# Data Modeling with Snowflake: A Practical Guide to Accelerating Snowflake Development Using Universal Data Modeling Techniques

## Part 2: Applied Modeling from Idea to Deployment

* This part focuses on the ***practical* application of data modeling techniques across different stages of the database design process**
* You will learn **how to put conceptual modeling *into practice* by gaining a solid understanding of the data entities, relationships, + business rules that define the underlying data structure**
* You will also learn **how to enrich these details + add nuance using a logical model that cannot be captured in a physical database**
* This part also delves into the critical concept of **database normalization**, essential **for minimizing data redundancy** and **ensuring accurate + consistent data storage**
* After covering database naming + structure + exploring the best practices for creating intuitive, meaningful, + scalable database schemas, you’ll learn how to **put physical modeling into practice while optimizing database performance**

### Chapter 7: Putting Conceptual Modeling into Practice

* **Conceptual database modeling** **= a high-level approach to designing a database that focuses on capturing business entities + their relationships**
* Allows designers to **develop a deeper understanding of the data**, making it easier to identify potential issues or inconsistencies in the design
* Can also **make the database more flexible + adaptable to future changes** and **make it easier to understand + use for people who are unfamiliar w/ it**
* As a result, **conceptual database modeling can help make a database more effective + efficient at supporting the needs of an organization**
* We will **generate a conceptual model from scratch using Kimball’s dimensional modeling (DM) technique**
* This approach **brings the data team together w/ business experts to ensure database designs reflect the reality of business processes *from inception***, thereby saving costly rework down the line
* However, **conceptual models** are not *solely* used in initiating *new* designs 🡪 they **carry all the same benefits for *existing* physical schemas**
* We will also explain a simple method for adapting a physical design to a conceptual model and generating an easy-to-read diagram that can be used by business + technical teams alike
* Main topics:
* Discovering **DM** and its applications
* Constructing a **bus matrix** using Kimball’s four-step method
* Creating a **conceptual model** using the bus matrix
* **Reverse engineering** a conceptual model **from a physical schema**

#### Embarking on Conceptual Design

* Out of all modeling types, **conceptual captures + displays the *least* amount of detail**
* Makes **conceptual modeling ideal for getting acquainted w/ a database landscape at a high-level and for designing one from scratch**
* **Designing data models** = an art honed over **many iterations**, *but where do you begin if you are new to modeling?*

##### Dimensional Modeling

* Early 2000s 🡪 Ralph Kimball + Margy Ross published the groundbreaking book The Data Warehouse Toolkit (whose latest edition fittingly carries the subtitle, The Definitive Guide to Dimensional Modeling), which has persisted for decades as the authoritative blueprint for constructing database designs
* Many terms, concepts, + techniques described in later chapters of this book trace their origins to it
* To be clear, Kimball’s approach is *NOT* the *only* way to go about creating a conceptual model
* The agile-based **Business Event Analysis and Modeling (BEAM) method** (see “*Further reading”*) and other techniques are also worth exploring
* However, **Kimball methodology is widely used + universally recognized + will serve as a springboard for our modeling journey**

