

**ETL- Trabalho Prático 2/3**

Ricardo Gonçalves – 22005012

Miguel Carreta – 21901101

SSD | LEI | 7/05/2022

# **Exercício 1**

1.2

Através do step “*Table Input*” com uma *conection* à base de dados “Pégasus”, gerada pelo script disponibilizado de onde, no *select* especificámos os parâmetros para efetuar a transformação à tabela de dimensões “*Customers*” de acordo com o enunciado.

Criou-se um “*Dimension Lookup/Update*”, conectou-se esta à base de dados “*FoodMart*” e especificou-se o *schema* e a tabela de destino desejados.

Posteriormente identificou-se a *key* da tabela de origem e estabeleceu-se correspondência para a transformação. Especificou-se os *fields* a serem atualizados. No ano mínimo no parâmetro Date range start field, colocámos como Min. Year 2011

Uma imagem com texto

Descrição gerada automaticamenteDe acordo com o enunciado do exercício foi identificada como “*Surrogate Key*” o *field* “*CustomerKey*”.

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com mesa

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamentePor fim executou-se o código SQL e consequentemente a transformação.

1.3

Como de acordo com o enunciado “Não é relevante manter o histórico da informação ao nível das características da dimensão”, a técnica de Slowly Changing Dimensions a adotar foi a de ***punch through***.

Esta tipo substitui todos os antigos valores de informação pelos novos valores. Desta forma a atualização da tabela de dimensões é simples e o seu tamanho nunca irá aumentar, mantendo sempre os registos mais recentes.

# **Exercício 2**

2.2

Criaram-se dois “*Table Inputs*” com uma *conection* à base de dados “Pégasus”, gerada pelo script disponibilizado de onde, através de um *select* especificámos os parâmetros para efetuar as transformações às tabelas de dimensões “*Products*” e “*ProductCategories*”.

Em ambas as tabelas de input aplicou-se um *sort* pelo parâmetro “*CategoryID*”. Efetuou-se então um “*Merge join*” que aplica um “*Inner join*” nessas tabelas.

Criou-se um “*Dimension Lookup/Update*”, conectou-se esta à base de dados “*FoodMart*” e especificou-se o *schema* e a tabela de destino desejados.

Posteriormente identificou-se a *key* da tabela de origem e estabeleceu-se correspondência para a transformação. Especificou-se os *fields* a serem atualizados e o respetivo tipo de transformação ***punch******through*** e ***insert***.

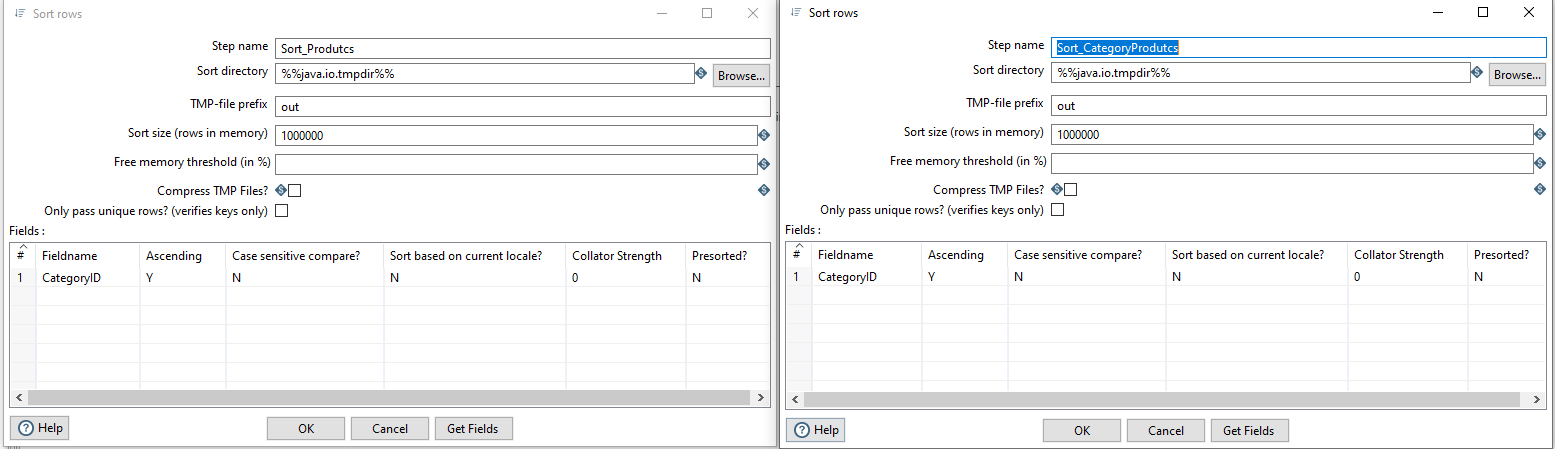
De acordo com o enunciado do exercício foi identificada como “*Surrogate Key*” o *field* “*ProductKey*”.

