

# BT5110 Data Management and Warehousing

## Tutorial 9: Data Warehousing and Dimension Modelling

Mark Meng Huasong

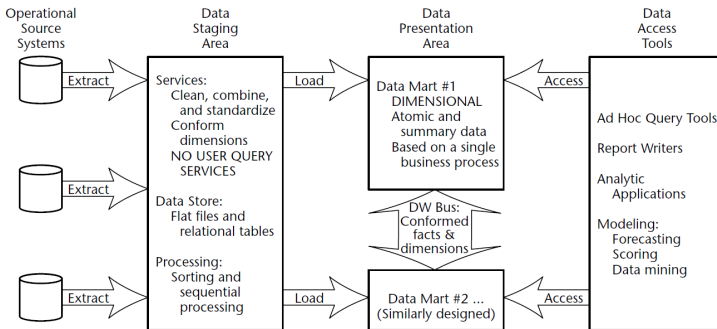
School of Computing  
National University of Singapore

1 - 5 Nov, 2021



All the materials within presentation slides are protected by copyrights.  
It is forbidden by NUS to upload these materials to the Internet.

# Quick Recap 1: Components of a Complete Data Warehouse Environment



Page 7, Kimball Chapter 1

## 1. Operational Source Systems:

Capture the transactions of the business.

Be outside the data warehouse.

Main priorities: processing performance & availability.

# Quick Recap 1 (Cont.)

## 2. Data Staging Area:

Is both a storage area & a set of processes called *Extract-transformation-load* (ETL).

Is off-limits to business users (like the kitchen of a restaurant).

Usually supported by creating a normalized database (poor understand-ability and therefore is excluded from the presentation area).

## 3. Data Presentation:

Is the area where data is organized, stored, and made available for direct querying.

Dimensional modeling is used for making database simple and understandable.

Tables are referred to as *star schemas* if the presentation area is based on a *relational database*. If the presentation area is based on multidimensional database or OLAP technology, then the data is stored in *cubes*.

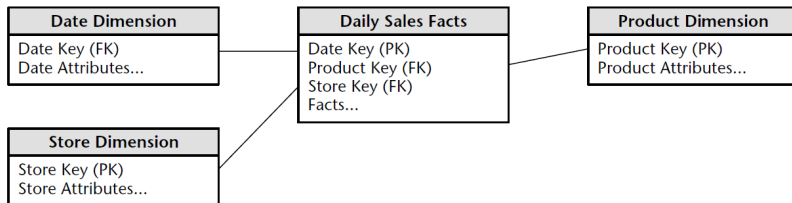
### Dimensional modeling vs. 3NF modeling:

Why 3NF modeling is for data staging area and dimensional modeling is for data presentation area? (refer to pg. 11 of *Kimball* Chapter 1).

## 4. Data Access Tools:

Are provided to business users to leverage the presentation area for analytic decision making.

## Quick Recap 2: Fact Tables and Dimension Tables



**Figure 1.4** Fact and dimension tables in a dimensional model.

Fact Tables	Dimension Tables
PK as composite of FKs	Single PK
Express a many-to-many relationship (take reference from more than 1 dimension tables)	Integral companions to a fact table
Usually make up >90% of spaces	Usually <10%, shallow in terms of rows but with many large columns
Additive numeric facts are preferred	Texture and discrete attributes are preferred
Categorised by grains (single transaction, periodic, accumulative)	Represented hierarchically in the business (e.g., product, family, brand)

# Quick Recap 3: Dimensional Modeling Steps

**1. Select the Business Process:** (e.g., the POS system for the case study.)

**2. Declare the Grain**

Identify the most granular data and proceed with that grain (e.g., an individual line item on a POS transaction)

A data warehouse almost always demands data expressed at the lowest possible grain of each dimension.

**3. Choose the Dimensions**

Grain statement determines the primary dimensionality of the fact table. Then add descriptive dimensions into the dimension table.

**4. Identify the Facts**

Derived facts (e.g., “gross profit”) are usually kept to eliminate user error. Percentage and ratios, such as “gross margin”, are non-additive and therefore should not be included in the fact table (instead we should keep the numerator and denominator in the fact table). The ratio can be calculated in a data access tool (the 4-th component).