##### Understanding Dimensional Modeling

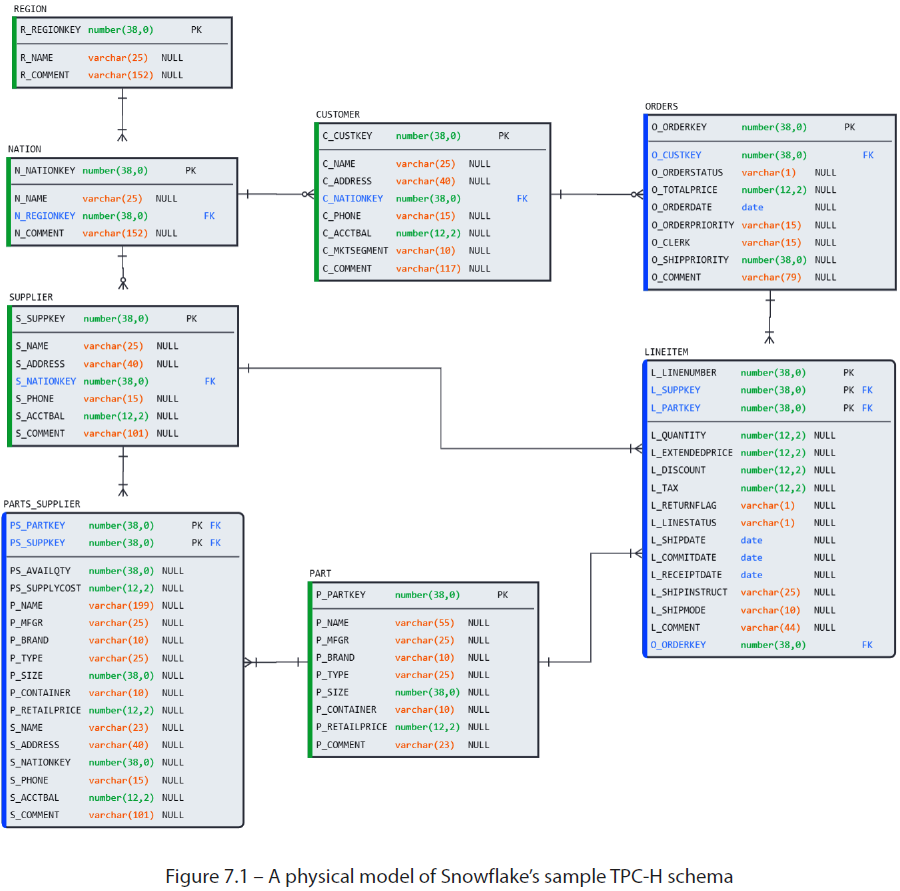
* **Dimensional modeling (DM) = a longstanding technique for making databases simple by aligning them to business processes** (a theme that should now be familiar to you)
* As simple as that sounds, **DM covers many wide-ranging topics, from business requirement workshops to types of fact + dimension tables, to industry-specific modeling best practices**
* This chapter will focus on the **early stages of DM** + discuss how data teams can engage w/ business experts to uncover the **dimensions** + **business processes** that **lay the groundwork for a conceptual model**
* However, one point needs to be addressed before getting started

##### Setting the Record Straight on Dimensional Modeling

* Critics of DM have argued that platforms such as Snowflake, which provides cheap + easily-scalable computing resources, have made the effort + rigor involved in DM obsolete
* **Nonsense, since a variable-spend platform such as Snowflake can get very expensive if left unchecked**
* **Grounding designs in time-tested frameworks is a good way to keep costs under control**
* Another frequent argument is that **analytics engineering** through tools such as **dbt** has quickened development to such a pace that allows you to build + fix a design, if need be, in less time than what DM calls for
* **While analytics engineering *does* accelerate the delivery cycle, not having a solid grounding in the fundamentals only enables you to make mistakes faster**
* Ex: **dbt snapshots****use Kimball’s Type-2 slowly changing dimensions (SCD) under the hood and require a good understanding of their fundamentals to use them effectively**
* The 3rd criticism of the Kimball methodology goes something like “*Kimball is no longer relevant because* **Data Vault** *or <some other modeling> is easier to scale*”
* **Ensemble methodologies**, such as Data Vault 2.0 (See Chapter 17) **are indeed highly scalable and effective modeling frameworks**
* However, **they don’t suit *ALL* scenarios, and still require knowing + implementing DM concepts**
* Ex: A **satellite** is *also* a Type-2 SCD
* Finally, despite having debunked main arguments against DM, it remains to be said that **this book is *NOT* advocating for DM to be used as the *principal* framework for database or DW design**
* **Rather, since DM objects + language permeate database design, it is essential to know the rules + fundamentals and to understand when + where to deviate**
* With this in mind, let’s see how we can use DM principles to kick-start a conceptual model or validate an existing physical design through the same means