Por fim executou-se o código SQL e consequentemente a transformação.

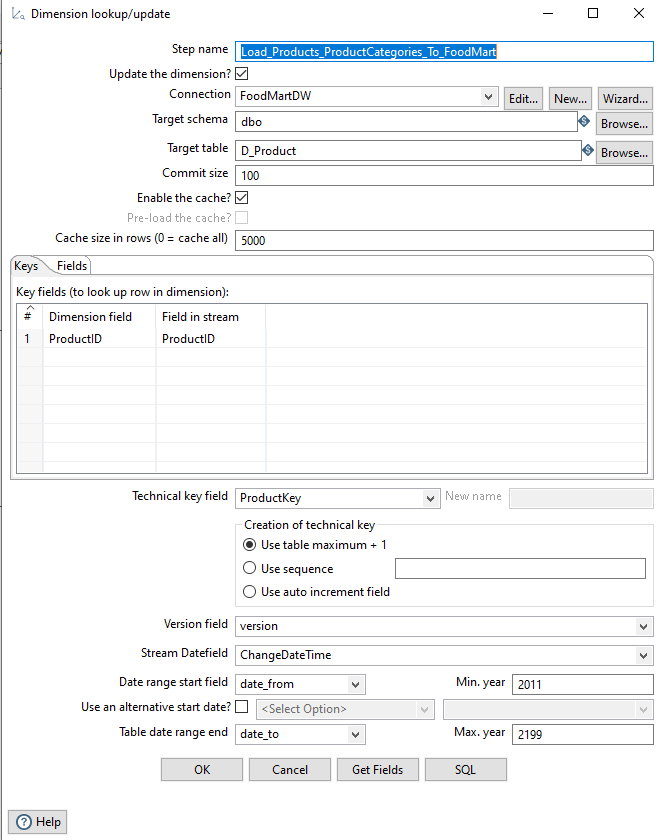
Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

