

# Curso: SQL Básico

*Glauco Luiz Rezende de Carvalho  
glauco.rezende@banese.com.br*

*Este material foi baseado e adaptado pelo material do Prof. MSc.  
Fábio Soares Silva da Universidade Tiradentes - UNIT*

## SQL

- Padrão de linguagem de consulta relacional aceito pelos comitês ANSI e ISO.
  
- **Histórico**
  - Criada no início da década 70 pela IBM.
  - SQL-86 ou SQL1.
  - SQL-92 ou SQL2.
  - SQL-99 ou SQL3.
  
- **Características**
  - Linguagem declarativa.
  - DML e DDL.

# SQL

## ■ DML

- SELECT
- INSERT
- UPDATE
- DELETE

## ■ DDL

- CREATE TABLE
- ALTER TABLE
- DROP TABLE
- CREATE VIEW
- DROP VIEW
- CREATE INDEX
- DROP INDEX

# DML

***LINGUAGEM DE  
MANIPULAÇÃO DE DADOS***

## Comando SELECT

### ■ Estrutura Básica

**SELECT** <cláusula-projeção>

**FROM** <origem-dados>

**WHERE** <condição-seleção>

**GROUP BY** <critério-grupo>

**HAVING** <condição-grupo>

**ORDER BY** <critério-ordenamento>

## Comando SELECT

### ■ Cláusula de projeção

- Lista de elementos separados por vírgula.
- Os elementos de projeção podem ser: colunas, constantes e expressão.
- Cada elemento projetado pode ter um apelido associado.
- Os apelidos são indicação pela especificação explícita da cláusula **AS**, ou através da simples colocação do apelido entre apóstrofo ( ' ' ) após o elemento projetado.

## Exemplo SQL

- Listar para tabela jogadores exemplos de projeção de constantes, colunas e expressões.

```
select 'Exemplo de Constante',  
        nom_jog,  
        'Expressao: ' +  
        convert(varchar(20), salario * 1.5)  
from jogadores;
```

## Comando SELECT

### ■ Cláusula **FROM**

- Determina a tabela onde os dados serão obtidos na resolução da consulta.
- Juntamente com a cláusula **SELECT**, a cláusula **FROM** define a especificação mínima para um comando de seleção.
- Na sua formação mais simples, o comando de seleção implementa apenas a operação de projeção, recuperando assim todas as linhas da tabela.

## Exemplo SQL

- Listar o código, o nome e a data de nascimento de todos os jogadores. Os apelidado para as colunas devem ser código, nome e data, respectivamente.

## Comando SELECT

### ■ Cláusula WHERE

- Define uma expressão condicional cujo resultado da avaliação irá determinar quais as linhas da tabela que serão recuperadas.
- Implementa a operação de seleção da álgebra relacional.
- A condição é avaliada para cada linha da tabela
  - Se o resultado for verdadeiro, a linha é retornada pelo comando.
  - Caso contrário a linha é descartada da composição do resultado

## Exemplo SQL

- Obter o código, o nome e o salário de todos os jogadores do time 1 que ganham mais que R\$ 30.000.

## Comando SELECT

### ■ Considerações

- O identificador **DISTINCT** pode ser utilizado na projeção para retirar linhas repetidas do resultado.
- Na especificação da condição de seleção podem ser utilizadas funções para composição de expressões que criem filtros sobre os dados recuperados.
- Os comandos SQL não são sensíveis a letras maiúsculas e minúsculas.

## Exemplo SQL

- Obter o nome e o código do time de todos os jogadores.

## Comando SELECT

### ■ Funções de agregação

#### □ **Count**( [distinct] coluna )

- Recupera o número de elementos de um determinado grupo. Quando especificado com o identificador **DISTINCT**, efetua a contagem dos valores diferentes existentes na coluna.
- O asterisco ( \* ) pode ser utilizado como argumento da função para indicar a contagem do número de linhas.

#### □ **Sum**( [distinct] coluna )

- Calcula o somatório dos valores de uma coluna.

## Comando SELECT

### ■ Funções de agregação

- **Avg**( [distinct] coluna )
  - Obtém a média entre os valores de uma coluna.
  