##### Starting a Conceptual Model in 4 Easy Steps

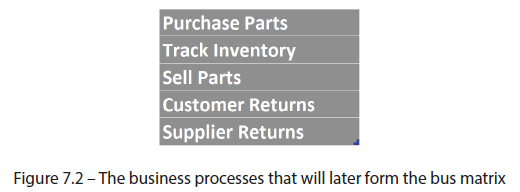
* This section will review the **4-step DM method for creating a conceptual model** by **identifying the business processes + dimensions involved**
* Recall that **the aim of conceptual modeling + database modeling as a whole is to align as closely as possible to the *reality* of the business operations**
* For this reason, ***this* stage of the modeling process will require the *most* collaboration from the business teams to identify the conceptual building blocks for future technical developments**
* To witness this process in action, we use the tables in the SNOWFLAKE\_SAMPLE\_DATA.TPCH\_SF1 schema to represent a real-world retail organization
* To understand what we’re working toward, explore the sample tables in the Snowflake UI/see the following diagram resulting from this exercise:



* Before reviewing each step individually, here are the **4 steps that Kimball suggests for starting DM at a conceptual level**:
* Define the **business process** 🡪 Declare the **grain** 🡪 Identify **dimensions** 🡪 Identify **facts**
* **Modeling begins by organizing workshops w/ business domain experts + department stakeholders to determine the answers to these 4 points**
* This is **where the *reality* of the business is discussed, then formally defined, + then agreed upon by the stakeholders so that its relevant data can be modeled + captured**
* Everyone must understand that although the **concepts are being discussed at a high level**, they’ll **later be used to determine design decisions regarding database details, business rules, and their governing logic** (details that can be extremely difficult to alter in later stages of development)
* The **outcomes + decisions established during the collaborative workshops should be considered binding contracts between the business + technical teams**, who will later carry them forward to implementation
* Before making its way to an ERD**,** the **information captured during these discovery sessions is captured in** what Kimball calls the **bus matrix**
* In a bus matrix, **each row represents a business process or subject area**
* And **each column represents a dimension or fact table**
* Their **intersections form the relationships or transactions w/in an organization**

###### Define the Business Process

* As the 1st step in the process, **begin by working w/ domain experts to identify the existing business processes** in the TPC-H organization
* While the answer to “what do you do?” for a given organization might sound simple (such as *selling merchandise* or *taking reservations*), **the reality is usually much more complex when additional activities, such as procurement, inventory, promotions, + cancelations are considered**
* Recall that the **primary goal of modeling is to capture the *relevant* metrics + properties that we want to track, measure, + analyze**
* Think of **business processes** as **the transactions that occur *naturally* as the business operates**
* **In DM, transactions are referred to as** **facts**, and **tables that record them and (nearly always) pair them w/ metrics are called fact tables**
* **Starting w/ the company’s *primary* function** (ex: selling, registering, or manufacturing), ***extend* the discussion to the *supporting* functions** (ex: sourcing inventory, stocking warehouses, + contracting w/ suppliers), **until every department has been heard from + has relevant operational processes cataloged**
* **The resulting list generated from these meetings forms the *left* side of the bus matrix (the rows)**
* In the TPC-H example, the list may look something like this:



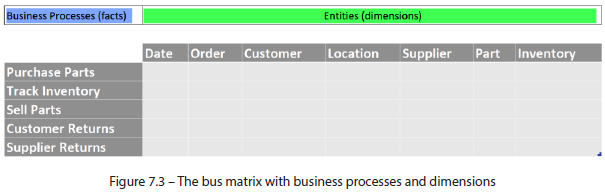
* **Once the business processes have been identified, it’s time to go into more detail + determine their *lowest* grain of analysis**

