POLI
ESCOLA SUPERIOR
TECNOLOGIA
GESTÃO
TÉCNICO
GUARDA

TRABALHO PRÁTICO DE BASE DE DADOS II

Projeto de Data Warehouse

Curso:	Engenharia Informática
Unidade curricular:	Base de Dados II
Ano letivo:	2021/2022
Nº e nome de aluno:	Nº 1703826 Gonçalo Silva
Docente:	JOSÉ CARLOS FONSECA
Data:	24 de março de 2022

Índice

1. Definição do produto e Plano geral do projeto

Na atual situação de concorrência nos mercados internacionais, as empresas instaladas em países mais desenvolvidos apresentam condições favoráveis nos fatores complexos de competitividade e menos vantajosas nos fatores tradicionais de produção, designadamente, em fatores como por exemplo a mão-de-obra. O nosso produto estende-se numa empresa que faz vendas de vestuário e calçado. Este produto será exposto ao público através dos meios de comunicação tal como internet, catálogo tradicional e televendas e está disponível para compra no mercado internacional. Para que a empresa tenha uma ótima gestão face ao produto em causa, foi desenvolvida uma base de dados que permite a facilidade de gerir os produtos vendidos, os produtos mais vendidos, em que circunstâncias e lugares geográficos. Para que esta base de dados seja embutida nos computadores ou POS´s da empresa, é necessário respeitar os requisitos recomendados para que a base de dados funcione corretamente e sem perdas de desempenho. E esses requisitos quanto a hardware são:

Componentes	**Requisitos recomendados**
Disco	6 GB
Monitor	Resolução 800x600 ou superior
Internet	Requer acesso à internet
Memória	1 GB
Velocidade do processador	2.0 GHz ou superior
Tipo de processador (x64)	AMD Opteron, AMD Athlon 64 ou Intel Pentium IV

Requisitos a nível de software são:

Componentes	**Requisitos**
Sistema Operativo	Windows 10
.NET Framework	---
Network Software	---

No desenvolvimento deste projeto de vestuário e calçado em que é posta em causa um projeto no âmbito de Data Warehouse, este permite responder a várias questões comerciais e financeiras colocadas aos nossos gerentes e dar aos nossos gestores de vendas: dados únicos, dados relevantes para contexto de criação de gráficos de vendas ao longo do tempo, facilidade de gestão e dados estatísticos para se realizarem comparações relativamente ao mercado concorrente. Sendo a principal tarefa extrair dados das vendas do mercado em causa, esta base de dados irá ajudar muito no estudo dos gestores inseridos neste negócio, sendo uma das principais vantagens da implementação desta base de dados, fazer com que as vendas dos produtos aumentem na empresa.

2. Modelo relacional

Na Figura1 encontra-se o modelo entidade relacionamento do reverse engineering da base de dados operacional que se encontra disponível numa máquina cujo hostname é

bd.ipg.pt. Essa base de dados operacional encontra-se num servidor cuja instância (SID) se chama oltp. Nesta instância temos dois utilizadores que são o oltp_query e o oltp2022 cujo é o dono das tabelas da base de dados operacional, logo não temos acesso à sua password. No entanto não é necessário sabermos a sua palavra-passe, pois quem desenvolve a data wharehouse vai aceder a conta do utilizador oltp_query porque esta conta tem privilégios de fazer select às tabelas do utilizador oltp2022. E assim temos acesso à base de dados operacional.

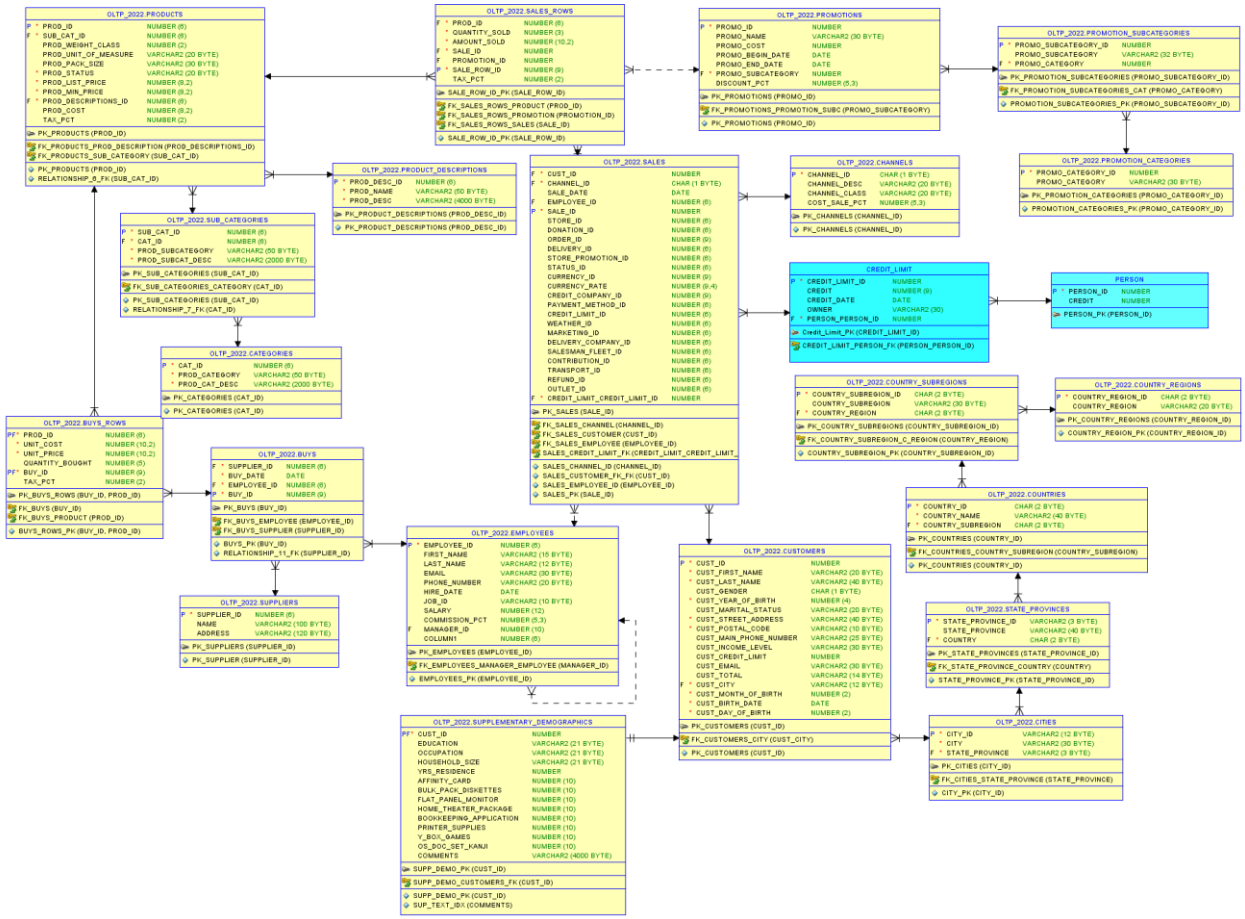

FIGURA1- MODELO RELACIONAL DO REVERSE ENGINEERING DA BASE DE DADOS OPERACIONAL COM TABELAS EXTRA

3. Database link e Synonyms

Um database link é um objeto do schema de uma base de dados que ao ser criado, possibilita o acesso a tabelas e views (conjuntos de registos que não são armazenados na BD, mas foram calculados através de queries) existentes noutra base de dados.

