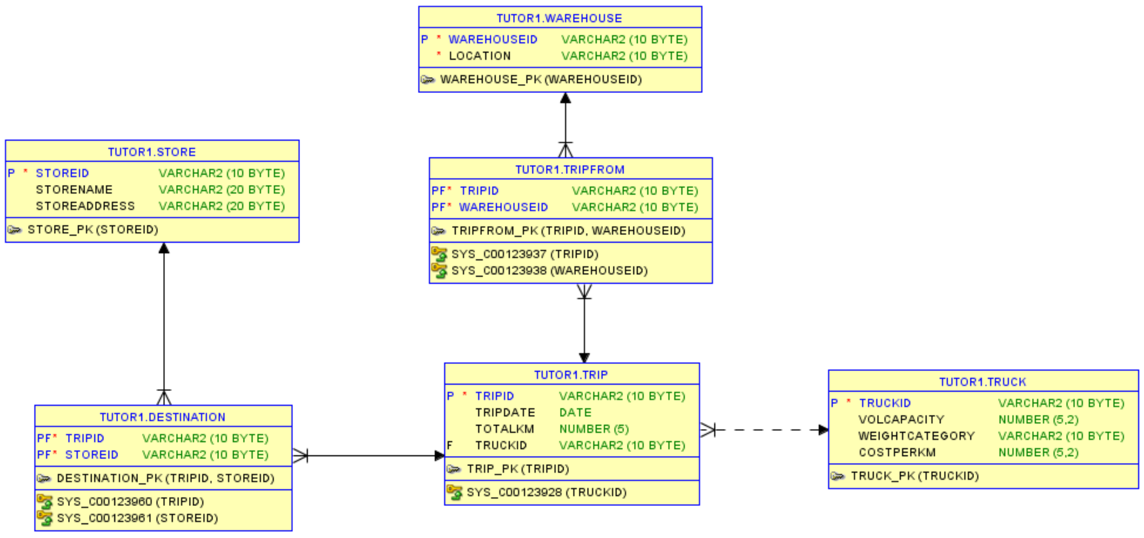
Advanced Bridge Tables
Case Study #5 (TRUCK DELIVERY)

1. A Truck Delivery Case Study – Description

A trucking company is responsible for picking up shipments from warehouses of a retail chain called MYER, and delivering the shipments to individual retail store of MYER. A truck may carry several shipments during a single trip, which is identified by TripID, and delivers those shipments to multiple stores. Trucks have different capacities for both the volumes they can hold and the weights they can carry. At the moment, a truck makes several trips each week. An operational database is being used to keep track the deliveries, including the scheduling of trucks, which provide timely deliveries to stores. The following is an E/R diagram of the truck delivery system.



TUTOR1.WAREHOUSE
P * WAREHOUSEID VARCHAR2 (10 BYTE)
* LOCATION VARCHAR2 (10 BYTE)
WAREHOUSE_PK (WAREHOUSEID)
TUTOR1.STORE
P * STOREID VARCHAR2 (10 BYTE)
STORENAME VARCHAR2 (20 BYTE)
STOREADDRESS VARCHAR2 (20 BYTE)
STORE_PK (STOREID)
TUTOR1.TRIPFROM
PF* TRIPID VARCHAR2 (10 BYTE)
PF* WAREHOUSEID VARCHAR2 (10 BYTE)
TRIPFROM_PK (TRIPID, WAREHOUSEID)
SYS_C00123937 (TRIPID)
SYS_C00123938 (WAREHOUSEID)
TUTOR1.TRIP
P * TRIPID VARCHAR2 (10 BYTE)
TRIPDATE DATE
TOTALKM NUMBER (5)
F TRUCKID VARCHAR2 (10 BYTE)
TRIP_PK (TRIPID)
SYS_C00123928 (TRUCKID)
TUTOR1.TRUCK
P * TRUCKID VARCHAR2 (10 BYTE)
VOLCAPACITY NUMBER (5,2)
WEIGHTCATEGORY VARCHAR2 (10 BYTE)
COSTPERKM NUMBER (5,2)
TRUCK_PK (TRUCKID)
TUTOR1.DESTINATION
PF* TRIPID VARCHAR2 (10 BYTE)
PF* STOREID VARCHAR2 (10 BYTE)
DESTINATION_PK (TRIPID, STOREID)
SYS_C00123960 (TRIPID)
SYS_C00123961 (STOREID)

