

Banco de Dados

Jordana S. Salamon

jssalamon@inf.ufes.br

jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

A Linguagem SQL

- Todo Banco de Dados apresenta uma Linguagem para definição e uma para manipulação de dados.
- SQL - Structured Query Language
 - Com relação aos Bancos de Dados Relacionais, é a linguagem mais utilizada
 - abrange tanto comandos de definição quanto comandos de manipulação de dados.
- A SQL foi criada pela IBM.
- Apresenta algumas extensões e variações em seus comandos de um SGBD (Sistema de Gerência de Banco de Dados) para outro, porém mantendo um determinado padrão.
- Vamos usar o MySQL para este curso.

A Linguagem SQL

- ▶ Vantagens do uso:
 - ▶ Independência de fabricante
 - ▶ Portabilidade entre computadores
 - ▶ Redução dos custos com treinamento
 - ▶ Inglês estruturado de alto nível
 - ▶ Consulta interativa
 - ▶ Múltiplas visões dos dados
 - ▶ Definição dinâmica dos dados



SQL - Comandos

▶ Definição de Dados: DDL

- ▶ Criar (Create)
- ▶ Excluir (Drop)
- ▶ Modificar (Alter)

▶ Manipulação de Dados: DML

- ▶ Consultar (Select)
- ▶ Inserir (Insert)
- ▶ Excluir/Apagar (Delete)
- ▶ Atualizar (Update)

▶ Segurança, Controle e Administração

- ▶ Commit
- ▶ Rollback
- ▶ Grant
- ▶ Revoke



A Linguagem SQL

- ▶ LDD - Linguagem de Definição de Dados (ou DDL)
- ▶ LMD - Linguagem de Manipulação de Dados (DML)



SQL - Comandos

- ▶ **/* Comentário */** Comentário dentro ou antes de um comando SQL
- ▶ **Comment** Inclui no dicionário de dados um comentário relativo a uma tabela ou coluna
- ▶ **Commit** Efetiva no banco de dados (tornando permanentes) as modificações que já foram completadas
- ▶ **Grant** Atribui privilégios relativos a tabelas e visões
- ▶ **Revoke** Revoga privilégios relativos ao BD ou acesso a tabelas
- ▶ **Rollback** Desfaz, em caso de falha do sistema, qualquer alteração que ainda não tenha sido efetivada, de modo que a integridade do BD seja mantida. Permite ao usuário desfazer qualquer alteração antes de um COMMIT .

LDD - Linguagem de Definição de Dados (ou DDL)

- ▶ **Create Table** - cria Tabelas
- ▶ **Create View** - cria Visões do Banco de Dados
- ▶ **Create Index** - define índice sobre uma Tabela
- ▶ **Alter Table** - altera a estrutura de Tabelas existentes
- ▶ **Drop Table** - elimina uma Tabela do Banco de Dados
- ▶ **Drop View** - elimina uma Visão do sistema
- ▶ **Drop Index** - elimina um índice definido para uma Tabela



SQL - Criando Tabelas

► Criando as Tabelas

Formato: **Create Table** <nome-da-tabela>
(<descrição das colunas>);
(<descrição das chaves>);

<nome da tabela> - Nome da tabela a ser criada

<Descrição das colunas> - É uma lista de colunas (campos) e seus respectivos tipos de dados (smallint, char, varchar, integer, money, decimal, float, real, date, time, timestamp, logical)

<descrição das chaves> - É a lista de colunas que são tratadas como chave primária e chave estrangeira.



