Part V
Data Warehousing Granularity and Evolution

Chapter 14
Data Warehousing Granularity and Levels of Aggregation

The central concept of data warehousing is *Aggregation*. In the previous chapters, we have been focusing on fact measures as the aggregated values. Since fact measures are the focus of star schemas, data warehousing is then about aggregation and aggregate values.

Digging deeper into the nature of aggregate values, aggregate values have different levels of granularity. For example, Total Sales per Year has lower granularity than Total Sales per Quarter; or Number of Logins in the lab per Semester has a different level of focus (or granularity) than Number of Logins in the lab per Month. Consequently, there is a notion of levels of granularity, or level of detail, and in the context of data warehousing, we use the term *levels of aggregation*. This chapter focuses on levels of aggregation, which is basically the foundation of *data warehousing granularity*.

14.1 Levels of Aggregation

Let's use the Computer Lab Usage case study to illustrate the concept of *levels of aggregation* and *data warehousing granularity*. Figure 14.1 shows a simple operational database that keeps track of the usage of computers in the labs.

This computer lab usage operational database has only four tables. Student Table is a typical table that stores student details, including the Degree in which the student is enrolled and the Study Type (e.g. Full-Time, Part-Time, Online, etc.). The Degree Code attribute is obviously a Foreign Key to the Degree Table that stores a list of degrees offered by the university (e.g. Bachelor of Computer Science, Master of Information Technology, etc.). The Computer Table stores the details of each computer in the labs, including Computer Model, Operating Systems, etc. The transaction table is the Lab Activities table which keeps the details of every student who logged in to each computer. Each time a student logs in, a Login No is created

D. Taniar, W. Rahayu, *Data Warehousing and Analytics*, Data-Centric Systems and Applications, https://doi.org/10.1007/978-3-030-81979-8_14

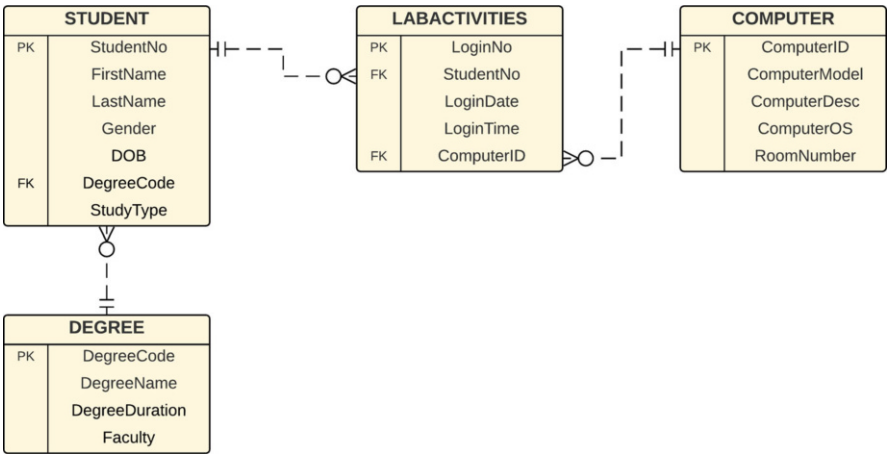

Fig. 14.1 An E/R diagram of the Computer Lab Usage

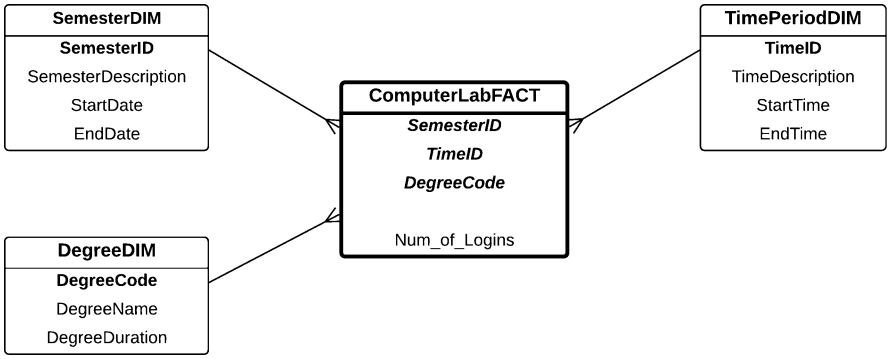

Fig. 14.2 The Computer Lab Usage star schema

and other details are stored, including Student No, Login Date and Time as well as ComputerID.

A star schema is created to analyse the usage of computers in the labs. For simplicity, only one fact measure is included in the star schema, that is, Num of Logins. The dimensions are Semester Dimension (e.g. semester 1, semester 2), Time Period Dimension (e.g. morning, afternoon, night) and Degree Dimension. The star schema is shown in Fig. 14.2. Using this simple star schema, we can query the number of logins per semester, per time period and per degree. Because we count the number of logins, the same student who logins several times will be counted as many times as the student logs in.

Data warehousing is basically precomputed aggregate values. The fact measure, Num of Logins, is basically a precomputed value by counting the number of records in the transaction table, namely the Lab Activities table, grouped by Semester, Time

Period and Degree. With this precomputed aggregate value, querying to the star schema can directly obtain the values (i.e. number of logins), by retrieving the fact measure, by specifying the desired dimensions, whether it be Semester, Time Period and/or Degree. With this precomputed aggregate value, the retrieval of the Num of Logins can be done more efficiently. This is why a data warehouse is preferred by management in relation to the decision-making process, simply because the required information is already precomputed. To be more specific, a data warehouse is built through a transformation process from the operational database by extracting, cleaning and aggregating the original data in the operational databases. This is why decision-making does not normally query operational databases because decision-making queries usually focus on aggregate values and it will be more convenient if the required information is readily available in the data warehouse because the data warehouse is already cleaned, transformed, aggregated, etc.

Because decision support queries usually focus on aggregate values, the Fact Tables often contain highly aggregated values, such as Num of Logins, grouped by Semester, Time Period and Degree. When management finds interesting data in the Fact Table, they often want to drill down to find more detailed information. For example, in the Computer Lab Usage case study, if the number of logins at night in semester 1 is rather high, we might want to see the breakdown, such as the number of logins per hour at night, etc. The main question is how to do this. Unfortunately, because the Fact Table does not store this information (e.g. the hour by hour breakdown), the obvious answer is to drill down into the operational database. However, this is not a preferable practice because the operational database might not be easily accessible (note that a data warehouse might be built from several operational databases and some operational databases are not even a database system, such as an Excel spreadsheet, etc.). Additionally, the data from the operational databases have been cleaned, transformed, etc. when building the data warehouse. Therefore, accessing the operational database for drilling down purposes is not desirable.

To address this issue, the concept of *data warehousing granularity* is introduced. A data warehousing granularity consists of a number of granularity layers of a star schema. The highest granularity star schema which contains the most detailed data is known as **Level-0**. **Level-1** is built on top of Level-0 star schemas, and Level-1 star schemas have a lower granularity of the fact measure (e.g. less detail data as the data is already aggregated). Subsequently, **Level-2** star schemas are built on top of Level-1 star schemas and have an even lower granularity of the fact measures. Figure 14.3 shows data warehousing granularity consisting of several levels of star schemas in various levels of aggregations.

So, Level-0 has the highest level of granularity where no aggregation exists. All domain tables are incorporated as dimension tables, and there is no grouping in the dimension attributes. Basically, Level-0 means no aggregation. Since Level-0 has no aggregation, Level-0 star schema is almost identical to the E/R diagram but with a different structure. Level-0 star schema is structured in a star schema model, whereas the E/R diagram is not. But in essence, both Level-0 star schema and the

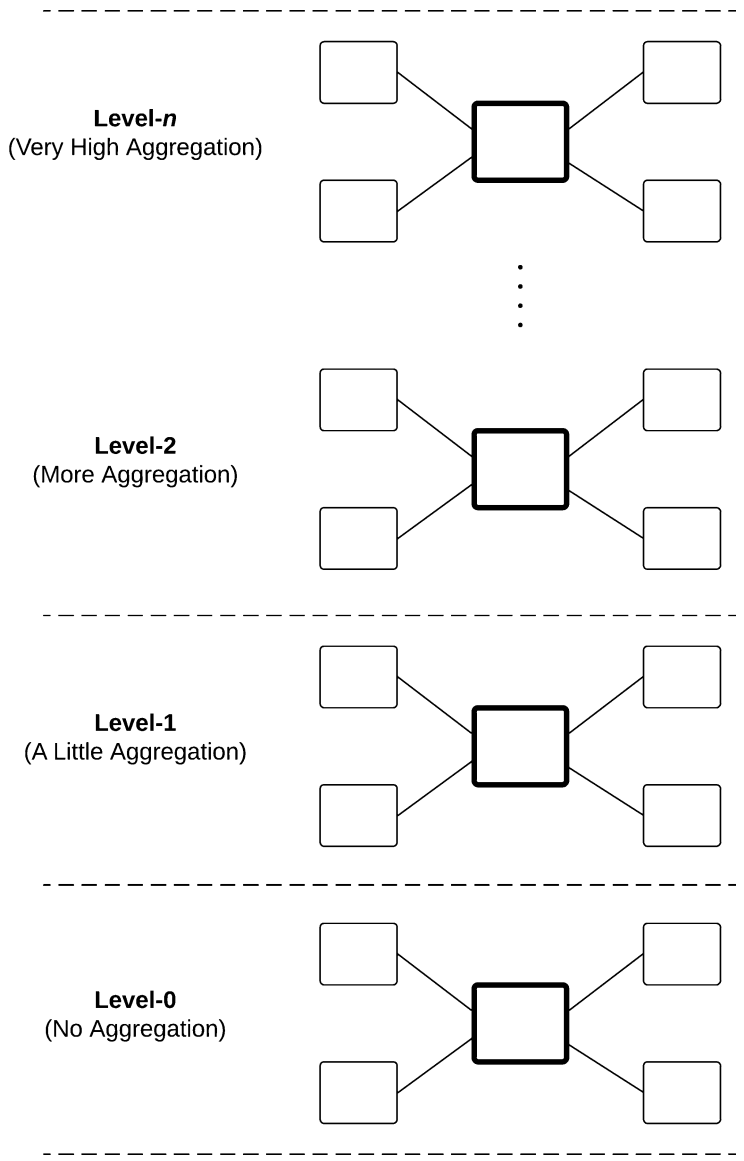

Fig. 14.3 Data warehousing granularity

corresponding E/R diagram convey the same information at the same granularity level.

The lower the level of granularity, the higher the level of aggregation. Because star schemas and data warehousing are about aggregation and aggregate values, we will use the term **level of aggregation** to refer to Level-0, Level-1, Level-2, etc. (instead of level of granularity). The higher the level of aggregation, the higher the level number (e.g. Level-2 has more aggregation than Level-1, Level-3 has more aggregation than Level-2, etc.). There is no particular rule about determining the level of aggregation (with the exception that Level-0 always means no aggregation). The numbering is a sequence number. Therefore, given a star schema, we can only determine whether it is Level-0 or not Level-0. If it is not Level-0, then it depends on how many non-Level-0 there are below this star schema and this will determine the position of this star schema in the data warehousing architecture.

Now let's return to the Computer Lab Usage star schema as shown in Fig. 14.2. This star schema is definitely not a Level-0 star schema because the fact measure is an aggregate value. When creating the Fact Table, a `count` function is used in the SQL and a `group by` SemesterID, TimeID and Degree Code attributes. This star schema must be a higher level of aggregation (e.g. Level-1 or Level-2 or higher). This is the reason why we are not able to drill down into the Num of Logins when using this star schema. To be able to drill down to a lower level of aggregation, we need to query the lower level of aggregation of this star schema. Therefore, we must create another star schema with a lower level of aggregation.

There are two ways to lower the level of aggregations of a star schema:

1. *Add a new dimension*. When we add a new dimension, each value in the fact measure will literally be broken down more details on each record of the new dimension.
2. *Replace an existing dimension with a higher granularity dimension*. The values of the fact measures will also be broken down more details because the fact measure has a lower detail dimension.

Using the star schema in Fig. 14.2, to lower this star schema, we could add a new dimension called Student Dimension (see a new star schema in Fig. 14.4). For simplicity, the star schema in Fig. 14.2 is **Level-2**, and the new star schema in Fig. 14.4 is **Level-1**. This Level-1 star schema is created by adding other information available from the tables in the operational database. We can add the student information, and this will subsequently lower the level of aggregation of the Fact Table. This Level-1 star schema is the drill down of the Num of Logins fact measure, which provides the details of who (which students) contributed to this Num of Logins fact measure. In other words, the Fact Table now contains the StudentID. Note that the lower the level of aggregation, the more data we will have in the Fact Table.

To lower the Level-1 star schema, in this example, we use the second method for lowering the aggregation level, that is, by changing the level of granularity of the existing dimensions. The Semester Dimension is now changed to the highest level of granularity, namely Login Date Dimension. This means that the level of aggregation

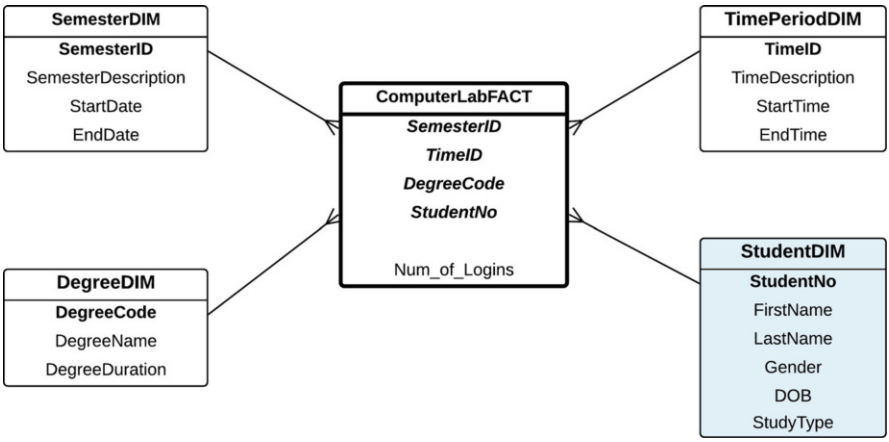

Fig. 14.4 The Computer Lab Usage star schema—a lower level of aggregation

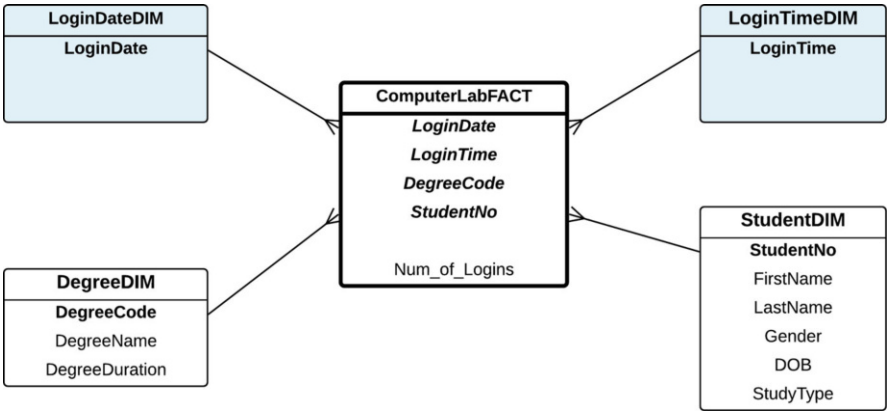

Fig. 14.5 The Computer Lab Usage star schema—Level-0

of the Num of Logins has now dropped to the actual login date, which is the lowest possible level of detail. Note that Level-0 must use the lowest dimension level, and the Login Date Dimension is the lowest from the Semester Dimension perspective.

Another dimension that needs to be lowered is the Time Period Dimension. At the moment, the Time Period Dimension indicates either morning, afternoon or night only (e.g. TimeID 1, 2, 3 represents morning, afternoon, night, respectively). This is a high level of abstraction. To lower this dimension, a Login Time Dimension is used which represents the actual login time for each login.

Level-0 star schema is shown in Fig. 14.5. In this example, Level-0 star schema is the drilling down of the semester and time period which should now contain the actual login date (instead of semester) and the actual login time (instead of time period). This is the level where we do not have any grouping of dimension attributes, and hence Level-0.

In summary, Level-0 star schema provides the most detailed information about the data warehouse. The upper levels provide some levels of aggregated information which have a higher level of aggregation. There is no particular guideline on how many levels we should have in the data warehousing architecture. It can be less or more than three, depending on how many levels of aggregation are needed. Additionally, there is no particular rule on what kind of aggregation is needed on a particular level (except that Level-0 must not have any aggregation). The only rule is that level $(x + 1)$ star schema is more aggregated than level x star schema. Moreover, there could be more than one star schema in each level. For example, Level-0 has one star schema, but Level-1 might have five star schemas, and Level-2 might have ten star schemas.

The upper-level star schema is often implemented as a *dashboard*, where users are able to interact with the dashboard which provides users with highly aggregated information for decision-making. In the implementation, Level-0 star schema (dimensions and Fact Tables) is implemented by the `create table` statements in SQL, whereas the upper-level star schemas are often implemented as a view (implemented by the `create view` statements in SQL).

Now let's compare the three levels of the Computer Lab Usage star schemas, by exploring the differences in their respective Fact Tables. Sample data in the Fact Table of the Level-2 star schema are shown in Table 14.1. Note that *S*1 is Semester 1, and TimeID 3 indicates an evening time period.

Suppose that the management retrieves this data from the Fact Table and is interested in knowing further details about Num of Logins of 1500 which is the number of students in lab in Semester 1, evening, doing a Bachelor of IT. In other words, the management would like to drill down further to find out more information about these 1500 logins. We cannot do this in Level-2 because the Level-2 Fact Table only has data about SemesterID, TimeID and DegreeCode. In order to drill down further, we need a lower level of aggregation of star schema (e.g. Level-1 star schema).

So, a sample data for Level-1 star schema is as follows (see Table 14.2). It basically breaks down the 1500 logins (semester 1, evening, Bachelor of IT) into a number of login figures based on Student No. The total of logins of semester 1, evening, Bachelor of IT must be equal to 1500.

Now suppose the management would like to drill down into the 120 logins (see the highlighted cell in Table 14.2). In particular, management would like to see the actual date and time of login. Then we must go to Level-0 star schema. Level-0 Fact Table is shown in Table 14.3.

Table 14.1 Level-2 Fact Table

SemesterID	TimeID	DegreeCode	Num of logins
S1	3	BIT	**1500**
S1	3	BEng	1250
S1	3	BSc	788
S1	3	MBA	980
...	...	...	...

Table 14.2 Level-1 Fact Table

SemesterID	TimeID	DegreeCode	StudentNo	Num of logins
S1	3	BIT	21002	**120**
S1	3	BIT	21023	90
S1	3	BIT	21025	55
S1	3	BIT	21066	37
S1	3	BIT	...	...
S1	3	BIT	...	...
...	...	...	...	...

Table 14.3 Level-0 Fact Table

Login date	Login time	DegreeCode	StudentNo	Num of logins
4-Apr	19:00	BIT	21002	**1**
6-Apr	21:20	BIT	21002	**1**
8-Apr	02:30	BIT	21002	**1**
3-May	18:55	BIT	21002	**1**
7-May	19:30	BIT	21002	**1**
8-May	19:45	BIT	21002	**1**
...	...	...	...	...

In this example, S1 is broken down into the actual dates that represent Semester 1, and TimeID 3 (night) is broken down into the actual login time that represents the time period night. It is natural that the fact measure in Level-0 star schema contains 1s only, and this shows no aggregation because there is only one login per student at each particular date and time. The total number of logins for this particular student in this example (in Semester 1 at night) must be equal to 120 as indicated by the Level-1 Fact Table.

These three Fact Tables illustrate how the values of the fact measure (e.g. Num of Logins in this example) are broken down into more detail in the lower levels of aggregations. Because the fact measure is a `count`, the fact measure in Level-0 contains all 1s.

As previously mentioned, there could be more than one star schema at each level of aggregation. For example, we could have a new level in between Level-0 and Level-1 in the Computer Lab Usage case study. The current Level-1 has SemesterID ($S1$ and $S2$) and TimeID (1 = morning, 2 = afternoon, 3 = night), whereas Level-0 has the actual LoginDate and the actual LoginTime. We could have an intermediate level where the Semester is broken down into months rather than directly into the actual LoginDate. Similarly, TimeID (morning, afternoon, night) can be broken down into hours rather than directly into the actual LoginTime. Hence, we could have two star schemas in the new Level-1 (note that the old Level-1 becomes Level-2 and the old Level-2 becomes Level-3), as shown in Tables 14.4–14.8.

Note that there are two star schemas in the new Level-1, Level-1a and Level-1b. Level-1a is a lower version of Level-2 because SemesterID in Level-2 is lower than MonthID in Level-1a. In Level-2, the data on the student who has 120 logins

Table 14.4 The new Level-3 Fact Table

SemesterID	TimeID	DegreeCode	Num of logins
S1	3	BIT	**1500**
S1	3	BEng	1250
S1	3	BSc	788
S1	3	MBA	980
...	...	...	...

Table 14.5 The new Level-2 Fact Table

SemesterID	TimeID	DegreeCode	StudentNo	Num of logins
S1	3	BIT	21002	**120**
S1	3	BIT	21023	90
S1	3	BIT	21025	55
S1	3	BIT	21066	37
S1	3	BIT	...	...
S1	3	BIT	...	...
...	...	...	...	...

Table 14.6 The new Level-1a Fact Table

MonthID	TimeID	DegreeCode	StudentNo	Num of logins
Jan	3	BIT	21002	**10**
Feb	3	BIT	21002	**5**
Mar	3	BIT	21002	**7**
Apr	3	BIT	21002	**10**
May	3	BIT	21002	**25**
Jun	3	BIT	21002	**20**
Jul	3	BIT	21002	**34**
...	...	...	...	...

Table 14.7 The new Level-1b Fact Table

SemesterID	HourID	DegreeCode	StudentNo	Num of logins
S1	6:00-6:59pm	BIT	21002	**25**
S1	7:00-7:59pm	BIT	21002	**10**
S1	8:00-8:59pm	BIT	21002	**5**
S1	9:00-9:59pm	BIT	21002	**5**
S1	10:00-10:59pm	BIT	21002	**3**
...	...	...	...	...

is broken down into each month in Level-1a. If the duration of Semester 1 is from 1-Jan to 14-Jul and Semester 2 is from 15-Jul to 31-Dec, then the total logins for this student from Jan to Jul will not necessarily be equal to 120 because the month of July covers two semesters.

Table 14.8 Level-0 Fact Table

Login date	Login time	DegreeCode	StudentNo	Num of logins
4-Apr	19:00	BIT	21002	**1**
6-Apr	21:20	BIT	21002	**1**
8-Apr	02:30	BIT	21002	**1**
3-May	18:55	BIT	21002	**1**
7-May	19:30	BIT	21002	**1**
8-May	19:45	BIT	21002	**1**
...	...	...	...	...

Although the month of July covers two semesters, the fact measure (Num of Logins) will not be double counted, because the raw records will be assigned correctly to the month, without double allocation. Consequently, the Month Dimension does not need to be a Determinant Dimension.

The new Level-1b does not lower down to Semester; rather, it lowers down to TimeID, instead. So, instead of TimeID of morning, afternoon, night, in Level-1b, it uses HourID in which the TimeID is broken down into an hourly base but the SemesterID is kept. The number of logins for this student in Level-1b must be equal to 120, as indicated in Level-2.

Comparing Level-1a and Level-1b, it cannot be said that one level is higher than the other; they are simply not comparable, because MonthID in Level-1a is lower than SemesterID in Level-1b, but TimeID in Level-1a is not lower than HourID in Level-1b. Hence, they are not comparable. But what we are certain of is that both star schemas (Level-1a and Level-1b) are lower than Level-2 and higher than Level-0.

Level-0 Fact Table is not changed by the introduction of the new Level-1a and Level-1b star schemas.

14.2 Facts Without Fact Measures

Looking at the Level-0 star schema of the Computer Lab Usage case study, as shown in Fig. 14.5, we can see that the fact measure, which is Num of Logins, has all 1s in Table 14.8. This means that there is only 1 login for a particular student (who is enrolled in a particular degree) at a certain date and time. Because it is Level-0, this means there is no aggregation, and as a result, each Num of Logins must be equal to 1.

If the fact measure only has the value of 1, then the fact measure may be excluded from the fact. Consequently, the Level-0 star schema of the Computer Lab Usage case study has no fact measure (see the new star schema in Fig. 14.6). This is then known as **facts without fact measures**. Facts without fact measures are only possible if the fact measure is a `count`, such as Num of Logins in this case. If the

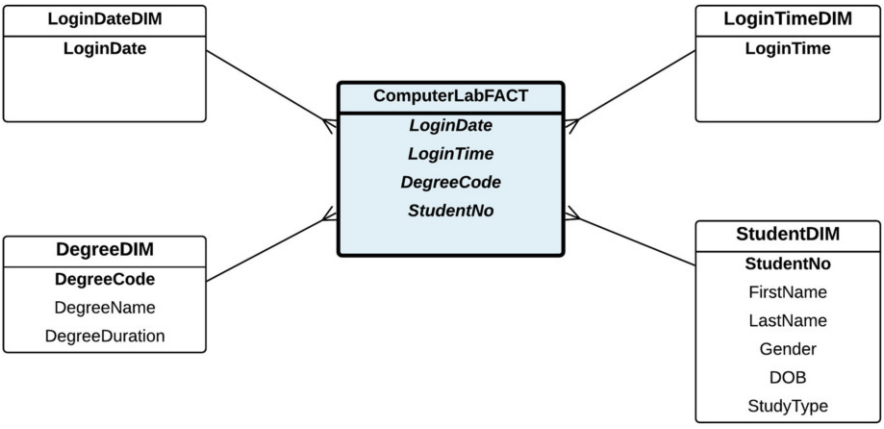

Fig. 14.6 Facts without fact measures

fact measure is not a count, but some other aggregate functions such as `sum` (e.g. Total Sales), then Level-0 star schema must still include the fact measure (e.g. the amount of sales of each sales).

When designing Level-0 star schema, people often choose the ID as the smallest unit, such as StudentID as the fact measure. The reason is that later in Level-1, when the granularity is reduced, the StudentID will be replaced by an aggregate function (e.g. `count`), and therefore the fact measure will become Number of Students. However, this kind of design for Level-0 using StudentID as the fact measure is not desirable, because StudentID is linked to a dimension, namely Student Dimension. Fact measure should not be linked to any dimension. So, in this case, it is more desirable to use Fact without Fact Measure, as described above.

The star schema in Fig. 14.6 can be made more compact by combining LoginDate and LoginTime Dimensions into one dimension, called Lab Activities Dimension, because login date and time attributes are part of the Lab Activities entity in the E/R diagram. The new star schema is shown in Fig. 14.7. It does not change the level of granularity because there is only one login for each LoginID.

The most complete Level-0 star schema for the Computer Lab Usage case study is shown in Fig. 14.8. It has four dimensions, including the new Computer Dimension. A comparison of the E/R diagram of the Computer Lab Usage case study (see Fig. 14.1) shows that the star schema in Fig. 14.8 contains all the information from the E/R diagram, but it is structured differently in the star schema. A star schema is basically an n-ary relationship between all dimensions with the fact, and the fact actually represents the transactions in the E/R diagram.

When designing a data warehouse, we can start designing from a higher level of aggregation, as shown in this book. Lowering down the level of aggregation can be done either by adding a new dimension to the star schema or by changing the level of granularity of an existing dimension. However, there is a second method for designing a data warehouse, that is, by starting from Level-0 by converting an

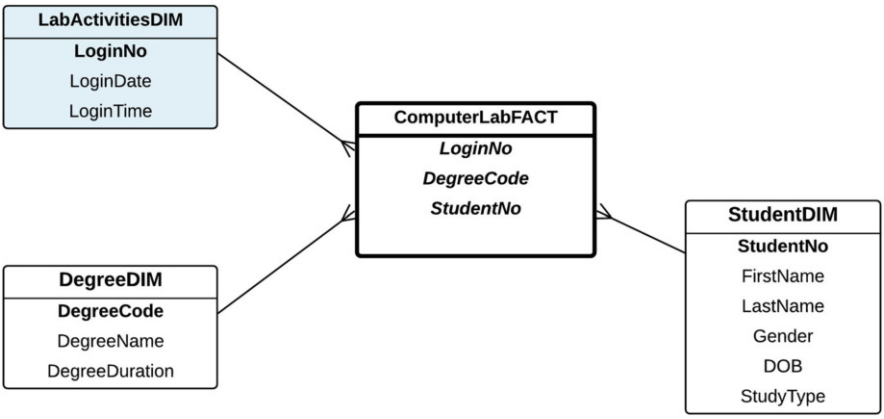

Fig. 14.7 Facts without fact measures—combining LoginDate and LoginTime into LabActivities

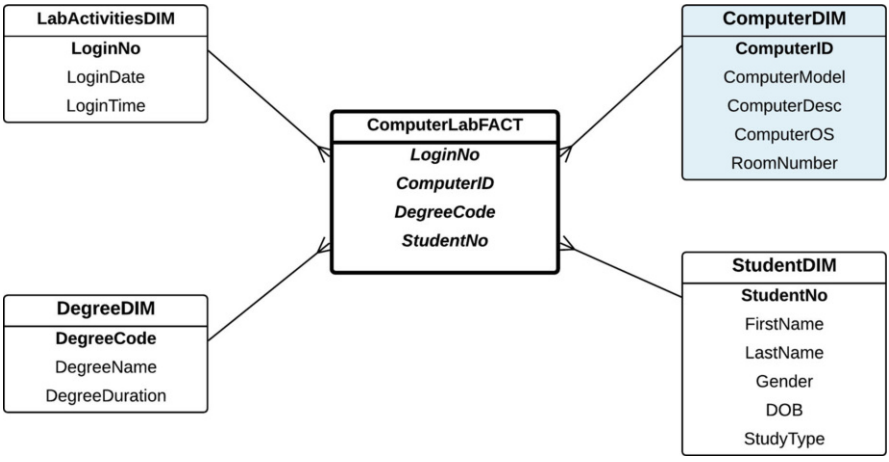

Fig. 14.8 Fact without Fact Measure—most complete star schema

E/R diagram into a star schema structure, which is an n-ary relationship between all dimensions and the fact. Then to move up a level, the opposite method is applied, that is, changing the level of granularity of a dimension or removing an existing dimension.

