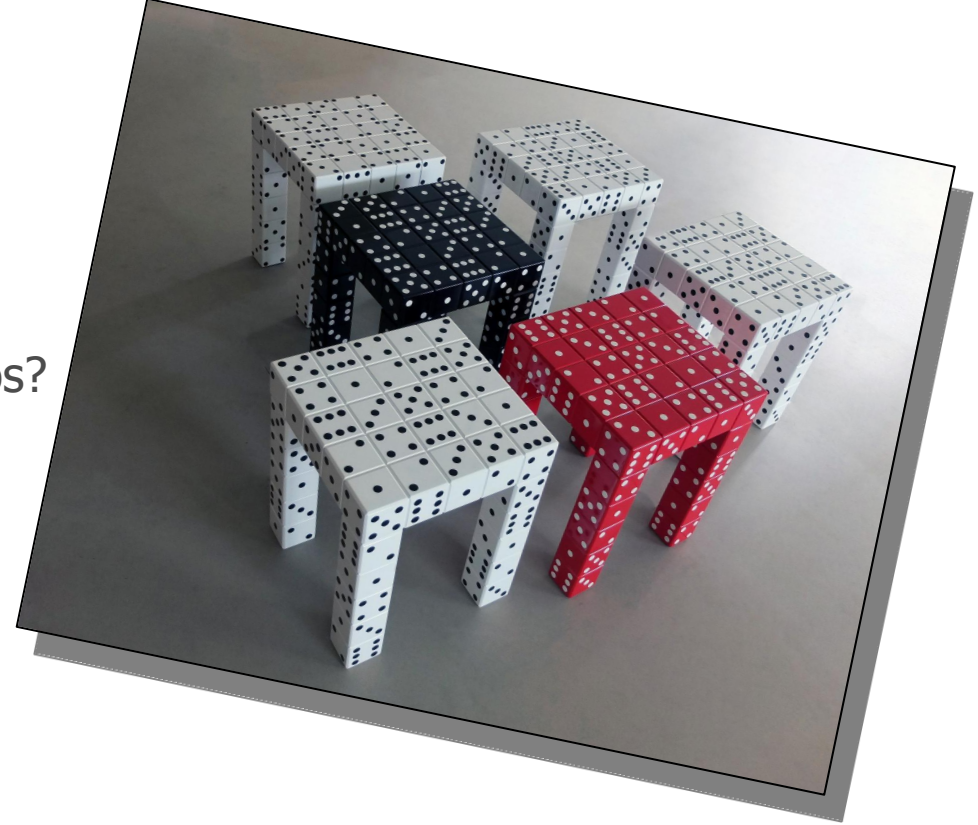
# Curso de Férias
# SQL e PL/SQL Básico

MATERA systems

1. Introdução a banco de dados.
2. Comandos DDL
3. Comandos DML.
4. Comandos DML II.
5. PLSQL.
6. Procedures e Functions.
7. SubQuerys.
8. Mais Objetos Oracle.
9. Packages.

# Introdução a banco de dados.

- O que é um banco de dados?
- Porque precisamos de um banco de dados?
- O que pode ser um banco de dados?





MATERA systems

# Introdução a banco de dados.

- O que é um SGBD?
- Tipos de bancos de dados.
- O que é SQL (Structured Query Language).








MATERA systems

# Comandos DDL

São comandos utilizados para definirem as estruturas de dados, como as tabelas que compõem  um banco de dados, os cinco tipos básicos de instruções DDL são:

- CREATE: cria uma estrutura de banco de dados. Por exemplo, CREATE TABLE é usada para criar uma tabela; outro exemplo é CREATE USER, usada para criar um usuário do banco de dados.

- ALTER: modifica uma estrutura de banco de dados. Por exemplo, ALTER TABLE é usada para modificar uma tabela.

- DROP: remove uma estrutura de banco de dados. Por exemplo, DROP TABLE é usada para remover uma tabela.

# Comandos DDL

Tipos de dados.

