SQL (Structure Query Language)

SQL

- Linguagem para:
  - Definição de dados: criação das estruturas
    - Data Definition Language (DDL)
  - Manipulação de dados: atualização e consultas
    - Data Manipulation Language (DML)

Manipulação de Dados

- Define operações de manipulação de dados
  - C (INSERT)- Create - Criar
  - R (SELECT) – Read - Ler
  - U (UPDATE) – Update - Atualizar
  - D (DELETE) – Delete - Apagar
- Instruções declarativas
  - manipulação de conjuntos
  - especifica-se o *que fazer* e não *como fazer*

Inserções, Alterações e Exclusões

SQL – Insert

- Inserção de dados

```
INSERT INTO nome_tabela [(lista_atributos)]
VALUES (lista_valores_atributos)
       [, (lista_valores_atributos)]
```

- Exemplos

```
INSERT INTO Ambulatorios VALUES (1, 1, 30)

INSERT INTO Medicos
(codm, nome, idade, especialidade, CPF, cidade)
VALUES (4, 'Carlos', 28,'ortopedia',
          11000110000, 'Betim');
```

SQL – Inserção a partir de outra tabela

- Inserção de dados

  Permite inserir em uma tabela a partir de outra tabela
  A nova tabela terá os mesmos atributos, com os mesmos domínios

- Exemplos

```
INSERT into cliente as
SELECT * from funcionario
```

SQL – Update

- Alteração de dados

```
UPDATE nome_tabela
SET nome_atributo_1 = Valor
    [{, nome_atributo_n = Valor}]
[WHERE condição]
```

- Exemplos

```
UPDATE Medico
SET cidade = 'Belo Horizonte'

UPDATE Ambulatorios
 SET capacidade = capacidade + 5, andar = 3
 WHERE nroa = 2
```

SQL – DML

- Exclusão de dados

```
DELETE FROM nome_tabela
[WHERE condição]
```

- Exemplos

```
DELETE FROM Ambulatorios

DELETE FROM Medicos
 WHERE especialidade = 'cardiologia'
or cidade < > 'Contagem'
```

Exercícios

- Crie um banco de dados Hospital e efetue as operações a seguir. Crie as tabelas e atributos que se façam necessários

  - Inserir 3 médicos na tabela de médicos
  - Cadastrar 4 ambulatórios, com numeroA sendo 1,2,3 e 4
  - Cadastrar 2 pacientes
  - Cadastrar 3 consultas
  - Alterar o numero do ambulatório de todos os médicos para 3
  - Alterar o nome do paciente 1 para Pedro da Silva

Consultas: SELECT

Estrutura Básica

- Uma consulta em SQL tem a seguinte forma:

select A_1, A_2, ..., A_n
from r_1, r_2, ..., r_m
where P

  - A_i representa um atributo
  - R_i representa uma tabela
  - P é um predicado

- Esta consulta é equivalente a uma expressão da Algebra Relacional

Estrutura Básica:

SELECT `lista de atributos desejados`

FROM `uma ou mais tabelas`

WHERE `com restrições sobre atributos`

- Exemplo: encontre o nome e o salário dos funcionarios da relação *funcionário*

SELECT *nome, salario*

FROM *funcionario*

- Equivalente a operação de PROJEÇÃO na Álgebra

$$\Pi_{nome,\ salario}\ (funcionario)$$

Distinct

- O SQL permite duplicatas em relações e resultados em consultas

- Para eliminar duplatas, usa-se a cláusula DISTINCT depois do SELECT

Exemplo:

SELECT distinct *nome*
FROM *funcionario*

A cláusula *

- O asterisco na cláusula SELECT denota TODOS OS ATRIBUTOS

SELECT *
FROM *funcionario*

- Expressões artitméticas podem ser usadas na cláusula SELECT
  +, –, *, /

- Exemplo: **SELECT** *nome, salario + 200*
  FROM *funcionario*

A cláusula FROM

- Equivale a operação de Produto Cartesiano da Álgebra
- Lista as relações envolvidas na consulta
- Exemplo: SELECT *
  FROM *funcionario, departamento*

A cláusula FROM

- Quando mais de uma tabela é utilizada é necessário dar um apelido para elas que deve ser utilizado para diferenciar atributos iguais

- Exemplo: SELECT **f.***
  FROM *funcionario f, departamento d*
  WHERE **f.codDepto** = **d.codDepto**

A cláusula WHERE

- A cláusula **where** especifica as condições que o resultado precisa satisfazer
- Corresponde ao predicado de seleção da álgebra

- Exemplo: **SELECT** nome, salario
  FROM *funcionario*
  WHERE *salario > 2000*

- operadores AND, OR e NOT podem ser usados
- Exemplo: **SELECT** nome, salario
  FROM *funcionario*
  WHERE *salario > 2000 AND idade < 30*

Renomeando atributos

- Renomeação de atributos

  old-name **as** *new-name*

- Exemplo: **SELECT** *nome* ***as*** *nomeCliente, (salario+200) as comissao*

  FROM *funcionario*

Operações com Strings

- O SQL permite comparar strings com o operador *like*
- Pode ser combinado com outros caracteres
  - % compara substrings

- Exemplo 1: encontre o nome dos funcionarios cujos nomes iniciam com “Pedro”

```
select nome
from funcionario
where nome like 'Pedro%'
```

