Banco de Dados 1
Prof. Gilberto Assis
Roteiro de Aula

Introdução aos comandos SQL

SQL ou Structure Query Language é a linguagem padrão de interface com os bancos de dados.

A SQL está conceitualmente dividida em 3 grupos:
DML (Data Manipulation Language) – que trata da manipulação dos registros armazenados no banco de dados através dos comandos SELECT, INSERT, DELETE e UPDATE.
DDL (Data Definition Language) – responsável pelos comandos que manipulam as estruturas dos objetos (Tabelas, Views, Triggers entre outros), destacam-se o CREATE, ALTER e DROP.
DCL (Data Control Language) – usados na gestão (conscessão e revogação) das permissões sobre os objetos através dos comandos GRANT e REVOKE.

DML - Data Manipulation Language

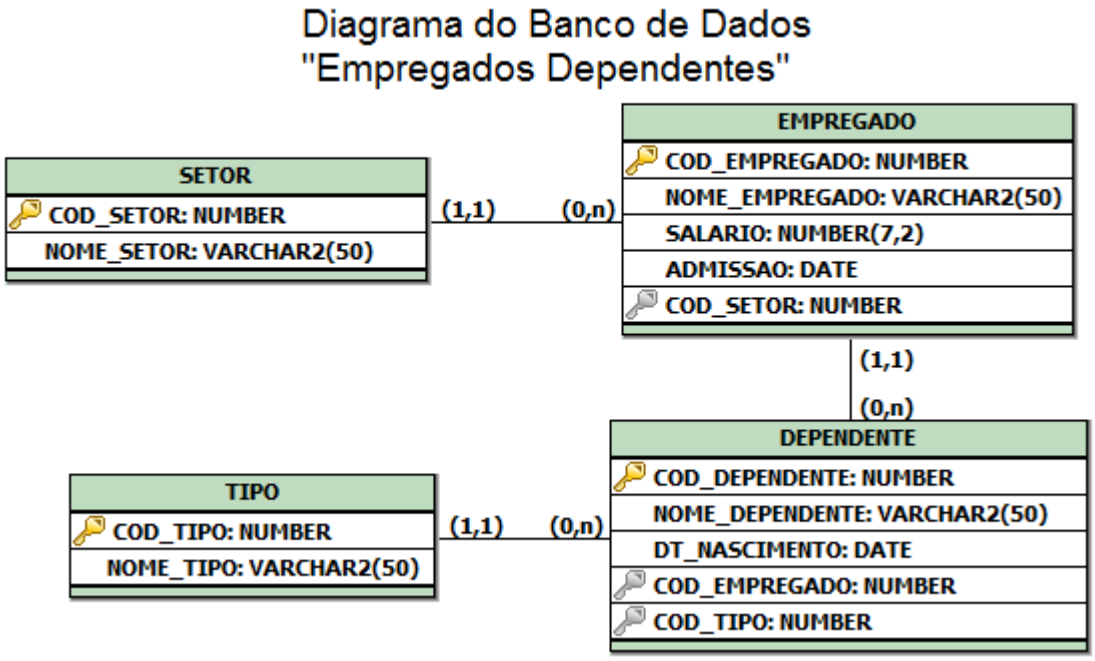
O comando SQL que trabalharemos a partir de agora será o SELECT que objetiva buscar, selecionar ou ainda apresentar os registros de uma ou mais tabelas do banco de dados.
O comando SELECT é extremamente poderoso e capaz de retornar qualquer informação, respeitadas as cláusulas de segurança, na ordem que se deseja, independente do volume de informações abordado.
A estrutura do comando SELECT é:

```
select [distinct] {*, colunas [alias], expressões, funções, ..}
from {tabelas [alias],}
[where condição]
[group by colunas]
[having condição]
[order by colunas [asc | desc]];
```

Reparem que o comando é divido em 6 partes:

a) SELECT <o que será mostrado> - obrigatoriamente o comando select deve ser acompanhado da lista de informações que serão mostradas. Essa lista pode ser composta por colunas da tabela, funções de data ou strings e também por expressões matemáticas.
b) FROM <de onde serão buscadas as informações> - essa parte do comando também é obrigatória e consiste na relação de tabelas onde serão buscadas as informações.
c) WHERE <indica a condição de filtro dos resultados (registros)> - essa parte do comando é opcional e usada apenas quando se deseja limitar o retorno das informações disponíveis nas tabelas.
d) GROUP BY <colunas> - também opcional essa parte do comando é usada para funções de agrupamento dos resultados.
e) HAVING <condição> - equivale ao WHERE, mas com abrangência apenas nos resultados retornados pelo GROUP BY.
f) ORDER BY <colunas> - ordena o resultado da consulta pela coluna destacada, essa ordenação poderá ser ascendente ASC ou descendente DESC.

Para todos os exemplos será usado o modelo de dados:



SELEÇÃO DE UMA TABELA

Se quisermos retornar todos os registros presentes em todas as colunas de uma dada tabela, ou seja, fazer uma busca indiscriminada temos que indicar ao banco de dados todas as colunas da tabela.

SELECT cod_setor, nome_setor FROM setor;

```
COD_SETOR            NOME_SETOR
-------------------- --------------------
1                    Diretoria
2                    Recursos Humanos
3                    Secretária
4                    Produção de Sólidos
5                    Produção de Liquidos
6                    Estoque
7                    Comercial

7 rows selected
```

Deve-se observar a estrutura do comando acima:

Primeiro o comando SELECT (para indicar que queremos fazer uma pesquisa no banco de dados), depois a lista de colunas separadas por vírgula (cod_setor, nome_setor) presentes na tabela que desejamos recuperar as informações, depois o comando FROM com a lista de tabelas que buscaremos os registros (setor) e finalmente o caractere de fim de comando ;.

