BASES DE DADOS AVANÇADAS

Luis Pedro Ribeiro {luispribeiro9@gmail.com}
Miguel Valério {migu3lval3riol@gmail.com}
José Reigado{jreigado@gmail.com}
Carlos Matos{ccdematos@hotmail.com}

Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda

Com este trabalho pretende-se a implementação de uma base de dados backend para uma plataforma de gestão de tarefas (neste caso tarefas de limpeza). Além do registo das tarefas diárias de limpeza, esta base de dados terá funcionalidades como gerar as tarefas para o dia atual, otimizando a gestão dos recursos humanos, duração de tarefas, turnos dos funcionários e zonas das tarefas.

Índice

1. ER NORMALIZADO E DESNORMALIZADO

Este trabalho tem como base o modelo Entidade-Relacionamento (ER). Este modelo é constituído por Entidades, Atributos e Relacionamentos. A **Erro! A origem da referência não foi encontrada.** representa o modelo ER para o problema.

O nosso ER está na 2ª forma de normalização. Possui as características da 1ª forma " *(possui chave primária, cada coluna é atómica e não há grupos repetidos de colunas) e cada coluna não chave só é dependente da chave primária"* [1].

Neste trabalho existe o campo "duracao" presente na tabela Tarefas_Funcionario, que está desnormalizado. Este campo contém o tempo de duração total de cada tarefa. Embora este campo seja redundante (pois é possível obtê-lo através da diferença entre os campos "fim_tarefa" e "inicio_tarefa"), servirá para se obterem estatísticas referentes à duração de cada tarefa em específico.

Para outras situações, vamos utilizar TRIGGERS, VIEWS e vários métodos para colmatar as soluções que a desnormalização nos oferece. Um exemplo disto é a nossa VIEW nom_desc, que nos retorna os funcionários e a função e que podia ser resolvida com a inserção de uma coluna redundante.

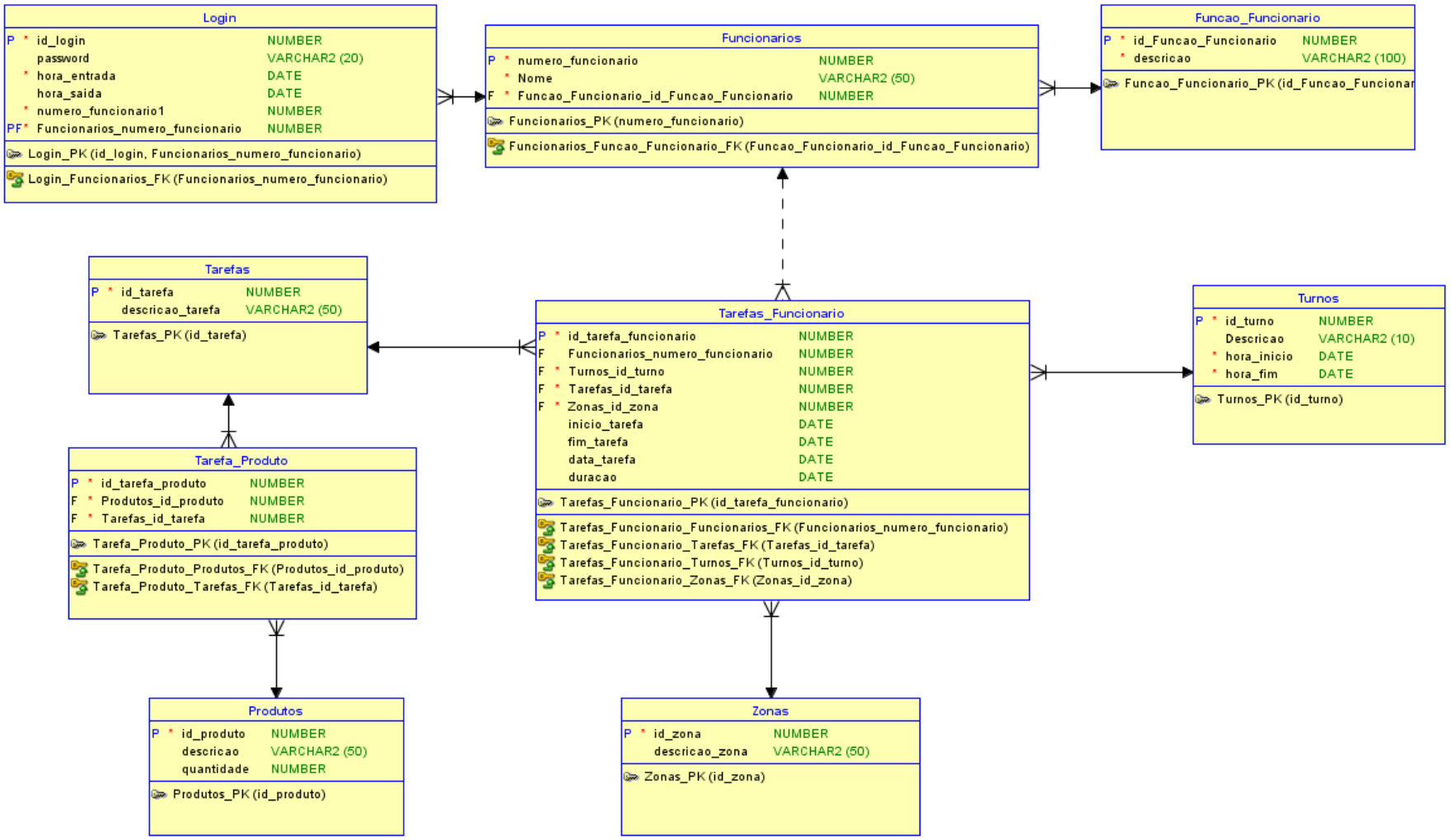

Figura 1 - Modelo ER

2. ER SQLITE

Na Figura 2 apresentamos o nosso modelo Entidade - Relacionamento para o SQLite.

