UNIPAC - CENTRO UNIVERSITÁRIO PRESIDENTE ANTÔNIO CARLOS
CAMPUS BARBACENA

Bacharelado em Ciência da Computação

Banco de Dados

Material de Apoio

Parte IX – DML – Estruturas Complexas

Prof. José Osvano da Silva, PMP, PSM I
joseosvano@unipac.br

1º sem / 2022

Sumário

- Estruturas mais complexas
- Comparações envolvendo NULL e lógica de três valores
- Consultas aninhadas
- As funções EXISTS e UNIQUE em SQL
- Tabelas de junção em SQL e junções externas (outer joins)
- Funções de agregação em SQL
- Exercício de Fixação.
- Exercício Avaliativo.

Estruturas mais complexas

- De agora em diante veremos estruturas mais complexas em SQL!!!
- Recursos adicionais que permitem que os usuários especifiquem recuperações mais complexas do banco de dados:
  - Consultas aninhadas, tabelas de junção, junções externas, funções agregadas e agrupamento

Comparações envolvendo NULL e lógica de três valores

- Significados de NULL
  - **Valor desconhecido**
  - **Valor indisponível ou retido**
  - **Atributo não aplicável**
- Cada valor NULL individual é considerado diferente de qualquer outro valor NULL
- SQL usa uma lógica de três valores:
  - TRUE, FALSE e UNKNOWN

Comparações envolvendo NULL e lógica de três valores

Tabela 5.1

Conectivos lógicos na lógica de três valores.

(a)

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

(b)

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

(c)

NOT			
TRUE	FALSE		
FALSE	TRUE		
UNKNOWN	UNKNOWN		

Comparações envolvendo NULL e lógica de três valores

- A SQL permite consultas que verificam se o valor de um atributo é NULL
  - IS ou IS NOT NULL

Consulta 18. Recuperar os nomes de todos os funcionários que não possuem supervisores.

```
C18:    SELECT    Pnome, Unome
        FROM      FUNCIONARIO
        WHERE     Cpf_supervisor IS NULL;
```

Consultas aninhadas, tuplas e comparações de conjunto/multiconjunto

- 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*

Consultas aninhadas

```
C4A: SELECT DISTINCT Projnumero
     FROM   PROJETO
     WHERE  Projnumero IN
            ( SELECT    Projnumero
              FROM      PROJETO,
                        DEPARTAMENTO,
                        FUNCIONARIO
                        Dnum=Dnumero AND
              WHERE     Cpf_gerente=Cpf
                        AND Unome='Silva' )
          OR Projnumero IN
            ( SELECT    Pnr
              FROM      TRABALHA_EM,
                        FUNCIONARIO
              WHERE     Fcpf=Cpf AND
                        Unome='Silva' );
```

Consultas aninhadas

- Permite o uso de tuplas de valores em comparações
  - Colocando-os entre parênteses

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

Consultas aninhadas

- Outros operadores de comparação podem ser usados para comparar um único valor *v*
  - O operador = ANY (ou = SOME)
    - Retorna TRUE se o valor *v* for igual a *algum valor* no conjunto *V* e portanto é equivalente a IN
  - Outros operadores que podem ser combinados com ANY (ou SOME): >, >=, <, <= e <>

```
C6: SELECT  Pnome, Unome
    FROM    FUNCIONARIO
    WHERE   NOT EXISTS ( SELECT*
                         FROM   DEPENDENTE
                         WHERECpf=Fcpf );
```

Consultas aninhadas

- Para evitar erros e ambiguidades em potencial
  - Criar variáveis de tupla (apelidos) para todas as tabelas referenciadas em uma consulta SQL

Consulta 16. Recuperar o nome de cada funcionário que tem um dependente com o mesmo nome e com o mesmo sexo do funcionário.

```
C16: SELECT F.Pnome, F.Unome
     FROM   FUNCIONARIO AS F
     WHERE  F.Cpf IN ( SELECT D.Fcpf
                       FROM   DEPENDENTE
                              AS D
                       WHERE  F.Pnome=
                              D.Nome_
                              dependente
                              AND
                              F.Sexo=
                              D.Sexo );
```

Consultas aninhadas correlacionadas

- Consultas aninhadas **correlacionadas**
  - É avaliada uma vez para cada tupla (ou combinação de tuplas) na consulta externa

As funções EXISTS e UNIQUE em SQL

- Função EXISTS
  - Verificar se o resultado de uma consulta aninhada correlacionada é vazio ou não
- EXISTS e NOT EXISTS
  - Costumam ser usados em conjunto com uma consulta aninhada correlacionada
- Função SQL UNIQUE(C)
  - Retorna TRUE se não houver tuplas duplicadas no resultado da consulta C

Conjuntos explícitos e renomeação de atributos em SQL

- É possível usar um conjunto explícito de valores na cláusula WHERE
- Usar o qualificador AS seguido pelo novo nome desejado
  - Renomear qualquer atributo que apareça no resultado de uma consulta

```
C8A: SELECT   F.Unome AS Nome_funcionario,
             S.Unome AS Nome_supervisor
     FROM     FUNCIONARIO AS F,
             FUNCIONARIO AS S
     WHERE    F.Cpf_supervisor=S.Cpf;
```

Tabelas de junção em SQL e junções externas (outer joins)