3.1 Criação do database link e do synonym

Na figura a seguir vemos a criação do database link chamado “OLTP_2022” que se vai conectar ao utilizador “oltp_query”, cuja password desse utilizador é “oltp_query” que usa como identificador de conexão (net service name), “oltp”.

```
create database link OLTP_2022
connect to oltp_query
identified by oltp_query
using 'oltp';
```

Um sinónimo é um objeto que criamos na base de dados que fornece um nome alternativo (criado pelo utilizador) para outro objeto existente na base de dados, conhecido como objeto base. O sinónimo criado, chamado "Sales" para evocar o database link acima é:

```
create SYNONYM Sales
for OLTP_2022.sales@oltp_2022;
```

Com a execução deste database link e do synonym, através da operação select, é possível ver a tabela "Sales" na minha conta "bdi_ei_1703826_extra".

Nas caixas de texto que se seguem vemos a criação de um synonym e o seu resultado, que nos permite visualizar todas as tabelas que existem na conta do utilizador "OLTP_2022", respetivamente.

```
SELECT 'create synonym '||table_name||' for
oltp_2022.'||table_name||'@oltp_2022;'
FROM all_tables@oltp_2022
where owner = 'OLTP_2022';
```

```
create synonym BUYS for oltp_2022.BUYS@oltp_2022;
create synonym BUYS_ROWS for oltp_2022.BUYS_ROWS@oltp_2022;
create synonym CATEGORIES for oltp_2022.CATEGORIES@oltp_2022;
create synonym CHANNELS for oltp_2022.CHANNELS@oltp_2022;
create synonym CITIES for oltp_2022.CITIES@oltp_2022;
create synonym COUNTRIES for oltp_2022.COUNTRIES@oltp_2022;
create synonym COUNTRY_REGIONS for oltp_2022.COUNTRY_REGIONS@oltp_2022;
create synonym COUNTRY_SUBREGIONS for
oltp_2022.COUNTRY_SUBREGIONS@oltp_2022;
create synonym CUSTOMERS for oltp_2022.CUSTOMERS@oltp_2022;
create synonym EMPLOYEES for oltp_2022.EMPLOYEES@oltp_2022;
create synonym PRODUCTS for oltp_2022.PRODUCTS@oltp_2022;
create synonym PRODUCT_DESCRIPTIONS for
oltp_2022.PRODUCT_DESCRIPTIONS@oltp_2022;
create synonym PROMOTIONS for oltp_2022.PROMOTIONS@oltp_2022;
create synonym PROMOTION_CATEGORIES for
oltp_2022.PROMOTION_CATEGORIES@oltp_2022;
create synonym PROMOTION_SUBCATEGORIES for
oltp_2022.PROMOTION_SUBCATEGORIES@oltp_2022;
create synonym SALES for oltp_2022.SALES@oltp_2022;
create synonym SALES_ROWS for oltp_2022.SALES_ROWS@oltp_2022;
create synonym STATE_PROVINCES for oltp_2022.STATE_PROVINCES@oltp_2022;
create synonym SUB_CATEGORIES for oltp_2022.SUB_CATEGORIES@oltp_2022;
create synonym SUPPLEMENTARY_DEMOGRAPHICS for
oltp_2022.SUPPLEMENTARY_DEMOGRAPHICS@oltp_2022;
create synonym SUPPLIERS for oltp_2022.SUPPLIERS@oltp_2022;
```

4. Código de inserção dos registos nas tabelas extra

Relativamente ao modelo Entidade Relacionamento visto na Figura1, verificamos que existem duas tabelas pintadas a cor azul, essas são as duas tabelas extra criadas, uma nomeada de "Credit_Limit" (tabela extra principal) e a outra nomeada de "Person" (tabela extra lookup). Para inserirmos dados nestas tabelas extra tivemos que criar procedimentos (procedures), que são objetos criados que executam determinada ação.

Procedimento chamado "insert_person_lookup" que insere dados nos campos da tabela lookup ou tabela chamada "Person":

```
create or replace procedure insert_person_lookup is

begin
    insert into PERSON(PERSON_ID,CREDIT) values(123456,3000);
    insert into PERSON(PERSON_ID,CREDIT) values(234567,1500);
    insert into PERSON(PERSON_ID,CREDIT) values(345678,1000);
    insert into PERSON(PERSON_ID,CREDIT) values(456789,2000);
    insert into PERSON(PERSON_ID,CREDIT) values(654321,2500);
    insert into PERSON(PERSON_ID,CREDIT) values(765432,5000);
end;
/
```

O procedimento abaixo chamado "insert_creditLimitId_principal", é desenvolvido para inserir dados nos campos da tabela "Credit_Limit". Na implementação deste procedimento, primeiramente é criado um cursor chamado "cur_credit_limit_id", uma operação "select" onde vemos que filtra os dados duplicados do atributo "credit_limit_id" da tabela "Sales"(tabela de cor amarelada) usando a cláusula "distinct", ainda retira os valores "null" usando a clausula "is not null", e de seguida ordena pelo atributo "credit_limit_id" que é a chave primária da tabela "Credit_Limit". A seguir à implementação do cursor encontra-se a secção declarativa do meu procedimento, onde declaro as variáveis necessárias para a resolução do problema em questão. Próxima etapa, abro o meu cursor a seguir à clausula "begin". Dentro de um ciclo (loop), faço um "fetch" que me permite limitar o número de registos devolvidos, esse número é o resultado do meu cursor que vai ser guardado na variável "v_credit_limit_id" e é quebrado o loop caso o valor do cursor não for encontrado.
No próximo passo é feita uma operação de "select" que armazena na variável "v_valor" o número de linhas da tabela "Person". A seguir vê-se outra operação "select" que guarda na variável "v_valor_gerado" um número aleatório gerado entre zero e o número de linhas da tabela "Person". Para a inserção dos campos do tipo "Date", "Varchar" e "Number" da tabela "Credit_Limit", são guardadas nas variáveis "v_credit_date", "v_owner" e "v_credit" valores aleatórios gerados, respetivamente. Por último insiro todos os valores gerados para os campos da tabela "Credit_Limit", fecha-se o ciclo, o cursor e o procedimento.

```
create or replace procedure insert_creditLimitId_principal as

cursor cur_credit_limit_id is
select distinct credit_limit_id
from sales
where credit_limit_id is not null
order by credit_limit_id;

v_credit_limit_id sales.credit_limit_id%TYPE;
v_credit_date credit_limit_id.credit_date%TYPE;
v_credit credit_limit_id.credit%TYPE;
v_owner credit_limit_id.owner%TYPE;
v_person_id credit_limit_id.person_person_id%TYPE;

v_valor number(6);
v_valor_gerado number(6);

begin
    open cur_credit_limit_id;
    loop
        fetch cur_credit_limit_id into v_credit_limit_id;
        exit when cur_credit_limit_id%NOTFOUND;

        select count(*)
        into v_valor
        from person;

        select trunc(dbms_random.value(0, v_valor))
        into v_valor_gerado
        from dual;

        select person.person_id into v_person_id
        from person
        offset v_valor_gerado rows
        fetch first 1 row only;

        v_credit_date :=
to_date(trunc(dbms_random.value(to_char(to_date('1/1/1980','dd/mm/yyyy'),'j
'),to_char(CURRENT_DATE+1,'j'))),'j');
        v_owner := dbms_random.string('a', dbms_random.value(2,10));
        v_credit := trunc(dbms_random.value(1,10));
        insert into credit_limit_id values(v_credit_limit_id, v_credit,
v_credit_date, v_owner, v_person_id);

    end loop;
    close cur_credit_limit_id;
end;
/
```

