DATA WAREHOUSES - part3

1. **Introduction to the DWH discipline**
2. **Brief history** (Inmon &Linstedt,2015) **and Data Architecture** (Kimball&Ross,2016)
3. **Dimensional Modeling Fundamentals** (Kimball&Ross,2016)
4. **Technical Architecture Considerations** (Kimball&Ross,2016)
5. **Extract Transform Load and Data Quality** (Kimball&Ross,2016)
6. **DWH Lifecycle** (Kimball&Ross,2013)
7. **Trends in the evolution of DWH:**

   Extended RDBMS Architecture (Kimball&Ross,2013)**.**

   Pushing into the Future (Reeves,2009)**.**

   DWH 1.0 vs. 2.0 (Krishnan,2013)

Assessment

Type of activity	10.1 Assessment criteria	10.2 Assessment methods	10.3 Share of final grade
Developing support components for a DWH application prototype	Real-world application, complexity, validity and originality	Six face-to-face lab presentations of the homework about: the design of DWHcubes & DM models implemented, the support MDX & DMX queries and the code execution behind .NET forms in weeks No.: 3, 5, 7, 9, 11 and 13.	40% (6 * 6.66%)
Theoretical presentations during lecture hours (see those 14 themes on domains at section 8.1, pp.2)	Format, consistent pro-or-cons arguments, originality of comments and conclusions	Theoretical presentation and responses to questions	20%
Theoretical exam	Knowledge about DWH theory, real-world scenarios and personal implementations	Final theoretical test with at least three open questions	40%

