Bacharelado em Sistemas de Informação

Banco de Dados
Aula 08
SQL – DML
Parte 3

Dr. Diego Buchinger
diego.buchinger@udesc.br

UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA

Introdução

SQL

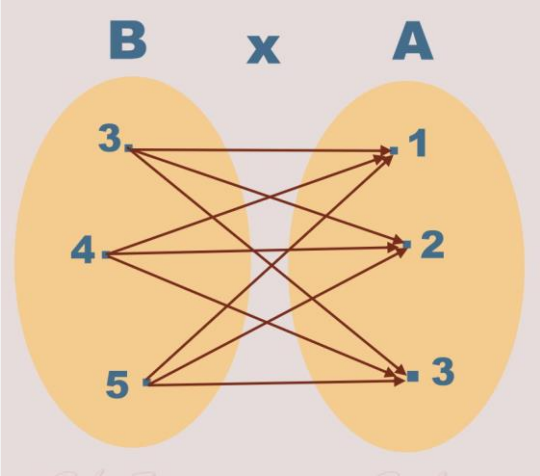
Structured Query Language

DML – *Data Manipulation Language*

Consultas Envolvendo Múltiplas Tabelas

MySQL – Consultas com Múltiplas Tabelas

- SQL prevê a operação de **PRODUTO CARTESIANO**
  - ❑ Um produto cartesiano relacionada todos os registros de uma tabela com todos os registros de outra tabela
  - ❑ Geralmente é necessário vincular as duas tabelas através de uma condição envolvendo as chaves estrangeiras




```
SELECT a1, ..., an
FROM t1, ..., tm
WHERE c
```

Execute esta consulta para ver o resultado do produto cartesiano sem condições de filtro

```
SELECT Funcionarios.nome, Medicos.nome
FROM Funcionarios, Medicos
```

MySQL – Consultas com Múltiplas Tabelas

- Exemplo: Buscar o CPF e nome dos pacientes e suas respectivas datas de consultas agendadas para o período da tarde, isto é, com hora maior do que 13h

```
SELECT cpf, nome, data
FROM pacientes, consultas
WHERE hora > '13:00' AND
    pacientes.codp = consultas.codp;
```

Aqui adicionamos o vínculo entre as tabelas utilizando a chave estrangeira

OBS: não existe conflito de nome entre as colunas **cpf**, **nome** e **data**, pois as duas primeiras só existem na tabela **pacientes** e a última só existe na tabela **consultas**. Quando as colunas são iguais é necessário usar o nome da tabela como prefixo.

MySQL – Consultas com Múltiplas Tabelas

- Exemplo 2: Buscar o nome e a cidade dos funcionários que possuem o mesmo nome de um médico
  - Neste caso existe conflito de nome entre as colunas, pois os campos nome e cidade aparecem tanto na tabela funcionários quanto na tabela médicos
  - Assim é necessário especificar o campo utilizando o nome da tabela seguido de ponto e do nome da coluna

```
SELECT funcionarios.nome, funcionarios.cidade
FROM funcionarios, medicos
WHERE funcionarios.nome = medicos.nome;
```

A condição que associa as tabelas não precisa necessariamente envolver as chaves estrangeiras, mas é muito comum que elas sejam utilizadas para associar os registros das tabelas.

MySQL – Consultas com Múltiplas Tabelas

- Exemplo 2: Buscar o nome e a cidade dos funcionários que possuem o mesmo nome de um médico
  - Para simplificar é possível renomear o nome de uma tabela e utilizar este novo nome na especificação das colunas utilizadas na consulta

```
SELECT f.nome, f.cidade
FROM funcionarios AS f, médicos AS med
WHERE f.nome = med.nome;
```

MySQL – Consultas com Múltiplas Tabelas

- SQL prevê também operações de **JUNÇÃO**

```
SELECT a1, ..., an
FROM t1 JOIN t2 ON condição_junção
WHERE c
```

- **JUNÇÕES INTERNAS**
  - Junções que mantêm apenas os elementos relacionados
  - Comando `JOIN` ou `INNER JOIN`
  - Exemplo: Buscar o CPF e nome dos pacientes e suas respectivas datas de consultas agendadas para o período da tarde (consultas agendadas a partir das 13h)

```
SELECT p.cpf, p.nome, c.data
FROM pacientes AS p JOIN consultas AS c
  ON p.codp = c.codp
WHERE hora > '13:00';
```