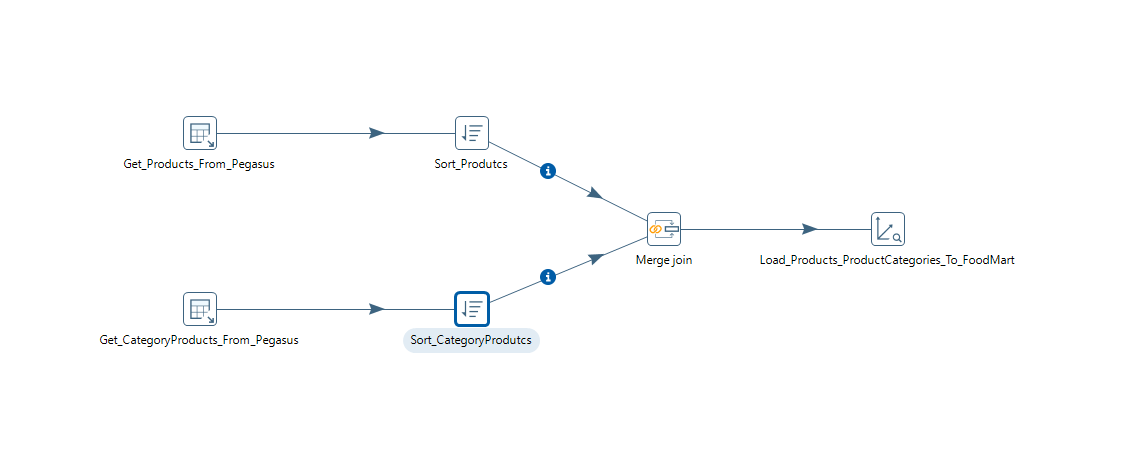


Uma imagem com mesa

Descrição gerada automaticamente

Uma imagem com mesa

Descrição gerada automaticamente



2.3

Como é referenciado no enunciado deste exercício apenas é relevante manter o histórico das informações relativas ao “*UnitCosPrice*” e ao “*UnitPrice*”, assim utilizou-se uma *Slowly Changing Dimension* de tipo 1 e 2 (equivale ao *punch through* e *insert* na aplicação *Spoon*). Foi aplicado o new nos campos UnitCostPrice e UnitPrice e nos restantes punch through, deste modo caso apareçam novos dados, apenas quando um dos campos (UnitCostPrice, UnitPrice) é diferente é criado um novo registo, para os restantes fields apenas é atualizado o registo.

# **Exercício 3**

3.2

Criaram-se dois “*Table Inputs*” com uma *conection* à base de dados “Pégasus”, gerada pelo script disponibilizado de onde, através de um *select* especificámos os parâmetros para efetuar as transformações às tabelas de dimensões “*Orders*” e “*OrdersDetails*”.

De seguida foi utilizado um Sort e um Merge Join para agregar a informação das duas tabelas sobre uma chave comum (OrderID).

Foi implementado um step do spoon “Calculator” para gerar os respectivos factos e por fim, o output é transformado e inserido na respectiva tabela através de um “table output”.

Uma imagem com texto

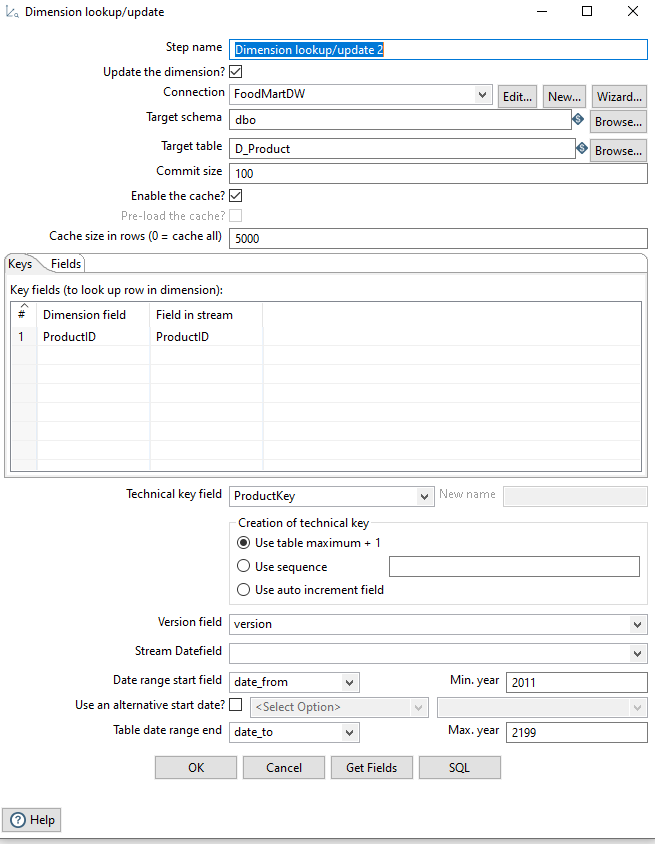
Descrição gerada automaticamentePor fim executou-se o código SQL e consequentemente a transformação.