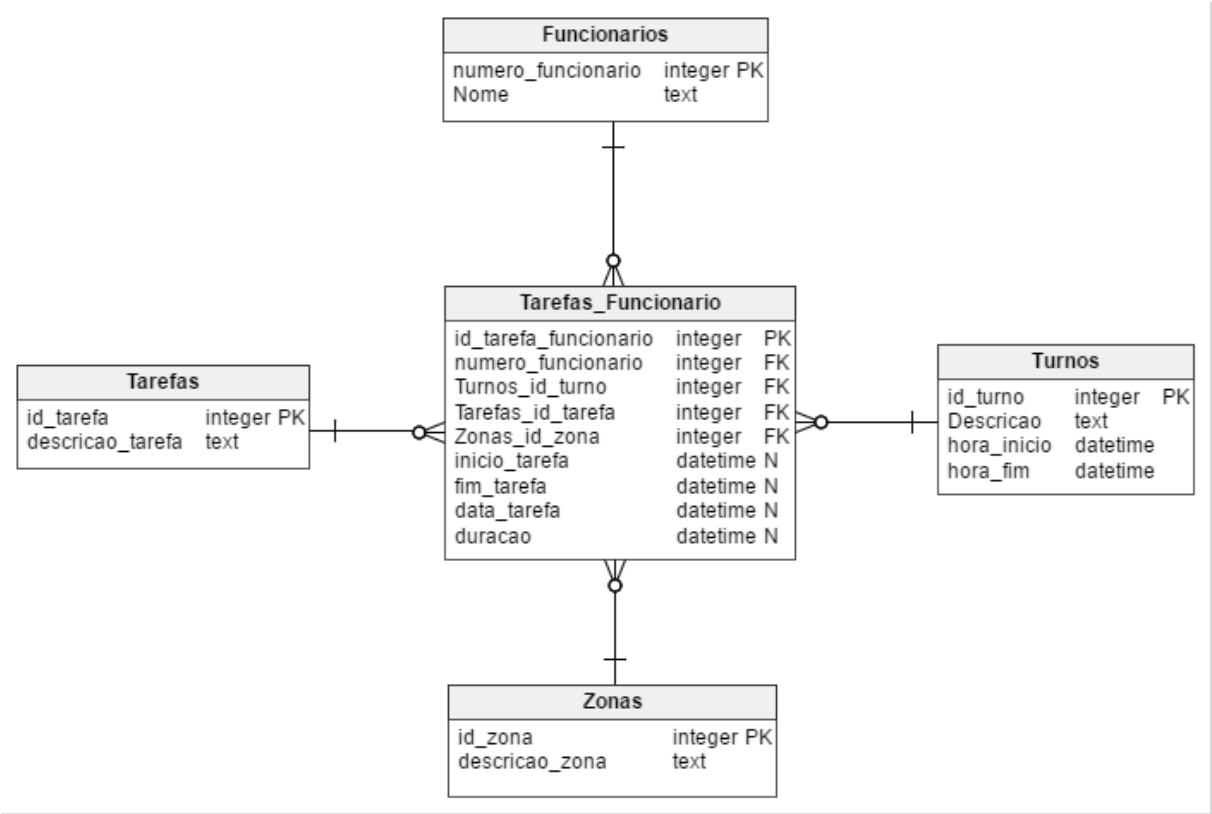

Figura 2 - ER SQLITE

3. DICIONÁRIO DE DADOS

O dicionário de dados é importante porque apresenta uma lista organizada de todos os elementos de dados pertinentes para o sistema. É imperativo ter um ponto de referência de todos os elementos envolvidos, que permita associar um significado a cada termo utilizado. Nesse sentido, passamos a apresentar o Dicionário de Dados:

Tabela 1 - Função Funcionário

Nº	Nome	Descrição	PK	FK	M	Tipo de Dados
1	id_Funcao Funcionario	Chave Primária	P		Y	NUMERIC (9)
2	Descrição	Nome da Função			Y	Varchar (100)

Tabela 2 - Funcionário

Nº	Nome	Descrição	PK	FK	M	Tipo de Dados
1	numero_funcionario	Chave Primária	P		Y	NUMERIC (9)
2	Nome	Nome do Funcionário			Y	VARCHAR (50)
3	id_Funcao_Funcionario	Chave estrangeira		F	Y	NUMERIC (9)

Tabela 3 - Login

Nº	Nome	Descrição	PK	FK	M	Tipo de Dados
1	id_login	Chave Primária	P		Y	NUMERIC (9)
2	Pasword	Password do Funcionario			Y	VARCHAR (20)
3	numero_funcionario	Chave Estrangeira		F	Y	NUMERIC (9)
4	hora_entrada	Hora Login			Y	Datetime
5	hora_saida	Hora Logout			Y	Datetime

Tabela 4 - Produtos

Nº	Nome	Descrição	PK	FK	M	Tipo de Dados
1	id_produto	Chave Primária	P		Y	NUMERIC (9)
2	Descrição	Nome do Produto				VARCHAR (50)
3	Quantidade	Quantidade Existente				NUMERIC (9)

Tabela 5 - Tarefa Produto

Nº	Nome	Descrição	PK	FK	M	Tipo de Dados
1	id_tarefa_produto	Chave Primária	P		Y	NUMERIC (9)
2	Produtos_id_produto	Chave Estrangeira		F	Y	NUMERIC (9)
3	Tarefas_id_tarefa	Chave Estrangeira		F	Y	NUMERIC (9)

Tabela 6 - Tarefas

Nº	Nome	Descrição	PK	FK	M	Tipo de Dados
1	id_tarefa	Chave Primária	P		Y	NUMERIC (9)
2	descricao_tarefa	Descrição da tarefa a realizar				VARCHAR (50)

Tabela 7 - Tarefa Funcionário

Nº	Nome	Descrição	PK	FK	M	Tipo de Dados
1	id_tarefa_funcionario	Chave Primária	P		Y	NUMERIC (9)
2	numero_funcionario	Chave Estrangeira		F	Y	NUMERIC (9)
3	Turnos_id_turno	Chave Estrangeira		F	Y	NUMERIC (9)
4	Tarefas_id_tarefa	Chave Estrangeira		F	Y	NUMERIC (9)
5	Zonas_id_zona	Chave Estrangeira		F	Y	NUMERIC (9)
6	inicio_tarefa	Hora de inicio da tarefa				Datetime
7	fim_tarefa	Hora fim da tarefa				Datetime
8	data_tarefa	Dia da tarefa				Date
9	duracao	Duração da tarefa				Datetime

Tabela 8 - Turno

Nº	Nome	Descrição	PK	FK	M	Tipo de Dados
1	id_turno	Chave Primária	P		Y	NUMERIC (9)
2	Descricao	Descrição do Turno				VARCHAR (10)
3	hora_inicio	Hora de inicio do turno			Y	Datetime
4	hora_fim	Hora de fim do turno			Y	Datetime

Tabela 9 - Zonas

Nº	Nome	Descrição	PK	FK	M	Tipo de Dados
1	id_zona	Chave Primária	P		Y	NUMERIC (9)
2	descricao_zona	Descrição da Zona ou Área				VARCHAR (50)

4. RESTRIÇÕES NAS TABELAS

Todas as nossas tabelas estão de acordo com as restrições de integridade, ou seja, cumpriu-se as regras de restrições no que diz respeito ás:

- Restrições de Entidade, as nossas chaves primárias são únicas e nunca nulas;
- Restrições de Referencial, as nossas chaves estrangeiras só podem ter valores das chaves primárias;
- Restrições de Domínio, para cada campo foi verificado a existência da obrigatoriedade, dos tipos de dados, tamanho e limites;
- Regras complexas de negócio, não são aplicáveis ao nosso modelo.