# Quick Recap 3: Dimensional Modeling Steps

## Given an existing normalized ER diagram, how to transform it to a set of dimension models?

(In the exam generally you are given some SQL and normalization questions prior to this)

You need to know that the ER diagram is for the 2nd component of a data warehousing. Although a normalized ER diagram has many advantages (recall the “anomalies” part in the lecture of normalization), a normalized ER diagram is off-limits to business users’ querying because of its poor understandability.

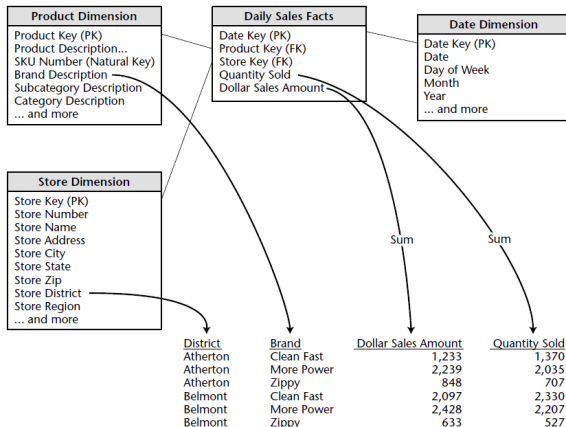
(Following content are cited from page 23, *Kimball* Chapter 1) If you already have an existing normalized ER diagram, the first step in converting it into a set of dimensional models is to separate the ER diagram into its discrete business processes and then model each one separately.

The second step is to select those many-to-many relationships in the ER diagrams that contain numeric and additive nonkey facts and designate them as fact tables.

The final step is to denormalize all the remaining tables into flat tables with singlepart keys that join directly to the fact tables. These tables become the dimension tables.

# Quick Recap 3 (Cont.)

Keep the figure below (page 24, Kimball Chapter 1) in your mind, it is helpful to know the big picture of the role of dimension modeling in the entire data warehousing story.



**Figure 1.5** Dragging and dropping dimensional attributes and facts into a simple report.

# Question 1

Answer the following questions by quoting the relevant excerpts in the chapter, recalling the relevant examples from the chapter, synthesising an answer in your own words while keeping the main keywords and proposing your own illustrating example or examples.

**Question:** What are the advantages of modeling the data warehouse around business processes rather than around organizational business departments or as a holistic organization wide data warehouse?

Discussed in Lecture (at around 00:14:30 of the Zoom recording)

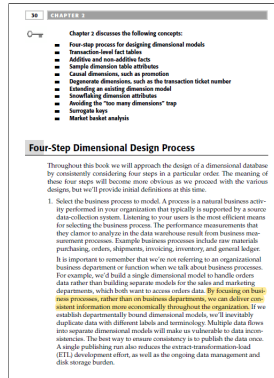


# Question 1 (Cont.)

**Solution:** In this document, we only provide the quote from Kimball but the complete answer should also contain Kimball's example, your answer and your own illustrating examples.

From page 30 of *Kimball* Chapter 2:

“By focusing on business processes, rather than on business departments, we can deliver consistent information more economically throughout the organization.”



# Question 1 (Cont.)

## **Elaboration** (with original text in page 30):

- We'd build a single dimensional model to handle orders data rather than building separate models for the sales and marketing departments, which both want to access orders data.
- ... establish departmentally bound dimensional models, we'll inevitably duplicate data with different labels and terminology. Multiple data flows into separate dimensional models will make us vulnerable to data inconsistencies.
- The best way to ensure consistency is to publish the data once.

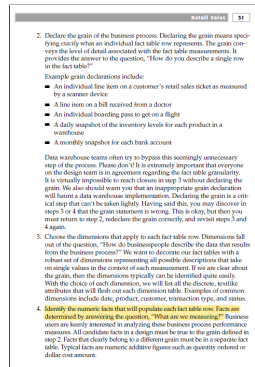
# Question 2

**Question:** What question entails the choice of the facts?

Discussed in Lecture (at around 00:31:20 of the Zoom recording)

**Solution:** From page 31 of *Kimball* Chapter 2, “Facts are determined by answering the question, “What are we measuring?”.”

Facts that clearly belong to a different grain must be in a separate fact table. Typical facts are numeric additive figures such as quantity ordered or dollar cost amount.