- **Tabela de junção**
  - Permite aos usuários especificar uma tabela resultante de uma operação de junção na cláusula FROM de uma consulta
- A cláusula FROM em C1A
  - Contém uma única tabela de junção

```
C1A: SELECT  Pnome, Unome, Endereco
     FROM    (FUNCIONARIO JOIN
               DEPARTAMENTO
               ON Dnr=Dnumero)
     WHERE   Dnome='Pesquisa';
```

Tabelas de junção em SQL e junções externas (outer joins)

- Especifique diferentes tipos de junção
  - NATURAL JOIN
  - Vários tipos de OUTER JOIN
- NATURAL JOIN sobre duas relações R e S
  - Nenhuma condição de junção é especificada
  - Condição EQUIJOIN implícita para cada par de atributos com o mesmo nome de R e S

Tabelas de junção em SQL e junções externas (outer joins)

- **Inner join**
  - O tipo padrão de junção em uma tabela de junção
  - Tupla é incluída no resultado somente se uma tupla combinar na outra relação
- LEFT OUTER JOIN
  - Toda tupla na tabela da esquerda precisa aparecer no resultado;
  - Se não tiver uma tupla combinando
    - Preenchida com valores NULL para os atributos da tabela da direita

Tabelas de junção em SQL e junções externas (outer joins)

- RIGHT OUTER JOIN
  - Toda tupla na tabela da direita precisa aparecer no resultado
  - Se não tiver uma tupla combinando
    - Preenchida com valores NULL para os atributos da tabela da esquerda
- FULL OUTER JOIN
- É possível aninhar especificações de junção

Funções de agregação em SQL

- São usadas para resumir informações de várias tuplas em uma síntese de tupla única
- **Agrupamento**
  - Cria subgrupos de tuplas antes do resumo
- Funções de agregação embutidas
  - **COUNT**, **SUM**, **MAX**, **MIN** e **AVG**
- Essas funções podem ser usadas na cláusula SELECT ou em uma cláusula HAVING

Funções de agregação em SQL

- Valores NULL são descartados quando as funções de agregação são aplicadas a determinada coluna (atributo)

Consulta 20. 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.

```
C20: SELECT  SUM (Salario), MAX (Salario),
             MIN (Salario), AVG (Salario)
     FROM    (FUNCIONARIO JOIN
             DEPARTAMENTO ON Dnr=Dnumero)
     WHERE   Dnome='Pesquisa';
```

Consultas 21 e 22. Recuperar o número total de funcionários na empresa (C21) e o número de funcionários no departamento 'Pesquisa' (C22).

```
C21: SELECT  COUNT (*)
     FROM    FUNCIONARIO;
C22: SELECT  COUNT (*)
     FROM    FUNCIONARIO, DEPARTAMENTO
     WHERE   Dnr=Dnumero
             AND Dnome='Pesquisa';
```

Agrupamento: as cláusulas GROUP BY e HAVING

- **Particionar** a relação em subconjunto de tuplas
  - Baseado no **atributo(s) de agrupamento**
  - Aplicar a função a cada grupo desse tipo independentemente
- Cláusula **GROUP BY**
  - Especifica os atributos de agrupamento
- Se houver NULLs no atributo de agrupamento
  - Um grupo separado é criado para todas as tuplas com um valor NULL no atributo de agrupamento.

Agrupamento: as cláusulas GROUP BY e HAVING

- Cláusula **HAVING**
  - Oferece uma condição sobre a informação de resumo referente ao grupo de tuplas associado a cada valor dos atributos de agrupamento.

```
C28: SELECT Dnumero, COUNT (*)
     FROM   DEPARTAMENTO, FUNCIONARIO
     WHERE  Dnumero=Dnr AND Salario>40.000 AND
            ( SELECT  Dnr IN
              FROM  FUNCIONARIO
              GROUP BY Dnr
              HAVING  COUNT (*) > 5)
```

Resumo das consultas em SQL

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

Exercício de Fixação 10

- Basei-se nas Relações Abaixo:
  - Ambulatório (<u>númeroA</u>, andar, capacidade)
  - Médico (<u>CRM</u>, rg, nome, idade, cidade, especialidade, #*númeroA*)
  - Paciente (<u>RG</u>, nome, idade, cidade, doença)
  - Consulta (#<u>*CRM*, #*RG*, data,</u> hora)
  - Funcionário (<u>RG</u>, nome, idade, cidade, salário)

1º) Criar um Banco de Dados de Consulta, caso ainda não tenha criado;
2º) Inserir 3 registros em cada tabela;
3º) Começar a realizar as consultas do Exercícios.

1. Usando SQL escreva os scripts para as necessidades abaixo:
   a) buscar o nome dos médicos que atenderam todos os pacientes
   b) buscar o andar dos ambulatórios nos quais todos os médicos ortopedistas dão atendimento
   c) buscar os dados de todos os médicos e, para aqueles que têm consultas marcadas, mostrar o nome do médico e o RG do paciente
   d) buscar os números de todos os ambulatórios e, para aqueles ambulatórios nos quais médicos dão atendimento, exibir o CRM e o nome dos médicos associados
   e) mostrar em uma relação o RG e nome de todos os pacientes e de todos os médicos, apresentando estes dados de forma relacionada para aqueles que possuem consultas marcadas

Dúvidas

José Osvano da Silva
joseosvano@unipac.br