5. PROCEDIMENTOS, FUNÇÕES, TRIGGERS, PACKAGES, VIEWS, SEQUÊNCIAS E SINÓNIMOS

a. Views

Podemos automatizar consultas simples á nossa base de dados utilizando as Views. As Views são guardadas e a qualquer momento podemos invoca-las e utilizar para diversas operações, tais como:

- Simplificar queries, isto é, longos selects são reduzidos ao nome da View;
- Fornecer rápido acesso á informação, ao chamarmos a View;
- Controlar acesso a informação por parte de determinados utilizadores, dado que podemos criar views que tenham permissões limitadas a nível de colunas de determinadas tabelas e determinados utilizadores ou grupos tenham acesso a ele.

No nosso trabalho tornou-se importante saber o nome dos funcionários e a respetiva função. Criamos um view que nos permite ter acesso a essa informação, com o nome nome_desc.

```
create view nom_desc as select f.nome, g.descricao
From funcionarios f, FUNCAO_FUNCIONARIO g where
f.ID_FUNCAO_FUNCIONARIO=g.ID_FUNCAO_FUNCIONARIO;
```

Outra necessidade neste projeto é saber quais as tarefas agendadas para o dia atual, assim como, quem as realiza, em que local e se já foram iniciadas ou finalizadas. Assim criou-se a seguinte View com o nome VIEW_TAREFAS_DIA.

```
CREATE OR REPLACE FORCE EDITIONABLE VIEW "LUIS"."VIEW_TAREFAS_DIA" ("NOME",
"TAREFA", "DESCRICAO", "ZONA", "INICIO", "FIM") AS
  SELECT funcionarios.nome as nome, funcao_funcionario.descricao as tarefa,
tarefas.descricao_tarefa as descricao, zonas.descricao_zona as zona,
tarefas_funcionario.inicio_tarefa as inicio, tarefas_funcionario.fim_tarefa as fim
  FROM  funcionarios, tarefas, zonas, tarefas_funcionario, funcao_funcionario
  WHERE funcionarios.numero_funcionario = tarefas_funcionario.numero_funcionario
  and tarefas.id_tarefa = tarefas_funcionario.tarefas_id_tarefa
  and zonas.id_zona = tarefas_funcionario.zonas_id_zona
  and funcao_funcionario.id_funcao_funcionario =
funcionarios.id_funcao_funcionario
  and tarefas_funcionario.data_tarefa like to_date(sysdate);
```

(Esta VIEW será depois utilizada no webservice, sendo que o cliente Android receberá um JSON com estes dados)

b. Sequências

O recurso a sequências no Oracle é utilizado para Auto incremento. É um objeto que é capaz de devolver números de forma ordenada e garantindo que nunca é devolvido o mesmo numero duas vezes.

No nosso trabalho usamos duas sequências com os nomes:

PRODUTOS_SEQ

```
CREATE SEQUENCE  "LUIS"."PRODUTOS_SEQ"  MINVALUE 1 MAXVALUE 99999 INCREMENT BY 1
START WITH 99 NOCACHE  NOORDER  NOCYCLE  NOPARTITION;
```

NEXT_VAL

```
CREATE SEQUENCE  "LUIS"."NEXT_VAL"  MINVALUE 1 MAXVALUE
9999999999999999999999999999 INCREMENT BY 1 START WITH 101 CACHE 100 NOORDER
NOCYCLE  NOPARTITION ;
```

START WITH	Valor inicia a sequência, no nosso primeiro exemplo foi 99
INCREMENT BY	Qual vai ser o incremento, no nosso caso, de 1 em 1
MAXVALUE	Valor máximo da sequência, no nosso caso, 99999
MINVALUE	Valor mínimo da sequência, no nosso caso, 1
NOCACHE / CACHE N	Quantidade de cache pré-reservada no Oracle, no nosso caso, nenhuma
NOCYCLE / CYCLE	Caso a sequência chegue ao seu limite, se volta ao início ou não.

É de salientar que para eliminar uma sequência, basta usar “drop sequence nome_da_sequencia” .

c. Sinónimos

Os sinónimos são objetos da base de dados que efetuam ligações com outros objetos, tipo um link. No nosso projeto o *owner* é “luis”, ou seja, se outro utilizador pretender fazer um select à tabela funcionários tem que inserir, select * from luis.funcionarios, partindo do pressuposto que tem permissões para isso. Assim, se criarmos um sinónimo não temos que estar sempre a referenciar o *owner*.

Assim criamos um sinonimo privado para o utilizador miguel e já podemos fazer o select sem mencionar o *owner*.

```
create or replace synonym funcionarios for luis.funcionarios;

select * from funcionarios;
```

Criamos de seguida um sinonimo publico, para funcionar com todos os utilizadores.

```
create or replace public synonym produtos for luis.produtos;
```

Com este sinonimo todos os utilizadores com permissões suficientes podem aceder á tabela produtos sem ter que mencionar o *owner*.

d. Procedimentos

1. novo_funcionario - este procedimento tem como objetivo inserir um novo funcionário, tendo como parâmetros de entrada o nome do funcionário e a função do mesmo:

```
create or replace procedure novo_funcionario
( nome_in IN funcionarios.nome_funcioario%type , id_funcao_in IN
funcionarios.ID_FUNCAO_FUNCIONARIO%type )
is
begin

   INSERT INTO funcionarios
   ( NOME,ID_FUNCAO_FUNCIONARIO )
   VALUES
   ( nome_in,id_funcao_in);

   commit;
end;
```