# Question 3

**Question:** What is an “additive fact”? Give examples and counter-examples (semi-additive, non-additive).

**Solution:**

You may refer to the page 17 to find the definition of various types of facts.

More details are given in following slides.

Dimensional Modeling primer 17

Daily Sales Fact Table	
Date Key (DK)	
Product Key (PK)	
Store Key (SK)	
Quantity Sold	
Dollar Sales Amount	

**Figure 1.2** Sample fact table.

We use the term fact to represent a business measure. We can imagine standing in the marketplace watching products being sold and writing down the quantity sold and dollar sales amount each day for each product in each store. A measurement is taken at the intersection of all the dimensions (day, product, and store). This list of dimensions defines the *grain* of the fact table and tells us what the scope of the measurement is.

⚡ A row in a fact table corresponds to a measurement. A measurement is a row in a fact table. All the measurements in a fact table must be at the same grain.

The most useful facts are numeric and additive, such as dollar sales amount. Throughout this book we will use dollars as the standard currency to make the case study examples more tangible—please bear with the authors and substitute your own local currency if it doesn't happen to be dollars.

Additivity is crucial because data warehouse applications almost never retrieve a single fact table row. Rather, they bring back hundreds, thousands, or even millions of fact rows at a time, and the most useful thing to do with so many rows is to add them up. In Figure 1.2, no matter what slice of the database the user chooses, we can add up the quantities and dollars to a valid total. We will see later in this book that there are facts that are semadditive and still others that are nonadditive. Semadditive facts can be added only along some of the dimensions, and nonadditive facts simply can't be added at all. With nonadditive facts we are forced to use counts or averages if we wish to summarize the rows or are reduced to joining out the fact rows one at a time. This would be a dull exercise in a fact table with a billion rows.

⚡ The most useful facts in a fact table are numeric and additive.

We often describe facts as continuously valued mainly as a guide for the designer to help sort out what is a fact versus a dimension attribute. The dollar sales amount fact is continuously valued in this example because it can take on virtually any value within a broad range. As observers, we have to stand

## Question 3 Cont.

**Additive facts:** Facts that can be summed up through all dimensions in the fact table.

**Semi-additive facts:** Facts that can be summed up for some of the dimensions in the fact table, but not others.

**Non-additive facts:** Facts that cannot be summed up for any of the dimensions present in the fact table.

Examples are given in following slides.

## Question 3 Cont.

Date
Store
Product
Sales_Amount

Date
Account
Current_Balance
Profit_Margin

Sales\_Amount is an additive fact, because you can sum up this fact along any of the three dimensions present in the fact table\_date, store, and product.

Current\_Balance and Profit\_Margin are the facts.

Current\_Balance is a semi-additive fact, as it makes sense to add them up for all accounts (what's the total current balance for all accounts in the bank?), but it does not make sense to add them up through time (adding up all current balances for a given account for each day of the month does not give us any useful information). Profit\_Margin is a non-additive fact, for it does not make sense to add them up for the account level or the day level.

Square footage (area of the store) is semi-additive but it is not a fact.

## Question 4

**Question:** Should calculated (derived) facts be stored?

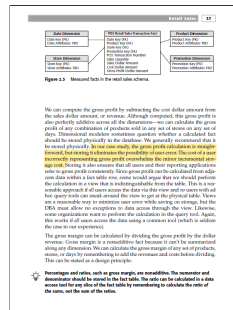
**Solution:** From page 37 of *Kimball* Chapter 2, “Dimensional modelers sometimes question whether a calculated fact should be stored physically in the database. We generally recommend that it be stored physically. In our case study, the gross profit calculation is straight-forward, but storing it **eliminates the possibility of user error.**”.

Calculated fact eliminates error. They most likely do not consume significantly more space. They can therefore be calculated at the time of staging. They could also be defined using views (the fact table becomes a view of background base tables). The views could be materialised.

# Question 4 (Cont.)

## Elaboration:

Calculated fact eliminate error. They most likely do not consume significantly more space. They can therefore be calculated at the time of staging. They could also be defined using views (the fact table becomes a view of background base tables). The views could be materialised. More details are given in following slides.