Based on the above E/R diagram:

- A trip may come from many warehouse (many-to-many relationship between Warehouse and Trip)
- A trip uses one truck only, and obviously a truck may have many trips in the history (many-to-1 relationship between Trip and Truck)
- A trip deliver goods (e.g. TVs, fridges, etc) may deliver to several stores (many-to-many relationship between Trip and Store, which is represented by the Destination table in between Trip and Store).

The tables from the operational database are as follows:

Warehouse (**WarehouseID**, Location)
Trip (**TripID**, Date, TotalKm, *TruckID*)
TripFrom (**TripID, WarehouseID**)
Truck (**TruckID**, VolCapacity, WeightCategory, CostPerKm)
Store (**StoreID**, StoreName, Address)
Destination (**TripID, StoreID**)

Some sample data in the operational database are as follows:

Warehouse Table

WarehouseID	**Location**
W1	Warehouse1
W2	Warehouse2
W3	Warehouse3
W4	Warehouse4
W5	Warehouse5
...	...

Trip Table

TripID	Date	TotalKm	TruckID
Trip1	14-Apr-2013	370	Truck1
Trip2	14-Apr-2013	570	Truck2
Trip3	14-Apr-2013	250	Truck3
Trip4	15-Jul-2013	450	Truck1
Trip5	15-Jul-2013	175	Truck2
...	...	...	...

TripFrom Table

TripID	**WarehouseID**
Trip1	W1
Trip1	W4
Trip1	W5
Trip2	W1
Trip2	W2
Trip3	W1
Trip3	W5
Trip4	W1
Trip5	W4
Trip5	W5
...	...

Truck Table

TruckID	VolCapacity	WeightCategory	CostPerKm
Truck1	250	Medium	$1.20
Truck2	300	Medium	$1.50
Truck3	100	Small	$0.80
Truck4	550	Large	$2.30
Truck5	650	Large	$2.50
...	...	...	...

Store Table

StoreID	StoreName	Address
M1	Myer City	Melbourne
M2	Myer Chaddy	Chadstone
M3	Myer HiPoint	High Point
M4	Myer Donc	Doncaster
M5	Myer North	Northland
M6	Myer South	Southland
M7	Myer East	Eastland
M8	Myer Knox	Knox
...	...	

Destination Table

TripID	StoreID
Trip1	M1
Trip1	M2
Trip1	M4
Trip1	M3
Trip1	M8
Trip2	M4
Trip2	M1
Trip2	M2
...	...

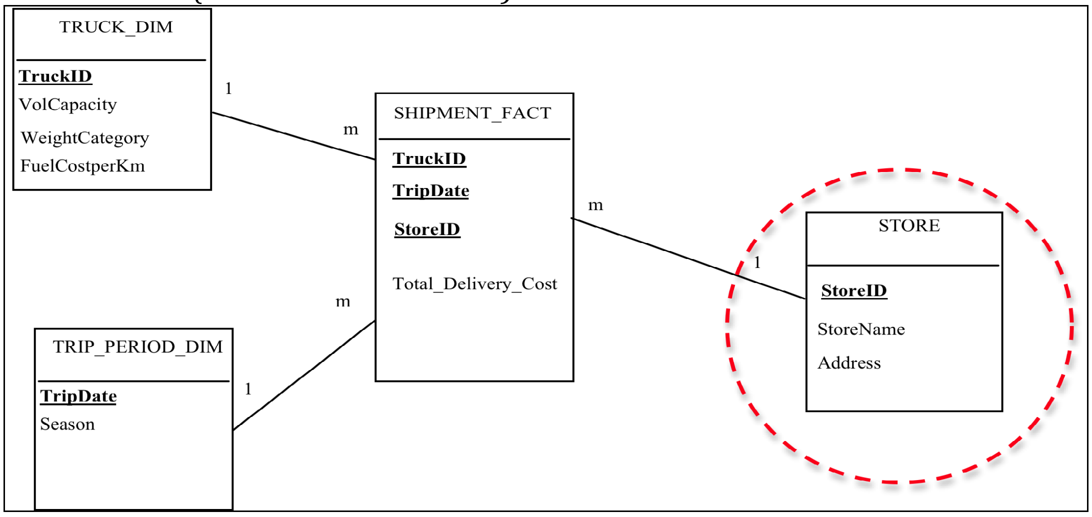
The management of this trucking company would like to analyse the deliver cost, based on Trucks, Period, and Store.

