



JOBSHEET 3

ETL untuk Fact Table

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu membuat paket ETL untuk melakukan loading data ke transactional dan periodic snapshot fact table

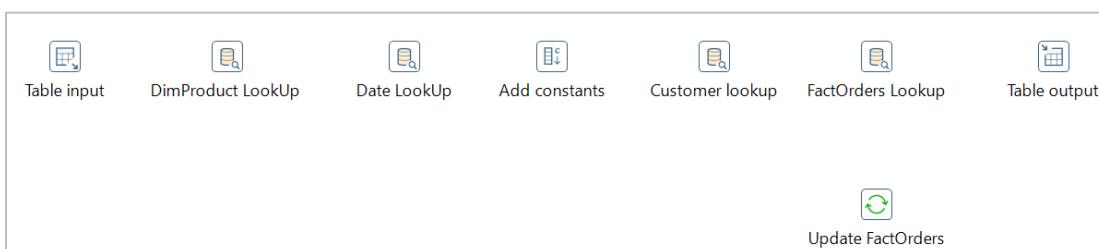
2. Praktikum

Berdasarkan granularitynya, fact table dapat dibedakan menjadi transactional, periodic snapshot, dan accumulating snapshot. Granularity adalah tingkat detail atau kehalusan data yang dicatat. Semakin tinggi granularity, semakin rinci data yang tercatat di data warehouse. Transactional fact table adalah fact table yang granularitynya atomic, artinya setiap row pada fact table berelasi dengan setiap peristiwa/transaksi. Granularity periodic snapshot adalah agregat transaksi dalam periode waktu tertentu. Jadi setiap row pada fact table menyimpan data event pada periode waktu tertentu, misalnya hari, minggu, bulan, dst. Tujuan dari periodic snapshot adalah mempercepat query dalam analisis tren/laporan berkala. Sementara itu, accumulating snapshot menyimpan status dari proses yang berlangsung dari awal hingga akhir. Berbeda dengan periodic snapshot yang menyimpan data periodic, accumulating snapshot mengikuti life cycle dari suatu proses.

Paket ETL yang akan dibuat pada praktikum ini akan menggunakan database legendvehicle sebagai data source dan dw_legendvehicle sebagai destination seperti pada praktikum sebelumnya. Anda akan mencoba merancang paket ETL untuk transactional dan periodic snapshot fact table.

2.2 Transactional Fact Table

1. Buat Transformation baru **File → New → Transformation**.
2. Drag and drop beberapa object berikut:





- **Table input:** melakukan ekstraksi data
 - **DimProduct lookup:** mengetahui nilai id_dimProduct
 - **DimDatelookup:** mengetahui nilai id_dimDate
 - **Add Constant:** menambahkan kolom baru bernilai 1 untuk filter dimCustomer
 - **DimCustomer lookup:** mengetahui nilai id_dimCustomer
 - **FactOrder lookup:** mengecek apakah data sudah pernah tersimpan sebelumnya di FactOrder
 - **Table output:** insert data baru yang belum tersedia di FactOrders
 - **Update:** update jika data produk sudah tersedia pada table FactOrders
3. Konfigurasi step **Table Input**, set connection ke oltp dan gunakan query seperti pada gambar berikut untuk **join table orders dan orderdetails**. Granularitynya adalah setiap orderdetail dari transaksi karena analisis ingin dilakukan terhadap barang yang dibeli.

Table input

Step name: Table input

Connection: conn_oltp_legendvehicle

SQL:

```
SELECT orderdetails.*, orders.customerNumber, orders.orderDate, orders.status
FROM orderdetails
INNER JOIN orders ON orderdetails.orderNumber = orders.orderNumber
```

Line 1 Column 0

Store column info in step meta

Enable lazy conversion

Replace variables in script?

Insert data from step

Execute for each row?

Limit size: 0

Help OK Preview Cancel

4. Hubungkan ouput step Table Input ke step DimProduct LookUp
5. Konfigurasi **DimProduct LookUp**, set connection, lookup schema, and lookup table. Kita akan mengecek berapa nilai id_dimproduct untuk suatu productCode. Nilai id_dimproduct nantinya akan digunakan untuk mengisi nilai foreign key pada factorders. Oleh karena itu, field yang akan dibandingkan adalah **productCode** dari table dimProduct dengan **productCode** yang berasal step sebelumnya (hasil ekstraksi dari OLTP).