O comando acima retornará todos os registros da tabela setor na ordem em que foram inseridos na tabela.

SELEÇÃO DE TODAS AS COLUNAS DA TABELA
Para mostrar todos os registros de todas as colunas de uma dada tabela usamos o caractere *

SELECT * FROM setor;

```
COD_SETOR            NOME_SETOR
-------------------- --------------------
1                    Diretoria
2                    Recursos Humanos
3                    Secretária
4                    Produção de Sólidos
5                    Produção de Liquidos
6                    Estoque
7                    Comercial

7 rows selected
```

Deve-se observar no comando acima:
A estrutura do comando NÃO mudou (select <o que será mostrado> FROM <de onde será mostrado>), entretanto usamos o caractere * para indicar que queremos TODAS as colunas daquela tabela. O resultado será o mesmo do comando anterior.

EXIBIR AS COLUNAS EM ORDEM DIFERENTE DE SUA CRIAÇÃO
O comando SELECT nos permite mostrar as colunas da maneira que quisermos, inclusive fora de sua ordem de criação, para tanto basta escrever o comando na ordem desejada.

Considere a estrutura da tabela empregado:

```
DESC empregado;
Nome                 Tipo
-------------------- --------------------
COD_EMPREGADO        NUMBER
NOME_EMPREGADO       VARCHAR2(50)
IDADE_EMPREGADO      NUMBER
ADMISSAO             DATE
COD_SETOR            NUMBER

5 rows selected
```

* Usa-se o comando DESC para mostrar a estrutura de uma tabela.
O comando abaixo retornará os registros da tabela empregado em ordem de coluna diferente da existente na tabela.

```
SELECT admissao, salario, cod_setor, nome_empregado, cod_empregado
FROM empregado;
```

Deve-se observar no comando acima:
A ordem em que as colunas foram colocadas no comando será a apresentada no resultado, independente da ordem que elas foram criadas na tabela. Novamente usamos a vírgula como separador de colunas e o ponto e vírgula como finalizador do comando.

ADMISSAO	SALARIO	COD_SETOR	NOME_EMPREGADO	COD_EMPREGADO
18/01/12	12652,88	1	Chico Bento	1
18/01/12	13000	1	Tio Patinhas	2
19/01/12	2552,78	3	Madame Mafalda	3
19/01/12	3556	2	Tião Gavião	4
19/01/12	1244,21	4	Pato Donald	5
19/01/12	1244,21	4	Pateta	6
19/01/12	2100,19	5	Patolino	7
30/04/12	3100	7	Saci Pererê	8
02/04/12	1900	5	Pernalonga	9
02/04/12	1900	5	Helena Pêra	10
30/04/12	1900	5	Marge Simpson	11
14/05/12	980	6	Homer Simpson	12
23/05/12	1300	6	Cebolinha	13
30/05/12	2950,23	7	Visconde de Sabugosa	14
18/06/12	1110	4	Mickey Mouse	15

15 linhas selecionadas

ALIASES ou APELIDOS

Os comandos SQL permitem a criação de ALIAS ou APELIDOS para colunas e/ou tabelas. Isso é útil para formatar o cabeçalho da coluna do resultado do comando e também na referencias a objetos (será abordado no relacionamento entre tabelas).

```
SELECT cod_empregado matricula, nome_empregado "Empregado"
FROM empregado;
```

Vejam que no comando acima que acrescentamos o apelido matricula logo após a coluna cod_matricula e o apelido "Empregado" após a coluna nome_empregado. Quando colocamos o apelido entre aspas "" estamos dizendo ao Oracle para exibir o cabeçalho exatamente da forma que escrevemos, caso contrário será exibido em maiúsculo.

Importante: ALIASES com palavras compostas devem obrigatoriamente ser limitados por aspas "".

```
MATRICULA     Empregado
------------- ----------------
1             Chico Bento
2             Tio Patinhas
3             Madame Mafalda
4             Indiana Jones
5             Pato Donald
6             Pateta
7             Patolino
8             Pernalonga
9             Piu-Piu
10            Marge Simpson
11            Homer Simpson
12            João Bafo
13            Batman
14            Robbin
15            Mickey Mouse

15 rows selected
```

ORDENANDO OS RESULTADOS

É possível, através do comando SELECT, escolher em qual ordem os resultados serão apresentados. Basta para isso acrescentar, no final do comando, a cláusula ORDER BY.

```
SELECT nome_setor FROM setor ORDER BY nome_setor;
```

No comando acima deve-se observar:
A inclusão da cláusula ORDER BY no final do comando indicando a coluna que será ordenada (nome_setor). Quando a chave ASC (ascendente) ou DESC (descendente) é omitida no comando o banco de dados considera a ordem ascendente.

```
NOME_SETOR
------------------------
Comercial
Diretoria
Estoque
Produção de Liquidos
Produção de Sólidos
Recursos Humanos
Secretária

7 rows selected
```

Já o comando abaixo trará o mesmo resultado ordenado inversamente por causa do DESC.

```
SELECT nome_setor FROM setor ORDER BY nome_setor DESC;
```

```
NOME_SETOR
----------------------
Secretária
Recursos Humanos
Produção de Sólidos
Produção de Liquidos
Estoque
Diretoria
Comercial

7 rows selected
```