- **Min**( coluna )
  - Obtém o menor valor armazenado em uma coluna.
  
- **Max**( coluna )
  - Obtém o maior valor armazenado em uma coluna.
  
- Os valores nulos não são considerados para o cálculo do resultado da função.

## Exemplo SQL

- Listar o total de linhas, o total de times diferentes e o maior salário existente na tabela jogadores.



## Comando SELECT

### ■ Cláusula **GROUP BY**

- A cláusula **GROUP BY** permite que os dados consultados sejam agrupados segundo um determinado critério.
- Sobre os grupos formados podem ser extraídas informações sumarizadas.
- O critério de agrupamento (itens de grupo) é definido a partir de uma lista de elementos separados por vírgula.

## Comando SELECT

### ■ Cláusula **GROUP BY**

- Cada item de grupo pode ser uma coluna ou uma expressão.
- Em uma consulta agrupada, só podem ser projetados itens de grupo (elementos da lista da cláusula **GROUP BY**) ou uma das funções de agregação count, sum, avg, max e min.
- Um item de grupo não necessariamente precisa ser projetado.

## Exemplo SQL

- Recuperar para cada time, a quantidade de jogadores, o menor salário, o maior salário e a média salarial.

## Exemplo SQL

cod_jog	cod_time	nom_jog	salário
1908	01	José da Silva	35.000,00
2096	02	Luis Carlos	25.000,00
8578	03	Eduardo Souza	47.500,00
2536	02	João Luis	88.000,00
4598	01	Marcelo Santos	42.000,00



cod_time	count(*)	min(salario)	max(salario)	avg(salario)
01	02	35.000,00	42.000,00	38.500,00
02	02	25.000,00	88.000,00	56.500,00
03	01	47.500,00	47.500,00	47.500,00

## Comando SELECT

### ■ Considerações sobre consultas agrupadas

- A consulta agrupada pode ser combinada com a cláusula **WHERE**, sendo que os dados só são agrupados depois de filtrados.
- Quando o critério de agrupamento possuir mais de um item de grupo, a classificação dos dados nos vários grupos levará em consideração a combinação dos valores das colunas que compõem a cláusula **GROUP BY**.

## Exemplo SQL

- Listar quanto cada patrocinador investiu em cada ano a partir do ano 2000. O resultado deve conter o código do patrocinador, o ano do patrocínio e o total investido.

## Comando SELECT

### ■ Cláusula **HAVING**

- Funciona de modo similar à cláusula **WHERE**, sendo que o filtro é aplicado aos grupos formados pela cláusula **GROUP BY**.
- A cláusula **HAVING** só pode ser definida combinada com a cláusula **GROUP BY**.
- Somente itens de grupo e funções de agregação podem ser utilizadas nas expressões condicionais da cláusula **HAVING**.
- As funções de agregação utilizadas na cláusula **HAVING** não precisam ser necessariamente mesmas utilizadas na projeção.

## Exemplo SQL

- Selecionar o código, o total de atletas, o maior salário e a média salarial daqueles times que possuam salários maiores do que R\$ 30.000.

## Exemplo SQL

- Listar o código, o ano do patrocínio e o total investido daqueles patrocinadores que investiram mais de R\$ 6.000.000,00 em um mesmo ano a partir de 2001.

## Comando SELECT

### ■ Cláusula **ORDER BY**

- Um resultado pode ser ordenado por qualquer coluna ou elemento projetado.
- A especificação da ordem é feita por uma lista de elementos (colunas ou expressões) separados por vírgula.
- Para o caso de consultas agrupadas, somente itens de grupo ou funções de agregação podem ser utilizadas para ordenamento.

## Comando SELECT

### ■ Cláusula ORDER BY

- Alguns SGBDs permite que a cláusula de ordenação possa ser indicada por números que representam os elementos projetados no **SELECT**.
- O número 1 representa o primeiro elemento projetado, o 2 o segundo, o 3 o terceiro, e o número n o n-ésimo.
- É possível obter a ordem descendente para um dado elemento de ordenação especificando o identificador **DESC** após o elemento.