2. actualiza_stock_produto - o objetivo deste procedimento, é atualizar o stock dos produtos. Tem como parâmetros de entrada, o ID do produto e a quantidade a acrescentar ao stock:

```
create or replace procedure actualiza_stock_produto
( id_produto_in IN produtos.id_produto%type, quantidade_in IN
produtos.quantidade%type)
is

v_qt_actual_produto produtos.quantidade%type;
v_quantidade_final produtos.quantidade%type;

begin
        select quantidade into v_qt_actual_produto
        from produtos
        where id_produto = id_produto_in;

        v_quantidade_final := v_qt_actual_produto+quantidade_in;

        update produtos
        set quantidade = v_quantidade_final
        where id_produto = id_produto_in;

   commit;
end;
```

3. insert_new_login: este procedimento tem a finalidade de inserir um novo registo de login, quando o utilizador der “entrada ao serviço”. Recebe como parâmetros de entrada o número de funcionário.

```
create or replace procedure insert_new_login

( in_num_funcionario IN login.numero_funcionario%type)
is
begin
        insert into luis.login (numero_funcionario,hora_entrada)
        values (in_num_funcionario,sysdate);
   commit;
end;
```

4. do_logout: este procedimento tem como objectivo atualizar o campo hora_saida na tabela login. Ou seja, é registar a saída do funcionário do serviço. Tem como parâmetro de entrada o número de funcionário.

```
create or replace procedure do_logout
( in_num_funcionario IN login.numero_funcionario%type)
is
begin
        update luis.login
        set hora_saida = sysdate
        where numero_funcionario = in_num_funcionario
        and hora_saida is null
        and to_char(hora_entrada, 'DD') = to_char(sysdate, 'DD');
   commit;
end;
```

Nota: neste procedimento, não é permitido ao utilizador efetuar uma saída de serviço no dia seguinte. Ou seja, no caso de "esquecimento" de picagem de saída por parte do funcionário, não é permitido efetuá-la no dia seguinte.

e. Funções

1. Safety_instructions: A função seguinte tem como objectivo devolver o JSON com os dados do rótulo de cada produto de cada tarefa. O objectivo desta função é "consumir" um webservice remoto. O cliente Android não tem acesso directo a esse webservice, logo irá "consumir" um webservice na Base de dados, que por sua vez esta, via backend irá "consumir" este webservice remoto. No fim, este último webservice devolve à base de dados um JSON com as precauções do produto que é utilizado na tarefa. O parâmetro de entrada nesta função é o id_tarefa_funcionario, da tabela tarefa_funcionario.

```
create or replace function safety_instructions(
in_id_tarefa_funcionario IN TAREFAS_FUNCIONARIO.ID_TAREFA_FUNCIONARIO%type)
return VARCHAR2

is

v_request_handle UTL_HTTP.REQ;
v_response_handle UTL_HTTP.RESP;
lv_response_line VARCHAR2 (32767);
v_nome_produto produtos.descricao%type;
v_json_formated VARCHAR2 (32767) default '';

BEGIN

select produtos.descricao into v_nome_produto
from produtos, tarefas, tarefa_produto
where  tarefas.ID_TAREFA =
(select tarefas_id_tarefa
from tarefas_funcionario
where tarefas_funcionario.ID_TAREFA_FUNCIONARIO = in_id_tarefa_funcionario)
and tarefa_produto.tarefas_id_tarefa = tarefas.id_tarefa
and produtos.id_produto = tarefa_produto.PRODUTOS_ID_PRODUTO;

if v_nome_produto like 'Limpa vidros' then
  v_nome_produto:='limpavidros';
end if;


v_request_handle:=
UTL_HTTP.BEGIN_REQUEST('http://192.168.1.25:8000/ProdutoDetalhes/' ||
lower(v_nome_produto));
UTL_HTTP.SET_HEADER(v_request_handle, 'User-Agent', 'Mozilla/4.0');

v_response_handle:= UTL_HTTP.GET_RESPONSE(v_request_handle);

LOOP
UTL_HTTP.READ_LINE(v_response_handle, lv_response_line, TRUE);
--DBMS_OUTPUT.PUT_LINE(lv_response_line);
v_json_formated := v_json_formated || lv_response_line;
```

```
END LOOP;

EXCEPTION
WHEN UTL_HTTP.END_OF_BODY THEN
UTL_HTTP.END_RESPONSE(v_response_handle);

return v_json_formated;

END;
```

f. Triggers

Foram utilizados triggers para as seguintes tabelas: funcao_funcionario, funcionários, login, produtos, tarefa_produto, tarefas, tarefas_funcionario, turnos, zonas.
Exemplos:

```
create or replace TRIGGER trigger_tarefas
  BEFORE INSERT ON tarefas
  FOR EACH ROW
BEGIN
  SELECT NEXT_VAL.nextval
    INTO :new.ID_TAREFA
    FROM dual;
END;
```

```
create or replace TRIGGER trigger_tarefa_produto
  BEFORE INSERT ON Tarefa_produto
  FOR EACH ROW
BEGIN
  SELECT NEXT_VAL.nextval
    INTO :new.ID_TAREFA_PRODUTO
    FROM dual;
END;
```

Todos os triggers são “triggered” antes do “insert” nas respectivas tabelas. E todos têm como objectivo inserir, recorrendo a uma sequência, o valor da chave primária para cada registo em cada tabela.

6. PACKAGES

Neste projeto foram criados dois packages sendo um dedicado à configuração do webservice, e um outro dedicado à automatização da escala de tarefas para os funcionários.

1. Package “servicos”

O package seguinte irá conter uma procedure que define a configuração do webservice. A procedure contém código para operações do tipo GET.

```
create or replace package servicos as
PROCEDURE primaria(p_path IN VARCHAR2);
end;
```