2. Solution Model 1 – using a Bridge Table

The measurable fact to be included in the fact table is "Total Delivery Cost", which is calculated by *distance* (total kilometres in the Trip table) and *cost per kilometre* (from the Truck table). The dimensions are Truck, Trip Period and Store.

A possible star schema is as follows:

Star Schema-1 (An Initial Star Schema):



Based on the sample data above:

- From the Truck point of view, Truck1 has two trips (e.g. Trip1 and Trip4), with the total kilometres of 820km (370km + 450km). The cost for Truck1 is $1.2. Hence, calculating the cost for Truck1 is straightforward. Other trucks can be calculated this way.

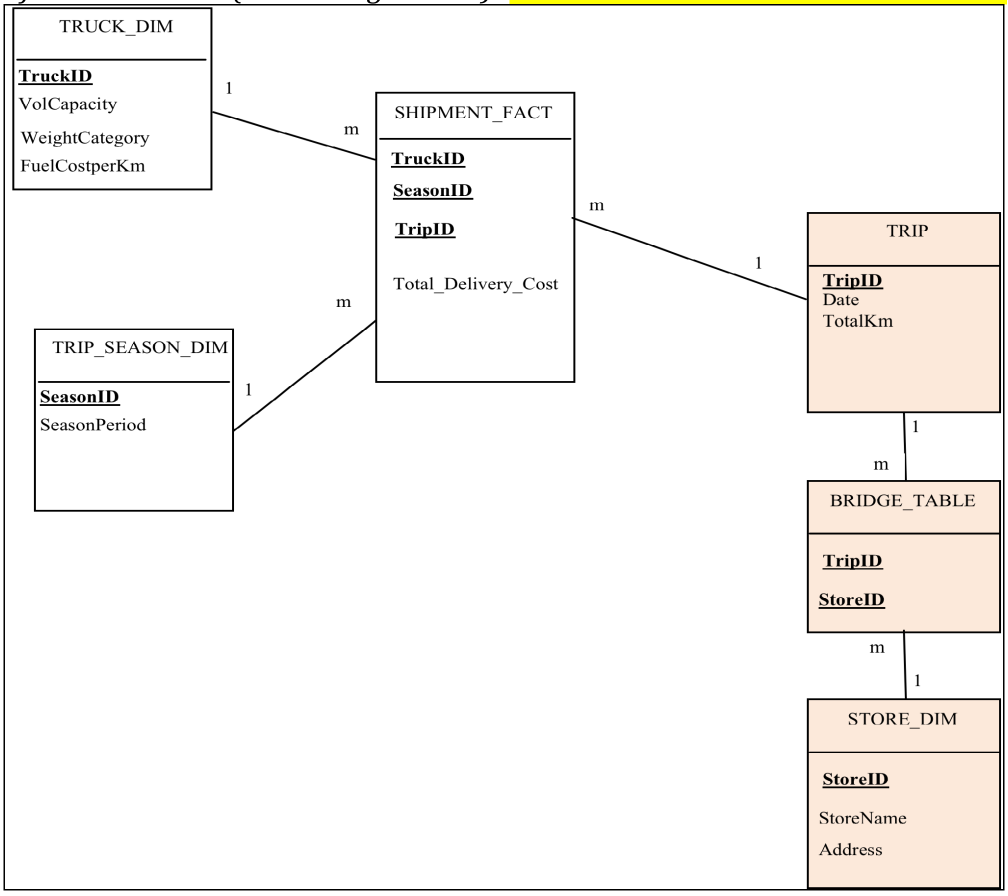
- From the Period point of view (say from a date point of view), 14-Apr-2013 has three trips (e.g. Trip1, Trip2, and Trip3). Trip1 (370km) is delivered by Truck1 which costs $1.2/km. Trip2 and Trip3, on the same day, can be calculated the same way. Hence, on 14-Apr-2013, the total cost can be calculated.

