UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

UFRJ

Data WareHouse - Sakila

SIAC - 2019

Alunas: Danielle de Figueiredo,
Thainá Queiroz

Professor: Geraldo Xexéo

Dezembro/2019

1. Introdução

Este relatório tem como objetivo descrever o processo de criação de um Data WareHouse a partir da base de dados Sakila. A ideia é gerar um Data WareHouse orientado ao assunto, neste caso como a base de dados trata dos dados de uma locadora, teremos que a tabela fato será relacionada ao aluguel dos filmes pelos clientes que foram atendidos por um determinado vendedor em uma determinada loja.

2. Base Sakila

A base de dados Sakila é uma base disponível para download como exemplo do MySql. Esta base se trata de uma empresa que aluga filmes, então temos que as tabelas do banco estão relacionadas em geral aos *clientes* que alugam *filmes* com um determinado *vendedor*.

A base é composta por vinte e duas tabelas que guardam os dados dos clientes, dos filmes, dos atores dos filmes, dos pagamentos, dos endereços das lojas e dos clientes e dos vendedores.

3. Criação do Data WareHouse

Nosso Data Warehouse segue a abordagem do Kimball, logo nosso DW é orientado ao assunto, no caso de uma locadora o assunto principal seria o aluguel dos filmes, então a tabela fato de nosso Data WareHouse será dado pelo evento *aluguel*.

O Data Warehouse seguirá o modelo estrela (Figura 1), então as dimensões que estão circundando nossa tabela fato serão: Filme, Loja, Cliente, Data, Funcionário e Linguagem. Então teremos que: Um cliente aluga um filme com uma determinada linguagem em uma loja com um vendedor em uma data de data de retirada, possui uma data de pagamento e uma data de devolução.

Na tabela fato além das chaves estrangeiras relacionadas a cada uma das dimensões, esperamos calcular o preço pago por cada um dos clientes em seus respectivos aluguéis e os dias que o aluguel de um filme durou. Todo trabalho de integração entre as dimensões e a tabela fato foi realizado no Pentaho Data Integration.

Figura 1 - Modelo Entidade Relacionamento DW Sakila

4. Integração de Dados

Para integração dos dados foram usados sete transformações e um fluxo de Job, em que cada transformação corresponde a uma das tabelas do *schema* do banco de dados ***dw_sakila***. Em geral, utilizamos sempre uma *task* SQL para a entrada dos dados a partir do

banco ***sakila***, e sua transformação (em alguns casos), modificamos alguns itens com *tasks* adicionais disponibilizadas pela ferramenta e inserimos os resultados na *task* Input / Update nos dados do banco de Data Warehouse.

A Figura 2 retrata a transformação usada para popular a tabela de clientes (Customer), nela é possível observar o uso da *task Concat,* que foi utilizada para juntar os campos **first_name** e **last_name**, da *task Filter,* utilizada para filtrar as linhas que não possuíssem determinados campos preenchidos que seriam necessários as junções feitas posteriormente, e da *task Merge Join*, que teve como principal utilidade a agregação de informações que estavam em tabelas auxiliares.