Tipos de dado (Domínio) em SQL

- **char(n)**. String com tamanho fixo n .
- **varchar(n)**. String com tamanho variavel, no máximo até n .
- **int**. Inteiro
- **smallint**. Inteiro pequeno
- **numeric(p,d)**. Numero de pontos, precisão de p digitos, com n digitos a direita.
- **real, double precision**. Numero real e ponto flutuante com precisão da máquina.
- **float(n)**. Ponto flutuante com n digitos de precisão



SQL - Criando Tabelas

Create Table Cliente

```
(código_cliente smallint not null,  
nome_cliente char(20),  
endereço char(30),  
cidade char(15),  
CEP char(08),  
UF char(02),  
CGC char(20),  
IE char(20),
```

não pode ser nulo

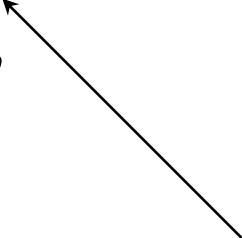


```
Primary Key (código_cliente)); /* Chave Primária */
```

SQL - Criando Tabelas

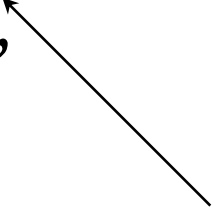
Create Table Vendedor

```
(código_vendedor    smallint not null,  
nome_vendedor      char(20),  
salario_fixo       money,  
faixa_comissão     char(01),  
Primary Key (código_vendedor);  /* Chave Primária */
```



Create Table Produto

```
(código_produto    smallint not null,  
unidade           char(03),  
descrição_produto char(30),  
val_unit          money,  
Primary Key (código_produto);  /* Chave Primária */
```



SQL - Criando Tabelas

Create Table Pedido

```
(num_pedido          integer    not null,  
prazo_entrega       smallint   not null,  
código_cliente      smallint   not null,  
código_vendedor     smallint   not null,
```

```
Primary Key (num_pedido),          /* Chave Primária */  
Foreign key (código_cliente)      /* Chave Estrangeira */  
    references Cliente  
Foreign key (código_vendedor)     /* Chave Estrangeira */  
    references Vendedor);
```

SQL - Criando Tabelas

Create Table Item_do_Pedido

```
(num_pedido      integer    not null,  
 código_produto  smallint  not null,  
 quantidade      decimal,  
 código_vendedor smallint  not null,
```

```
Primary Key (num_pedido,código_produto),
```

```
Foreign key (num_pedido)
```

```
references Pedido
```

```
Foreign key (código_produto)
```

```
references Produto);
```

```
/* Chave Primária */
```

```
/* Chave Estrangeira */
```

```
/* Chave Estrangeira */
```



Linguagem de Manipulação de Dados (ou DML)

- ▶ A DML da SQL apresenta os seguintes comandos para manipulação dos dados em um banco de dados:
- ▶ Comandos de modificação de dados:
 - ▶ Inclusão de uma ou mais linhas: ***Insert***
 - ▶ Atualização de uma ou mais linhas: ***Update***
 - ▶ Exclusão de uma ou mais linhas: ***Delete***
- ▶ Comando de consulta - ***Select***

SQL - Inserindo Registros

Adicionando tuplas (linhas) à Tabela:

Formato: INSERT INTO <nome da tabela>
(<nome das colunas>
VALUES (<valores>);

Exercício: Adicionar o produto 'Queijo' à tabela produto

```
Insert into Produto  
values (25, 'Kg', 'Queijo', 0.97); /* valores posicionais  
com os respectivos  
atributos */
```



SQL - Modificando Registros

Modificando uma Tupla (linha) da Tabela

Formato: Update <nome da tabela>
 set <nome da coluna> = valor
 where <condição>;

Update Produto
 set val_unit = 1.62
 where descrição_produto = 'Chocolate';

Update Vendedor
 set salário_fixo = (salário_fixo * 1.27) + 100;

SQL - Excluindo Registros

Apagando tuplas (linhas) da Tabela

Formato: DELETE FROM <nome da tabela>
where <condição>;

```
DELETE FROM Vendedor  
where faixa_comissão IS NULL;
```

SQL - Seleccionando Registros

Formato: Select <lista de atributos>
From <lista de relações>;
Where <lista de predicados)

Cláusula Select: Operação de Projeção da Álgebra. Contém a lista de atributos desejados no resultado da consulta.

Cláusula From: Operação de Produto Cartesiano da Álgebra. Contém a lista das relações que serão utilizadas na consulta.

Cláusula Where: Operação de Seleção da Álgebra. Contém a lista de predicados.