14.3 Star Schemas with No Aggregation

When designing a data warehouse, it is common to begin with a star schema at a high level of aggregation. If we started from Level-0, the data in the data warehouse is basically identical to the data in the operational database, only with a different way

of structuring it. The reason why it is the same is because Level-0 star schema has no aggregation. If we started designing a star schema from Level-0, the question as to why we need a star schema in the first place is often asked. This is why designing a star schema often starts from a high level of aggregation, where it is clear that the fact measures are basically aggregated values.

Then to lower the level of aggregation of a star schema, two options are usually available: (i) add a new dimension to the star schema, or (ii) change the level of granularity of an existing dimension to a higher granularity (or more detailed dimension). If we keep lowering the level of aggregation of a star schema, it will ultimately reach the lowest level of aggregation, known as **Level-0** of aggregation, which is no aggregation in the star schema. Level-0 star schema has all the fact measures not aggregated.

This section examines some complexities and pitfalls of identifying whether or not a star schema is on Level-0. Consider the following Purchase Order case study as an example. A typical E/R diagram of the Purchase Order operational database is shown in Fig. 14.9. A Customer may have several Purchase Orders (POs). The Customer entity contains typical customer details; the Purchase Order entity contains OrderID, Order Date, Pay Method (e.g. credit card, cash, etc.), Ordering Method (e.g. Online Sales, Direct Sales, etc.) and a reference to CustID.

Each Purchase Order may contain several Order Lines, and each Order Line is an Item with a specified quantity of order. The relationship between Purchase Order and Item is a *m-m* through the Order Line entity. The Order Line entity records the Order Price and Quantity of each Item being Purchase Ordered. Each item contains ItemID, QOH (Quantity On Hand), ProductID, Size and Colour. Finally, the Product entity which consists of ProductID, Current Price, Description and Category may

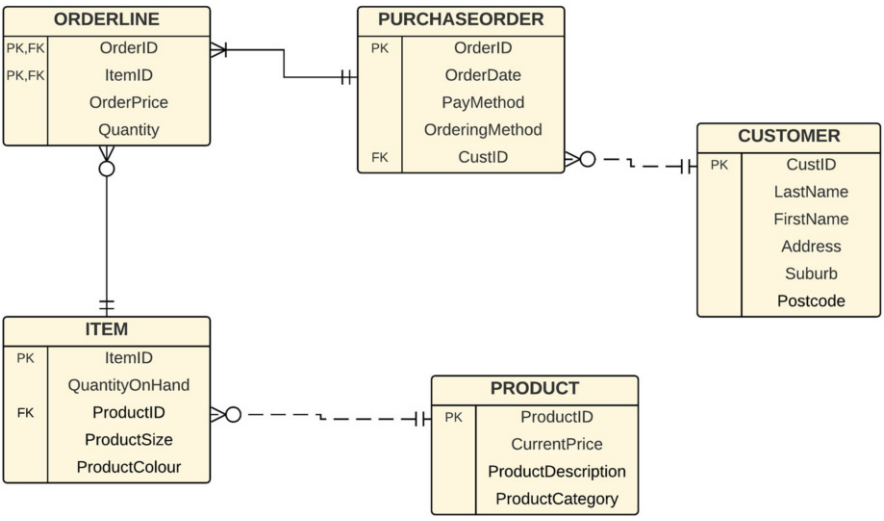

Fig. 14.9 Purchase Order E/R diagram

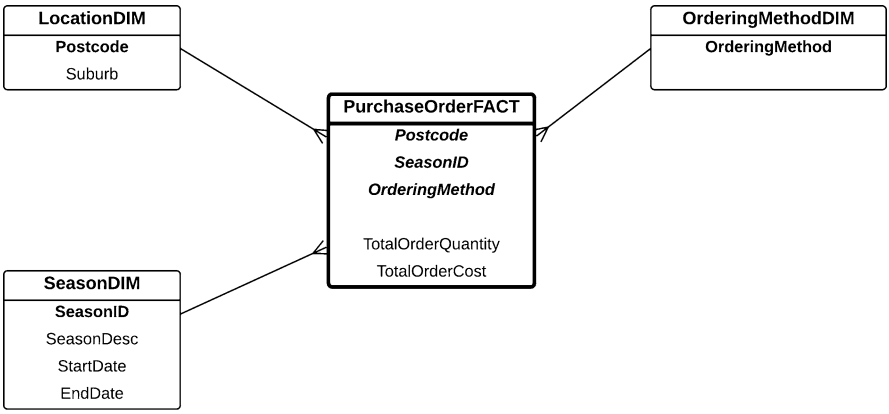

Fig. 14.10 Purchase Order star schema—high level of aggregation

have several Items. In other words, Product is an abstraction and Item is a physical item, which is why Product-Item cardinality is 1-*m*.

A simple star schema captures two fact measures, Total Order Quantity and Total Order Cost (both can be calculated by summing the Order Price and Quantity attributes from the Order Line table). For simplicity, the star schema includes three dimensions, Location Dimension (which is the Suburb and Postcode of Customer), Season and Ordering Method (e.g. Online Sales, Direct Sales, etc.). The star schema is shown in Fig. 14.10 (for simplicity, the Ordering Method Dimension includes one attribute only). This star schema is at a high level of aggregation, because the granularity (or the details) of the dimensions is rather low. For example, the Total Order Quantity is shown at a Season level or at a Suburb level; hence, they are quite general. It is quite common to start designing a star schema at this high level.

To make this star schema more detailed by lowering the level of aggregation, we can change the granularity of the Season Dimension to make it a higher granularity (or more detailed). For example, we change from Season Dimension to Order Date Dimension. This lower level of aggregation of the star schema is shown in Fig. 14.11 (for simplicity, the Order Date Dimension has one attribute only).

For the time being, we shall denote the star schema with Season Dimension (Fig. 14.10) as **Level-2** and the star schema with Order Date Dimension (Fig. 14.11) as **Level-1**, where Level-1 has a lower level of aggregation than Level-2.

From the time dimension point of view, Order Date is already at the highest level of granularity because it cannot be broken down further. The Ordering Method Dimension is also at the highest level of granularity. Hence, to lower down the Level-1 star schema (Fig. 14.11), we choose the Location Dimension, and in this case, we need to make the Location Dimension more detailed (or a higher granularity). Currently, the granularity of the Location Dimension is actually at the Suburb level. To make the Suburb more detailed, we need to drill down further to the addresses within each suburb. Assuming the address of each customer is unique

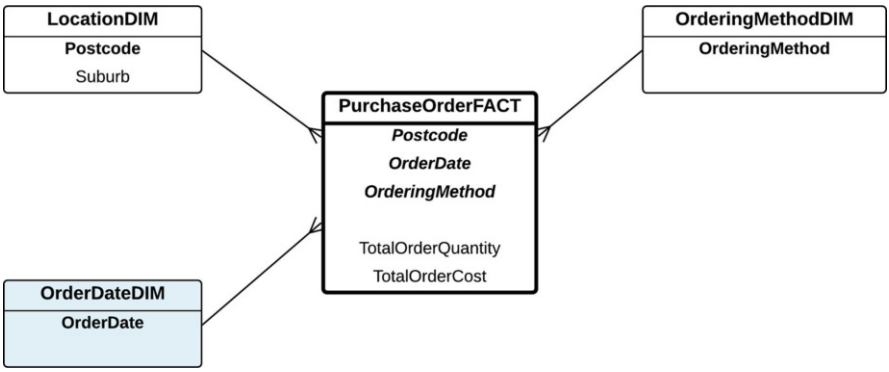

Fig. 14.11 Purchase Order star schema—with Order Date Dimension

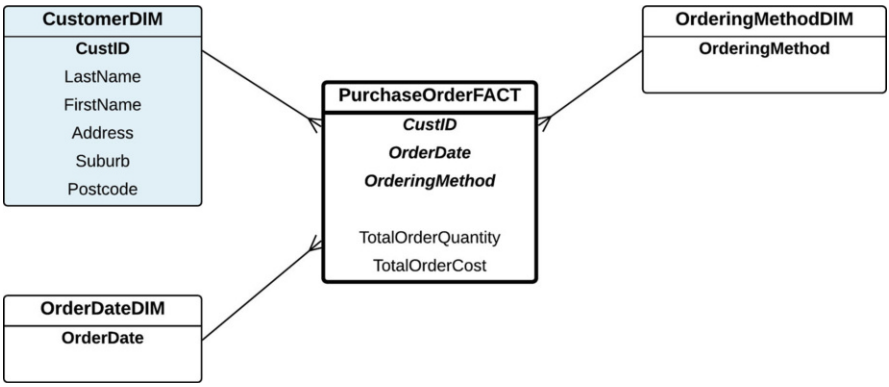

Fig. 14.12 Purchase Order star schema—with Customer Dimension

(hence, each customer is one household), to make the Location Dimension (Suburb) a higher granularity, we can use Customer as a dimension, and hence, Customer Dimension. The new star schema is shown in Fig. 14.12, which is lower than the Level-1 star schema in Fig. 14.11.

As a result, there are three star schemas at different levels of aggregation: **Level-2** (Fig. 14.10) with Season Dimension, **Level-1** (Fig. 14.11) with Order Date Dimension and **Level-0** (Fig. 14.12) with Customer Dimension. The method used to lower the level of aggregation is by changing the level of granularity of a dimension.

Is the star schema in Fig. 14.12 at the lowest possible level (aka Level-0)? Note that Level-0 is the level where there is no aggregation. In other words, in Level-0, it is not possible to break down the fact measures further as there are no aggregate values. Looking at the Purchase Order E/R diagram, as shown earlier in Fig. 14.9, Order Date in Purchase Order is not a candidate key, meaning that it is possible that there is more than one Purchase Order on the same date. When there is more than one Purchase Order per day and the grouping is based on day, this means that

Purchase Order is an aggregated value; therefore, it is not Level-0. This means that the star schema in Fig. 14.12 is **not Level-0**. For the sake of level numbers, we need to move all of the three star schemas one level up because the star schema in Fig. 14.12 is not Level-0; therefore, they become **Level-3** (Fig. 14.10), **Level-2** (Fig. 14.11) and **Level-1** (Fig. 14.12).

Now we need to work out what a Level-0 star schema is. The main reason why the star schema in Fig. 14.12 is not Level-0 is because the same Order Date may have several Purchase Orders. Hence, we need to use Purchase Order as a dimension (which includes Order Date as an attribute). Therefore, the Purchase Order entity in the E/R diagram becomes a dimension in the star schema. Also notice that Ordering Method is already part of Purchase Order, as the Ordering Method is an attribute in the Purchase Order entity in the E/R diagram. Therefore, a new star schema should have a Purchase Order Dimension which basically combines the Order Date Dimension with the Ordering Method Dimension. In the Purchase Order Dimension, where OrderID is the identifier, there is only one OrderID for each Purchase Order, and therefore the number of Purchase Orders per Purchase Order is always one, and it is not aggregated. This new star schema, with the Purchase Order Dimension, as shown in Fig. 14.13, is lower than the Level-1 star schema (Fig. 14.12). Notice that when we change the granularity of a dimension to be more detailed, it is possible that two existing dimensions are captured in the new dimension (e.g. the new Purchase Order Dimension captures both Order Date Dimension and Ordering Method Dimension), so the new star schema has one less dimension.

So now we have four star schemas: **Level-3** (Fig. 14.10), **Level-2** (Fig. 14.11), **Level-1** (Fig. 14.12) and **Level-0** (Fig. 14.13). The main question is whether the star schema in Fig. 14.13 is really a Level-0? We need to check whether the fact measures are still aggregated values. Level-0 should not have aggregated values in the fact measures. However, note that the two dimensions (Customer Dimension and Purchase Order Dimension) are already their lowest level and it is not possible to break them down further. But having the lowest level dimensions does not guarantee that the fact measure will not be an aggregated value. To check whether the fact measures are aggregated or not, we need to go back to the E/R diagram.

Fact measures in the star schema always focus on transactions. In this case study, the transaction happens when a Purchase Order is raised or when a purchase is

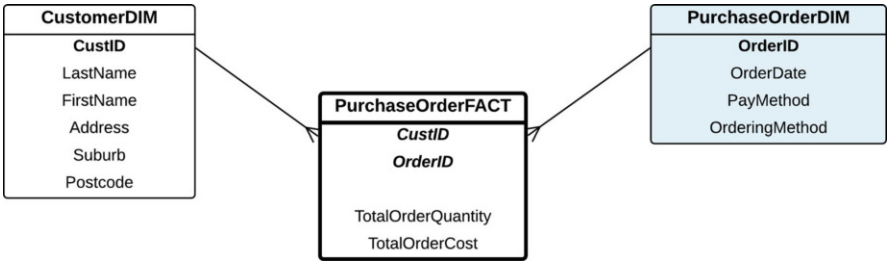

Fig. 14.13 Purchase Order star schema—with Purchase Order Dimension

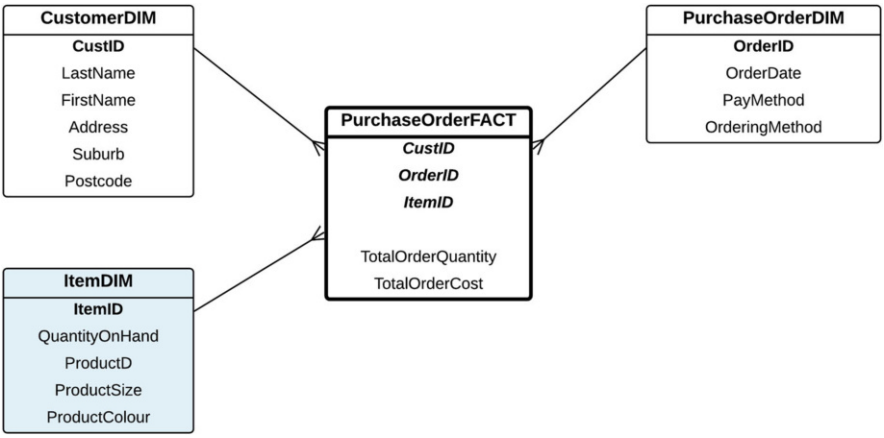

Fig. 14.14 Purchase Order star schema—with Item Dimension

ordered. The fact measures of this star schema are about calculating the quantity and the cost for each purchase order. Because one Purchase Order can have several Items, this means the lowest level of transaction is about each item in the Order Line—each individual record in the Order Line table. A transaction usually occurs in a *m*-*m* relationship, which in this case is the relationship between Purchase Order and Item entities, which is basically the Order Line entity.

The star schema in Fig. 14.13 is **not Level-0** because the fact measures do not capture both sides of the *m*-*m* relationship between Purchase Order and Item entities in the E/R diagram; rather, it captures one side only, which is from the Purchase Order side and not from the Item side. As a result, the fact measures are still aggregated as one Purchase Order may contain several Order Lines. To include the Item, we need an Item Dimension in the star schema. The new star schema with Item Dimension is shown in Fig. 14.14. This is a true **Level-0** star schema. The final line-up becomes:

- **Level-4** (Fig. 14.10)
- **Level-3** (Fig. 14.11)
- **Level-2** (Fig. 14.12)
- **Level-1** (Fig. 14.13)
- **Level-0** (Fig. 14.14)

As a conclusion, the Level-0 star schema should focus on the transaction level of the E/R diagram, which is denoted by the *m*-*m* relationship. If we need to include all possible dimensions in the star schema, we can add Product Dimension to the star schema, as shown in Fig. 14.15. Note that this star schema does not have a lower level of aggregation than previous star schema shown in Fig. 14.14 as both have the same level of detail.

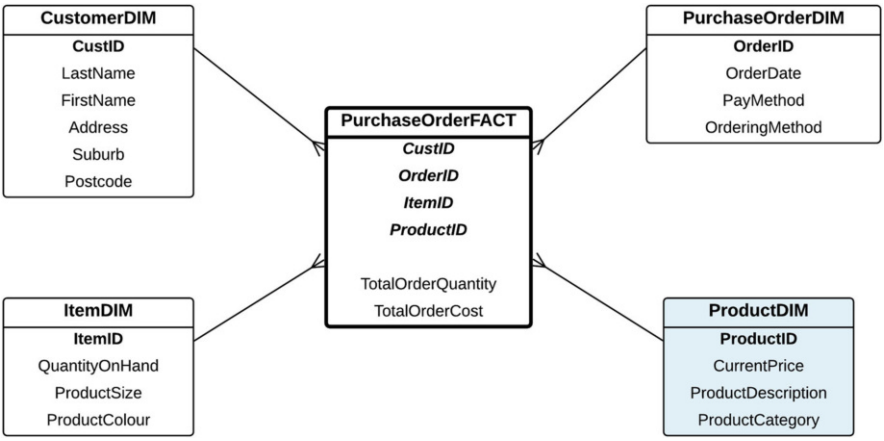

Fig. 14.15 Purchase Order star schema—with Product Dimension

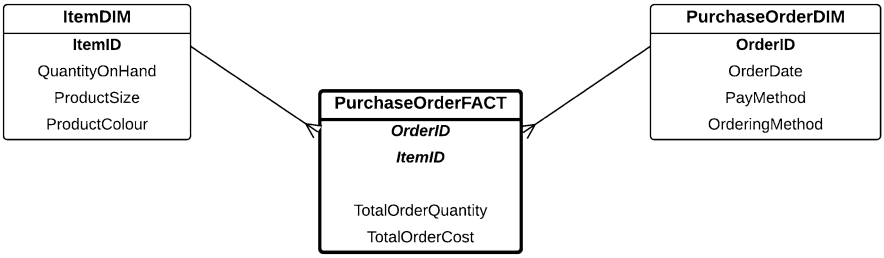

Fig. 14.16 Purchase Order star schema—the minimum requirement

The reason for this is that both star schemas already focus on the transaction level of the operational database, which is the record level of the Order Line table. One Item has one product, and if the star schema is already at the Item level, having Product Dimension will not lower the detail. This is also the same with the Customer because the star schema is at the Purchase Order level and one Purchase Order has only one Customer. Therefore, having a Customer Dimension does not lower the star schema either.

The bottom line is that the Level-0 star schema must have the Purchase Order Dimension and Item Dimension as its core. Therefore, the star schema with these two dimensions, as shown in Fig. 14.16, is Level-0. So there are four possible Level-0 star schemas, the minimal star schema with Purchase Order Dimension and Item Dimension (Fig. 14.16), the most complete star schema with Customer Dimension and Product Dimension (Fig. 14.15) or with either Customer Dimension (Fig. 14.14) or Product Dimension only.

14.4 Understanding the Relationship Between Transactions and Fact Measures

The central point of understanding the Level-0 star schema is by understanding the concept of transactions in the operational databases and where transactions are recorded in the E/R diagram.

In the Computer Lab Usage case study, the transaction is captured in the Lab Activities entity, which is a *m-m* relationship between Student and Computer entities. The fact measure of the star schema is Num of Logins, which is a `count` of records in the Lab Activities table grouped by the dimensions (Student, Degree, Login Date and Login Time).

Using the Purchase Order case study, the transaction is captured in the *m-m* relationship between Purchase Order entity and Item entity through Order Line entity. The fact measures of the star schema in the Purchase Order case study are Total Order Quantity and Total Order Cost (see the star schema in Fig. 14.14). Total Order Quantity is a `sum` of the Quantity attribute in the Order Line entity, whereas the Total Order Cost is also a `sum` of (Order Price × Quantity). Both attributes are also in the Order Line entity.

Suppose that in the Purchase Order case study, the fact measure is Num of Purchase Orders instead of Total Order Quantity and Total Order Cost. The star schema is shown in Fig. 14.17. The transaction focus which is the basis for this star schema is now different from the previous star schema in Fig. 14.14.

In the new star schema with Num of Purchase Orders as the fact measure, the focus of the transaction is no longer on the Order Line entity, but on the Purchase Order entity because the counting of the Num of Purchase Orders is in the Purchase Order entity and not in the Order Line entity. The implication is not seen in the star schema at this level (Fig. 14.17) as the star schema looks similar to the star schema in Fig. 14.9, as both have three dimensions, Location Dimension,

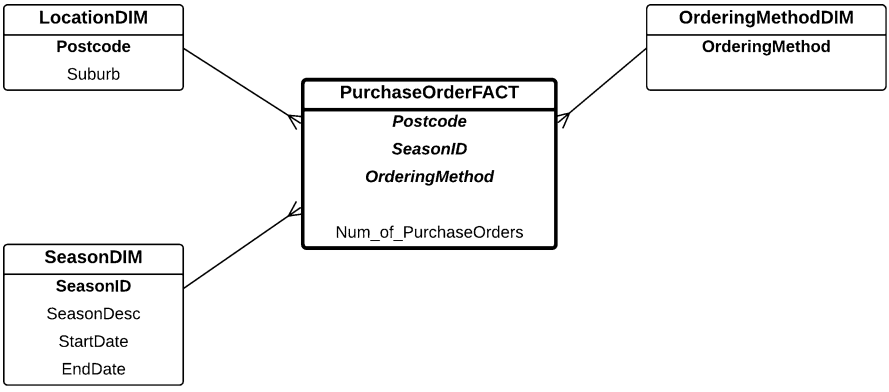

Fig. 14.17 Purchase Order star schema—with Number of Purchase Orders as the fact measure

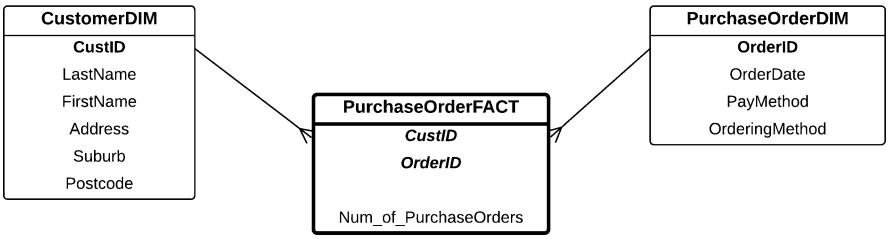

Fig. 14.18 Purchase Order star schema—Level-0

Season Dimension and Ordering Method Dimension. But when we lower down the level of aggregation, by changing the Season Dimension and the Ordering Method Dimension into the Purchase Order Dimension, and the Location Dimension into the Customer Dimension, as shown in Fig. 14.18, this star schema is now already in Level-0. In contrast, the same star schema with Customer and Purchase Order Dimensions, shown in Fig. 14.13, is not Level-0 as previously discussed. Where is the difference?

The difference is in the focus of the fact measure in the transactions. The focus of the previous star schema with Total Order Quantity and Total Order Cost is in the Order Line entity in the E/R diagram. Order Line entity is a *m*-*m* relationship between Purchase Order entity and Item entity. On the other hand, the star schema with the Num of Purchase Orders has a focus on the Purchase Order entity, not on the Order Line entity. So, the focus of the transaction is different. The granularity of the focus of the transaction is different. This is why the star schema in Fig. 14.18 is Level-0, whereas the star schema in Fig. 14.13 is not.

The difference is highlighted even more when we add a new dimension called Item Dimension to the star schema. In the star schema with Total Order Quantity and Total Order Cost, the star schema needs to have Item Dimension to make the star schema Level-0 (see the star schema in Fig. 14.16). In contrast, the star schema with the Num of Purchase Orders does not have to have an Item Dimension, because the star schema is already Level-0 (refer to the star schema in Fig. 14.18). But if we want to include an Item Dimension into this star schema, a bridge table is needed to connect to the Item Dimension because for each Purchase Order, there are many Items. The new star schema with Item Dimension is shown in Fig. 14.19. With the addition of the Item Dimension, the star schema is still Level-0.

Notice the main difference between this star schema with a bridge and the previous star schema where Item Dimension is required to make the star schema Level-0. This difference is due to the different focus of the fact measure in the transactions in the E/R diagram.

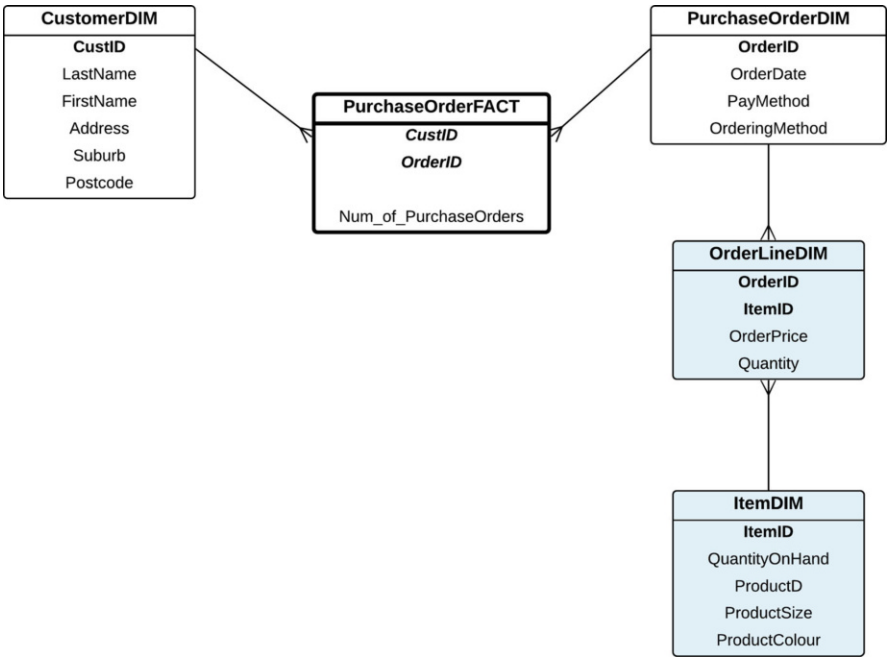

Fig. 14.19 Purchase Order star schema—Level-0 with a Bridge to Item Dimension

14.5 Levels of Aggregations, Hierarchy and Multi-Fact

Levels of aggregation are often discussed in the context of Dimension Hierarchy as well as Multi-Fact. In order to understand these, let's have a look at the Sales star schema, as shown in Fig. 14.20. This star schema is a simple Sales star schema with a hierarchy. The hierarchy is Branch, City and Country Dimensions. There are two other dimensions: Product and Time (Year). The Sales Fact has one fact measure, which is Total Sales.

The Sales Fact shows that Total Sales is actually based on the Branch granularity. The hierarchy is merely an implementation of a normalisation concept in the Relational Database Management Systems (RDBMS), to minimise repetition and to avoid anomalies. The hierarchy is not meant for the drilling down or rolling up of the fact measure.

The hierarchy shows the level of granularity. The hierarchy should start from the most detailed, which in this case is Branch, and go down to the more general, which is City and then Country. If these dimensions in the hierarchy are split into three star schemas, then there will be three Fact Tables. The three star schemas are shown in Fig. 14.21. The first star schema uses the Branch Dimension; therefore, Total Sales is based on the Branch granularity. To differentiate this from the other

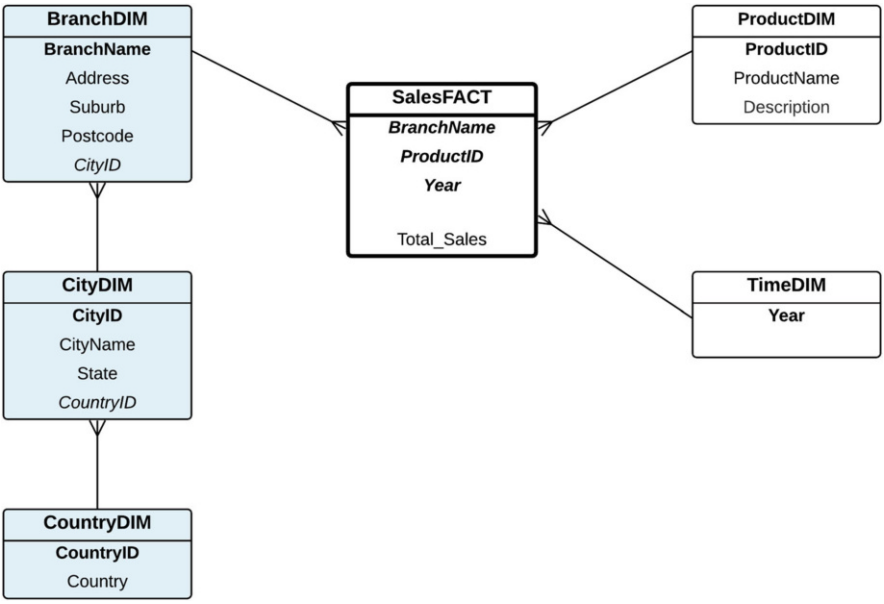

Fig. 14.20 Sales star schema

two star schemas, the Fact Table is called the Branch Sales Fact with Branch Total Sales as the fact measure.