Dentro deste package existe uma procedure que terá as funcionalidades de fazer update ao campo “fim_tarefa” na tabela tarefas_funcionario, devolver, através de um Select a uma View, as tarefas que estão planeadas para o dia atual, e por último consumir um webservice remoto (existe depois uma função que será mais detalhada no seguimento deste relatório). Notar que será através desta (procedure) que o cliente móvel irá “consumir” estas três funcionalidades.

```
create or replace package body servicos as

g_request_method_get           constant varchar2(10) := 'GET';
g_request_method_post          constant varchar2(10) := 'POST';
g_resource_type_employees      constant varchar2(255) := 'employees';


PROCEDURE primaria(
    p_path IN VARCHAR2)
AS
  l_request_method CONSTANT VARCHAR2(10) :=
owa_util.get_cgi_env('REQUEST_METHOD');
  l_resource       VARCHAR2(2000);
  l_path           apex_application_global.vc_arr2;
  l_id             NUMBER(4);
  v_aux          number(4);
  v_aux2 number(4);
  v_aux_string           varchar2(20);
  v_formated_json varchar2(5000);
  v_contador_rows number default 0;
  var_p_in_actualiza_tarefa number;
  affected_rows number;

BEGIN
```

```
if (l_request_method = g_request_method_get) then
  v_aux := instr(p_path,'/');
  if (v_aux != 0) then
    v_aux_string := substr(p_path,0,v_aux-1);
  else
    v_aux_string := p_path;
  end if;

        IF (lower(v_aux_string) like 'tarefas_dia') then
    v_formated_json := '[';
          FOR l_rec IN
        (select * from luis.view_tarefas_dia
        )
        LOOP
       v_contador_rows := v_contador_rows +1;

       if (v_contador_rows > 1) then
        v_formated_json := v_formated_json || ',' ;
       end if;

                  v_formated_json := v_formated_json || '{"id_tarefa":"' ||
l_rec.id_tarefa_funcionario || '","nome":"' || l_rec.nome || '","tarefa":"'
           || l_rec.tarefa || '","descricao":"' || l_rec.descricao || '","zona":"'
|| l_rec.zona
           || '","inicio":"' || l_rec.inicio || '","fim":"' || l_rec.fim || '"}';

        END LOOP;
        v_formated_json := v_formated_json || ']';
        htp.p(v_formated_json);
  end if;

  IF (lower(v_aux_string) like 'actualiza_tarefa') then

      var_p_in_actualiza_tarefa := to_number(substr(p_path,v_aux+1));
      update tarefas_funcionario set
      fim_tarefa = to_date(sysdate, 'YYYY/MM/DD HH24:MI:SS')
      where ID_TAREFA_FUNCIONARIO = var_p_in_actualiza_tarefa;
      affected_rows := sql%rowcount;
      commit;

     htp.p('{"affected_rows":"'|| affected_rows || '"}');

    end if;

IF (lower(v_aux_string) like 'safety_instructions') then
        var_p_in_actualiza_tarefa := to_number(substr(p_path,v_aux+1));
        begin
                v_safety_instructions :=
luis.SAFETY_INSTRUCTIONS(var_p_in_actualiza_tarefa);
        end;
        htp.p(v_safety_instructions);
end if;

end if;
end primaria;
end servicos;
```

2. Package “pkg_gera_tarefas_funcs_diario”

Um outro package que foi criado contém as funções que servirão para gerar tarefas aleatórias diárias para os empregados. Dentro deste package existe um tipo de variável *own created* de modo a suportar o output e input das funções neste package.

```
create or replace package pkg_gera_tarefas_funcs_diario as
  type array_random_tarefas IS VARRAY(300) of Number;
  FUNCTION aleatorio_task_emps RETURN array_random_tarefas;
  FUNCTION    gera_turnos_zonas(random_task_emps    array_random_tarefas)    RETURN
VARCHAR2;
end pkg_gera_tarefas_funcs_diario;
```

Este package, contém duas funções:

2.1. aleatorio_task_emps - esta função não recebe nenhum parâmetro de entrada, e tem a finalidade de associar cada tarefa a um funcionário, aleatoriamente. No fim retorna um array, que terá a seguinte estrutura:

```
[numero_funcionario, id_tarefa, numero_funcionario, id_tarefa…]
```

```
FUNCTION aleatorio_task_emps
RETURN array_random_tarefas
IS

CURSOR c_funcionarios IS
SELECT * FROM funcionarios
order by dbms_random.value();
v_funcionarios  c_funcionarios%rowtype;

CURSOR c_tarefas IS
SELECT * FROM tarefas
order by dbms_random.value();
v_tarefas c_tarefas%ROWTYPE;

v_contador integer := 1;
type array_funcionarios IS VARRAY(100) OF c_funcionarios%rowtype;
type array_tarefas IS VARRAY(100) OF c_tarefas%rowtype;
gera_tarefas array_random_tarefas := array_random_tarefas();
funcionarios array_funcionarios := array_funcionarios();
tarefas array_tarefas := array_tarefas();
v_funcionarios_size integer;
v_random integer;
v_id_tarefa number;
v_id_funcionario number;
v_contador_gera_tarefas number := 1;
v_total_funcionarios number := 0;
v_contador_funcionarios number := 1;
v_contador_tarefas number := 1;

v_total_tarefas number := 0;
```

```
BEGIN
  open c_funcionarios;
  LOOP
    fetch c_funcionarios
    into v_funcionarios;
    exit when c_funcionarios%NOTFOUND;
    funcionarios.EXTEND(1);
    funcionarios(v_contador) := v_funcionarios;
    v_contador := v_contador+1;
  END LOOP;
  close c_funcionarios;

  v_contador := 1;
  open c_tarefas;
  LOOP
    fetch c_tarefas
    into v_tarefas;
    exit when c_tarefas%NOTFOUND;
    tarefas.EXTEND(1);
    tarefas(v_contador) := v_tarefas;
    v_contador := v_contador+1;
  END LOOP;
  close c_tarefas;

   v_contador := 1;
  v_total_funcionarios := funcionarios.COUNT();
  v_total_tarefas := tarefas.COUNT();
    WHILE (v_total_funcionarios>0)
      LOOP
        gera_tarefas.EXTEND(2);
        gera_tarefas(v_contador_gera_tarefas)                                    :=
funcionarios(v_contador_funcionarios).numero_funcionario;
        v_contador_gera_tarefas := v_contador_gera_tarefas +1;
        gera_tarefas(v_contador_gera_tarefas)                                    :=
tarefas(v_contador_tarefas).id_tarefa;
        v_contador_gera_tarefas := v_contador_gera_tarefas +1;

                if (v_contador_tarefas = v_total_tarefas) then
                v_contador_tarefas :=0;
                end if;

        v_contador_funcionarios := v_contador_funcionarios +1 ;
        v_contador_tarefas := v_contador_tarefas +1;
        v_total_funcionarios := v_total_funcionarios -1;
      end loop;
return gera_tarefas;

END aleatorio_task_emps;
```

2.2 gera_turnos_zonas – recebe como parâmetro de entrada um array, que é proveniente da função aleatorio_task_emps, e tem como função associar um turno e uma zona à tarefa do funcionário. O algoritmo implementado está descrito abaixo:

2.2.1 É feito um ciclo de modo a percorrer todos os números dos funcionários no array proveniente da função aleatorio_task_emps

2.2.2 Para o respetivo funcionário é validado o número de vezes que durante os 7 dias anteriores à data atual fez o mesmo turno, ordenando pelo turno que fez mais vezes. Se este valor for superior a 4x, significa que o funcionário já fez durante a semana o mesmo turno 4x, logo transitará para o turno diferente do atual. Isto fica mais simplificado pois neste modelo de dados apenas existem dois turnos.