## Exemplo SQL

- Listar os dados dos jogadores das posições de código 1, 2 e 3 que ganham mais que R\$ 30.000. O resultado deve ser ordenado ascendentemente pelo código do time, e para um mesmo time, a ordem dos nomes deve ser decrescente.

## Exemplo SQL

- Listar os dados dos times que pagam um salário inferior a R\$ 20.000. O resultado deve estar organizado em ordem decrescente de média salarial e maior salário.

## Comando SELECT

### ■ Produto cartesiano

- A cláusula **FROM** admite que seja especificada uma lista de tabelas separadas por vírgula. Se nenhuma condição de junção for definida na cláusula **WHERE**, o resultado da seleção é o produto cartesiano entre as tabelas.
- Para resolver conflitos de nomes de colunas, é permitida a definição de apelidos para as tabelas.
- Os apelidos são definidos por identificadores colocados após os nomes das tabelas.

## Exemplo SQL

- Listar a combinação de todos os times com todos os jogadores ordenando pelo nome do time e nome do jogador.

## Comando SELECT

### ■ Junção natural

- Uma forma comum e genérica para definição de junção natural é utilizar as operações de seleção e produto cartesiano combinadas.
- A junção natural é realizada especificando na cláusula **WHERE** a condição de junção.
- A condição de junção é implementada no SQL igualando a chave estrangeira de uma tabela com a respectiva chave primária da outra.
- Em junções que envolvam mais de duas tabelas, deve existir pelo menos uma condição de junção para cada uma das tabelas.



# Comando SELECT

## ■ Junção natural

- As várias condições de junção devem formar uma "corrente de junções" entre todas as tabelas da cláusula **FROM**.
- Se uma das junções não for especificada, é realizado o produto cartesiano entre as tabelas não ligadas.



## Exemplo SQL

- Listar o código, nome, o código do time e o nome do time de todos os jogadores cadastrados. O resultado deve ser ordenando por nome do time e nome do jogador.

## Exemplo SQL

- Listar os patrocinadores de cada time no ano de 2001. O resultado deve conter ordenadamente o nome do time, o nome do patrocinador e o valor patrocinado.

## Comando SELECT

### ■ Junção natural

- Outra forma de implementar a junção natural é utilizar o operador **JOIN** na cláusula **FROM** para indicar as tabelas que irão fazer parte da junção.
- Na cláusula **JOIN** a condição de junção é especificada na cláusula **ON**.
- Sintaxe:
  - Tabela-1 **JOIN** Tabela-2 **ON** <condição-junção>

## Exemplo SQL

- Listar o nome, o nome do time e a posição de todos os jogadores. O resultado deve ser ordenado por nome do time e nome do jogador.

## Comando SELECT

### ■ Junção natural

- A existência da junção natural em um determinado comando não inviabiliza a utilização das demais cláusulas do comando **SELECT**.
- É importante ressaltar que em termos lógicos, as cláusulas **WHERE** e **FROM**, e portanto a junção natural, são avaliadas antes das outras cláusulas.
- Assim, as cláusulas **GROUP BY** e **ORDER BY** são aplicadas sobre o resultado da junção.

## Exemplo SQL

- Listar a média salarial de cada time. O resultado deve conter além da média salarial, o código e o nome do time ordenando pelo nome do time.

## Comando SELECT

### ■ Outer join ( junção externa )

- A junção natural (inner join) possui como característica básica a exclusão de linhas das tabela que não tenham respectiva referência.
- O outer join recupera linhas da tabela mesmo que não haja combinação de valores.
- Existem quatro tipos de junção externa:
  - LEFT JOIN
  - RIGHT JOIN
  - FULL JOIN
  - INNER JOIN

## **Exemplo SQL – LEFT JOIN**

- Recuperar para cada time todos os seus jogadores. O resultado deve conter o código e o nome do time e o nome do jogador. Os times que não possuem jogadores devem ser recuperados ordenando pelo nome do time e nome do jogador.

## **Exemplo SQL – RIGHT JOIN**

- Listar o código e o nome dos jogadores com seus respectivos times. Jogadores sem time devem ser listados no resultado ordenando pelo nome do jogador e nome do time.