Deve-se ressaltar que o resultado poderá ser ordenado por mais de uma coluna ao mesmo tempo, de tal modo que o banco ordenará a primeira coluna do ORDER BY e havendo valores idênticos ordenará pela próxima coluna do ORDER BY e assim por diante.

```
SELECT nome_empregado, salario
FROM empregado
ORDER BY salario DESC, nome_empregado ASC;
```

No comando acima deve-se observar:
A cláusula ORDER BY contém duas colunas: SALARIO que será a primeira a ser ordenada, e nesse caso inversamente (DESC), e depois a coluna NOME_EMPREGADO que é a próxima a ser ordenada, nesse caso de ascendente (ASC).

```
NOME_EMPREGADO            SALARIO
---------------------------------
Tio Patinhas              13000
Chico Bento               12652,88
Tião Gavião               3556
Saci Pererê               3100
Visconde de Sabugosa      2950,23
Madame Mafalda            2552,78
Patolino                  2100,19
Helena Pêra               1900
Marge Simpson             1900
Pernalonga                1900
Cebolinha                 1300
Pateta                    1244,21
Pato Donald               1244,21
Mickey Mouse              1110
Homer Simpson             980

 15 linhas selecionadas
```

O resultado anterior mostra a relação dos empregados ordenados pelo salário, do maior para o menor salário, deve-se observar que nos casos em que os funcionários têm o mesmo salário a ordenação passa para a coluna de nome do empregado por ordem alfabética (ASC). Repare no fragmento abaixo que todos os funcionários têm o mesmo salário então a ordenação se deu pelo nome.

```
Helena Pêra               1900
Marge Simpson             1900
Pernalonga                1900
```

DICA: pode-se indicar a qual coluna deseja-se ordenar indicando a posição da coluna em relação ao select.

```
Ex: SELECT admissao, salario, cod_setor, nome_empregado, cod_empregado
FROM empregado
ORDER BY 4 DESC;
```

No comando acima a ordenação se dará pela quarta coluna (4) do comando select, ou seja pela coluna nome_empregado.

RESTRINGIR OS RESULTADOS POR COLUNAS

O comando SELECT nos permite mostrar apenas as colunas que desejamos, ou seja, nos permite restringir o que será mostrado, para tanto basta indicar somente colunas alvo da pesquisa.

SELECT nome_empregado FROM empregado;

Deve-se observar que a estrutura do comando acima NÃO mudou, mas ficou explicito que desejamos receber os registros apenas da coluna NOME_EMPREGADO da (FROM) tabela EMPREGADO. Cujo resultado será:

```
NOME_EMPREGADO
------------------------
Chico Bento
Tio Patinhas
Madame Mafalda
Tião Gavião
Pato Donald
Pateta
Patolino
Saci Pererê
Pernalonga
Helena Pêra
Marge Simpson
Homer Simpson
Cebolinha
Visconde de Sabugosa
Mickey Mouse

 15 linhas selecionadas
```

RESTRINGIR OS RESULTADOS PELA CLÁUSULA WHERE

Essa cláusula nos permite filtrar os resultados do comando SELECT de modo a diminuir o universo de registros retornados para somente aqueles desejados.

Basicamente a cláusula WHERE (onde) compara o registro da tabela com a regra estabelecida pelo usuário.
Assim, o comando

```
SELECT nome_empregado
FROM empregado
WHERE salario > 5000.00;
```

retorna apenas os registros (NOME_EMPREGADO) que atendem a regra de salário maior que R$5000,00.

```
NOME_EMPREGADO
------------------
Chico Bento
Tio Patinhas
2 linhas selecionadas
```

Perceba que a estrutura do comando SELECT não mudou, apenas foi acrescentado a cláusula de restrição de registros WHERE indicando que o banco de dados deverá retornar apenas aqueles empregados com salário maior que 5000,00.
OBS: o separador de decimal é representado por ponto (.) e não por vírgula (,).

Na sequência mostraremos as várias opções de restrições com a cláusula WHERE.

OPERADORES DE COMPARAÇÃO

OPERADOR	FUNÇÃO
=	Igual a
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Maior ou igual a
<> ou !=	Diferente de

Operador =
Realiza uma comparação simples e literal retornando somente aqueles registros que satisfazem exatamente a igualdade.

```
SELECT salario
FROM empregado
WHERE nome_empregado = 'Chico Bento';
```

Esse comando retorna somente o salário do empregado de nome Chico Bento, o usuário deve perceber que a estrutura do comando SELECT não mudou, deve também observar que o nome do empregado está entre aspas simples com o objetivo de indicar ao banco de dados que o valor a ser comparado é do tipo texto (caracteres de texto). Finalmente devemos nos ater ao fato de que a comparação com registros da tabela é CASE SENSITIVE, ou seja diferencia MAIÚSCULA de minúscula.
Exemplo: 'chico bento' é diferente de 'CHICO BENTO' que é diferente de 'Chico Bento'.