## Question 5

**Question:** What is the “grain” or “granularity”? How to determine the appropriate grain?.

Discussed in Lecture (at around 00:49:30 of the Zoom recording)

**Solution:** From page 31 of *Kimball* Chapter 2, “Declaring the grain means specifying exactly what an individual fact table row represents. The grain conveys the level of detail associated with the fact table measurements.”

### Question 5 (Cont.)

**Elaboration:**

“Grain” or “granularity” is the level of details of the data stored in the fact table. The finer the grain, the more rows in the fact table. The coarser the grain the more aggregated the data in the fact table.

The grain is determined by specifying exactly what an individual fact table row represents. It answers the question “How do you describe a single row in the fact table?” A grain declaration could be:

- An individual line item on a customer's retail sales ticket as measured by a scanner device;
- A line item on a bill received from a doctor;
- A daily snapshot of the inventory levels for each product in a warehouse;
- A monthly snapshot for each bank account.

	Initial sales	21
2. Decline the grant of the business tax process. Declaring the grant exactly what an individual tax return represents. The grant covers the legal benefit associated with the tax table measurements. It is the answer to the question, "How do you declare a single rate in the tax table?"		
Example grants declarations include:		
<ul style="list-style-type: none"> <li>• An individual line item on a customer's initial sales sheet as measured by a scanner device</li> <li>• A line item on a bill received from a dealer</li> <li>• An individual boarding pass to get on a flight</li> <li>• A daily snapshot of the inventory levels for each product in a warehouse</li> <li>• A monthly snapshot for each bank account</li> </ul>		
Data warehouse tables often try to bypass this seemingly unnecessary step of the process. Please don't! It is extremely important that everyone on the design team is in agreement regarding the table grants because it is virtually impossible to reach claims in step 2 without declaring the grant. We also should never say that an inappropriate grant declaration is a data warehouse implementation. Declaring the grant is a critical step that can't be taken lightly. Having said this, you may discover in step 4 that the grant statement is wrong. This is okay, but you must return to step 2, redeclare the grants correctly, and revisit steps 3 and 4 again.		
3. Create the dimensions that apply to each tax table row. Dimensions fall into the categories of "How do businesspeople describe the data that results from the business process?" They want to describe our tax tables with a subset set of dimensions representing all possible facts that can be used on single values in the context of each dimension. If we are clear about this, then the dimensions are created. The dimensions are created with the choice of each dimension, we will list all the dimensions, terrible attributes that will fill out each dimension table. Examples of common dimensions include data, product, location, transaction type, and name.		
4. Identify the numeric facts that will populate each tax table row. Facts are determined by answering the question, "What are we measuring?" Business is always interested in a number of facts. These facts are performance measures. All candidate facts is a design team to figure out the grants defined in the table that clearly belong to the facts. The facts must be in a separate fact table. Typical facts are numeric additive values such as quantity ordered or dollar cost amount.		

# Question 6

**Question:** Should there be null values in the fact table? Why?


**Solution:**

From page 49 of *Kimball* Chapter 2:

“You **must** avoid null keys in the fact table. A proper design includes a row in the corresponding dimension table to identify that the dimension is not applicable to the measurement.”

RETAIL SALES 49

Typically, many sales transaction line items involve products that are not being promoted. We will need to include a row in the promotion dimension, with its own unique key, to identify “No Promotion in Effect” and avoid a null promotion key in the fact table. Referential integrity is violated if we put a null in a fact table column declared as a foreign key to a dimension table. In addition to the referential integrity alarms, null keys are the source of great confusion to our users because they can’t join on null keys.

 You must avoid null keys in the fact table. A proper design includes a row in the corresponding dimension table to identify that the dimension is not applicable to the measurement.