SQL - Selecionando Registros

Selecionando Colunas Específicas da Tabela:

Select descrição_produto, unidade, val_unit *from* Produto;

Select nome_cliente, endereço, CGC *from* Cliente;

Selecionando todas as Colunas da Tabela:

Formato: *Select * from* <tabela>;

*Select * from* Vendedor;



SQL - Selecionando Registros

Formato: Select <nome das colunas> from <tabela>
Where <restrições>;

Comparações na Cláusula WHERE:

Where <nome da coluna> <operador> <valor>

Observações:

- Quando a coluna é do tipo caractere, o <valor> deve estar entre aspas ('). Ex.: 'PARAFUSO'.

Observações - Comando Select:

- Na cláusula *Select* devem ser especificadas as colunas (ou campos) que deverão aparecer no resultado da consulta.
- Caso se deseje todos os campos da tabela no resultado de uma consulta, então um * pode ser utilizado após a palavra-chave *Select*.
- O resultado da execução de uma ou consulta (ou query) é uma tabela “virtual” (não básica - ou seja, não original no banco de dados).



SQL - Operadores Relacionais

=	Igual
< >	Diferente
<	Menor do que
>	Maior do que
<=	Menor ou igual do que
>=	Maior ou igual do que

```
Select num_pedido, código_produto, quantidade  
from item_do_pedido  
where quantidade = 35;
```

```
Select nome_cliente from cliente  
where cidade = 'Niterói';
```

SQL - Operadores Lógicos

AND “e” lógico
OR “ou” lógico”
NOT negação

```
Select descrição_produto    from produto  
      where unidade = 'M' AND val_unit = 1.05;
```

```
Select nome_cliente, endereço  
      from cliente  
      where (CEP >= '30077000' AND CEP <=  
          '30079000')  
          OR cidade = 'São Paulo';
```

```
Select num_pedido        from pedido  
      where NOT (prazo_entrega = 15);
```

SQL - Operador Between

Where <nome coluna> **BETWEEN** <valor1> AND <valor2>

Where <nome coluna> **NOT BETWEEN** <valor1> AND <valor2>

```
Select código_produto, descrição_produto  
from produto  
where val_unit between 0.32 and 2.00;
```

Observação: Inclusive os limites inferior e superior.

SQL - Operador Is NULL

Where <nome da coluna> IS NULL

Where <nome da coluna> IS NOT NULL

```
Select *  
  from cliente  
  where IE IS NULL
```



SQL - Ordenando Dados Seleccionados

- ▶ Quando se realiza uma seleção, os dados recuperados não estão ordenados.
- ▶ A SQL prevê a cláusula ORDER BY para realizar a ordenação dos dados seleccionados.

Formato: Select <nome das colunas>

from <tabela>

Where <condições>

Order by <nome das colunas>

OU:

Order by <número das colunas>

ASC ou DESC



SQL - Ordenando Dados Seleccionados

- ▶ A informação <número das colunas> se refere à posição relativa das colunas quando for apresentado o resultado da consulta, e não à posição na tabela original, contada da esquerda para a direita.
- ▶ As palavras **ASC** e **DESC** significam respectivamente, *ascendente* e *descendente*. A forma ascendente é assumida como padrão (default).



SQL - Ordenando Dados Selecionados

```
Select nome_vendedor, salario_fixo  
from vendedor  
order by nome_vendedor;
```

```
Select nome_cliente, cidade, UF  
from cliente  
order by UF DESC, cidade DESC;
```

```
Select descrição_produto, val_unit  
from produto  
Where unidade = 'M'  
order by 2 ASC;
```



Funções Embutidas da SQL

- ▶ Também chamadas funções de agregação ou de grupo.
- ▶ São elas:
 - ▶ **COUNT** - conta o número de registros ou número de valores em uma coluna
 - ▶ **SUM** - somatório de valores de um campo numérico
 - ▶ **MAX** - valor máximo em uma determinada coluna
 - ▶ **MIN** - valor mínimo em uma determinada coluna
 - ▶ **AVG** - média dos valores de um campo numérico