###### Determine the Grain

* In the previous step, **we listed the operational transactions that make up our business processes**
* **These transactions will become fact tables** in the database that we are designing
* *But what does a row for each transaction represent?*
* Ex: The primary function of our business may be to sell parts, but do we record + analyze such a transaction at the *part* or the *customer* level? Or can it be *both* or *either*, depending on the business unit?
* The answer to this question, we **must now determine the grain** = **the lowest level of detail required for meaningful analysis of each business process**
* Generally, the **recommended grain is the *lowest* level of detail captured by the business processes themselves** (the **atomic grain**)
* **Grounding our design at the atomic grain ensures it can accommodate any subsequent rollups and aggregations to higher levels of analysis**
* With this approach, **changes in business requirements will not require changing the model b/c the model already contains the maximum granularity recorded by the business processes**
* When selecting the grain, we **must also ensure that *different* grains are not mixed in the *same* fact table**
* Ex: A sales fact table maintained at the grain of daily transactions must NOT include monthly or yearly totals
* **Doing so would result in costly + inefficient database UPDATE operations on the table every time a transaction was recorded**
* **Conversations about grain will naturally bring to light the entities involved in the operational functions**: customers, suppliers, + parts
* This information will help us determine the *other* half of the bus matrix

###### Determine the Dimensions

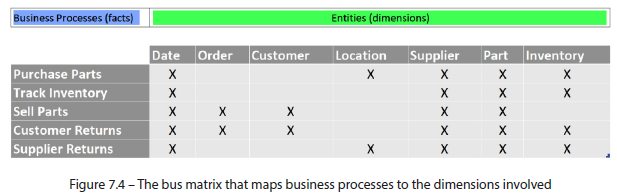
* The **entities that came to light during the conversations surrounding business processes help give *context* to the nature of our business**
* They **help answer questions such as *who*, *what*, + *where***and **will often contain attributes that offer further descriptive context**
* In our example, the conversation may have started w/ sales + moved through the logical paces of the part being sold, the supplier who provided the part, + the warehouse used to store the inventory of available parts, *until all the relevant entities were identified*
* **Entities** (which **often correspond to singular nouns in spoken language**) **will become dimension tables** in the database we’re building + **contain attributes** that **need to be defined + modeled**
* Having identified several key dimensions, such as customer, order, + part, we can draw the other axis of the bus matrix (the columns)
* The following screenshot shows the resulting grid when business processes are placed next to their component dimensions:



* **Notice that the date dimension is the first one named**
* **Facts (such as sales, streams, + logins) have at least one relevant *time* dimension that needs to be recorded** (although **fact-less fact tables** also exist + are covered later)
* **When all dimensions have been identified, it’s time to finalize the bus matrix by nailing down the facts**

###### Identify the Facts

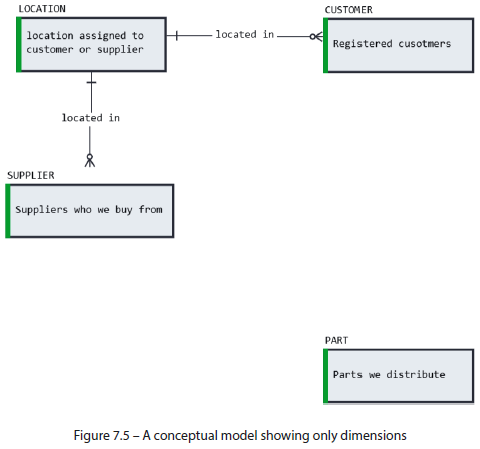
* **DM defines facts as tables containing transactions *among business dimensions* that’re paired w/ relevant metrics**
* The **metrics record information is *pertinent to the grain of the fact table*, which comprises relevant dimensions**
* This **relationship can be mapped directly in the bus matrix by putting an *X* at the intersection of business processes + their required dimensions**
* In our example, the finished bus matrix may look something like this:



