U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

21/05/2017

Modelação de Data Warehousing

Análise aos dados multidimensional dos serviços e praças de táxis no Porto
Tópicos Avançados em Bases de Dados – CC4007

Fábio Teixeira – 201305725
Sara Pereira – 201304112
Vanessa Silva – 201305731

Objetivos

Este trabalho prático consiste na modelação de um *data warehouse* sobre o conjunto de dados guardados nas tabelas *taxi_services* e *taxi_stands*, apresentadas nas aulas, bem como na análise destes dados multidimensionais.

Não existe uma definição rigorosa de *data warehouse*, mas pode ser considerada uma base de dados de suporte à decisão, que é antida separadamente da base operacional da organização e que suporta processamento de informação, de forma a fornecer uma plataforma sólida para análise de dados históricos, consolidados. O processo de construir e usar *data warehouses* é denominado de *data warehousing*.

O *data warehouse* criado para este trabalho não inclui qualquer informação espacial (na forma de *geometry* ou variantes), para tal foi criada uma hierarquia conceptual sobre a dimensão espacial (disponível no site do Instituto Geográfico do Exército), com dois níveis, o concelho e a freguesia.

Por último, após a criação do *data warehouse* realizámos uma análise do mesmo. Esta análise foi feita com o auxílio de consultas OLAP (*Online Analytical Processing*), e em particular consultas de histogramas, *cross-tabulation*, e a utilização do operador *CUBE BY*. OLAP refere-se à capacidade de manipular um grande volume de dados sob múltiplas perspetivas e as suas aplicações são normalmente usadas por analistas de mercado com o objetivo de suporte a decisões.

Processo

Como referido anteriormente, partimos para a realização do trabalho com o suporte de duas tabelas já existentes: *taxi_services* e *taxi_stands*. A primeira representa um conjunto de dados relativos a serviços de táxis, nomeadamente, informação sobre o tempo em que o serviço inicia, tempo em que termina, identificador do táxi e localização espacial (no tipo de dados *geometry*) do local onde inicia o serviço e outra do local onde termina; e a segunda representa a informação sobre as praças de táxis existentes, nomeadamente, o nome da praça e a sua localização espacial (no tipo de dados *geometry*).

O primeiro passo dado para a obtenção do *data warehouse* pedido foi criar um esquema Floco de Neve, com 5 novas tabelas, *dw_taxi_services*, correspondente à tabela de factos, e *dw_tempo*, *dw_taxi*, *dw_local* e *dw_stand*, correspondentes às tabelas dimensão.

Neste esquema a tabela de factos central é ligada ao conjunto das tabelas dimensão (*dw_tempo*, *dw_taxi*, *dw_local*), onde parte desta hierarquia dimensional (*dw_local*), é normalizada numa tabela dimensão mais pequena (*dw_stand*), semelhantemente a um floco de neve.

A partir da informação contida nas tabelas base, conseguimos obter novos dados, como a duração de cada serviço (*tempoTotal*, presente em *dw_taxi_services*), resultado da simples diferença entre o tempo em que o serviço termina (*final_ts*) e o tempo em que inicia (*initial_ts*), valores obtidos da tabela *taxi_services*.

Trabalhando e manipulando os dados convenientemente, preenchemos finalmente as 5 tabelas com informação que nos conseguisse não só permitir detalhar e sumarizar o vasto conteúdo de mais e melhores formas, como também satisfazer todos os pedidos e intenções do projeto, nomeadamente não conter qualquer localização espacial no tipo de dados *geometry* ou variantes. No final obtivemos um modelo com o aspeto representado na figura 1.