## Exemplo SQL – FULL JOIN

- Recuperar para cada time todos os seus jogadores. O resultado deve conter o código, o nome do time e o nome do jogador. Os times sem jogadores, e os jogadores sem time, devem ser listados ordenando pelo nome do time e nome do jogador.

## Exemplo SQL – INNER JOIN

- Recuperar para cada time todos os seus jogadores. O resultado deve conter o código, o nome do time e o nome do jogador, devem ser listados ordenando pelo nome do time e nome do jogador.

## Comando SELECT

### ■ Subselect

- É um recurso oferecido pelo SQL que permite a um comando obter dinamicamente dados de um outro para composição do resultado.
- O subselect pode ocorrer nas cláusulas **SELECT**, **FROM**, **WHERE** e **HAVING**.
- A única exigência para o uso de subselects é o respeito à compatibilidade entre os operadores das condições e o resultado do subselect.

## Exemplo SQL

- Listar o código e o nome do time por onde o jogador de código 2 já atuou.

## Comando SELECT

- No exemplo anterior, os times obtidos do resultado da consulta à tabela históricos são automaticamente utilizados na seleção dos dados da tabela times.
- Na comparação da cláusula **WHERE** é utilizado o operador **IN**, que é compatível com a coleção recuperada pelo subselect.
- Para operadores como o de igualdade, o resultado do subselect deve recuperar apenas um valor.

## Exemplo SQL

- Recuperar o código, o nome e o salário dos jogadores de melhor remuneração.



## Comando SELECT

### ■ Subselects com ligação

- É possível para um subselect utilizar dados oriundos do select mais externo.
- O escopo de visibilidade dos dados só permite a um subselect mais interno enxergar dados de um select mais externo.
- Neste contexto, o mecanismo lógico de execução é o seguinte: para cada linha do select mais externo, o subselect é executado utilizando os valores da linha atual do select mais externo como parâmetro.

## Exemplo SQL

- Obter o código, o nome do time, o nome do jogador e o salário dos jogadores mais bem remunerados em cada time ordenando pelo nome do time e nome do jogador.

## Comando SELECT

- No exemplo anterior, o subselect utiliza na sua condição de seleção o valor do código do time do select externo ( j.cod\_time ).
- Um algoritmo lógico de execução:  
**para** cada linha da junção **faça**  
    **obter** linha\_atual;  
    **executar** subselect **substituindo** j.cod\_time  
        **por** linha\_atual.cod\_time;  
    **obter** maior\_salario;  
    **se** linha\_atual.salário = maior\_salario **então**  
        **adicionar** linha\_atual ao resultado;

## Comando SELECT

### ■ O operador EXISTS

- É normalmente utilizado em situações de subselects onde o operador **IN** pode ser aplicado.
- Em sua lógica de execução, os elementos do select mais externo serão retornados no resultado se a execução do subselect recuperar algum valor.
- Na maioria das vezes, é utilizado em subselects com ligação.

## Exemplo SQL

- Recuperar o código e o nome do time por onde o jogador 2 já atuou.

## Comando SELECT

- Uma lógica de execução possível para o exemplo anterior.

**para** cada linha da tabela times **faça**  
    **obter** linha\_atual;  
    **executar** subselect **substituindo** t.cod\_time  
                            **por** linha\_atual.cod\_time;  
    **obter** total\_linhas do resultado do subselect;  
    **se** total\_linhas > 0 **então**  
        **adicionar** linha\_atual ao resultado;

## Comando SELECT

- O subselect na cláusula **HAVING** funciona de maneira análoga ao subselect na cláusula **WHERE**.
- A diferença está no fato que os elementos utilizados na composição da condição devem ser itens de grupo ou funções de agregação.
- As condições têm que respeitar os tipos de operadores em relação à resposta do subselect.

## Exemplo SQL

- Listar os times que possuem uma média salarial superior a média salarial de todos os jogadores.

## Exemplo SQL

- Obter os dados dos patrocinadores que em um determinado ano, investiram mais que a média anual.