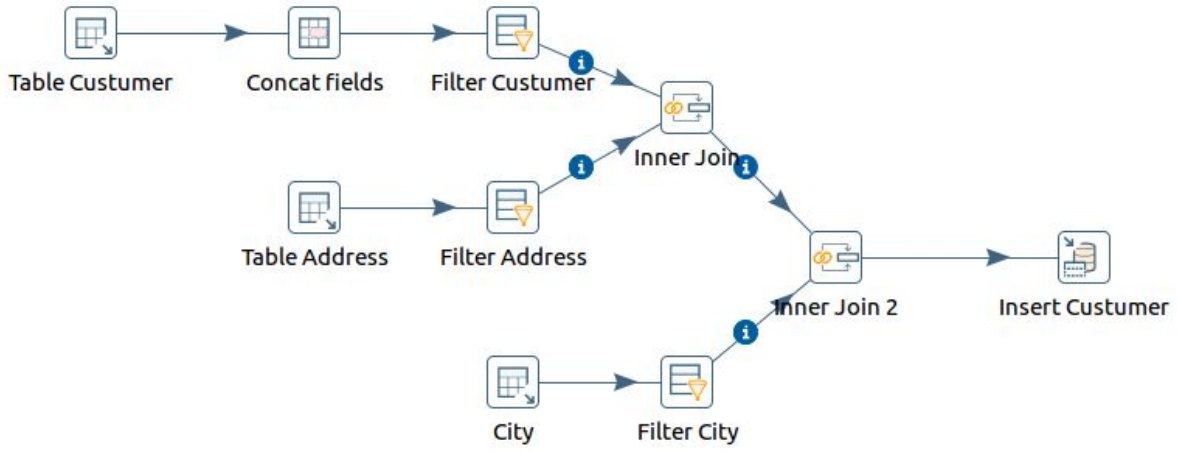

Figura 2 - Transformação da tabela Customer

A Figura 3 mostra duas *tasks* de entrada de dados que essencialmente não possuem diferenças, apenas se utilizam de consultas SQL um pouco mais complexas capazes de unir os 3 campos de datas que aparecem no modelo sakila: **rental_date**, **return_date** e **payment_date**. Além disso, nessas mesmas *tasks* são utilizadas funções SQL que retornam o dia da semana e o trimestre, ou seja, cada task retorna a data no formato 'DD/MM/YYYY', o dia da semana com a codificação 0 para segunda-feira e 6 para domingo e a coluna de final de semana. Paralelamente, utilizamos uma *task* que gera uma sequencia, ela será o elo utilizado para ligar as duas entradas de dados, uma vez que elas foram ordenadas no início. Logo após, em um fluxo paralelo com o auxílio da *task Split Field* separamos o campo da data em três novas colunas que representam dia, mês e ano. Juntamos novamente os dados e inserimos na tabela Date do modelo de Data Warehouse.

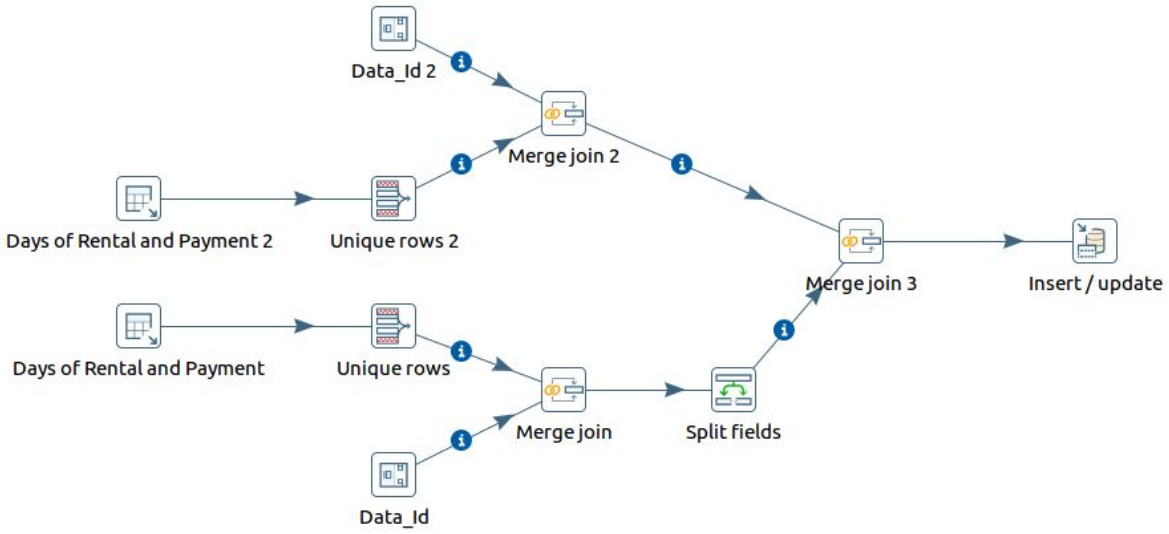

Figura 3 - Transformação da tabela Date

Na Figura 4, apresenta-se a transformação de duas tabelas em que não foi necessário enriquecer os dados com as tasks de transformação da ferramenta, porque as próprias tabelas já apresentavam os dados necessário. A *task Insert / Update* foi utilizada para correlacionar as colunas das tabelas de cada schema.

