Operar consultas com o comando SELECT

ESTRUTURA BÁSICA DE UM COMANDO SELECT

Obs: O comando SELECT é usado para exibir dados resultantes de uma consulta. Os dados podem ser colunas físicas de uma tabela, colunas calculadas ou mesmo resultado do uso de expressões e funções.

```
SELECT COLUNA1 [[AS] APELIDOCOLUNA1],
COLUNA2 [[AS] APELIDOCOLUNA2],
…
COLUNAN [[AS] APELIDOCOLUNAN]
FROM TABELA;
```

Uma sintaxe complexa envolve uma série de cláusulas e recursos bastante úteis para consultas de maior complexidade.

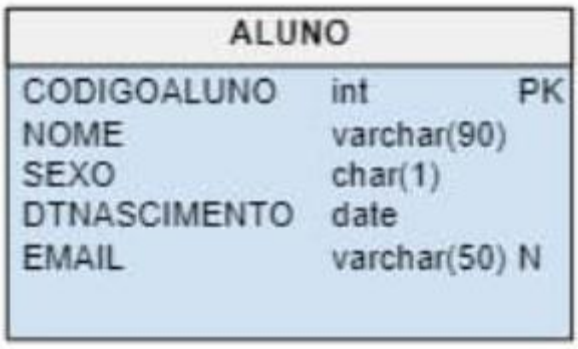
Na prática o comando SELECT depende da consulta desejada, pode ser usado de diferentes formas para obter o mesmo resultado. A cláusula SELECT realiza a operação de projeção da Álgebra Relacional;

Caso haja interesse em exibir todas as colunas especificadas em uma consulta, basta adicionar um "*", conforme a seguir:

SELECT * FROM TABELA;

Alguns SGBDs, como o PostgreSQL, implementam uma forma simplificada do comando SELECT * FROM TABELA, que é simplesmente TABLE tabela (você pode testar isso no PostgreSQL).

Vamos estudar um exemplo:




```
CREATE TABLE ALUNO (
      CODIGOALUNO int NOT NULL,
      NOME varchar(90) NOT NULL,
      SEXO char(1) NOT NULL,
```

```
	DTNASCIMENTO date NOT NULL,
	EMAIL varchar(30) NULL,
	CONSTRAINT ALUNO_pk PRIMARY KEY (CODIGOALUNO)
);

INSERT INTO ALUNO (CODIGOALUNO,NOME,SEXO,DTNASCIMENTO,EMAIL) VALUES
(1,'JOSÉ FRANCISCO TERRA','M','28/10/1989','JFT@GMAIL.COM');
INSERT INTO ALUNO (CODIGOALUNO,NOME,SEXO,DTNASCIMENTO,EMAIL) VALUES
(2,'ANDREY COSTA FILHO','M','20/10/1999','ANDREYCF@HOTMAIL.COM');
INSERT INTO ALUNO (CODIGOALUNO,NOME,SEXO,DTNASCIMENTO,EMAIL) VALUES
(3,'PATRÍCIA TORRES LOUREIRO','F','20/10/1980','PTORRES@GMAIL.COM');
INSERT INTO ALUNO (CODIGOALUNO,NOME,SEXO,DTNASCIMENTO,EMAIL) VALUES
(4,'CARLA MARIA MACIEL','F','20/11/1996',NULL);
INSERT INTO ALUNO (CODIGOALUNO,NOME,SEXO,DTNASCIMENTO,EMAIL) VALUES
(5,'LEILA SANTANA COSTA','F','20/11/2001',NULL);
```

CONSULTA 01

Exibir todas as informações dos alunos.
SELECT * FROM ALUNO;

Esse comando irá exibir todas as linhas da tabela;

CONSULTA 02

Retornar o código, o nome e a data de nascimento de todos os alunos.

```
SELECT CODIGOALUNO, NOME, DTNASCIMENTO
FROM ALUNO;
```

Em especial, pode ser interessante “renomear” as colunas resultantes da consulta, visando tornar os resultados mais “apresentáveis” ao usuário da aplicação. Por exemplo, a consulta 02 pode ser reescrita conforme a seguir:

```
SELECT CODIGOALUNO AS "Matrícula",
NOME AS "Nome do discente",
DTNASCIMENTO AS "Data de nascimento"
FROM ALUNO;
```

É importante ressaltar que, na tabela anterior, o nome apresentado para cada coluna não existe fisicamente no banco de dados.