SQL - Utilizando Funções

▶ *Buscando Máximos e Mínimos (MAX, MIN)*

```
Select  MIN (salário_fixo), MAX (salário_fixo)
        from vendedor;
```

▶ *Totalizando Colunas (SUM)*

```
Select  SUM (quantidade)
        from item_do_pedido
        Where código_produto = '78';
```



SQL - Utilizando Funções

▶ *Calculando Médias (AVG)*

```
Select  AVG (salário_fixo)
        from vendedor;
```

▶ *Contando as Linhas (COUNT)*

```
Select  COUNT(*)
        from vendedor
        Where salário_fixo > 2500;
```



SQL - Utilizando Funções: COUNT

- ▶ *Única função que usa * como parâmetro;*
- ▶ *Conta o número de linhas na tabela (cardinalidade);*

Seja: empregado <matric, nome, depto, salario>
dependente <matric_E, nome_dep>

```
Select  COUNT (distinct Salario)
        from empregado; /* número de diferentes salarios*/
```

```
Select  nome
        from empregado
        Where (Select COUNT(*)
              from dependente
              Where matric = matric_E) >= 2; /*mais de 2 dependentes*/
```


SQL - Utilizando Cláusula Distinct

- ▶ *Problema:* Quais as unidades de produtos, diferentes, existem na tabela produto ?

```
Select  DISTINCT unidade,  
        from produto;
```

- ▶ A tabela Produto pode possuir vários produtos com a mesma unidade de medida (L, M, Kg ...). A SQL não retira automaticamente estas repetições. Para sabermos quais unidades existem, sem repetições, usamos a cláusula **DISTINCT**.



SQL - Recuperando Dados de Várias Tabelas

- ▶ *Problema:* Juntar a tabela CLIENTE com PEDIDO.

```
Select nome_cliente, pedido.código_cliente, num_pedido  
from cliente, pedido;
```

- ▶ Como podemos ver, o **qualificador de nome** consiste no nome da tabela seguido de um ponto e o nome da coluna na tabela, por exemplo:
 - ▶ Coluna DESCRIÇÃO da tabela PRODUTO
PRODUTO.descrição_produto
 - ▶ Os qualificadores de nomes são utilizados para efetivar o JOIN

SQL - Recuperando Dados de Várias Tabelas

- ▶ **Problema:** Que clientes fizeram os pedidos ?

```
Select nome_cliente, pedido.código_cliente, num_pedido
      from cliente, pedido
      Where cliente.código_cliente = pedido.código_cliente;
```

- ▶ Podemos utilizar as cláusulas LIKE, NOT LIKE, IN, NOT IN, NULL, NOT NULL e misturá-las com os operadores AND, OR e NOT, dentro de uma cláusula WHERE na junção entre tabelas.

SQL - Recuperando Dados de Várias Tabelas

- ▶ **Problema:** Quais clientes que têm prazo de entrega superior a 15 dias e que pertencem aos estados de São Paulo (SP) ou Rio de Janeiro (RJ) ?

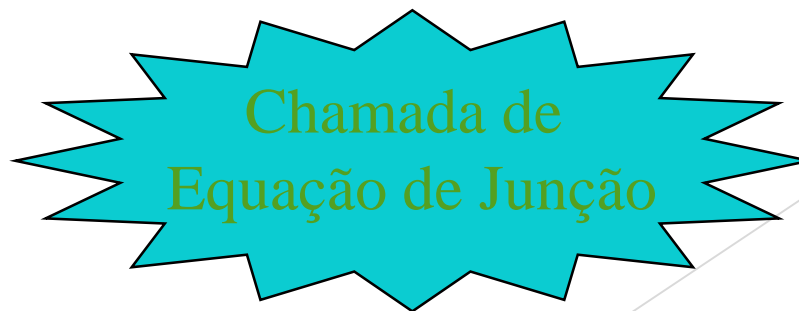
```
Select nome_cliente, UF, prazo_entrega
from cliente, pedido
Where UF IN ('SP', 'RJ') AND
prazo_entrega > 15 AND
cliente.cod_cliente = pedido.cod_cliente;
```



