BANCO DE DADOS
SQL – Prof. Gale

O nome SQL significa "Structure Query Language" – Linguagem estruturada de pesquisa. É uma linguagem estruturada para manipulação de dados. É padronizada para os bancos de dados relacionais, mas cada gerenciador pode assumir uma extensão própria dessa linguagem.

Vantagens e Desvantagens da linguagem SQL:

- Independência de fabricante: é oferecido a todos os sistemas gerenciadores de banco de dados
- Portabilidade entre computadores: pode ser usada em um computador pessoal, passando a uma estação de trabalho, até um computador de grande porte.
- Inglês estruturado de alto nível: é formada por um conjunto bem simples de sentenças em inglês
- Consulta interativa: A SQL provê um acesso rápido aos dados, fornecendo respostas ao usuário, a questões complexas
- Definição dinâmica dos dados: pode-se alterar, expandir ou incluir, as estruturas dos dados armazenados

Desvantagem:

- Como é feita uma padronização, ocorre assim uma inibição da criatividade, pois quem desenvolve aplicações fica preso a soluções padronizadas.

Criação e Distribuição de Tabelas:

CREATE TABLE

Segue a seguinte forma:

Create table <tabela>
(<descrição das colunas>);
(<descrição das chaves>);

onde:

<tabela>: é o nome da nova tabela a ser criada

<descrição das colunas> : é uma lista de colunas (campos) e seus respectivos tipos de dados. (smallint, char, money, varchar, integer, decimal, float, real, date, time, logical)

<descrição das chaves> : é a lista de colunas que são tratadas como chave estrangeira

Alguns campos podem receber o valor NULL (nulo) e o campo definido como chave primária, além de não poder receber NULL, deve ser um campo UNIQUE (sem repetições – chave primária).

Exemplo: Figura 1

```
Create Table cliente
    (código_cliente smallint not nul unique,
     nome_cliente char(20),
    endereço char(30),
    cidade char(15),
    CEP char(8),
    UF char(2),
    CGC char(20),
    IE char(20) );
```

```
Create Table Pedido
(num_pedido int not null unique,
 prazo_entrega smallint not null,
código_cliente smallint not null,
código_vendedor smallint not null,
FOREIGN KEY (código_cliente)
                REFERENCES CLIENTE) ;
```

Para eliminar uma tabela criada é utilizado o comando DROP:

Drop table <tabela>;

Extraindo Dados de uma Tabela: Comando SELECT:

Forma:
Select <nome(s) da(s) coluna(s)> from <tabela>;

Exemplo: listar todos os produtos com respectivas descrições, unidades e valores unitários? (Diagrama 1)

select descrição, unidade, valor_unitário from produto;

(Diagrama 2)

- Selecionando todas as colunas da tabela:

Select *
From <tabela>;

- Selecionando somente alguns registros da tabela:

Select <nome(s) da(s) coluna(s)>
From <tabela>
Where <restrições>;

Onde: where <nome da coluna> <operador> <valor>

OBS1: quando a coluna é do tipo caractere, o <valor> deve estar entre apóstrofes
OBS2: Em SQL existe diferenciação entre maiúscula e minúscula

Exemplo: Listar o num_pedido, o código_produto e a quantidade dos itens do pedido com a quantidade igual a 35.

Select num_pedido, código_produto, quantidade
From item_pedido
Where quantidade = 35

Exemplo2: Quais os clientes moram em Niterói?

Operadores Lógicos:

- And
- Or
- Not

Exemplo: listar os produtos que tenham unidade igual a ‘M’ e valor unitário igual a R$ 1,05

Select descricão_produto
From produto
Where unidade = ‘M’ and val_unit = 1.05;

Exemplo: Liste os clientes e seus respectivos endereços, que moram em ‘São Paulo’ou estejam na faixa de CEP entre ‘30077000’ e ‘30079000’.

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

Obs: A utilização de parênteses é fundamental para a construção correta da frase, pois sem eles as consultas podem ser analisadas de forma errada, devido a prioridade do operador AND ser maior que a prioridade do operador OR.

