementos em V
- Em geral, pode haver vários níveis de consultas aninhadas
- Uma referência a um atributo não qualificado é atribuída à relação declarada na consulta mais interna do aninhamento




## Consultas Aninhadas Correlacionadas

- Se uma condição na cláusula WHERE de uma consulta aninhada se referir a algum atributo da relação declarada na consulta externa, as duas consultas são chamadas correlacionadas
  - O resultado de uma consulta aninhada correlacionada é diferente para cada tupla (ou combinação de tuplas) de relações da consulta externa
- Consulta 12: Recupere o nome de cada empregado que tenha um dependente com o mesmo primeiro nome como o empregado.

```
Q12: SELECT     E.PNOME, E.UNOME
     FROM       EMPREGADO AS E
     WHERE      E.SSN IN
                    (SELECT     ESSN
                    FROM        DEPENDENTE
                    WHERE       ESSN=E.SSN AND
                    E.PNOME=DEPENDENTE_NOME)
```

## Consultas Aninhadas Correlacionadas (cont.)

- Em Q12, a consulta aninhada tem um resultado diferente na consulta externa
- Uma consulta escrita com blocos SELECT... FROM... WHERE... aninhados e usando o operador de comparação = ou IN pode sempre ser expressada como um bloco simples de consulta. Por exemplo, Q12 pode ser escrita como em Q12A

```
Q12A:  SELECT   E.PNOME, E.UNOME
       FROM     EMPREGADO E, DEPENDENTE D
       WHERE    E.SSN=D.ESSN AND
                E.PNOME=D.DEPENDENTE_NOME
```




## Consultas Aninhadas Correlacionadas (cont.)

- A especificação original da SQL também tinha um operador de comparação chamado **CONTAINS**, que era utilizado para comparar dois conjuntos ou multiconjuntos
  - Esse operador foi retirado da linguagem, possivelmente pela dificuldade de sua implementação eficiente
  - A maioria das implementações de SQL não tem esse operador
  - O operador CONTAINS compara dois conjuntos de valores e retorna TRUE se um conjunto contém todos os valores do outro

## Consultas Aninhadas Correlacionadas (cont.)

- Consulta 3: Recupere o nome de cada empregado que trabalha em todos os projetos controlados pelo departamento número 5.

```
Q3:   SELECT    PNOME, UNOME
      FROM      EMPREGADO
      WHERE  (  (SELECT    PNO
                FROM       TRABALHA_EM
                WHERE      SSN=ESSN)
                     CONTAINS
                (SELECT    PNUMERO
                FROM       PROJETO
                WHERE      DNUM=5) )
```




## Consultas Aninhadas Correlacionadas (cont.)

- EM Q3, a segunda consulta aninhada, que não é correlacionada com a consulta externa, recupera os números de todos os projetos controlados pelo departamento 5
- A primeira consulta aninhada, que é correlacionada, retorna os números dos projetos em que o empregado trabalha, que é diferente para cada tupla de empregado, por causa da correlação.

## A Função EXISTS

- EXISTS é usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio (não contém tuplas) ou não
  - A consulta 12 pode ser formulada de forma alternativa usando EXISTS




## A Função EXISTS (cont.)

- Consulta 12: Recupera o nome de cada empregado que tem um dependente com o mesmo primeiro nome com do empregado.

```
Q12B:   SELECT   PNOME, UNOME
        FROM     EMPREGADO
        WHERE    EXISTS  (SELECT   *
                          FROM     DEPENDENTE
                          WHERE    SSN=ESSN
                          AND
                 PNOME=DEPENDENTE_NAME)
```

## A Função EXISTS (cont.)

- Consulta 6: Recupere os nomes dos empregados que não têm dependentes.

```
Q6:   SELECT   PNOME, UNOME
      FROM     EMPREGADO
      WHERE    NOT EXISTS  (SELECT    *
                           FROM      DEPENDENTE
                           WHERE     SSN=ESSN)
```

- EM Q6, a consulta aninhada correlacionada recupera todas as tuplas DEPENDENTE relacionadas a uma tupla EMPREGADO. Se não existir nenhuma, a tupla EMPREGADO será selecionada
  - EXISTS é necessária para a expressividade da SQL