- Now from the Store point of view. The cost is calculated based on Trip, but a trip delivers goods to many stores. Therefore, the delivery cost for each store cannot be calculated. The delivery cost is for the trip; and not for the store.

Therefore, Star Schema-1, which may look correct, is **incorrect**, because there is **NO** direct relationship between Store and total delivery cost in the Fact table. This is due to the many-to-many relationship between Trip Entity and Store Entity in the E/R diagram of the operational database.

To solve this problem, a bridge table can be used. Snowflake Schema-1 with Bridge Table is shown as follows:

*Snowflake Schema-1 (with Bridge Table): The **CORRECT** SNOWFLAKE SCHEMA!!!*



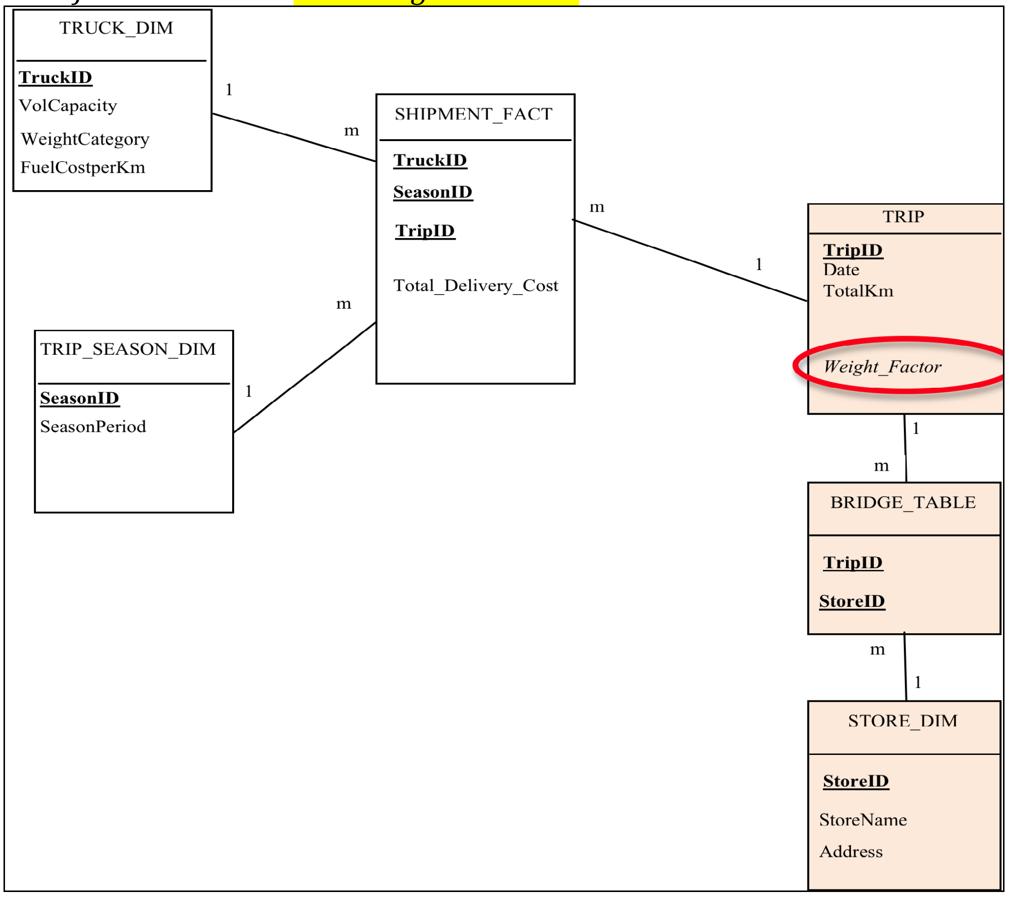
3. Solution Model 2 – add a Weight attribute to the Bridge