Exemplo: Mostrar todos os pedidos que não tenham prazo de entrega igual a 15 dias

```
Select num_pedido
        From pedido
        Where NOT (prazo_entrega = 15);
```

- Operadores Between e NOT Between

  - where <nome da coluna> BETWEEN <valor1> AND <valor2>
  - where <nome da coluna> NOT BETWEEN <valor1> AND <valor2>

  Este operador propicia a pesquisa por uma determinada coluna e que esteja dentro de uma faixa de valores, sem a necessidade dos operadores >=, <= e AND. Tanto o valor1 quanto o valor2 tem que ser do mesmo tipo de dado da coluna.

  Exemplo: Listar o código e a descrição dos produtos que tenham o valor unitário na faixa de R$ 0,32 até R$ 2,00.

```
Select codigo_produto, descricão
        From produto
        Where valor between 0.32 and 2.00;
```

- Operadores LIKE e NOT LIKE

  - Where <nome da coluna> LIKE <valor>;

Os operadores LIKE e NOT LIKE só trabalham sobre colunas que sejam do tipo CHAR. Eles têm praticamente o mesmo funcionamento que os operadores = e <>, porém o poder desses operadores está na utilização dos símbolos (%) e (_) que podem fazer o papel de coringa:

- % - substitui uma palavra
- _ - substitui um caractere

Ex: 'Lápis %' pode enxergar os seguintes registros:

- Lápis Preto
- Lápis Cera
- Lápis Borracha

Exemplo: Listar os produtos que tenham a sua unidade começando por K:

```
Select codigo, descrição
From produto
Where unidade LIKE 'K_';
```

Exemplo2: Listar os vendedores que não começam por 'Jo'

```
Select codigo, nome_vendedor
From vendedor
Where nome_vendedor NOT LIKE 'Jo%';
```

- Operadores IN e NOT IN

  - Where <nome da coluna> IN <valores>;
  - Where <nome da coluna> NOT IN <valores>;

Estes operadores pesquisam registros que estão ou não contidos no conjunto de <valores> fornecido.

Exemplo: Listar os vendedores que são da faixa de comissão A e B

```
Select nome_vendedor
From vendedor
Where faixa_comissão IN ('A', 'B');
```

- Operadores IS NULL e IS NOT NULL
  - Where <nome da coluna> IS NULL;
  - Where <nome da coluna> IS NOT NULL;

A utilização do valor nulo (NULL) é muito problemática, pois cada implementação da linguagem pode adotar qualquer representação para o valor nulo.

Exemplo: mostrar os clientes que não tenham inscrição estadual

```
Select *
From cliente
Where IE IS NULL;
```

- Ordenando os dados Selecionados

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

Forma:

```
Select < nome da (s) coluna(s)>
From <tabela>
<where < condição(ões)> >
order by <nome da(s) coluna(s)>          | ASC
                ou                       |
order by <número da(s) coluna(s)>        | DESC
```

Obs: o default é ASC

Exemplo: Mostrar em ordem alfabética a lista de vendedores e seus respectivos salários fixos

```
Select nome_vendedor, salário_fixo
From vendedor
Order by nome_vendedor;
```

Exemplo2: listar os nomes, cidades e estados de todos os clientes, ordenados por estado e cidade de forma descendente.

```
Select nome_cliente, cidade, UF
```

```
From cliente
Order by UF DESC, cidade DESC;
```

Exemplo3: mostrar a descrição e o valor unitário de todos os produtos que tenham a unidade ‘KG’, em ordem de valor unitário ascendente.

```
Select descrição, val_unit
From produto
Where unidade = ‘KG’
Order by val_unit ASC;
```

- Realizando Cálculos com informação Selecionada

Com a SQL pode-se criar um campo que não pertença à tabela original, e que seja fruto do cálculo sobre alguns campos da tabela.