## Comando SELECT

- Um subselect na cláusula **FROM** funciona como uma tabela dinâmica cujos dados são obtidos em tempo de execução do comando de seleção.
- Logicamente, o subselect da cláusula **FROM** é executado, e a tabela resultante da sua execução é utilizada na resolução da consulta.
- Não há restrição quanto à composição do subselect na cláusula **FROM**.

## Exemplo SQL

- Recuperar o código do time, o nome do jogador e o salário dos jogadores mais bem remunerados de cada time.

## Comando SELECT

- O subselect também pode ser definido como um elemento de projeção.
- A lógica destes subselects é a mesma da aplicada na cláusula **WHERE**.
- Os subselects definidos na projeção são normalmente utilizados para capturar dados de outras tabelas que devem compor o resultado.

## Exemplo SQL

- Listar o código, o nome, o salário e o nome do time de todos os jogadores que ganham mais que R\$ 30.000.

## Comando SELECT

### ■ Operador UNION

- Implementa a operação relacional de união, ou seja, recupera, sem repetição, as linhas pertencentes aos dois comandos de seleção.
- A união só é possível se os dois selects que a compõe tiverem o mesmo número de itens projetados, e os i-ésimos domínios compatíveis.
- A cláusula de ordenação é única para toda a operação de união.
- Os nomes das colunas do resultado são os mesmos do primeiro select da união.

## Exemplo SQL

- Obter os patrocinadores de 2000 e os times que participaram do campeonato 1 ordenando pelo nome do patrocinador.

## Exemplo SQL

- Encontrar os patrocinadores que jamais patrocinaram o time de código 4 ordenando pelo nome do patrocinador.
- **OBS: (subtração de conjuntos ≠ UNION)**



## Exemplo SQL

- Obter os times onde já atuaram tanto o jogador 2 quanto o jogador 3 ordenando pelo nome do time.
- **OBS: (Intercessão de conjuntos)**

## Comando INSERT

### ■ Sintaxe

```
INSERT INTO <tabela>  
VALUES( <lista_de_valores> );
```

- Nesta sintaxe, a ordem em que os valores são informados na lista de valores, deve ser a mesma em que a tabela está definida.
- O número de valores informados deve ser o mesmo do número de colunas da tabela.

## Comando INSERT

### ■ Sintaxe

```
INSERT INTO <tabela>  
VALUES( <lista_de_valores> );
```

- Nesta sintaxe, os valores da lista serão atribuídos na mesma ordem em que se apresenta a lista de colunas.
- A quantidade de elementos da lista de valores deve ser a mesma quantidade de elementos da lista de colunas.

## Comando INSERT

### ■ Sintaxe

```
INSERT INTO <tabela>  
VALUES( <lista_de_valores> );
```

- Colunas opcionais podem não ser informadas, sendo automaticamente atribuído o valor nulo.

## Exemplo SQL

```
insert into times  
  values ( 90, 'SE', 'Sergipe' );
```

```
insert into patrocinadores (pais, cod_pat, nom_pat)  
  values ('EUA', 100, 'Ford');
```

```
insert into jogos  
  values ( 1, 2, 5, getdate(), null );
```

```
insert into jogos ( cod_camp, cod_time1, cod_time2, data )  
  values ( 1, 1, 5, getdate() );
```

## Exemplo SQL

```
insert into times  
  select cod_time, uf_time, nom_time  
  from times  
  where cod_time = 99;
```

## Comando DELETE

### ■ Sintaxe

```
DELETE FROM <tabela>  
[ WHERE <condição> ];
```

- O comando exclui as linhas da tabela, mas não a própria tabela.
- Se a cláusula **WHERE** não for especificada, todas as linhas da tabela são excluídas.

## Comando DELETE

### ■ Sintaxe

```
DELETE FROM <tabela>  
[ WHERE <condição> ];
```

- Quando a cláusula **WHERE** é definida, apenas as linhas cuja condição seja verdadeira serão excluídas.
- Pode ser usado subselect na cláusula **WHERE**.

