DATA WAREHOUSING END SEM TASK

GOM4DW to Snowflake schema

Group 16:

Rahul Maheshwari (MT19027)

P. Akshay Kumar (MT19094)

Nikunj Agarwal (MT19093)

Objective

To add functionality to the existing tool of converting a GOM4DW schema to Snowflake schema and run it on Vyapari Dataset. This tool will have a button 'Convert to ROLAP' that will perform this activity.

To run this tool, please execute **WelcomeScreen.java** if you want to work on a new project. If you want to launch an already existing project then execute **LoginScreen.java** and provide project name as **P1.** The java files are located in **DW_endSemProj\src\com\FinalInfo**

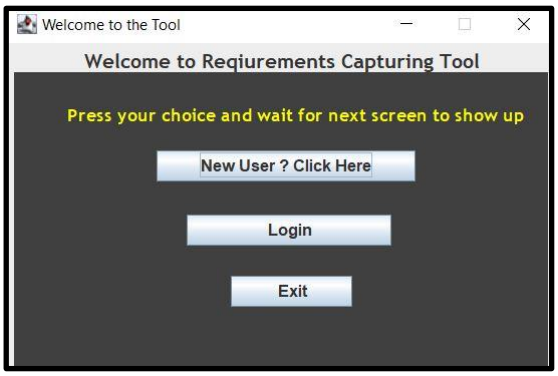
The database P1 is exported to an excel file named **P1.xlsx** and this is included in the project folder.

To import this, excel file as database, please refer to the following link.

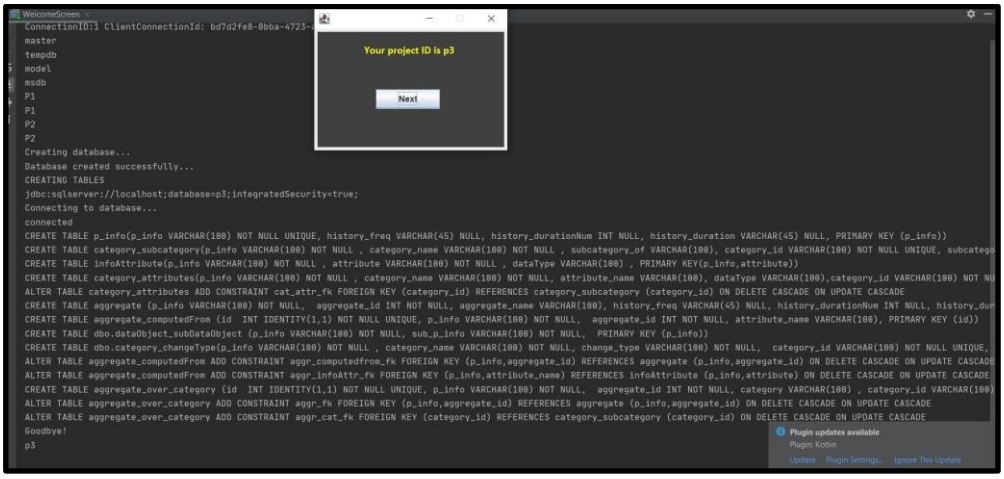
https://www.sqlserverlogexplorer.com/import-and-export-database/

How was the tool used to capture the GOM4DW data and category objects?