Para concluir a resolução deste problema ainda é criado um novo procedimento chamado “insert_tables_credit_limit_id” que não é nada mais do que evocar os dois procedimentos acima criados para a inserção dos registos nas duas tabelas, mas antes disso, é feito uma operação “delete” para apagar os registos dos campos da tabela “Credit_Limit” e “Person”. Em primeiro lugar faz-se o “delete” dos registos da tabela “Credit_Limit”, pois esta

contém a chave estrangeira que é a chave primária da tabela “Person” e só depois é que se faz o “delete” dos registos da tabela “Person”.

```
create or replace procedure
insert_tables_credit_limit_id is
begin
    delete credit_limit_id;
    delete person;

    insert_person_lookup;
    insert_creditLimitId_principal;
end;
/
```

5. Data Warehouse

Designa-se por Data Warehouse (DW) uma base de dados de apoio à decisão estratégica, guiada pelas necessidades da gestão. São bases de dados de grande dimensão e construídas a partir de bases de dados operacionais e de outros sistemas usados numa organização.

Na figura seguinte, está ilustrada o modelo Entidade-Relacionamento da minha Data Warehouse. Esta é composta no total por 6 tabelas, onde as tabelas que são da cor bege nas Dw’s chamam-se “Dimensões”, a tabela representada pela cor amarela chama-se nas Dw’s “Mini-Dimensão” e a tabela central de cor azul nas Dw’s chama-se de “Tabela de factos”, onde esta tem de ser a única na Dw normalizada.

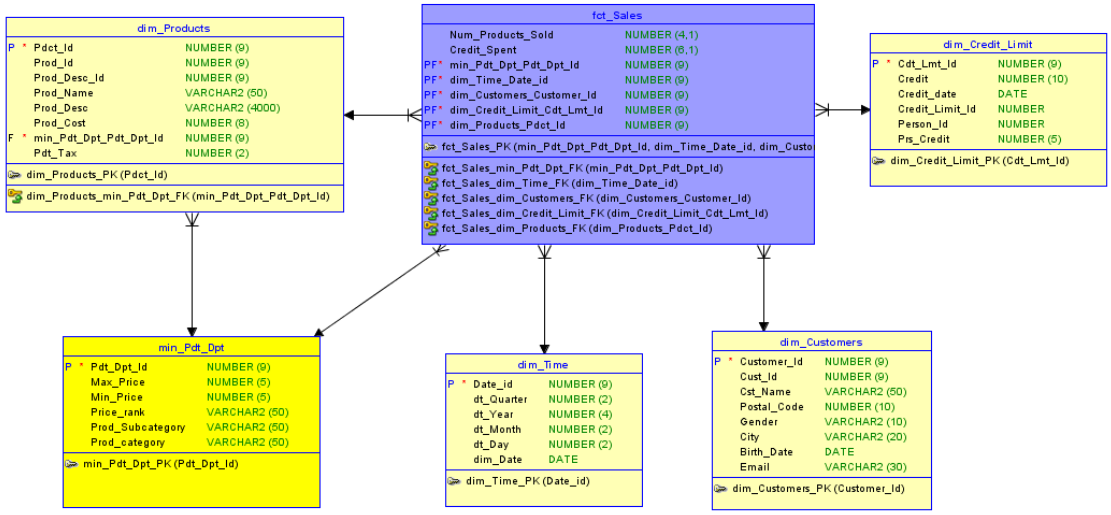

FIGURA2 - MODELO ER DA DW

6. Procedimentos para inserção de dados nas tabelas da Data Warehouse

6.1 Procedimento de inserção de dados na tabela "dim_Credit_Limit"

Neste subtópico pretende-se desenvolver um procedimento que insira dados na minha tabela chamada "dim_Credit_Limit" do modelo Data Warehouse (Figura2).

A base da construção deste procedimento chamado "insert_dim_creditLimit", é realizado uma operação "insert" baseada numa operação "select" entre duas tabelas da base de dados operacional, nomeadamente as tabelas "Credit_Limit_Id" e "Person". O que é de facto realizado neste procedimento, é selecionar e filtrar campos das duas tabelas acima referidas da base de dados operacional em comum com os campos da base de dados da DW e inserir nesta.

```
create or replace procedure insert_dim_creditLimit is

begin
insert into dim_Credit_Limit(Cdt_Lmt_Id, Credit, Credit_date,
Owned_by, Credit_Limit_Id, Person_Id,Prs_Credit)
select seq_pk.nextval, Credit_Limit_Id.Credit,
Credit_Limit_Id.Credit_date, Credit_Limit_Id.owner,
Credit_Limit_Id.Credit_Limit_Id, Person.Person_Id, Person.Credit
from Credit_Limit_Id, Person
where Credit_Limit_Id.Person_Person_Id = Person.Person_Id
and CREDIT_LIMIT_ID not in (
select CREDIT_LIMIT_ID
from dim_Credit_Limit
);
end;
```

6.2 Procedimento de inserção de dados na tabela "dim_Customers"

Neste subtópico, o objetivo é desenvolver um procedimento chamado "insert_dim_Customers" de modo a preencher os campos da tabela "dim_Customers" da base de dados Data Warehouse. Este procedimento baseia-se num "insert" baseado num "select", onde nesta última operação, vai-se buscar dados existentes nos atributos da tabela chamada "Customers" da base de dados operacional, nomeadamente os atributos: "cust_id", "cust_first_name", "cust_postal_code", "cust_gender", "cust_city", "cust_birth_date", "cust_email", estes que vão ser inseridos respetivamente nos campos da dimensão "dim_Customers", e esses campos são: "Customer_Id", "Cust_Id, Cst_Name", "Postal_Code", "Gender", "City", "Birth_Date", "Email".

Na seguinte caixa de texto vemos o procedimento completo para a inserção destes registos.

```
create or replace procedure insert_dim_Customers is

begin
insert into dim_Customers(Customer_Id, Cust_Id, Cst_Name,
Postal_Code, Gender, City, Birth_Date, Email)
select seq_dim_customers.nextval, CUST_ID, CUST_FIRST_NAME,
CUST_POSTAL_CODE, CUST_GENDER, CUST_CITY, CUST_BIRTH_DATE,
CUST_EMAIL
from CUSTOMERS
where (CUST_ID, CUST_FIRST_NAME, CUST_POSTAL_CODE, CUST_GENDER,
CUST_CITY, CUST_BIRTH_DATE, CUST_EMAIL)
not in (select Cust_Id, Cst_Name, Postal_Code, Gender, City,
Birth_Date, Email from dim_Customers);
end;
```

6.3 Procedimento de inserção de dados na tabela "dim_Time"

De modo a inserir dados na tabela dimensão "dim_Time", primeiramente o procedimento recebe dois parâmetros de entrada que são dois anos, um ano inicial e outro final. Sabemos que todos os anos começam no dia 1 de janeiro de "x" ano e terminam no dia 31 de dezembro do respetivo "x" ano, logo o raciocínio a seguir foi converter estas duas datas para juliano, concatenadas com os anos introduzidos nos parâmetros.