Snowflake schema-1 above shows that there is no direct relationship between Total Delivery Cost, and Store. Total delivery cost is calculated at the Trip level, and not at the Store level.

For example, Trip-1 travelled for 370km on 14-April using Truck-1, and Truck-1 cost per kilometre is $1.20. Hence Trip-1 costs $444 on that day. On that day, Trip-1 delivered goods to five stores. Using the current data that we have, it is impossible to calculate the delivery cost for each of these five stores that Trip-1 delivered to. What we can say is that the cost for Trip-1 is $*x* and Trip-1 delivered to *y* number of stores.

However, we can *estimate* the total delivery cost per store, if we want to. This can be estimated through the "Weight Factor" (see the Weight Factor attribute in the Trip Dimension in the Snowflake Schema-2 below).

Snowflake Schema-2: with Weight Factor!!!



What is a "Weight Factor"?
A weight factor is a proportion of the trip that goes to each store for that particular trip. For example, if Trip-1 went to 5 stores, then the Weight Factor is 0.2 (or 20%). This implies that each store "contributes" 20% of the total delivery

cost for that trip. This is certainly not accurate, but this is the only estimate that we can do, based on the data that we have.

Should all snowflake schemas have a "Weight Factor"?
No. A weight factor is only needed if we want to estimate the contribution that a dimension made to the fact.

Using snowflake schema-1 above, if a weight factor is not used, then we will not be able to find out an estimate of the total delivery cost per store. Is missing this information very critical? It depends. After all, the information about total delivery cost per store is an estimate anyway.

How is the data looked like in the Trip, Bridge Table, and Store dimensions?

Trip Dimension

TripID	**Date**	**TotalKM**	**WeightFactor**
Trip1	14-Apr-2013	370	0.2
Trip2	14-Apr-2013	570	0.33
Trip3	14-Apr-2013	250	
…	…	…	…

Bridge_Table

TripID	**StoreID**
Trip1	M1
Trip1	M2
Trip1	M4
Trip1	M3
Trip1	M8
Trip2	M4
Trip2	M1
Trip2	M2
…	…

Store Table

StoreID	StoreName	Address
M1	Myer City	Melb City
M2	Myer Chaddy	Chadstone
M3	Myer HiPoint	HighPoint
M4	Myer Donc	Doncaster
M5	Myer North	Northland
M6	Myer South	Southland
M7	Myer East	Eastland
M8	Myer Knox	Knox
…	…	…

The 0.2 weight factor for Trip 1 does not mean that Trip 1 has 20% of the total delivery cost; rather, Trip 1 delivers to 5 stores, and each store contributes 20% (or 0.2) to the total delivery cost. If this what it means, why not have the Weight Factor attribute in the Bridge Table, such as this – to indicate, for example, that store M1 (from Trip 1) contributes 20% (0.2) to the total delivery cost of Trip 1.

Trip Dimension

TripID	Date	TotalKM
Trip1	14-Apr-2013	370
Trip2	14-Apr-2013	570
Trip3	14-Apr-2013	250
…	…	…

Bridge_Table

TripID	StoreID	WeightFactor
Trip1	M1	0.2
Trip1	M2	0.2
Trip1	M4	0.2
Trip1	M3	0.2
Trip1	M8	0.2
Trip2	M4	0.33
Trip2	M1	0.33
Trip2	M2	0.33
…	…	

The answer is *yes*, we could do that, but then it will have redundant information, because all stores in Trip 1 should have a weight factor of 20%. So, instead of storing the 0.2 in each of the store for Trip 1, we store the 0.2 in the Trip 1 record itself in the Trip Dimension table.