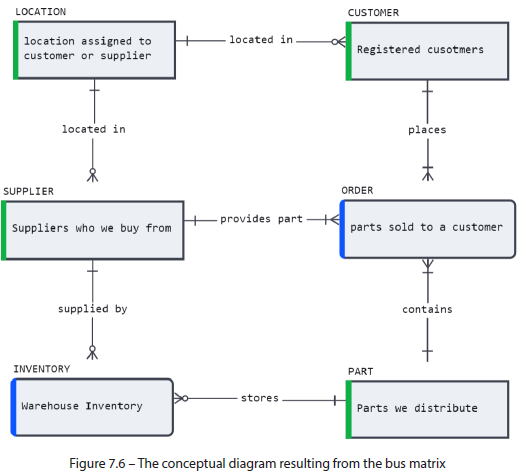
* Using the resulting matrix, we can revisit the earlier question of sales **grain** + see whether we’ve answered it correctly
* **While the problem may seem technical, the answer is always business driven**
* Perhaps the atomic grain of a sale consists of its date + time, details of the customer + product being purchased, + the metrics surrounding the transaction, such as price, quantity, + discount
* Our business teams may want to track total sales + cancelations at the customer level while reviewing product margins + discounts at the part level
* **We can construct tables to accommodate business needs by capturing this atomic grain in the design**
* **Performance + maintenance decisions at further stages of the modeling process may result in multiple rollup tables being created from the granular atomic facts**
* Still, **the analysis at the conceptual level is sufficient to accommodate any such decisions**
* Having laid the groundwork for uncovering the facts + dimensions, let’s **visualize the bus matrix in a conceptual model**

##### From Bus Matrix to Conceptual Mode

* **Now that we have a bus matrix worked out + agreed upon** with the business teams, we **can begin to transfer the details onto a conceptual diagram**
* **1st step in the conceptual model process = to create the dimensions as entities**
* *Usually*, the most relevant **attributes** would have also been identified as part of the business workshops + could also be included in the model
* But, to split the exercise into cleanly separated steps, this will be done in the next chapter, where we discuss **logical modeling**
* Here is what the diagram would look like with *only* dimensions included:



* **W/ the entities in place, we can now create the fact tables + connect them to the dimension tables using the relationships + business rules defined in the earlier workshops**
* The model below now contains all facts (blue) + dimensions (green) listed in the bus matrix, as shown below:



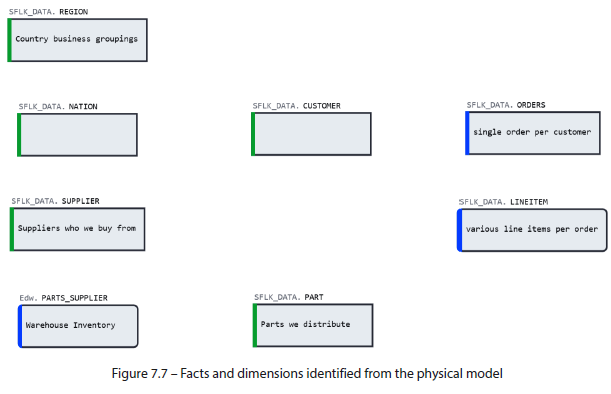
* Notice that while 5 business processes are listed in the matrix, only 2 fact tables have been modeled in the diagram
* **Often, the same fact table can include enough details to cover *multiple* business processes**
* Ex: Customer sales and returns, b/c they occur at the same grain.
* **By asking a few business-oriented questions, we’ve seen how to determine the grain of the various operations of a company and identify the related facts + dimensions, putting it all together into the bus matrix**
* A **conceptual diagram can be easily generated + used as a foundation for later steps in the modeling process using the information in the bus matrix**
* But **just as a conceptual model can be used to initiate database design by formalizing existing business processes, it can also work in *reverse* to help sense-check existing physical designs**

#### Modeling in Reverse

* Real-world enterprise operational systems + DW’s can reach heights of great complexity, especially if they have evolved over many years + absorbed untold numbers of dubious design decisions
* Even the standing members of the data team may not be able to explain every detail, let alone a new hire or a business analyst
* Fortunately, **conceptual modeling can help simplify + visualize *existing* database landscapes as quickly as it does when starting new ones**
* **Database modeling tools allow users to take schema definitions as DDL + ingest them to generate a corresponding diagram** in a **reverse engineering** process
* Since all the table structures + constraints are already defined in the database, generating an ERD + visualizing the database landscape can be done in seconds
* **Because reverse engineering uses Snowflake object DDL, it produces *physical* models**
* **As *physical models contain the highest level of detail of all modeling types*, they can be easily abstracted into conceptual**
* While **modeling tools can automate most of the steps of going from physical to conceptual**, the task **can also be performed by hand w/out much effort by following the 4-step method in reverse**
* i.e., Embarking on an exercise of going from *what is* to *what ought to be*
* **The latter part cannot be known from the database but verified w/ the business teams through workshops + discussions**
* This exercise aims to generate a best-guess conceptual model from *what is* so that it can facilitate those discussions