Seguidamente conto os "Date_Id" e guardo na variável "v_last_year", caso esta variável se encontre superior a zero é feito uma operação "select" que me devolve o maior ano que se encontra registado no atributo "dt_year" da tabela dimensão "dim_Time". Este "maior ano" é guardado novamente na variável "v_last_year".

Em ciclo, e se o ano da data inicial for maior do que o último ano (v_last_year) que se encontra registado na base de dados, continua-se a inserir dados até ao ponto de o ano da data inicial for igual ao ano da data final, isto para evitar dados repetidos na base de dados.

```
create or replace procedure insert_dim_data(p_start_year number,
p_end_year number) is
v_init_date number(9);
v_end_date number(9);
v_last_year number(4);
begin
 v_init_date :=
to_number(to_char(to_date('1/1/'||p_start_year,'dd/mm/yyyy'),'j'));
 v_end_date :=
to_number(to_char(to_date('31/12/'||p_end_year,'dd/mm/yyyy'),'j'));
select count(Date_id) into v_last_year from dim_Time;
if(v_last_year) > 0 then
    select max(dt_year) into v_last_year from dim_Time;
end if;
 loop
    if to_number(to_char(to_date(v_init_date,'j'), 'yyyy')) >
v_last_year then
        insert into dim_Time(Date_id, dt_Quarter, dt_Year, dt_Month,
dt_Day, dim_date)
        values(v_init_date, to_number(to_char(to_date(v_init_date,
'j'),'q')), to_number(to_char(to_date(v_init_date,'j'),'yyyy')),
to_number(to_char(to_date(v_init_date,'j'),'mm')),
to_number(to_char(to_date(v_init_date,'j'),'dd')),
to_date(v_init_date,'j'));
    end if;
    exit when v_init_date = v_end_date;
    v_init_date := v_init_date + 1;
 end loop;
end;
```

6.4 Procedimento de inserção de dados na tabela "min_Pdt_Dpt"

Na realização do procedimento responsável por inserir dados na tabela mini-dimensão "min_Pdt_Dpt" da base de dados Data Warehouse, em primeiro lugar é criado um cursor que aponta para os dados: "Prod_Subcategory", "Prod_Category" e "Prod_List_Price" das tabelas da base de dados operacional: "Sub_Categories", "Categories" e "Products", respetivamente. Estes atributos vão ser transferidos para os campos: "Prod_Subcategory", "Prod_Category" e "Prod_List_Price" da tabela mini-dimensão.

Neste procedimento foi necessário realizar uma função chamada "Get_Price_Rank", esta que recebe um parâmetro (p_price) do tipo "number", onde é inserido um valor de um preço. E o que a função retorna é um dado tipo "varchar", que conforme o preço que é inserido no parâmetro, e os intervalos indicados na função, esta diz-nos se o preço inserido é: "Cheap", "Medium price" ou "Expensive", ou seja, diz-nos a que categoria corresponde determinado preço inserido. Na caixa de texto seguinte vê-se a função descrita.

```
create or replace function get_price_rank(p_price number)
return varchar2 is

v_price_rank varchar2(50);

begin

if 0 <= p_price and  p_price <= 250 then
    v_price_rank := 'Cheap';
else if 250 <= p_price and  p_price <= 700 then
    v_price_rank := 'Medium price';
else
    v_price_rank := 'Expensive';
end if;
end if;

return v_price_rank;
end;
```

Voltando ao procedimento, é chamada a função “Get_Price_Rank” e o valor retornado por esta é guardado na variável “v_price_rank”, variável esta que de seguida vai-se enquadrar em intervalos definidos no procedimento, estes intervalos vão estipular e preencher a gama de dados: “min_price” e “max_price” da tabela mini-dimensão.

Para evitar dados repetidos antes da inserção na tabela mini-dimensão “min_Pdt_Dpt”, é feito uma operação “count” guardada numa variável de controlo (v_controlo) para saber o número total de registos que já existem na mini-dimensão. Se o número retornado pela operação “count” for zero, então são inseridos os dados na mini-dimensão.

Na caixa de texto seguinte é mostrado o procedimento capaz de inserir dados na tabela mini-dimensão.

```
create or replace procedure insert_mdim_Pdt_Dpt is

cursor c_mdim is
select Prod_subcategory, Prod_category, Prod_list_price
from sub_categories, categories, products
where (Prod_subcategory, Prod_category, Prod_list_price) not in
(select Prod_subcategory, Prod_category, Prod_list_price from
min_Pdt_Dpt)
and Sub_categories.Cat_id = Categories.Cat_id
and Products.sub_cat_id = sub_categories.sub_cat_id;

v_cur_data c_mdim%ROWTYPE;
v_max_price min_Pdt_Dpt.max_price%TYPE;
v_min_price min_Pdt_Dpt.min_price%TYPE;
v_price_rank min_Pdt_Dpt.price_rank%TYPE;
v_controlo number;

begin
open c_mdim;
loop
    fetch c_mdim into v_cur_data;
    exit when c_mdim%NOTFOUND;

    v_price_rank := get_price_rank(v_cur_data.PROD_LIST_PRICE);

    if v_price_rank = 'Cheap' then
        v_min_price := 0;
        v_max_price := 250;
    else if v_price_rank = 'Medium price' then
        v_min_price := 250;
        v_max_price := 700;
    else if v_price_rank = 'Expensive' then
        v_min_price := 700;
        v_max_price := 1200;
    end if;
    end if;
    end if;

    select count(*) into v_controlo
    from min_Pdt_Dpt
    where Prod_Category = v_cur_data.Prod_category
    and Prod_subcategory = v_cur_data.Prod_subcategory
    and Price_rank =  v_price_rank;
    if v_controlo = 0 then
        insert into
min_Pdt_Dpt(Pdt_Dpt_Id,Prod_subcategory,Prod_category, Max_Price,
Min_Price, Price_rank)
        values(seq_mdim_pdt_dcp.NEXTVAL,
v_cur_data.Prod_subcategory, v_cur_data.Prod_category, v_max_price,
v_min_price, v_price_rank);
    end if;

end loop;
close c_mdim;
end;
```

6.5 Procedimento de inserção de dados na tabela "dim_Products"

Na elaboração deste procedimento, temos que ter em conta que realizámos este procedimento em segundo lugar, pois a tabela "dim_Products" tem chave estrangeira que é preenchida com a chave primária da tabela "min_Pdt_Dpt", tabela que já se encontra preenchida com dados (ver subtópico 6.4).