The second star schema uses the City Dimension instead of the Branch. Therefore, the Total Sales fact measure is based on the City granularity. The Fact Table is renamed to City Sales Fact with City Total Sales as the fact measure. Note that the City star schema is more general than the Branch star schema.

The third star schema uses the Country Dimension, which is the most general. The Fact Table becomes the Country Sales Fact with Country Total Sales as the fact measure.

These three star schemas (i.e. Branch star schema, City star schema and Country star schema) focus on the same fact measure, which is Total Sales. The difference is that they have a different level of granularity or level of aggregation. The Branch star schema is the most detailed, whereas the Country star schema is the most general. Hence, there are three levels in this data warehousing architecture, Level-1 for the Branch star schema, Level-2 for the City star schema and Level-3 for the Country star schema.

These three levels can be joined into a multi-fact star schema through the hierarchy, as shown in Fig. 14.22. The hierarchy is maintained among the dimensions from a different level of granularity (e.g. Branch, City and Country Dimensions), and the other dimensions (e.g. Product and Time) are shared by the three Fact Tables.

So, in short, there is a correlation between the concept of data warehousing granularity (or level of granularity), dimension hierarchy and multi-fact. The

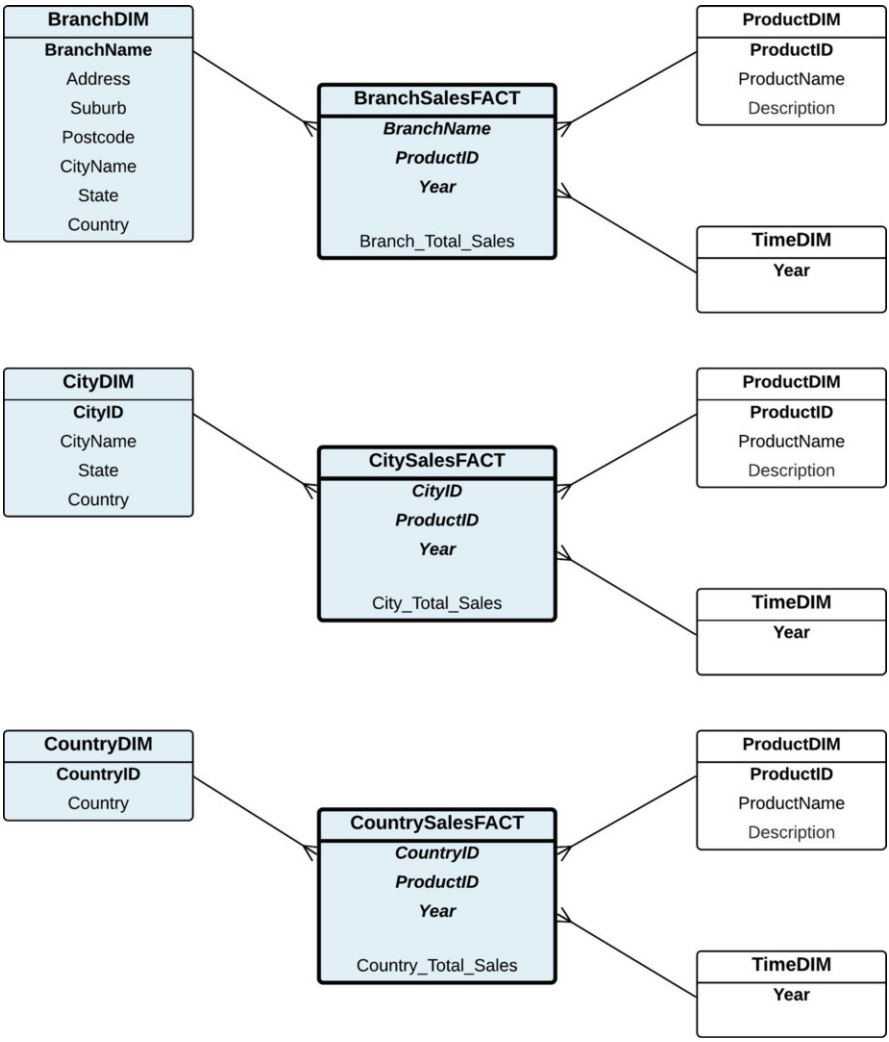

Fig. 14.21 Three star schemas (Branch star schema, City star schema, Country star schema)

different levels of star schema depict the multi-fact and the dimensions in different levels of the star schema can form a hierarchy. As a matter of fact, hierarchy is often used in conjunction with the level of granularity in the data warehousing architecture.

The star schema will become more complex when multiple hierarchies exist. Suppose the Time Dimension has a hierarchy, such as Quarter and then Year, as shown in the star schema in Fig. 14.23.

Using the hierarchy, multi-fact and level of granularity concepts described above, we can have a star schema consisting of two facts, one for Quarterly Sales and the

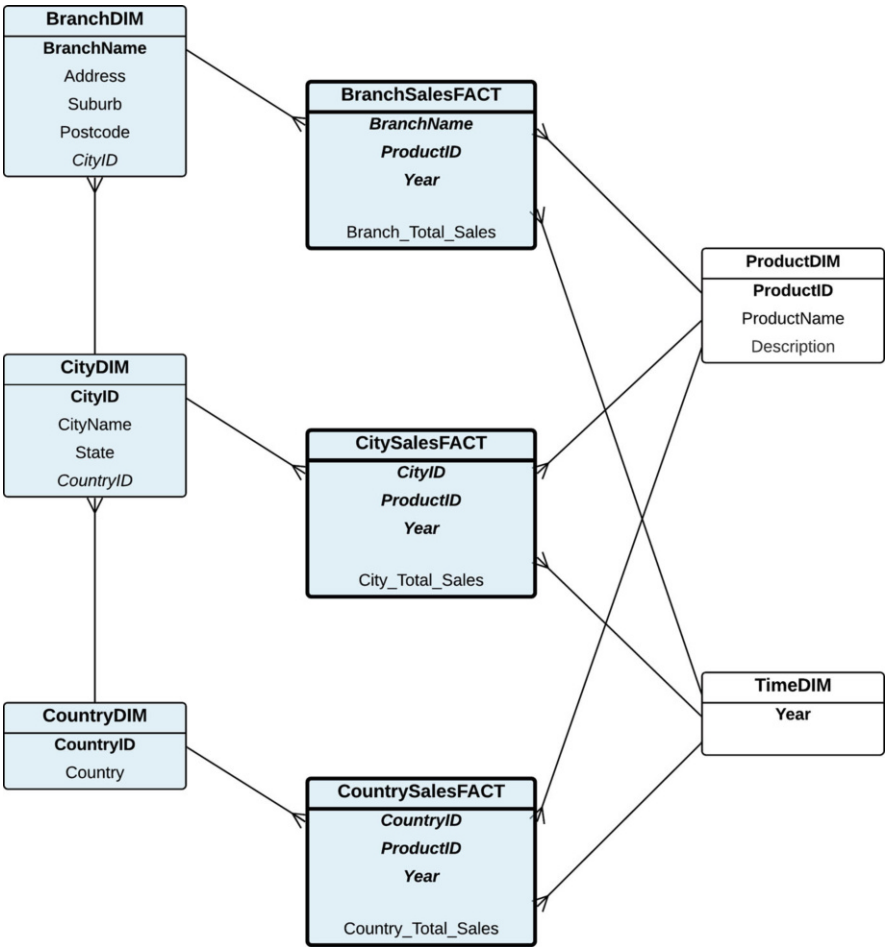

Fig. 14.22 A Multi-Fact star schema

other for Yearly Sales. The hierarchy between Quarter and Year is maintained in the following star schema (refer to Fig. 14.24). So, this is another star schema with Time hierarchy.

If we combine the previous star schema (with the Branch hierarchy) and this star schema (with the Time hierarchy), we will end up with six Fact Tables, as shown in the following star schema (Fig. 14.25) (for simplicity, the Product Dimension is not shown in the picture). This multi-fact, multi-hierarchy star schema is often called a *fact constellation*.

Using this combined star schema, drilling down and rolling up the Total Sales can be done by querying the right level of the star schema; as there are six options to choose from, namely Quarterly and Yearly Branch, City and Country Facts, each with a different granularity of Total Sales.

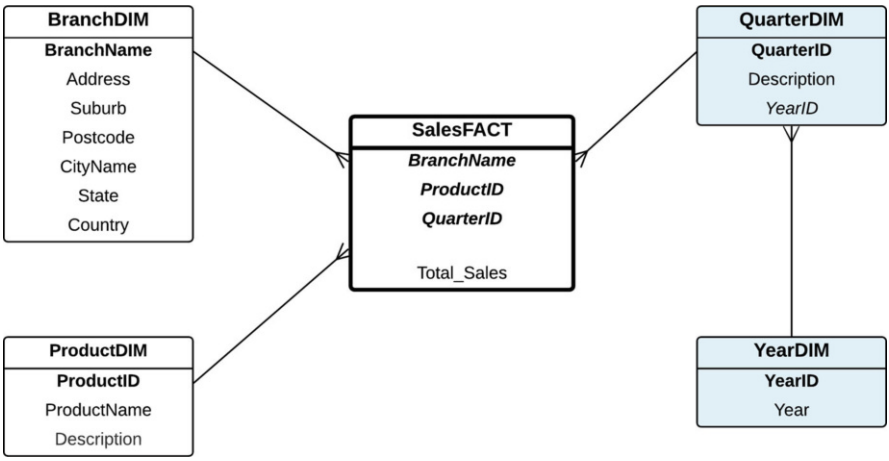

Fig. 14.23 Another hierarchy (Quarter, Year)

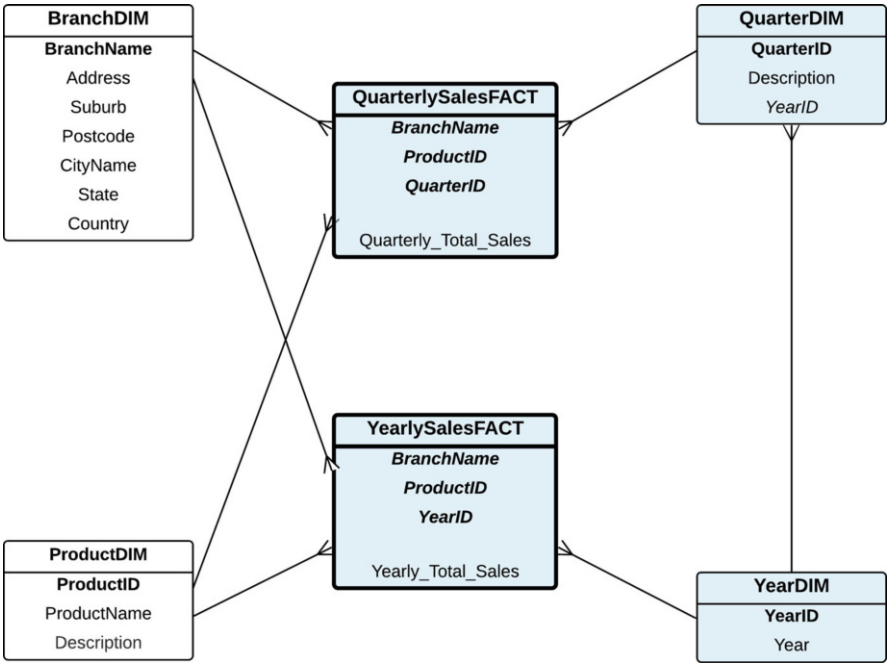

Fig. 14.24 A multi-fact star schema with Quarter and Year hierarchy

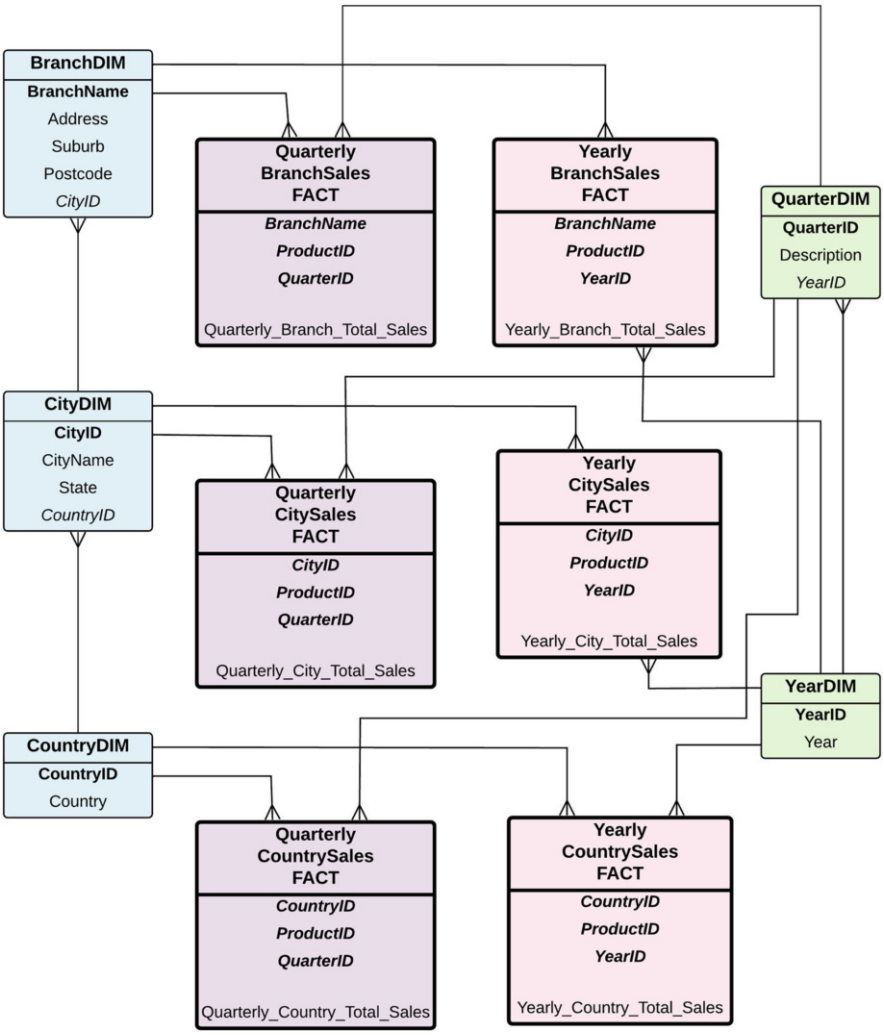

Fig. 14.25 A fact constellation

The numbering system in the data warehousing architecture is not static or fixed. What we know is that the Country Sales star schema is more general (has a higher level of aggregation) than City Sales and Branch Sales. The Yearly Sales star schema is more general and hence has a higher level of aggregation than the Quarterly Sales star schema. The following six levels of aggregation for the above star schema are not correct:

- Level-6: Yearly Country Sales
- Level-5: Quarterly Country Sales

- Level-4: Yearly City Sales
- Level-3: Quarterly City Sales
- Level-2: Yearly Branch Sales
- Level-1: Quarterly Branch Sales

Take the Quarterly Country Sales and the Yearly City Sales as an example. It is incorrect to say that the Quarterly Country Sales is more general than the Yearly City Sales because they are not formed by one hierarchy. This is the reason why the above level numbering is incorrect.

When comparing two star schemas, we only need to know whether one star schema is more general than the other or whether the two star schemas cannot be compared. In the above example, Quarterly Country Sales and Yearly City Sales are not comparable, whereas Yearly Country Sales is definitely more general than Quarterly Country Sales. Therefore, we should have two partitions or groups of levels of granularity, one based on time and the other based on location.

The relationship between the six star schemas is actually depicted in Fig. 14.26. It can be seen that there are four different "paths" of the level of aggregation. One path is from Country Yearly being the most general to City Yearly less general, then to Branch Yearly and finally to Branch Quarterly being the most detailed.

The second path of the level of aggregation is also from Country Yearly being the most general to Country Quarterly less general, then to City Quarterly and to Branch Quarterly which is the most detailed. The third path of the level of aggregation is Country Yearly, City Yearly, City Quarterly and Branch Quarterly, and the fourth path is Country Yearly, Country Quarterly, Branch Yearly and Branch Quarterly.

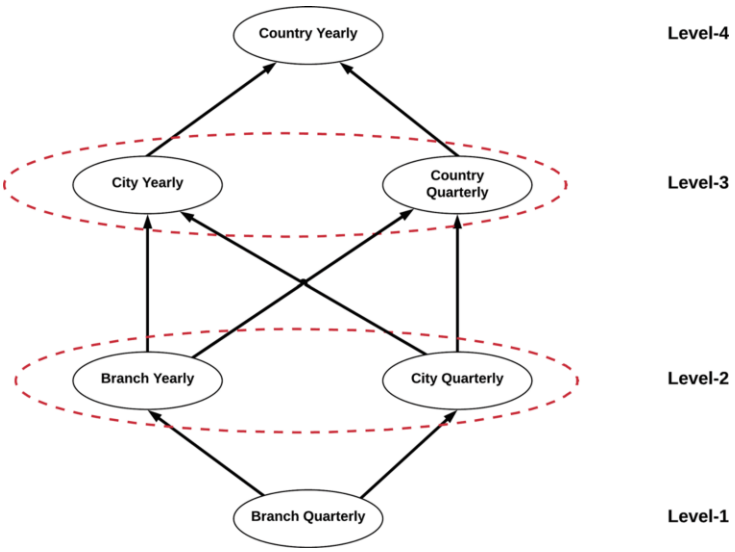

Fig. 14.26 A simpler fact constellation

All of the four paths have the same most general star schema, which is Country Yearly, and the most detailed star schema, which is Branch Quarterly. Each of the four paths of the level of aggregation guarantees that the level of aggregation within each path is accurate. For example, in the first path, City Yearly is more general than Branch Yearly.

It is clear from these four paths of the level of aggregation that Country Yearly being the most general is at the top level of the data warehousing architecture (e.g. Level-4 in this example) and the next level down (e.g. Level-3) is City Yearly and Country Quarterly. Level-2 contains Branch Yearly and City Quarterly. Level-1 which is the most detailed is Branch Quarterly.

14.6 Summary

This chapter introduces the concept of levels of aggregation. A series of star schemas actually form a hierarchy of levels, whereby Level-0 aggregation is the star schema without aggregation in the fact measures. The higher the level of aggregation, the more aggregation in the fact measure. This hierarchy of the level of aggregation is called *data warehousing granularity*.

It is important to understand the concept of data warehousing architecture because a data warehouse is built primarily for drilling down to find some interesting data for business decisions. Some users (such as top managers) may focus on high levels of aggregation as the data is already highly aggregated, whereas other users might want to dig further down into the data; hence, more detailed data is needed—a data warehouse with a lower level of aggregation.

It is common when we design a data warehouse that we start from a high level of aggregation where fact measures contain aggregated values. Lowering the level of aggregation can be done by changing the granularity of the dimension or by simply adding new dimensions. Designing a data warehouse from the lowest level of aggregation (Level-0) is often counterproductive because the operational database itself is the lowest level; hence, creating a Level-0 data warehouse can be seen as unnecessary and duplicating the operational database.

Level-0 data warehouses also often include facts without fact measures because there is no need for a numerical value to represent the fact measures. This is permitted because Level-0 contains no aggregation anyway. However, when it moves to Level-1, the fact measure is necessary to capture the aggregated value of the fact.

Determining whether a star schema is in Level-0 or not can be tricky. Not having an aggregated fact measure does not always mean that the star schema is in Level-0. Hence, it is important to understand the concept of the transaction recorded in the E/R diagram of the operational database.

Data warehousing granularity that contains star schemas of various levels of aggregation can be seen as multi-fact star schemas formed in a global hierarchy, which is also known as *fact constellation*. Hence, having a global overview of all

star schemas in the fact constellation is important, especially in data investigation during business decision-making.

14.7 Exercises

14.1 In the Computer Lab Usage case study explained earlier in this chapter, the E/R diagram of the operational database is shown in Fig. 14.1. It has four tables: Computer, Lab Activities, Student and Degree. The three levels of star schemas are shown in Fig. 14.2 for Level-2, Fig. 14.4 for Level-1 and Fig. 14.5 for Level-0.

Tasks: Create these three star schemas, including their Fact Tables and all of the dimension tables, using SQL statements.

14.2 In the Purchase Order case study in this chapter, the E/R diagram of the operational database is shown in Fig. 14.9. This Purchase Order operational database has five tables: Customer, Purchase Order, Order Line, Item and Product. Five levels of star schema have been designed. The Level-4 star schema (Fig. 14.10) consists of the Location Dimension, Season Dimension and Ordering Method Dimension. The Level-3 star schema (Fig. 14.11) replaces the Season Dimension with the Order Date Dimension. The Level-2 star schema (Fig. 14.12) replaces the Location Dimension with the Customer Dimension, as the location is part of each customer. The Level-1 star schema (Fig. 14.13) combines the Order Date Dimension and Ordering Method Dimension to become the Purchase Order Dimension, because the order date and ordering method attributes are both in the Purchase Order entity. Finally, the Item Dimension is added to Level-0 (Fig. 14.14).

Tasks: Create these five star schemas, including their Fact Tables and all of the dimension tables, using SQL statements. As there are two more Level-0 star schemas (see Figs. 14.15 and 14.16), create these two Level-0 star schemas as well.

14.3 Note that the Level-0 star schemas represent the transactions in the operational database. A transaction is normally denoted by a *m*-*m* cardinality relationship. In the Computer Lab Usage case study, as shown in the E/R diagram in Fig. 14.1, the transactions are represented by the *m*-*m* relationship between Student and Computer entities, through the Lab Activity entity. In the Product Order case study (the E/R diagram is shown in Fig. 14.9), the transaction is a *m*-*m* relationship between Purchase Order and Item entities, through the Order Line entity. Both Level-0 star schemas of these two case studies focus on these particular entities and entity relationships.

In the Purchase Order case study, the Level-0 star schema includes two main dimensions: Purchase Order and Item Dimensions (refer to the Level-0 star schema in Fig. 14.16), because these two dimensions are basically the two entities in the E/R diagram which are the basis of the purchase order transactions. It may include Customer and Product Dimensions, but these are optional.

On the other hand, in the Computer Lab Usage case study, the Level-0 star schema includes Student Dimension, Degree Dimension, Login Date Dimension

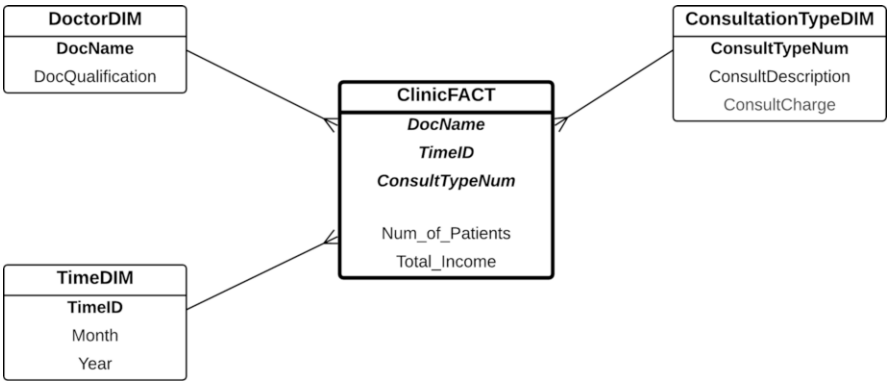

Fig. 14.27 Clinic Star Schema Version 1

and Login Time Dimension. The transaction in the Computer Lab Usage case study is a *m*-*m* relationship between Student and Computer entities. The Computer Dimension is not used in the Level-0 star schema (see Fig. 14.5). Why is Computer Dimension not used in this Level-0 star schema, and yet the star schema in Fig. 14.5 is Level-0? In contrast, the Item Dimension must be used in the Level-0 star schema in the Purchase Order case study. What is the difference between these two case studies to result in the Level-0 star schema solutions being different?

14.4 A medical clinic employs four general practitioners (doctors): Dr Adele, Dr Ben, Dr Kate and Dr Chris. Some of these doctors do not practise every day. For example, Dr Adele practises on Mondays and Wednesdays only, whereas Dr Ben is there only on Thursdays. When a patient comes to the clinic and has a consultation with a doctor, the patient pays a certain consultation fee, depending on the type of consultation the patient had. For example, a general consultation fee (code 113) is $37.50. Because of the nature of medical practice, there are more than 100 different codes for different types of consultations.

The clinic maintains an operational database that records every payment for each consultation by every doctor. A data warehouse is needed for reporting purposes. There are two versions of star schema for this clinic. Figures 14.27 and 14.28 show the two star schemas.

Questions: Which schema (Star Schema Version 1 or Star Schema Version 2) has a higher level of aggregation, or are they of the same level of aggregation? State your answer, and explain your reasons as well.

14.5 A tourist bus company hires buses to groups of people for tourism purposes. Their customers include primary and secondary schools, various organisations, groups of private foreign tourists, etc. The company has many buses of various sizes (e.g. 45-seater buses, 24-seater buses or minibuses and 12-seater cars). When clients hire a bus, the driver is included in the hiring package as well.

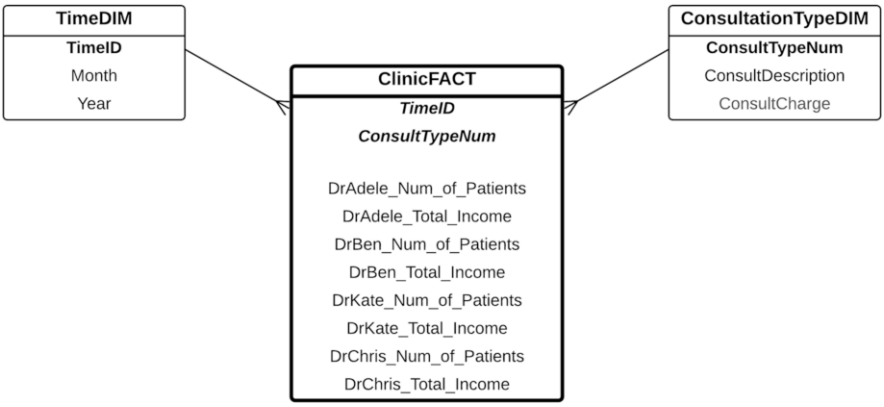

Fig. 14.28 Clinic Star Schema Version 2

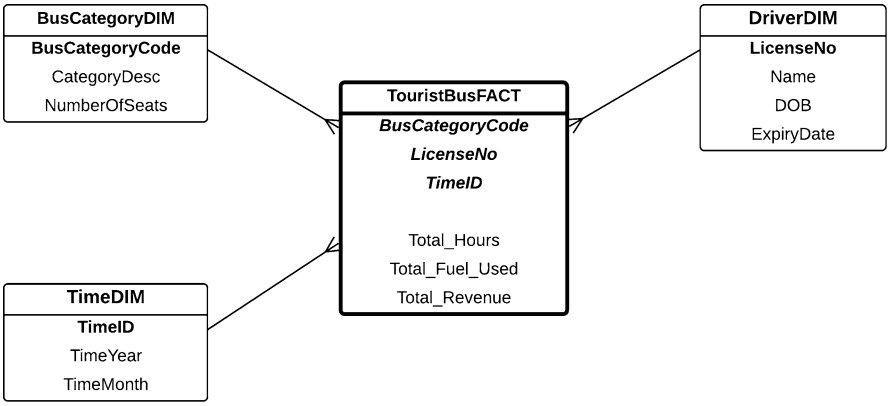

Fig. 14.29 Tourist Bus Star Schema Version 1

Two star schemas have been developed. Figures 14.29 and 14.30 show the two star schemas.

Questions: Which schema (Star Schema Version 1 or Star Schema Version 2) has a higher level of aggregation, or are they of the same level of aggregation? State your answer and explain your reasons as well.

14.6 There is a toll way (or toll road) in a metropolitan city (such as CityLink or EastLink in Melbourne, or any similar toll roads in other major cities in the world). This toll road has a number of toll points where motorists are electronically charged for travelling on the toll road. Every time a motorist passes through this tool point, the registration number of the vehicle, vehicle type (e.g. car, bus, truck, etc.), amount paid and time are recorded in the operational database.

A data warehouse needs to be built to analyse the revenue from the toll payments. The management would like to drill down into this revenue based on the toll

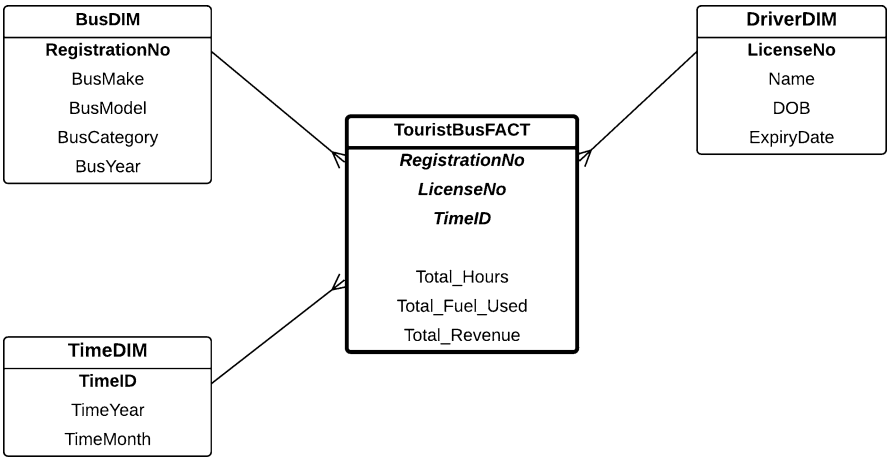

Fig. 14.30 Tourist Bus Star Schema Version 2

point (there are a number of toll points along the toll way), day of the week (e.g. weekdays, weekends) and time period of a day (e.g. peak hours, non-peak hours, late at night).