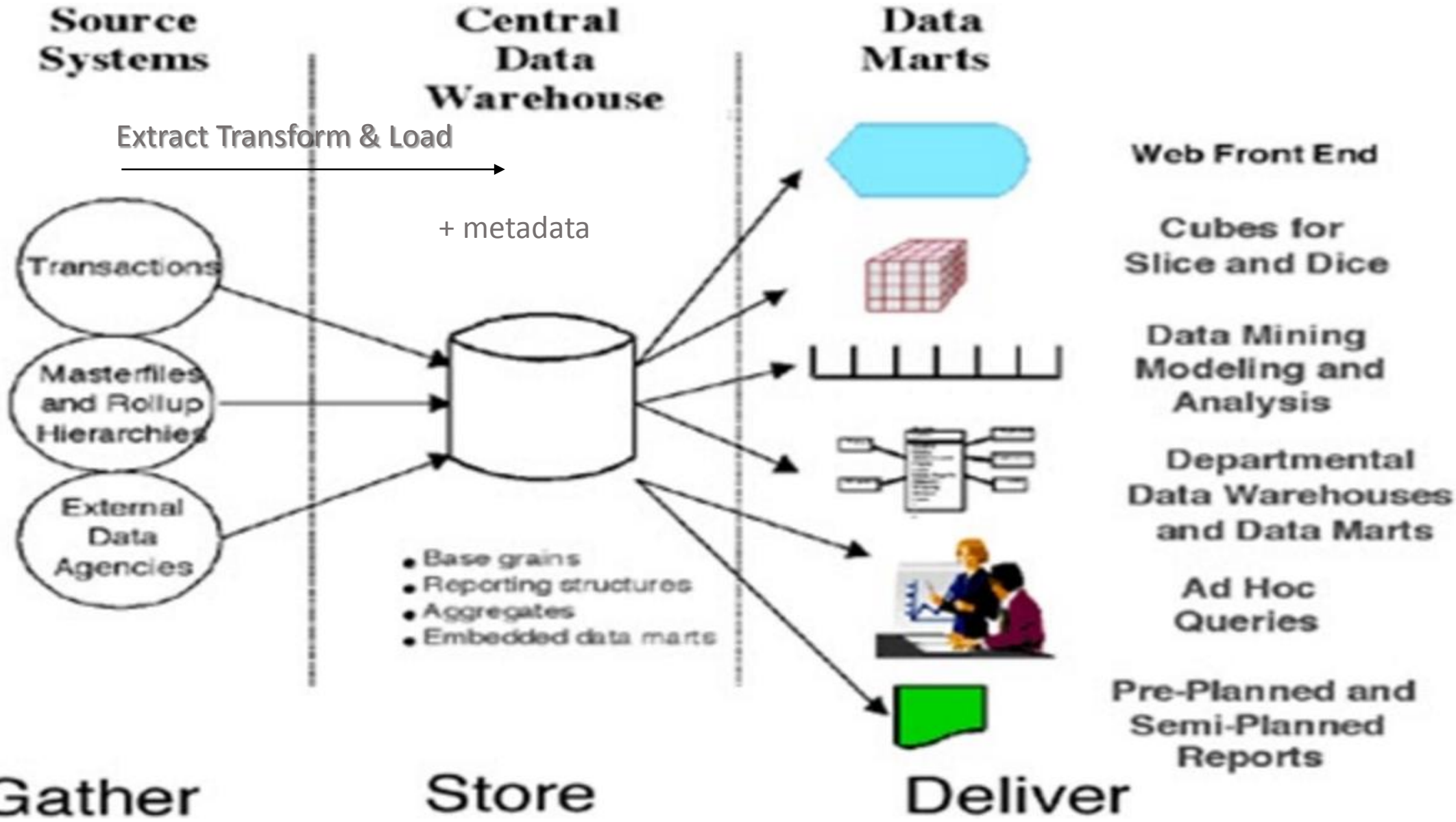
10.6 Minimum performance standard

- Design and implement a DWH cube, a DM model, two interfaces to programmatically connect to and query them by using MDX and DMX queries, alternative examples (user mode) with Microsoft Excel add-ins;
- Each master student must create, present and answer questions for a part of those at least 10 specific presentations (14 themes on domains - section 8.1, pp.2) during lecture hours;
- The average grade for all those six face-to-face lab presentations (section 8.2, pp.3) of the homework is grater than or equal to 5;
- The grade for the final theoretical test must be greater than or equal to 5.

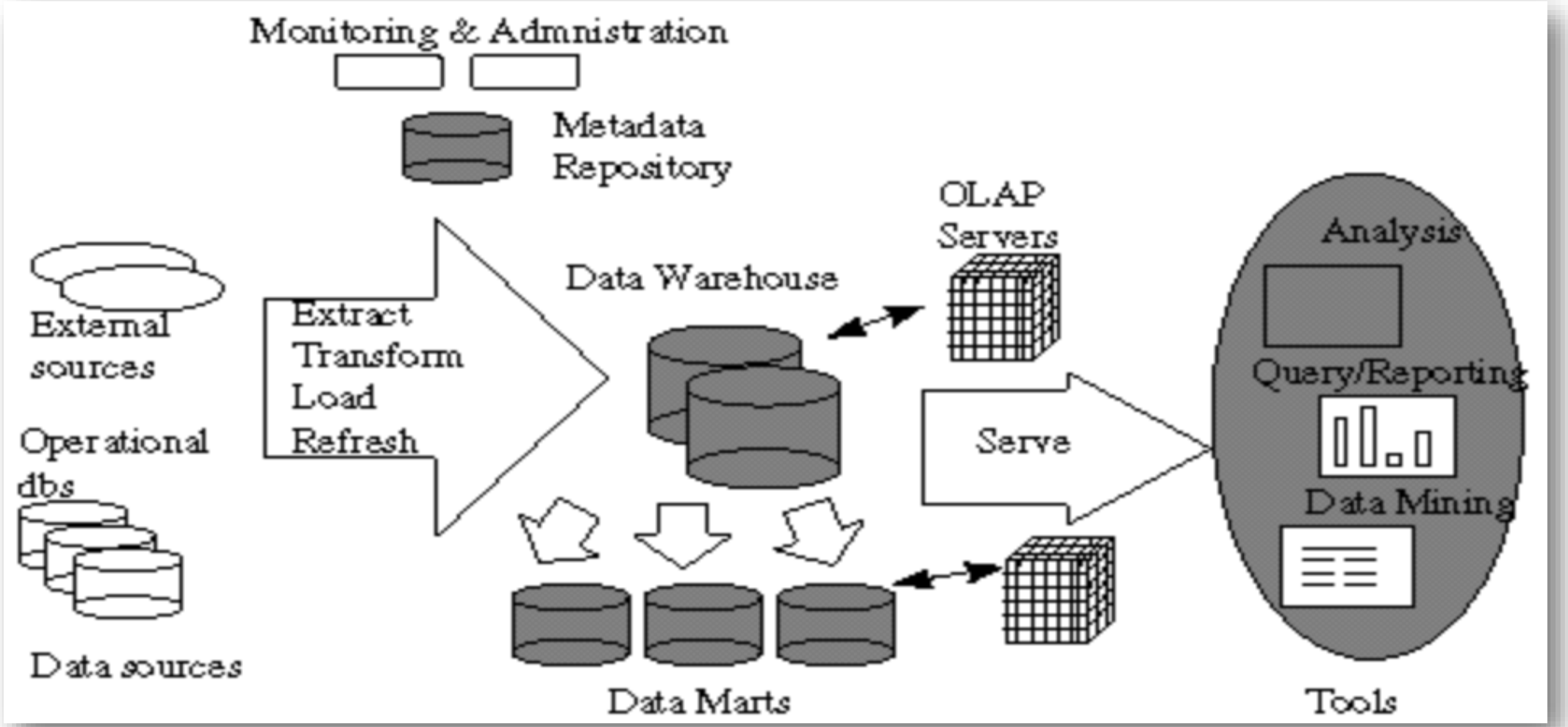
Planning those at least 10 presentations (course hours, 20% in final grades) of master students (teams) about:
(1.) retail sales, (2.) inventory, (3.) procurement, (4.) order management, (5.) accounting, (6.) CRM, (7.) HRM,
(8.) financial services, (9.) telecommunications,
(10.) transportation, (11.) education, (12.) healthcare,
(13.) e-commerce, (14.) insurance
details: **Kimball&Ross,2013**.
by **e-mail**: dan.homocianu@gmail.com
(**subject**: *DWH course pres. theme TITLE.. on DATE..*
content: *team: MEMBER1, MEMBER2, MEMBER3*)
Each master student (team member) must present a part of his team's support presentation for at least 5-10 minutes
and answer questions.