Cada valor manipulado pelo Oracle Database possui um tipo de dados.

| Tipo | Descrição |
|---|---|
| VARCHAR2(comprimento_máximo) | Carácter de tamanho variável, podendo atingir o tamanho máximo de até 32767 bytes. |
| NUMBER [precisão, escala] | Tipo numérico fixo e de ponto flutuante. |
| DATE | Tipo para acomodar data e hora |

# Comandos DDL

- Criação de tabelas

Os dados são armazenados em estruturas chamadas tabelas, abaixo é apresentada a composição do comando create table.

```
Create table time
(
        id_time     number          not null,
        nome        varchar2(400)   not null
);
```

# Comandos DDL

- Constraints

Constraints são objetos fundamentais para a escalabilidade, flexibilidade e integridade dos dados armazenados em um banco de dados. Elas aplicam regras específicas para os dados, garantem que os dados estejam em conformidade com os requisitos definidos. Existem alguns tipos de constraints no Oracle, a seguir elas são apresentadas:

Primary key: Cada tabela pode ter, no máximo, uma constraint de primary key (em português chave primária). A primary key pode ter mais que uma coluna da tabela. A constraint de primary key força que cada chave primária só pode ter um valor único, impondo em simultâneo a constraint unique e NOT NULL. Uma primary key vai criar um indice único, caso ainda não exista para a coluna em causa.
Foreign Key: A foreign key (em português chave estrangeira) é definida para uma tabela (conhecida como filha) que tem um relacionamento com outra tabela (conhecida como pai). O valor guardado na foreign key deverá ser o mesmo presente na primary key respectiva.

```
alter table TIME add constraint pk_time primary key (ID_TIME);
alter table jogador add constraint fk_time foreign key (id_time) references time(id_time);
```

# Comandos DDL

- Comentários

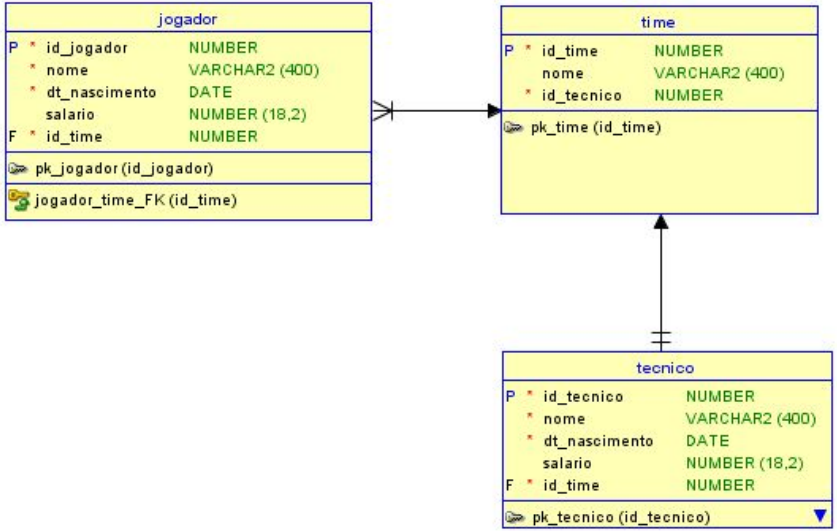
Ao criar uma tabela, é possível definir comentários para a tabela e colunas, isso auxilia no entendimento do objetivo da tabela e colunas.

```
comment on table TIME  is '[Cadastro] Tabela para armazenamento de times.';


comment on column TIME.id_time  is 'Código Identificador do time.';
comment on column TIME.nome  is 'Nome do Time.';
```

# Comandos DDL

- Exercícios:
  - Criar as tabelas time, técnico e jogador;
  - Definir constraints para as tabelas;
  - Criar comentários para as tabelas e as colunas;






MATERA systems

# Comandos DML

**Data Manipulation Language** ( Linguagem de manipulação de dados) são utilizados para o gerenciamento de dados dentro de objetos do banco.

- Manipulações como Inserção, alteração e deleção de registros.




MATERA systems

# Comandos DML

A instrução **SELECT** é utilizada para recuperar os dados do banco de dados.

```
Select      id_time, nome
from        time
where       nome = 'BARCELONA'
order by nome;
```

# Comandos DML

- Comando ***COMMIT*** é um comando utilizado no controle transacional, faz com que o dado inserido, alterado ou removido seja realmente persistido(salvo) no banco de dados.

```
insert into time (id_time, nome) values (1,'BARCELONA');
--
commit;
--
select nome
from   time;
```

# Comandos DML

- O **ROLLBACK** é um comando utilizado também no controle transacional, ele desfaz as alterações de dados realizadas desde o início da Rotina, Checkpoint(savepoint) ou último *COMMIT*.

```
delete time;
--
rollback;
--
select nome
from   time;
```