2.2.3 Para o respetivo funcionário é validado o número de vezes que durante os 7 dias anteriores à data atual fez a mesma zona, ordenando pela zona que fez mais vezes. Então, se essa zona já tiver sido a mesma 3 vezes no espaço temporal dos 7 dias (zona_repetida), será escolhida uma outra zona diferente. Como neste modelo de dados temos 3 zonas diferentes, a seleção de uma nova zona é feita com recurso à aleatoriedade, em que são selecionadas todas as zonas, exceto a zona_repetida, e é escolhida a primeira zona disponível, mas cuja escolha variará pela ordem (crescente ou decrescente) em que são devolvidas as zonas disponíveis. Ou seja, caso o número aleatório gerado seja 0, a seleção das zonas é feita ordenando pela ordem decrescente do id_zona; caso seja 1 o resultado será ordenado por ordem crescente do id_zona.

2.2.4 No fim é retornado uma string que contém os campos:

- Número de tarefas
- Número do funcionário
- Identificador da tarefa
- Identificador do turno
- Identificador da zona

```
FUNCTION gera_turnos_zonas( random_task_emps array_random_tarefas )
RETURN VARCHAR2

IS

v_tasks VARCHAR2(1000) default '';
v_id_turno turnos.id_turno%type;
desc_turno turnos.descricao%type;
v_id_zona zonas.id_zona%type;
v_contador_emps number;
v_total_emps number;
v_dias_mesmo_turno number;
v_dias_mesma_zona number;
v_ordena_zona_aleatorio number;
v_string_ordena_zona varchar(4);

BEGIN

v_contador_emps := 1;
v_total_emps := random_task_emps.COUNT()/2;
DBMS_OUTPUT.PUT_LINE('Total empregados = ' || v_total_emps);

v_tasks:=v_total_emps||'/';
```

```
while (v_total_emps>0)
loop
-------------------------------------------------TURNOS-----------------------------
--------------------------
--devolve o numero do funcionario
DBMS_OUTPUT.PUT_LINE('Numero funcionario = ' ||
random_task_emps(v_contador_emps));

begin
--conta quantas X o funcionario fez o mesmo turno e se for maior que 4x, irá para
um turn diferente.
select * into v_dias_mesmo_turno, v_id_turno from (
select count(turnos_id_turno) as contador, turnos_id_turno
from TAREFAS_FUNCIONARIO
where numero_funcionario = random_task_emps(v_contador_emps)
and data_tarefa between trunc(sysdate-7) AND
trunc(sysdate)
group by turnos_id_turno
order by contador desc
) where ROWNUM = 1;


exception
 when no_data_found then
    --devolve um turno aleatorio quando o funcionario nunca fez nenhuma tarefa
        select * into v_id_turno
        from (select id_turno
        from turnos
        where id_turno != -1
        order by dbms_random.value()
        ) where ROWNUM = 1;

 when too_many_rows then
    --devolve o turno que o funcionario fez mais vezes
        DBMS_OUTPUT.PUT_LINE('id_turno = ' || v_id_turno);

        if v_dias_mesmo_turno > 1 then --trocar o 1 por 4 (4x o mesmo turno em 7
dias)
                select id_turno into v_id_turno
                from turnos
                where id_turno != v_id_turno;
        end if;
  end;


--dmbs_output dos turnos

DBMS_OUTPUT.PUT_LINE('Numero de turnos feitos = ' || v_dias_mesmo_turno);
DBMS_OUTPUT.PUT_LINE('Novo turno = ' || v_id_turno);

-------------------------------------------------ZONAS-----------------------------
--------------------------
--conta quantas vezes o funcionario fez a mesma zona
begin
select * into v_id_zona, v_dias_mesma_zona from (
select zonas_id_zona, count(zonas_id_zona) as conta_zonas
from tarefas_funcionario
```

```
where NUMERO_FUNCIONARIO = random_task_emps(v_contador_emps)
and data_tarefa between trunc(sysdate-7) AND
trunc(sysdate)
group by zonas_id_zona
order by conta_zonas desc
)where  ROWNUM=1;

--devolve a zona que o funcionario ja esteve mais vezes
exception
 when no_data_found then
    --devolve uma zona aleatoria quando o funcionario nunca fez nenhuma
tarefas_funcionario
        select * into v_id_zona
        from (select id_zona
        from zonas
        where id_zona != -1
        order by dbms_random.value())
        where ROWNUM = 1;


 when too_many_rows then

        DBMS_OUTPUT.PUT_LINE('id_zona = ' || v_id_zona);
        if v_dias_mesma_zona > 2 then
        select round(dbms_random.value(0,1)) into v_ordena_zona_aleatorio from
dual;
        if (v_ordena_zona_aleatorio = 0) then
                v_string_ordena_zona := 'desc';
        else
                v_string_ordena_zona := 'asc';
        end if;

        select * into v_id_zona from(
        select id_zona
        from zonas
        where id_zona != v_id_zona
        order by id_zona || v_string_ordena_zona
        ) where ROWNUM = 1;

        end if;
end;
--dmbs_output das zonas
DBMS_OUTPUT.PUT_LINE('Numero de zonas feitas = ' || v_dias_mesma_zona);
DBMS_OUTPUT.PUT_LINE('Nova zona = ' || v_id_zona);

v_tasks:= v_tasks ||(random_task_emps(v_contador_emps)||','||v_id_turno
||','||random_task_emps(v_contador_emps+1)|| ',' || v_id_zona ||';');

v_total_emps:= v_total_emps-1;
v_contador_emps := v_contador_emps+2;

end loop;

return v_tasks;

end;
```

7. PRIVILÉGIOS, ROLES E USERS

A nível de roles e privilégios, o acesso à Base de Dados via cliente Android, é feito através de dum utilizador denominado “SERVICO”. Assim, esta aplicação está desenhada de forma que todos os utilizadores Android utilizem este mesmo utilizador “SERVICO” para o acesso à Base de dados.