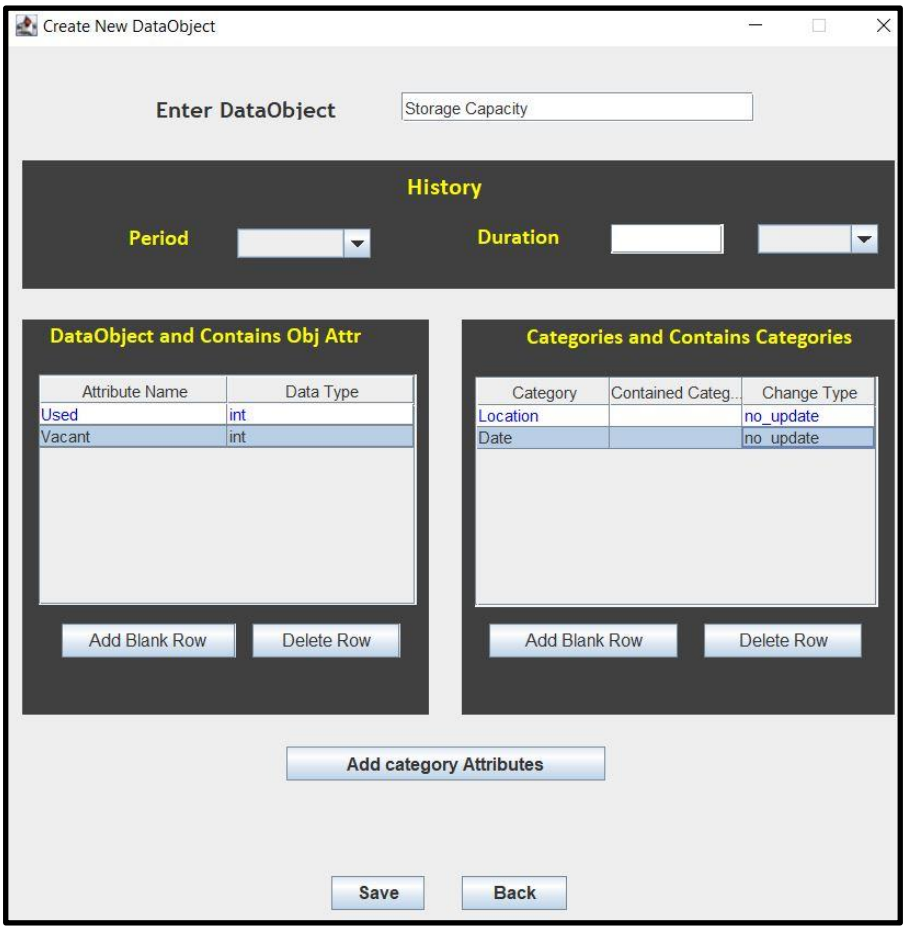
Step 1: When the welcome screen was launched, the following GUI showed up.



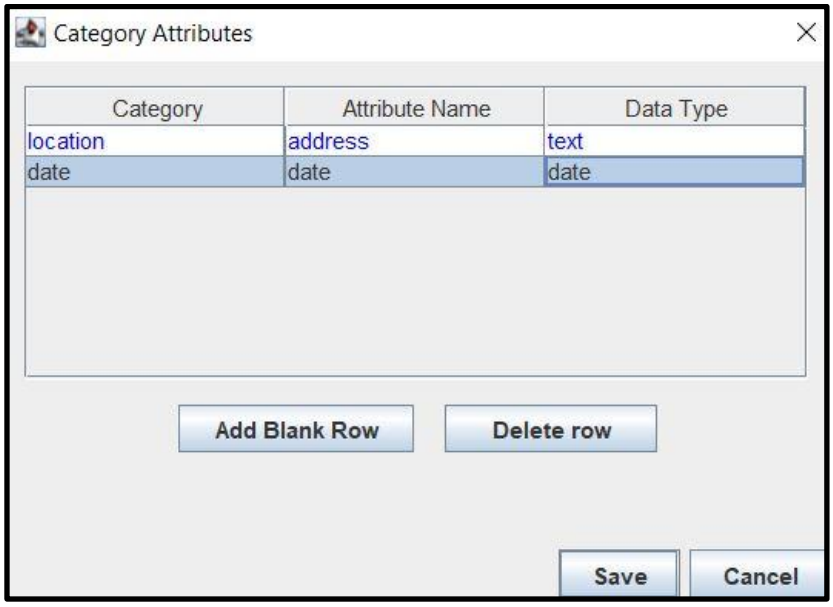
Step 2: Once we click on “New User? Click Here”, the tool generates all the tables required to capture the data and category objects.