You are required to draw three levels of star schemas showing three different levels of aggregation for the above data warehouse. You also need to explain each of the three star schemas by contrasting the level of aggregation. The Level-0 star schema contains the most detailed data, whereas the Level-2 star schema is the highly aggregated (e.g. containing highly aggregated data).

Tasks

- Draw a Level-2 star schema and explain why it is a Level-2 schema.
- Draw a Level-1 star schema and explain why it is a Level-1 schema. You may want to add a new dimension called vehicle (e.g. cars, trucks, buses, etc.). You also need to explain the difference between Level-1 and Level-2 schemas.
- Draw a Level-0 star schema and explain why it is a Level-0 schema. You also need to explain the difference between Level-1 and Level-0 schemas.

14.8 Further Readings

The concept of granularity is crucial in understanding in-depth star schema design, without which an incorrect perception of the design will come to surface quite easily. This book is the only book that discusses the concept of granularity in data warehousing, including star schema design using higher level and lower level of aggregation. More general star schema design and modelling has been covered in various data warehousing books, such as [1–10] and [11]. Data modelling

books, such as [12–16] and [17], would also give readers a broader perspective on differences between various levels of aggregation in data modelling.

The concept of data granularity has been used in various applications. For example, in the digital image processing area, images or pictures can be stored in multiple pixel sizes [18]. The sharper the picture, the denser the number of pixels. For the same images, it can be stored in different levels of granularity. The concept of granularity is also used in digital video processing and audio processing, where video and audio files can be stored in various levels of details [19, 20].

Data produced by sensors in the form of data streams, which is raw data, may be processed into lower granularity, by aggregating data into coarser-grained data [21–23].

Granularity, particularly time granularity, has been discussed in the context of databases and data mining [24], as well as in distributed database transactions and real-time database logical design [25, 26]. Granularity in data modelling is discussed in [27].

References

1. C. Adamson, *Star Schema The Complete Reference* (McGraw-Hill Osborne Media, 2010)
2. R. Laberge, *The Data Warehouse Mentor: Practical Data Warehouse and Business Intelligence Insights* (McGraw-Hill, New York, 2011)
3. M. Golfarelli, S. Rizzi, *Data Warehouse Design: Modern Principles and Methodologies* (McGraw-Hill, New York, 2009)
4. C. Adamson, *Mastering Data Warehouse Aggregates: Solutions for Star Schema Performance* (Wiley, London, 2012)
5. P. Ponniah, *Data Warehousing Fundamentals for IT Professionals* (Wiley, London, 2011)
6. R. Kimball, M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (Wiley, London, 2013)
7. R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, B. Becker, *The Data Warehouse Lifecycle Toolkit* (Wiley, London, 2011)
8. W.H. Inmon, *Building the Data Warehouse*. ITPro Collection (Wiley, London, 2005)
9. M. Jarke, *Fundamentals of Data Warehouses*, 2nd edn. (Springer, Berlin, 2003)
10. E. Malinowski, E. Zimányi, *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications*. Data-Centric Systems and Applications (Springer, Berlin, 2008)
11. A. Vaisman, E. Zimányi, *Data Warehouse Systems: Design and Implementation*. Data-Centric Systems and Applications (Springer, Berlin, 2014)
12. T.A. Halpin, T. Morgan, *Information Modeling and Relational Databases*, 2nd edn. (Morgan Kaufmann, Los Altos, 2008)
13. J.L. Harrington, *Relational Database Design Clearly Explained*. Clearly Explained Series (Morgan Kaufmann, Los Altos, 2002)
14. M.J. Hernandez, *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*. For Mere Mortals (Pearson Education, 2013)
15. N.S. Umanath, R.W. Scamell, *Data Modeling and Database Design* (Cengage Learning, 2014)
16. T.J. Teorey, S.S. Lightstone, T. Nadeau, H.V. Jagadish, *Database Modeling and Design: Logical Design*. The Morgan Kaufmann Series in Data Management Systems (Elsevier, Amsterdam, 2011)
17. G. Simsion, G. Witt, *Data Modeling Essentials*. The Morgan Kaufmann Series in Data Management Systems (Elsevier, Amsterdam, 2004)

18. B. Jähne, *Digital Image Processing*. Engineering Online Library (Springer, Berlin, 2005)
19. M. Parker, S. Dhanani, *Digital Video Processing for Engineers: A Foundation for Embedded Systems Design* (Elsevier, Amsterdam, 2012)
20. M.G. Christensen, *Introduction to Audio Processing* (Springer, Berlin, 2019)
21. P. Sangat, M. Indrawan-Santiago, D. Taniar, Sensor data management in the cloud: data storage, data ingestion, and data retrieval. Concurr. Comput. Pract. Exp. **30**(1), e4354 (2018)
22. L. Golab, M.T. Ozsu, *Data Stream Management*. Synthesis Lectures on Data Management (Morgan & Claypool Publishers, 2010)
23. H.C.M. Andrade, B. Gedik, D.S. Turaga, *Fundamentals of Stream Processing: Application Design, Systems, and Analytics* (Cambridge University Press, Cambridge, 2014)
24. C. Bettini, S. Jajodia, S. Wang, *Time Granularities in Databases, Data Mining, and Temporal Reasoning* (Springer, Berlin, 2000)
25. M.T. Özsu, P. Valduriez, *Principles of Distributed Database Systems*. (Springer, New York, 2011)
26. T.J. Teorey, *Database Modeling and Design*. The Morgan Kaufmann Series in Data Management Systems (Elsevier, Cambridge, 1999)
27. G. Powell, *Database Modeling Step by Step* (CRC Press, Boca Raton, 2020)

Chapter 15
Designing Lowest-Level Star Schemas

So far, this book has explained that designing a star schema starts from a higher level of aggregation. The main reason for this is that Level-0 star schemas are often very similar, if not identical, to the operational database represented by the E/R diagram. Therefore, from a pedagogical point of view, it is often difficult to understand why a data warehouse is needed if we can get everything from the operational database.

A higher-level star schema represents the precomputed values of fact measures. With these, data retrieval from the star schema may well suit decision-making, which is often based on a higher level of aggregation as it depicts a global picture of the business operation. Therefore, a Level-0 star schema is often seen as unnecessary. Additionally, going up or down the level of aggregation can be done quite easily by changing the granularity of the dimensions and/or adding/removing dimensions from the star schema.

However, in some cases, designing a Level-0 star schema is needed. This chapter focuses on when and how to design a Level-0 star schema. The first case study is the median house price and the second case study is exploring the use of some of the statistical functions that require the star schemas to be at Level-0.

15.1 Median House Price

An operational database which stores the sales records of properties (e.g. houses, apartments, units) already exists. The E/R diagram is shown in Fig. 15.1. The database is very simple, as it contains only four tables. The main table is the Property table which keeps the details of each property, with Property No as its Primary Key. The transaction table is the Sold table, which stores the sold price and sold date. For simplicity of design, the relationship between Property and Sold is 1-1. There are two other minor tables: Sold Method (e.g. auction, private sale) and Property Type (e.g. house, apartment, unit) tables.

D. Taniar, W. Rahayu, *Data Warehousing and Analytics*, Data-Centric Systems and Applications, https://doi.org/10.1007/978-3-030-81979-8_15

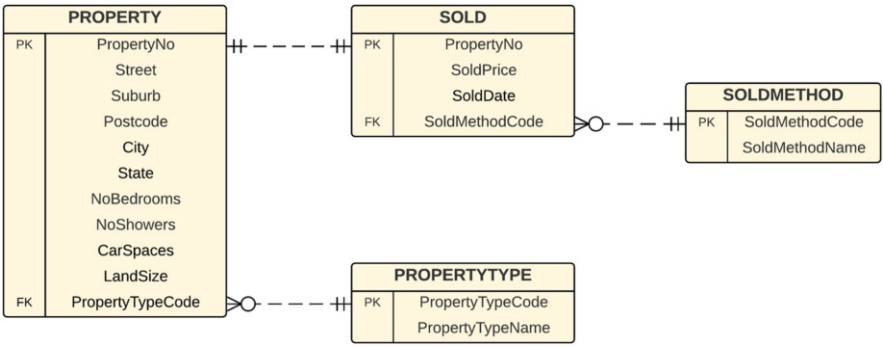

Fig. 15.1 Sold Properties E/R diagram

One of the main indicators in the housing market is *median price*, which is the "middle" price. If the houses are sorted based on the Sold Price, the Sold Price of the house in the middle of this list is the median price. This means we need to keep each individual Sold Price as it is not possible to aggregate the houses, either based on Suburb, or Month, or any other parameters. Each individual house is needed in the data warehouse, because without this, the median price cannot be determined. Consequently, the star schema must be at the lowest level, which is Level-0. Assume we would like to have four dimensions, such as Suburb, Month, Property Type and Sold Method, and the query to the data warehouse is something like: "what is the median price of houses sold in 2020 in a certain suburb?" This kind of question is often asked in the property market which may determine the performance of properties in a certain place.

If we examine, particularly, the Suburb and Month Dimensions, these dimensions are not in the highest granularity; rather, they are already coarse-grained because month is more general than dates and suburb is more general than streets, for example. Because determining the median price needs each individual property, we need to go to the lowest level of these two dimensions. Instead of Suburb, we need to go down to the actual address of the property (note that just the street is not detailed enough because there is a possibility that there is more than one house on the same street in this database). Also, for the Month, we need to go down to the actual sold date, as this is the most detailed and cannot be broken down further.

Therefore, the star schema's four dimensions are Address, Sold Date, Property Type and Sold Method Dimensions. Note that the Property Type and Sold Method Dimensions are already the most detailed or at the highest granularity. The fact measure is Sold Price. Note that the fact measure is not an aggregated value, it is an individual value, that is, a Level-0 fact measure (e.g. no aggregation). The star schema is shown in Fig. 15.2. Note that the Address Dimension uses Property No as the identifier as it is unique to each property.

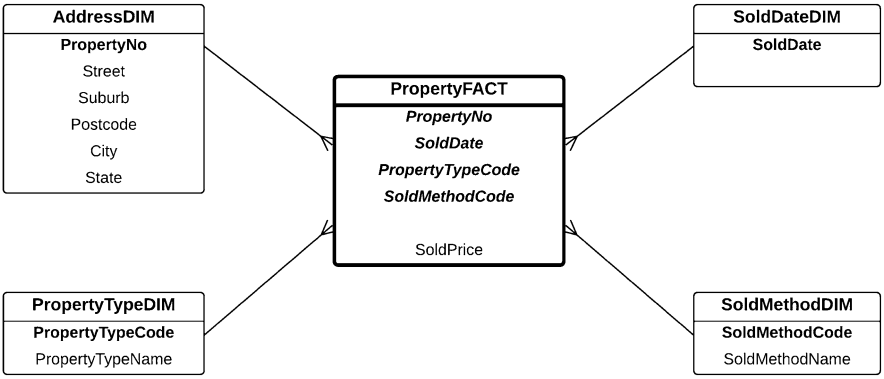

Fig. 15.2 Level-0 star schema

Property Type and Sold Method Dimensions are identical to the respective tables/entities in the E/R diagram. The Address Dimension is also taken from the Property entity, with only selected attributes, and the Sold Date Dimension contains the Sold Date attribute taken from the Sold entity. The SQL commands to create the Level-0 star schema are as follows:

```
create table AddressDim as
select
  PropertyNo,
  Street,
  Suburb,
  Postcode,
  City,
  State
from Property;

create table SoldDateDim as
select distinct SoldDate
from Sold;

create table PropertyTypeDim as
select * from PropertyType;

create table SoldMethodDim as
select * from SoldMethod;

create table PropertyFact as
select
  P.PropertyNo,
  S.SoldDate,
  P.PropertyTypeCode,
  S.SoldMethodCode,
  S.SoldPrice
from Property P, Sold S
where P.PropertyNo = S.PropertyNo;
```

The Property Fact Table only needs data from the Property and Sold tables, and there is no aggregate function used in the SQL.

If we examine this Level-0 star schema further, it looks similar to the E/R diagram. The only difference is perhaps the view; the star schema is centred around the fact, so the view is clear, whereas the E/R diagram shows the binary relationships between two entities. So, the focus is slightly different.

With this Level-0 star schema, to answer the query "retrieve the median price of a house sold in 2020 in the Kew suburb", the SQL is as follows. In this example, the `median` aggregate function is used. But not all DBMS support this function. As an exercise at the end of this chapter, you will be asked to write an SQL query without using this function.

```
select median(T.SoldPrice)
from
  (select
     A.Suburb,
     P.PropertyTypeName,
     to_char(F.SoldDate, 'YYYY'),
     F.SoldPrice
  from PropertyFact F, AddressDim A, PropertyTypeDim P
  where F.PropertyNo = A.PropertyNo
  and F.PropertyTypeCode = P.PropertyTypeCode
  and A.Suburb = 'Kew'
  and P.PropertyTypeName = 'House'
  and to_char(F.SoldDate, 'YYYY') = '2020'
  order by F.SoldPrice) T;
```

If we would like to create a higher-level star schema from this Level-0 star schema, we could change the granularity of the two dimensions, Address and Sold Date Dimensions, to more general dimensions. So, instead of the Address Dimension, we change it to the Suburb Dimension, and instead of the Sold Date Dimension, we change it to the Sold Month Dimension. By changing these two dimensions, the star schema becomes more general. However, the Sold Price fact measure needs to change as well because when the properties are grouped together, the individual Sold Price disappears.

In the new star schema with a high level of aggregation, the individual Sold Price needs to be aggregated. So, instead of answering the median query, the query for the higher level of star schema could be the mean query (or the average query). In order to answer the mean query, the star schema must have the Total Sold Price and Number of Properties; these two become the new fact measure of the higher-level star schema, as shown in Fig. 15.3. Note that to answer the mean or the average query, the star schema does not need to be in Level-0.

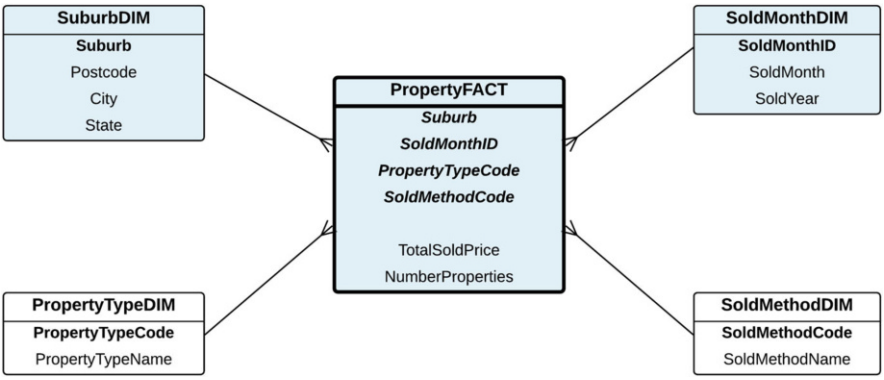

Fig. 15.3 Higher level of aggregation star schema

The SQL commands to create this new star schema are as follows. Note that we do not need to recreate the Property Type Dimension and Sold Method Dimension tables. They can be reused from the Level-0 star schema.

```
create table SuburbDim as
select distinct
  Suburb,
  Postcode,
  City,
  State
from Property;

create table SoldMonthDim as
select distinct
  to_char(SoldDate, 'MMYYYY') as SoldMonthID,
  to_char(SoldDate, 'Mon') as SoldMonth,
  to_char(SoldDate, 'YYYY') as SoldYear
from Sold;

create table PropertyFactLevel1 as
select
  P.Suburb,
  to_char(S.SoldDate, 'MMYYYY') as SoldMonthID,
  P.PropertyTypeCode,
  S.SoldMethodCode,
  sum(S.SoldPrice) as TotalSoldPrice,
  count(*) as NumberProperties
from Property P, Sold S
where P.PropertyNo = S.PropertyNo
group by
  P.Suburb,
  to_char(S.SoldDate, 'MMYYYY'),
  P.PropertyTypeCode,
  S.SoldMethodCode;
```

15.2 Other Statistical Functions

There are other statistical functions that need data at the lowest level (e.g. Level-0 star schema). This section will discuss a case study from the *weather* data. The weather data comes from a number of weather stations in Melbourne, Australia, as shown in Fig. 15.4. Each weather station has StationID, Station Name, Latitude, Longitude and Height. The data itself contains various measurements, such as temperature, humidity, wind, pressure and rainfall, all with a timestamp. Weather data from various stations in Melbourne can be unioned to obtain a large weather data file.

From this weather data file, we can create a Level-0 star schema, which appears similar to the schema in Fig. 15.5. The source weather data file is already the Fact Table of the data warehouse. However, for simplicity, only a few fact measures are selected, namely, Temperature, Humidity, Wind Speed, Wind Gust, Pressure and Rainfall.

The star schema has three chosen dimensions: Weather Station, Time and Wind Direction Dimensions. Weather Station and Time Dimensions are quite obvious because the weather data measurements are recorded based on the time at a particular weather station. However, the Wind Direction Dimension is not that obvious. If we look at the source weather data, there is important information about wind, particularly wind speed, as well as wind direction. Wind Speed is a fact measure because it is a numerical value, which later can be aggregated in a higher level of star schema. However, Wind Direction is not; hence, it is not suitable to become a fact measure of a data warehouse. Additionally, it is not possible to aggregate Wind Direction in a higher level of data warehouse. A solution for this is to make Wind Direction a dimension. But this violates the concept of the *two-*

Australian Government
Bureau of Meteorology

Latest Weather Observations for Melbourne Airport

IDV60901

Issued at 7:41 pm EDT Thursday 14 November 2019 (issued every 10 minutes, with the page automatically refreshed every 10 minutes)

Station Details ID: 086282 Name: MELBOURNE AIRPORT Lat: -37.67 Lon: 144.83 Height: 113.4 m

Data from the previous 72 hours. | See also: Recent months at Melbourne Airport

Date/Time EDT	Temp °C	App Temp °C	Dew Point °C	Rel Hum %	Delta-T °C	Wind					Press QNH hPa	Press MSL hPa	Rain since 9am mm
						Dir	Spd km/h	Gust km/h	Spd kts	Gust kts			
14/07:30pm	19.1	15.2	8.6	50	5.5	SW	19	28	10	15	1011.8	1011.6	0.0
14/07:00pm	20.3	15.8	8.5	46	6.2	WSW	22	30	12	16	1011.8	1011.6	0.0
14/06:30pm	20.5	15.4	7.8	44	6.6	SW	24	33	13	18	1011.9	1011.7	0.0
14/06:00pm	20.6	15.8	7.4	42	6.8	W	22	30	12	16	1011.9	1011.7	0.0
14/05:30pm	21.0	16.8	7.5	41	7.0	WSW	19	22	10	12	1012.0	1011.8	0.0
14/05:00pm	21.9	17.7	7.5	39	7.5	WNW	19	32	10	17	1012.0	1011.8	0.0
14/04:30pm	21.3	16.5	7.2	40	7.3	WNW	22	37	12	20	1012.3	1012.1	0.0
14/04:00pm	21.3	17.5	7.4	40	7.2	WNW	17	30	9	16	1012.3	1012.1	0.0
14/03:30pm	20.2	16.8	7.6	44	6.5	WNW	15	20	8	11	1012.9	1012.7	0.0
14/03:00pm	20.5	16.1	7.5	43	6.7	W	20	30	11	16	1013.4	1013.2	0.0

Fig. 15.4 Sample data from Melbourne Airport Weather Station

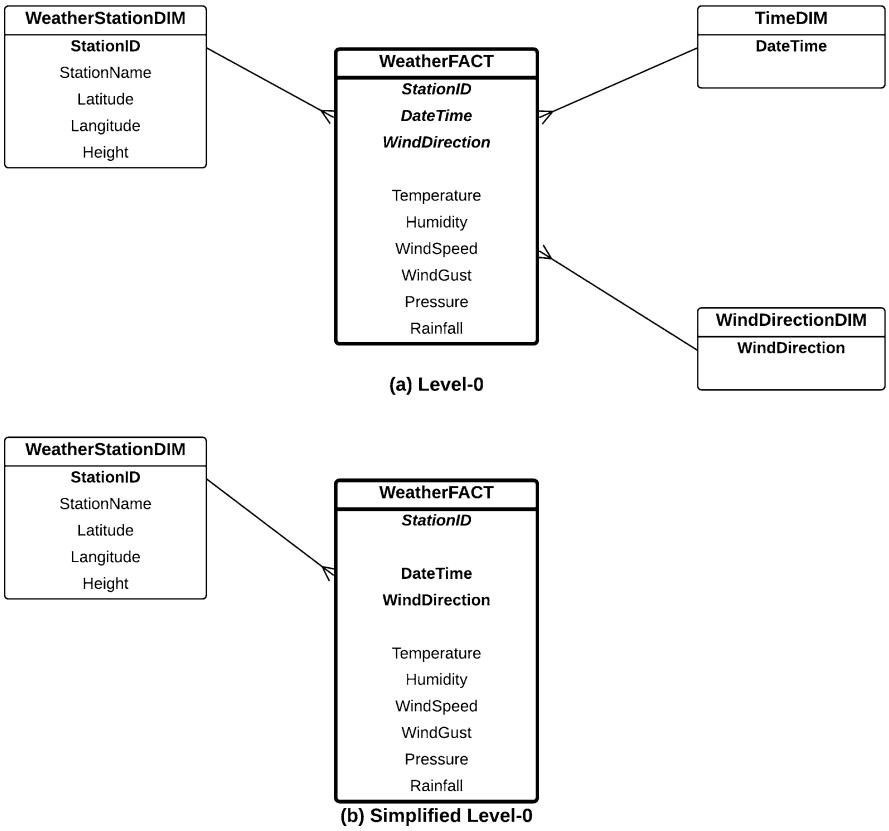

Fig. 15.5 Level-0 star schema for the weather data

column table methodology introduced earlier to validate a star schema, since the Wind Direction Dimension is only related to one fact measure, Wind Speed, and not to other fact measures.

Since two dimensions, Time and Wind Direction Dimensions, are one-attribute dimensions, the star schema is simplified by making Data Time and Wind Direction dimensionless keys in the Fact Table. Figure 15.5 shows the original and simplified Level-0 star schema for the weather data. It is easily noticeable that the Fact Table is almost *identical* to the source weather data (the Fact Table only selected a few important fact measures from the source weather data even though it could have selected all), with only an additional Weather Station Dimension which stores the details of the weather station.

With this Level-0 star schema, we could perform more statistical queries about the weather, such as *Mean*, *Standard Deviation* and *Variance*. The mean (or average) does not require the Fact Table to be at Level-0, as already discussed in the earlier chapters. But for the standard deviation and variance, we need to get each individual

record. Mathematical, standard deviation and variance have the following formulas:

$$stddev = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}} \tag{15.1}$$

$$variance = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N} \tag{15.2}$$

In the formulas, N is the number of records in the Fact Table, x_i is the fact measure of the ith record, and $\bar{x}$ is the mean of the fact measure. Note that *standard deviation* is simply a square root of *variance*.

The SQL commands for standard deviation and variance using the `stddev` and `variance` are as follows. In this example, we only retrieve the temperature and humidity. Without using the built-in `stddev` and `variance` functions, you would need to implement the formulas above manually.

```
select
  stddev(Temperature),
  stddev(Humidity)
from WeatherFact0;

select
  variance(Temperature),
  variance(Humidity)
from WeatherFact0;
```

If we make the star schema a higher level of aggregation, we will lose each individual record, as the records are being aggregated. The Level-1 star schema is shown in Fig. 15.6. Note that Level-1 is made by changing the granularity of the Time Dimension in Level-0 to be the Date Dimension in Level-1. Consequently, the fact measures need to change. In this example, Level-1 has three fact measures: High Temperature, Low Temperature and Highest Wind Gust. High and Low Temperature indicate the maximum and minimum temperature recorded for each day, whereas Highest Wind Gust is the maximum value of Wind Gust for each day. These maximum and minimum values are aggregated values from the original records. Therefore, it will only have one value for each day instead of one value for each recording, which is, in this case, every 30 min. For the clarity of the star schema of Level-1, the Weather Date Dimension and Wind Direction Dimension are kept as one-attribute dimensions. They could be changed to dimensionless keys in the fact if these two dimensions are removed from the star schema. Obviously, using the Level-1 star schema, we cannot query the standard deviation and variance, as they require the original records and they cannot work with already aggregated records. However, calculating the mean can be done in a higher level of aggregation star schema if we have the total of a fact measure and the number of records of a fact measure. However, in this example, the Level-1 star schema does not have these two fact measures. Hence, calculating the mean (e.g. of temperature) still needs to use a Level-0 star schema.

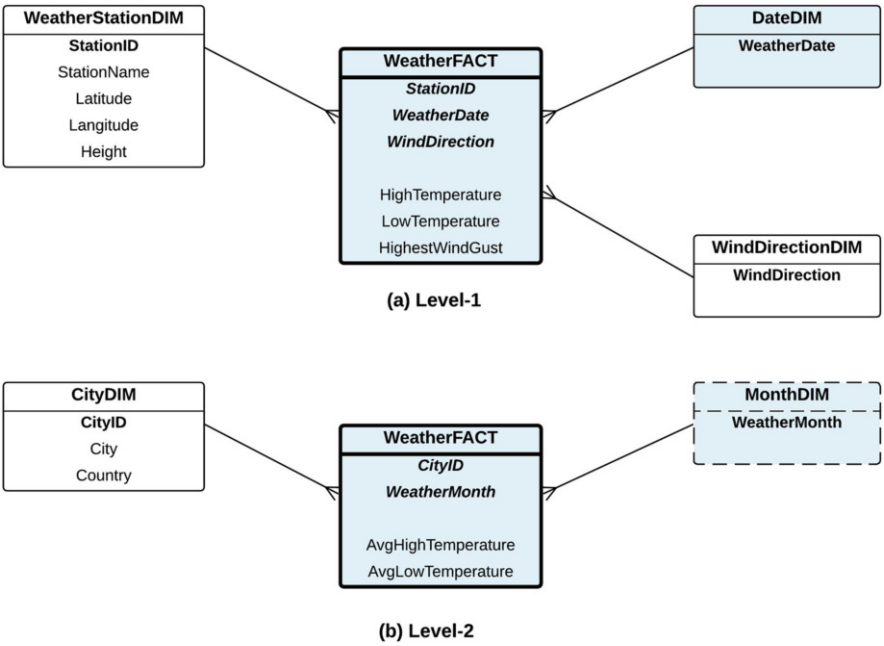

Fig. 15.6 Higher level of aggregate star schemas for the weather data

Figure 15.6 also shows the Level-2 star schema where the Weather Date Dimension is now changed to the Weather Month Dimension in order to make it more general. Usually, monthly weather information only involves Average High and Low Temperatures. In this case, the average is used; hence, the Weather Month Dimension becomes a Determinant Dimension, and because it is a Determinant Dimension, it has to be kept as a dimension. The Determinant Dimension cannot be removed from the star schema.

15.3 Querying Level-0 or a Higher-Level Star Schema

When we want to query the data warehouse, sometimes we need to determine which level of star schema will be able to answer the query. If the query requires the original individual records, obviously we need to query the Level-0 star schema. Previous sections discuss examples of median queries as well as standard deviation and variance queries. These require the original individual records; hence, the query needs to be directed to the Level-0 star schema.

If we need to query something else, and it is already provided by the fact measure, then it is also obvious to which star schema we should send our queries. For example, if we want to find the High Temperature on a particular date, then

the Level-1 star schema as shown in Fig. 15.6 would be able to answer the query immediately. Or if we want to find the Temperature (i.e. Average High Temperature) of a particular city in a particular month, then the query can be directed to a Level-1 star schema. The primary reason for this is that the fact measures have already precomputed the required values.

However, what if we want to query something that is not in the fact measures? For example, it is quite common to query the rainfall on a particular date. The original record has rainfall data every 30 min as the weather recording is taken every 30 min. However, in the Level-1 star schema (daily data), the rainfall fact measure is no longer there. This daily rainfall is not the "average" rainfall of the day (e.g. total rainfall of the day divided by the number of recordings on that day), but may be something else, depending on the definition of rainfall for the day, which could be maximum, or something else determined by the domain expert or regulator. In this example, this daily rainfall is not captured in the Level-1 star schema. Therefore, querying the daily rainfall needs to be directed to the Level-0 star schema.

So, querying something that is not in the fact measure of a higher-level star schema must be directed to the lowest level of star schema which stores each individual record. Basically, fact measures of higher-level star schemas store precomputed values. This also means that these precomputed values must be predefined at the design stage of data warehousing. This also means that we predict what precomputed values might be useful and used quite often later. Obviously, not all aggregated values can be predicted in advance. If this is the case, querying these values must be directed to the lowest level of star schema. On the other hand, if the desired fact measure is already there in the higher-level star schema, querying them will be direct and easy.

15.4 Summary

Designing a star schema from Level-0 is sometimes necessary due to some expected queries using statistical functions requiring data to be presented in its original form. This chapter illustrates the use of some of these basic statistical functions, including median, standard deviation and variance.