##### 1) Identify the Facts and Dimensions

* **Start by taking the *existing* tables in the physical schema and classify them into facts + dimensions using previously discussed rules of thumb**
* **Dimension tables will typically be named using singular nouns + contain descriptive attributes but *not* quantitative measurements**
* **Facts are usually named using transitive verbs, including measures that quantify each transaction**
* In our example, the result of this exercise would look something like the following diagram:



* Now that the **entities have been mapped**, we **must identify the relationships**

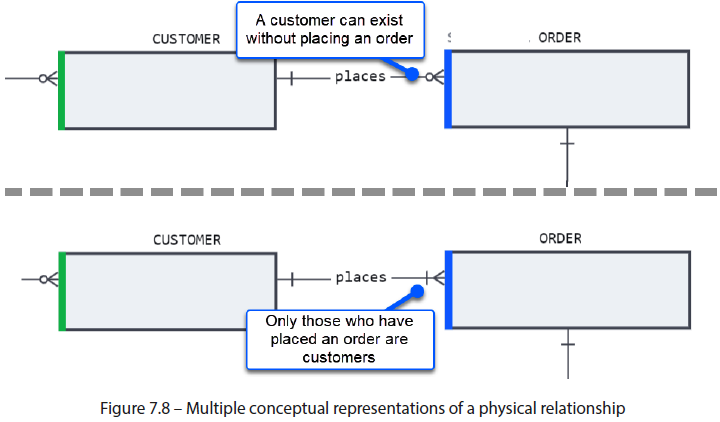
##### 2) Establish the Relationships

* **Identifying the relationships between tables can be swift or arduous *depending on whether standards + best practices have been observed in the physical schema***
* Ex: If PKs + FKs have been declared, modeling tools will use this information to automatically establish + visualize all the relationships in a schema
* ***Absent* these constraints, we can rely on *column naming conventions* to intuit the relationships**
* Starting w/ the dimensions, look for columns that resemble the table name + include something that hints at their being a PK (keywords such as *id*, *key*, or *code*)
* Ex: A *NATION* dimension contains a column called N\_NATIONKEY and is, therefore, likely to be the PK
* *Test* the assertion with GROUP BY HAVING, which should return no values > 1 if the column is a PK, and the underlying data is clean
* The following query should return no results if N\_NATIONKEY is the PK for NATION:
* SELECT N\_NATIONKEY, COUNT(\*) AS CNT FROM NATION