Exemplo: Mostrar o novo salário fixo dos vendedores, de faixa de comissão ‘C’, calculado com base no reajuste de 75% acrescido de R$ 120,00 de bonificação. Ordenar pelo nome do vendedor.

```
Select nome_vendedor,
    Novo_salário = (salário_fixo * 1.75) + 120
    From vendedor
    Where faixa_comissão = ‘C’
    Order by nome_vendedor;
```

- Utilizando funções sobre conjuntos
  - Buscando máximos e mínimos (MAX, MIN)

Exemplo: Mostrar o menor e o maior salário de vendedor.

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

- Totalizando colunas (SUM)

Exemplo: Mostrar a quantidade total pedida para o produto ‘VINHO’ de código ‘78’

```
Select SUM(quantidade),
From pedido
Where código = ‘78’;
```

- Calculando Médias: (AVG)

Exemplo: Qual a média dos salários fixos dos vendedores?

```
Select AVG(salário_fixo),
From vendedor;
```

- Contando os Registros (COUNT)

Exemplo: Quantos vendedores ganham acima de R$ 2.500,00 de salário fixo?

```
Select COUNT (*),
From vendedor
Where salario_fixo > 2500;
```

- Utilizando a cláusula DISTINCT

  Normalmente, vários registros dentro de uma tabela podem conter os mesmos valores, com exceção da chave primária. Com isso muitas consultas podem trazer informações erradas.

Exemplo: quais as unidades de produtos, diferentes, na tabela produto?

```
Select DISTINCT unidade,
From produto;
```

- Agrupando informações selecionadas (GROUP BY)

  Utilizando a cláusula GROUP BY, é possível organizar a seleção de dados em grupos determinados.

  - Forma:

```
Select  <nome da(s) coluna(s)>
From <tabela>
<where condição(ões)>
group by <nome da(s) coluna(s)>;
order by <nome da(s) coluna(s)>;
```

Exemplo: Listar o número de produtos que cada pedido contém.

```
Select num_pedido, total_produtos = COUNT (*)
From item_pedido
Group by num_pedido
```

- Agrupando de forma condicional (HAVING)

Exemplo: Listar os pedidos que tem mais do que 3 produtos.

```
Select num_pedido, total_produtos = COUNT(*)
From item_pedido
Group by num_pedido;
Having COUNT(*) > 3;
```

Ramificações do Comando Select

SELECT *

Sintaxe: SELECT *
FROM *table_name*

ESCOLHENDO COLUNAS

```
SELECT column_name[,column_name...]
 FROM table_name
```

```
SELECT au_id, au_fname, au_lname
  FROM authors
```

USANDO LETRAS

SELECT *column_name*|*'string literal'[,column_name'string_literal'*...]
FROM *table_name*

```
SELECT au_fname, au_name, 'Identification number:', au_id
  FROM authors
```

SELECT *column_heading=column_name[,column_name...]*
FROM *table_name*
ou
SELECT *column_name column_heading*[,*column_name*...]
FROM *table_name*

```
SELECT FIRST = au_fname, LAST = au_lname, IDENTIFICATIO# =
'Identification number:', Author_ID = au_id
FROM authors
```

OPERADORES ARITMÉTICOS

Operação	tipos de dados que podem usar esta operação
+	int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
-	int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
/	int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
*	int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
%	int, smallint e tinyint

Sintaxe

{ constant | column_name | function | (subquery) }
[{ arithmetic_operator | bitwise_operator | string_operator }
{ constant | column_name | function | (subquery) }...]

```
SELECT price, (price * 1.1), title
  FROM titles
```

MANIPULAÇÃO DE DADOS NUMÉRICOS

Função	Parâmetros
ABS	(numeric_expr)
ACOS,ASIN,ATAN,ATN2	(float_expr)
COS,SIN,COT,TAN	(float_expr)
CEILING	(numeric_expr)
DEGREES	(numeric_expr)
EXP	(float_expr)
FLOOR	(numeric_expr)
LOG	(float_expr)
LOG10	(float_expr)
PI	()