# Comandos DML

A instrução INSERT é utilizada para inserir dados no banco de dados.

```
insert into time (id_time, nome)
values (1,'BARCELONA');
```

```
insert into time
values (1,'BARCELONA');
```

# Comandos DML

**SEQUENCE**.

Sequences é um tipo de objeto utilizado para incrementar valores. A Oracle disponibiliza este objeto para facilitar o controle dos valores já que, é controlado pelo db e você só precisa consumi-lo.

```
create sequence seq_jogador
minvalue 1
maxvalue 999999999999999999999999999
start with 1
increment by 1
nocache;
```

*CACHE n / NOCACHE* :
especifica quantos valores são pré-alocados pelo Oracle Server e mantidos na memória (por padrão, o Oracle Server armazena 20 valores em cache).



MATERA systems

# Comandos DML

**SEQUENCE**.
Este objeto possui duas funções expostas já pré-definidas.

**nextval** que incrementa e retorna o próximo valor.
**currval** que retorna o último valor gerado.

```
insert into time (id_time, nome)
values (seq_jogador.nextval,'BARCELONA');
```

```
select seq_jogador.currval
from dual;
```

# Comandos DML

- Exercícios:
  - Inserir 2 times.
  - Inserir 2 técnicos.
  - Inserir 11 jogadores em um time.
  - Listar todos jogadores de um determinado time.
  - Listar todos times.
  - Listar técnicos com mais de 40 anos.
  - Inserir os jogadores existentes para o outro time (select insert com sequence).

# Comandos DML

A instrução UPDATE é utilizada para alterar dados já existentes no banco de dados.

```
update time
set nome = 'BARCELONA FUTEBOL'
where id_time = 1;
```

```
update time
set nome = 'BARCELONA FUTEBOL ALTERADO'
where nome = 'BARCELONA FUTEBOL';
```

# Comandos DML

A instrução UPDATE é utilizada para alterar dados já existentes no banco de dados.

```
update time
set nome = 'BARCELONA' || 'FUTEBOL'
where id_time = 1;
```

```
update time
set nome = nome || 'FUTEBOL'
where id_time = 1;
```

# Comandos DML

A instrução UPDATE é utilizada para alterar dados já existentes no banco de dados.

```
update time
set nome = 'BARCELONA' ,
segundo_nome = 'SEGUNDO NOME DO TIME'
where id_time = 1;
```

# Comandos DML

- Exercícios:
  - Inserir um time novo.
  - Alterar todos jogadores de um time para o novo time.
  - Aumentar em 10% o salário de todos jogadores do novo time.
  - Aumentar o salário de todos técnicos em 20%.

# Comandos DML

A instrução **DELETE** é utilizada para remover dados no banco de dados.

```
delete time
where id_time = 1;
```

```
delete jogador
where salario>= 100000;
```

# Comandos DML

- Exercícios:
  - Inserir um time novo.
  - Inserir 3 jogadores extras no time novo.
  - Alterar o salário de 3 jogadores para valores acima de R$ 100.000,00.
  - Remover jogadores do novo time com salários superiores R$ 100.000,00.
  - Remover times que estejam sem jogadores e técnicos.

# Comandos DML II

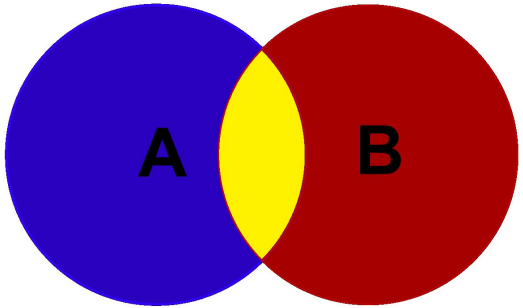
- Junções de Dados e Apelidos
  - INNER JOIN
  - LEFT JOIN
  - RIGHT JOIN

# Comandos DML II | Inner Join

**Quando queremos juntar duas ou mais tabelas, que internamente, tenham valores correspondentes (parte amarela).**

SQL INNER JOIN

```
SELECT JOG.NOME NOME,
       EQU.NOME AS NOME_DA_EQUIPE
FROM   JOGADOR JOG INNER JOIN EQUIPE EQU
                   ON JOG.ID_EQUIPE = EQU.ID_EQUIPE;
```