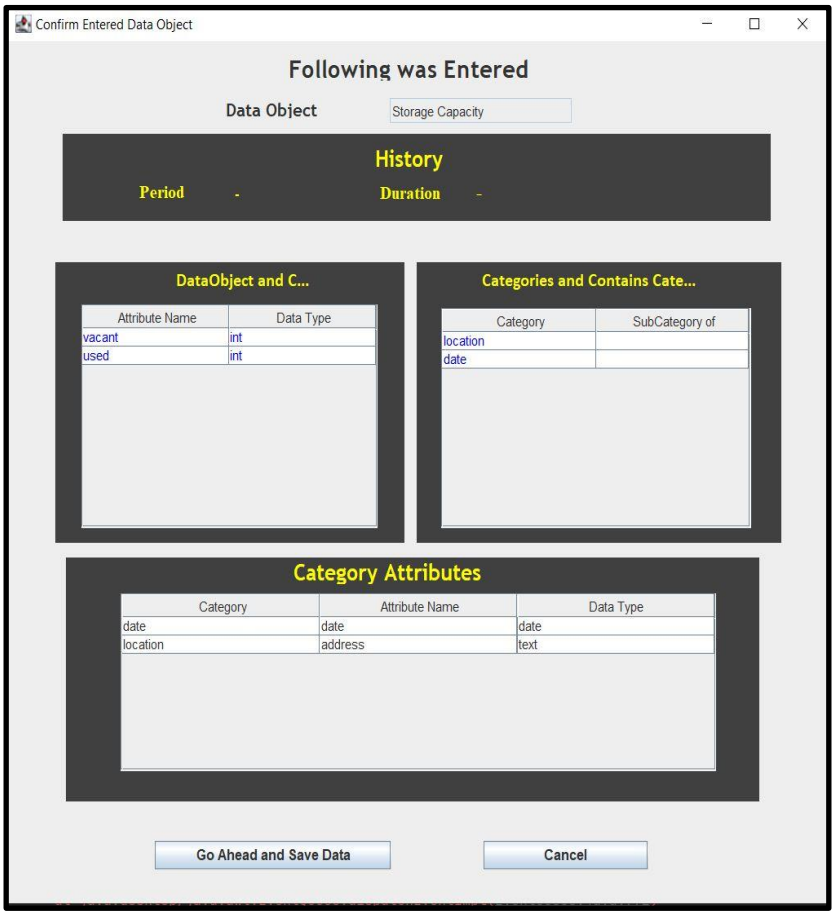
Step 3: Next. click on “Create new object” and make entries for all data objects, their attributes, categories, sub-categories (if any) and their change types.



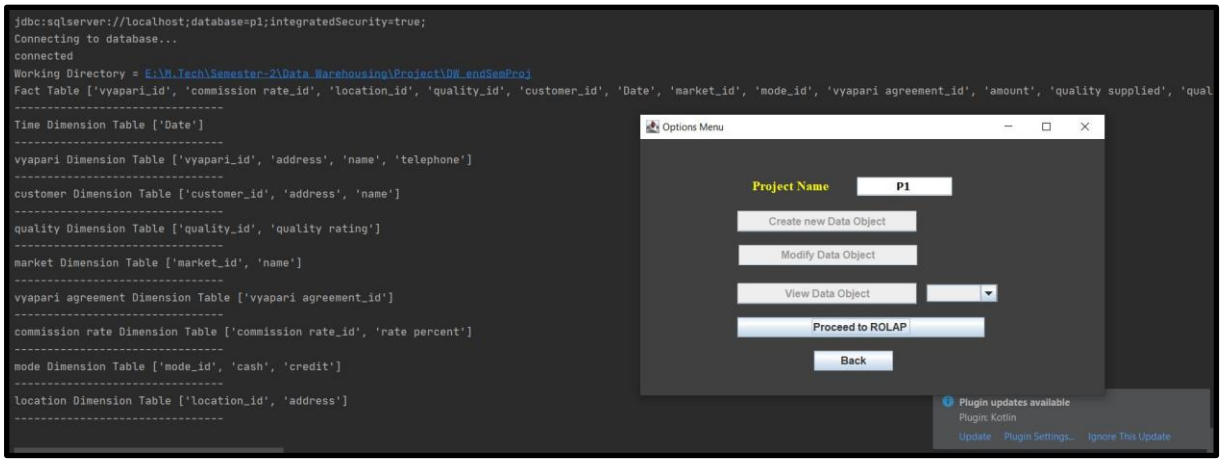
Step 4: We then click on add category attributes to add the attributes corresponding to each category.



Step 5: Once we click on "Save" button, the GUI prompts for confirmation. On successful insertion, the GUI show success message "Saved successfully".



Step 6: Proceed to ROLAP triggers button on the project home screen triggers the python script and ROLAP schema is generated as output.



How Data object within a data object was handled?

A datatype “Data Object” was added in the list of datatypes associated with attributes of a data object. This differentiates a data object from other attributes and this condition was used to check if another data object was contained in a data object.