## Conjuntos Explícitos

- Também é possível usar um **conjunto explícito (enumerado) de valores** na cláusula WHERE ao invés de uma consulta aninhada
- Consulta 13: Recupere os números do seguro social de todos os empregados que trabalhem nos projetos número 1, 2, ou 3.

```
Q13:  SELECT   DISTINCT ESSN
      FROM     TRABALHA_EM
      WHERE    PNO IN  (1, 2, 3)
```

## NULLs em Consultas SQL

- A SQL permite que consultas verifiquem se um valor é **NULL** (desconhecido, omitido ou não aplicável)
- A SQL usa **IS** ou **IS NOT** para comparar NULLs porque cada NULL é considerado diferente de outro valores NULLs; portanto, comparação de igualdade não é apropriada.
- Consulta 14: Recupere os nomes de todos os empregados que não têm supervisor.

```
Q14:    SELECT    PNOME, UNOME
        FROM      EMPREGADO
        WHERE     SUPERSSN  IS  NULL
```

  - Nota: Se uma condição de junção for especificada, tuplas com valores NULL para o atributo de junção não são incluídas no resultado




## Funções de Agregação

- Inclui **COUNT, SUM, MAX, MIN, and AVG**
- Consulta 15: Encontre o salário máximo, mínimo e a média de salário entre todos os empregados.

```
Q15:    SELECT    MAX(SALARIO),
                  MIN(SALARIO),
        AVG(SALARIO)
        FROM      EMPREGADO
```

- Algumas implementações da SQL podem não permitir mais de uma função na cláusula SELECT

## Funções de Agregação (cont.)

- Consulta 16: Encontre o salário máximo, mínimo e a média de salário entre os empregados que trabalham para o departamento ‘Pesquisa’

```
Q16:  SELECT  MAX(SALARIO),
              MIN(SALARIO),
              AVG(SALARIO)
      FROM    EMPREGADO,
              DEPARTAMENTO
      WHERE   DNO=DNUMERO AND
              DNOME=‘Pesquisa'
```




## Funções de Agregação (cont.)

- Consulta 17 e 18: Recupere o número total de empregados da empresa (Q17), e o número de empregados do departamento ‘Pesquisa' (Q18).

```
Q17:  SELECT  COUNT (*)
      FROM    EMPREGADO

Q18:  SELECT  COUNT (*)
      FROM    EMPREGADO, DEPARTAMENTO
      WHERE   DNO=DNUMERO AND
              DNOME=‘Pesquisa’
```

## Agrupamento

- Em muitos casos, precisa-se aplicar as funções de agregação a subgrupos de tuplas de uma relação
- Cada subgrupo de tuplas consistirá em tuplas que tenham o mesmo valor em algum de seus atributos, chamado atributo(s) de agrupamento
- A função de agregação é aplicada a cada subgrupo independentemente
- A SQL tem uma cláusula **GROUP BY** para especificar os atributos de agrupamento, que devem aparecer também na cláusula SELECT




## Agrupamento (cont.)

- Consulta 20: Para cada departamento, recupere o número do departamento, o número de empregados do departamento e sua média salarial.

```
Q20:    SELECT    DNO, COUNT (*), AVG (SALARIO)
        FROM      EMPREGADO
        GROUP BY  DNO
```

  - As tuplas de EMPREGADO são divididas em grupos
    - Cada grupo tendo o mesmo valor para o atributo de agrupamento DNO
  - As funções COUNT e AVG são aplicadas a cada subgrupo de tuplas, separadamente
  - A cláusula SELECT inclui somente os atributos de agrupamento e as funções a serem aplicadas a cada subgrupo de tuplas
  - Uma condição de junção pode ser usada em conjunto com o agrupamento

## Agrupamento (cont.)

- Consulta 21: Para cada projeto, recupere o número do projeto, o nome e o número de empregados que trabalham nesse projeto.

```
Q21:    SELECT      PNUMERO, PNOME, COUNT (*)
        FROM        PROJETO, TRABALHA_EM
        WHERE       PNUMERO=PNO
        GROUP BY    PNUMERO, PNOME
```

  - Neste caso, o agripamente e a função são aplicados após a junção das duas relações




## A Cláusula HAVING

- As vezes, precisa-se recuperar valores de funções de agregação somente para os grupos que satisfazem certas condições
- A cláusula **HAVING** é usada para especificar uma condição de seleção em grupos (ao invés de em tuplas individuais)