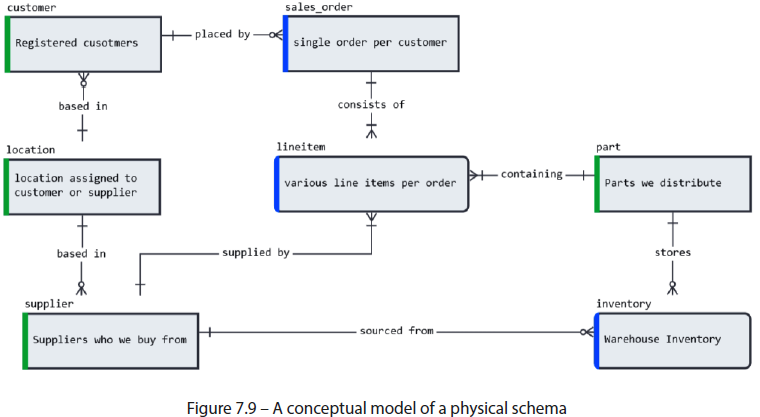
GROUP BY 1

HAVING CNT > 1

* **Remember that even the data cannot *conclusively* prove what the PK should be, it can only point to a highly likely candidate**
* ***Only the business users can conclusively confirm the grain of a business dimension***
* **Once PK candidates have been proposed for dimensions, look for instances of the PK columns in facts and other dimension tables**
* In DM**, columns w/ the same structure + meaning across multiple tables are known as** **conformed dimensions**
* Ex: An ID column in the CUSTOMER table is a conformed dimension of CUSTOMER\_ID in SALES if both columns share the same data type + values
* **When individual attributes w/in a dimension are consistent across multiple tables, they are considered conformed attributes**
* Ex: STAR\_RATING may be a conformed attribute in both PRODUCT and SUPPORT\_SURVEY
* **Consistent naming of columns across tables is immensely helpful in this type of analysis**
* **Modeling tools rely on the naming of *conformed* attributes to automatically propose FK relationships**
* **Once relationships have been identified, attempt to establish cardinality and optionality details**
* **While the table constraints give some clues about the nature of the relationship, *only the business teams can confirm the reality* (although the data can be an indicator)**
* In the following example, the physical constraints can have multiple conceptual interpretations:



* **Once we have established all the entities + relationships and made an informed guess as to their cardinality, we have completed the *draft* of a conceptual model**, which might look something like the following diagram:



* **Using the conceptual model**, which provides a clear visual representation of the existing database design, we can **open the dialogue with the various domain teams responsible for each business process + their respective dimensions to *validate* what has been built**

##### Propose and Validate the Business Processes

* Even though we’re have a relatively small schema, **classifying + identifying tables + relationships has *not* been easy**
* We’ve **generated a simplified vision of the business model using a conceptual diagram that must now be validated w/ the business teams + product owners**
* Recall that **the same fact table can capture multiple business processes + identifying them cannot be done w/out input from the business teams**
* **Based on the input of business owners + domain experts, the conceptual model can be adjusted and refined to reflect the *true reality* of business operations**
* The **conceptual review may even result in changes to the physical model if the business and database models have deviated over time**
* Whatever the scenario, afterwards we **now have a clear + accurate conceptual database model that fully aligns w/ the business it supports**
* The conceptual model can help analysts + new hires quickly get acquainted w/ the business
* After having put in the work to validate it, **the conceptual model should be maintained going forward by repeating the process described in this chapter**

#### Summary

* In this chapter, we’ve seen how a **conceptual model** + its **accompanying diagram make it easy to visualize an organizational business model** **+ validate it with domain experts** against the intended business operations that it is meant to support
* To get started in conceptual modeling, we used **Kimball’s DM methodology**, used for decades to guide the design of database + DW architectures
* **Kimball uses a 4-step method to initiate discussions between the data team + business teams, as well as domain experts to identify the business processes that an organization engages in**
* **Once business processes have been identified, we determine their grain/lowest level of detail pertinent to our business**
* **Describing the grain will also help us discover the core dimensions + facts of our business**
* Plotting dimensions + business processes on a chart, known as **the bus matrix**, produces an elegant cross-section of our business + lets us transfer its details easily to a conceptual diagram
* **It is not only *new* designs that benefit from the aid of conceptual models**
* A conceptual model can also be **used to visualize + validate a physical database**
* Modeling tools can automatically generate physical diagrams from existing DDL through **reverse engineering**
* Otherwise, modelers can **reverse the 4-step Kimball method** to arrive at a conceptual model starting from a physical database.
* **Depending on its initial design or changes that have occurred over time, a physical database may deviate from the actual business model of the organization**
* **Conceptual diagrams are an important tool in opening the conversation w/ business teams to align the physical + business models by giving non-technical users a simple way to understand what has been built**
* However, **collaboration between the data + business teams does not stop at the conceptual model**
* We will expand on what we have built by adding functional details + preparing our model for physical deployment

#### Further Reading

* Kimball, Ralph, and Margy Ross. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. John Wiley and Sons, 2013.
* Kimball, Ralph, et al. The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence Remastered Collection. John Wiley and Sons, 2015.
* Corr, Lawrence, and Jim Stagnitto. Agile Data Warehouse Design: Collaborative Dimensional Modeling, From Whiteboard to Star Schema. DecisionOne Consulting, 2011