

Banco de Dados – Ciência da Computação

SQL (*Structured Query Language*)

Prof. Mariane Moreira

mariane.souza@unifal-mg.edu.br

Universidade Federal de Alfenas

Instituto de Ciências Exatas

Departamento de Ciência da Computação

SQL

- Linguagem de consulta para banco de dados **relacionais**.
- Surgiu no início dos anos 70, nos laboratórios da IBM.
- Linguagem **descritiva** e **não-procedural**.
 - Maior parte das características baseadas na **álgebra relacional**.
- Tornou-se um **padrão** para banco de dados.
 - Simplicidade.
 - Facilidade de uso.

SQL

- Padronização:
 - SQL-86 (ANSI)
 - SQL-87 (ISO)
 - SQL-92
 - SQL:1999 (também chamado de SQL-3)
 - Adição de queries recursivas, triggers, algumas características OO.
 - SQL:2003
 - Características relacionadas à XML.
 - ...
 - SQL 2022 (segurança, escalabilidade e desempenho)

SQL

- DDL
- DML
- Linguagem transacional
- SQL embutida

DDL

- Permite a **especificação** das relações e as informações relacionadas às mesmas:
 - Criação de esquemas para relações
 - Definição de domínio dos valores de atributos
 - Restrições de integridade
 - Índices a serem mantidos para cada relação
 - Informações de segurança e autorização para cada relação
 - Estrutura de armazenamento físico de cada relação no disco
 - Criação de visões

DDL

- Atributos e tipos de domínio:
 - **char(n)**: string de caracteres de tamanho fixo com tamanho n especificado pelo usuário
 - **varchar(n)**: string de caracteres de tamanho variável com tamanho n máximo especificado pelo usuário
 - **int**: inteiro (um subconjunto finito de inteiros que é dependente da máquina)
 - **smallint**: inteiro pequeno
 - **numeric(p,d)**: número de ponto fixo, com precisão de p dígitos especificada pelo usuário, com d dígitos à direita do ponto decimal
 - **real, double precision**: números de ponto flutuante e ponto flutuante de precisão dupla com precisão dependente da máquina
 - **float(n)**: número de ponto flutuante, com precisão de pelo menos n dígitos

DDL

- Criação de tabelas:

```
create table r (A1 D1,  
               A2 D2,  
               ...,  
               An Dn,  
               (restrição-de-integridade1),  
               ...,  
               (restrição-de-integridadek))
```

DDL

- Criação de tabelas:
- Ex: criar tabela agencia com os atributos nome (50), cidade (50), ativo (inteiro), codigo (5).

```
create table agencia  
    (codigo char(5),  
     nome varchar(50),  
     ativo int,  
     cidade varchar(50))
```


DDL

- Toda tabela deve ter uma **chave primária**
 - Comando: *primary key (atributos)*

```
create table agencia  
  (codigo char(5),  
   nome varchar(50),  
   ativo int,  
   cidade varchar(50),  
   primary key (codigo))
```

DDL

- Para se especificar **chaves estrangeiras** deve-se usar o seguinte comando após a declaração do atributo que representa a chave:
 - **FOREIGN KEY**(chaveestrangeira1) **REFERENCES** tabela(campochave)

DDL

- Exemplo de criação de uma tabela em um dado esquema, com restrições de integridade:
 - `CREATE TABLE` empregado (
 ssn int `NOT NULL AUTO_INCREMENT`,
 pnome varchar(45),
 datanasc datetime,
 sexo enum('M','F'),
 salario double,
 superssn int,
 dnumero int,
 `PRIMARY KEY` (ssn)
 `FOREIGN KEY`(superssn) `REFERENCES` empregado(ssn)
 `FOREIGN KEY`(dnumero) `REFERENCES` departamento
 (dnumero))

DDL

- Exemplo de criação de uma tabela em um dado esquema, com outras restrições de integridade:
 - `CREATE TABLE empregado (`
 `ssn int NOT NULL PRIMARY KEY,`
 `pnome varchar(45),`
 `datanasc datetime,`
 `sexo enum('M','F'),`
 `salario double DEFAULT NULL,`
 `superssn int,`
 `dnumero int,`
 `UNIQUE (pnome),`
 `FOREIGN KEY(superssn) REFERENCES empregado(ssn)`
 `FOREIGN KEY(dnumero) REFERENCES departamento`
 `(dnumero))`

DDL

- Apagar (remover) tabela:
 - `drop table r`
 - Remove a tabela *r* do banco (apaga as tuplas e o esquema)
 - `delete from r`
 - Remove somente as tuplas (mantém o esquema)

DML

- Modificação estrutural de tabelas:
 - Comando geral: *alter table*
 - Adição de atributos:
Comando geral: *alter table r add a d*
Ex: alter table agencia add telefone varchar(15)
 - Remoção de atributos:
Comando geral: *alter table r drop a*
Ex: alter table agencia drop cidade

Consultas

- Estrutura básica:

```
select A1, A2, ..., An  
from r1, r2, ..., rm  
where P
```

Onde:

A1, A2, ..., An: atributos

r1, r2, ...rm: relações

P é o predicado da consulta

Consultas

- Estrutura básica
select ***A1, A2, ..., An***
from ***r1, r2, ..., rm***
where ***P***

$$\Pi_{A_1, A_2, \dots, A_n}(\sigma_P(r_1 \times r_2 \times \dots \times r_m))$$

Consultas

- **Select**
 - Lista os atributos
 - Projeção da álgebra relacional
 - Ex: Encontre os nomes de todos os empregados
select enome
from empregado
 - O * depois do select retorna todos os atributos
select *
from empregado

Consultas

- **Select**
 - SQL permite tuplas com valores duplicados
 - `Select nome from empregado`
Joao (o Joao com SSN = 001)
Joao (o Joao com SSN = 002)
Joao (o Joao com SSN = 003)
 - **Distinct** (remove tuplas com valores duplicados)
 - `Select distinct nome from empregado`
 - **ALL**
 - `Select all nome from empregado` (mesmo resultado de `Select nome from empregado`)
 - Não remove tuplas com valores duplicados
 - Aplicado em outros contextos a serem demonstrados posteriormente.