DWH – overview

- *Data mart* has been replaced with *business process dimensional model*, *business process subject area*, or just *subject area*, depending on the context. (Kimball&Ross,2016)



DWH – overview



http://redbook.cs.berkeley.edu/redbook3/lec28.html

Dimensional modeling - Business Processes & Common Dimensions

(Kimball Group)

	COMMON DIMENSIONS						
BUSINESS PROCESSES	Date	Product	Warehouse	Store	Promotion	Customer	Employee
Issue Purchase Orders	X	X	X				
Receive Warehouse Deliveries	X	X	X				X
Warehouse Inventory	X	X	X				
Receive Store Deliveries	X	X	X	X			X
Store Inventory	X	X		X			
Retail Sales	X	X		X	X	X	X
Retail Sales Forecast	X	X		X			
Retail Promotion Tracking	X	X		X	X		
Customer Returns	X	X		X	X	X	X
Returns to Vendor	X	X		X			X
Frequent Shopper Sign-Ups	X			X		X	X

Business Process / Event	Time	Customer	Service	Rate Category	Local Svc Provider	Calling Party	Called Party	Long Dist Provider	Internal Organization	Employee	Location	Equipment Type	Supplier	Item Shipped	Account Status
Customer Billing	X	X	X	X	X			X			X				X
Service Orders	X	X	X		X			X	X	X	X	X			X
Trouble Reports	X	X	X		X	X		X	X	X	X	X	X	X	X
Yellow Page Ads	X	X		X		X			X	X	X				X
Customer Inquiries	X	X	X	X	X	X		X	X	X	X				X
Promotions & Communication	X	X	X	X	X	X		X	X	X	X	X	X	X	X
Billing Call Detail	X	X	X	X	X	X	X	X	X		X	X	X	X	X
Network Call Detail	X	X	X	X	X	X	X	X	X		X	X	X	X	X
Customer Inventory	X	X	X	X	X			X	X		X	X	X	X	X
Network Inventory	X		X						X	X	X	X	X	X	
Real Estate	X								X	X	X	X			
Labor & Payroll	X								X	X	X				
Computer Charges	X	X	X		X			X	X	X	X	X	X	X	
Purchase Orders	X								X	X	X	X	X	X	
Supplier Deliveries	X								X	X	X	X	X	X	