So, a more efficient way is to have the Weight Factor attribute in the Trip Dimension table, like this:

Trip Dimension

TripID	Date	TotalKM	WeightFactor
Trip1	14-Apr-2013	370	0.2
Trip2	14-Apr-2013	570	0.33
Trip3	14-Apr-2013	250	
…	…	…	…

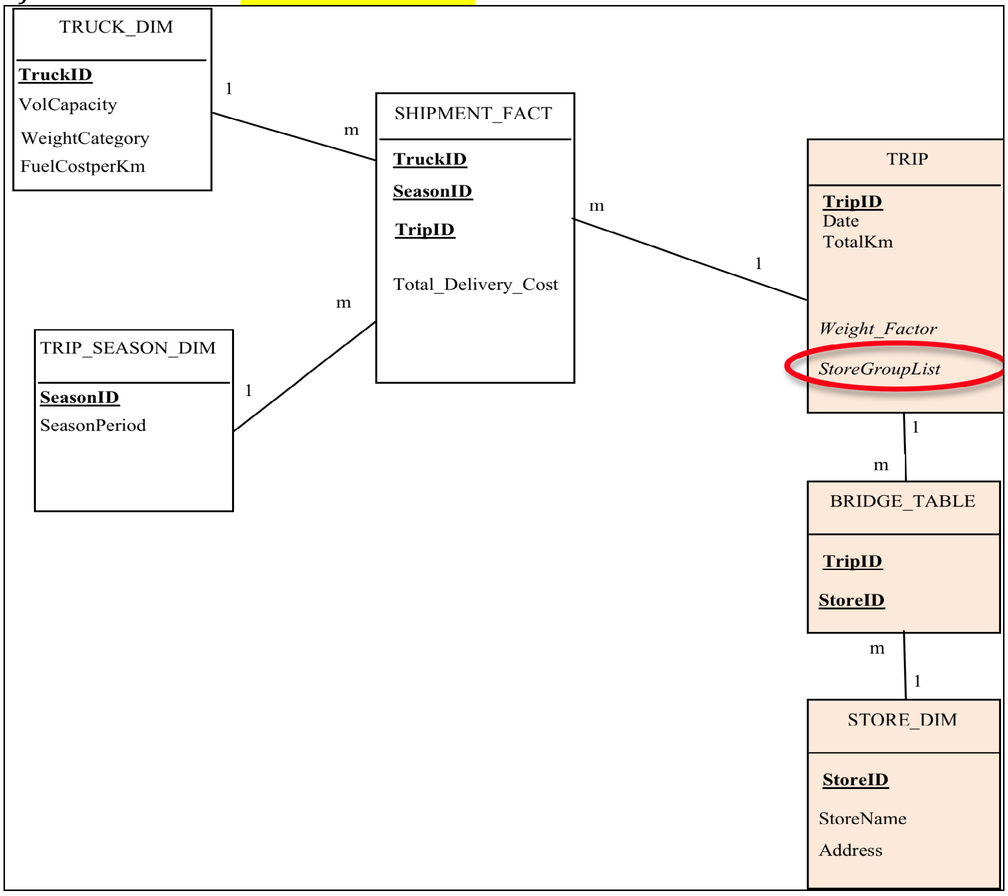
To create the Trip dimension table having the WeightFactor attribute is done by the following SQL:

```
Create Table TripDim2 As
Select
   t.tripid,
   t.tripdate,
   t.totalkm,
   1.0/count(*) as weightfactor
From trip t, destination d
Where t.tripid = d.tripid
Group by t.tripid, t.tripdate, t.totalkm;
```

4. Solution Model 3 – A ListAGG version

In a ListAGG option, we have one additional attribution in the parent dimension (e.g. Trip Dimension) which holds the information of the group of each record in the parent dimension table (e.g. Stores within each trip).

Snowflake Schema-3: A ListAGG version!!!



In snowflake schema-3, an attribute called "StoreGroupList" is added to the Trip dimension. In this attribute, we keep all stores for each Trip.