Figura 4 - Transformação das tabelas Language e Film

Para a tabela Store, como mostrado na Figura 5, foi necessário buscar campos da tabela de endereço e cidade do *schema sakila* para preencher todos os campos levantados na tabela correspondente no schema de Data Warehouse. Do mesmo modo, a tabela Store foi usada como complemento para obtenção dos campos de gerente de cada loja, como mostra a Figura 6.

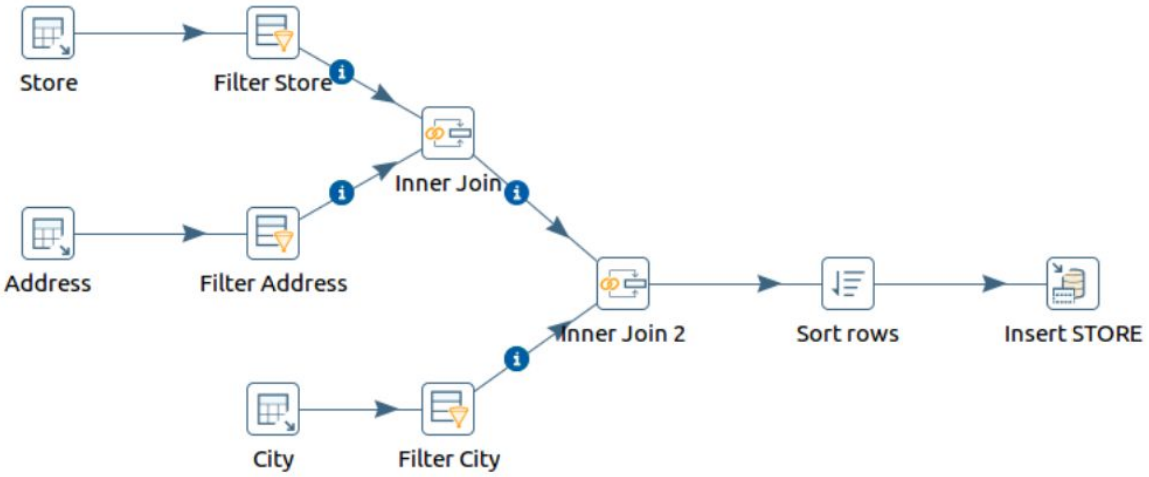

Figura 5 - Transformação da tabela Store

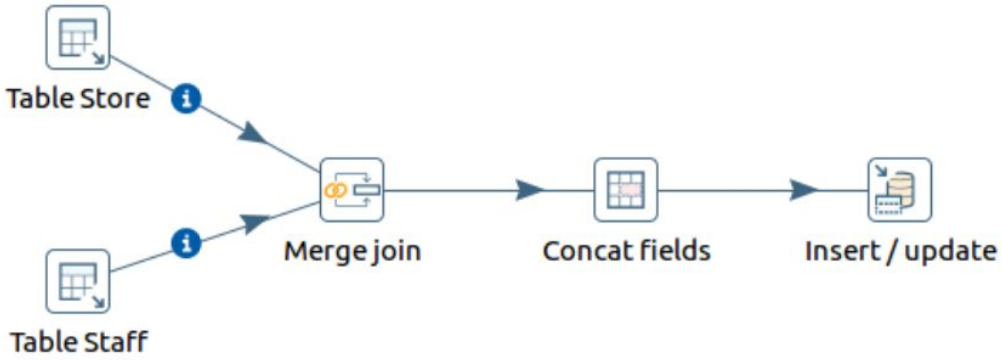

Figura 6 - Transformação da tabela Staff

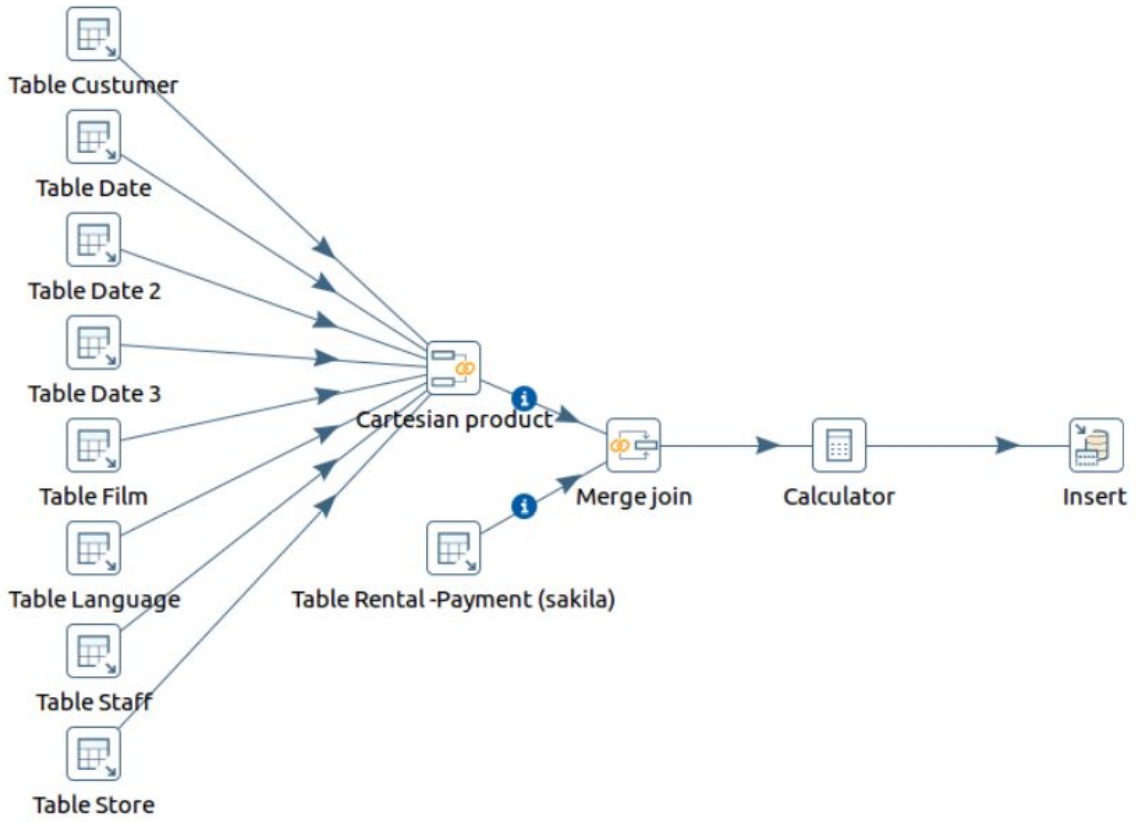