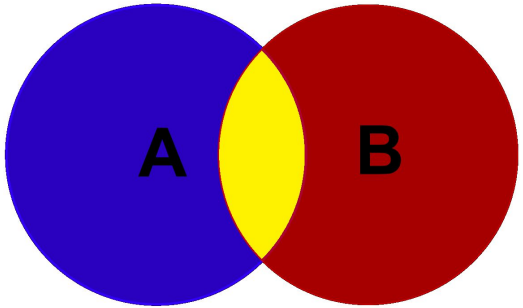

MATERA systems

# Comandos DML II | Inner Join

**Quando queremos juntar duas ou mais tabelas, que internamente, tenham valores correspondentes (parte amarela).**

SQL JOIN

```
SELECT JOG.NOME NOME,
       EQU.NOME AS NOME_DA_EQUIPE
FROM   JOGADOR JOG JOIN EQUIPE EQU
                   ON JOG.ID_EQUIPE = EQU.ID_EQUIPE;
```





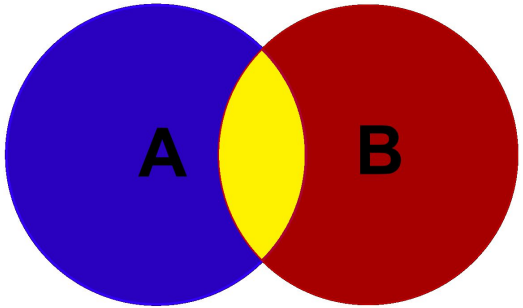

MATERA systems

# Comandos DML II | Inner Join

**Quando queremos juntar duas ou mais tabelas, que internamente, tenham valores correspondentes (parte amarela).**

PL/SQL JOIN

```
SELECT JOG.NOME NOME,
       EQU.NOME AS NOME_DA_EQUIPE
FROM   JOGADOR JOG,  EQUIPE    EQU
WHERE JOG.ID_EQUIPE = EQU.ID_EQUIPE;
```





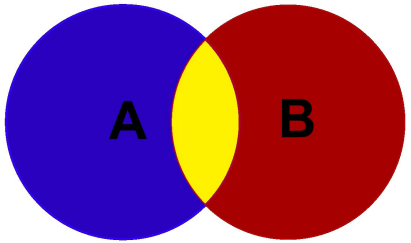

MATERA systems

# Comandos DML II | Left Join

- **É utilizado para selecionar todos os itens de uma tabela A com uma tabela B mesmo que A não esteja relacionado com a tabela B (Parte Azul).**

```
SELECT JOG.NOME NOME, EQU.NOME AS NOME_DA_EQUIPE
FROM  JOGADOR JOG LEFT JOIN EQUIPE EQU
                          ON JOG.ID_EQUIPE = EQU.ID_EQUIPE;
```

```
SELECT JOG.NOME    NOME,
          EQU.NOME AS NOME_DA_EQUIPE
FROM   JOGADOR JOG, EQUIPE    EQU
WHERE JOG.ID_EQUIPE = EQU.ID_EQUIPE(+);
```





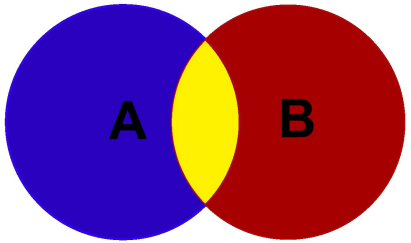

MATERA systems

# Comandos DML II | Right Join

**Funciona como o left outer join, mas ao contrário. (Parte Vermelha).**

```
SELECT JOG.NOME NOME, EQU.NOME AS NOME_DA_EQUIPE
FROM  JOGADOR JOG RIGHT JOIN EQUIPE EQU
                                ON JOG.ID_EQUIPE = EQU.ID_EQUIPE;
```

```
SELECT JOG.NOME    NOME,
          EQU.NOME AS NOME_DA_EQUIPE
FROM   JOGADOR JOG, EQUIPE    EQU
WHERE JOG.ID_EQUIPE(+) = EQU.ID_EQUIPE;
```






MATERA systems

# Comandos DML II | Ordenações

Utilizamos as ordenações para ordenar os resultados de uma consulta.
Podemos ordenar por ordem crescente(*asc*) ou decrescente(***desc***).

```
SELECT NOME, DT_NASCIMENTO AS DATA_NASC
FROM JOGADOR
ORDER BY NOME ASC
```

```
SELECT NOME, DT_NASCIMENTO AS DATA_NASC
FROM JOGADOR
ORDER BY NOME, DT_NASCIMENTO ASC
```