How change type was handled?

Change type was added as a field in “Create User Info” screen and its corresponding table was created with the schemas as follows.

dbo.category_changeType(p_info varchar, category_name varchar, change_type varchar, category_id varchar, time_stamp datetime, primary_key(category_id));

“change_type” was a combo list with values “no_change”, “type-1”, “type-2”. The default value of “time_stamp” was set to be CURRENT_TIMESTAMP and that of “change_type” was “no_change”

An entry in the change_type table is shown below.

	p_info	category_name	change_type	category_id	time_stamp
1	Agreed Commission	day	no_change	Agreed Commission_day	2020-05-18 21:41:08.557

Implementation of Conversion algorithm.

Database connection

Conversion algorithm was implemented in Python and the script gets triggered onclick of “Proceed to OLAP” button on the “createNewInfo” screen on the GUI. Pyodbc object was used to connect to the MSSQL database.

Handling Data Objects

Each data object was stored in a ‘fact’ list and its associated attributes were stored in “facts_attributes_dict” with key as the ‘data object’ (fact). If the data object had other data objects as attributes, then the data object was added as a dimension along with its attributes to “dimensions_attributes_dict”. For every dimension, an attribute with name “dimension_id” was added to the dimension attributes list which will acts as the primary key for dimension table when the ROLAP schema is built. This is also added to “foreign_key_list” which will be later added to the fact table since this will act as the foreign key constraint link between the fact table and the dimension table.

Handling category objects

Each category associated with the data object that is not present in the list of dimensions, is added to the dimensions list and its associated attributes are linked with the dimension. Also, if the change type is “no_change” and timestamp is NULL, then the timestamp is added as an attribute to the dimension dict. Each dimension is then linked to the data object which is initially fetched from database. If there are subcategories then a subcategory dictionary is created with subcategory as key and its attributes as values. This is then later linked with the associated dimension.

Handling the time attributes

All the date/day attributes associated with the “Potato Adhati vyapari” usecase are replaced with a single “Time Dimension” with attribute as (date,day,week,month,quarter,year) since most of the ROLAP operations that occurs with respect to this use case is operated on daily basis to retrieve useful information and maintaining history information in the system.

Output:

Facts:

['market_id', 'mode_id', 'vyapari_agreement_id', 'vyapari_id', 'quality_id', 'time_id', 'commission_rate_id', 'location_id', 'customer_id', 'quality_supplied', 'sgst', 'commission_amount', 'price', 'total_quantity_in_hand', 'quality', 'quantity', 'commission_rate_agreed', 'easy_quality_change', 'amount', 'easy_new_price', 'discount', 'quantity_supplied', 'punctual_supply', 'selling_rate', 'delivered_quality', 'commission_rate', 'price_agreed', 'original_price_agreed', 'quality_agreed', 'used', 'vacant', 'price_in_market', 'delivery_time_price_agreed', 'delivery_lead_time', 'quantity_agreed', 'quantity_sold_in_market', 'quality_required', 'transport_cost', 'quantity_thrown', 'agreed_commission_rate', 'easy_new_quantity', 'storage_cost', 'cgst', 'quantity_in_market', 'quantity_required']

Dimensions:

Time: ['time_id','date', 'day', 'week', 'month', 'quarter', 'year']

vyapari: ['vyapari_id', 'name', 'telephone', 'address']

customer: ['customer_id', 'name', 'address']

quality: ['quality_id', 'quality_rating']

market: ['market_id', 'name']

vyapari agreement: ['vyapari_agreement_id']

commission rate: ['commission_rate_id', 'rate_percent']

mode: ['mode_id', 'cash', 'credit']

location: ['location_id', 'address']

Create SQLs for FACT and DIMENSION tables:

CREATE TABLE DBO.vyapari_TABLE (vyapari_id int PRIMARY KEY,telephone int,name text,address text);

CREATE TABLE DBO.customer_TABLE (customer_id int PRIMARY KEY,name text,address text);

CREATE TABLE DBO.quality_TABLE (quality_id int PRIMARY KEY,quality_rating int);

CREATE TABLE DBO.market_TABLE (market_id int PRIMARY KEY,name text);

CREATE TABLE DBO.vyapari_agreement_TABLE (vyapari_agreement_id int PRIMARY KEY);