Trip Dimension

TripID	**Date**	**TotalKM**	WeightFactor	StoreGroupList
Trip1	14-Apr-2013	370	0.2	M1_M2_M3_M4_M8
Trip2	14-Apr-2013	570	0.33	M1_M2_M4
Trip3	14-Apr-2013	250		
...	...	...	...	...

Bridge_Table

TripID	**StoreID**
Trip1	M1
Trip1	M2
Trip1	M4
Trip1	M3
Trip1	M8
Trip2	M4
Trip2	M1
Trip2	M2
...	...

Store Table

StoreID	StoreName	Address
M1	Myer City	Melb City
M2	Myer Chaddy	Chadstone
M3	Myer HiPoint	HighPoint
M4	Myer Donc	Doncaster
M5	Myer North	Northland
M6	Myer South	Southland
M7	Myer East	Eastland
M8	Myer Knox	Knox
…	…	…

To create the Trip dimension table having the StoreGroupList attribute, we can use the ListAGG function in SQL. The ListAGG function has the following format:

```
LISTAGG (attr1, '_') Within Group (Order By attr1) As columnname
```

Where attr1 is the StoreID, and the '_' is to indicate that the store IDs are concatenated with the '_' symbol (e.g. M1_M2_M3_M4_M8). If we want to have the stores listed in a descending order (e.g. M8_M4_M3_M2_M1), then we use Within Group (Order By attr1 Desc).

The SQL to create the Trip dimension table becomes:

```
Create Table TripDim3 As
Select
   T.TripID,
   T.TripDate,
   T.TotalKm,
   1.0/count(D.StoreID) As WeightFactor,
   LISTAGG (D.StoreID, '_') Within Group
     (Order By D.StoreID) As StoreGroupList
From Trip T, Destination D
Where T.TripID = D.TripID
Group By T.TripID, T.TripDate, T.TotalKm;
```

Visually, the ListAGG attribute (e.g. the storegrouplist attribute in the Trip dimension) is appealing, because it is very easy to see the complete list of stores for each trip. However, this information is rather redundant, because we can get the same information from the Bridge Table.

Snowflake schema-1 and snowflake schema-2 previously shown are actually a non-listAGG version. A non-ListAGG version is simpler and cleaner. The implementation is simpler and more straightforward. Additionally, there is no redundant information about stores within each trip.

However, in industry, the ListAGG version is often a preferred option. It is because a group list is physically listed in the parent dimension, which may visually help the decision makers to understand the completeness of the group list (e.g. storeID for each trip). Unfortunately, from an implementation point of view, creating a ListAGG can be very complex. Nevertheless, the ListAGG version is not uncommon in industry. Additionally, when producing a report, if we would like to join the Trip and Store dimensions, using the ListAGG attribute only needs two tables: the trip and store dimension tables, without the need for joining with the bridge table. The

```
Select *
From trip_dim3 t, store_dim3 s
Where t.storegrouplist LIKE '%'||s.storeid||'%';
```

In this SQL, the joining is based on the storegrouplist attribute in the Trip dimension table, and the storied in the Store dimension table. However, the join condition is not a simple `t.storegrouplist = s.storeid`, because the join condition is not a simple match. Instead, we use a LIKE operator to check if the storeID value exists in the storegrouplist.

In contrast, without the storegrouplist attribute in the Trip dimension, we need to join three tables: the Trip dimension, the Bridge table, and the Store dimension tables, using the following SQL:

```
Select *
From trip_dim3 t, bridge_table3 b, store_dim3 s
Where t.tripid = b.tripid
And b.storeid = s.storeid;
```

5. Summary

(i) A Bridge Table is needed if we cannot link a dimension directly to the fact.

(ii) A Weight Factor is used to estimate the contribution of a dimension in the calculation of the measurable fact. Because this is only an estimate, a weight factor is option.

(iii) Every snowflake schema (whether it has Weight Factor or not) can be implemented in two ways: a ListAGG version, and a NON-ListAGG version.