Aqui adicionamos o vínculo entre as tabelas utilizando a chave estrangeira

MySQL – Consultas com Múltiplas Tabelas

- **JUNÇÃO NATURAL**
  - Junção utilizando as colunas que possuem mesmo nome entre as tabelas relacionadas. Assim, a condição é implícita e não é declarada

```
SELECT a1, ..., an
FROM t1 NATURAL JOIN t2
WHERE c
```

  - Exemplo: Buscar o CPF e nome dos médicos e suas respectivas datas de consultas agendadas para o período da manhã (horário até as 12h)

```
SELECT m.cpf, m.nome, c.data
FROM medicos AS m NATURAL JOIN consultas AS c
WHERE hora <= '12:00';
```

MySQL – Consultas com Múltiplas Tabelas

- **JUNÇÕES EXTERNAS**
  - ❑ Junções que mantêm os elementos (linhas) não relacionadas de uma ou mais tabelas no resultado

OBS: Nas demais junções apenas as linhas relacionadas aparecem no resultado da consulta

```
SELECT a1, ..., an
FROM t1 LEFT|RIGHT|FULL [OUTER] JOIN t2
WHERE c
```

  - ❑ Exemplo: Listar todos os pacientes e, caso existam, as datas e horários de suas consultas.

```
SELECT p.cpf, p.nome, c.data
FROM pacientes AS p LEFT JOIN consultas AS c
ON p.codp = c.codp;
```

MySQL – Consultas com Múltiplas Tabelas

- **JUNÇÕES EXTERNAS**
  - Considerando a consulta SQL do exemplo anterior, qual seria o resultado ao usar:
    - LEFT OUTER JOIN: retorna todos os pacientes (tabela da esquerda, ligada ao FROM) e, se houver alguma consulta relacionada com um paciente, apresenta a data da consulta também. Se um paciente não tiver consulta, então o campo data vem sem valor
    - RIGHT OUTER JOIN: retorna todas as datas das consultas (tabela da direita, ligada ao JOIN) e, se houver algum paciente relacionado a consulta apresenta também os seus dados (cpf e nome)
    - FULL OUTER JOIN: retorna todos os nomes e cpfs dos pacientes e todas as datas das consultas. Caso algum registro de paciente tenha vínculo com uma consulta, vincula os registros em uma única linha.

Não existe **FULL JOIN** no MySQL, mas podemos emular seu resultado com junções LEFT e RIGHT, combinadas com união.

MySQL – Consultas com Múltiplas Tabelas

- **JUNÇÕES EXTERNAS**

  - FULL JOIN é suportado como uma sintaxe válida em outros SGBDs como PostgreSQL e MariaDB
  - Ex: como emular o FULL JOIN no MySQL:

```
SELECT p.cpf, p.nome, c.data
FROM pacientes AS p LEFT JOIN consultas AS c
ON p.codp = c.codp
UNION
SELECT p.cpf, p.nome, c.data
FROM pacientes AS p RIGHT JOIN consultas AS c
ON p.codp = c.codp
```

MySQL – Consultas com Múltiplas Tabelas

- **JUNÇÕES EXTERNAS**
  - ❑ Considerando a seguinte consulta SQL:

```
SELECT m.nome AS "nome medico", p.codp AS
  "cod paciente" FROM medicos AS m
  [JOIN] pacientes AS p ON m.cpf = p.cpf;
```

  - ❑ qual seria o resultado ao substituir [JOIN] por:

LEFT JOIN | **RIGHT JOIN** | **FULL JOIN**

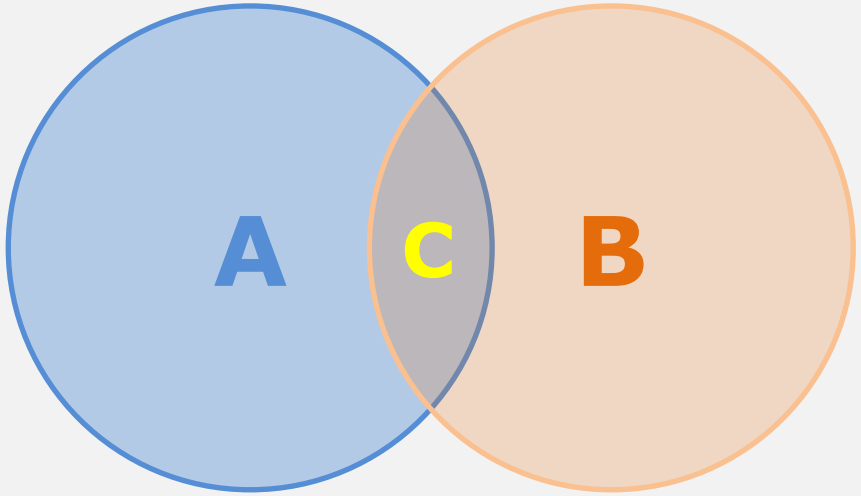
nome medico	cod paciente
João	NULL
Maria Dantas	NULL
Pedro	NULL
Carlos	4
Márcia	NULL

nome medico	cod paciente
Carlos	4
NULL	1
NULL	2
NULL	3

nome medico	cod paciente
João	NULL
Maria Dantas	NULL
Pedro	NULL
Carlos	4
Márcia	NULL
NULL	1
NULL	2
NULL	3

MySQL – Consultas com Múltiplas Tabelas

- **RESUMO DE JUNÇÕES**
  - Considere os seguintes conjuntos que representam os registros em duas tabelas A e B. Nesta relação C representa os registros que atendem a uma dada condição de junção.
  - Logo, teremos os seguintes resultados:
    - **JOIN** ou **INNER JOIN**: apenas C
    - **LEFT JOIN**: A + C
    - **RIGHT JOIN**: B + C
    - **FULL JOIN**: A + C + B



MySQL – Consultas Aninhadas

- Já vimos que é possível realizar consultas alinhadas usando a cláusula [NOT] IN, e com isso podemos realizar operações de:
  - **Diferença**: remover um conjunto de registros do resultado.
    - Ex: mostrar todos os nomes dos funcionários, exceto aqueles que tem cadastro como pacientes:

```
SELECT nome FROM funcionarios
WHERE cpf NOT IN (SELECT cpf FROM pacientes)
```

  - **Interseção**: mostrar um conjunto de registros que também aparecem em um segundo grupo de registros
    - Ex: mostrar todos os nomes dos funcionários que tenham cadastro como pacientes:

```
SELECT nome FROM funcionarios
WHERE cpf IN (SELECT cpf FROM pacientes)
```

MySQL – Consultas Aninhadas

- Podemos fazer uso também de **subconsultas unitárias**
  - ❑ Cardinalidade da subconsulta precisa ser igual a 1
  - ❑ Neste caso não é necessário utilizar cláusula de subconsulta

    - ❖ Ex: mostrar o nome e o CPF dos médicos que possuem a mesma especialidade do que o médico de CPF 10000100000 dos funcionários, exceto aqueles que tem cadastro como pacientes:

```
SELECT nome, cpf FROM medicos
WHERE cpf != 10000100000 AND
    especialidade = (
        SELECT especialidade FROM medicos
        WHERE cpf = 10000100000
    )
```

Esta subconsulta precisa retornar apenas um registro com uma única coluna

MySQL – Consultas Aninhadas

- Existem ainda as cláusulas especiais que podem ser utilizadas nas consultas aninhadas

  - **ANY**: verifica se uma dada condição é verdadeira para pela menos um valor da consulta aninhada
    - A subconsulta pode ter múltiplos registros resultantes, mas precisa ser composta por apenas uma coluna
    - Ex: buscar o nome e a idade dos médicos que são mais velhos do que pelo menos um funcionário

```
SELECT nome, idade FROM medicos
WHERE idade > ANY (
    SELECT idade FROM funcionarios
)
```

Esta subconsulta retorna uma única coluna

MySQL – Consultas Aninhadas

- Existem ainda as cláusulas especiais que podem ser utilizadas nas consultas aninhadas