FUNÇÕES DE DATA E HORA

Função	O que retorna?
current_date	data de hoje
current_time	hora do dia
current_timestamp	data e a hora
extract (campo from fonte)	subcampos de data e hora: século, ano, dia, mês

Vejamos alguns exemplos:

```
--Funções de data e hora;
SELECT CURRENT_DATE AS "DATA ATUAL",
        CURRENT_TIME AS "HORA ATUAL",
        CURRENT_TIMESTAMP "DATA E HORA ATUAIS",
        EXTRACT( DOY FROM CURRENT_DATE) AS "DIA DO ANO",
        --DIAS DA SEMANA--
        EXTRACT(DOW FROM CURRENT_DATE) AS "DIAS DA SEMANA",
        EXTRACT(DAY FROM CURRENT_DATE) AS "DIA ATUAL",
        EXTRACT(MONTH FROM CURRENT_DATE) AS "MÊS ATUAL",
        EXTRACT(YEAR FROM CURRENT_DATE) AS "ANO ATUAL",
        EXTRACT(CENTURY FROM CURRENT_DATE) AS "SÉCULO ATUAL";
```

Observe que utilizamos o qualificador AS “Apelido” para facilitar o entendimento do retorno de cada função. Note também que não há cláusula FROM na consulta, visto que todas as colunas retornadas representam o resultado de funções do PostgreSQL sem envolver qualquer tabela do domínio da aplicação.

Exibindo o nome do dia da semana

```
--Exibindo o nome do dia da semana;
SELECT CASE
WHEN EXTRACT(DOW FROM CURRENT_DATE) = 0 THEN 'DOMINGO'
WHEN EXTRACT(DOW FROM CURRENT_DATE) = 1 THEN 'SEGUNDA'
WHEN EXTRACT(DOW FROM CURRENT_DATE) = 2 THEN 'TERÇA'
WHEN EXTRACT(DOW FROM CURRENT_DATE) = 3 THEN 'QUARTA'
WHEN EXTRACT(DOW FROM CURRENT_DATE) = 4 THEN 'QUINTA'
```

```
WHEN EXTRACT(DOW FROM CURRENT_DATE) = 5 THEN 'SEXTA'

WHEN EXTRACT(DOW FROM CURRENT_DATE) = 6 THEN 'SÁBADO'

END AS "NOME DO DIA DA SEMANA";
```

Observe que construímos uma lógica utilizando o comando CASE, que é equivalente ao comando IF:, cada linha com a cláusula WHEN avalia expressão que retorna um inteiro representativo do dia da semana, caso a expressão tenha valor lógico verdadeiro.

A tradução do END AS ao final do comando idica: terminar como.

Ou seja como o campo sera chamado ao final de tudo.

WHEN significa quando, e THEN s significa então;

Calculando idade e faixa etária

Em geral, quando estamos diante de alguma coluna representativa da data de nascimento de uma pessoa, é comum extrair informações derivadas, tais como idade e faixa etária. Por exemplo, o código a seguir retorna o nome, a data de nascimento e a idade dos alunos:

```
--Calculando a idade dos alunos;

SELECT NOME,

        DTNASCIMENTO,

        AGE(DTNASCIMENTO) AS "IDADE [ANO/MÊS/DIA]",

        EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) AS "IDADE DO ALUNO"

FROM ALUNO;
```

Perceba que na linha 3 utilizamos a função AGE, que retorna uma frase representativa da informação sobre a idade em questão. Na linha 4, usamos a função EXTRACT para exibir a idade do aluno.

OBS: AGE significa IDADE;

Muito bem, agora, vamos exibir o nome, a idade e a faixa etária dos alunos.

```
--Exibindo o nome, a idade e a faixa etaria dos alunos;

SELECT  NOME,

EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) AS "IDADE DO ALUNO",

CASE WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) <=20 THEN '1.ATÉ OS 20
ANOS'

WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 21 AND 30 THEN '2.
21 ATÉ OS 30 ANOS'

WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 31 AND 40 THEN '3.
31 ATÉ OS 40 ANOS'

WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 41 AND 50 THEN '4.
41 ATÉ OS 50 ANOS'
```

```
WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 51 AND 60 THEN '5.
51 ATÉ OS 60 ANOS'

END AS "FAIXA ETÁRIA"

FROM ALUNO;
```

Perceba que cada linha com a cláusula WHEN avalia a expressão que retorna uma faixa etária de acordo com a idade do aluno.

FUNÇÕES DE RESUMO OU DE AGREGAÇÃO

As funções a seguir são úteis para obtermos resumo dos dados de alguma tabela:

Função	O que retorna?
COUNT(*)	número de linhas da consulta
MIN(COLUNA/EXPRESSÃO)	menor de uma coluna ou expressão
AVG(COLUNA/EXPRESSÃO)	valor médio da coluna ou expressão
MAX(COLUNA/EXPRESSÃO)	maior valor de uma coluna ou expressão
SUM(COLUNA/EXPRESSÃO)	soma dos valores de uma coluna ou expressão
STDDEV(COLUNA/EXPRESSÃO)	desvio padrão dos valores de uma coluna ou expressão
VARIANCE(COLUNA/EXPRESSÃO)	variância dos valores de uma coluna ou expressão

Vejamos um alguns exemplos:

```
SELECT
      COUNT(*) AS "NÚMERO DE ALUNOS",
      MIN(EXTRACT(YEAR FROM AGE(DTNASCIMENTO))) AS "MENOR DE IDADE",
      AVG(EXTRACT(YEAR FROM AGE(DTNASCIMENTO))) AS "IDADE MÉDIA",
      MAX(EXTRACT(YEAR FROM AGE(DTNASCIMENTO))) AS "MAIOR DE IDADE",
      SUM(EXTRACT(YEAR FROM AGE(DTNASCIMENTO)))/COUNT(*) AS "IDADE
MEDIA"
FROM ALUNO;
```

Perceba que, como estamos usando somente o comando SELECT/FROM, cada função é calculada levando em consideração todos os registros da tabela.