Level-0 star schemas are very similar to the E/R diagram, but with a different focus. Star schemas focus on fact measures, whereas the E/R diagram focuses on the relationships between entities. The Level-0 fact measure is non-aggregated. This is very different from fact measures of higher-level star schemas where the fact measures must be aggregated.

Examining higher-level star schemas further, we can see that they are actually precomputed fact measures. Using these precomputed fact measures, querying high-level star schemas can be simplified, compared with retrieving high-level information from Level-0 star schemas. Therefore, precomputed fact measures in high-level star schemas are useful, simple and fast.

Making a star schema a higher level can be done easily by changing the granularity of dimension. This will make the new star schema more general with a higher level of aggregation. Nevertheless, from the pedagogical point of view, sometimes it is easier to learn a star schema by designing the star schema from a higher level of aggregation. Making the star schema more detailed can be done by changing the granularity of the dimension to be more specific as well as adding new dimensions. This method was covered in the previous chapter and will further be explored in the next chapter.

15.5 Exercises

15.1 There are many possible ways to write an SQL query to find the median value. Using the Median House Price case study above, write an SQL query to retrieve the median house price sold in January 2020 in the suburb called Kew without using the median function.

15.2 Again, using the Median House Price case study, write the SQL commands to retrieve the *Mean*, *Standard Deviation* and *Variance* house prices sold in 2020 (e.g. the Property Type is House, and the houses involved in these calculations are located anywhere in the database).

15.6 Further Readings

This book is the only book that discusses thoroughly the distinction between star schema design using higher level and lower level of aggregation. This book particularly focuses on Level-0 which is a star schema without any aggregation. Understanding this concept is crucial in learning how to design star schemas.

General star schema design can be found in various data warehousing books, such as [1–10] and [11].

Data modelling books, such as [12–16] and [17], would also give readers a broader perspective on differences between various levels of aggregation in data modelling.

References

1. C. Adamson, *Star Schema The Complete Reference* (McGraw-Hill Osborne Media, 2010)
2. R. Laberge, *The Data Warehouse Mentor: Practical Data Warehouse and Business Intelligence Insights* (McGraw-Hill, New York, 2011)
3. M. Golfarelli, S. Rizzi, *Data Warehouse Design: Modern Principles and Methodologies* (McGraw-Hill, New York, 2009)

4. C. Adamson, *Mastering Data Warehouse Aggregates: Solutions for Star Schema Performance* (Wiley, London, 2012)
5. P. Ponniah, *Data Warehousing Fundamentals for IT Professionals* (Wiley, London, 2011)
6. R. Kimball, M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (Wiley, London, 2013)
7. R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, B. Becker, *The Data Warehouse Lifecycle Toolkit* (Wiley, London, 2011)
8. W.H. Inmon, *Building the Data Warehouse*. ITPro Collection (Wiley, London, 2005)
9. M. Jarke, *Fundamentals of Data Warehouses*, 2nd edn. (Springer, Berlin, 2003)
10. E. Malinowski, E. Zimányi, *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications*. Data-Centric Systems and Applications (Springer, Berlin, 2008)
11. A. Vaisman, E. Zimányi, *Data Warehouse Systems: Design and Implementation*. Data-Centric Systems and Applications (Springer, Berlin, 2014)
12. T.A. Halpin, T. Morgan, *Information Modeling and Relational Databases*, 2nd edn. (Morgan Kaufmann, Los Altos, 2008)
13. J.L. Harrington, *Relational Database Design Clearly Explained*. Clearly Explained Series (Morgan Kaufmann, Los Altos, 2002)
14. M.J. Hernandez, *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*. For Mere Mortals (Pearson Education, 2013)
15. N.S. Umanath, R.W. Scamell, *Data Modeling and Database Design* (Cengage Learning, 2014)
16. T.J. Teorey, S.S. Lightstone, T. Nadeau, H.V. Jagadish, *Database Modeling and Design: Logical Design*. The Morgan Kaufmann Series in Data Management Systems (Elsevier, Amsterdam, 2011)
17. G. Simsion, G. Witt, *Data Modeling Essentials*. The Morgan Kaufmann Series in Data Management Systems (Elsevier, Amsterdam, 2004)

Chapter 16
Levels of Aggregation: Adding and Removing Dimensions

From the previous chapter, we learned that lowering the level of aggregation of a star schema can be done by adding new dimensions to the star schema. Naturally, when a new dimension is added, in which a new attribute is added to the Fact Table, the level of details of the fact measure will become more detailed; hence, the level of aggregation will be lower.

The opposite is also true, in that, when a dimension is removed from a star schema, the level of aggregation of the star schema becomes higher, resulting in a more general star schema.

These are general rules about adding or removing dimensions to lower or raise the level of aggregation. However, there are some exceptions, that is, when adding a new dimension, the level of aggregation of the star schema does not become lower. We are also going to learn some complexities when a dimension is removed from a star schema. Hence, the aim of this chapter is to understand the impact of the level of aggregation of a star schema when we add or remove a dimension from the star schema.

16.1 Adding New Dimensions

We must be careful when adding a star schema with a new dimension. In particular, there are two things: (i) in some cases, adding a new dimension may not lower down the level of aggregation, and (ii) adding a new dimension may double count values in the fact measure, and hence the fact measure will be incorrect.

D. Taniar, W. Rahayu, *Data Warehousing and Analytics*, Data-Centric Systems and Applications, https://doi.org/10.1007/978-3-030-81979-8_16

16.1.1 Adding New Dimensions Does Not Lower Down the Level of Aggregation

Consider the Olympic star schema (refer to Fig. 16.1), consisting of three dimensions: Country Dimension, Sport Dimension and Olympic Dimension. For simplicity, in each dimension, only the identifier attribute is listed and none of the non-identifier attributes are included. The Country Dimension maintains a list of countries which have participated in the Olympic Games. The Olympic Dimension lists all the Olympic names, including the city and the year (e.g. the London Olympics was in London in 2012, and the Rio Olympics was in Rio, Brazil, in 2016). The Sport Dimension lists all the sport types in the Olympics, such as Swimming, Athletics, Volleyball, etc. Each sport is a category of the sport, not the actual sport event. So, for example, Swimming is a sport type instead of the actual event, such as "100 m Butterfly Men", or "4 × 100 m Freestyle Relay Women". As another example, Volleyball is a sport type, whereas the events are "Volleyball Men" or "Volleyball Women". Hence, Sport Dimension has a high level of aggregation.

Now we are going to add a new dimension called **Medal Dimension**, which has only three records, Gold, Silver and Bronze. The fact measure is reduced from three to one, from Number of Gold Medals, Number of Silver Medals and Number of Bronze Medals, to just Number of Medals. The new star schema with four dimensions is shown in Fig. 16.2.

Note that usually when a dimension is added to a star schema, the level of aggregation of the new star schema is naturally lower than the star schema before the new dimension is added. *Does this statement hold for this case study?* Does the new star schema with four dimensions have a lower level of aggregation than the previous star schema with only three dimensions? To answer this question, we need to examine the data (or the records) in the Fact Table.

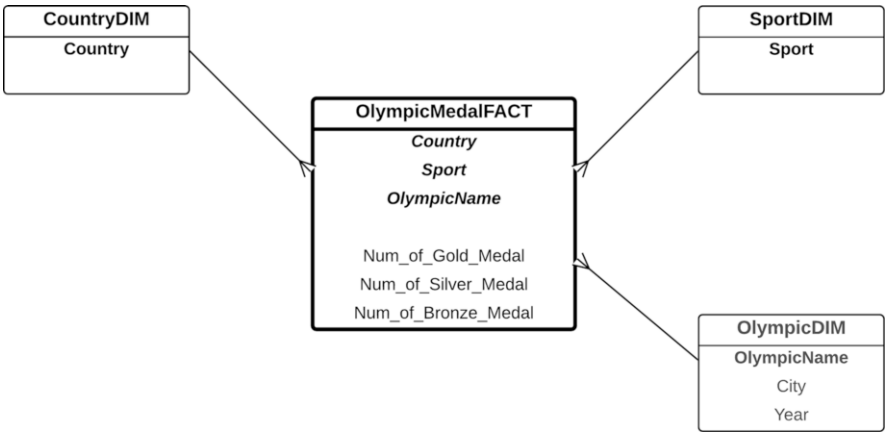

Fig. 16.1 The Olympic Star Schema Version 1: with three dimensions

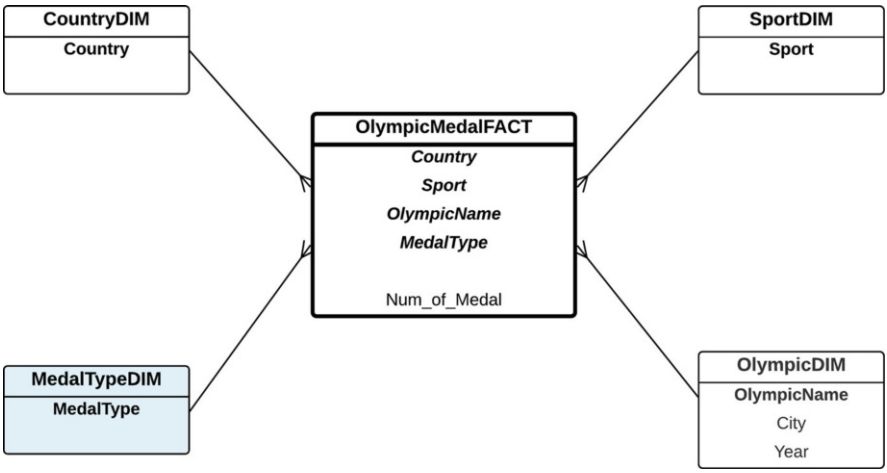

Fig. 16.2 The Olympic Star Schema Version 2: with four dimensions

Table 16.1 The Olympic Games Fact Table (star schema version 1)

Country	Sport	Olympic name	Num of Gold	Num of Silver	Num of Bronze
USA	Swimming	London 2012	16	9	6
China	Swimming	London 2012	5	1	4
Australia	Swimming	London 2012	1	6	3

Table 16.2 The Olympic Games Fact Table (star schema version 2)

Country	Sport	Olympic name	Medal type	Num of medal
USA	Swimming	London 2012	Gold	16
USA	Swimming	London 2012	Silver	9
USA	Swimming	London 2012	Bronze	6
China	Swimming	London 2012	Gold	5
China	Swimming	London 2012	Silver	1
China	Swimming	London 2012	Bronze	4
Australia	Swimming	London 2012	Gold	1
Australia	Swimming	London 2012	Silver	6
Australia	Swimming	London 2012	Bronze	3

The Fact Table for star schema version 1 (without Medal Dimension) has six attributes: three from the dimensions and the other three for the fact measures. The contents of the Fact Table are shown in Table 16.1. For simplicity, we only show three countries which participated in the London 2012 Olympic Games. The Fact Table for star schema version 2 (with Medal Dimension) consists of only five attributes, four from the dimensions but only one fact measure (see Table 16.2).

Comparing the two Fact Tables, it is clear that the Fact Table from star schema version 2 is not more detailed than that from version 1 or vice versa. Hence, star schema version 2 is not of a lower level of aggregation than star schema version 1. Both have the same level of detail and hence the same level of aggregation.

Why is this? This is because the fact measures in both star schemas are different. Star schema version 1 has three fact measures, whereas star schema version 2 has only one fact measure. So, in this case, adding a new dimension to star schema version 1 does not lower the level of aggregation.

The fact in the star schema version 1, which has three fact measures, is often known as a **Pivoted Fact Table** from the fact in star schema version 1 that has only one fact measure. A pivoted Fact Table is often more desirable since it clearly shows the three types of medals as the fact measures. A pivoted Fact Table is only possible when the number of records (which is often the *type* or the *category*) is very small, such as three medal types.

When comparing two star schemas in the context of their level of aggregation, the fact measures must be identical. They cannot be different as in this example. If they are different, the level of aggregation between the two star schemas is not comparable.

16.1.2 Adding New Dimensions May Result in a Double Counting in the Fact Measure

Before discussing the double counting problem, let's lower the level of aggregation of star schema version 2 in the previous section by changing the level of granularity of Sport Dimension. We now replace Sport Dimension (which is basically the sport category) with **SportEvent Dimension**, as shown in Fig. 16.3.

In SportEvent Dimension, we keep the actual sport event, not the sport category. For example, instead of "Swimming", we now have "100 m Butterfly Men", "4 × 100 m Freestyle Relay Women", etc. Naturally, this star schema has a lower level of aggregation compared to the previous star schema.

Now let's examine the records in the Fact Table. In comparing and contrasting the Fact Tables of the two star schemas, let's look at the content of both Fact Tables (Fig. 16.2 is now a Level-1 star schema and Fig. 16.3 is a Level-0 star schema). For simplicity, we focus on the Australian records only (refer to Tables 16.3 and 16.4).

By changing the level of granularity from Sport to Sport Event, the level of granularity of the fact changes. Instead of having only one record for swimming with a bronze medal, now we have the breakdown, which is the three bronze medal records. The Fact Table with Sport Event naturally has a higher level of granularity (or a lower level of aggregation) compared to the Fact Table with Sport, because the Fact Table with Sport Event has 1s in the number of medals (i.e. the fact measure). This star schema is a Level-0 star schema, which means there is no aggregation.

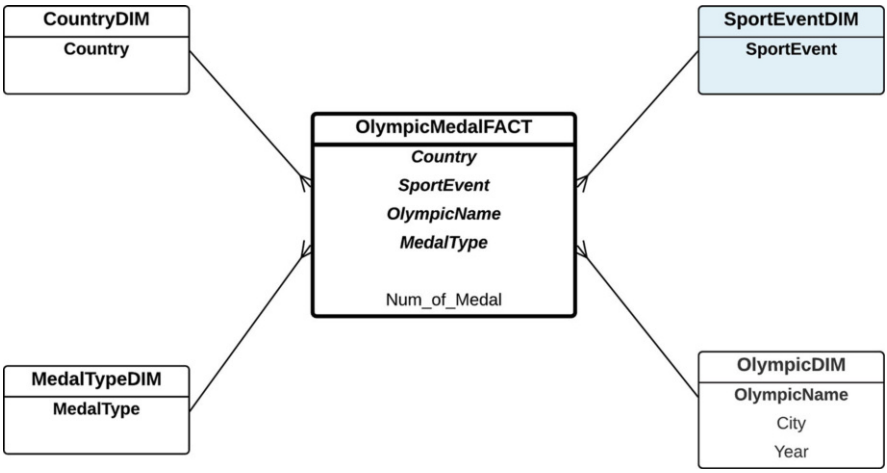

Fig. 16.3 The Olympic star schema: with SportEventDim

Table 16.3 The Olympic Games Fact Table (Level-1 star schema)

Country	Sport	Olympic name	Medal type	Num of medals
Australia	Swimming	London 2012	Gold	1
Australia	Swimming	London 2012	Silver	6
Australia	Swimming	London 2012	Bronze	3

Table 16.4 The Olympic Games Fact Table (Level-0 star schema)

Country	Sport event	Olympic name	Medal type	Num of medal
Australia	4 × 100 m Freestyle Relay Women	London 2012	Gold	1
Australia	4 × 100 m Medley Relay Women	London 2012	Silver	1
Australia	4 × 200 m Freestyle Relay Women	London 2012	Silver	1
Australia	100 m Breaststroke Men	London 2012	Silver	1
Australia	100 m Freestyle Men	London 2012	Silver	1
Australia	200 m Individual Medley Women	London 2012	Silver	1
Australia	100 m Backstroke Women	London 2012	Silver	1
Australia	4 × 100 m Medley Relay Men	London 2012	Bronze	1
Australia	100 m Butterfly Women	London 2012	Bronze	1
Australia	200 m Freestyle Women	London 2012	Bronze	1

What will happen if we decide to add a new dimension called **Athlete Dimension**. The rationale behind this is very simple, that is, to drill down to each winning athlete. The new star schema is then shown in Fig. 16.4.

A snapshot of the Fact Table, focusing on the Australian records, is shown in Table 16.5. By looking at the Fact Table, the last record in this example shows that *Bronte Barratt* was the athlete who received a bronze medal in the 200 m Freestyle

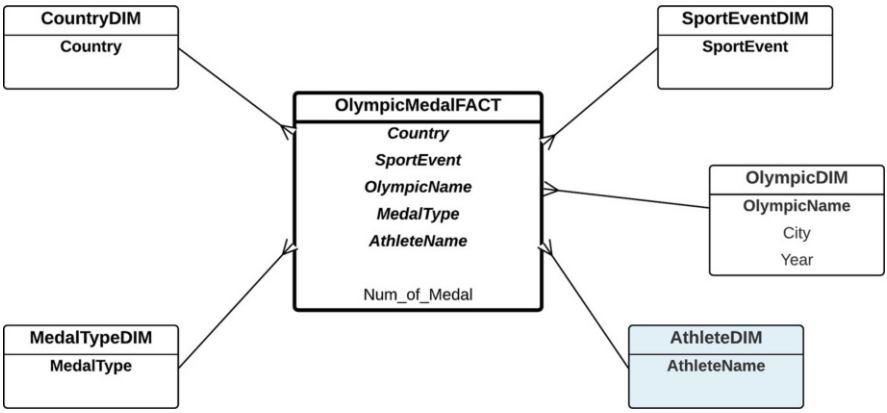

Fig. 16.4 The Olympic Games star schema with an Athlete Dimension

Women. So, drilling down to the winning athlete seems to be correct. From the Fact Table, it is also correct to see that *Bronte Barratt* received two medals: one bronze (200 m Freestyle Women) and one silver (4 × 200 m Freestyle Relay Women).

However, this Fact Table is incorrect because if we query the number of gold medals that Australia received in swimming in the London Olympics 2012, it will return four instead of one. Therefore, adding a new dimension, Athlete Dimension, will result in an incorrect fact measure (e.g. number of medals). The reason for this is because it is double counting the medals.

Double counting the number of medals occurs due to a different focus or a different subject of the star schema. One focus is from the Country point of view, and another focus is from the Athlete point of view. The two cannot be mixed and combined into one star schema.

The solution is to have two star schemas, one star schema which focuses on country and the other which focuses on athlete. The multi-fact star schema is shown in Fig. 16.5.

The Fact Tables are shown in Tables 16.6 and 16.7. If we examine these carefully, we can see that table AthleteMedalFact (see Table 16.7) is also incorrect because when we query "How many gold medals for 4 × 100 Freestyle Relay Women in London 2012 Olympic Games", the answer is four, which is incorrect. The query on the AthleteMedalFact will be correct if the Athlete Dimension is *always* used in any data retrieval on the AthleteMedalFact. Hence, Athlete Dimension must be a *Determinant Dimension*.

The correct multi-fact star schema is shown in Fig. 16.6 (note that we can still keep Country Dimension in the CountryMedalFact schema because Athlete Dimension is a Determinant Dimension).

Table 16.5 Athlete Fact Table

Country	Sport Event	Athlete	Olympic name	Medal type	Num of medal
Australia	4 × 100 m Freestyle Relay Women	Alicia Coutts	London 2012	Gold	1
Australia	4 × 100 m Freestyle Relay Women	Cate Campbell	London 2012	Gold	1
Australia	4 × 100 m Freestyle Relay Women	Brittany Elmslie	London 2012	Gold	1
Australia	4 × 100 m Freestyle Relay Women	Melanie Schlanger	London 2012	Gold	1
Australia	4 × 100 m Medley Relay Women	Emily Seebohm	London 2012	Silver	1
Australia	4 × 100 m Medley Relay Women	Leisel Jones	London 2012	Silver	1
Australia	4 × 100 m Medley Relay Women	Alicia Coutts	London 2012	Silver	1
Australia	4 × 100 m Medley Relay Women	Melanie Schlanger	London 2012	Silver	1
Australia	4 × 200 m Freestyle Relay Women	Bronte Barratt	London 2012	Silver	1
Australia	4 × 200 m Freestyle Relay Women	Melanie Schlanger	London 2012	Silver	1
Australia	4 × 200 m Freestyle Relay Women	Kylie Palmer	London 2012	Silver	1
Australia	4 × 200 m Freestyle Relay Women	Alicia Coutts	London 2012	Silver	1
Australia	100 m Breaststroke Men	Christian Sprenger	London 2012	Silver	1
Australia	100 m Freestyle Men	James Magnussen	London 2012	Silver	1
Australia	200 m Individual Medley Women	Alicia Coutts	London 2012	Silver	1
Australia	100 m Backstroke Women	Emily Seebohm	London 2012	Silver	1
Australia	4 × 100 m Freestyle Relay Men	Hayden Stoeckel	London 2012	Bronze	1
Australia	4 × 100 m Freestyle Relay Men	Christian Sprenger	London 2012	Bronze	1
Australia	4 × 100 m Freestyle Relay Men	Matt Targett	London 2012	Bronze	1
Australia	4 × 100 m Freestyle Relay Men	James Magnussen	London 2012	Bronze	1
Australia	100 m Butterfly Women	Alicia Coutts	London 2012	Bronze	1
Australia	200 m Freestyle Women	Bronte Barratt	London 2012	Bronze	1

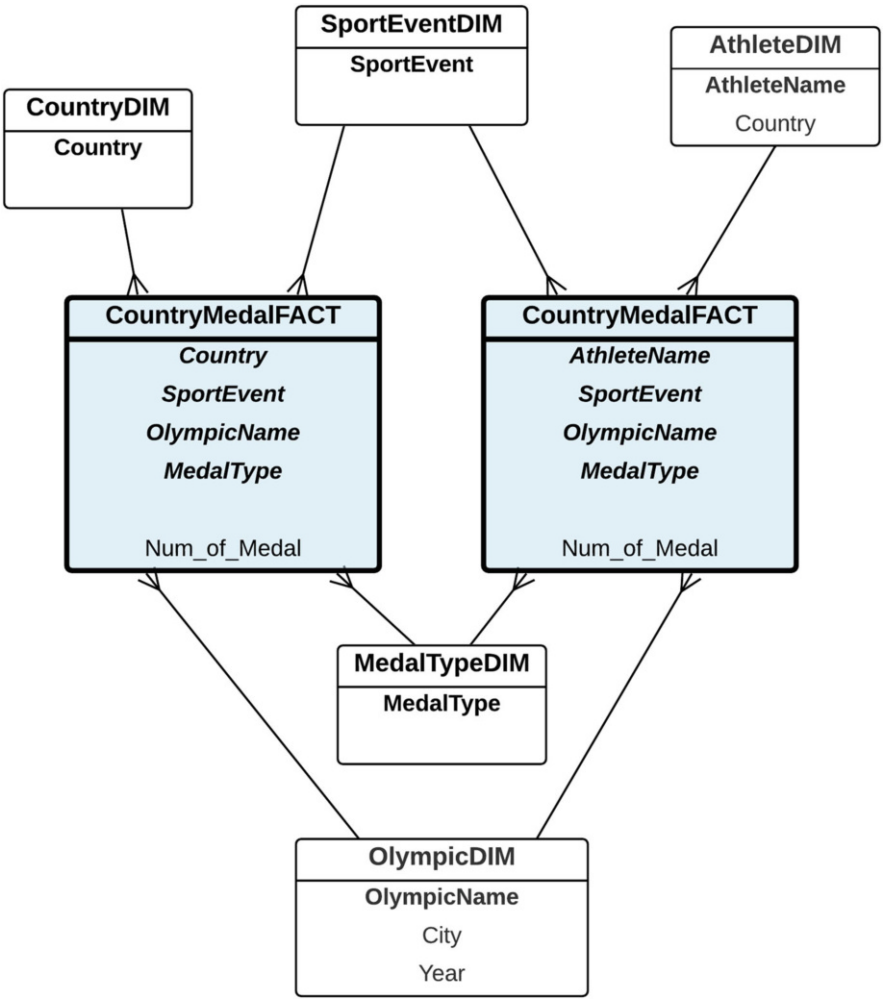

Fig. 16.5 A multi-fact star schema for the Olympic Games, which is still incorrect

16.1.3 The Final Star Schemas

We can avoid a multi-fact schema by having a different level of granularity for the Fact Table with Athlete Dimension. The levels of aggregation for the Olympic Games case study are shown in Figs. 16.7, 16.8, and 16.9.

The corresponding Fact Tables are previously shown in Tables 16.2, 16.6, and 16.7. Note that there are two Level-0s. Level-0a in Table 16.6 shows that the fact measure (e.g. Num_of_Medal) contains 1s (also shown in the Fig. 16.8 star schema). Adding the Athlete Dimension in Fig. 16.9 does not lower down the level of aggregation because neither of the star schemas (e.g. Figs. 16.8 and 16.9) have

Table 16.6 Country Medal Fact

Country	Sport event	Olympic name	Medal type	Num of medal
Australia	4 × 100 m Freestyle Relay Women	London 2012	Gold	1
Australia	4 × 100 m Medley Relay Women	London 2012	Silver	1
Australia	4 × 200 m Freestyle Relay Women	London 2012	Silver	1
Australia	100 m Breaststroke Men	London 2012	Silver	1
Australia	100 m Freestyle Men	London 2012	Silver	1
Australia	200 m Individual Medley Women	London 2012	Silver	1
Australia	100 m Backstroke Women	London 2012	Silver	1
Australia	4 × 100 m Medley Relay Men	London 2012	Bronze	1
Australia	100 m Butterfly Women	London 2012	Bronze	1
Australia	200 m Freestyle Women	London 2012	Bronze	1

any aggregation, and by definition, both are Level-0. Hence, they become Level-0a and Level-0b.

A multiple Level-0 star schema was illustrated in the previous example (see the Purchase Order case study). The thinnest Level-0 star schema has two dimensions, Purchase Order Dimension and Item Dimension. We can have additional two dimensions, Product Dimension and Customer Dimension, without lowering the star schema. The main reason for this is that the star schemas with Purchase Order and Item Dimensions are already at the lowest level of detail as they represent the transactions in the operational database. Each Purchase Order has only one Customer, and each Item has only one Product. Therefore, adding Customer and Product Dimensions does not break down the values of the fact measures.

It is also the same case with the Olympic case study. The star schema in Fig. 16.8 is already at the lowest level because the fact measure contains 1s. Each medal has one Athlete only; hence, adding the Athlete Dimension will not lower down the star schema.

The most important thing about the Level-0 star schema is that each record in the Fact Table in the Level-0 star schema represents a single record in the transaction table in the operational database. If the fact measure is a `count`, the fact measure Level-0 will contain 1s. If the fact measure is a `sum`, the fact measure in Level-0 is the original value of the transaction record. Once the star schema reaches Level-0, adding new dimensions will not break down the fact measures and hence will not lower the level of the star schema.

16.1.4 Summary

In general, adding a new dimension will result in a lower level of aggregation. However, one must take into account the following issues:

1. Adding a new dimension will not lower the level of aggregation, especially when the fact measure has changed. For example, the fact measure changes from three

Table 16.7 Athlete Medal Fact

Athlete	Sport event	Olympic name	Medal type	Num of medal
Alicia Coutts	4 × 100 m Freestyle Relay Women	London 2012	Gold	1
Cate Campbell	4 × 100 m Freestyle Relay Women	London 2012	Gold	1
Brittany Elmslie	4 × 100 m Freestyle Relay Women	London 2012	Gold	1
Melanie Schlanger	4 × 100 m Freestyle Relay Women	London 2012	Gold	1
Emily Seebohm	4 × 100 m Medley Relay Women	London 2012	Silver	1
Leisel Jones	4 × 100 m Medley Relay Women	London 2012	Silver	1
Alicia Coutts	4 × 100 m Medley Relay Women	London 2012	Silver	1
Melanie Schlanger	4 × 100 m Medley Relay Women	London 2012	Silver	1
Bronte Barratt	4 × 200 m Freestyle Relay Women	London 2012	Silver	1
Melanie Schlanger	4 × 200 m Freestyle Relay Women	London 2012	Silver	1
Kylie Palmer	4 × 200 m Freestyle Relay Women	London 2012	Silver	1
Alicia Coutts	4 × 200 m Freestyle Relay Women	London 2012	Silver	1
Christian Sprenger	100 m Breaststroke Men	London 2012	Silver	1
James Magnussen	100 m Freestyle Men	London 2012	Silver	1
Alicia Coutts	200 m Individual Medley Women	London 2012	Silver	1
Emily Seebohm	100 m Backstroke Women	London 2012	Silver	1
Hayden Stoeckel	4 × 100 m Freestyle Relay Men	London 2012	Bronze	1
Christian Sprenger	4 × 100 m Freestyle Relay Men	London 2012	Bronze	1
Matt Targett	4 × 100 m Freestyle Relay Men	London 2012	Bronze	1
James Magnussen	4 × 100 m Freestyle Relay Men	London 2012	Bronze	1
Alicia Coutts	100 m Butterfly Women	London 2012	Bronze	1
Bronte Barratt	200 m Freestyle Women	London 2012	Bronze	1

fact measures (Number of Gold Medals, Number of Silver Medals and Number of Bronze Medals) to one fact measure (Number of Medals). Adding a new Medal Dimension to the star schema will not change the level of granularity of the star schema when the fact measures have also changed.