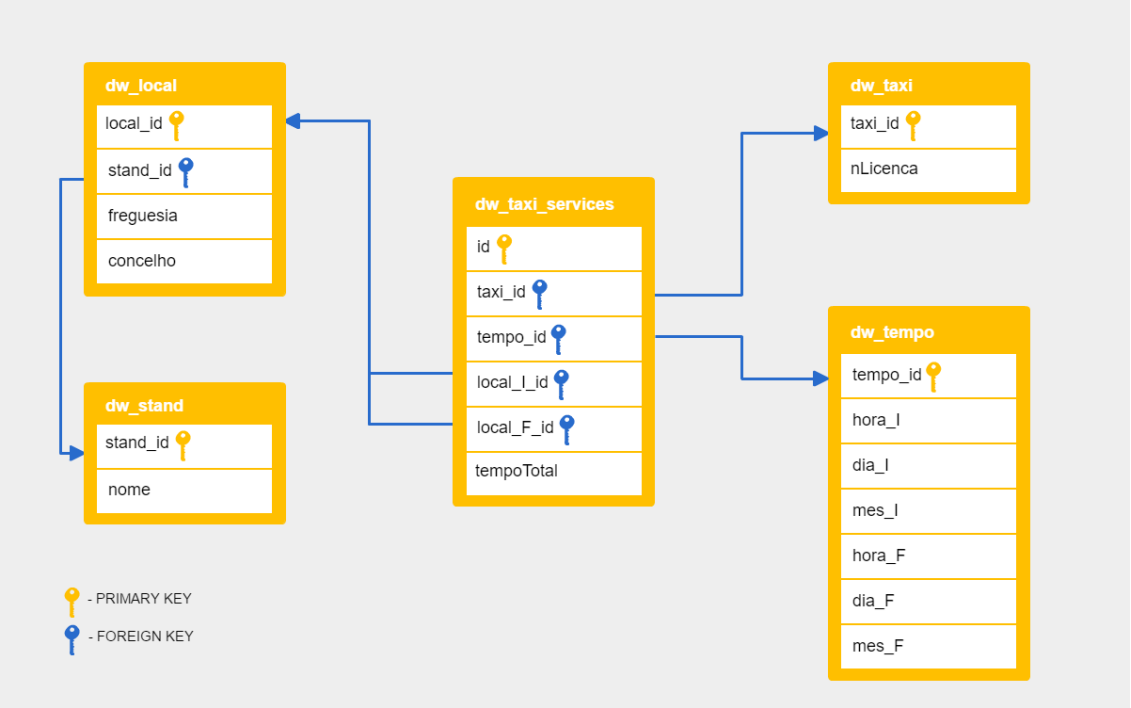

Figura 1 – Data warehouse em esquema de floco de neve.

Conteúdo

A tabela ***dw_taxi_services*** é a tabela central deste trabalho (tabela de factos), pois é nela que está reunida toda a informação proveniente das outras tabelas associadas (tabelas dimensão), por chaves externas. Esta tabela contém informações sobre todos os serviços de táxis na zona do Porto representadas em 6 colunas: *id* do tipo *int*, *taxi_id* do tipo *int*, que representa a identificação do táxi que realizou o serviço em causa, *tempo_id* do tipo *int*, que contém informação da duração do serviço, *local_I_id* do tipo *int*, identificador do ponto de partida, *local_f_id* do tipo *int*, identificador do ponto de chegada, e *tempoTotal* do tipo *text*, que indica a duração do serviço. As colunas *taxi_id*, *tempo_id*, *local_i_id*, *local_f_id* são chaves externas que ligam as tabelas dimensão.

A tabela dimensão ***dw_tempo***, como já mencionado, refere-se às informações temporais de um serviço de táxi. Esta informação é distribuída nas seguintes colunas que a compõem: *tempo_id* do tipo *int*, chave primária que faz ligação à tabela de factos, *hora_I* do tipo *text*, que representa a hora de início do serviço, *dia_I* do tipo *text*, que representa o dia em que se iniciou o serviço, e o mês presente em *mes_I* do tipo *text*, a *hora_F* do tipo *text*, *dia_F* do tipo *text* e *mes_F* do tipo *text*, que representam exatamente a mesma informação que as 3 colunas anteriores, mas referentes ao término do serviço.

dw_taxi, tabela com a informação de cada táxi, possui apenas 2 colunas: *taxi_id* do tipo *int*, chave primária que faz ligação à tabela *dw_taxi_services* e *nLicenca* do tipo *int*, que representa o número de licença de cada táxi.

A tabela ***dw_local***, para além de estar ligada à tabela de factos, é também normalizada pela tabela de dimensão mais pequena *dw_stand* através de chaves externas. Esta tabela conta com 4 tipos de informação: chave primária (identificador), que se estende à tabela de factos presente na coluna *local_id* do tipo *int*, referência à tabela *dw_stand* na coluna *stand_id* do tipo *int*, e informação sobre a freguesia e concelho desse local, que estão presentes nas colunas *freguesia* e *conselho*, ambas do tipo *text*.

Por último, a tabela ***dw_stand*** é constituída por 2 colunas e ligada por uma chave externa à tabela *dw_local*. A coluna *stand_id* do tipo *int* é a chave primária e estende-se à tabela *dw_local*, e a coluna *nome* do tipo *text* indica o nome da praça de táxis em questão.