## Exemplo SQL

```
delete from times;
```

```
delete from times  
  where cod_time in( 10, 15, 30 );
```

```
delete from jogadores  
  where cod_time in( select cod_time  
                    from times  
                    where uf_time in( 'RJ',  
                                       'SP' ) );
```

## Comando UPDATE

### ■ Sintaxe

```
UPDATE <tabela>  
  SET <lista_de_atribuições>  
  [ WHERE <condição> ];
```

- Atualiza as linhas da tabela.
- Se a cláusula **WHERE** não for especificada, todas as linhas da tabela são atualizadas.

## Comando UPDATE

### ■ Sintaxe

```
UPDATE <tabela>  
  SET <lista_de_atribuições>  
  [ WHERE <condição> ];
```

- A cláusula **WHERE** restringe as linhas a serem atualizadas.
- Pode ser utilizado subselect.

## Exemplo SQL

```
update jogadores  
  set dat_nasc = '1979-10-09'),  
     cod_pos = 5  
  where cod_jog = 1;
```

```
update jogadores j  
  set salario = salario * 1.1  
  where exists( select t.cod_time  
                from times t  
                where t.uf_time = 'RJ' and  
                       t.cod_time = j.cod_time );
```

# DDL

## LINGUAGEM DE DEFINIÇÃO DE DADOS

### Comando CREATE TABLE

#### ■ Sintaxe

##### CREATE TABLE

```
[ database_name . [ owner ] . | owner . ] table_name  
( { < column_definition >  
  | column_name AS computed_column_expression  
  | < table_constraint > ::= [ CONSTRAINT constraint_name ]  
    | [ { PRIMARY KEY | UNIQUE } [ ,...n ]  
)
```

## Exemplo SQL

### ■ TABELA PERMANENTE

**create table** campeonatos

```
(
  cod_camp      int          not null check( cod_camp > 0 ),
  dsc_camp     varchar(40)  not null,
  ano          int          not null check( ano > 1972 ),
  tipo         char(1)      not null check( tipo in( 'E','N','R' ) ),
  dat_ini      smalldatetime not null,
  dat_fim     smalldatetime not null,
  def_tipo     char(2),
  constraint campeonatos_pk primary key( cod_camp ),
  constraint campeonatos_ck_datas check( dat_ini < dat_fim )
);
```

## Exemplo SQL

### ■ TABELA TEMPORÁRIA – LOCAL “ # ”

**create table** #campeonato

```
(
  cod_camp      int          not null check( cod_camp > 0 ),
  dsc_camp     varchar(40)  not null,
  ano          int          not null check( ano > 1972 ),
  tipo         char(1)      not null check( tipo in( 'E','N','R' ) ),
  dat_ini      smalldatetime not null,
  dat_fim     smalldatetime not null,
  def_tipo     char(2),
  constraint campeonatos_pk primary key( cod_camp ),
  constraint campeonatos_ck_datas check( dat_ini < dat_fim )
);
```



## Exemplo SQL

### ■ TABELA TEMPORÁRIA – GLOBAL “ ## ”

**create table ##campeonato**

```
(
  cod_camp      int          not null check( cod_camp > 0 ),
  dsc_camp     varchar(40)  not null,
  ano          int          not null check( ano > 1972 ),
  tipo         char(1)      not null check( tipo in( 'E','N','R' ) ),
  dat_ini      smalldatetime not null,
  dat_fim      smalldatetime not null,
  def_tipo     char(2),
  constraint championatos_pk primary key( cod_camp ),
  constraint championatos_ck_datas check( dat_ini < dat_fim )
);
```

## Comando ALTER TABLE

### ■ Sintaxe

```
ALTER TABLE table
{ [ ALTER COLUMN column_name
  { new_data_type [ ( precision [ , scale ] ) ]
    [ COLLATE < collation_name > ]
    [ NULL | NOT NULL ]
    [ { ADD | DROP } ROWGUIDCOL ]
  ]
| ADD
  { [ < column_definition > ]
    | column_name AS computed_column_expression
  } [ ,...n ]
| [ WITH CHECK | WITH NOCHECK ] ADD
  { < table_constraint > } [ ,...n ]
| DROP
  { [ CONSTRAINT ] constraint_name
    | COLUMN column } [ ,...n ]
| { CHECK | NOCHECK } CONSTRAINT
  { ALL | constraint_name [ ,...n ] }
| { ENABLE | DISABLE } TRIGGER
  { ALL | trigger_name [ ,...n ] }
}
```