- Exemplo 2: encontre o nome dos funcionarios cujos nomes contém “Pedro” no nome

```
select nome
from funcionario
where nome like ‘%Pedro%'
```

Operações de Conjunto

- Envolvem ao menos 2 tabelas

- Interseção e União: elimina automaticamente repetições
  - Relações precisam ser compatíveis (mesmo número de atributos)
  - Union ALL e intersects ALL preserva duplicatas

    - Encontre os clientes que tenham empréstimos e contas

      (**select** *nome* **from** *conta*) **intersect** **(select** *nome* **from** *emprestimo)*

      (**select** *nome* **from** *conta*) **union (select** *nome* **from** *emprestimo)*

Ordenando tuplas com *Order By*

- Exemplo: Liste em ordem alfabética os funcionarios que trabalham no departamento financeiro

select distinct funcionario.nome
from *funcionario, departamento*

where *funcionario.codDepto=departamento.codDepto AND*

departamento.nome='financeiro'

order by *funcionario.nome*

- Order by pode ser em ordem descendente
  - Exemplo: **order by** *nome* **desc**

Funções de Agregação

- Operam sobre múltiplos valores de uma coluna da tabela e retornam um valor

avg: média
min: valor mínimo
max: valor máximo
sum: soma de valores
count: número de valores

Funções de Agregação

- Exemplos:

  - Encontre o número de tuplas da relação CLIENTE

```
select count(*)
FROM cliente
```

  - Encontre a soma dos salarios dos funcionarios

```
select SUM(salario)
FROM funcionario
```

Funções de Agregação e Group By

- Encontre o total de funcionarios de cada departamento

select *d.nome,* ***count****(f.*) as numeroFuncionarios*

FROM funcionario f, departamento d

WHERE f.codDepto=d.codDepto

GROUP BY ***d.nome***

Funções de Agregação e Having

- A função HAVING é utilizada para aplicar condições sobre **grupos** e não sobre uma única tupla
- Exemplo: Quais são os departamentos onde a soma dos salários dos funcionários ultrapassa 50.000

```
select d.nome, sum(f.salario)

from funcionario f, departamento d

where f.codDepto=d.codDepto

group by d.nome

having (salario) > 50.000
```

Consultas Aninhadas

- Uma **subconsulta select-from-where** está aninhada dentro de outra consulta

- Exemplo: Selecione os clientes que são funcionários

select nomeCliente
from cliente
where nomeCliente **in** *(select nomeFuncionario*
from funcionario)

Valores nulos

- Consulta sobre valores inexistentes

- Exemplo: Encontre os funcionarios que não possuem carteira de habilitação
  - **select** *nome*
    from *funcionario*
    where *carteiraHabilitacao* **is null**

- OBS: cuidado que valores nulos em operações matemáticas podem dar problemas

SQL e Algebra

Álgebra	SQL
π_{nome} ((Médicos θ X θ = Médicos.codm = Consultas.codm (π_{codm} ($\sigma_{\text{data = '16/04/24'}}$ (Consultas)))))	Select nome From Médicos Where codm in (select codm from Consultas where data = '16/04/24')
(π_{CPF} (Funcionários)) — (π_{CPF} (Pacientes))	Select CPF From Funcionários Where CPF not in (select CPF from Pacientes)
(π_{CPF} (Médicos)) ∩ (π_{CPF} (Pacientes))	Select CPF From Médicos Where CPF in (select CPF from Pacientes)

Trabalho da Disciplina

Dado o Esquema Relacional:

- Ambulatório (<u>númeroA</u>, andar, capacidade)
- Médico (<u>CRM</u>, 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)

a) **Implementar o banco de dados e criar as expressões SQL DDL**
b) **Escrever as expressões em algebra relacional**
c) **Escreva, em linguagem SQL, cada inserção e consulta DML vista a seguir**
d) **Criar o conjunto com os dados necessários para satisfazer as consultas**

Exercícios

1) buscar os dados dos pacientes que estão com dengue

2) buscar os dados dos médicos cardiologistas com mais de 44 anos

3) buscar os dados das consultas, exceto aquelas marcadas para os médicos com CRM 4656 e 1879

4) buscar os dados dos ambulatórios do quarto andar que ou tenham capacidade igual a 50 ou tenham número superior a 10

5) buscar o nome e a especialidade de todos os médicos

6) buscar os números dos ambulatórios, exceto aqueles do segundo e quarto andares, que suportam mais de 50 pacientes

7) buscar o nome dos médicos que têm consulta marcada e as datas das suas consultas

8) buscar o número e a capacidade dos ambulatórios do quinto andar e o nome dos médicos que atendem neles

Exercícios

9) buscar o nome dos médicos e o nome dos seus pacientes com consulta marcada, assim como a data destas consultas

10) buscar os nomes dos médicos ortopedistas com consultas marcadas para o período da manhã (7hs-12hs) do dia 20/06/24

11) buscar os nomes dos pacientes, com consultas marcadas para os médicos João Carlos Santos ou Maria Souza, que estão com pneumonia

12) buscar os nomes dos médicos e pacientes cadastrados no hospital

13) buscar os nomes e idade dos médicos, pacientes e funcionários que residem em Ribeirão das Neves

14) buscar os nomes e RGs dos funcionários que recebem salários abaixo de R$ 3000,00 e que não estão internados como pacientes

15) buscar os números dos ambulatórios onde nenhum médico dá atendimento

16) buscar os nomes e RGs dos funcionários que estão internados como pacientes