Manipulação dos dados

Após termos definido as tabelas e respetivas dimensões, começamos por associar/preencher os valores a/em cada coluna de acordo com o que foi mencionado acima.

dw_stand:

Esta tabela foi praticamente uma exportação da tabela base *taxi_stands*. Começamos por saber quantas linhas existiam, de seguida obtivemos os nomes das praças e os seus identificadores. Após estas consultas criamos um ciclo onde adicionamos à tabela essa informação.

```
for i in range(0, nTuplos[0][0]):
     cursor.execute("insert into dw_stand (stand_id, nome)
                          values (%s, %s)", (i+1, nomes[i][0]))
```

dw_local:

A tabela *dw_local* foi uma das tabelas mais complexas para construir, uma vez que envolvia encontrar a freguesia e o conselho de cada uma das praças. O tipo de dados em que estas informações estavam, localização espacial das freguesias e concelhos em *geometry(MultiPolygon, 4326)* e localização espacial das praças em *geometry,* revelou-se um problema inicial,.

O objetivo era verificar em que freguesias as praças estavam contidas, e após algumas investidas, encontrámos a função ideal para resolver o nosso problema: **ST_Within** que retorna ***true*** se um *geometry A* (*táxi*_stands.*location*) estiver completamente dentro de uma *geometry B* (*cont_freg_v5.geom*).

Os resultados são obtidos iterativamente e inseridos após cada obtenção.

```
cursor.execute("select freguesia from cont_freg_v5
     where st_within(%s, st_astext(geom))
     and distrito like 'PORTO'", (location[i][0],)
```

dw_tempo:

Aqui apenas tivemos de consultar o *initial_ts* e o *final_ts* da tabela *taxi_services* e associar a cada tempo um identificador. Tivemos de realizar ainda conversões especiais desses valores, pois estão no formato UNIX TIME nas tabelas base. Fizemos então consultas usando *TIMESTAMP* e *TIME*, o que nos permitiu converter esses valores para o formato pretendido. A *query* seguinte representa um exemplo:

```
select TIMESTAMP 'epoch' + initial_ts * INTERVAL '1 second',
          TIMESTAMP 'epoch' + final_ts * INTERVAL '1 second'
     from taxi_services
```

O resultado obtido em determinada linha foi: "*2015-01-01 00:00:09 | 2015-01-01 00:08:01*". Usando as conversões necessárias apenas "recolhemos" o mês, dia e hora (formato *h:m:s*), referentes ao tempo de início e de fim do serviço, e associámos os valores aos respetivos campos (colunas).

dw_taxi:

Para esta tabela apenas obtemos os diferentes números de licença dos táxis e guardamos essa informação na coluna (*nLicenca*).

```
select distinct taxi_id from taxi_services
```

dw_taxi_services:

Nesta tabela a maior parte das colunas são identificadores externos, e para a preenchermos de forma concisa e correta tivemos de realizar uma *query* que referenciasse todas as tabelas de dimensão envolvidas, de forma a realizar o *matching* correto.

Esta tabela é a tabela principal que representa os dados para todos os serviços existentes na tabela *taxi_services*. Para recebermos todos os dados, precisámos de juntar as tabelas *taxi_services*, *dw_taxi* e *dw_tempo*, para obter os indentificadores correspondentes das tabelas de dimensão a cada serviço (verificando a correspondência pelo número de licença, no caso do *taxi_id*, e a correspondência pelos valores temporais - hora, dia e mês iniciais e finais - no caso do *tempo_id*), e para obter a correspondência à tabela *dw_local* (através das colunas *local_i_id* e *local_f_id*) tivemos de verificar qual a praça mais próxima do ponto de partida do serviço e a mais próxima do ponto de chegada de modo a atribuir esse identificador.

Consultas OLAP

Após a conclusão do *data warehouse*, pudemos realizar uma série de consultas para análise dos serviços de táxis no Porto. Para isso usámos as consultas OLAP aprendidas nas aulas, como histogramas, *cross-tabulation* e recorrendo ao operador de *CUBE BY*. Vejamos a seguir.