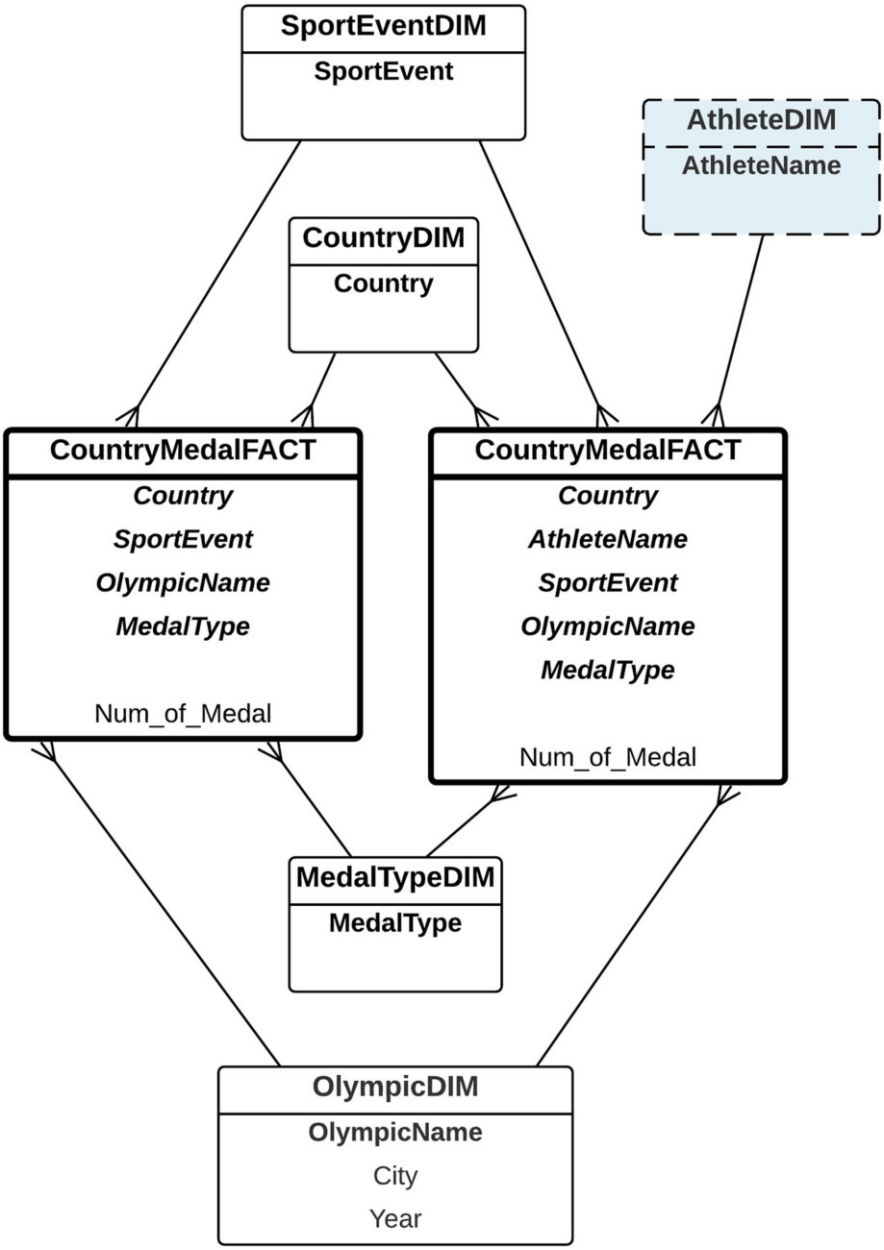

Fig. 16.6 Athlete Dimension is a Determinant Dimension (the correct multi-fact star schema)

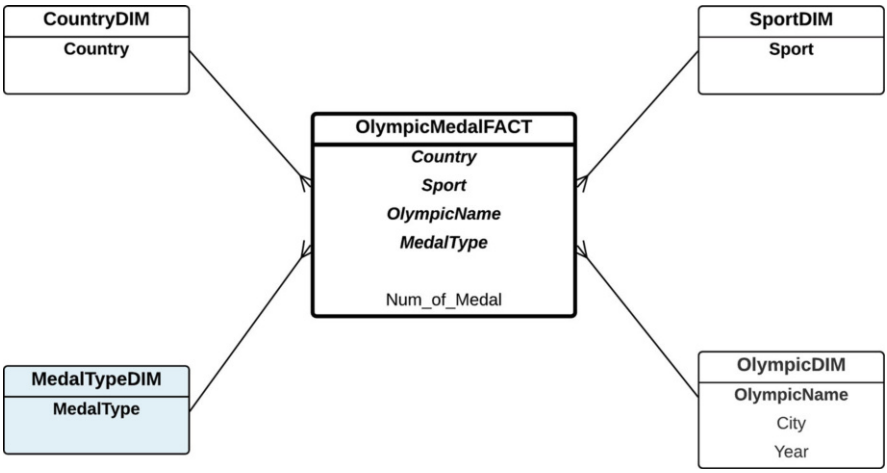

Fig. 16.7 Level-1 star schema with the initial four dimensions

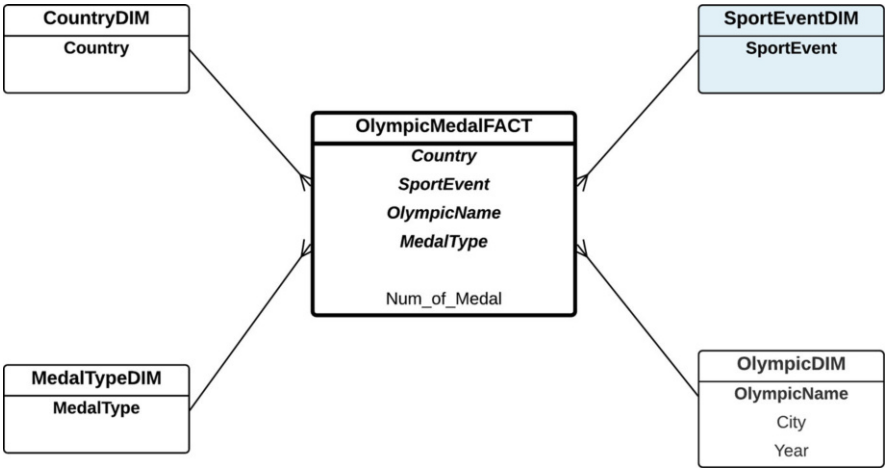

Fig. 16.8 Level-0a star schema with SportEventDim, instead of SportDim

2. Adding a new dimension will create an incorrect Fact Table if the new dimension is double counting the fact measure. For example, adding a new Athlete Dimension will result in double counting the number of medals for the same sport event (e.g. a group sport event).
3. Adding a new dimension to the Level-0 star schema will not lower the level of aggregation of the new star schema. For example, adding Athlete Dimension to Level-0 star schema will not lower down the star schema, because it is already the lowest level.

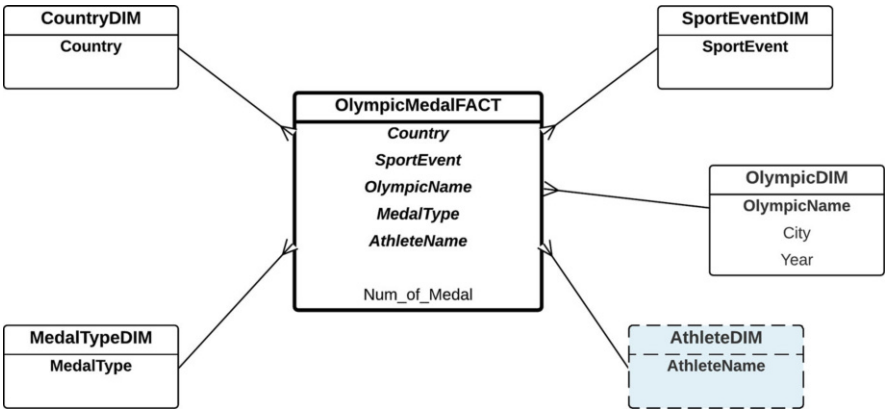

Fig. 16.9 Level-0b star schema with AthleteDim as a Determinant Dimension

16.2 Removing Dimensions

There are two important lessons from the previous section on "Adding New Dimensions" and its implications for the level of aggregation: (i) a double counting of the fact measure problem when adding a new dimension which is a Determinant Dimension and (ii) the breaking down of 1s in the fact measure into a lower level of aggregation.

In this section, we are going to examine these two elements, namely (i) the impact of removing dimensions to level of aggregation, especially in regard to removing a Determinant Dimension, and (ii) aggregating values of the fact measure into a higher level of aggregation. Hence, in this section, we are going learn about the impact of Determinant Dimensions on the level of aggregation.

16.2.1 An Employee Case Study

Consider the E/R diagram as shown in Fig. 16.10 which contains employee, department and job entities. Each employee has a department and a jobID. The sample records are shown in Tables 16.8, 16.9, and 16.10. For simplicity, the details of some attributes are not displayed. However, pay particular attention to the Hire Date attribute which indicates when each employee commenced employment.

The star schema as shown in Fig. 16.11 captures the number of employees for each job department and time (month). The calculation of the fact measure, namely Number of Employees, is not straightforward. Employee 101 started her job in November 2016, and she was the only employee to be recruited that month. It was not until February 2017 that Employee 102 started their job. This means that in

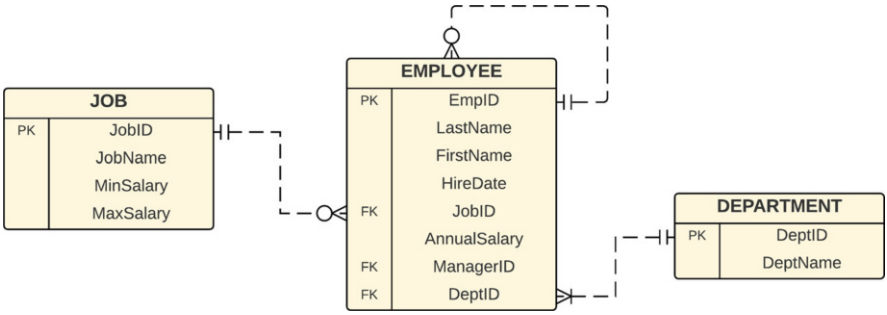

Fig. 16.10 Employee E/R diagram

Table 16.8 Employee table

EmpID	Last name	First name	Hire date	JobID	Annual salary	ManagerID	DeptID
101	Koh	Katie	1-Nov-2016	SRep			D01
102	Li	Liam	1-Feb-2017	SRep			D01
103	Mao	Mary	1-May-2017	SRep			D01
104	Qi	Queeny	1-May-2017	Acc			D02
...	...	...	...	...			...

Table 16.9 Department table

DeptID	Department name
D01	Cosmetic
D02	Accounting
...	...

Table 16.10 Job table

JobID	Job name	Min salary	Max salary
SRep	Sales Representative		
Acc	Accounting		
...	...		

February 2017, there were two employees (employees 101 and 102). Therefore, the TimeID is not only the Hire Date but also the months after the hire date.

Let's assume that the data warehouse captures only for the duration from November 2016 to June 2017 (see the Time Dimension shown in Table 16.11). The Fact Table is shown in Table 16.12.

The SQL command to create the Employee Fact Table is as follows. Assume that the Time Dimension table has been created, which consists of records showing each month between November 2016 and June 2017. When creating the Employee Fact Table, we need to join the Employee table and the Time Dimension table and the join condition is Hire Date $\leq$ TimeID. This means that the months after the hire date will capture the information that the employee is still working. Consequently, when counting the number of employees in any month after the hire date, this employee will be counted.

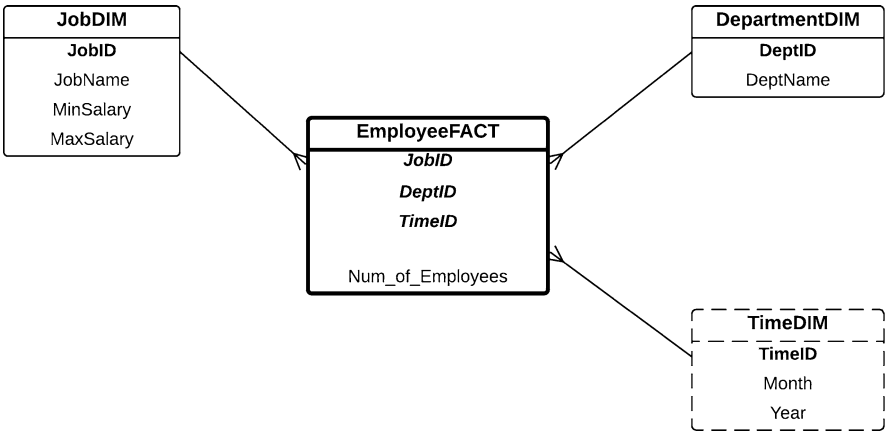

Fig. 16.11 Employee star schema

Table 16.11 Time Dimension

TimeID	Month	Year
201611	Nov	2016
201612	Dec	2016
201701	Jan	2017
201702	Feb	2017
201703	Mar	2017
201704	Apr	2017
201705	May	2017
201706	Jun	2017

Table 16.12 Employee Fact

JobID	DeptID	TimeID	Num of employees
SRep	D01	201611	1
SRep	D01	201612	1
SRep	D01	201701	1
SRep	D01	201702	**2**
SRep	D01	201703	**2**
SRep	D01	201704	**2**
SRep	D01	201705	**3**
SRep	D01	201706	**3**
Acc	D02	201705	1
Acc	D02	201706	1

```
create table EmployeeFact as
select E.JobID, E.DeptID, T.TimeID,
   count(*) as Num_of_Employees
from Employee E, TimeDim T
where to_char(E.HireDate, 'YYYYMM') <= T.TimeID
group by JobID, DeptID, TimeID;
```

Table 16.13 Employee table

EmpID	Last name	First name	Hire Date	Cease Date	JobID	Annual Salary	ManagerID	DeptID
101	Koh	Katie	1-Nov-2016	31-Mar-2017	SRep			D01
102	Li	Liam	1-Feb-2017	null	SRep			D01
103	Mao	Mary	1-May-2017	null	SRep			D01
104	Qi	Queeny	1-May-2017	null	Acc			D02
...	...	...	...	...	...			...

When querying the Employee Fact about Number of Employees, the Time Dimension must always be used; hence, the Time Dimension is a Determinant Dimension, as shown in the above star schema. One example is to ask "how many employees were working in March 2017?", or "how many employees were working in June 2017?". The answers are 2 and 4, respectively. The SQL for the second query is as follows:

```
select TimeID, sum(Num_of_Employees)
from EmployeeFact
where TimeID = '201706'
group by TimeID;
```

The problem will be more complex if there is an attribute Cease Date for each employee record (see the sample data in Table 16.13). For example, Employee 101 quits her job at the end of March 2017. The number of employees must be counted month by month. In this case, there were two employees in Feb and Mar 2017, but in Apr 2017, there was only one employee (not two) because one employee quit her job.

The SQL to create the EmployeeFact Table is as follows. Note that it needs to incorporate the CeaseDate attribute in the join condition.

```
create table EmployeeFact as
select E.JobID, E.DeptID, T.TimeID,
   count(*) as Num_of_Employees
from Employee E, TimeDim T
where to_char(E.HireDate, 'YYYYMM') <= T.TimeID
and to_char(E.CeaseDate, 'YYYYMM') >= T.TimeID
group by JobID, DeptID, TimeID;
```

16.2.2 Removing a Determinant Dimension

In the previous lesson, we learned that to make a star schema more detailed (i.e. lowering the level of aggregation), you can add a dimension to the star schema. The opposite is also true. When we remove a dimension from a star schema, the level of aggregation of the star schema is higher and the star schema becomes more general.

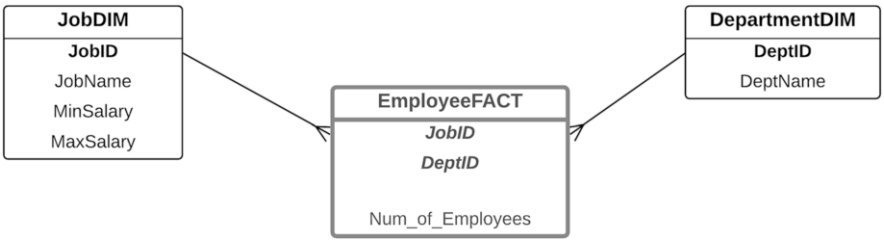

Fig. 16.12 Employee star schema with two dimensions

Table 16.14 Employee Fact—incorrect

JobID	DeptID	Num of employees
SRep	D01	15
SRep	D01	2

Table 16.15 Employee Fact—correct

JobID	DeptID	Num of employees
SRep	D01	3
SRep	D01	1

Suppose we remove the Time Dimension from the above star schema (refer to Fig. 16.11). The new star schema is shown in Fig. 16.12. The new star schema now has only two dimensions, Job and Department Dimensions.

When moving the level of aggregation up or down, usually the value in the fact measure is broken down (when we lower the level of aggregation) or is aggregated or summed up (when we increase the level of aggregation). For example, in the case of Num of Employees, when we remove the Time Dimension, we expect that the Num of Employees of the same TimeID will simply be aggregated or summed up. However, when we do this, we will get the incorrect number of employees, as shown in Table 16.14. There are 15 employees employed in a Sales Rep job in Department D01. Therefore, simply summing up the records from the lower level of aggregation is not correct in this case study. The correct Employee Fact should be as shown in Table 16.15. There are 3 employees (not 15) employed in a Sales Rep job in Department D01.

The main question is why doesn't simply summing up the fact measure when we move to a higher level of aggregation work in this case? The answer is because the dimension that we remove is a "Determinant Dimension". Therefore, removing a Determinant Dimension implies that we need to recalculate the fact measure as simply aggregating up or summing up will not produce the correct results.

The SQL command to produce the new Employee Fact is as follows. Note that joining with the Time Dimension is removed because there is no Time Dimension in this level of star schema.

```
create table EmployeeFact as
select E.JobID, E.DeptID,
   count(*) as Num_of_Employees
from Employee E
group by JobID, DeptID;
```

However, if we remove the Time Dimension, as in the second star schema, the calculation of the Number of Employees can be difficult if the star schema does not maintain the history of employees. Therefore, this higher level of aggregation star schema (the second star schema above) will not accurately reflect the status of the number of employees as the Time Dimension is missing.

16.2.3 Summary

In general, removing a dimension will result in a higher level of aggregation (less granularity or more general). However, one must take into account the following issues:

1. Removing a Determinant Dimension may affect the calculation of the fact measure when increasing the level of aggregation.
2. Aggregating or summing up the fact measure values to a higher level may be incorrect. Hence, the fact measure must be recalculated at each level.

16.3 Exercises

16.1 Given the Fact Table in Table 16.2, write the SQL commands to convert this table to the table in Table 16.1.

16.2 Looking at the Country Fact Table in Table 16.6, write the SQL command (using `Group By Rollup`) to produce the report as shown in Table 16.16.

16.3 Given the Athlete Fact Table in Table 16.7, write the SQL commands to convert this table to the Country Fact Table shown in Table 16.6.

16.4 Further Readings

This book is the only book that discusses thoroughly the distinction between star schema design using higher level and lower level of aggregation. Understanding this

Table 16.16 Country Medal Report

Olympic name	Country	Sport event	Medal type	Num of medal
London 2012	Australia	4 × 100 m Freestyle Relay Women	Gold	1
London 2012	Australia	All Sports Events	Gold	1
London 2012	Australia	4 × 100 m Medley Relay Women	Silver	1
London 2012	Australia	4 × 200 m Freestyle Relay Women	Silver	1
London 2012	Australia	100 m Breaststroke Men	Silver	1
London 2012	Australia	100 m Freestyle Men	Silver	1
London 2012	Australia	200 m Individual Medley Women	Silver	1
London 2012	Australia	100 m Backstroke Women	Silver	1
London 2012	Australia	All Sports Events	Silver	6
London 2012	Australia	4 × 100 m Medley Relay Men	Bronze	1
London 2012	Australia	100 m Butterfly Women	Bronze	1
London 2012	Australia	200 m Freestyle Women	Bronze	1
London 2012	Australia	All Sports Events	Bronze	3
London 2012	Australia	All Sports Events	All Medals	10

concept is crucial in learning how to design star schemas, and this chapter focuses on the impact of adding and removing dimensions to the granularity level of a star schema.

General star schema design can be found in various data warehousing books, such as [1–11]. Data modelling books, such as [12–17], would also give readers a broader perspective on the differences between various levels of aggregation in data modelling.

References

1. C. Adamson, *Star Schema The Complete Reference* (McGraw-Hill Osborne Media, New York, 2010)
2. R. Laberge, *The Data Warehouse Mentor: Practical Data Warehouse and Business Intelligence Insights* (McGraw-Hill, New York, 2011)
3. M. Golfarelli, S. Rizzi, *Data Warehouse Design: Modern Principles and Methodologies* (McGraw-Hill, New York, 2009)
4. C. Adamson, *Mastering Data Warehouse Aggregates: Solutions for Star Schema Performance* (Wiley, New York, 2012)
5. P. Ponniah, *Data Warehousing Fundamentals for IT Professionals* (Wiley, New York, 2011)
6. R. Kimball, M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (Wiley, New York, 2013)
7. R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, B. Becker, *The Data Warehouse Lifecycle Toolkit* (Wiley, New York, 2011)
8. W.H. Inmon, *Building the Data Warehouse*. ITPro Collection (Wiley, New York, 2005)
9. M. Jarke, *Fundamentals of Data Warehouses*, 2nd edn. (Springer, Berlin, 2003)
10. E. Malinowski, E. Zimányi, Advanced data warehouse design: from conventional to spatial and temporal applications, in *Data-Centric Systems and Applications* (Springer, Berlin, 2008)

11. A. Vaisman, E. Zimányi, Data warehouse systems: design and implementation, in *Data-Centric Systems and Applications* (Springer, Berlin, 2014)
12. T.A. Halpin, T. Morgan, *Information Modeling and Relational Databases*, 2nd edn. (Morgan Kaufmann, Los Altos, 2008)
13. J.L. Harrington, Relational database design clearly explained, in *Clearly Explained Series* (Morgan Kaufmann Publishers, New York, 2002)
14. M.J. Hernandez, *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*. For Mere Mortals. (Pearson Education, London, 2013)
15. N.S. Umanath, R.W. Scamell, *Data Modeling and Database Design* (Cengage Learning, Boston, 2014)
16. T.J. Teorey, S.S. Lightstone, T. Nadeau, H.V. Jagadish, Database modeling and design: logical design, in *The Morgan Kaufmann Series in Data Management Systems* (Elsevier, Amsterdam, 2011)
17. G. Simsion, G. Witt, Data modeling essentials, in *The Morgan Kaufmann Series in Data Management Systems* (Elsevier, Amsterdam, 2004)

Chapter 17
Levels of Aggregation and Bridge Tables

The levels of aggregation discussed so far are in the context of normal dimensions which are directly linked to the Fact. This chapter discusses the impact of the levels of aggregation when bridge tables exist in the star schema. The main question to be answered is how a dimension connected to a bridge table can be made more general, making the level of aggregation of the star schema higher.

17.1 Bridge Table: Truck Delivery Case Study

Let's revisit the Truck Delivery case study, which has a bridge table connecting the Trip and the Store Dimensions. The E/R diagram of the Truck Delivery operational database system is shown in Fig. 17.1.

The sample data is shown in Tables 17.1, 17.2, 17.3, and 17.4 (we focus only on the tables that will be used in the star schema, i.e. the Trip table, Truck table, Destination table and Store table).

The star schema of the Truck Delivery system is shown in Fig. 17.2. The Trip Dimension is linked to the Store Dimension through a Bridge Table.

The Trip Dimension has a WeightFactor and a StoreGroupList (ListAgg) attributes. The Trip Dimension table is shown in Table 17.5. The StoreGroupList attribute contains a list of stores for each trip, whereas the WeightFactor indicates the proportion of each store's contribution to the Total Delivery Cost in the Fact. The Bridge Table and Store Dimension table are shown in the Destination and Store tables in Tables 17.3 and 17.4. We can easily see that the StoreGroupList attribute values in the Trip Dimension table come from the Destination table, which lists all the stores for each trip.

The granularity of the Fact Table is determined by the Trip. In other words, the granularity is on a "Trip" level. Each trip will have an entry in the Fact Table. The Fact Table is shown in Table 17.6.

D. Taniar, W. Rahayu, *Data Warehousing and Analytics*, Data-Centric Systems and Applications, https://doi.org/10.1007/978-3-030-81979-8_17

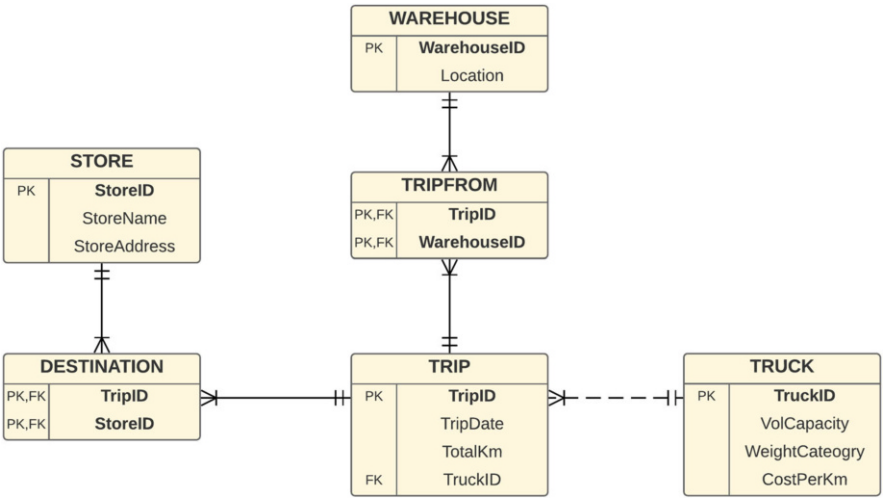

Fig. 17.1 Truck Delivery E/R diagram

Table 17.1 Trip table

TripID	Date	TotalKm	TruckID
Trip1	14-Apr-2018	370	Truck1
Trip2	14-Apr-2018	250	Truck2
Trip3	15-Apr-2018	375	Truck1
Trip4	16-Jul-2018	340	Truck1
Trip5	16-Jul-2018	175	Truck2
...	...	...	...

Table 17.2 Truck table

TruckID	VolCapacity	WeightCategory	CostPerKm
Truck1	250	Medium	$1.20
Truck2	300	Medium	$1.50
...	...	...	...

In order to understand this Fact Table, let's look at the Trip table in the operational database, which is shown in Table 17.1. There are five trips (namely Trip1 to Trip5). Trip1 to Trip3 were done in April (in the Autumn season), whereas Trip4 and Trip5 were done in July (in the Winter season). Because the granularity of the Fact Table is based on the Trip, the Fact Table has five records, the same number of records as in the Trip table.

Table 17.3 Destination table

TripID	StoreID
Trip1	M1
Trip1	M2
Trip1	M4
Trip1	M3
Trip1	M8
Trip2	M4
Trip2	M1
Trip2	M2
Trip3	M1
Trip3	M2
Trip3	M3
Trip3	M4
Trip3	M8
Trip4	M1
Trip4	M3
Trip4	M4
Trip4	M2
Trip4	M8
Trip5	M1
Trip5	M2
...	...

Table 17.4 Store table

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

17.1.1 Combining Trips: TripGroupList

One method to increase the level of aggregation of a star schema is by changing the granularity of an existing dimension by making the dimension more general or a lower granularity. Since the Truck Delivery star schema has the Trip as the granularity, to increase the level of aggregation of this star schema, we should not use the Trip as the level of granularity, because the Trip level granularity is rather high. In other words, the level of aggregation is low because it is done at a Trip level. Increasing the level of aggregation means that some trips need to be combined.

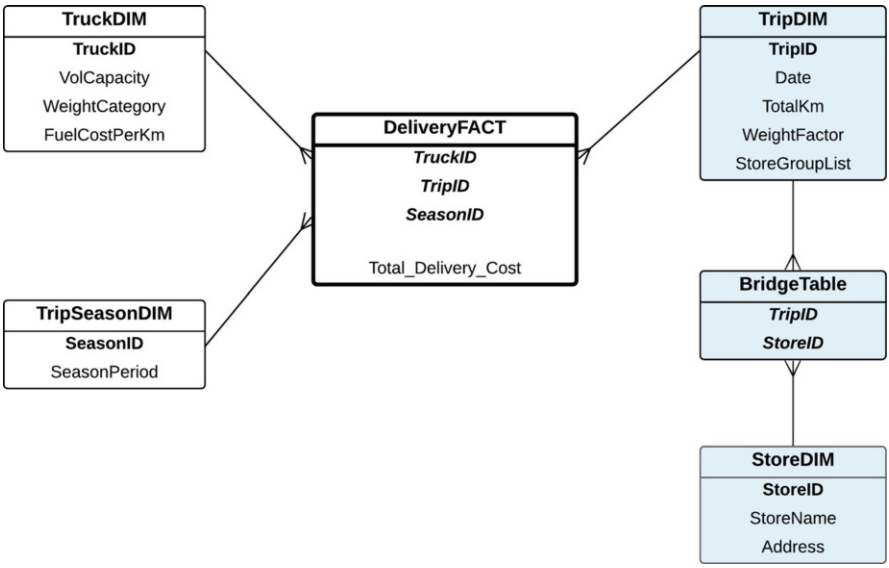

Fig. 17.2 Truck Delivery star schema

Table 17.5 Trip Dimension table

TripID	Date	TotalKm	WeightFactor	StoreGroupList
Trip1	14-Apr-2018	370	0.20	M1_M2_M3_M4_M8
Trip2	14-Apr-2018	250	0.33	M1_M2_M4
Trip3	15-Apr-2018	375	0.20	M1_M2_M3_M4_M8
Trip4	16-Jul-2018	340	0.20	M1_M2_M3_M4_M8
Trip5	16-Jul-2018	175	0.50	M1_M2
...	...	...	...	...

