Requirements for the Star Schema

Part A of Take Home MSIS 543 Star Schema Exercise

Design a dimensional model.

There will be only one line in each invoice for each product. That is, the same product will not show up in multiple lines in the same invoice.

For the fact table, show all foreign keys and the fact measures.

For the dimension tables, you can simplify the list of attributes for each table. You don't need to provide a detailed list of attributes.

For example: a table can be listed as its key, and just the words: Other Attributes.

Date, other attributes: for Date table

When information is given about other attributes, include them in your list: for example, Product dimension.

When information is not given about any attributes at all, and you conclude we need a dimension for this, you can make suitable assumptions. A similar example would be: you know Policy is a dimension in a model, but don't know attributes, and you write: Policy Key, Policy Terms.

You can either draw up the star schema diagram as connected tables, or simply provide a list: fact table, attributes, dimension tables, attributes.

Some of the comments above are designed in anticipation of specific questions you may have; that also implies that certain sentences may initially appear odd to you - 'should I do something about this? If so, what?' - and when they do, move on. Sketch out a model using your basic understanding and then revisit the description.

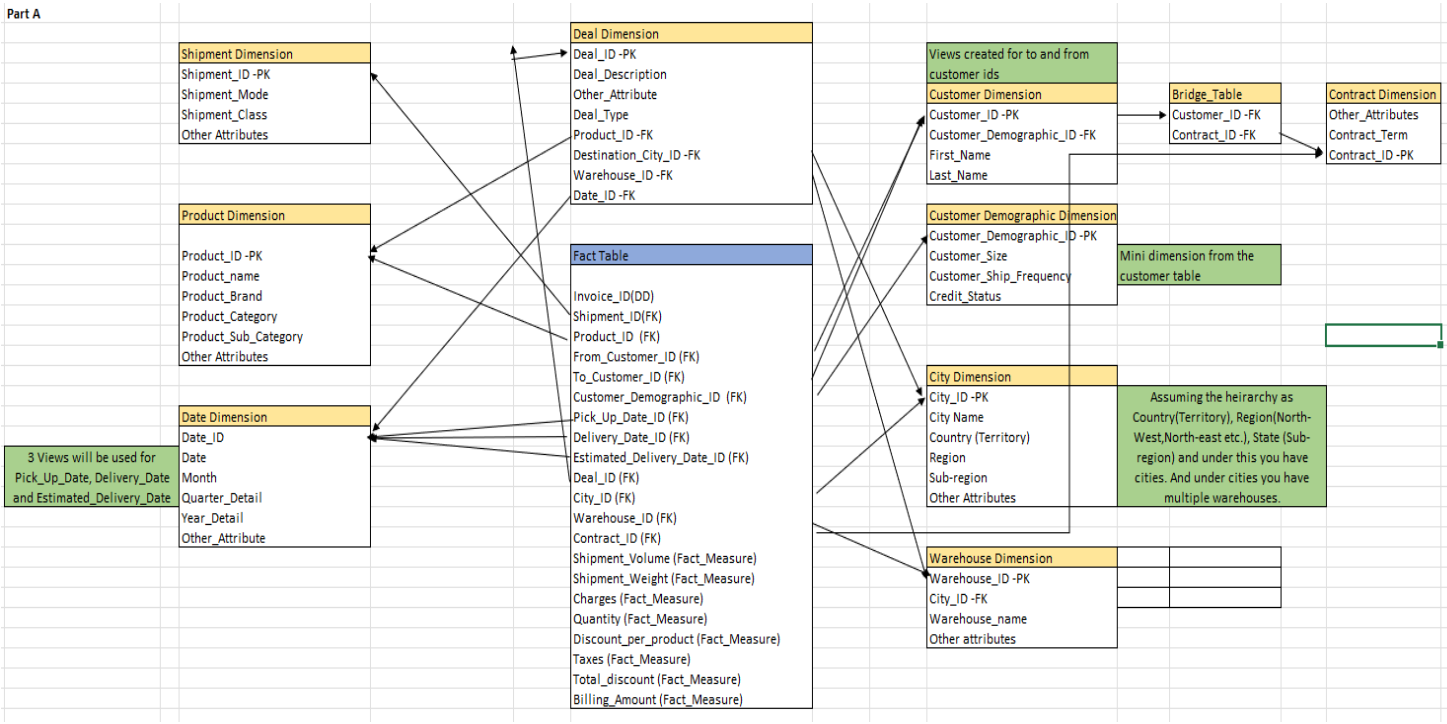
Also Answer the following question:

If we want to track information about deals in existence on any given date, will the fact table of Invoice line items give us that info? If not, what do we need to add to the model?