## Comando DROP TABLE

### ■ Sintaxe

**DROP TABLE** *table\_name*

## Comando CREATE VIEW

### ■ Sintaxe

```
CREATE VIEW [ < database_name > . ] [ < owner > . ]  
             view_name [ ( column [ ,...n ] ) ]  
 [ WITH < view_attribute > [ ,...n ] ]
```

**AS**

*select\_statement*

[ **WITH CHECK OPTION** ]

< view\_attribute > ::=

{ **ENCRYPTION** | **SCHEMABINDING** | **VIEW\_METADATA** }

## Exemplo SQL

- **CREATE VIEW**

```
create view v_jogadores
```

```
as
```

```
select t.cod_time, t.nom_time, j.cod_jog,  
        j.nom_jog, p.dsc_pos
```

```
from times t, jogadores j, posicoes p
```

```
where t.cod_time = j.cod_time and
```

```
        j.cod_pos = p.cod_pos
```

## Comando DROP VIEW

- **Sintaxe**

```
DROP VIEW VIEW_name
```

## Comando CREATE INDEX

### ■ Sintaxe

```
CREATE [ UNIQUE ]  
      [ CLUSTERED | NONCLUSTERED ]  
INDEX index_name  
ON { table | view } ( column [ ASC | DESC ] [ ,...n ] )
```

## Exemplo SQL

```
create index jogadores_idx_cod_time  
on jogadores( cod_time );
```

```
create index jogadores_idx_nom_jog  
on jogadores( nom_jog );
```

## Comando DROP INDEX

### ■ Sintaxe

**DROP INDEX** *'table.index | view.index'* [ ,...*n* ]

***FUNÇÕES DO  
DIA-A-DIA***

## FUNÇÕES DO DIA – A – DIA

- GETDATE();
- DATEADD();
- DATEPART();
- DATENAME();
- DATEDIFF();
- CONVERT();
- CASE;
- RTRIN / LTRIN
- LIKE;
- SUBSTRING();
- CHARINDEX();
- REPLACE();
- DATALENGTH()  
ou LEN();
- REPLICATE();
- SP\_HELP / TEXT
- DICAS

## Exemplo SQL

- GETDATE()
  - Retorna a data e hora do sistema.

```
SELECT GETDATE()
```

## Exemplo SQL

### ■ DATEADD()

- Adiciona ou subtrai dia, mês e ano em uma data qualquer.

```
SELECT DATEADD(DAY      , 5, GETDATE())  
SELECT DATEADD(MONTH   , 5, GETDATE())  
SELECT DATEADD(YEAR    , 5, GETDATE())
```

## Exemplo SQL

### ■ DATEPART()

- Retorna uma parte da data, dia, mês ou ano.

```
SELECT DATEPART(DAY      , GETDATE())  
SELECT DATEPART(MONTH   , GETDATE())  
SELECT DATEPART(YEAR    , GETDATE())
```

## Exemplo SQL

### ■ DATENAME()

- Retorna somente o nome por extenso do mês solicitado.

```
SELECT DATENAME(DAY , GETDATE())  
SELECT DATENAME(MONTH , GETDATE())  
SELECT DATENAME(YEAR , GETDATE())
```

## Exemplo SQL

### ■ DATEDIFF()

- Retorna a diferença entre datas em dia, mês ou ano.

```
SELECT DATEDIFF(DAY, GETDATE(),  
               DATEADD(DAY, 8 , GETDATE()))  
  
SELECT DATEDIFF(MONTH, GETDATE(),  
               DATEADD(DAY, 5 , GETDATE()))  
  
SELECT DATEDIFF(YEAR, GETDATE(),  
               DATEADD(DAY, 300, GETDATE()))
```