Em auxílio ao procedimento foi elaborada uma função chamada "GetOuttrigger", que resolve o problema de preencher a chave estrangeira da tabela "dim_Products". Esta função desempenha o papel de ir buscar a chave primária "Pdt_Dpt_Id" da tabela "min_Pdt_Dpt", mas para isto ocorrer, no parâmetro desta função é passado um Id de um produto "Prod_Id". Nesta mesma função é evocada a função "Get_Price_Rank", onde é passado como parâmetro uma variável onde está armazenada o atributo "Prod_List_Price" da tabela "Products" da base de dados operacional, para sabermos o "Price_rank" daquele determinado "Prod_Id". De seguida é feito um select à tabela "min_Pdt_Dpt" para ir buscar a chave primária e enviá-la para a variável "v_outtigger", variável retornada pela função "GetOuttrigger".

```
create or replace function getOutTrigger(p_prod_id number)
return number is

v_outtrigger number;
v_prod_subcategory sub_categories.prod_subcategory%TYPE;
v_prod_category categories.prod_category%TYPE;
v_prod_list_price products.prod_list_price%TYPE;
v_price_rank min_Pdt_Dpt.price_rank%TYPE;

begin

select prod_subcategory, prod_category, prod_list_price
into v_prod_subcategory, v_prod_category, v_prod_list_price
from sub_categories, categories, products
where sub_categories.sub_cat_id = products.sub_cat_id
and categories.cat_id = sub_categories.cat_id
and products.prod_id = p_prod_id;

v_price_rank := get_price_rank(v_prod_list_price);

select Pdt_Dpt_Id
into v_outtrigger
from min_Pdt_Dpt
where v_prod_subcategory = prod_subcategory
and v_prod_category = prod_category
and v_price_rank = price_rank;

return v_outtrigger;
end;
```

Voltando ao procedimento, em ciclo, é guardado o resultado da função "GetOuttrigger" na variável "v_outtrigger". De seguida são inseridos os registos na tabela "dim_Products". Na caixa de texto seguinte, encontra-se o procedimento chamado "insert_dim_products".

```
create or replace procedure insert_dim_Products is

cursor c_dim_products is
select Prod_Id, Prod_Desc_Id, Prod_Name, Prod_Desc, Prod_Cost,
Tax_Pct
from Products, Product_Descriptions
where Prod_Desc_Id = Prod_Descriptions_Id
and (Products.Prod_Id, Product_Descriptions.Prod_Desc_Id)
not in (select Prod_Id, Prod_Desc_Id from dim_Products);

v_cur_data c_dim_products%ROWTYPE;
v_outtrigger dim_Products.min_Pdt_Dpt_Pdt_Dpt_Id%TYPE;

begin
    open c_dim_products;

    loop
        fetch c_dim_products into v_cur_data;
        exit when c_dim_products%NOTFOUND;

        v_outtrigger := getouttrigger(v_cur_data.Prod_Id);

        insert into dim_products(Pdct_Id, Prod_Id, Prod_Desc_Id,
Prod_Name, Prod_Desc, Prod_Cost, min_Pdt_Dpt_Pdt_Dpt_Id, Pdt_Tax)
        values(seq_dim_products.nextval, v_cur_data.Prod_Id,
v_cur_data.Prod_Desc_Id, v_cur_data.Prod_Name, v_cur_data.Prod_Desc,
v_cur_data.Prod_Cost, v_outtrigger, v_cur_data.Tax_Pct);
    end loop;

    close c_dim_products;
end;
```

6.6 Procedimento de inserção de dados na tabela "fct_Sales"

Neste subtópico, foi criado um procedimento capaz de inserir dados na tabela dos factos, "fct_Sales" da Data Warehouse. Esta tabela é constituída apenas por dois factos que têm de respeitar a granularidade. Os factos que constituem a minha tabela dos factos são: "Num_Products_Sold" e "Credit_Spent". O resto dos atributos da tabela dos factos são chaves concatenadas, chaves essas provenientes dos relacionamentos entre dimensões e mini-dimensão. Para que estas chaves da tabela dos factos sejam preenchidas, foram criadas as funções "Get_Dim_Credit_Limit_Id", "Get_Dim_Customers_Id" e "Get_Dim_Products_Id", que recebem como parâmetros as chaves primárias da base de dados operacional. Caso estas chaves primárias da bd operacional sejam idênticas às "surrogate key's" das tabelas da Data Warehouse, a função devolve uma variável que vai ser preenchida na chave concatenada da tabela dos factos.

```
create or replace function
get_dim_Credit_Limit_id(p_credit_limit_id
Credit_Limit_Id.Credit_Limit_Id%TYPE)
return number is

v_dim_Credit_Limit_Id dim_Credit_Limit.Cdt_Lmt_Id%TYPE;

begin
select Cdt_Lmt_Id into v_dim_Credit_Limit_Id
from dim_Credit_Limit
where dim_Credit_Limit.Credit_Limit_Id = p_credit_limit_id;

return v_dim_Credit_Limit_Id;

end;
```

```
create or replace function get_dim_Customers_id(p_customers_id
Customers.Cust_Id%TYPE)
return number is

v_dim_Customers_Id dim_Customers.Customer_Id%TYPE;

begin
select Customer_Id into v_dim_Customers_Id
from dim_Customers
where dim_Customers.Cust_Id = p_customers_id;

return v_dim_Customers_Id;

end;
```

```
create or replace function get_dim_Products_id(p_product_id
Products.Prod_Id%TYPE)
return number is

v_dim_Product_Id dim_Products.Pdct_Id%TYPE;

begin
select Pdct_Id into v_dim_Product_Id
from dim_Products
where dim_Products.Prod_Id = p_product_id;

return v_dim_Product_Id;

end;
```

A função “Get_Dim_Time_Id” recebe como parâmetro uma data válida. Caso esta data for igual à data do campo “dim_Date” da tabela “dim_Time”, a função retorna esta mesma data numa variável.

```
create or replace function get_dim_time_id(p_date date)
return number is

v_dim_time_id dim_time.Date_Id%TYPE;

begin

select Date_Id into v_dim_time_id
from dim_Time
where dim_time.dim_Date = trunc(p_date);

return v_dim_time_id;
end;
```

A função “Get_Min_Pdt_Dpt” recebe como parâmetro de entrada a chave primária da tabela “dim_Products”, ou seja, recebe um id de um produto. Caso a chave estrangeira da tabela “dim_Products” for igual ao parâmetro da função, esta devolve a variável que contém guardada a própria chave estrangeira.

```
create or replace function get_min_Pdt_Dpt(p_product_id
dim_Products.Pdct_Id%TYPE)
return number is

v_min_Pdt_Dpt_Fk min_Pdt_Dpt.Pdt_Dpt_Id%TYPE;

begin
select min_Pdt_Dpt_Pdt_Dpt_Id into v_min_Pdt_Dpt_Fk
from dim_products
where dim_Products.Pdct_Id = p_product_id;

return v_min_Pdt_Dpt_Fk;

end;
```

Com todas estas funções construídas, já temos realizada a ideia de preencher as chaves concatenadas da tabela dos factos.