CREATE TABLE DBO.commission_rate_TABLE (commission_rate_id int PRIMARY KEY,rate_percent float(4));

CREATE TABLE DBO.mode_TABLE (mode_id int PRIMARY KEY,cash bit,credit bit);

CREATE TABLE DBO.location_TABLE (location_id int PRIMARY KEY,address text);

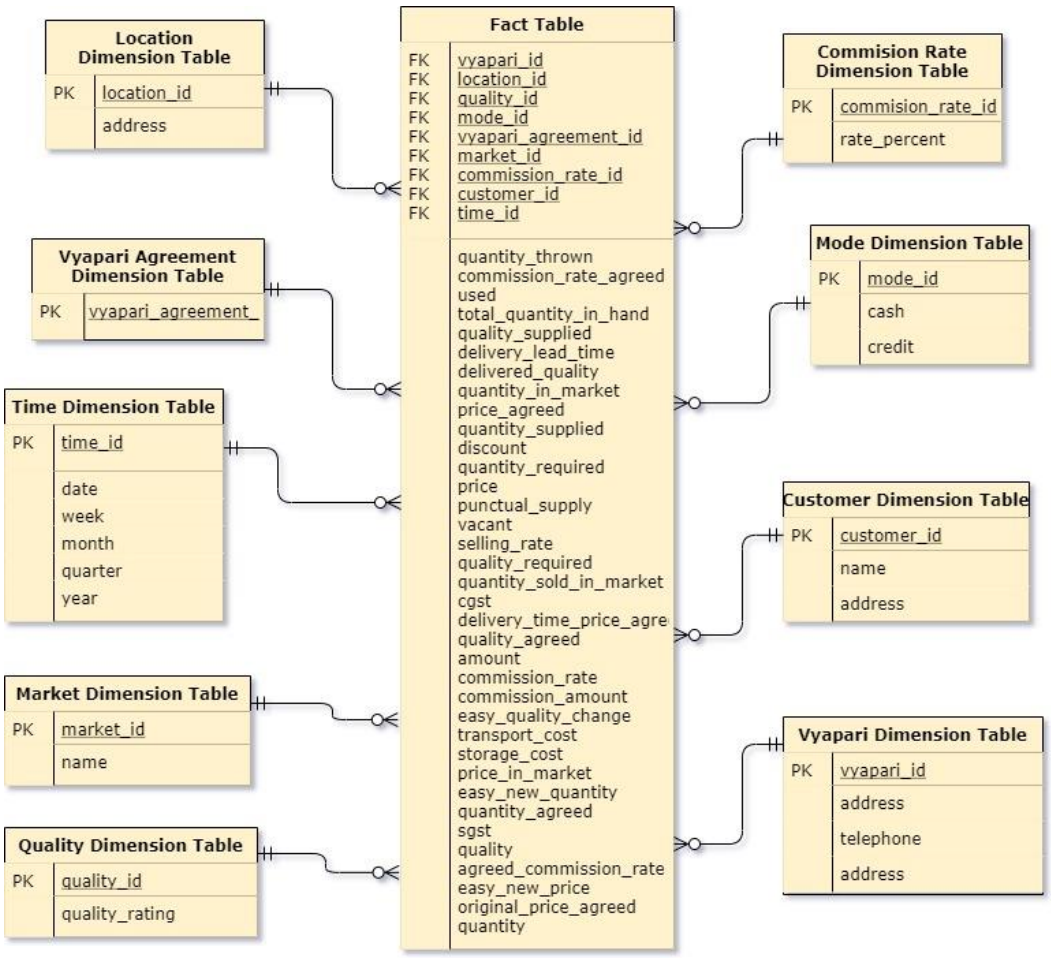
CREATE TABLE DBO.TIME_TABLE (time_id int PRIMARY KEY,date date,day int,week int,month int,quarter int,year int);

