BANCO DE DADOS

Marinêz da Silva

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)

HISTÓRICO

- Linguagem SQUEL desenvolvida pela IBM para um banco de dados experimental R. Evoluiu e o mudou o nome para SQL (Structured Query Language).
- Baseada no padrão ANSI e ISO:
  - SQL-86
  - SQL-89
  - SQL-92
  - SQL:1999
  - SQL:2003

SQL

- SQL é considerada a razão principal para o sucesso dos bancos de dados relacionais comerciais
  - Tornou-se a linguagem padrão para bases relacionais
  - Funciona entre diferentes produtos
  - Fácil uso para o usuário

SQL como Linguagem de Definição de Dados

Permite especificar:

- O esquema de cada relação
- O domínio dos valores associados a cada atributo
- Restrições de integridade
- O conjunto de índices
- Visões
- Permissão de acesso às relações


Universidade Federal de Santa Maria
1960
Colégio Politécnico
UFSM

DDL

CRIANDO UMA BASE DE DADOS

- Criação de um BD
  - `create database` *nome_BD*
  - `drop database` *nome_BD*

CRIANDO ESQUEMAS EM SQL

- Comandos para definição de esquemas
  - **`create table`**
    - define a estrutura da tabela, suas restrições de integridade e cria uma tabela vazia
  - **`alter table`**
    - modifica a definição de uma tabela
      - atributos chave não podem ser removidos de uma tabela
      - atributos NOT NULL não podem ser inseridos em uma tabela
  - **`drop table`**
    - remove uma tabela com todas as suas tuplas

CRIAÇÃO DE TABELAS

CREATE TABLE

- Colunas são especificadas primeiro
- Depois Chaves, integridade referencial e restrições de integridade

```
CREATE TABLE <nome_da_tabela>
(C1 D1, C2 D2, ..., Cn Dn,
   ...
PRIMARY KEY <lista_de_Colunas>,
FOREIGN KEY <nome_da_coluna> REFERENCES
   <nome_tab_ref>(<nome_da_coluna_ref>));
```

  - cada C_i é uma coluna no esquema da tabela
  - D_i é o tipo de dado no domínio da coluna C_i

CRIANDO ESQUEMAS EM SQL

```
CREATE TABLE Ambulatorios (
          nroa                int,
          andar               numeric(3) NOT NULL,
          capacidade          smallint,
          PRIMARY KEY(nroa)
)

CREATE TABLE Medicos (
    codm                 int,
    nome                 varchar(40) NOT NULL,
    idade               smallint NOT NULL,
   especialidade       char(20),
   CPF                 numeric(11) UNIQUE,
   cidade              varchar(30),
   nroa                int,
   PRIMARY KEY(codm),
   FOREIGN KEY(nroa) REFERENCES Ambulatorios
)
```

CREATE TABLE

- Exemplo:

```
create table produto
(codigo integer not null,
 descricao varchar(30),
 tipo varchar (20)
PRIMARY KEY codigo)
```

Codigo	Descricao	tipo

ALTERANDO TABELAS

```
ALTER TABLE nome_tabela
ADD [COLUMN] nome_atributo_1 tipo_1 [{RIs}]
   [{, nome_atributo_n  tipo_n [{RIs}]}]
|
MODIFY [COLUMN] nome_atributo_1 tipo_1 [{RIs}]
   [{, nome_atributo_n  tipo_n [{RIs}]}]
|
DROP COLUMN nome_atributo_1
   [{, nome_atributo_n }]
|
ADD CONSTRAINT nome_RI_1 def_RI_1
   [{, nome_RI_n def_RI_n}]
|
DROP CONSTRAINT nome_RI_1
   [{, nome_RI_n}]
|
[ADD|DROP] [PRIMARY KEY ...|FOREIGN KEY ...]
```

ALTERANDO TABELAS

Exemplos:

```
ALTER TABLE medicos
ADD COLUMN endereco varchar(30)
```

```
ALTER TABLE medicos
MODIFY COLUMN endereco varchar(50)
```

```
ALTER TABLE medicos
DROP COLUMN endereco
```

```
ALTER TABLE cliente
ADD Constraint PK_Cliente
Primary Key (Codigo_cliente)
```

ALTERANDO TABELAS

Exemplos:

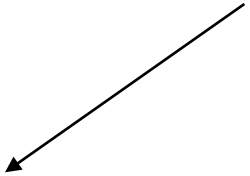
```
ALTER TABLE pedido
DROP Constraint FK_Pedido
Foreing key (Codigo_Cliente)
References Cliente(Codigo_Cliente)
```


Universidade Federal de Santa Maria
1960
Colégio Politécnico
UFSM

DOMÍNIOS

RESTRIÇÕES

- **NOT NULL**
  - Restrição aplicada em colunas cujos valores não podem ser nulos



create table *produto*
(codigo integer **not null,**
descricao varchar(30),
*tipo var*char (20)
PRIMARY KEY codigo)

TIPOS DE DOMÍNIOS EM SQL

- **char(n).** Character de tamanho *n* definido pelo usuário
- **varchar(n).**
- **int**
- **Smallint**
- **numeric(p,d)**
- **real, double precision**
- **float(n)**
- …

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
  - `INSERT`
  - `UPDATE`
  - `DELETE`
  - `SELECT`
- 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, 'Joinville');
```

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 = 'Florianopolis'

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 < > 'Florianopolis'
```

Consultas: SELECT

ESTRUTURA BÁSICA

- Uma consulta em SQL tem a seguinte forma:

$$\textbf{select } A_1, A_2, \ldots, A_n$$
$$\textbf{from } r_1, r_2, \ldots, r_m$$
$$\textbf{where } P$$

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

- O resultado de uma consulta SQL é sempre uma tabela.

Estrutura Básica: resumindo....

SELECT `lista de atributos desejados`
FROM `uma ou mais tabelas`
WHERE `com restrições sobre atributos`

- Exemplo: encontre o nome e o salário dos funcionários da relação *funcionario*

SELECT *nome, salario*

FROM *funcionario*

DISTINCT

- O SQL permite duplicatas em relações e resultados em consultas
- Para eliminar duplicatas, 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

- 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

- 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 I: encontre o nome dos funcionarios cujos nomes iniciam com "Pedro"

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

- Exemplo II: 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***

Nota: Atributos na cláusula SELECT que estão FORA da função de agregação precisam aparecer na lista de atributos do GROUP BY

VALORES NULOS

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