# Comandos DML II | Ordenações

Utilizamos as ordenações para ordenar os resultados de uma consulta.
Podemos ordenar por ordem crescente(*asc*) ou decrescente(***desc***).

```
SELECT NOME, DT_NASCIMENTO AS DATA_NASC
FROM JOGADOR
ORDER BY 2, 1 ASC
```

```
SELECT NOME, DT_NASCIMENTO AS DATA_NASC
FROM JOGADOR
ORDER BY DATA_NASC ASC
```

# Comandos DML II | Ordenações

- Exercícios:
  - Selecione os Times em ordem crescente.
  - Selecione os nomes de jogadores e seus respectivos nomes dos times ordenado(asc) pela data de nascimento dos jogadores.

# Comandos DML II | Agrupamentos de Dados

Utilizamos agrupamento para juntar os dados equivalentes com a palavra group by.

- Com a utilização e grupos podemos utilizar as funções de agregação, que permitem realizar cálculos sobre o resultado da consulta retornada.
- Todas colunas selecionadas que não estão sendo utilizadas em algum tipo de função de agregação deverão estar declaradas no “group by”.

```
SELECT T.NOME
FROM   JOGADOR J, TIME T
WHERE  T.ID_TIME = J.TIME_ID_TIME
GROUP  BY T.NOME
```

# Comandos DML II | Funções de Agregação

As **Funções de Agregação** são utilizadas para manipular os dados agrupados.

COUNT → Conta o número de linhas afetadas pelo comando.
SUM → Calcula o somatório do valor das colunas especificadas.
AVG → Calcula a média aritmética dos valores das colunas.
MIN → Seleciona o menor valor da coluna de um grupo de linhas.
MAX → Seleciona o maior valor da coluna de um grupo de linhas.

```
SELECT COUNT(*), T.NOME
FROM   JOGADOR AS J,
       TIME    AS T
WHERE  T.ID_TIME = J.TIME_ID_TIME
GROUP  BY T.NOME
```

# Comandos DML II | Funções de Agregação

- Exercícios:
  - Gere uma consulta retornando a folha de pagamento de cada equipe.
  - Gere uma consulta retornando a média salarial de cada equipe.
  - Gere uma consulta que retorne o menor salário de cada equipe.
  - Gere uma consulta que retorne o maior salário de cada equipe.

# PLSQL

- O que é PL/SQL ?
- As vantagens do PL/SQL
- Diferenças da Sintaxe SQL e PLSql

# PLSQL | Blocos Anônimos

- **Um bloco PL/SQL que é utilizado para realizar alguma lógica que não é definido ou nomeado como procedure, function, trigger ou outro objeto nativo do Oracle é comumente chamado de Bloco Anônimo.**

```
1  declare
2    -- Local variables here
3    i integer;
4  begin
5    -- Test statements here
6  end;
```

# PLSQL | Blocos Anônimos

- Composição de um bloco Anônimo.
  - Declarativa (opcional).
  - Executável (obrigatória).
  - Manipulação de Exceções e Erros (opcional).
  - Finalização do bloco.

```
declare
  --
  vnNumber number;
  --
begin
  --
  null;
  --
exception
  when other then
    --
    dbms_output.put_line('error');
    --
end;
```

# PLSQL | Blocos Anonimos

- Exercícios:
  - Criar Blocos Anônimos que:
    - Gere uma saída DBMS básica (‘Hello DBMS’) utilizando o pacote da Oracle DBMS_OUTPUT.PUT_LINE(‘text’).
    - Gerar uma saída DBMS contendo as informações de um time e do seu técnico.
    - Gerar um jogo composto de dois times diferentes e escalar os jogadores participantes.
    - Marcar alguns gols para o jogo gerado respeitando o placar definido no jogo.




MATERA systems

# PLSQL | Comentários

```
--line comments.

/* block comments */

/*

  block comments
  here

*/
```

# PLSQL | Comentários

- Exercícios:
  - Incluir comentários nos blocos anônimos anteriores sem alterar o funcionamento.