Este utilizador é totalmente diferente do “owner” das tabelas, possuindo o nível de privilégios mínimo. Para auxílio recorreu-se a uma matriz CRUD de modo a identificar que ações este utilizador pode executar (só para tabelas):

Utilizador	Funcionarios	tarefa_funcionario	funcao_funcionario	tarefas	zonas	login	produtos	Tarefa_produto
servico	R	RU	R	R	R	CU	R	R

Este utilizador terá também permissões para executar uma função denominada de safety_instructions.

A configuração efetuada foi:

- Criação do utilizador:

```
create user servico identified by S34viC0w3bSe4vice;
grant connect to servico;
```

- Criação da role para este utilizador:

```
create role user_servico_web;
```

- Adição de privilégios a esta role:

```
grant select on funcionarios to user_servico_web;
grant select on zonas to user_servico_web;
grant select on tarefas to user_servico_web;
grant select on tarefas_funcionario to user_servico_web;
grant select on funcao_funcionario to user_servico_web;
grant update(inicio_tarefa, fim_tarefa) on luis.tarefas_funcionario to
user_servico_web;
grant insert on luis.login to user_servico_web;
grant update (hora_saida) on luis.login to user_servico_web;
grant select on luis.view_tarefas_dia to user_servico_web;
grant select on produtos to user_servico_web;
grant select on tarefa_produto to user_servico_web;
grant execute on safety_instructions to user_servico_web;
```

- Atribuição da role ao utilizador

```
grant user_servico_web to servico;
```

8. WEBSERVICE E APLICAÇÃO MÓVEL

Na base de dados Oracle foi configurado um webservice para que o cliente Android “consuma” determinados serviços. Estes serviços são: receber as tarefas definidas para o dia atual, e atualizar a data/hora de fim de uma tarefa. Segue a configuração deste webservice:

```
begin
DBMS_EPG.create_dad(dad_name=>'luisWS', path=>'/luisWS/*');
dbms_epg.set_dad_attribute (dad_name => 'luisWS', attr_name => 'path-alias',
attr_value => 'servicos');
dbms_epg.set_dad_attribute (dad_name => 'luisWS', attr_name => 'path-alias-
procedure', attr_value => 'servicos.primaria');
DBMS_EPG.set_dad_attribute(dad_name=> 'luisWS', attr_name=> 'database-username',
attr_value=> 'LUIS');
DBMS_EPG.authorize_dad(dad_name=> 'luisWS', user=> 'SERVICO');
end;
```

BIBLIOGRAFIA

[1] J. C. Fonseca, *MODELO RELACIONAL (ORACLE E MYSQL),* Guarda, 2016.

[2] O. D. Online, “How to Create and Manage Views in Oracle,” [Online]. Available: http://www.oracle-dba-online.com/sql/create_and_manage_views.htm.

[3] J. C. Fonseca, *Normalização e Desnormalização,* 2016.

[4] D. Atilio, “https://terminaldeinformacao.com/2014/02/18/criando-sequences-no-oracle/,” [Online].

[5] M. Braten, “ORA-00001: Unique constraint violated: Creating a REST web service with PL/SQL (and making pretty URLs for your Apex apps),” 5 Julho 2009. [Online]. Available: http://ora-00001.blogspot.pt/2009/07/creating-rest-web-service-with-plsql.html. [Acedido em 12 2016].

ANEXOS (SQL CRIAÇÃO DAS TABELAS)

Tabela funcao_funcionario

```
   CREATE TABLE "LUIS"."FUNCAO_FUNCIONARIO"
   (    "ID_FUNCAO_FUNCIONARIO" NUMBER,
        "DESCRICAO" VARCHAR2(100 BYTE)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
REM INSERTING into LUIS.FUNCAO_FUNCIONARIO
SET DEFINE OFF;
Insert into LUIS.FUNCAO_FUNCIONARIO (ID_FUNCAO_FUNCIONARIO,DESCRICAO) values
('2','LIMPEZA');
Insert into LUIS.FUNCAO_FUNCIONARIO (ID_FUNCAO_FUNCIONARIO,DESCRICAO) values
('4','ROUPEIRO');
--------------------------------------------------------
--  DDL for Index FUNCAO_FUNCIONARIO_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "LUIS"."FUNCAO_FUNCIONARIO_PK" ON
"LUIS"."FUNCAO_FUNCIONARIO" ("ID_FUNCAO_FUNCIONARIO")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

--------------------------------------------------------
--  Constraints for Table FUNCAO_FUNCIONARIO
--------------------------------------------------------

  ALTER TABLE "LUIS"."FUNCAO_FUNCIONARIO" ADD CONSTRAINT "FUNCAO_FUNCIONARIO_PK"
PRIMARY KEY ("ID_FUNCAO_FUNCIONARIO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "LUIS"."FUNCAO_FUNCIONARIO" MODIFY ("DESCRICAO" NOT NULL ENABLE);
  ALTER TABLE "LUIS"."FUNCAO_FUNCIONARIO" MODIFY ("ID_FUNCAO_FUNCIONARIO" NOT NULL
ENABLE);
```

Tabela funcionarios

```
  CREATE TABLE "LUIS"."FUNCIONARIOS"
   (    "NUMERO_FUNCIONARIO" NUMBER,
        "NOME" VARCHAR2(50 BYTE),
        "ID_FUNCAO_FUNCIONARIO" NUMBER
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
REM INSERTING into LUIS.FUNCIONARIOS
SET DEFINE OFF;

--------------------------------------------------------
--  DDL for Index FUNCIONARIOS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "LUIS"."FUNCIONARIOS_PK" ON "LUIS"."FUNCIONARIOS"
("NUMERO_FUNCIONARIO")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

--------------------------------------------------------
--  Constraints for Table FUNCIONARIOS
--------------------------------------------------------

  ALTER TABLE "LUIS"."FUNCIONARIOS" ADD CONSTRAINT "FUNCIONARIOS_PK" PRIMARY KEY
("NUMERO_FUNCIONARIO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "LUIS"."FUNCIONARIOS" MODIFY ("ID_FUNCAO_FUNCIONARIO" NOT NULL
ENABLE);
  ALTER TABLE "LUIS"."FUNCIONARIOS" MODIFY ("NOME" NOT NULL ENABLE);
  ALTER TABLE "LUIS"."FUNCIONARIOS" MODIFY ("NUMERO_FUNCIONARIO" NOT NULL ENABLE);
```