Listando resumos em uma linha

Suponha que haja interesse em conhecer os quantitativos de cursos, disciplinas e alunos do nosso banco de dados.

Curso:

```
SELECT COUNT(*) NCURSOS FROM CURSO;
```

Disciplina:

```
SELECT COUNT(*) NDISCIPINAS FROM DISCIPLINA;
```

Aluno:

```
SELECT COUNT(*) NALUNOS FROM ALUNO;
```

OBS: Convém ressaltar que o comando é válido, visto que, no PostgreSQL, a cláusula FROM não é obrigatória.

CRIANDO TABELA A PARTIR DE CONSULTA

Em alguns momentos, você terá interesse em salvar os resultados de uma consulta em uma nova tabela.

```
--criando tabela  apartir de consulta;

CREATE TABLE TESTE AS

SELECT NOME,

         EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) AS "IDADE ALUNO",

         CASE WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) <= 20 THEN '1.
ATÉ 20 ANOS'

                  WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 21
AND 30 THEN '2. 21 A 30 ANOS'

                  WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 31
AND 40 THEN '3. 31 A 40 ANOS'

                  WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 41
AND 50 THEN '4. 41 A 50 ANOS'

                  WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 51
AND 60 THEN '5. 51 A 60 ANOS'

                  WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) > 60 THEN
'6. MAIS DE 60 ANOS'

            END AS "FAIXA ETÁRIA"

FROM ALUNO;

SELECT * FROM TESTE;
```

CRIANDO *VIEW* A PARTIR DE CONSULTA

Outro recurso interessante, diretamente relacionado ao processo de construção de consultas, é o objeto *view* (visão). Uma *view* encapsula a complexidade da consulta

SQL, que a forma. Para criar esse objeto, usa-se o comando CREATE VIEW <CONSULTA>.

```
--Creando view;

CREATE VIEW VTESTE AS

SELECT NOME,

        EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) AS "IDADE ALUNO",

        CASE WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) <= 20
THEN '1. ATÉ 20 ANOS'

                WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO))
BETWEEN 21 AND 30 THEN '2. 21 A 30 ANOS'

                WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO))
BETWEEN 31 AND 40 THEN '3. 31 A 40 ANOS'

                WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO))
BETWEEN 41 AND 50 THEN '4. 41 A 50 ANOS'

                WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO))
BETWEEN 51 AND 60 THEN '5. 51 A 60 ANOS'

                WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) > 60
THEN '6. MAIS DE 60 ANOS'

            END AS "FAIXA ETÁRIA"

FROM ALUNO;

SELECT * FROM TESTE;
```

No exemplo, o SGBD criará uma view denominada VTESTE. Na prática, quando usuário submeter, por exemplo, a consulta SELECT * FROM VTESTE, o SGBD executará o código associado à view em questão.

O que é uma view?

Ao criarmos uma view, podemos filtrar o conteúdo de uma tabela a ser exibida, já que a função da view é exatamente essa: filtrar tabelas, servindo para agrupá-las, protegendo certas colunas e simplificando o código de programação.

É importante salientar que, mesmo após o servidor do SQL Server ser desligado, a view continua “viva” no sistema, assim como as tabelas que criamos normalmente. As views não ocupam espaço no banco de dados.

Vantagens das Views

Temos muitos motivos e vantagens para usarmos views em nossos projetos. A seguir são citados três que podem fazer a diferença:

Reuso: as views são objetos de caráter permanente. Pensando pelo lado produtivo isso é excelente, já que elas podem ser lidas por vários usuários simultaneamente.

Segurança: as views permitem que ocultemos determinadas colunas de uma tabela. Para isso, basta criarmos uma view com as colunas que acharmos necessário que sejam exibidas e as disponibilizarmos para o usuário.

Simplificação do código: as views nos permitem criar um código de programação muito mais limpo, na medida em que podem conter um SELECT complexo. Assim, criar views para os programadores a fim de poupá-los do trabalho de criar SELECT's é uma forma de aumentar a produtividade da equipe de desenvolvimento.

Comandos:

CREATE VIEW <nome da view> AS <select>;

ALTER VIEW <nome da view> AS <select>;

DROP VIEW <nome da view>;