# PLSQL | Desvios Condicionais

```
IF (condição) THEN

   /* comandos aqui */

END IF;
```

```
IF (condição) THEN
   /* comandos aqui */
ELSE
   /* comandos aqui */
END IF;
```

# PLSQL | Desvios Condicionais

```
IF (condição) THEN
  /* comandos aqui */
ELSIF (condição2) THEN
   /* comandos aqui */
ELSIF (condição3) THEN
  /* comandos aqui */
ELSE
   /* comandos aqui */
END IF;
```

# PLSQL | Desvios Condicionais

```
declare
  vnNumero  number(1) := 1;
  vNRetorno number;

begin

  vNRetorno := case
                 when vnNumero = 1 then
                   11
                 when vnNumero = 2 then
                   22
                 else
                   33
               end;

  dbms_output.put_line(vNRetorno);

end;
```

# PLSQL | Desvios Condicionais

```
select decode(nome, null,
                    'Técnico sem nome',
                    'chuck norris',
                    ' !!!!!! ',
                    nome)
from tecnico;
```

# PLSQL | Desvios Condicionais

- Exercícios:
  - Altera bloco anônimo que gerar o jogo composto de dois times diferentes e adicionar quantos “if” forem necessários para que não permita inserir um jogo sendo o mesmo time para “ambos os lados”.

  - Criar uma consulta que retorne os jogos, times que estão participando, e placar, sendo que na coluna placar deve trazer o número de gols de cada time respeitando o seu lado na ordenação das colunas (4 - 3) e caso seja o mesmo número de gols deve aparecer ‘empate’.

# PLSQL | Exceptions

Quando um bloco **PL/SQL** é executado ele tem um fluxo de vida, com execução TOP-DOWN porém com desvios de fluxos,
durante a execução pode ocorrer algum erro ou anormalidade.. então uma exceção é levantada (raise exception) e o fluxo normal do programa é desviado para o parágrafo declarado na área de *EXCEPTION ( caso tenha sido declarado)*.

Dentro da tratativa EXCEPTION, podemos tratar a exceção da melhor maneira possível.

- Caso uma exceção não seja "tratada" a execução da rotina ou procedimento termina em falha abortando todo processo!

# PLSQL | Exceptions

## Exceptions mais utilizadas:

- **OTHERS**, qualquer erro disparado pode ser tratado por este.
- **TOO_MANY_ROWS**, quando uma consulta retorna mais uma linha (erro cartesiano).
- **NO_DATA_FOUND**, quando uma consulta não retorna nenhuma informação.

```
BEGIN

  --codigoQuePodeHaverExcessoes;

EXCEPTION
  WHEN OTHERS THEN

    DBMS_OUTPUT.PUT_LINE('Erro ao executar operacoes. Erro :' || SQLERRM);

END;
```

# PLSQL | Exceptions

- Exercícios:
  - Criar bloco anônimo com algum comando dml “insert” que irá causar erro, e tratar com a exception OTHERS.
  - Criar bloco anônimo com alguma consulta que irá obter um valor e popular em uma variável e “forçar” um erro, e tratar com a exception TOO_MANY_ROWS.

# PLSQL | Cursores/Loops

Loops comumente são laços de repetições sendo a base de conhecimento para o entendimento de manipulação de cursores.

```
Loop
  --
  dbms_output.put_line(vnContador);
  --
  vnContador := vnContador + 1;
  --
  exit when vnContador = 20;
  --
end loop;
```

# PLSQL | Cursores

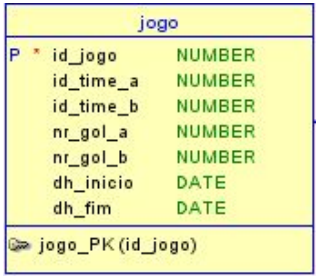
```
while vnContador < 10 loop
  --
  dbms_output.put_line(vnContador);
  --
  vnContador := vnContador + 1;
  --
end loop;
```

# PLSQL | Cursores

```
Declare
  --
  cursor vcCursor is
    select *
    from time;
    --
Begin
  --
  For meuCursor in vcCursor loop
    --
    dbms_output.put_line(meuCursor.nome);
    --
  end loop;
  --
end;
```

# PLSQL | Cursores

- Exercícios:
  - Criar um laço de repetição que imprima os números de 0 a 100 via DBMS.
  - Criar um laço de repetição que imprima os números de 0 a 100 **pares** via DBMS.
  - Criar uma tabela de jogo e uma que marque os os gols do jogo.
  - Percorrer todos os jogos e inserir um gol para cada jogador.






MATERA systems