Database lookup

Step name: DimProduct LookUp
 Connection: conn_dw_legendvehicle
 Lookup schema: dw_legendvehicle
 Lookup table: dimproduct
 Enable cache?
 Cache size in rows (0=cache) 0
 Load all data from table

The key(s) to look up the value(s):

#	Table field	Comparator	Field1	Field2
1	productCode	=	productCode	

Values to return from the lookup table :

#	Field	New name	Default	Type
1	productCode			None
2	id_dimProduct			None

Do not pass the row if the lookup fails
 Fail on multiple results?
 Order by

6. Hubungkan output step DimProduct LookUp ke step DimDate LookUp
7. Konfigurasi **DimDate LookUp**, set connection, lookup schema, dan lookup table. Kita akan mengecek berapa nilai id_dimdate untuk suatu orderDate. Nilai id_dimdate nantinya akan digunakan untuk mengisi nilai foreign key pada factorders. Oleh karena itu, field yang akan dibandingkan adalah **date dari table dimDate dengan orderDate yang berasal step Table Input (hasil ekstraksi dari OLTP)**.

Database lookup

Step name: DimDate LookUp
 Connection: conn_dw_legendvehicle
 Lookup schema: dw_legendvehicle
 Lookup table: dimdate
 Enable cache?
 Cache size in rows (0=cache) 0
 Load all data from table

The key(s) to look up the value(s):

#	Table field	Comparator	Field1	Field2
1	date	=	orderDate	

Values to return from the lookup table :

#	Field	New name	Default	Type
1	id_dimDate			None
2	date			None

Do not pass the row if the lookup fails
 Fail on multiple results?
 Order by



8. Hubungkan output step DimDate LookUp ke step Add Constant
9. Konfigurasi **Add Constant** untuk menambahkan konstanta baru bernilai 1 dengan nama streamTrue. Nilai ini akan digunakan untuk memfilter table dimCustomer

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value	Set empty
1	streamTrue	Integer							1	N

10. Hubungkan output step Add Constant ke step DimCustomer LookUp
11. Konfigurasi **DimCustomer LookUp**, set connection, lookup schema, and lookup table.
Kita akan mengecek berapa nilai id_dimCustomer untuk suatu customerNumber. Nilai id_dimCustomer nantinya akan digunakan untuk mengisi nilai foreign key pada factorders.
Oleh karena itu, field yang akan dibandingkan adalah **customerNumber** dari table **dimCustomer** dengan **customerNumber** yang berasal step Table Input (hasil ekstraksi dari OLTP). Lookup tambahan juga perlu dilakukan dengan memastikan nilai kolom isCurrent = streamTrue (isCurrent = 1) karena dimCustomer menggunakan SCD Type 2.

Step name: Customer lookup
Connection: conn_dw_legendvehicle
Lookup schema: dw_legendvehicle
Lookup table: dimcustomer
Enable cache?
Cache size in rows (0=cache): 0
Load all data from table

The key(s) to look up the value(s):

#	Table field	Comparator	Field1	Field2
1	customerNumber	=	customerNumber	
2	isCurrent	=	streamTrue	

Values to return from the lookup table :

#	Field	New name	Default	Type
1	id_dimCustomer			None
2	customerNumber			None

Do not pass the row if the lookup fails
Fail on multiple results?
Order by:



12. Hubungkan output step DimCustomer LookUp ke Step FactOrder Lookup
13. Konfigurasi **FactOrder LookUp**, set connection, lookup schema, and lookup table.
Langkah ini dilakukan untuk mengecek apakah orderdetails sudah pernah tersimpan pada table FactOrders.
Pengecekan akan dilakukan dengan membandingkan kombinasi **orderNumber** dan **orderLineNumber** dari table **factOrders** dengan **orderNumber** dan **orderLineNumber** yang berasal step Table Input (hasil ekstraksi dari OLTP).

The screenshot shows the 'Database lookup' configuration dialog box. It includes fields for Step name (FactOrders Lookup), Connection (conn_dw_legendvehicle), Lookup schema (dw_legendvehicle), and Lookup table (factorders). There are also options for Enable cache? and Cache size in rows (0=cache). Below these, there's a section for 'The key(s) to look up the value(s)' with a table:

#	Table field	Comparator	Field1	Field2
1	orderNumber	=	orderNumber	
2	orderLineNumber	=	orderLineNumber	

Below this, there's a section for 'Values to return from the lookup table:' with another table:

#	Field	New name	Default	Type
1	orderNumber			None
2	orderLineNumber			None

At the bottom, there are checkboxes for 'Do not pass the row if the lookup fails' and 'Fail on multiple results?', and an 'Order by' dropdown. The dialog box has standard buttons at the bottom: Help, OK, Cancel, Get Fields, and Get lookup fields.

14. Hubungkan output step FactOrder Lookup (**error**) ke step Table Output.
15. Konfigurasi **Table Output**, set connection, lookup schema, and lookup table. Untuk setiap baris data dengan kombinasi orderNumber + orderLineNumber yang tidak berhasil di-lookup, maka akan ditambahkan baris baru pada tabel factOrders.