**Promotion Coverage Factless Fact Table**

Regardless of the handling of the promotion dimension, there is one important question that cannot be answered by our retail sales schema: What products were on promotion but did not sell? The sales fact table only records the SKUs that actually sold. There are no fact table rows with zero facts for SKUs that didn’t sell because doing so would enlarge the fact table unnecessarily. In the relational world, a second promotion coverage or event fact table is needed to help answer the question concerning what didn’t happen. The promotion coverage fact table keys would be date, product, store, and promotion in our case study. This obviously looks similar to the sales fact table we just designed; however, the grain would be significantly different. In the case of the promotion coverage fact table, we’d find one row in the fact table for each product on promotion in a store each day (or week, since many retail promotions are a week in duration) regardless of whether the product sold or not. The coverage fact table allows us to see the relationship between the keys as defined by a promotion, independent of other events, such as actual product sales. We refer to it as a *factless* fact table because it has no measurement metrics; it merely captures the relationship between the involved keys. To determine what products were on promotion but didn’t sell requires a two-step process. First, we’d query the promotion coverage table to determine the universe of products that were on promotion on a given day. We’d then determine what products sold from the POS sales fact table. The answer to our original question is the set difference between these two lists of products. Stay tuned to Chapter 12 for more complex coverage of factless fact tables, as it illustrates the promotion coverage table and provides the set difference SQL. If you’re working with data in a multidimensional online analytical processing (OLAP) cube environment, it is often easier to answer the question regarding what didn’t sell because the cube typically contains explicit cells for nonbehavior.

## Question 6 (Cont.)

The fact table is composed of attributes corresponding to surrogate keys referencing dimensions and to facts corresponding to measurements. The former cannot be null.

Special situations such as that of a description not being available can be handled by referencing to an explicit description of the situation in the dimension table.

Measurements, however, can be null, if their value is unknown or does not exist. This is however to be handled with care when asking queries, given the not always appropriate handling of null values in SQL.

# Question 7

**Question:** What questions entails the choice of the dimensions?

**Solution:** From page 31 of *Kimball* Chapter 2, “Dimensions fall out of the question, “How do business people describe the data that results from the business process?”.”

We want to decorate our fact tables with a robust set of dimensions representing all possible descriptions that take on single values in the context of each measurement. If we are clear about the grain, then the dimensions typically can be identified quite easily. With the choice of each dimension, we will list all the discrete, textlike attributes that will flesh out each dimension table. Examples of common dimensions include date, product, customer, transaction type, and status.

Detail Table 31

2. Declare the grain of the business process. (Declaring the grain means specifying exactly what an individual fact table row represents. The grain captures the level of detail associated with the fact table measurement. It provides the answer to the question, “How do you describe a single row in the fact table?”)

Example grain declarations include:

- An individual line item on a customer's retail sales ticket as measured by a scanner device
- A line item on a bill received from a doctor
- An individual boarding pass to get on a flight
- A daily snapshot of the inventory levels for each product in a warehouse
- A monthly snapshot for each bank account

Data warehouse teams often try to bypass this seemingly unnecessary step of the process. Some don't. It is commonly important that everyone on the design team is in agreement regarding the fact table granularity. It is virtually impossible to reach consensus in step 3 without declaring the grain. We also should warn you that an inappropriate grain declaration will leave a data warehouse implementation. (Declaring the grain is a critical step that can't be taken lightly. Having said this, you may discover in steps 3 or 4 that the grain statement is wrong. This is okay, but then you must return to step 2, redetermine the grain correctly, and restart steps 3 and 4 again.)

3. Choose the dimension that apply to each fact table row. (Dimensions fall out of the question, “How do businesspeople describe the data that results from the business process?” We want to decorate our fact tables with a robust set of dimensions representing all possible descriptions that take on single values in the context of each measurement. If we are clear about the grain, then the dimension typically can be identified quite easily. With the choice of each dimension, we will list all the discrete, textlike attributes that will flesh out each dimension table. Examples of common dimensions include date, product, customer, transaction type, and status.)

4. Identify the numeric facts that will populate each fact table row. Facts are discovered by answering the question, “What are we measuring?” Business users are heavily interested in analyzing these business process performance measures. All candidate facts in a design must be true in the grain defined in step 2. Facts that clearly belong to a different grain must be in a separate fact table. Typical facts are numeric additive figures such as quantity ordered or dollar cost amount.

# Question 8

**Question:** What is the role of descriptive attributes in dimensions?

**Solution:**

They are entry point for queries. There can be many and should be as many as needed.

From page 21 of *Kimball* Chapter 1, “the fact table consisting of numeric measurements is joined to a set of dimension tables filled with descriptive attributes.”