O operador de igualdade também funciona para valores numéricos e de data.

```
SELECT nome_empregado
FROM empregado
WHERE cod_empregado = 5;
```

Esse comando retornará somente o empregado cujo o código é igual a 5.

```
NOME_EMPREGADO
---------------
Pato Donald
```

Operadores <, <=,>, >= e !=

Para retornar os empregados e respectivos salários que recebem menos que R$1900,00 teríamos:

```
SELECT nome_empregado, salario
FROM empregado
WHERE salario < 1900;
```

```
NOME_EMPREGADO                                     SALARIO
-------------------------------------------------- ----------
Pato Donald                                        1244,21
Pateta                                             1244,21
Homer Simpson                                      980
Cebolinha                                          1300
Mickey Mouse                                       1110
```

Para retornar os empregados que recebem menos ou igual a R$1900,00, teríamos o comando:

```
SELECT nome_empregado, salario
FROM empregado
WHERE salario <= 1900;
```

```
NOME_EMPREGADO                                     SALARIO
-------------------------------------------------- --------
Pato Donald                                        1244,21
Pateta                                             1244,21
Pernalonga                                         1900
Helena Pêra                                        1900
Marge Simpson                                      1900
Homer Simpson                                      980
Cebolinha                                          1300
Mickey Mouse                                       1110
 8 linhas selecionadas
```

Para retornar somente os empregados que recebem mais que R$1900,00, teríamos o comando:

```
SELECT nome_empregado, salario
FROM empregado
WHERE salario > 1900;
```

```
NOME_EMPREGADO                                     SALARIO
-------------------------------------------------- ---------
Chico Bento                                        12652,88
Tio Patinhas                                       13000
Madame Mafalda                                     2552,78
Tião Gavião                                        3556
Patolino                                           2100,19
Saci Pererê                                        3100
Visconde de Sabugosa                               2950,23

 7 linhas selecionadas
```

Para retornar os empregados que recebem mais ou igual a R$1900,00, teríamos o comando:

```
SELECT nome_empregado, salario
FROM empregado
WHERE salario >= 1900;
```

```
NOME_EMPREGADO                                     SALARIO
-------------------------------------------------- ---------
Chico Bento                                        12652,88
Tio Patinhas                                       13000
Madame Mafalda                                     2552,78
Tião Gavião                                        3556
Patolino                                           2100,19
Saci Pererê                                        3100
Pernalonga                                         1900
Helena Pêra                                        1900
Marge Simpson                                      1900
Visconde de Sabugosa                               2950,23

 10 linhas selecionadas
```

O comando abaixo retorna todos os empregados que não recebem igual a R$1900,00.

```
SELECT nome_empregado, salario
FROM empregado
WHERE salario != 1900;
```

```
NOME_EMPREGADO                                     SALARIO
-------------------------------------------------- -----------
Chico Bento                                        12652,88
Tio Patinhas                                       13000
Madame Mafalda                                     2552,78
Tião Gavião                                        3556
Pato Donald                                        1244,21
Pateta                                             1244,21
Patolino                                           2100,19
Saci Pererê                                        3100
Homer Simpson                                      980
Cebolinha                                          1300
Visconde de Sabugosa                               2950,23
Mickey Mouse                                       1110

 12 linhas selecionadas
```

Banco de Dados 1
Prof. Gilberto Assis
Roteiro de Aula

Operadores Lógicos

Muitas das consultas em banco de dados dependem de mais de uma condição de filtro e para esses casos, fazemos uso dos operadores lógicos AND (e) ou OR (ou), com eles, é possível escrever expressões poderosas de pesquisas que atendam aos diversos critérios de buscas necessários ao usuário do sistema.
Esses operadores obedecem as mesmas regras de **tabela verdade** que as demais linguagens de programação:

condição 1 AND condição 2 será verdadeira quando ambas forem verdade
condição 1 AND condição 2 será falsa quando uma delas ou as duas são falsas

condição 1 OR condição 2 será verdadeira quando uma delas ou as duas são verdadeiras
condição 1 OR condição 2 será falsa quando ambas forem falsas

O comando abaixo mostra aqueles funcionários que trabalha no setor de codigo 4 e tem o salario igual a 1110. As duas condições de filtro devem ser satisfeitas (verdade).

```
select nome_empregado, cod_setor, salario
from empregado
where cod_setor = 4 or salario = 1100;
```

```
NOME_EMPREGADO        COD_SETOR    SALARIO
--------------------- ---------- ----------
Mickey Mouse                  4       1110
```

já o comando abaixo mostra aqueles empregados que trabalham no setor 4 ou tem salario igual a 1110. Bastou que uma das condições fosse verdadeira para retornar o registro.

```
select nome_empregado, cod_setor, salario
from empregado
where cod_setor = 4 or salario = 1100;
```

```
NOME_EMPREGADO        COD_SETOR    SALARIO
--------------------- ---------- ----------
Pato Donald                   4 1244,21
Pateta                        4 1244,21
Mickey Mouse                  4       1110
```

Operador BETWEEN
Esse operador deve ser usado quando o filtro está relacionado a valores que se encontram entre dois valores extremos.

O comando abaixo mostra o nome dos empregados que tem valor de salário entre 1000 e 1900, você deve perceber que o filtro é inclusivo ou seja, os valores de extremo (980 e 1300) também são considerados.

```
SELECT nome_empregado, salario
FROM empregado
WHERE salario BETWEEN 980 AND 1300;
```

```
NOME_EMPREGADO          SALARIO
-------------------- ----------
Pato Donald            1244,21
Pateta                 1244,21
Homer Simpson              980
Cebolinha                 1300
Mickey Mouse              1110
```

perceba que esse comando equivale ao comando abaixo

```
SELECT nome_empregado, salario
FROM empregado
WHERE salario >= 980 AND salario <=1300;
```

Operador LIKE
Esse operador é um tipo especial de comparação de TEXTO, pois possibilita ao usuário utilizar coringas:
_ - coringa de uma posição

```
SELECT nome_dependente
FROM dependente
WHERE nome_dependente like 'Ma_';
```

O comando retorna todos os dependentes que começam com Ma e tem 3 letras, não importando qual é a terceira.

```
NOME_DEPENDENTE
----------------
Max
Mat
```

OBS: esse comando só retorna nomes com 3 letras independente se é MAIÚSCULA ou minúscula, pois o coringa _ é de uma posição.

