dw-project

1 - Operational data sources inspection and proling

To import the data I wrote the initDb.sh script.

The script streamlines the database initialization process and solves the following error I have encountered:

- the database "student" may not exist in postgres. If it does not exist, it is created.
- some CSV files contained some empty lines that gave errors when imported by Postgres. The empty lines are therefore removed
- the EOLs of the CVS files were CTRLF. This may be a problem on the linux server. I coverted the EOLS of the files to LF.
- the AdventureworksDb.sql file imports the files using the space as a separator, while the values in the CSV files are separated by Tabs. I fixed the sql script.

2 - Conceptual design

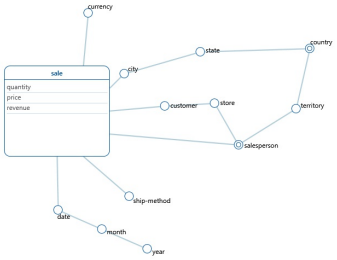
I decided to analysize one fact: Sales.

I started the process by analysing the ER schema (AdventureWorkdsDbDiagram) and then I decided dimensions ad measures for the fact.

Sale Fact

The Sale Fact most interesting measures are quantity, unit price and revenue.

The most interesting dimension root I've found are Product, Customer, City and Date, Salesperson, Currency and Ship-method.



Dimensions

Here I present an overview of the dimensions of the Sale fact:

- product
  - subcategory
    - category
- ship-method
- currency
- salesperson
  - territory
    - country
- customer
  - store
    - salesperson
      - territory
        - country
- city
  - province
    - country
- date
  - month
    - year

Customer and Salesperson share part of the dimension hierarchy. The same happens with SalesPerson, customer and city that share the country attribute.

Dynamicity

Product and its categorization may be dynamic. There may be a change in how a product is classified but this should happen quite unfrequently.

SalesPerson may move to a different territory and I expect this to be something to be careful about. I consider instead unlikely that a territory is assigned to a different country.

Customer may move to a different store quite often and the store may be assigned to a different salesperson.

City, province and country are dimensions that I expect to have only a small chanceto change.

Date is fixed.

Workload

I present here a list of logical queries:

q1

The sum of all revenues for the year 2013 for the product category "Bikes"

Sale[category = 'Bikes', year=2013].revenue

```
select round(sum(d.unitprice * (1 - d.unitpricediscount) * d.orderqty * coalesce(c.averagerate, 1)),2) as revenue
from production.product p
right join sales.salesorderdetail d on p.productid = d.productid
join sales.salesorderheader h on h.salesorderid = d.salesorderid
left join sales.currencyrate c on h.currencyrateid = c.currencyrateid
```

```
left join production.productsubcategory ps on p.productsubcategoryid = ps.productsubcategoryid
left join production.productcategory pc on ps.productcategoryid = pc.productcategoryid
where extract(year from h.orderdate) = 2013
and pc.name = 'Bikes';
```

q2

The list of the sums of all revenues for all the years divided by product category

Sale[category, year].revenue

```
select pc.name as category,
    extract(year from h.orderdate) as year,
    round(sum(d.unitprice * (1 - d.unitpricediscount) * d.orderqty * coalesce(c.averagerate, 1)),2) as revenue
from production.product p
right join sales.salesorderdetail d on p.productid = d.productid
join sales.salesorderheader h on h.salesorderid = d.salesorderid
left join sales.currencyrate c on h.currencyrateid = c.currencyrateid
left join production.productsubcategory ps on p.productsubcategoryid = ps.productsubcategoryid
left join production.productcategory pc on ps.productcategoryid = pc.productcategoryid
group by category, year
order by category, year;
```

q3

The list of all products of X category that generated a revenue greater than 200 during the week before Christmas of 2012 in the UK

Sale[date>=18/12/2012 AND date<=25/12/2012, revenue> 200, country='UK'].product

```
select distinct p.productid, p.name
from production.product p
right join sales.salesorderdetail d on p.productid = d.productid
join sales.salesorderheader h on h.salesorderid = d.salesorderid
left join sales.currencyrate c on h.currencyrateid = c.currencyrateid
join sales.salesterritory t on h.territoryid = t.territoryid
join person.countryregion country on country.countryregioncode = t.countryregioncode
where country.name = 'United Kingdom'
and h.orderdate between ('2012-12-25'::date - '1 week'::interval) and '2012-12-25'::date
group by country.countryregioncode, p.productid, p.name
having sum(d.unitprice * (1 - d.unitpricediscount) * d.orderqty * coalesce(c.averagerate, 1)) >= 1500
```

q4

The list of all cities where the product 'Mountain-200 Silver, 42' has been sold at least 5 times in the same date in 2013

Sale[product='Mountain-200 Silver, 42', date, year=2013, quantity>=5].city

```
select distinct h.orderdate, (a.city || ', ' || s.stateprovincecode) as city
from production.product p
right join sales.salesorderdetail d on p.productid = d.productid
join sales.salesorderheader h on h.salesorderid = d.salesorderid
left join sales.currencyrate c on h.currencyrateid = c.currencyrateid
join person.address a on h.shiptoaddressid = a.addressid
join person.stateprovince s on a.stateprovinceid = s.stateprovinceid
where p.name = 'Mountain-200 Silver, 42'
and extract(year from h.orderdate) = 2013
group by city, s.stateprovincecode, h.orderdate
having sum(d.orderqty) >= 5;
```

q5

The average revenue for all the currencies by year

Sale[currency, year].revenue

```
select coalesce(c.tocurrencycode, 'USD') as currency,
    round(sum(d.unitprice * (1 - d.unitpricediscount) * d.orderqty * coalesce(c.averagerate, 1)),2)
from sales.salesorderheader h
left join sales.salesorderdetail d on h.salesorderid = d.salesorderid
left join sales.currencyrate c on h.currencyrateid = c.currencyrateid
group by currency;
```

q6

Quantity of product sent by shipping method for every country

Sale[country, ship-method].quantity

```
select country.name as countryname,
    sm.name as shipmethodname,
    sum (d.orderqty) as quantity
from sales.salesorderheader h
left join sales.salesorderdetail d on h.salesorderid = d.salesorderid
join person.address a on h.shiptoaddressid = a.addressid
join person.stateprovince s on a.stateprovinceid = s.stateprovinceid
join purchasing.shipmethod sm on h.shipmethodid = sm.shipmethodid
join person.countryregion country on country.countryregioncode = s.countryregioncode
group by countryname, shipmethodname
order by countryname, shipmethodname;
```

Assumptions

- I assume that the sales.salesorderdetail.unitprice is the unit price *before* the discount (sales.salesorderdetail.unitpricediscount) application.
- When considering the measure "revenue" I always mean the actual revenue value converted in USD.
- As for the dimension "city", attribute city-name is "city-name, state-province" due to the ambiguity name we have with the city name only. I assume there can be no two cities with the same name inside the same province.

3 - Data warehouse ROLAP logical design

Data Volume

The sale fact includes 121317 events.

The following table shows the cardinality of every dimension:

Dimension name	Cardinality
product	504
date	1126
shipping-method	5
customer	19820
salesperson	7
currency	105
city	613

The maximum cardinality for the Sale event is ~ 25 * 10^16

The sparsity of the schema is 5 * 10^(-9) [(12 * 10^5) / (25 * 10^16)].

Views

4 - OLAP Queries

$1+1=2$