Table 17.6 Fact Table

TruckID	SeasonID	TripID	Total Delivery Cost
Truck1	Autumn	Trip1	444.00
Truck2	Autumn	Trip2	375.00
Truck1	Autumn	Trip3	450.00
Truck1	Winter	Trip4	408.00
Truck2	Winter	Trip5	262.50

Consequently, the level of aggregation is based on "multiple trips", not based on "single trips". The question is how to combine several trips into one record. What is the base for combining multiple trips into one?

Remember that each trip may deliver goods to many stores. For example, Trip1 delivers to five stores, namely M1, M2, M3, M4 and M8, as shown by the StoreGroupList attribute in the Trip Dimension table (refer to Table 17.5). Also notice that Trip3 delivers to the same five stores, also by Truck1 and in Autumn.

Table 17.7 TripGroupList Dimension table

TripGroupListID	TotalKm	WeightFactor	StoreGroupList
Trip1_Trip3	745	0.20	M1_M2_M3_M4_M8
Trip2	250	0.33	M1_M2_M4
Trip4	340	0.20	M1_M2_M3_M4_M8
Trip5	175	0.50	M1_M2
...	...	...	...

These two trips (e.g. Trip1 and Trip3) should be combined into one record, making the level of granularity lower (e.g. more general). Therefore, instead of storing Trip1 and Trip3 separately, we combine them into one record in the dimension table and make the TripGroupListID the identifier of the new dimension. In this example, we can simply concatenate both trips to become a new trip identifier (e.g. Trip1_Trip3). We call this new dimension table TripGroupListDim as shown in Table 17.7.

The TripGroupList Dimension replaces the Trip Dimension table with TripGroupListID as the identifier. This new dimension consists of three other attributes, which are TotalKm, WeightFactor and StoreGroupList. The TripDate attribute, originally in the Trip Dimension table, is no longer applicable here because one record potentially combines several trips on different dates (but in the same season, as indicated by the Season Dimension). The WeightFactor and StoreGroupList attributes are the same as those in the original Trip Dimension because those trips which are now combined share the same list of stores and hence have the same weight factor. The TotalKm is simply a sum of each individual trip.

The new star schema that uses Trip Group List as the base on the level of aggregation is shown in Fig. 17.3. So, if originally we have, for example, 100 trips, after combining several trips based on their destinations, we potentially reduce the number of records to much less than 100, thereby increasing the level of aggregation. Since the identifier of the TripGroupList Dimension, namely TripGroupListID, is new, this dimension identifier is now in the Fact Table as well as in the Bridge Table to replace the original TripID attribute.

The new Fact Table is shown in Table 17.8. Note that the third column of the Fact Table is TripGroupListID, which links to the TripGroupList Dimension table as shown in Table 17.7. Also compared with the original Fact Table shown earlier in Table 17.6, the new Fact Table in Table 17.7 has a higher level of aggregation because Trip1 and Trip3 are combined into one record. Also notice that the new Total Delivery Cost for these two trips is the sum of the original individual Total Delivery Cost.

The SQL commands to implement this new star schema can be challenging because we need to combine trips. To create the new dimension TripGroupList Dimension table, we need to join the Trip and Destination tables. In addition to calculating the WeightFactor and StoreGroupList, the TripDate attribute needs to be kept as well, because combining trips must have the same Season, Truck as well as StoreGroupList. So, it is essential, for the time being, to include TripDate.

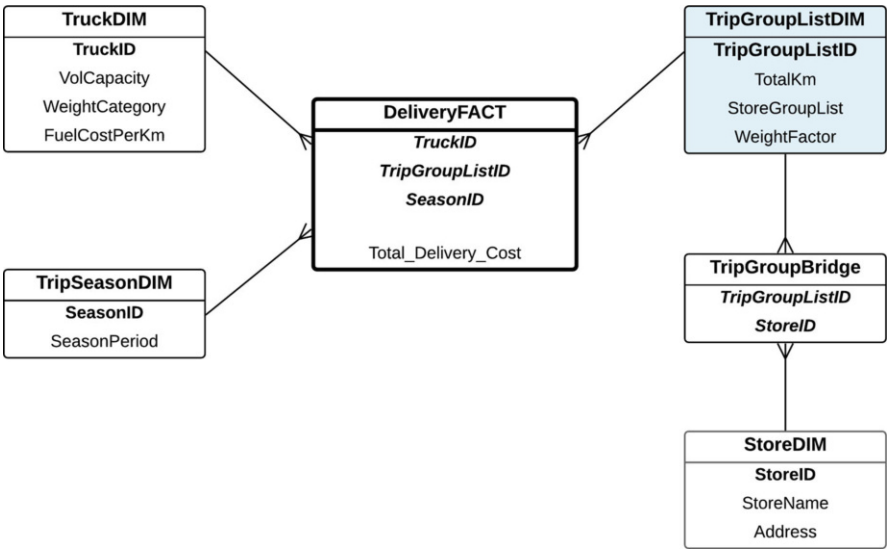

Fig. 17.3 A new star schema with TripGroupList Dimension

Table 17.8 Fact Table

TruckID	SeasonID	TripGroupListID	Total Delivery Cost
Truck1	Autumn	Trip1_Trip3	894.00
Truck2	Autumn	Trip2	375.00
Truck1	Winter	Trip4	408.00
Truck2	Winter	Trip5	262.50

After TripDate is converted into SeasonID, we can then combine trips based on SeasonID, WeightFactor and StoreGroupList. At the same time, the TotalKm must be recalculated. The SQL commands to create the TripGroupList Dimension table are as follows:

```
create table TripGroupListDimTemp 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;

alter table TripGroupListDimTemp
add (SeasonID varchar2(10));

update TripGroupListDimTemp
set SeasonID = 'Summer'
where to_char(TripDate, 'MM') in ('12', '01', '02');
```

```
update TripGroupListDimTemp
set SeasonID = 'Autumn'
where to_char(TripDate, 'MM') in ('03', '04', '05');

update TripGroupListDimTemp
set SeasonID = 'Winter'
where to_char(TripDate, 'MM') in ('06', '07', '08');

update TripGroupListDimTemp
set SeasonID = 'Spring'
where to_char(TripDate, 'MM') in ('09', '10', '11');

create table TripGroupListDim as
select
   listagg(TripID, '_') within group (order by TripID)
     as TripGroupListID,
   sum(TotalKm) as TotalKm,
   WeightFactor, StoreGroupList
from TripGroupListDimTemp
group by SeasonID, WeightFactor, StoreGroupList;
```

As can be seen from the star schema in Fig. 17.3, the Bridge Table connects TripGroupList Dimension and Store Dimension. It contains two attributes, TripGroupListID and StoreID. This Bridge Table is shown in Table 17.9.Note that Trip1 and Trip3 are now combined. They both share the same five stores. But also notice that Trip4, although having the same five stores, is not combined with Trip1 and Trip3. The reason for this is that Trip4 is in a different Season from Trip1 and Trip3. Notice `group by SeasonID, WeightFactor,`

Table 17.9 The New Bridge Table

TripGroupListID	StoreID
Trip1_Trip3	M1
Trip1_Trip3	M2
Trip1_Trip3	M3
Trip1_Trip3	M4
Trip1_Trip3	M8
Trip2	M4
Trip2	M1
Trip2	M2
Trip4	M1
Trip4	M2
Trip4	M3
Trip4	M4
Trip4	M8
Trip5	M1
Trip5	M2
...	...

StoreGroupList when creating the TripGroupList Dimension table. Hence, only the trips sharing the same season (as well as the WeightFactor and StoreGroupList) will be combined. This is why Trip4 is not combined with Trip1 and Trip3.

The SQL to create the Bridge Table is as follows. It needs to join the TripGroupList Dimension and the Destination table, where the TripID in the Destination appears in the TripGroupList Dimension.

```
create table TripGroupBridge as
select distinct TripGroupListID, StoreID
from   TripGroupListDim T, Destination D
where  T.TripGroupListID LIKE ('%'||D.TripID||'%');
```

Creating the TempFact and final Fact Tables is quite standard, except the listagg and sum when creating the Fact Table.

```
create table TruckTempFact2 as
select
   tk.TruckID,
   tp.TripID,
   tp.TripDate,
   (tk.CostPerKm * tp.TotalKm) as ShipmentCost,
   StoreGroupList
from Truck tk, Trip tp, TripGroupListDim tg
where tk.TruckID = tp.TruckID
and tg.TripGroupListID like ('%'||tp.TripID||'%');

alter table TruckTempFact2
add (SeasonID varchar2(10));

update TruckTempFact2
set SeasonID = 'Summer'
where to_char(TripDate, 'MM') in ('12', '01', '02');

update TruckTempFact2
set SeasonID = 'Autumn'
where to_char(TripDate, 'MM') in ('03', '04', '05');

update TruckTempFact2
set SeasonID = 'Winter'
where to_char(TripDate, 'MM') in ('06', '07', '08');

update TruckTempFact2
set SeasonID = 'Spring'
where to_char(TripDate, 'MM') in ('09', '10', '11');

create table TruckFact3 as
select
  TruckID, SeasonID,
  listagg(TripID, '_') within group (order by TripID)
    as TripGroupListID,
  sum(ShipmentCost) as TotalCost
from TruckTempFact2
group by TruckID, SeasonID, StoreGroupList;
```

17.1.2 Combining Trips: StoreGroupList

In the previous chapter, combining trips is done by concatenating trips sharing the same SeasonID, WeightFactor and StoreGroupList. For example, Trip1 is combined with Trip3 to become Trip1_Trip3. Hence, TripID becomes TripGroupListID.

A second method to combine trips is to use the StoreGroupList. The trips with the same StoreGroupList are combined into one record. For example, the trips sharing the same StoreGroupList, such as M1_M2_M3_M4_M8, will have only one record in the StoreGroupList Dimension. The new star schema is shown in Fig. 17.4, and the StoreGroupList Dimension table is shown in Table 17.10.

The main difference between the StoreGroupList Dimension and the TripGroupList Dimension is that in the StoreGroupList Dimension, there is no TripID or TripGroupListID. Also, TotalKm is not included. Another difference is that in StoreGroupList, trips with the same StoreGroupList will be combined into one, that is, Trip1, Trip3 and Trip4 with the same StoreGroupList are combined into one record (although the TripID is not shown). This is not possible in the TripGroupList method because TripGroupListID appears in the Fact. If Trip1, Trip3 and Trip4 are combined in one TripGroupListID, then in the Fact Table, this will conflict

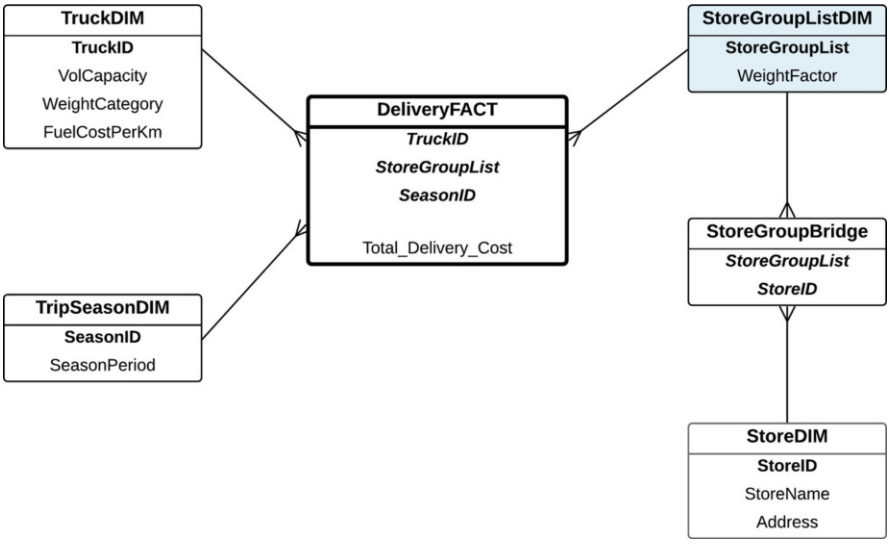

Fig. 17.4 A new star schema with StoreGroupList Dimension

Table 17.10 StoreGroupList Dimension table

StoreGroupList	WeightFactor
M1_M2_M3_M4_M8	0.20
M1_M2_M4	0.33
M1_M2	0.50
...	...

Table 17.11 Fact Table

TruckID	SeasonID	StoreGroupList	Total Delivery Cost
Truck1	Autumn	**M1_M2_M3_M4_M8**	894.00
Truck2	Autumn	M1_M2_M4	375.00
Truck1	Winter	**M1_M2_M3_M4_M8**	408.00
Truck2	Winter	M1_M2	262.50

with other dimensions, especially the Season Dimension because the three trips were not made in the same Season. However, with the StoreGroupList approach, it is possible to combine the trips with the same StoreGroupList, although they have different Season and different Truck because the Fact has StoreGroupList, not TripGroupListID.

The new Fact Table is shown in Table 17.11. Note that StoreGroupList M1_M2_M3_M4_M8 is repeated twice, one for Truck1 Autumn, which is basically Trip1 and Trip3, and the other for Truck1 Winter, which is Trip4. Imagine that in the TripGroupList approach, as discussed in the previous section, if Trip1, Trip3 and Trip4 are combined into one TripGroupListID, the Fact Table will be incorrect.

The SQL codes to create the TempFact and the new Fact are as follows:

```
create table TruckTempFact4 as
select
   tk.TruckID,
   tp.TripID,
   tp.TripDate,
   (tk.CostPerKM * tp.TotalKm) As ShipmentCost
from Truck tk, Trip tp
where tk.TruckID = tp.TruckID;

alter table TruckTempFact4
add (SeasonID varchar2(10));

update TruckTempFact4
set SeasonID = 'Summer'
where to_char(TripDate, 'MM') in ('12', '01', '02');

update TruckTempFact4
set SeasonID = 'Autumn'
where to_char(TripDate, 'MM') in ('03', '04', '05');

update TruckTempFact4
set SeasonID = 'Winter'
where to_char(TripDate, 'MM') in ('06', '07', '08');

update TruckTempFact4
set SeasonID = 'Spring'
where to_char(TripDate, 'MM') in ('09', '10', '11');

alter table TruckTempFact4
add (StoreGroupList varchar(100));
```

```
update TruckTempFact4 T
set StoreGroupList = (
  select listagg(D.StoreId, '_') within group
         (order by D.TripId) as StoreGroupList
  from Destination D
  where T.TripId = D.TripId
);

create table TruckFact4 as
select
  TruckId, SeasonId, StoreGroupList,
  sum(ShipmentCost) as TotalCost
from TruckTempFact4
where StoreGroupList is not null
group by TruckId, SeasonId, StoreGroupList;
```

The Bridge Table must also use StoreGroupList instead of TripGroupListID. It is shown in Table 17.12.

The SQL codes to create the Bridge Table are shown as follows:

```
create table StoreGroupBridge as
select distinct StoreGroupList, StoreID
from   TripGroupListDimTemp S, Destination D
where  S.StoreGroupList like ('%'||D.StoreID||'%');
```

This is similar to creating the Bridge Table in the TripGroupList method, that is, join the new dimension with the Destination table. But in this case, we check if the StoreID is part of the StoreGroupList, whereas in the Bridge Table using the TripGroupList method, it checks if the TripID is part of the TripGroupListID.

The Bridge Table in Table 17.12 is rather meaningless as the information is all redundant and unnecessary. We can actually bypass the Bridge Table and directly connect the StoreGroupList Dimension table with the Store Dimension table, as shown in the new star schema in Fig. 17.5. The relationship between the StoreGroupList Dimension and the Store Dimension is not purely a conventional primary key—foreign key relationship because StoreGroupList is a list of stores, whereas

Table 17.12 The New Bridge Table

StoreGroupList	StoreID
M1_M2_M3_M4_M8	M1
M1_M2_M3_M4_M8	M2
M1_M2_M3_M4_M8	M3
M1_M2_M3_M4_M8	M4
M1_M2_M3_M4_M8	M8
M1_M2_M4	M1
M1_M2_M4	M2
M1_M2_M4	M4
M1_M2	M1
M1_M2	M2
...	...

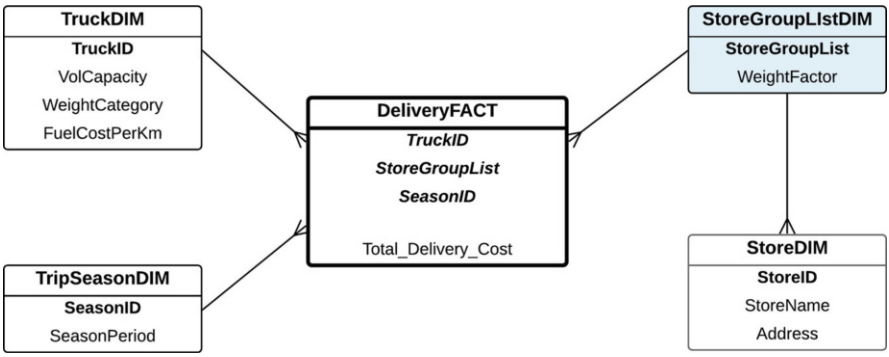

Fig. 17.5 The final star schema without the Bridge

StoreID is one store. So, when we join the StoreGroupList Dimension table and the Store Dimension table, it is not an equi-join like `where StoreGroupList = StoreID`, but is a non-equi-join using `where StoreGroupList like ('%'||StoreID||'%')`.

On the other hand, with the TripGroupList approach as shown in the star schema previously in Fig. 17.3 and the Bridge Table in Table 17.9, it is not possible to remove the Bridge Table, because the Bridge Table uses the TripGroupListID attribute, not StoreGroupList.

17.1.3 Summary

There are two options to combine the trips into one group of trips. The first method is by combining trips that share the same Season, Truck, WeightFactor and StoreGroupList. This is the TripGroupList method. The TripGroupListID, which is the identifier for the TripGroupList Dimension, is a concatenation of individual trips which are now combined. The second method is by combining StoreGroupList that shares the same stores. This is the StoreGroupList method. The StoreGroupList is now used instead of TripGroupListID. Because several trips are combined into one dimension record, the level of aggregation of the star schema is increased and the star schema becomes more general.

17.2 Bridge Table: Product Sales Case Study

The Truck Delivery case study mentioned in the above section shows two methods of combining trips (Trip Dimension) to become a higher level of aggregation: (*i*) combining trips using TripGroupList, which is a concatenation of the combined

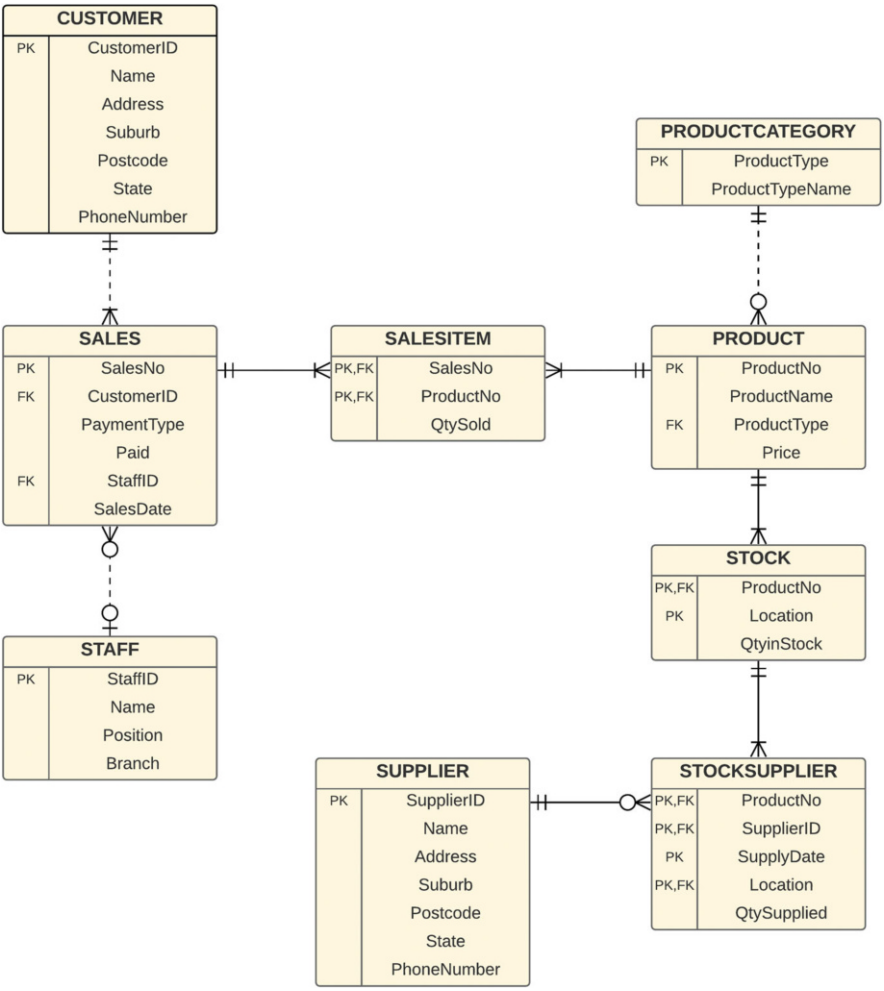

Fig. 17.6 E/R diagram of the Product Sales case study

trips, and (ii) combining trips using StoreGroupList using the list of stores of the combined trips.

This section shows that there is a third method to make a dimension that is connected to a bridge table less granular (making the star schema more general or having a higher level of aggregation). Let's use the Product Sales case study (this case study, together with the Truck Delivery case study, was also used in the Bridge Tables chapter). The E/R diagram of the Product Sales case study is shown in Fig. 17.6. Pay attention to the ProductCategory Entity, in which one Product Category may have several Products.

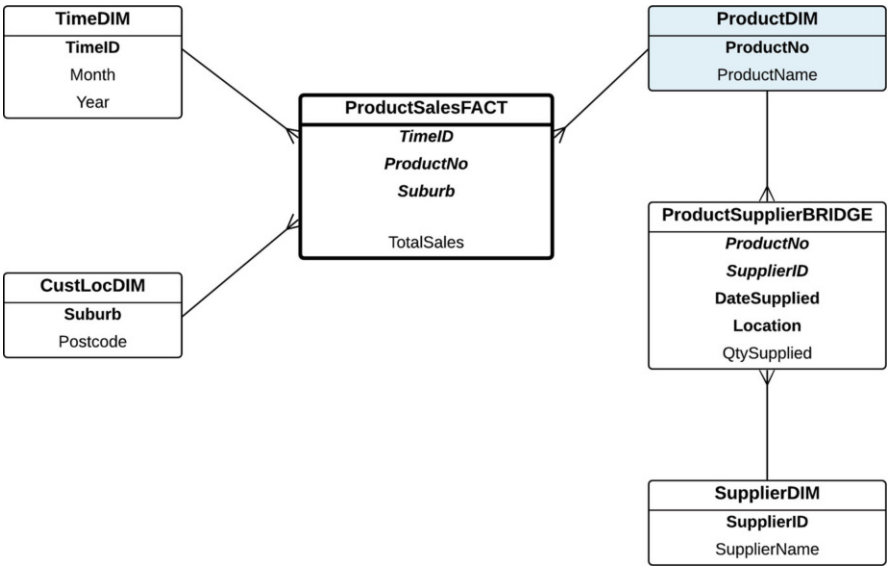

Fig. 17.7 Product Sales star schema

A star schema for the Product Sales case study is shown in Fig. 17.7 (refer to the Bridge Tables chapter for the requirements of this case study). In this star schema, Product Dimension is connected to the Supplier Dimension through a Bridge Table. For simplicity, Product Dimension does not feature the WeightFactor and ListAgg (e.g. SupplierGroupList) attributes.

Using the TripGroupList method as in the previous case study, products sharing the same suppliers can be combined in a ProductGroupList. Consequently, we are making the Product Dimension more general, and it becomes the ProductGroupList Dimension with ProductGroupList as its identifier. Because several products are grouped into one ProductGroupList, the new star schema is more general and has a higher level of aggregation (see the new star schema in Fig. 17.8).

Also notice that the Fact and the Bridge now have ProductGroupList instead of ProductNo. Because the Product Dimension does not have the WeightFactor and SupplierGroupList (e.g. ListAgg) attributes, the ProductGroupList Dimension has only one attribute, which is a combined ProductNo called ProductGroupList. Although the ProductGroupList Dimension has only one attribute, we might think that we can eliminate this dimension and make the Bridge directly connect to the Fact through the ProductGroupList attribute, as both of them share the same attribute. However, this is not possible because the cardinality relationship between the Fact and ProductGroupList Dimension is *m*-1 and ProductGroupList Dimension to the Bridge is 1-*m*. Otherwise, it would be a *m*-*m* between the Fact and the Bridge, which is not allowed in the design.

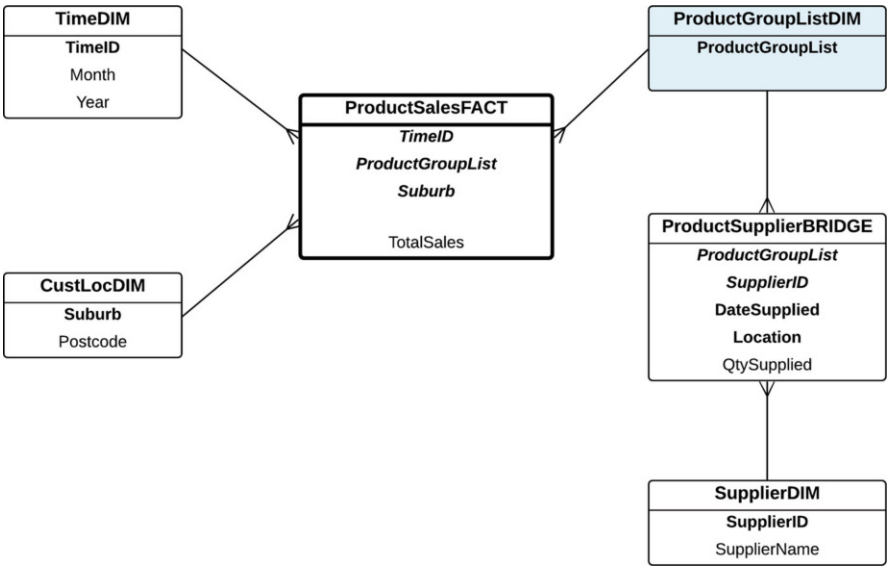

Fig. 17.8 ProductGroupList Dimension

Because the original Product Dimension does not have a SupplierGroupList attribute (e.g. the ListAgg attribute), we cannot apply the second method, like the StoreGroupList method in the Truck Delivery case study. So, the ProductGroupList method is the method that can be used to merge several products together, making the granularity lower or more general. However, if we think more deeply about this ProductGroupList approach, although it is correct that several products share the same supplier list, this is rather meaningless because having this ProductGroupList, each of which is a list of products with the same suppliers, will not have any meaning for the Total Sales. We may not need to find the total sales of a ProductGroupList, that is, finding the total sales of products with the same group of suppliers. Finding the total sales of a particular product, like the original star schema in Fig. 17.7, makes sense, but finding the total sales of ProductGroupList may not be.

So why does TripGroupList make sense but not ProductGroupList? One explanation is that in the Truck Delivery case study, the Total Delivery Cost is based on trips, although logically, it should have been based on stores. This is only because data on the delivery cost per score is not recorded in the operational database. So that is a logical error in the operational database due to the way the E/R diagram is designed. On the contrary, in the Product Sales case study, the Total Sales is already correctly attributed to Product and not to Supplier. Subsequently, grouping Products into ProductGroupList, which has suppliers as the basis, does not make any sense in the context of Total Sales.

Furthermore, there is no reason why grouping products based on suppliers makes sense. Each product has a product category (i.e. one product category may

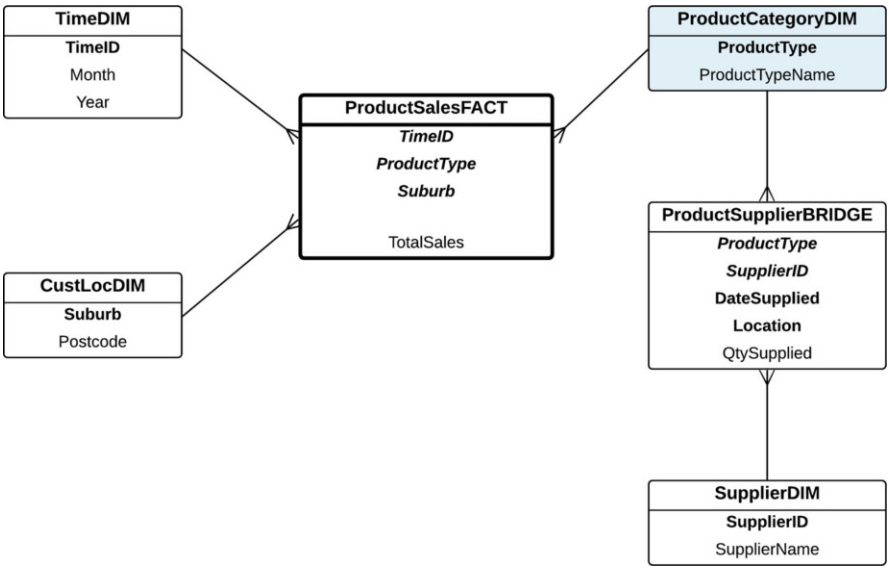

Fig. 17.9 ProductCategory Dimension

have several products). If we want to make the product more general, we should have used Product Category. Using Product Category instead of Product not only makes it more general or with a lower granularity, it also makes sense in terms of Total Sales as we can query the Total Sales for a particular Product Category. Therefore, to make Product more general, we should use Product Category instead of ProductGroupList. The revised star schema is shown in Fig. 17.9, where the ProductCategory Dimension replaces the ProductGroupList Dimension.