De volta ao procedimento de inserção na tabela “fct_Sales”, é criado um cursor que aponta para os dados principais, da respetiva operação “select” que, na minha opinião, tem de estar de acordo e respeitar a granularidade da DW. Em seguida, dentro do ciclo, na operação “insert” é onde chamo todas as funções acima citadas, estas responsáveis por inserir os dados de acordo com os campos na tabela dos factos.

```
create or replace procedure insert_fct is

cursor c_fct is select sum(quantity_sold) as quantity_sold,
sum(Credit) as Credit, credit_limit_id.Credit_Limit_Id,
trunc(sale_date) as sale_date, products.prod_id, cust_id
from sales_rows, sales, products, credit_limit_id
where sales_rows.sale_id = sales.sale_id
and products.prod_id = sales_rows.prod_id
and sales.Credit_Limit_Id = Credit_Limit_Id.Credit_Limit_Id
and sales.Credit_Limit_Id is not null
group by trunc(sale_date), products.prod_id, cust_id,
credit_limit_id.Credit_Limit_Id
order by trunc(sale_date);

cur_data c_fct%ROWTYPE;
v_prod_func dim_Products.Pdct_Id%TYPE;

begin
    open c_fct;
    loop
        fetch c_fct into cur_data;
        exit when c_fct%NOTFOUND;

        v_prod_func := get_dim_products_id(cur_data.prod_id);

        insert into fct_Sales(Num_Products_Sold, Credit_Spent,
min_Pdt_Dpt_Pdt_Dpt_Id, dim_Time_Date_id, dim_Customers_Customer_Id,
dim_Credit_Limit_Cdt_Lmt_Id, dim_Products_Pdct_Id)
        values(cur_data.quantity_sold, cur_data.Credit,
get_min_pdt_dpt(v_prod_func),
        get_dim_time_id(cur_data.sale_date),
get_dim_customers_id(cur_data.cust_id),
        get_dim_credit_limit_id(cur_data.Credit_Limit_Id),
        v_prod_func);

    end loop;
end;
```

7. *Cálculo do espaço ocupado*

Neste tópico é mostrado o cálculo do espaço que ocupa a minha Data Warehouse:

- Granularidade: dim_Products X dim_Customers X dim_Credit_Limit X dim_Time
- Tempo: 4 anos
- Nº de Produtos: 200.000 (apenas 30% dos produtos são vendidos diariamente em cada loja)
- Lojas: 200
- Tamanho médio de registo: 7 atributos * 4 bytes = 28 bytes
- Nº de registos de factos: 4 * 365 * (200.000 * 0.3) * 200 = 17.520.000.000
- Tamanho aproximado da DW: 28 * 17.520.000.000 = 490.560.000.000 ≅ 49 GBytes

8. Hierarquias e Personalização

No Discoverer Administrator foi possível carregar a minha DW para fins de dados estatísticos e alterações nos nomes dos meus campos e tabelas, a nível do gestor. O nome do negócio que lhe atribuí foi "Shoes Clothes Business Area", tal como mostra a seguinte imagem é possível ver o nome do negócio e as minhas tabelas constituintes da DW.

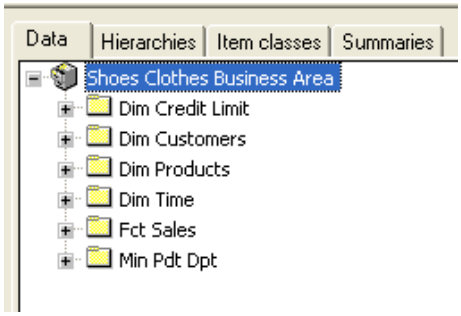

FIGURA 3 – NOME DO NEGÓCIO

Na seguinte imagem, é possível ver uma nova hierarquia chamada "ItemProductsHierarchy" criada no Discoverer que tem como bases: a categoria do produto, a subcategoria do produto e, por fim, o nome do produto.

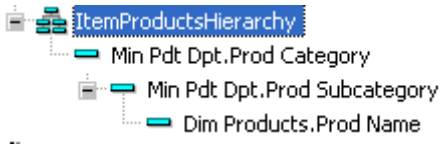

FIGURA 4 – HIERARQUIA CRIADA NO DISCOVERER ADMINISTRATOR

Na aplicação Discoverer também é possível alterar os nomes de campos das tabelas ou até os próprios nomes das tabelas, isto para que o gestor tenha melhor perceção dos dados que quer extrair. Como exemplo, nas seguintes imagens, pode-se ver um dos atributos da tabela "dim_Products" alterado:

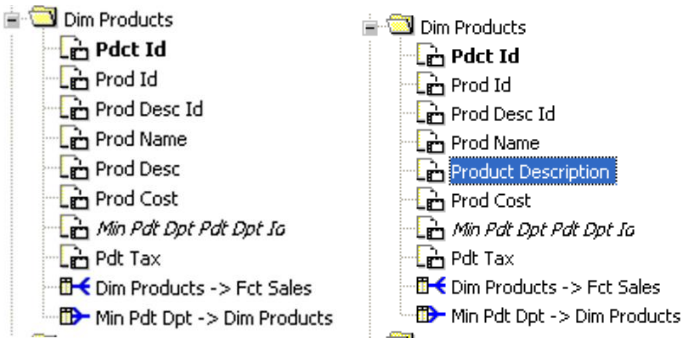

FIGURA 5 – ALTERAÇÃO DO CAMPO "PROD DESC"

9. Workbooks

Neste tópico, é onde abordamos diretamente questões estatísticas, logísticas e respostas a perguntas complexas de modo que para responder às mesmas temos ajudas a partir de vários gráficos.

Este tópico situa-se no patamar de quem está à frente na gestão dos recursos de uma empresa, a quem se propõe a estudar o mercado da sua área para que possa se sentir em vantagem à concorrência, tudo isto, com a ajuda da ferramenta Discoverer Desktop.

9.1 Perguntas e análise de resultados

Pergunta 1: Quais as cidades e as subcategorias dos produtos vendidos, onde o crédito gasto nas subcategorias dos produtos situa-se entre 5$ e 20$, e os trimestres encontram-se entre o primeiro e o segundo do ano?

Para que o crédito gasto seja entre 5$ e 20$ e os trimestres do ano se encontrem entre o primeiro e o segundo do ano, no Discoverer Desktop na funcionalidade "conditions" adicionamos lá estas duas condições como mostra a seguinte figura:

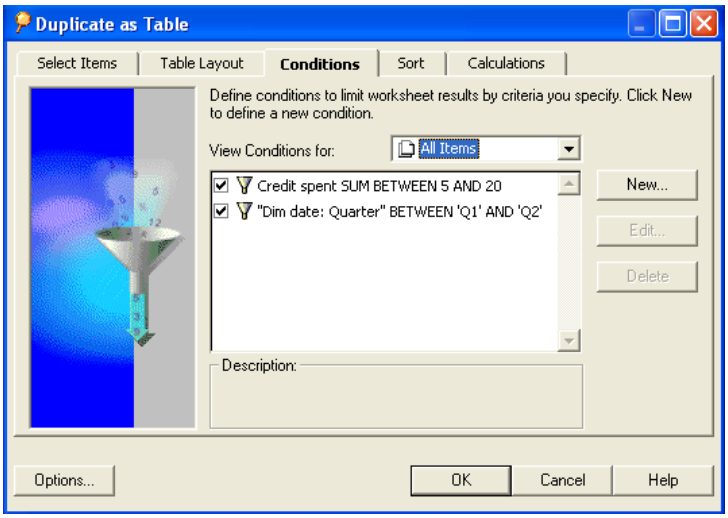

FIGURA 6 – CONDIÇÕES DE ENTRADA PARA RESOLVER A QUESTÃO 1

Em resultado disto obtemos a tabela seguinte com os dados filtrados e que queremos de facto:

	City	Product subcategory	Quarter	Credit spent SUM	Dim date
1	Honomu	Shorts - Boys	Q2	$6	25-JUN-1999
2	Killarney	Sportcoats - Men	Q1	$12	21-MAR-2000
3	Murnau	Trousers - Men	Q1	$8	05-FEB-1998
4	Tilburg	Sweaters - Girls	Q1	$6	21-JAN-1999
5	Honomu	Shorts - Girls	Q2	$6	25-JUN-1999
6	Warstein	Casual Shirts - Men	Q1	$8	12-JAN-1999
7	Goodhope	Sweaters - Boys	Q2	$7	01-MAY-2000
8	Joinville	Shirts And Jackets - Women	Q2	$8	28-APR-1998

FIGURA 7 – TABELA DE RESULTADOS À PRIMEIRA QUESTÃO

Caso queiramos ver um gráfico detalhado acerca desta informação, o software Discoverer Desktop possui vários tipos de gráficos para visualizarmos a nossa informação de um modo mais facilitado.

O gráfico que vê-mos a seguir, é um gráfico de barras que representa as cidades, subcategoria dos produtos e o trimestre em função do crédito gasto:

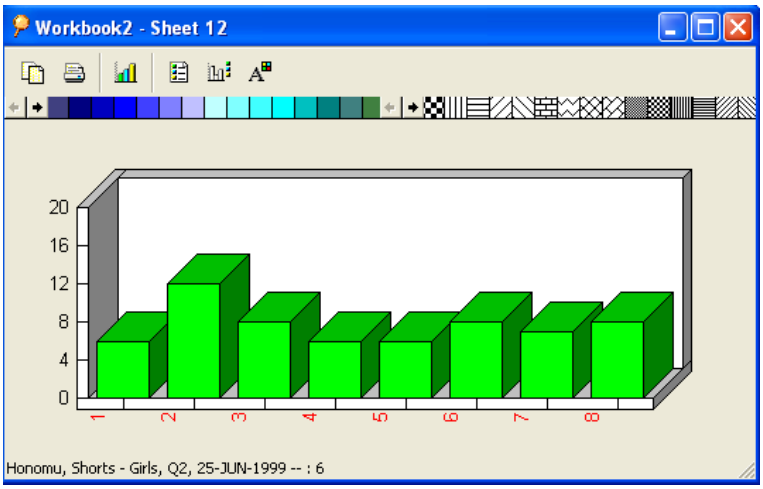

FIGURA 7 – GRÁFICO DE RESULTADOS DA DA QUESTÃO 1

Pergunta2: Qual é a soma do número de produtos vendidos, no mês de maio do ano de 1999, evidenciando os respetivos nomes dos produtos?

Para que o mês de maio e o ano de 1999 seja respeitado conforme a pergunta é introduzido nas condições esses mesmos parâmetros, tal como mostra a seguinte figura:

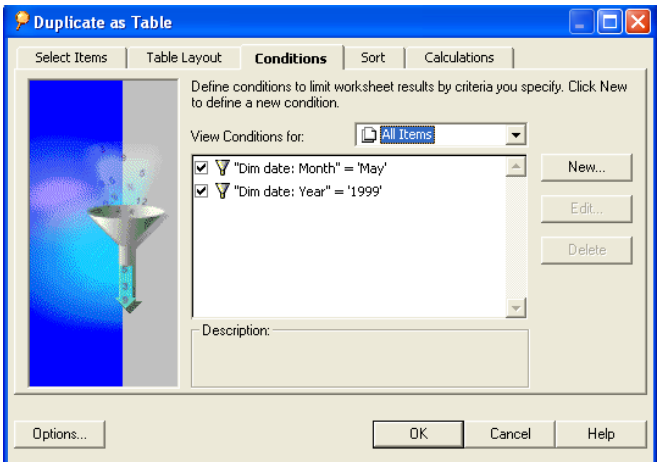

FIGURA 8 – CONDIÇÕES DE ENTRADA PARA RESOLVER A QUESTÃO 2

De seguida vemos o resultado das condições acima descritas na forma de uma tabela:

	Product name	Month	Year	Date	Num products sold SUM
1	Russell Jersey Trousers Kids	May	1999	03-MAY-1999	3.0
2	Sleepwear & Pajamas - Pink Roses Girls Robe	May	1999	16-MAY-1999	32.0
3	Russell Jersey Trousers Kids	May	1999	16-MAY-1999	25.0
4	Citizen Of The World! Flag T-Shirt - Children	May	1999	16-MAY-1999	17.0
5	Rabbit Jacket	May	1999	16-MAY-1999	1.0
6	Fagonnable Cotton Drawstring Trouser	May	1999	16-MAY-1999	18.0

FIGURA 9 – TABELA DE RESULTADOS DA QUESTÃO 2

A informação desta tabela foi extraída para o seguinte gráfico em forma de "Pie", onde cada cor distinta representa um nome de um produto e a sua percentagem em vendas:

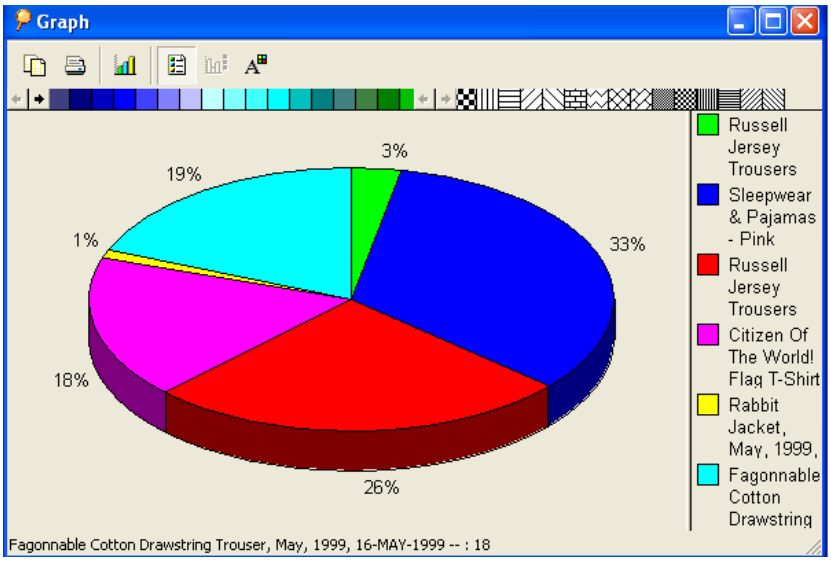

FIGURA 10 – GRÁFICO DE RESULTADOS DA QUESTÃO 2

A. Anexos

A1. Código de criação das tabelas extra

Para a tabela "Person":

```
CREATE TABLE person (
    person_id NUMBER(9) NOT NULL,
    credit    NUMBER
);

ALTER TABLE person ADD CONSTRAINT person_pk
PRIMARY KEY ( person_id );
```