CREATE TABLE DBO.FACT_TABLE (market_id int REFERENCES market_TABLE(market_id),mode_id int REFERENCES mode_TABLE(mode_id),vyapari_agreement_id int REFERENCES vyapari_agreement_TABLE(vyapari_agreement_id),vyapari_id int REFERENCES vyapari_TABLE(vyapari_id),quality_id int REFERENCES quality_TABLE(quality_id),time_id int REFERENCES TIME_TABLE(time_id),commission_rate_id int REFERENCES commission_rate_TABLE(commission_rate_id),location_id int REFERENCES location_TABLE(location_id),customer_id int REFERENCES customer_TABLE(customer_id),quality_supplied text,sgst float(4),commission_amount float(4),price float(4),total_quantity_in_hand int,quality text,quantity int,commission_rate_agreed float(4),easy_quality_change int,amount float(4),easy_new_price int,discount float(4),quantity_supplied int,punctual_supply bit,selling_rate float(4),delivered_quality text,commission_rate float(4),price_agreed float(4),original_price_agreed float(4),quality_agreed text,used float(4),vacant float(4),price_in_market float(4),delivery_time_price_agreed float(4),delivery_lead_time float(4),quantity_agreed int,quantity_sold_in_market int,quality_required text,transport_cost float(4),quantity_thrown int,agreed_commission_rate float(4),easy_new_quantity int,storage_cost float(4),cgst float(4),quantity_in_market int,quantity_required int);

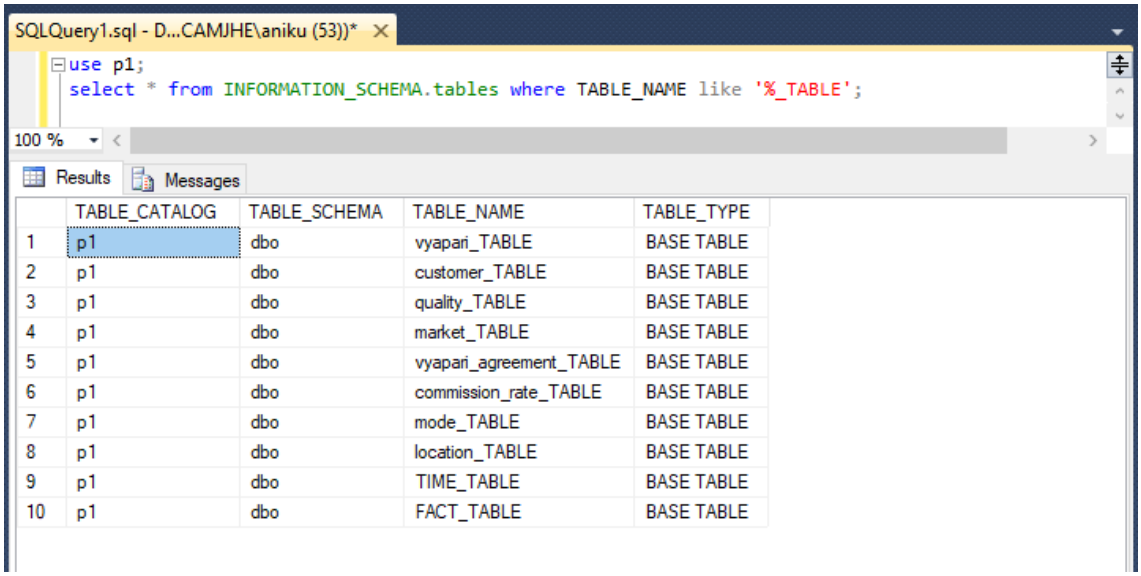
output.txt

* Refer to the output.txt file for detailed output

Following is the snowflake schema developed by code:



Screenshot for tables created in SQL Server:



	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
1	p1	dbo	vyapari_TABLE	BASE TABLE
2	p1	dbo	customer_TABLE	BASE TABLE
3	p1	dbo	quality_TABLE	BASE TABLE
4	p1	dbo	market_TABLE	BASE TABLE
5	p1	dbo	vyapari_agreement_TABLE	BASE TABLE
6	p1	dbo	commission_rate_TABLE	BASE TABLE
7	p1	dbo	mode_TABLE	BASE TABLE
8	p1	dbo	location_TABLE	BASE TABLE
9	p1	dbo	TIME_TABLE	BASE TABLE
10	p1	dbo	FACT_TABLE	BASE TABLE

ASSUMPTIONS:

- “Change_type” can be set only at the time of creation of category in GUI and its addition to the database.
- “Data Object” is used as a datatype to identify data objects within a data object.
- If history is to be maintained for any data object, then the attributes **'day', 'week', 'month', 'quarter', 'year'** automatically get added to time dimension in order to support ROLAP operations.
- All date and day related attributes are assumed as date and added to the time dimension.