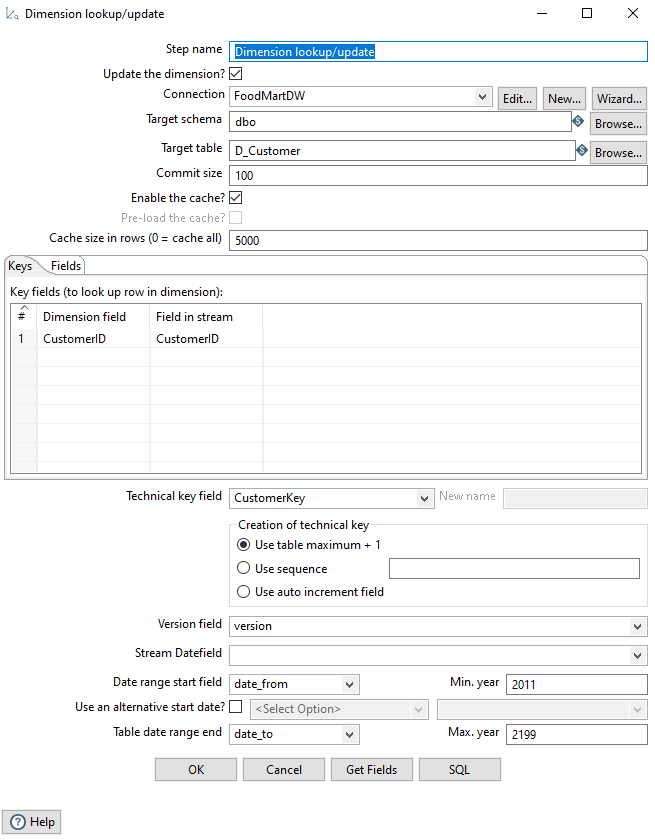
Uma imagem com texto, captura de ecrã, interior

Descrição gerada automaticamente

Uma imagem com mesa

Descrição gerada automaticamente





Uma imagem com mesa

Descrição gerada automaticamente

Uma imagem com mesa

Descrição gerada automaticamente

# **Exercício 4**

4.1

SELECT TOP 1 p.productname AS Nome, Max(( netsalesamount + salesamount - unitprice ) \* salesqty) AS 'Produto Mais Rentável'

FROM dbo.d\_product AS p

INNER JOIN dbo.f\_sales\_month AS f

ON p.productid = f.productkey

GROUP BY p.productname;

Uma imagem com texto

Descrição gerada automaticamente

4.2

SELECT DISTINCT d.Country AS 'Pais',

FROM dbo.D\_Customer d

INNER JOIN dbo.F\_Sales\_Month AS c

ON c.customerkey = d.CustomerKey

WHERE MonthKey BETWEEN 20116 AND 20117;

Uma imagem com texto

Descrição gerada automaticamente

4.3

SELECT distinct productname AS 'Nome do Produto', unitprice AS 'UnitPrice'

FROM dbo.D\_Customer

INNER JOIN dbo.F\_Sales\_Month AS c

ON c.customerkey = c.customerkey

INNER JOIN dbo.d\_product AS c1

ON c1.productkey = c1.productkey

where c.SalesAmount > 0 AND c.SalesAmount is not null AND unitprice > 100;

Uma imagem com texto

Descrição gerada automaticamente

4.4

SELECT distinct TOP 3 productname AS 'Nome do Produto', unitprice AS 'UnitPrice'

FROM dbo.D\_Customer

INNER JOIN dbo.F\_Sales\_Month AS c

ON c.customerkey = c.customerkey

INNER JOIN dbo.d\_product AS c1

ON c1.productkey = c1.productkey

where c.SalesAmount > 0 AND c.SalesAmount is not null

ORDER BY (unitprice ) DESC;