% - coringa de várias posições

Já o comando abaixo retornará todos os dependentes que terminam com "nho" independente dos caracteres e de quantos vem antes.

```
SELECT nome_dependente
FROM dependente
WHERE nome_dependente like '%nho';
```

```
NOME_DEPENDENTE
---------------
Huguinho
Zezinho
Luizinho
Floquinho
```

Os coringas podem ser usados em qualquer posição do valor de comparação, bem como quantas vezes for necessário, inclusive em conjunto.

```
SELECT nome_dependente
FROM dependente
WHERE nome_dependente like '_e%o';
```

```
NOME_DEPENDENTE
---------------
Teobaldo
Zezinho
```

O comando anterior retornou todos os dependentes que tem a segunda letra igual a "e" e a última letra igual a "o".

Comparando DATAS

É muito recomendável que ao trabalharmos com data sempre façamos uso dos parâmetros de formatação. Isso se deve ao fato do banco de dados armazenar a data em formato de timestamp (numérico) para que a mesma possa ser convertida de acordo com o padrão de datas configurada pelo DBA. Logo um banco de dados pode apresentar a mesma data armazenada em formato diferente para usuários com configuração de ambiente diferentes.

Parâmetros de formatação de data

FORMATO	*DESCRIÇÃO*
YYYY ou RRRR	Ano completo em número
YEAR	Ano por extenso
MM	Mês em número
MON	Mês em letras - formato abreviado
MONTH	Mês em letras - formato completo
DD	Dia em número
DY	Dia da semana em letras – formato abreviado
DAY	Dia da semana em letras – formato completo

TO_CHAR – será usado para extrair a data formatada do banco de dados;
TO_DATE – será usado quando o usuário passará a data para o banco de dados;

O comando abaixo retorna a data de admissão do empregado de código 1 sem nenhuma formatação, ou seja, respeitando as configurações de ambiente do sistema.

```
select nome_empregado, to_char(admissao,'DD/MM/YYYY')
from empregado
where cod_empregado = 1;
```

```
NOME_EMPREGADO                 ADMISSAO
------------------------------ -------------------------
Chico Bento                    18/01/12
```

Já o comando abaixo "força" o banco a mostrar a data no formato extenso.

```
select nome_empregado, to_char(admissao,'DD MONTH YYYY')
from empregado
where cod_empregado = 1;
```

```
NOME_EMPREGADO                 TO_CHAR(ADMISSAO,'DDMONTHYYYY')
------------------------------ --------------------------------
Chico Bento                    18 JANEIRO   2012
```

O comando abaixo mostra a data no formato americano mês, dia e ano.

```
NOME_EMPREGADO               TO_CHAR(ADMISSAO,'MM/DD/YYYY')
---------------------------- ------------------------------
Chico Bento                  01/18/2012
```

Repare que o resultado do comando anterior a ordem de apresentação de dia e mês é diferente do padrão brasileiro, ou seja diferente do formato configurado para o banco de exemplo dessa aula. Certamente teríamos problemas de manipulação de datas caso o aplicativo de interação com o banco de dados suportasse apenas um dos dois padrões.

OBS: é MUITO recomendável a formatação da data para sua manipulação.
OBS2: A formatação da data NÃO altera o conteúdo do banco de dados.

Uma vez garantido que a data estará no formato desejado basta aplicar os comparadores normalmente.

Quais os empregados admitidos após o dia 01/04/2012?

```
SELECT nome_empregado, TO_CHAR(admissao,'DD/MM/YYYY')
FROM empregado
WHERE admissao > to_date('01/04/2012','DD/MM/YYYY')
ORDER BY 2;
```

```
NOME_EMPREGADO                 TO_CHAR(ADMISSAO,'DD/MM/YYYY')
------------------------------------------------------------
Pernalonga                     02/04/2012
Helena Pêra                    02/04/2012
Homer Simpson                  14/05/2012
Mickey Mouse                   18/06/2012
Cebolinha                      23/05/2012
Saci Pererê                    30/04/2012
Marge Simpson                  30/04/2012
Visconde de Sabugosa           30/05/2012
 8 linhas selecionadas
```

OBS: A funções de Data serão novamente abordadas a frente.

Operador NULL

O valor nulo (NULL) significa ausência de valor no registro em questão. Com relação ao NULL deve-se saber que não é possível compará-lo com nenhum outro valor, ou seja somente é possível saber se aquela coluna da tabela é nula ou não.

O comando

```
SELECT nome_dependente, dt_nascimento
FROM dependente
where dt_nascimento is null;
```

retorna aquele dependentes que tem a data de nascimento nula, ou seja sem valor.

```
NOME_DEPENDENTE                                    DT_NASCIMENTO
-------------------------------------------------- --------------
Curupira
Mula Sem Cabeça
```

Observe que o comando

```
SELECT nome_dependente, dt_nascimento
FROM dependente
where dt_nascimento = '';
```

retorna resultado diferente

```
NOME_DEPENDENTE                                   DT_NASCIMENTO
------------------------------------------------- --------------
```

OBS: NÃO é possível comparar NULL com nenhum valor, só é possível determinar se o campo é nulo (IS NULL) ou não nulo (IS NOT NULL).