By using the ProductCategory Dimension where the ProductType attribute is the identifier of this dimension, the Fact and Bridge now use the ProductType attribute. The new star schema is able to answer queries such as finding the Total Sales of a particular Product Category and later drill down into which suppliers supplied that Product Category. However, the relationship between Product Category and Supplier through the Bridge may not be as strong as that between Product and Supplier in the original star schema. It is not necessary to use the Bridge and Supplier Dimension at all in the star schema with the ProductCategory Dimension. Hence, it makes more sense to have a simpler star schema with ProductCategory Dimension and without the Bridge and Supplier Dimension. This simple star schema is shown in Fig. 17.10.

Figure 17.11 shows the two levels of the Product Sales star schemas. The lower level contains the Product Dimension with the Bridge, whereas the upper level replaces the Product Dimension with the ProductCategory Dimension but without the Bridge.

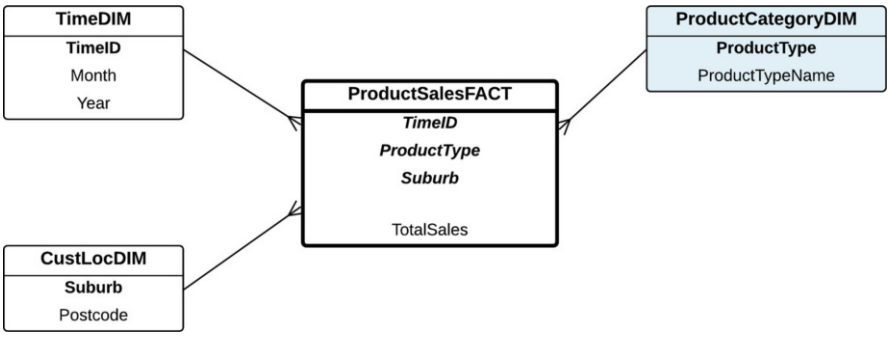

Fig. 17.10 ProductCategory Dimension without the Bridge

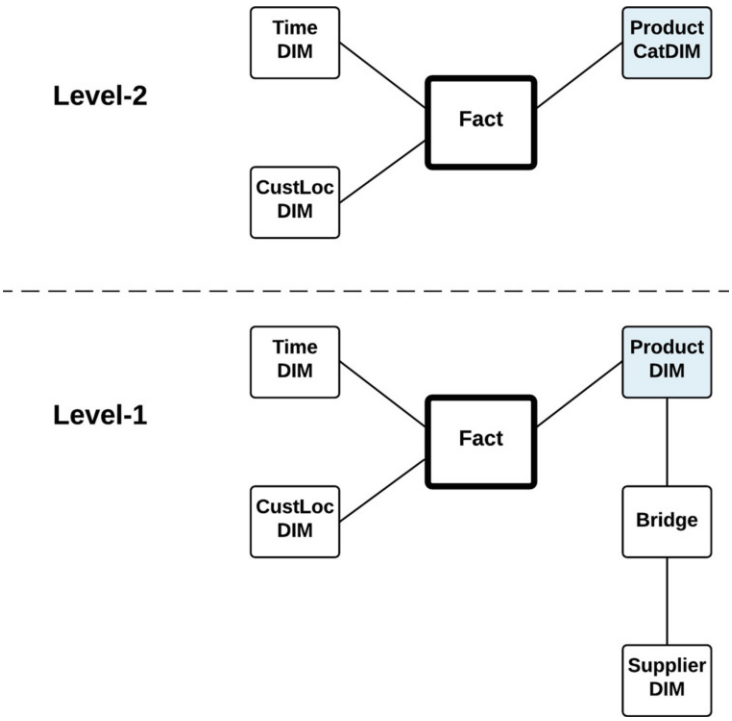

Fig. 17.11 Level of aggregations of the Product Sales star schemas

17.3 Summary

As a summary, in this chapter, we have learned how to increase the level of aggregation of a star schema when the original star schema has a Bridge Table, that is, by making a dimension more general (or lowering the level of granularity).

Using the Truck Delivery case study, the Trip Dimension is made more general, by combining several trips into one group of trips. There are two methods to do this: (*i*) the TripGroupList approach where trips are combined and (*ii*) the StoreGroupList approach where the lists of stores are combined.

The Product Sales case study offers a third option for making a star schema more general by replacing the Product Dimension with a more general dimension in the E/R diagram, namely Product Category. Not only does it make more sense to group products based on their category, it is also simpler. It is also because there is a natural hierarchy between Product and Product Category. In contrast, there is no natural hierarchy in Trip, that is, in the Trip example, a more artificial hierarchy is created using either TripGroupList or StoreGroupList.

The main lesson to make a dimension more general is to use a natural hierarchy if one exists in the operational database. Otherwise, we must resort to an artificial hierarchy.

17.4 Exercises

17.1 Using the Truck Delivery case study, the StoreGroupList approach, the StoreGroupList Dimension and the Fact are shown in Tables 17.13 and 17.14. The StoreGroupList attribute is a concatenation of stores for the same trip, which may potentially be a long list. It would be more desirable if a Surrogate Key is used as the identifier instead of StoreGroupList. In this case, the Surrogate Key is called StoreGroupKey. The new StoreGroupList Dimension table and the Fact are shown in Tables 17.15 and 17.16. Write the SQL commands to implement this StoreGroupKey approach.

17.2 Figure 17.12 is an E/R diagram of the Book Sales and Review case study. It stores information about books, the reviews each book has received as well as the entities related to the sales of books and the stores which sold the books. The star schema of the Book Sales is shown in Fig. 17.13. The star schema

Table 17.13 StoreGroupList Dimension table

StoreGroupList	WeightFactor
M1_M2_M3_M4_M8	0.20
M1_M2_M4	0.33
M1_M2	0.50
...	...

Table 17.14 Fact Table

TruckID	SeasonID	StoreGroupList	Total Delivery Cost
Truck1	Autumn	M1_M2_M3_M4_M8	894.00
Truck2	Autumn	M1_M2_M4	375.00
Truck1	Winter	M1_M2_M3_M4_M8	408.00
Truck2	Winter	M1_M2	262.50

Table 17.15 StoreGroupList Dimension table with StoreGroupKey as the Surrogate Key

StoreGroupKey	StoreGroupList	WeightFactor
1	M1_M2_M3_M4_M8	0.20
2	M1_M2_M4	0.33
3	M1_M2	0.50
...	...	...

Table 17.16 The New Fact Table with StoreGroupKey

TruckID	SeasonID	StoreGroupKey	Total Delivery Cost
Truck1	Autumn	1	894.00
Truck2	Autumn	2	375.00
Truck1	Winter	1	408.00
Truck2	Winter	3	262.50

has four dimensions: Store Dimension, Time Dimension, Category Dimension and Book Dimension. The Book Dimension has a Bridge Table connecting to the Author Dimension. In the Book Dimension, there are also WeightFactor and AuthorGroupList attributes.

If we want to make the star schema more general, we need to combine several books into one. There are two methods to do this:

1. The first method is by grouping books written by the same authors into one group of books. The star schema for this method is shown in Fig. 17.14. The Book Dimension is now replaced with the BookGroupList Dimension where ISBNGroupList becomes the new identifier, which is basically a concatenation of the ISBN of the books with the same authors.
2. The second method is by using the Category Dimension, to replace the original Book Dimension. The star schema of this method is shown in Fig. 17.15.

Compare and contrast these two approaches. It is also fair to assume that only a few authors are really prolific.

17.3 Figure 17.16 shows an E/R diagram of a Cable Television case study. The operational database stores information such as TV Programs, Channels and Plan, as well as Customer and Contract. The star schema of this Cable Television case study is shown in Fig. 17.17. The UsagePlan Dimension is connected to the Paid Channel Dimension through the Bridge. The WeightFactor and PaidChannelList attributes are also used in the UsagePlan Dimension.

Redesign the star schema to make it more general by combining several Plans into one group of Plans. Explain the two approaches, and design a star schema for

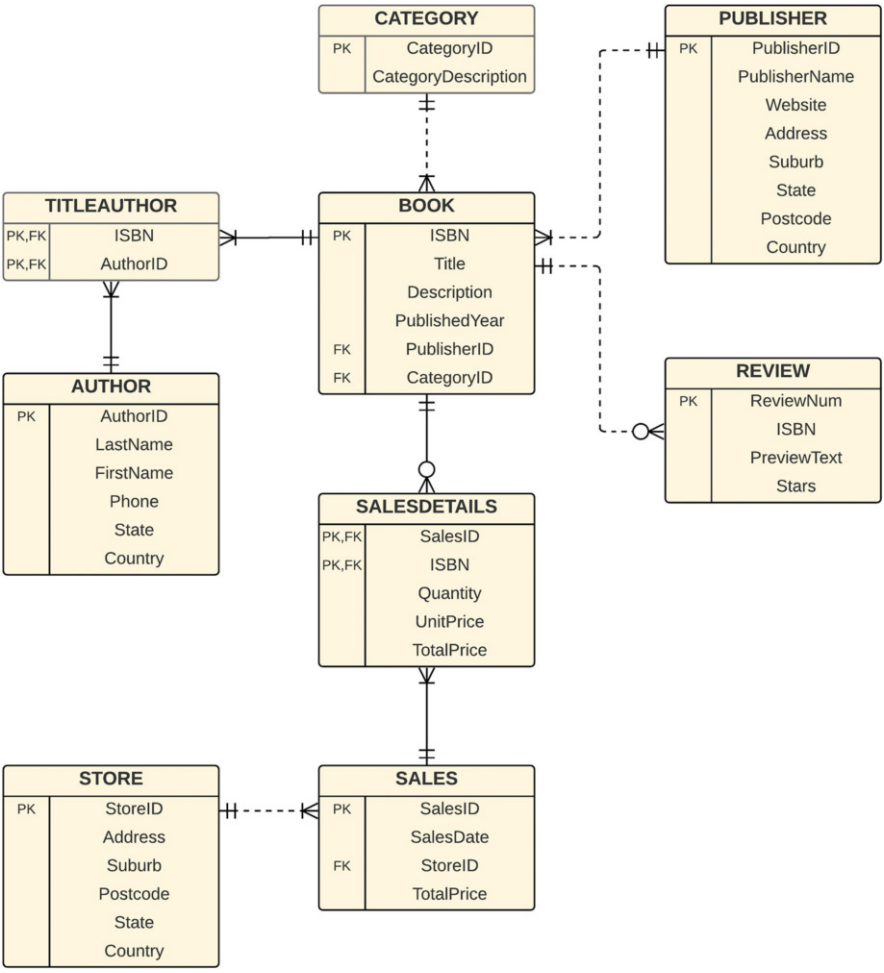

Fig. 17.12 Book Sales and Review E/R diagram

each approach. As a hint, make use of the information from the Paid Channel List, which is a `listagg` attribute.

17.4 The *Rural University of Victoria* has a number of campuses in several cities and towns in the state of Victoria. Each campus has several departments. Staff from one campus may need to travel to a different campus, for example, to teach, to attend administrative meetings or to undertake collaborative research with staff on another campus. To facilitate these travels, some departments have a fleet of cars which staff may borrow to travel between campuses in different cities. The campus also has a fleet of cars which staff may borrow if all the department cars are booked. These cars need a regular service as well as non-regular maintenance, including replacing

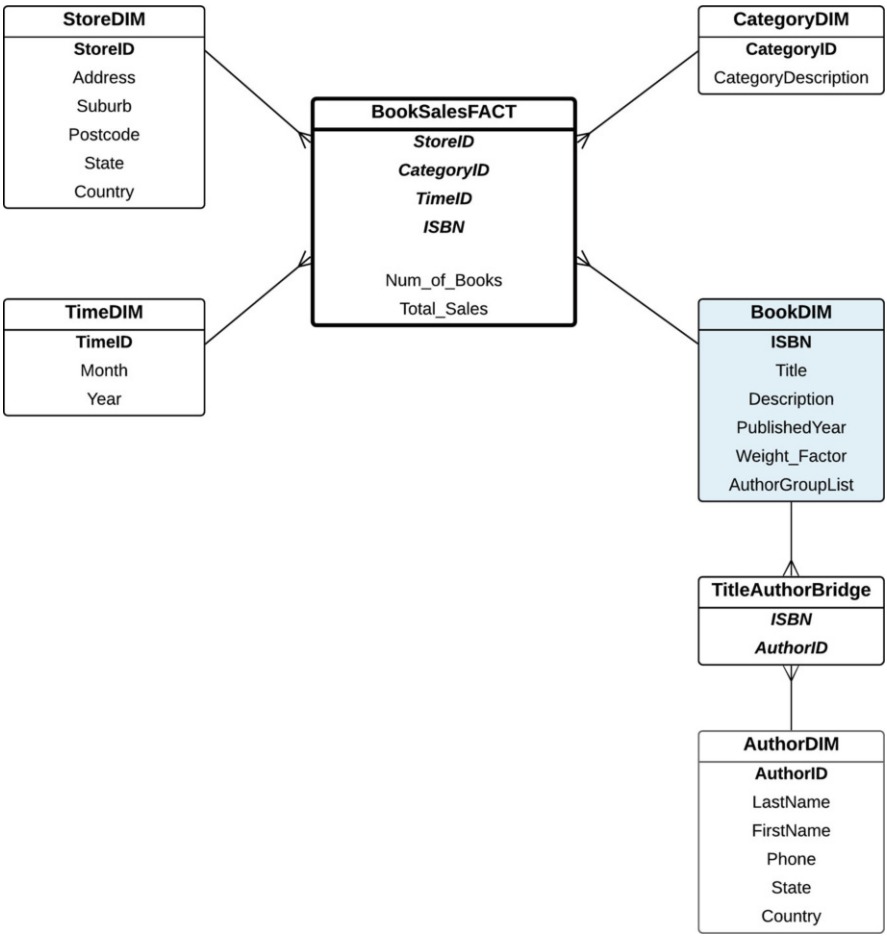

Fig. 17.13 Book Sales star schema

tyres and fixing other faults outside the regular service period. The E/R diagram of the operational database is shown in Fig. 17.18.

The star schema of this case study is shown in Fig. 17.19. The Car Dimension is connected to the User Dimension through the Bridge. The Car Dimension has also the WeightFactor and UserGroupList attributes.

We would like to redesign the star schema to make it more general, that is, by grouping several cars into one group of cars. Explain the four approaches, and design a star schema for each approach.

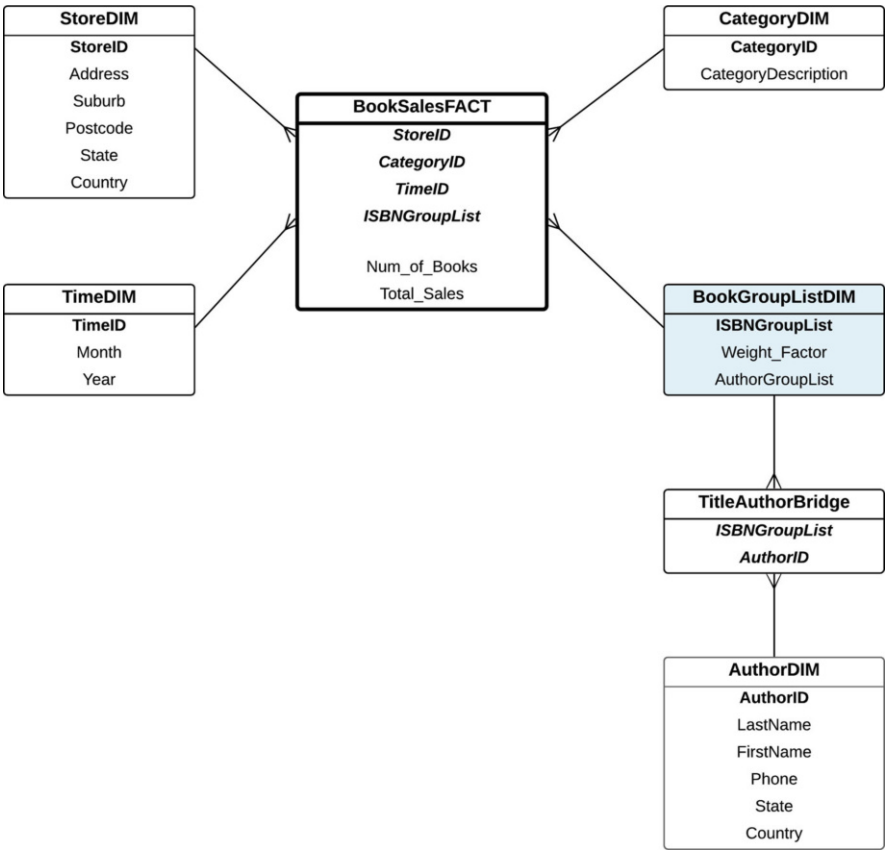

Fig. 17.14 Book Sales star schema with BookGroupList

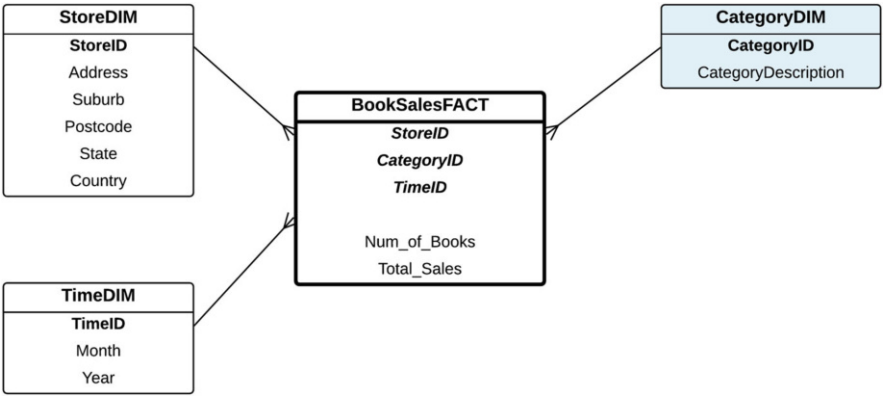

Fig. 17.15 Book Sales star schema with Category

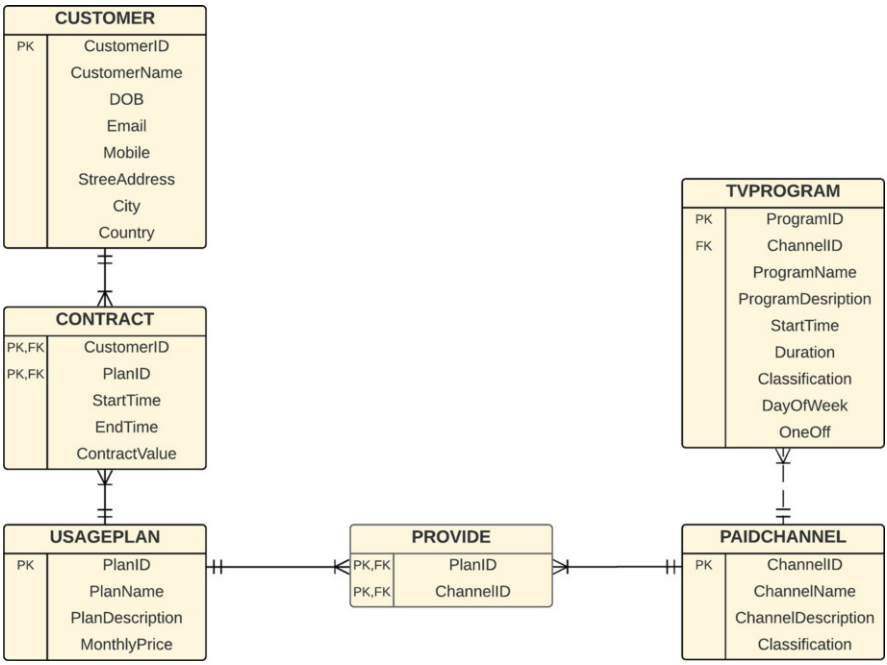

Fig. 17.16 Cable TV E/R diagram

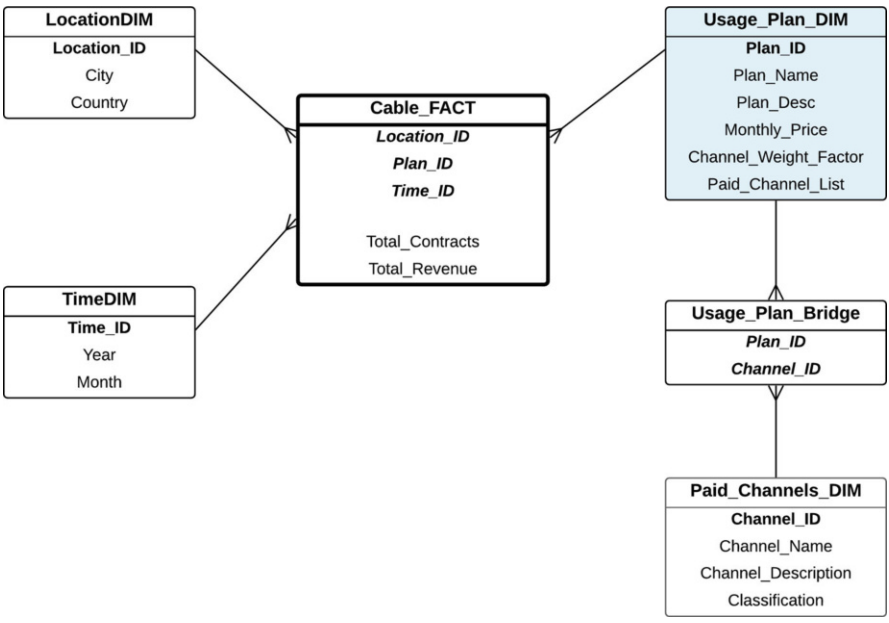

Fig. 17.17 Cable TV star schema with Bridge

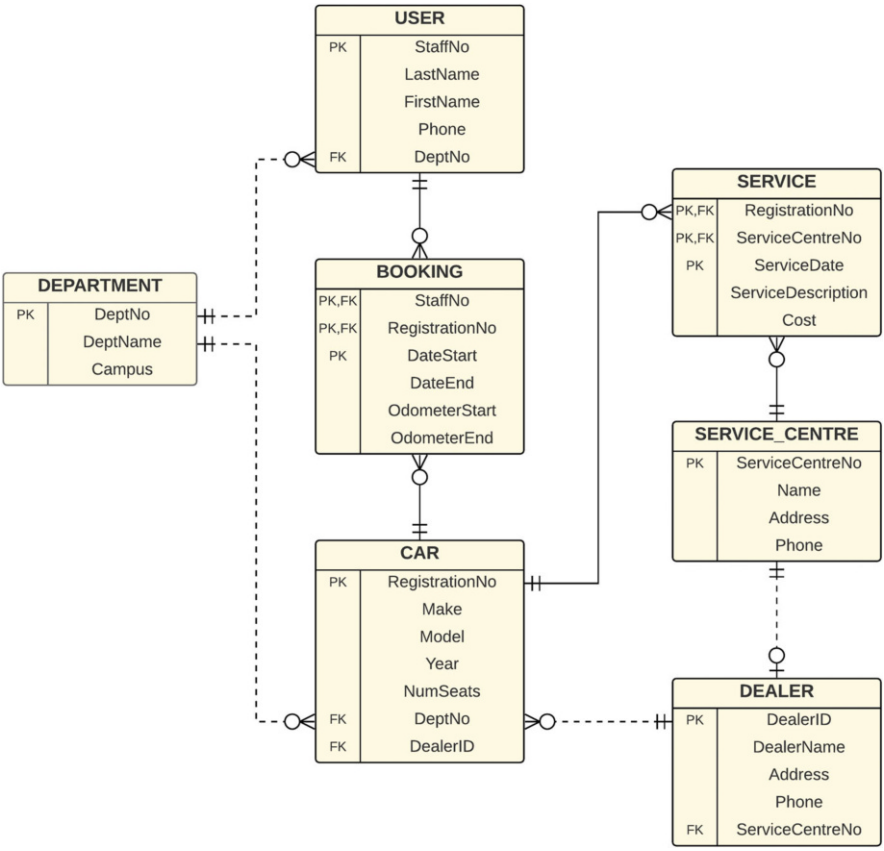

Fig. 17.18 Car Booking and Service E/R diagram

17.5 Further Readings

This book is the only book that discusses thoroughly the distinction between star schema design using higher level and lower level of aggregation. Understanding the concept of granularity in star schema design is crucial in data warehousing. This chapter drills down into more details the impact of Bridge Tables in data warehouse granularity. Since the Bridge Table is not directly connected to the Fact Table, it may raise some conflicts because of different granularity in the star schemas.

Various star schema design models have been discussed in data warehousing books, such as [1–11]. Data modelling books, such as [12–17], would also give readers a broader perspective on the differences between various levels of aggregation in data modelling.

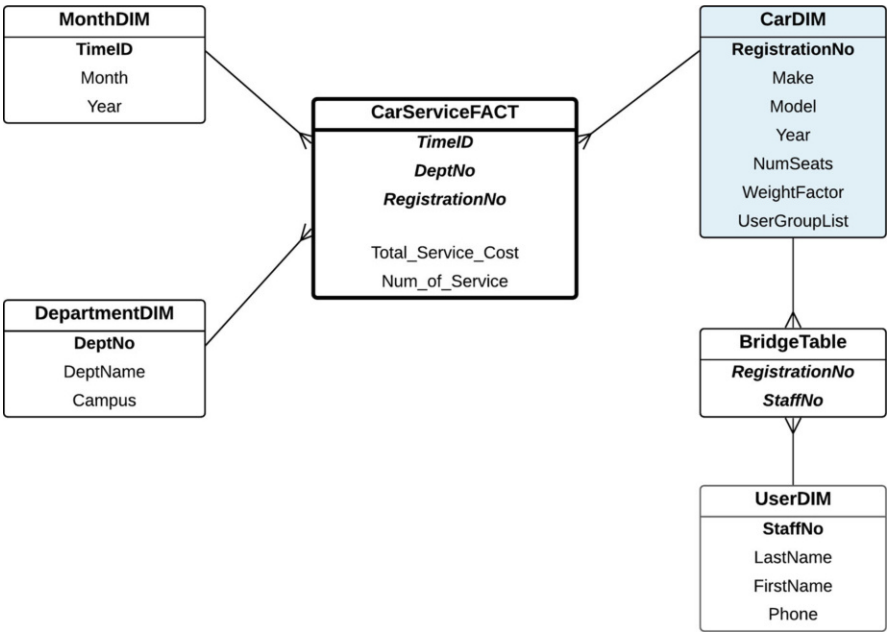

Fig. 17.19 Car Booking and Service star schema

References

1. C. Adamson, *Star Schema The Complete Reference* (McGraw-Hill Osborne Media, New York, 2010)
2. R. Laberge, *The Data Warehouse Mentor: Practical Data Warehouse and Business Intelligence Insights* (McGraw-Hill, New York, 2011)
3. M. Golfarelli, S. Rizzi, *Data Warehouse Design: Modern Principles and Methodologies* (McGraw-Hill, New York, 2009)
4. C. Adamson, *Mastering Data Warehouse Aggregates: Solutions for Star Schema Performance* (Wiley, New York, 2012)
5. P. Ponniah, *Data Warehousing Fundamentals for IT Professionals* (Wiley, New York, 2011)
6. R. Kimball, M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (Wiley, New York, 2013)
7. R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, B. Becker, *The Data Warehouse Lifecycle Toolkit* (Wiley, New York, 2011)
8. W.H. Inmon, *Building the Data Warehouse* ITPro Collection (Wiley, New York, 2005)
9. M. Jarke, *Fundamentals of Data Warehouses*, 2nd edn. (Springer, Berlin, 2003)
10. E. Malinowski, E. Zimányi, Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications. *Data-Centric Systems and Applications* (Springer, Berlin, 2008)
11. A. Vaisman, E. Zimányi, Data warehouse systems: design and implementation *Data-Centric Systems and Applications* (Springer, Berlin, 2014)
12. T.A. Halpin, T. Morgan, *Information Modeling and Relational Databases*, 2nd edn. (Morgan Kaufmann, Los Altos, 2008)
13. J.L. Harrington, Relational database design clearly explained, in *Clearly Explained Series* (Morgan Kaufmann, Los Altos, 2002)

14. M.J. Hernandez, *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*. For Mere Mortals (Pearson Education, London, 2013)
15. N.S. Umanath, R.W. Scamell, *Data Modeling and Database Design* (Cengage Learning, Boston, 2014)
16. T.J. Teorey, S.S. Lightstone, T. Nadeau, H.V. Jagadish, Database modeling and design: logical design, in *The Morgan Kaufmann Series in Data Management Systems* (Elsevier, Amsterdam, 2011)
17. G. Simsion, G. Witt, Data modeling essentials, in *The Morgan Kaufmann Series in Data Management Systems* (Elsevier, Amsterdam, 2004)