Table output

Step name: Table output
 Connection: conn_dw_legendvehicle
 Target schema: dw_legendvehicle
 Target table: factorders
 Commit size: 1000
 Truncate table:
 Ignore insert errors:
 Specify database fields:

Main options Database fields

Fields to insert:

#	Table field	Stream field
1	orderNumber	orderNumber
2	quantityOrdered	quantityOrdered
3	priceEach	priceEach
4	orderLineNumber	orderLineNumber
5	status	status
6	id_dimProduct	id_dimProduct
7	id_dimDate	id_dimDate
8	id_dimCustomer	id_dimCustomer

Get fields Enter field mapping

OK Cancel SQL

16. Hubungkan output step FactOrder Lookup (**main**) ke step Update.
17. Konfigurasi step **Update**. Untuk setiap baris data dengan kombinasi orderNumber + orderLineNumber yang berhasil di-lookup, maka akan dilakukan update pada tabel factOrders. Pastikan update dilakukan terhadap kombinasi orderNumber + orderLineNumber yang sesuai.

Update

Step name: Update FactOrders
 Connection: conn_dw_legendvehicle
 Target schema: dw_legendvehicle
 Target table: factorders
 Commit size: 100
 Use batch updates?
 Skip lookup
 Ignore lookup failure? Flag field (key found)

The key(s) to look up the value(s):

#	Table field	Comparator	Stream field1	Stream
1	orderNumber	=	orderNumber	Get fields
2	orderLineNumber	=	orderLineNumber	

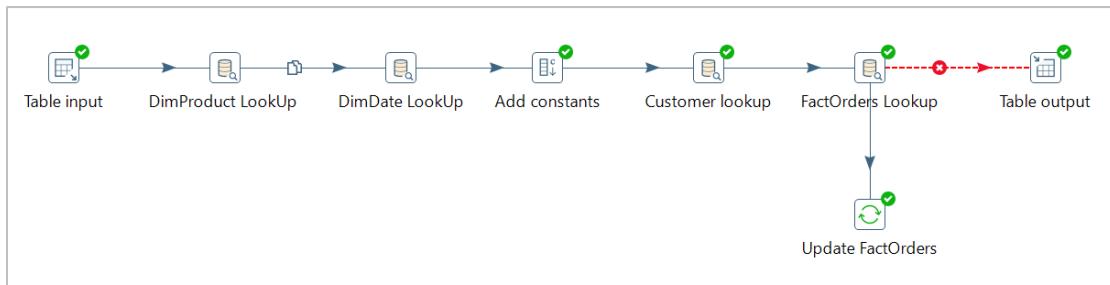
Update fields:

#	Table field	Stream field
1	quantityOrdered	quantityOrdered
2	priceEach	priceEach
3	status	status
4	id_dimProduct	id_dimProduct
5	id_dimDate	id_dimDate
6	id_dimCustomer	id_dimCustomer

Get update fields

OK Cancel SQL

18. Konfigurasi akhir transformation terlihat pada gambar berikut



Tugas 1

1. Save dan run transformation. Screenshot Step Metrics pada Execution Results.
2. Perhatikan tab preview data (tidak perlu discreenshot) dari setiap langkah kemudian beri penjelasan apa yang dilakukan oleh setiap langkah.
3. Periksa apakah data order sudah tersimpan di factOrders
4. Run transformation kembali dan screenshot ulang Step Metrics pada Execution Results.
5. Apakah ada duplikasi data orders? Mengapa?

Tugas 2

1. Buat sebuah periodic snapshot fact table. Misalnya data transaksi customer per hari. Di database dw_legendvehicle, tambahkan table baru **factCustomerDailyOrders** dengan kolom id_factCustDailyOrder, id_dimCustomer (FK), id_dimDate (FK), dan total.
2. Buat transformation untuk mengisi periodic snapshot tersebut.

Hint:

Untuk proses ekstraksi dari OLTP, gunakan aggregate function untuk menghitung total transaksi setiap customer per hari.

```

SELECT orders.customerNumber, orders.orderDate,
       SUM(priceEach * quantityOrdered) as total
  FROM orderdetails
 INNER JOIN orders ON orderdetails.orderNumber = orders.orderNumber
 GROUP BY orders.customerNumber, orders.orderDate
  
```

Selanjutnya gunakan langkah serupa dengan praktikum di atas untuk menemukan id_dimCustomer dan id_dimDate.

3. Screenshot konfigurasi untuk setiap step.
4. Save dan run transformation. Screenshot Step Metrics pada Execution Results.