Para a tabela "Credit_Limit":

```
CREATE TABLE credit_limit_id (
    credit_limit_id  NUMBER NOT NULL,
    credit           NUMBER(9),
    credit_date      DATE,
    owner            VARCHAR2(10),
    person_person_id NUMBER(9) NOT NULL
);

ALTER TABLE credit_limit_id ADD CONSTRAINT
credit_limit_id_pk PRIMARY KEY (
credit_limit_id );

ALTER TABLE credit_limit_id
    ADD CONSTRAINT credit_limit_id_person_fk
FOREIGN KEY ( person_person_id )
        REFERENCES person ( person_id );
```

Permissão ao utilizador “bdii_ei_1703826” para fazer operações de “selects” às tabelas “Person” e “Credit_Limit”, respetivamente:

```
grant select on person to bdii_ei_1703826;
grant select on credit_limit_id to bdii_ei_1703826;
```

A2. Scrip de criação das tabelas da Data Warehouse

```
DROP TABLE dim_credit_limit CASCADE CONSTRAINTS;
DROP TABLE dim_customers CASCADE CONSTRAINTS;
DROP TABLE dim_products CASCADE CONSTRAINTS;
DROP TABLE dim_time CASCADE CONSTRAINTS;
DROP TABLE fct_sales CASCADE CONSTRAINTS;
DROP TABLE min_pdt_dpt CASCADE CONSTRAINTS;


CREATE TABLE dim_credit_limit (
    cdt_lmt_id      NUMBER(9) NOT NULL,
    credit          NUMBER(10),
    credit_date     DATE,
    credit_limit_id NUMBER,
    person_id       NUMBER,
    prs_credit      NUMBER(5)
);

ALTER TABLE dim_credit_limit ADD CONSTRAINT dim_credit_limit_pk PRIMARY KEY (
cdt_lmt_id );

CREATE TABLE dim_customers (
    customer_id NUMBER(9) NOT NULL,
    cust_id     NUMBER(9),
    cst_name    VARCHAR2(50),
    postal_code NUMBER(10),
    gender      VARCHAR2(10),
    city        VARCHAR2(20),
    birth_date  DATE,
    email       VARCHAR2(30)
);

ALTER TABLE dim_customers ADD CONSTRAINT dim_customers_pk PRIMARY KEY (
customer_id );

CREATE TABLE dim_products (
    pdct_id                 NUMBER(9) NOT NULL,
    prod_id                 NUMBER(9),
    prod_desc_id            NUMBER(9),
    prod_name               VARCHAR2(50),
    prod_desc               VARCHAR2(4000),
    prod_cost               NUMBER(8),
    min_pdt_dpt_pdt_dpt_id NUMBER(9) NOT NULL,
    pdt_tax                 NUMBER(2)
);

ALTER TABLE dim_products ADD CONSTRAINT dim_products_pk PRIMARY KEY ( pdct_id
);

CREATE TABLE dim_time (
    date_id    NUMBER(9) NOT NULL,
    dt_quarter NUMBER(2),
    dt_year    NUMBER(4),
    dt_month   NUMBER(2),
    dt_day     NUMBER(2),
    dim_date   DATE
);

ALTER TABLE dim_time ADD CONSTRAINT dim_time_pk PRIMARY KEY ( date_id );
```

```
CREATE TABLE fct_sales (
    num_products_sold           NUMBER(4, 1),
    credit_spent                NUMBER(6, 1),
    min_pdt_dpt_pdt_dpt_id      NUMBER(9) NOT NULL,
    dim_time_date_id            NUMBER(9) NOT NULL,
    dim_customers_customer_id   NUMBER(9) NOT NULL,
    dim_credit_limit_cdt_lmt_id NUMBER(9) NOT NULL,
    dim_products_pdct_id        NUMBER(9) NOT NULL
);

ALTER TABLE fct_sales
    ADD CONSTRAINT fct_sales_pk PRIMARY KEY ( min_pdt_dpt_pdt_dpt_id,
                                              dim_time_date_id,
                                              dim_customers_customer_id,
                                              dim_credit_limit_cdt_lmt_id,
                                              dim_products_pdct_id );

CREATE TABLE min_pdt_dpt (
    pdt_dpt_id       NUMBER(9) NOT NULL,
    max_price        NUMBER(5),
    min_price        NUMBER(5),
    price_rank       VARCHAR2(50),
    prod_subcategory VARCHAR2(50),
    prod_category    VARCHAR2(50)
);

ALTER TABLE min_pdt_dpt ADD CONSTRAINT min_pdt_dpt_pk PRIMARY KEY (
pdt_dpt_id );

ALTER TABLE dim_products
    ADD CONSTRAINT dim_products_min_pdt_dpt_fk FOREIGN KEY (
min_pdt_dpt_pdt_dpt_id )
        REFERENCES min_pdt_dpt ( pdt_dpt_id );

ALTER TABLE fct_sales
    ADD CONSTRAINT fct_sales_dim_credit_limit_fk FOREIGN KEY (
dim_credit_limit_cdt_lmt_id )
        REFERENCES dim_credit_limit ( cdt_lmt_id );

ALTER TABLE fct_sales
    ADD CONSTRAINT fct_sales_dim_customers_fk FOREIGN KEY (
dim_customers_customer_id )
        REFERENCES dim_customers ( customer_id );

ALTER TABLE fct_sales
    ADD CONSTRAINT fct_sales_dim_products_fk FOREIGN KEY (
dim_products_pdct_id )
        REFERENCES dim_products ( pdct_id );

ALTER TABLE fct_sales
    ADD CONSTRAINT fct_sales_dim_time_fk FOREIGN KEY ( dim_time_date_id )
        REFERENCES dim_time ( date_id );

ALTER TABLE fct_sales
    ADD CONSTRAINT fct_sales_min_pdt_dpt_fk FOREIGN KEY (
min_pdt_dpt_pdt_dpt_id )
        REFERENCES min_pdt_dpt ( pdt_dpt_id );
```

BIBLIOGRAFIA

[1] J. Fonseca, “Bases de Dados II,” *Data Warehousing e OLAP,* pp. 5, 6, 27, 28, 67, 89, 90, 91, 110,, 2022.

[2] J. Fonseca, “TRABALHO PRÁTICO DE BASES DE DADOS II,” *PROJECTO DE DATAWAREHOUSE,* pp. 6,7,8, 22 03 2022.

[3] Google, “JamBoard,” Google, 2022. [Online]. Available: https://jamboard.google.com. [Acedido em 07 04 2022].

[4] Microsoft, “SQL Server 2019: Requisitos de hardware e software,” 2022. [Online]. Available: https://docs.microsoft.com/pt-br/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server-ver15?view=sql-server-ver15. [Acedido em 20 03 2022].

[5] Oracle, “SQL Functions,” Oracle, 2022. [Online]. Available: https://docs.oracle.com/cd/B19306_01/server.102/b14200/functions001.htm. [Acedido em 15 04 2022].

[6] Oracle, “Oracle,” Oracle, 2022. [Online]. Available: https://www.oracle.com/database/technologies/appdev/sqldeveloper-landing.html. [Acedido em 14 03 2022].