Figura 7 - Transformação da tabela Rent

Para a tabela de Aluguel (Rent) foi usado uma *task* que faz o produto cartesiano de todas as tabelas do *schema* de Data Warehouse para cobrir todas as possibilidades de junções. Deste modo, ao juntar este resultado com a tabela aluguel do *schema* sakila já enriquecida com as demais informações necessárias, pode-se calcular os campos de agregação: **rent_days** e **price**.

O Job passa, então, a ser responsável por traçar um plano sequencial para que todas as tabelas dimensão sejam preenchidas antes da tabela fato, como na Figura 8.

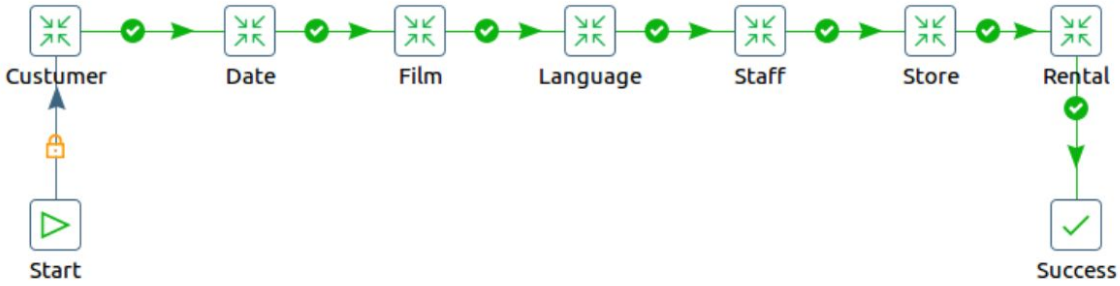

Figura 8 - Fluxo do Main Job