Realizámos a seguinte consulta de *cross-tabulation*, onde podemos verificar o número de praças por cada umas das freguesias e quais os seus identificadores locais (*local_id*):

```
SELECT concelho, freguesia, local_id, COUNT(stand_id) AS num_stands
FROM dw_local WHERE concelho = 'PORTO'
GROUP BY concelho, freguesia, local_id
UNION ALL
SELECT concelho, freguesia, NULL, COUNT(stand_id)
FROM dw_local WHERE concelho = 'PORTO'
GROUP BY concelho, freguesia
UNION ALL
SELECT concelho, NULL, NULL, COUNT(stand_id)
FROM dw_local WHERE concelho = 'PORTO'
GROUP BY concelho UNION ALL
SELECT NULL, NULL, NULL, COUNT(stand_id)
FROM dw_local WHERE concelho = 'PORTO'
ORDER BY concelho, freguesia, local_id;
```

E também análise do total de serviços em cada mês do ano por freguesia:

```
SELECT freguesia, COUNT(mes_i),
      SUM(case when mes_i = '1' then 1 else 0 end) Janeiro,
      SUM(case when mes_i = '2' then 1 else 0 end) Fevereiro,
      SUM(case when mes_i = '3' then 1 else 0 end) Março,
      SUM(case when mes_i = '4' then 1 else 0 end) Abril,
      SUM(case when mes_i = '5' then 1 else 0 end) Maio,
      SUM(case when mes_i = '6' then 1 else 0 end) Junho,
      SUM(case when mes_i = '7' then 1 else 0 end) Julho,
      SUM(case when mes_i = '8' then 1 else 0 end) Agosto,
      SUM(case when mes_i = '9' then 1 else 0 end) Setembro,
      SUM(case when mes_i = '10' then 1 else 0 end) Outubro,
      SUM(case when mes_i = '11' then 1 else 0 end) Novembro,
      SUM(case when mes_i = '12' then 1 else 0 end) Dezembro
FROM(SELECT t.mes_i, l.freguesia FROM dw_taxi_services AS s
      INNER JOIN dw_tempo t ON s.tempo_id = t.tempo_id
      INNER JOIN dw_local l ON l.local_id = s.local_i_id) AS cs
GROUP BY freguesia ORDER BY freguesia;
```

Também utilizámos o operador *CUBE BY*, de forma a analisar a duração total de todos os serviços feitos por cada táxi:

```
SELECT dw_taxi.nlicenca, SUM(CAST(tempototal AS interval)) AS total_time
FROM dw_taxi_services
INNER JOIN dw_taxi ON
      dw_taxi.taxi_id = dw_taxi_services.taxi_id
GROUP BY CUBE(dw_taxi.nlicenca);
```

Assim como algumas consultas de histogramas que achámos interessantes, como é o caso da soma total da duração dos serviços feitos por mês, por cada uma das freguesias:

```
SELECT mes, freguesia, SUM(CAST(tempototal AS interval))
FROM (SELECT mes_i AS mes, freguesia, tempototal
      FROM dw_taxi_services
      INNER JOIN dw_local ON
            dw_local.local_id = dw_taxi_services.local_i_id
      INNER JOIN dw_stand ON
            dw_stand.stand_id = dw_local.stand_id
      INNER JOIN dw_tempo ON
            dw_tempo.tempo_id = dw_taxi_services.tempo_id) AS  aux
GROUP BY mes, freguesia
LIMIT 10;
```

E o número de serviços feitos por mês por cada táxi:

```
SELECT mes, taxi, COUNT(*)
FROM (SELECT mes_i AS mes, nlicenca AS taxi
      FROM dw_taxi_services
      INNER JOIN dw_taxi ON
            dw_taxi_services.taxi_id = dw_taxi.taxi_id
      INNER JOIN dw_tempo ON
            dw_taxi_services.tempo_id = dw_tempo.tempo_id) AS aux
GROUP BY mes, taxi
LIMIT 10;
```

Conclusões

Durante a elaboração/criação do trabalho/*data warehouse* demos conta de vários aspetos, principalmente que o *data warehouse* poderia ser contruído de várias formas diferentes (tanto a tabela de factos como as de dimensão), dependendo do objetivo da análise sobre o mesmo. Esta diferenças residem no tipo de granularidade pretendida, no tipo de descrição em cada uma das tabelas dimensão, entre outras. Por exemplo, no nosso trabalho definimos uma granularidade fina, no sentido em que descrevemos todos os serviços a nível temporal, pois na tabela dimensão *dw_tempo* definimos tanto as características de tempo em que o serviço inicia, como as em que o serviço termina e realizámos algo semelhante para o local de início e fim do serviço, mas com a dissemelhança que estas duas informações ficaram na tabela de factos e não na *dw_local* o que implica uma dimensão adicional ao nosso *cubo*.