Consultas

- Cláusula **Where**
 - Define uma expressão condicional que identifica as tuplas que deverão ser recuperadas pela consulta
 - Ex: Selecione o numero dos empregados que moram no endereço: Rua 16, Casa 13.

Select enum

From empregado

Where endereco = 'Rua 16, Casa 13'

Consultas

- Cláusula **Where**

- Selecione o numero dos empregados que tenham o nome Joao M. da Silva

Select enum

From empregado

Where pnome = 'Joao' and mnome = 'M. da' and unome = 'Silva'

- Selecione o numero dos empregados que não tenham o ultimo nome igual a "Barbosa"

Select enum

From empregado

Where unome <> 'Barbosa'

Consultas

- Cláusula **Where**
 - SQL inclui o operador de comparação between
 - Inclui as extremidades

Ex: empregado(ssn,nome,salario)

Selecione o ssn de todos os empregados que ganham entre 1500 e 10000, inclusive esses valores.

Select ssn

From empregado

Where salario between 1500 and 10000

Consultas

- Cláusula **From**
 - Define as tabelas que serão consideradas nas consultas
 - A cláusula “From” isolada com duas ou mais tabelas define um **produto cartesiano**.

Ex:

```
select ssn, essn  
from empregado, trabalha_em
```

(Define o produto cartesiano com relação aos atributos ssn e essn das duas relações passadas)

Consultas

- Cláusula **From**
 - Para definir uma junção é necessário especificar a **condição de junção**.
 - Ex: Selecione o nome dos empregados que trabalham no departamento 'Pesquisa'

Select pnome

From empregado, departamento

Where

empregado.dnumero = departamento.dnumero

and dnome = 'Pesquisa'

Condição de junção

Condição de seleção

Consultas

- Renomeação

- SQL permite ainda renomear **atributos** e relações

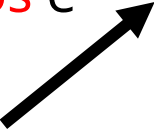
Atributos:

-Select dnome
From departamento


-Select dnome **as** nome
From departamento

Tabela:

Select * from empregado
as
empregados



dnome
Pesquisa
RH
Graduação



nome
Pesquisa
RH
Graduação

Consultas

- **Renomeação**
 - SQL permite ainda renomear atributos e **relações**:

A renomeação pode ser imprescindível em certos casos:

cliente(codigo,nome,etc)

dependente(codigo,nome, codCliente, etc ...)

Selecione o nome do cliente referente ao dependente D0001:

Select nome

From cliente, dependente

Where codCliente = codigo

and codigo = "D0001"

Qual nome será retornado? Qual codigo será considerado?

Consultas

- Renomeação

Select c.nome as nome_cliente

From cliente as c, dependente as d

where d.codCliente = c.codigo

and d.codigo = "D0001"

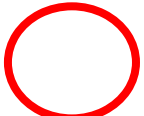
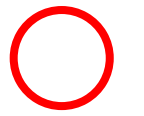
- ***c e d são variáveis de tupla***
- ***no resultado, o atributo nome de cliente é exibido como nome_cliente***

Consultas

Trabalha_em

<u>SSN</u>	<u>PNUMERO</u>	HORAS
------------	----------------	-------

- Outro exemplo com variáveis de tuplas:
 - Selecione o número dos projetos que possuem empregados que trabalham mais horas que o empregado de `ssn = 'E01'` no projeto `pnumero = 'P01'`

```
select t.pnumero    
from trabalha_em as t, trabalha_em s  
where t.horas > s.horas and  
s.ssn = 'E01' and s.pnumero= 'P01'
```

Consultas

- Operações com String
 - String são especificadas com apóstrofo
Ex: nome = 'Maria'
 - SQL permite o uso de caracteres especiais
 - % (corresponde a qualquer sequencia substring)
 - _ (corresponde a qualquer caracter)

Consultas

- Operações com String

nome ('M%')
Maria
maria
Melise
Marta Fonseca
Mauro Batista

ssn ('C_')
C1
C2

Consultas

- Operações com String
 - Operador like
 - caracteres especiais devem ser usados com o operador *like*.
 - Forma geral:

Select *

From R

Where coluna like

Consultas

- Ex. de operações com String:

Select nome

From empregado

Where nome like 'M%'

Select ssn

From empregado

Where ssn like 'C_'

nome
Maria
maria
Melise
Marta Fonseca
Mauro Batista

ssn
C1
C2

Consultas

- Operações com String
- Scape
 - Comando usado para permitir que um **caracter especial** seja especificado dentro da **string de busca**.

```
Select ssn  
From empregado  
Where ssn like 'C\%00%' escape  
  \'
```

ssn
C%001
C%0024
C%0023344

Consultas

- Operações com String
- SQL permite outras operações com string:
 - concatenação (usando “||”)
 - conversão de maiúscula em minúscula (e vice-versa)
 - upper(atributo ou string)
 - lower(atributo ou string)
 - Retornar tamanho da string
 - Extração de substrings e outras
- Algumas funções variam de um SGBD para outro

Consultas

- Ordenação
 - SQL permite a ordenação das tuplas.
 - Ex: Liste todos os empregados em ordem alfabética do primeiro nome.

Select *

From empregado

Order by pnome

- Operadores *desc* e *asc*
 - Ordem decrescente e crescente (padrão)
 - Acrescentados depois do nome do atributo

Próxima aula...

- SQL (continuação)