  - **ALL**: verifica se uma dada condição é verdadeira para todos os valores de uma consulta aninhada
    - A subconsulta pode ter múltiplos registros resultantes, mas precisa ser composta por apenas uma coluna
    - Ex: buscar o nome e a idade dos médicos que são mais velhos do que todos os funcionários de Florianópolis

```
SELECT nome, idade FROM medicos
WHERE idade > ALL (
    SELECT idade FROM funcionarios
    WHERE cidade = 'Florianopolis';
)
```

Esta subconsulta retorna uma única coluna

MySQL – Consultas Aninhadas

- Existem ainda as cláusulas especiais que podem ser utilizadas nas consultas aninhadas

  - **EXISTS**: verifica se um predicado é verdadeiro ou falso; a subconsulta é executada para cada linha da consulta externa
    - O predicado é considerado falso se a subconsulta não retornar registros; caso contrário será verdadeiro
    - Ex: buscar o nome dos médicos que possuem uma consulta para o dia 13 de junho de 2024

```
SELECT nome FROM medicos AS m
WHERE EXISTS (
    SELECT * FROM consultas
    WHERE data = '2024-06-13'
        AND codm = m.codm
)
```

MySQL – Consultas Aninhadas

- Subconsultas na cláusula FROM / JOIN
  - ❑ Uma subconsulta pode substituir o nome de uma tabela nas cláusulas FROM e JOIN
  - ❑ Pode ser útil para otimização, filtrando linhas e colunas antecipadamente

    - ❖ Ex: buscar os dados dos médicos e a hora das consultas que estão agendadas para o dia 13 de junho de 2024

```
SELECT medicos.*, aux.hora
FROM medicos JOIN
    (SELECT codm, hora FROM consultas
     WHERE data = '2024-06-13') AS aux
     ON medicos.codm = aux.codm
```

Exercícios

1. Realize as seguintes consultas usando produto cartesiano ou junção (quando possível use junção natural)
   a. Buscar nome e CPF dos médicos (ativos ou não) que também possuem registro como pacientes da clínica.
   b. Buscar os nomes dos funcionários e médicos que residem numa mesma cidade; mostrar também qual é essa cidade.
   c. Buscar código e nome dos pacientes com consulta marcada para horários após às 14 horas.
   d. Buscar o número e andar dos ambulatórios utilizados por médicos ortopedistas (esp.: ortopedia) ativos.
   e. Buscar nome e CPF dos pacientes que tiveram consultas marcadas entre os dias 13 e 16 de junho de 2024.

Exercícios

1. Realize as seguintes consultas usando produto cartesiano ou junção (quando possível use junção natural)

   f. Buscar o nome e a idade dos médicos que têm consulta com a paciente Ana.

   g. Buscar o código e nome dos médicos que atendem no mesmo ambulatório do médico Pedro (usando somente estas informações).

   h. Buscar o nome, CPF e idade dos pacientes que tem consultas marcadas com ortopedistas para antes do dia 21/06/2024.

   i. Nome e salário dos funcionários que moram na mesma cidade da funcionária Denise e possuem salário superior ao dela.

   j. Buscar os dados de todos os ambulatórios e, para aqueles onde médicos dão atendimento, exibir também os nomes dos médicos.

   k. Para cada consulta marcada, listar o nome do médico, o nome do paciente, a data, o horário e o ambulatório utilizado.

Exercícios

2. Realize as seguintes consultas usando subconsultas com **IN**, **ANY**, **ALL** e/ou **EXISTS**, ou **subconsultas** na cláusula FROM:
   a. Buscar nome e CPF dos médicos que são pacientes do hospital
   b. Buscar código e nome dos pacientes com consulta marcada para horários após às 14 horas.
   c. Buscar o número e andar dos ambulatórios onde nenhum médico dá atendimento.
   d. Buscar o número e o andar de todos os ambulatórios, exceto o de menor capacidade.
   e. Buscar nome e CPF dos médicos que atendem em ambulatórios com capacidade superior à menor capacidade dos ambulatórios do primeiro andar (usando somente estas informações).
   f. DESAFIO: buscar nome e CPF dos médicos que têm/tiveram consultas com todos os pacientes menores de 25 anos.