Name: Raghavendra Ghosh
Topic: Take home assignment

Part A

Grain would be the line-item invoice and each line item have a unique product within invoice associated with quantity attribute. The **primary key** for the fact table would be combination of **Invoice_ID** and the **Product_ID** and become a unique combination.



Schema Description

Foreign Keys in the fact table are Product_ID, From_Customer_ID, To_Customer_ID, Customer_Demographic_ID, Pick_Up_Date_ID, Delivery_Date_ID, Estimated_Delivery_Date_ID, Contract_ID, Deal_ID, City_ID, Warehouse_ID and the measurable fact items Shipment_Volume, Shipment_Weight, Charges, Quantity, Discount_per_product, Taxes, Total_discount and Billing_Amount. Dimensions in the above star schema are Shipment, Product, Date, Contract, City, Warehouse, Customer, Customer_Demographic and Deal Dimension.

Some dimensions would have different views like date dimensions would views for Pick_Up_Date, Delivery_Date and Estimated_Delivery_Date. Apart from this customer dimension would have a mini dimension called customer demographics which gives details of customer size, customer ship frequency and credit status and also the from_customer_ID and to_customer_ID will come from the customer dimension. This will be in the form of views of the customer table. Also the customer table would be connected with the contract table using the bridge dimension as it will give us information about the customers coming under each contract.

The deal dimension will be in the form of factless dimension table since it needs to be queried for product_id, start_date and end_date which will be obtained from the date dimension, destination _city which will come from the city dimension and warehouse which will come from the warehouse dimension. This would also allow us to query the deal dimension and obtain the existence of the deals on given date.

Assumption:
We are making the assumption that each shipment comes under one contract. One deal for unique combination of product, destination warehouse and city. Assuming the heirarchy as Country(Territory), Region(North-West,North-east etc.), State (Sub-region) and under this you have cities. And under cities you have multiple warehouses.

Part B

1.a. **Mini Dimension**: When we want to analyse frequently changing attributes or a small set of attributes which are already present in current dimension then it becomes difficult to query the current dimension since its huge. Also, in case of changing attributes it would be difficult to apply SCD2 and increase the rows of already huge dimension. Mini dimension has less rows than the original dimension table and these attributes are correlated to existing dimension and belong to a range. Eg. If we want to frequently analyse age, gender and address from a huge customer dimension.

b. **Junk Dimension**: Many businesses process have a large number of miscellaneous, low cardinality and unrelated indicators. Instead of adding them in the original dimensions and expanding the table, we can create junk dimensions and add these attributes there. These attributes will come from the source attribute.
Eg. In case we want to store information of the clerk or the cashier involved then we would need to keep this information in a separate junk dimension.

c. **Degenerate Dimension**: These are attributes present in the fact table which do not have a specific dimension table associated with it. Eg. In the dimension table in part A the grain is the product with the invoice attribute being mentioned in the fact table itself. There is no invoice dimension that is attached to the fact table for providing the invoice transaction details. It is only in the fact table that you find the details of the invoice. So, invoice attribute is degenerate dimension.

d. **Multivalued Dimension**: In many dimensions the attributes would have just one value but in some cases these attributes can have multiple values like in the case of phone numbers and emails. These multiple rows in the dimension can be associated with attributes in fact table using a bridge table where related rows can be grouped together and this grouped key serves as foreign key.

2.a. The mini dimension in the figure 16.2 will come from the policyholder dimension. The policyholder dimension would have attributes like age of the insurance policy status which keep changing very frequently and also the number of family members that come under the policy. These two attributes keep changing rapidly and can be kept as a mini dimension which people can query frequently and easily without having to query the big policy holder dimension. Also, the policyholder dimension almost has 1 million rows so it would be better to place these changing attributes in a mini dimension. Another dimension where we can use mini dimension is Cover Dimension, since the coverage members keep changing.

b. Employees have some unique attributes but at the same time they have common attributes and representing them in just one dimension table would result in many null values in the employee dimension. In order to prevent this scenario, we can use **supertype** dimension for the main employee table and then create **subtype** dimension for all the unique attributes. This makes it easier to analyse the customer employee data.