SQL - Recuperando Dados de Várias Tabelas

- **Problema:** Mostrar os clientes e seus respectivos prazos de entrega, ordenados do maior para o menor.

```
Select nome_cliente, prazo_entrega  
from cliente, pedido  
Where cliente.código_cliente = pedido.código_cliente  
Order by prazo_entrega desc;
```



SQL - Tipos de Junção

Seja: **empregado** <matric, noem depto, salario, ger>
departamento <numd, nomed>

- ▶ Padrão: INNER JOIN (junção interna)
 - ▶ Cláusula de junção no where

Ex1.: Empregados e seus gerentes

```
Select E.nome As nome_emp, G.nome As nome_ger  
from Empregado E, Empregado G  
Where E.ger = G.matric
```



SQL - Tipos de Junção

▶ NATURAL JOIN

- ▶ Nenhum predicado de junção é especificado;
- ▶ Uma condição implícita é criada, com os atributos de mesmo nome.

Ex2.: Nome e salário dos empregados do departamento de informática

```
Select nome, salario
```

```
from Empregado JOIN Departamento ON depto = numd
```

```
Where nomed = 'informatica'
```

```
Select nome, salario
```

```
from Empregado NATURAL JOIN
```

```
(Departamento as D (nomed, depto))
```

```
Where nomed = 'informatica'
```



SQL - Outros Exemplos

► Combinar Resultados de Pesquisas (UNION):

► **Problema:** Listar os nomes e códigos dos vendedores que têm salário fixo > R\$ 1.000,00 e clientes que residem no Rio.

```
Select código = código_cliente, nome = nome_cliente  
from cliente  
where UF = 'RJ'
```

UNION

```
Select código_vendedor, nome_vendedor  
from vendedor  
where salário_fixo > 1000;
```



SQL - Outros Exemplos

► Fazer um JOIN da tabela com ela mesma:

► **Problema:** Para todos os vendedores, listar os nomes e salários dos vendedores com salários fixo menores que os de outros vendedores .

```
Select nome_vendedor = V1.nome_vendedor,  
       salario_menor = V1.salario_fixo,  
       nome_maior = V2.nome_vendedor,  
       salario_maior = V2.salario_fixo,  
from Vendedor V1, Vendedor V2  
Where V1.salario_fixo < V2.salario_fixo  
order by 1;
```



Exercícios

- ▶ Selecione todos os campos da tabela projeto
- ▶ Selecione o código do projeto e as horas de cada projeto
- ▶ Liste o código de todos os empregados alocados em um projeto

EMP

EMPID	ENOME
E101	Sílvio
E105	João
E110	Adalto
E115	Sandra

PROJETO

PROJ#	PROJNOME	ORÇAMENTO
P10	HUDSON	500000
P15	COLUMBIA	350000
P20	PROSPECTUS	350000
P23	ANACONDA	600000

ALOCAÇÃO

EMPID	PROJ#	HORAS
E101	P10	200
E101	P15	300
E105	P10	400
E110	P10	250
E110	P20	350
E110	P15	700
E115	P10	300

Exercícios

- ▶ Liste o nome dos projetos com orçamento maior que 400000
- ▶ Liste os códigos dos empregados que trabalham no projeto P10
- ▶ Liste os códigos dos empregados que trabalham no projeto P10 com mais de 200 Horas
- ▶ Qual o nome do funcionário E105

EMP	
EMPID	ENOME
E101	Sílvio
E105	João
E110	Adalto
E115	Sandra

PROJETO		
PROJ#	PROJNOME	ORÇAMENTO
P10	HUDSON	500000
P15	COLUMBIA	350000
P20	PROSPECTUS	350000
P23	ANACONDA	600000

ALOCAÇÃO		
EMPID	PROJ#	HORAS
E101	P10	200
E101	P15	300
E105	P10	400
E110	P10	250
E110	P20	350
E110	P15	700
E115	P10	300



That's all Folks!



nemo