USANDO FUNÇÕES

As funções são usadas dentro do comando de maneira natural e podem aparecer tanto na cláusula SELECT quanto na cláusula WHERE. Para efeito de estudo dividiremos as funções por tipo.

FUNÇÕES DE AGRUPAMENTO

São aquelas que agrupam os resultados em função de uma única coluna.

MAX(coluna) – essa função retorna o maior valor encontrado na coluna informada. Deve-se ter em mente que esse valor pode ser número, data ou mesmo texto.

```
SELECT MAX(salario)
FROM empregado;
```

Deve-se observar que a estrutura do comando não muda (select <o que será mostrado> from <de onde será mostrado>). A inclusão da função MAX determina o retorno do maior valor da coluna selecionada. Mesmo havendo mais de um maior valor na coluna o resultado será de uma linha apenas.

```
MAX(SALARIO)
-------------
13000
```

MIN(coluna) – é o comando oposto ao MAX(), pois retorna o menor valor encontrado na coluna. Deve-se ter em mente que esse valor pode ser número, data ou mesmo texto.

```
SELECT MIN(nome_dependente)
FROM dependente;
```

Observe no comando acima que a função MIN() foi usada em uma coluna de texto e retornou o primeiro nome do dependente ordenado alfabeticamente.

```
MIN(NOME_DEPENDENTE)
--------------------
Bart Simpson

1 rows selected
```

COUNT(coluna) – função que conta a quantidade de registros retornados pelo comando SELECT.

```
SELECT COUNT(nome_dependente)
FROM dependente;
```

O resultado do comando acima será a quantidade de dependentes cadastrados.

```
COUNT(NOME_DEPENDENTE)
----------------------
19
```

AVG(coluna) – retorna a média aritmética de uma determinada coluna. Válida exclusivamente para colunas numéricas.

```
SELECT AVG(salario)
FROM empregado;
```

```
AVG(SALARIO)
--------------
3432,7
```

O comando anterior retornou a média da idade dos empregados com uma precisão de 36 casas decimais. Entretanto, isto pode ser mudado com as funções ROUND() e TRUNC().

ROUND() – arredonda, para baixo ou para cima, um determinado número. Nesta função deve-se indicar qual valor será arredondado e a precisão do arredondamento.

```
SELECT ROUND(AVG(salario), 0)
FROM empregado;
```

Nesse comando deve-se observar o uso da função ROUND() arredondando o valor da média de idade dos empregados para 0 casas decimais. Deve-se reparar que o arredondamento foi para cima.

```
ROUND(AVG(SALARIO),0)
---------------------
3433
```

TRUNC() – esse comando simplesmente trunca, ou seja, mostra apenas o número de casas decimais informados sem se preocupar com arredondamentos.

```
SELECT TRUNC(AVG(salario), 0)
FROM empregado;
```

```
TRUNC(AVG(SALARIO),0)
---------------------
3432
```

Observe que o resultado não foi arredondado.

RELACIONANDO TABELAS

Uma das principais vantagens de manter seus dados em um banco de dados relacional é a possibilidade de relacioná-los, ou seja, você poderá organizar seus dados em tabelas distintas e específicas para evitar problemas de redundâncias e integridade e depois poderá verificar a relação entre esses mesmos dados.

EquiJoin – esse relacionamento entre tabelas ocorre pela reunião de campos iguais de tabelas distintas, ou seja, o Oracle pesquisará por valores iguais nos campos das tabelas indicadas.

A relação EquiJoin pode ser escrita de duas maneiras:
SQL-86 – usa-se o símbolo de igualdade = para indicar as tabelas e os respectivos campos que serão relacionados.

```
SELECT s.cod_setor, nome_setor, e.cod_setor, nome_empregado
FROM setor s, empregado e
WHERE s.cod_setor = e.cod_setor;
```

O comando acima retornará a relação de setores e respectivos empregados, deve-se perceber que a estrutura do comando NÃO mudou, continuamos com SELECT <colunas> FROM <tabelas> WHERE <condição>. O que mudou foi o uso de mais de uma tabela na cláusula FROM e uma nova função para a cláusula WHERE que passa a exercer a função de relacionamento.
Pode-se dizer que o comando fará o Oracle retornar os registros das tabelas que obedecem a relação solicitada, isso quer dizer que serão retornados os registros que tem valores (cod_setor) em comum nas duas tabelas.
Outro detalhe importante do comando acima está no uso dos apelidos/alias de tabela "s" para SETOR e "e" para a tabela EMPREGADO, esse apelido foi muito útil na hora de identificar de qual tabela o campo cod_setor pertence.
A indicação da origem do campo cod_setor é obrigatória para evitar a ambigüidade do nome da coluna (Erro SQL: ORA-00918: coluna definida de maneira ambígua) que acontece em razão do banco não saber de onde buscar a informação.

COD_SETOR	NOME_SETOR	COD_SETOR	NOME_EMPREGADO
1	Diretoria	1	Chico Bento
1	Diretoria	1	Tio Patinhas
3	Secretária	3	Madame Mafalda
2	Recursos Humanos	2	Tião Gavião
4	Produção de Sólidos	4	Pato Donald
4	Produção de Sólidos	4	Pateta
5	Produção de Líquidos	5	Patolino
7	Comercial	7	Saci Pererê
5	Produção de Líquidos	5	Pernalonga
5	Produção de Líquidos	5	Helena Pêra
5	Produção de Líquidos	5	Marge Simpson
6	Estoque	6	Homer Simpson
6	Estoque	6	Cebolinha
7	Comercial	7	Visconde de Sabugosa
4	Produção de Sólidos	4	Mickey Mouse

15 linhas selecionadas

Perceba que o COD_SETOR da tabela SETOR é igual ao COD_SETOR da tabela EMPREGADO.