Dimensional Modeling Drivers 31

We strive to minimize the use of codes in our dimension tables by replacing them with more verbose textual attributes. We understand that you may have already trained the users to make sense of operational codes, but going forward, we'd like to minimize their reliance on arbitrary notes attached to their computer monitors for code translations. We want to have standard decoders for the operational codes available as dimension attributes so that the labeling on data warehouse queries and reports is consistent. We don't want to encourage decoders based in our reporting applications, where inconsistency is inevitable. Sometimes operational codes or identifiers have legitimate business significance to users or are required to communicate back to the operational world. In these cases, the codes should appear as explicit dimension attributes, in addition to the corresponding user-friendly textual descriptions. We have identified operational, natural keys in the dimension figures, as appropriate, throughout this book.

Operational codes often have intelligence embedded in them. For example, the first two digits may identify the line of business, whereas the next two digits may identify the global region. Rather than forcing users to interrogate or filter on the operational code, we pull out the embedded meanings and present them to users as separate dimension attributes that can be filtered, grouped, or opened on easily.

Dimension tables often represent hierarchical relationships in the business. In our sample product dimension table, products roll up into brands and then into categories. For each row in the product dimension, we store the brand and category descriptions associated with each product. We realize that the hierarchical descriptive information is stored redundantly, but we do so in the spirit of ease of use and query performance. We store our natural keys in some only the brand code in the product dimension and create a separate brand lookup table. This would be called a *model*. Dimension tables typically are highly denormalized. They are usually quite small (less than 10 percent of the total data storage requirements). Since dimension tables typically are geometrically smaller than fact tables, improving average efficiency by normalizing or snowflaking has virtually no impact on the overall database size. We almost always trade off dimension table space for simplicity and accessibility.

**Bringing Together Facts and Dimensions**

Now that we understand fact and dimension tables, let's bring the two building blocks together in a dimensional model. We illustrated Figure 14, the fact table consisting of numeric measurements is joined to a set of dimension tables filled with descriptive attributes. This characteristic model structure is often called a *star join scheme*. This term dates back to the earliest days of relational databases.

## Question 8 (Cont.)

Each attribute is a rich source for constraining and constructing row headers. If we want to drill down, we can drag virtually any other attribute and we automatically drill down to this next level of detail.

Drilling down is nothing more than adding row headers from the dimension tables. Drilling up is removing row headers.

A robust and complete set of dimension attributes translates into user capabilities for robust and complete analysis.

## Question 9

**Question:** Should there be null values in the dimension tables? Why?

**Solution:** Null values **should** be **avoided** in the dimension tables. Attributes of the dimension tables are used for aggregation. Group do not behave smoothly with null value



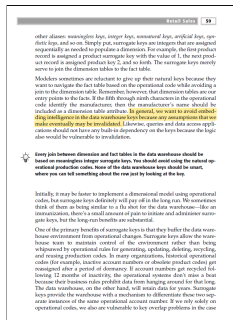
# Question 10

**Question:** What is Kimball's argument in favour of surrogate keys?

**Solution:**

From page 59 of *Kimball* Chapter 2:

"In general, we want to avoid embedding intelligence in the data warehouse keys because any assumptions that we make eventually may be invalidated."



## Question 10 (Cont.)

Surrogate keys = “fake” keys, such as an incremental ID.

Surrogate keys provide the warehouse with a mechanism to differentiate two separate instances of the same operational number that wouldn't be possible if the operational number was used directly.

If we rely solely on operational codes, we also are vulnerable to key overlap problems in the case of an acquisition or consolidation of data. Performance advantages: Can be a small integer.

The surrogate keys should merely serve to join the dimension tables to the fact table.

# Question 11

**Question:** What can be said about the relationship between grain and dimensions?

**Solution:** The finer the grain the more dimensions.

The grain of the dimensional model is the finest level of detail that is implied when the fact and dimension tables are joined.

For example, the granularity of a dimensional model that consists of the dimensions Date, Store, and Product is “product sold in store by day”.

For example, a dimension such as Date (with Year and Quarter hierarchies) has a granularity at the quarter level but does not have information for individual days or months.

For any further question, please feel free to email me:

huasong.meng@u.nus.edu

Cases in the extra practice are contributed by our students.

Copyright 2021 Mark H. Meng. All rights reserved.