Tabela login

```
  CREATE TABLE "LUIS"."FUNCIONARIOS"
   (    "NUMERO_FUNCIONARIO" NUMBER,
        "NOME" VARCHAR2(50 BYTE),
        "ID_FUNCAO_FUNCIONARIO" NUMBER
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
REM INSERTING into LUIS.FUNCIONARIOS
SET DEFINE OFF;

--------------------------------------------------------
--  DDL for Index FUNCIONARIOS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "LUIS"."FUNCIONARIOS_PK" ON "LUIS"."FUNCIONARIOS"
("NUMERO_FUNCIONARIO")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

--------------------------------------------------------
--  Constraints for Table FUNCIONARIOS
--------------------------------------------------------

  ALTER TABLE "LUIS"."FUNCIONARIOS" ADD CONSTRAINT "FUNCIONARIOS_PK" PRIMARY KEY
("NUMERO_FUNCIONARIO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
 ALTER TABLE "LUIS"."FUNCIONARIOS" MODIFY ("ID_FUNCAO_FUNCIONARIO" NOT NULL
ENABLE);
  ALTER TABLE "LUIS"."FUNCIONARIOS" MODIFY ("NOME" NOT NULL ENABLE);
  ALTER TABLE "LUIS"."FUNCIONARIOS" MODIFY ("NUMERO_FUNCIONARIO" NOT NULL ENABLE);
```

Tabela produtos

```
  CREATE TABLE "LUIS"."PRODUTOS"
   (    "ID_PRODUTO" NUMBER,
        "DESCRICAO" VARCHAR2(50 BYTE),
        "QUANTIDADE" NUMBER
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
REM INSERTING into LUIS.PRODUTOS
SET DEFINE OFF;
--------------------------------------------------------
--  DDL for Index PRODUTOS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "LUIS"."PRODUTOS_PK" ON "LUIS"."PRODUTOS" ("ID_PRODUTO")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

--------------------------------------------------------
--  Constraints for Table PRODUTOS
--------------------------------------------------------

  ALTER TABLE "LUIS"."PRODUTOS" ADD CONSTRAINT "PRODUTOS_PK" PRIMARY KEY
("ID_PRODUTO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "LUIS"."PRODUTOS" MODIFY ("ID_PRODUTO" NOT NULL ENABLE);
```

Tabela tarefa_produto

```
  CREATE TABLE "LUIS"."TAREFA_PRODUTO"
   (    "ID_TAREFA_PRODUTO" NUMBER,
        "PRODUTOS_ID_PRODUTO" NUMBER,
        "TAREFAS_ID_TAREFA" NUMBER
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
REM INSERTING into LUIS.TAREFA_PRODUTO
SET DEFINE OFF;

--------------------------------------------------------
--  DDL for Index TAREFA_PRODUTO_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "LUIS"."TAREFA_PRODUTO_PK" ON "LUIS"."TAREFA_PRODUTO"
("ID_TAREFA_PRODUTO")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

--------------------------------------------------------
--  Constraints for Table TAREFA_PRODUTO
--------------------------------------------------------

  ALTER TABLE "LUIS"."TAREFA_PRODUTO" ADD CONSTRAINT "TAREFA_PRODUTO_PK" PRIMARY
KEY ("ID_TAREFA_PRODUTO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "LUIS"."TAREFA_PRODUTO" MODIFY ("TAREFAS_ID_TAREFA" NOT NULL
ENABLE);
  ALTER TABLE "LUIS"."TAREFA_PRODUTO" MODIFY ("PRODUTOS_ID_PRODUTO" NOT NULL
ENABLE);
  ALTER TABLE "LUIS"."TAREFA_PRODUTO" MODIFY ("ID_TAREFA_PRODUTO" NOT NULL
ENABLE);
```

Tabela tarefas

```
  CREATE TABLE "LUIS"."TAREFAS"
   (    "ID_TAREFA" NUMBER,
        "DESCRICAO_TAREFA" VARCHAR2(50 BYTE)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
REM INSERTING into LUIS.TAREFAS
SET DEFINE OFF;
--------------------------------------------------------
--  DDL for Index TAREFAS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "LUIS"."TAREFAS_PK" ON "LUIS"."TAREFAS" ("ID_TAREFA")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

--------------------------------------------------------
--  Constraints for Table TAREFAS
--------------------------------------------------------

  ALTER TABLE "LUIS"."TAREFAS" ADD CONSTRAINT "TAREFAS_PK" PRIMARY KEY
("ID_TAREFA")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "LUIS"."TAREFAS" MODIFY ("ID_TAREFA" NOT NULL ENABLE);
```

Tabela turnos

```
  CREATE TABLE "LUIS"."TURNOS"
   (    "ID_TURNO" NUMBER,
        "DESCRICAO" VARCHAR2(10 BYTE),
        "HORA_INICIO" TIMESTAMP (6),
        "HORA_FIM" TIMESTAMP (6)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
REM INSERTING into LUIS.TURNOS
SET DEFINE OFF;
--------------------------------------------------------
--  DDL for Index TURNOS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "LUIS"."TURNOS_PK" ON "LUIS"."TURNOS" ("ID_TURNO")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

--------------------------------------------------------
--  Constraints for Table TURNOS
--------------------------------------------------------

  ALTER TABLE "LUIS"."TURNOS" ADD CONSTRAINT "TURNOS_PK" PRIMARY KEY ("ID_TURNO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "LUIS"."TURNOS" MODIFY ("HORA_FIM" NOT NULL ENABLE);
  ALTER TABLE "LUIS"."TURNOS" MODIFY ("HORA_INICIO" NOT NULL ENABLE);
  ALTER TABLE "LUIS"."TURNOS" MODIFY ("ID_TURNO" NOT NULL ENABLE);
```

Tabela zonas

```
  CREATE TABLE "LUIS"."ZONAS"
   (    "ID_ZONA" NUMBER,
        "DESCRICAO_ZONA" VARCHAR2(50 BYTE)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
REM INSERTING into LUIS.ZONAS
SET DEFINE OFF;
--------------------------------------------------------
--  DDL for Index ZONAS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "LUIS"."ZONAS_PK" ON "LUIS"."ZONAS" ("ID_ZONA")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

--------------------------------------------------------
--  Constraints for Table ZONAS
--------------------------------------------------------

  ALTER TABLE "LUIS"."ZONAS" ADD CONSTRAINT "ZONAS_PK" PRIMARY KEY ("ID_ZONA")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "LUIS"."ZONAS" MODIFY ("ID_ZONA" NOT NULL ENABLE);
```