SQL-92 – usa a expressão de junção por extenso, na verdade o resultado em nada difere do anterior.

```
SELECT nome_setor, nome_empregado
FROM setor s INNER JOIN empregado e on(s.cod_setor = e.cod_setor);
```

Do comando anterior deve-ser verificar o uso da expressão de relacionamento INNER JOIN e a identificação da coluna de junção USING(). Embora esse comando retorne o mesmo relacionamento do anterior, ele tem algumas desvantagens como a proibição da exibição do campo de junção (colocar o campo cod_setor na cláusula SELECT), outra desvantagem está na obrigatoriedade dos nomes dos campos que se relacionam serem iguais.

AGRUPANDO OS DADOS
Depois de aprender as funções de agrupamento, estaremos aptos a usar a função GROUP BY que consiste em agrupar o resultado da pesquisa de acordo com a função agrupadora usada no próprio comando. Basicamente para usar o GROUP BY deve-se destacar as demais colunas do comando SELECT.

```
SELECT nome_setor, COUNT(e.cod_empregado)
FROM setor s, empregado e
WHERE s.cod_setor=e.cod_setor
GROUP BY nome_setor;
```

Deve-se observar no comando acima que o GROUP BY vem depois da cláusula WHERE e contém as colunas que não fazem parte da função agrupadora. O resultado retornado será o número de funcionários agrupados por setor.

```
NOME_SETOR                                         COUNT(E.COD_EMPREGADO)
-------------------------------------------------- ----------------------
Comercial                                          2
Produção de Sólidos                                3
Estoque                                            2
Produção de Líquidos                               4
Diretoria                                          2
Recursos Humanos                                   1
Secretária                                         1

 7 linhas selecionadas
```

É possível também filtrar o resultado do GROUP BY através da cláusula HAVING, que terá a mesma função do WHERE, ou seja, limitar o resultados de acordo com a condição solicitada. Para sabermos a média de idade dos funcionários de cada setor, mas somente aqueles setores cujo a média de salário seja menor que R$2000,00.

```
SELECT nome_setor, TRUNC(AVG(idade_empregado),2)
FROM setor s, empregado e
WHERE s.cod_setor= e.cod_setor
GROUP BY nome_setor
HAVING AVG(salario) < 2000;
```

Repare que no comando acima a cláusula HAVING vem logo após o GROUP BY. Outro destaque é a cláusula TRUNC usada para mostrar até duas casas decimais.

FUNÇÕES DE DATA E HORA

Os próximos comandos são essenciais para o usuário do banco, pois garante a independência do formato de data usado no computador, isso porque o sistema operacional garante ao usuário do computador escolher qual padrão de data e hora usado. Por exemplo, no padrão brasileiro o formato de data é DD/MM/YYYY (23/09/2005), já o americano o padrão é MM/DD/YYYY (09/23/2005), repare que o dia e o mês estão trocados. Então, qual padrão seguir? Como o DBA deverá configurar o banco de dados? Na verdade, ele não precisará se preocupar com isso, pois o banco armazena a data e hora no formato 'YYYY-MON-DD HH24:MI:SS', entretanto caberá ao usuário do banco de dados especificar em que formato ele quer receber a informação. Naturalmente o banco de dados ORACLE contém uma série de funções para tratamento de DATA e HORA que serão estudadas aqui.

TO_CHAR(<data>, <formato>) – essa função retornará a data no formato desejado.
A tabela abaixo contém os principais parâmetros de formatação de data:

YY - Ano com 02 dígitos. Ex.: 98
YYYY - Ano com 04 dígitos. Ex.: 1998
NM - O número do mês.
MONTH - O nome completo do mês.
MON - O nome abreviado do mês. Ex.: Jan, Feb
DDD - O dia do ano.
DD - O dia do mês.
D - O dia da semana.
DAY - O nome do dia.
HH - A hora do dia, no formato de 12 horas.
HH24 - A hora do dia, no formato de 24 horas.
MI - Os minutos.
SS - Os segundos.

Para sabermos o dia de hoje usaremos o comando abaixo:

```
SELECT SYSDATE FROM dual;
```

Sobre o comando acima devemos destacar a função SYSDATE que retorna a data e hora do sistema e também a tabela dual que é uma tabela auxiliar do ORACLE usada apenas para retornar o resultado de funções.

```
SYSDATE
-------------------------
05-out-2010 19:54:37

1 rows selected
```

Entretanto podemos formatar a data para o nosso padrão de dia/mês/ano.

SELECT TO_CHAR(SYSDATE, 'DD/MM/YYYY') FROM dual;

```
TO_CHAR(SYSDATE,'DD/MM/YYYY')
-----------------------------
05/10/2010

1 rows selected
```

Essa função também pode ser usada na coluna da tabela, observe:

SELECT nome_empregado, TO_CHAR(admissao, 'DD/MM/YYYY')
FROM empregado;

```
NOME_EMPREGADO                                     TO_CHAR(ADMISSAO,'DD/MM/YYYY')
-------------------------------------------------- ------------------------------
Chico Bento                                        18/01/2012
Tio Patinhas                                       18/01/2012
Madame Mafalda                                     19/01/2012
Tião Gavião                                        19/01/2012
Pato Donald                                        19/01/2012
Pateta                                             19/01/2012
Patolino                                           19/01/2012
Saci Pererê                                        30/04/2012
Pernalonga                                         02/04/2012
Helena Pêra                                        02/04/2012
Marge Simpson                                      30/04/2012
Homer Simpson                                      14/05/2012
Cebolinha                                          23/05/2012
Visconde de Sabugosa                               30/05/2012
Mickey Mouse                                       18/06/2012

 15 linhas selecionadas
```

TO_DATE(<texto>, <formato>) – é a função oposta a anterior, ou seja, será usada para transformar um texto em uma data, isso será usado para comparar datas de uma determinada tabela com a escrita pelo usuário, ou ainda durante a inserção de valores.