## A Cláusula HAVING (cont.)

- Consulta 22: Para cada projeto em que mais de dois empregados trabalham, recupere o número do projeto, o nome e o número de empregados que trabalham nesse projeto.

```
Q22:    SELECT    PNUMERO, PNOME,
                  COUNT(*)
        FROM      PROJETO, TRABALHA_EM
        WHERE     PNUMERO=PNO
        GROUP BY  PNUMERO, PNOME
        HAVING    COUNT (*) > 2
```




## Comparação de Substring

- O operador de comparação **LIKE** é usado para comparar strings parciais
- Dois caracteres reservados são usados: '**%**' (ou '*****' em algumas implementações) substitui um número arbitrário de caracteres, e '_' substitui um único caractere

## Comparação de Substring (cont.)

- Consulta 25: Recupere todos os empregados cujo endereço é Houston, Texas. Aqui, o valor do atributo ENDERECO deve conter a substring 'Houston,TX'.

```
Q25:    SELECT    PNOME, UNOME
        FROM      EMPREGADO
        WHERE
                  ENDERECO LIKE '%Houston,TX%'
```




## SUBSTRING COMPARISON (contd.)

- Consulta 26: Recupere todos os empregados que nasceram durante a década de 1950s.
  - Aqui, '5' deve ser o 8º caractere da string (de acordo com o formato da data), assim o valor de DATANASC deve ser '_____5_', com cada sublinhado como um lugar para um caractere.

```
Q26:    SELECT    PNOME, UNOME
        FROM      EMPREGADO
        WHERE     DATANASC LIKE    '_____5_'
```

- A operação LIKE permite quebrar a impressão de que cada valor é atômico e indivisível
  - Consequentemente, em SQL, atributos string não são atômicos

## Operações Aritméticas

- As operações aritméticas padrão **'+', '-'. '*',** e **'/'** (para adição, subtração, multiplicação e divisão, respectivamente) podem ser aplicadas a valores numéricos em resultados de consultas SQL
- Consulta 27: Mostra o efeito de dar um aumento de 10% a todos os empregados que trabalham no projeto 'ProductX'.

```
Q27:    SELECT    PNOME, UNOME, 1.1*SALARIO
        FROM      EMPREGADO, TRABALHA_EM,
                  PROJECT
        WHERE     SSN=ESSN AND PNO=PNUMERO
                  AND PNOME='ProductX'
```




## ORDER BY

- A cláusula **ORDER BY** é usada para ordenar as tuplas no resultado de uma consulta baseado nos valores de alguns atributos
- Consulta 28: Recupere a lista de empregados e o projeto em que trabalham, ordenado pelo departamento e, dentro do departamento, ordenado alfabeticamente pelo último nome do empregado.

```
Q28:    SELECT    DNOME, UNOME, PNOME, PNOME
        FROM      DEPARTAMENTO, EMPREGADO,
                  TRABALHA_EM, PROJETO
        WHERE     DNUMERO=DNO AND SSN=ESSN
                  AND PNO=PNUMERO
        ORDER BY  DNOME, UNOME
```

## ORDER BY (cont.)

- A ordenação padrão (*default*) é em ordem ascendente de valores
- Pode-se especificar a palavra chave **DESC** se desejado ordenar de forma descendente; a palavra chave **ASC** pode ser usada para especificar explicitamente a ordenação ascendente, mesmo sendo *default*




## Resumo de Consultas SQL

- Um consulta em SQL pode consistir de seis cláusulas, mas somente as duas primeiras, SELECT e FROM, são obrigatórias. As cláusulas são especificadas na seguinte ordem:

**SELECT** <lista de atributos>
**FROM** <lista de tabelas>
[**WHERE** <condições>]
[**GROUP BY** <atributo(s) de agrupamento>]
[**HAVING** <condição de agrupamento>]
[**ORDER BY** <lista de atributos>]

# Resumo de Consultas SQL (cont.)

- A cláusula SELECT lista os atributos e funções a serem recuperadas
- A cláusula FROM especifica todas as relações (ou aliases) necessárias na consulta, mas não aquelas necessárias nas consultas aninhadas
- A cláusula WHERE especifica as condições para a seleção e junção das tuplas das relações especificadas na cláusula FROM
- A cláusula GROUP BY especifica os atributos de agrupamento
- A cláusula HAVING especifica uma condição para a seleção de grupos
- A cláusula ORDER BY especifica uma ordenação para mostrar o resultado de uma consulta
  - Uma consulta é avaliada, primeiramente, aplicando cláusula WHERE, depois a GROUP BY e a HAVING, e, finalmente,a cláusula SELECT