## Exemplo SQL

### ■ CONVERT()

- Retorna a data ou hora em formato específicos de acordo com o número informado no comando, [100..114].

```
DECLARE @I INT
SET @I = 100

WHILE (@I <= 114)
BEGIN
    SELECT CONVERT(VARCHAR(255),
        GETDATE(), @I) , @I 'NUMERO'
    SET @I = @I + 1
END
```

## Exemplo SQL

### ■ CASE

- Possibilidade de escolha em um determinado conjunto de valores.

```
SELECT CASE '6'
    when '1' then 'I'
    when '2' then 'II'
    when '3' then 'III'
    when '4' then 'IV'
    when '5' then 'V'
ELSE
    'Valor Fora da Faixa!'
END 'RESULTADO'
```

## Exemplo SQL

### ■ LTRIN / RTRIN

- Remove os espaços em brancos da esquerda com LTrim ou da direita com RTrim.

```
select LTrim('  Glauco Luiz')
```

```
select RTrim('Glauco Luiz  ')
```

## Exemplo SQL

### ■ LIKE

- Possibilidade de uma busca mais genérica.

```
SELECT *  
FROM MASTER.DBO.SYSOBJECTS  
WHERE NAME LIKE '%USER'
```

## Exemplo SQL

### ■ SUBSTRING()

- Subtrair uma parte de uma string.

```
SELECT 'RESULTADO' = SUBSTRING('BANESE', 5, 2)
```

## Exemplo SQL

### ■ CHARINDEX()

- Retorna uma posição do CHAR que se deseja localizar.

```
SELECT NAME, CHARINDEX('P', NAME, 4)  
      'RESULTADO'  
FROM MASTER.DBO.SYSOBJECTS  
WHERE NAME LIKE '%USER'
```

## Exemplo SQL

### ■ REPLACE()

- Substitui um conjunto de caracteres por outro conjunto.

```
SELECT  
    REPLACE('ABCDEFGHijklMNOPQRSTUVWXYZ', 'Hijlm  
        NOPQr', '-BANESE- ') 'RESULTADO'
```

## Exemplo SQL

### ■ DATALENGTH() ou LEN()

- Retorna a quantidade de CHAR em um determinado texto.

```
SELECT DATALENGTH('BANCO DO ESTADO  
    DE SERGIPE') 'RESULTADO'
```

```
SELECT LEN('BANCO DO ESTADO  
    DE SERGIPE') 'RESULTADO'
```

## Exemplo SQL

- **REPLICATE()**
  - Repete o valor informado.

```
SELECT REPLICATE('BANESE ', 3) 'RESULTADO'
```

## Exemplo SQL

- **SP\_HELP** ou **SP\_HELPTEXT**
  - Mostra os detalhes do objeto informado.

```
SP_HELP <NOME DO OBJETO>
```

```
SP_HELPTEXT <NOME DO OBJETO>
```

## Exemplo SQL

### ■ DICAS

- Preenche com zeros à esquerda.

```
select right('000000000000' +  
           convert(varchar(11), 123),11) CPF
```

- Extração de caracteres da tabela ASCII

## Exemplo SQL

```
declare @i int  
set @i = 100
```

```
while (@i < 200)  
begin  
    select char(@i) + convert(varchar(03), @i)  
    set @i = @i + 1  
end
```

## Exemplo SQL

```
declare
@DescricaoProduto      char (50)           , -- Ex: Freezer
@PrecoProduto          varchar (10)        , -- Ex: 450,00
@TempoGarantiaFabricante  varchar (02)    , -- Ex: 06, 12 ou Meses

set @DescricaoProduto      = 'Freezer'
set @PrecoProduto          = '450,00'
set @TempoGarantiaFabricante = '06'

select ltrim(@DescricaoProduto)      + char(124) +
ltrim(@PrecoProduto)                 + char(124) +
ltrim(@TempoGarantiaFabricante)      + char(124)

select rtrim(@DescricaoProduto)      + char(124) +
rtrim(@PrecoProduto)                 + char(124) +
rtrim(@TempoGarantiaFabricante)      + char(124)

select @DescricaoProduto              + char(124) +
@PrecoProduto                         + char(124) +
@TempoGarantiaFabricante              + char(124)
```

**Obrigado!!!**