SELECT nome_empregado, admissao
FROM empregado
WHERE admissao = to_date('02/04/2012', 'DD/MM/YYYY');

O comando acima retorna os funcionários e respectivas data de admissão cujo a data de admissão ocorreu em 02 de abril de 2012.

```
NOME_EMPREGADO                                     ADMISSAO
-------------------------------------------------- -----------
Pernalonga                                         02/04/12
Helena Pêra                                        02/04/12
```

FUNÇÕES DE CARACTERES

O ORACLE também nos permite trabalhar a formatação dos caracteres devolvidos por ele, isso será muito útil por questão de estética, mas também para garantir que as comparações sejam realizadas independente de como o texto/caractere foi inserido no banco.

LOWER()- essa função transforma todo seu conteúdo em caracteres minúsculos.

```
SELECT LOWER(nome_setor)
FROM setor;
```

No comando acima será retornado o nome dos setores em letras minúsculas, independente de como elas foram inseridas.

```
LOWER(DESC_SETOR)
-----------------------
diretoria
recursos humanos
secretária
produção de sólidos
produção de liquidos
estoque
comercial

7 rows selected
```

UPPER() – função oposta ao LOWER, ou seja, retorna todos os caracteres em maiúsculo.

```
SELECT UPPER(nome_setor)
from setor;
```

```
UPPER(DESC_SETOR)
---------------------
DIRETORIA
RECURSOS HUMANOS
SECRETÁRIA
PRODUÇÃO DE SÓLIDOS
PRODUÇÃO DE LIQUIDOS
ESTOQUE
COMERCIAL

7 rows selected
```

Essas funções serão úteis nas comparações de valores, pois garantirá a comparação independente da maneira que os valores foram inseridos no banco.

```
SELECT nome_setor
FROM setor
WHERE UPPER(nome_setor) LIKE UPPER('SEC%');
```

Deve-se perceber no comando acima o uso da função UPPER na cláusula WHERE. Já a resposta retornou conforme foi inserida no banco, pois na cláusula SELECT não foi usada nenhuma função.

```
DESC_SETOR
--------------------
Secretária

1 rows selected
```

INITCAP() – retorna a primeira letra de cada palavra em maiúsculo.

```
SELECT INITCAP(nome_empregado)
FROM empregado;
```

CONCATENAÇÃO DE PALAVRAS – o Oracle fornece duas funções de concatenação.
CONCAT('texto1', 'texto2') – essa função concatenará somente dois textos;

```
SELECT nome_empregado, CONCAT('R$ ', salario)
FROM empregado;
```

No comando acima a função CONCAT é usada para juntar o texto 'R$ ' com os registros da coluna SALARIO.

```
NOME_EMPREGADO                                     CONCAT('R$',SALARIO)
-------------------------------------------------- --------------------
Chico Bento                                        R$ 12652,88
Tio Patinhas                                       R$ 13000
Madame Mafalda                                     R$ 2552,78
Tião Gavião                                        R$ 3556
Pato Donald                                        R$ 1244,21
Pateta                                             R$ 1244,21
Patolino                                           R$ 2100,19
Saci Pererê                                        R$ 3100
Pernalonga                                         R$ 1900
Helena Pêra                                        R$ 1900
Marge Simpson                                      R$ 1900
Homer Simpson                                      R$ 980
Cebolinha                                          R$ 1300
Visconde de Sabugosa                               R$ 2950,23
Mickey Mouse                                       R$ 1110

 15 linhas selecionadas
```

O comando || é usado para concatenar vários textos, bastando acrescentá-lo entre as palavras.

```
SELECT nome_dependente||' é dependente de '||nome_empregado
FROM dependente d, empregado e
WHERE e.cod_empregado = d.cod_empregado
ORDER BY 1;
```

Sobre o comando acima deve-se perceber os espaços deixados entre as palavras.

```
NOME_DEPENDENTE||'ÉDEPENDENTEDE'||NOME_EMPREGADO
--------------------------------------------------
Bart Simpson é dependente de Homer Simpson
Curupira é dependente de Visconde de Sabugosa
Flecha Roberto Pêra é dependente de Helena Pêra
Floquinho é dependente de Cebolinha
Giselda é dependente de Chico Bento
Huguinho é dependente de Pato Donald
Lisa Simpson é dependente de Homer Simpson
Luizinho é dependente de Pato Donald
Maggie Simpson é dependente de Homer Simpson
Margarida é dependente de Pato Donald
Mat é dependente de Cebolinha
Max é dependente de Pateta
Mula Sem Cabeça é dependente de Visconde de Sabugosa
Roberto "Beto" Pêra é dependente de Helena Pêra
Rosinha é dependente de Chico Bento
Teobaldo é dependente de Chico Bento
Violeta Pêra é dependente de Helena Pêra
Zezé Pêra é dependente de Helena Pêra
Zezinho é dependente de Pato Donald
 19 linhas selecionadas
```