Função	Parâmetros
POWER	(numeric_expr,y)
RADIANS	(numeric_expr)
RAND	([seed])
ROUND	(numeric_expr,length)
SIGN	(numeric_expr)
SQRT	(float_expr)

```
SELECT title_id,
          ROUND(price*royalty/100,0)
          FROM titles
```

1 MANIPULANDO CARACTERES DE DADOS

Função	Parâmetros
+	(expression expression)
ASCII	(char_expr)
CHAR	(integer_expr)
CHARINDEX	(‘pattern’, expression)
DIFFERENCE	(char_expr1,char_expr2)
LOWER	(char_expr)
LTRIM	(char_expr)
PATINDEX	(‘%patern%’, expression)
REPLICATE	(char_expr, integer_expr)
REVERSE	(char_expr)
RIGHT	(char_expr,integer_expr)
RTRIM	(char_expr)
SOUNDEX	(char_expr)
SPACE	(integer_expr)
STR	(float_expr[,lenght[,decimal]])
STUFF	(char_expr1, start, lenght, char_expr2)
SUBSTRING	(expression, start, lenght)
UPPER	(char_expr)

```
SELECT au_lname + ', ' +
          Substring (au_fname,1,1) + '.',
          au_id
          FROM authors
```

MANIPULANDO DADOS DE DATA E TEMPO

FUNÇÃO	PARAMETROS
DATEADD	(datepart, number, date)
DATEDIFF	(datepart, date1, date2)
DATENAME	(datepart, date)
DATEPART	(datepart, date)
GETDATE	()

Tipos de data	Abreviações	Valores aceitos
year	yy	1752-9999
quarter	qq	1-4
mont	mm	1-12
day of year	dy	1-366
day	dd	1-31
week	wk	0-51
weekday	dw	1-7 (1 é domingo)
hour	hh	0-23
minute	mi	0-59
second	ss	0-59
millisecond	ms	0-999

```
SELECT
          DATEDIFF (MONTH, pubdate, GETDATE())
          FROM Titles
```

FUNÇÕES DE SISTEMA

FUNÇÃO	PARÂMETROS
COALESCE	(expression1,expression2,...expressionN)
COL_NAME	(‘table_id’, column_id)
COL_LENGHT	(‘table_name’, ’column_name’)
DATALENGHT	(‘expression’)
DB_ID	([‘databasename’])
DB_NAME	([database_id])
GETANSINULL	([‘databasename’])
HOST_ID	()
HOST_NAME	()
IDENT_INCR	(‘table_name’)
IDENT_SEED	(‘table_name’)
INDEX_COL	(‘table_name’, index_id, key_id’)
ISNULL	(expression, value)
NULLIF	(expression1,expression2)
OBJECT_ID	(‘object_name’)
OBJECT_NAME	(oblect_id)
STATS_DATE	(table_id,index_id)
SUSER_ID	([‘server_user_id’])
SUSER_NAME	([server_user_id])
USER_ID	([‘username’])
USER_NAME	([user_id])

```
SELECT length = DATALENGTH(pub_name), pub_name
   FROM publishers
```

Resultado:

```
length    pub_name
      14        New Moon Books
      16        Binnet & Hardley
      20        Algodata Infosystems
      21        Five Lakes Publishing
      (4 row(s) affected)
```

CONVERSÃO DE DADOS

CONVERT(datatype[(length)],expression[,style])

COM SEC.	SEC.	STANDARD	FORMATO DE SAIDA DOS DADOS
1	101	USA	mm/dd/yy
2	102	ANSI	yy.mm.dd
3	103	britânico	dd/mm/yy
10	110	USA	mm-dd-yy
12	112	ISO	yymmdd

```
SELECT 'Title Code' = pub_id +
   UPPER(SUBSTRING(type,1,3)) +
   SUBSTRING(CONVERT(CHAR(4),DATEPART(YY,pubdate)),3,3)
FROM titles
```

Resultado:

```
Title Code
     1389BUS91
     0736BUS91
     1389BUS91
     .
     .
     .
     (18 row(s) affected)
```

Bom Estudo!!!

Dúvidas: galesandro.capovilla@sj.unisal.br