Dimensional modeling

Kimball &Ross,2016

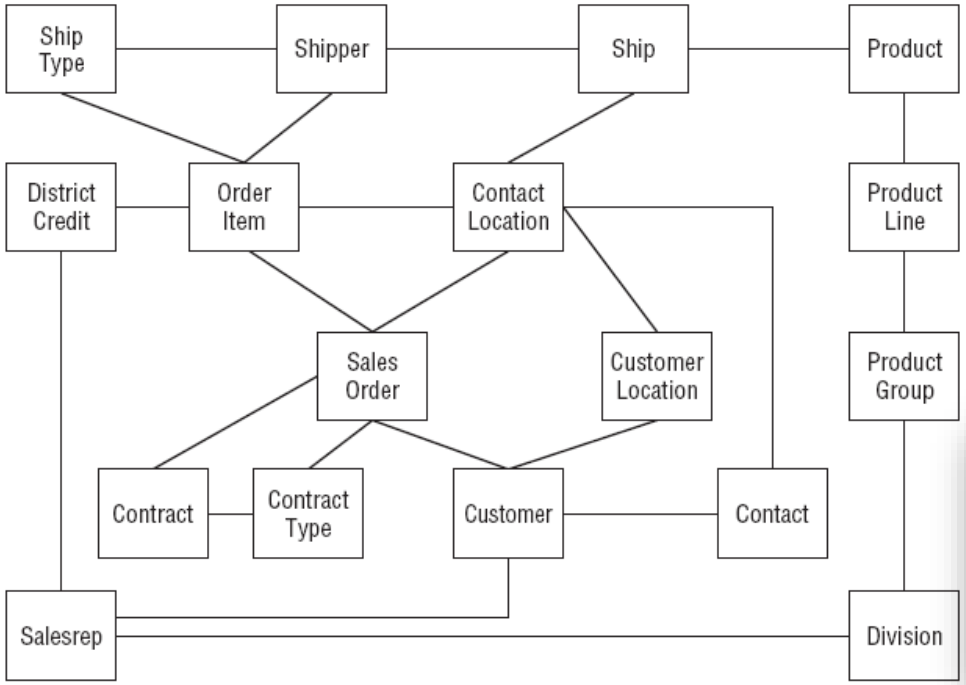


Figure 5-3: A high level 3NF model.

The 3NF modeling technique is a discipline used to illuminate the microscopic relationships among data elements. The highest art form of 3NF modeling is to remove all redundancy in the data. This is immensely beneficial to transaction processing because transactions are made very simple and deterministic. The transaction of updating a customer's address may devolve to a single record lookup in a customer address master table. This lookup is controlled by a customer address key, which defines uniqueness of the customer address record and allows an indexed lookup that is extremely fast. It is safe to say that the success of transaction processing in relational databases is mostly due to the discipline of 3NF modeling.

However, in our zeal to make transaction processing efficient, we have lost sight of our original, most important goal. We have created databases that cannot be queried! Even our simple orders example creates a database of dozens of normalized tables linked together by a bewildering spider web of joins, as illustrated in Figure 5-3.

This situation is not just an annoyance, it is a showstopper:

- Business users cannot understand, navigate, or remember a 3NF model. There is no graphical user interface (GUI) that takes a general 3NF model and makes it usable by users.
- Software cannot usefully query a general 3NF model. Cost-based optimizers that attempt to do this are notorious for making the wrong choices, with disastrous consequences for performance.
- Use of the 3NF modeling technique defeats the basic allure of data warehousing, namely intuitive and high performance retrieval of data.

Dimensional modeling

Kimball &Ross,2016

Normalized modeling is a powerful technique for designing transaction processing systems in relational environments. The normalization of physical data structures has greatly contributed to the phenomenal success of getting large amounts of data into relational databases. However, models normalized to third normal form do not contribute to the users' ability to query the data. I recommend a different technique, called dimensional modeling (DM), to structure data for querying. Now that you have succeeded in getting the data into your operational databases, it is time to get the data out.

Figure 5-1: A dimensional model for grocery store sales.

Star schema

Star schema

Dimensional modeling is a design discipline that straddles the formal relational model and the engineering realities of text and number data. Compared to third normal form entity-relationship modeling, it's less rigorous (allowing the designer more discretion in organizing the tables), but more practical because it accommodates database complexity and improves performance. Dimensional modeling has an extensive portfolio of techniques for handling real-world situations.

Dimensional modeling – Fact tables & Dimension tables

Kimball &Ross,2016

Star schema

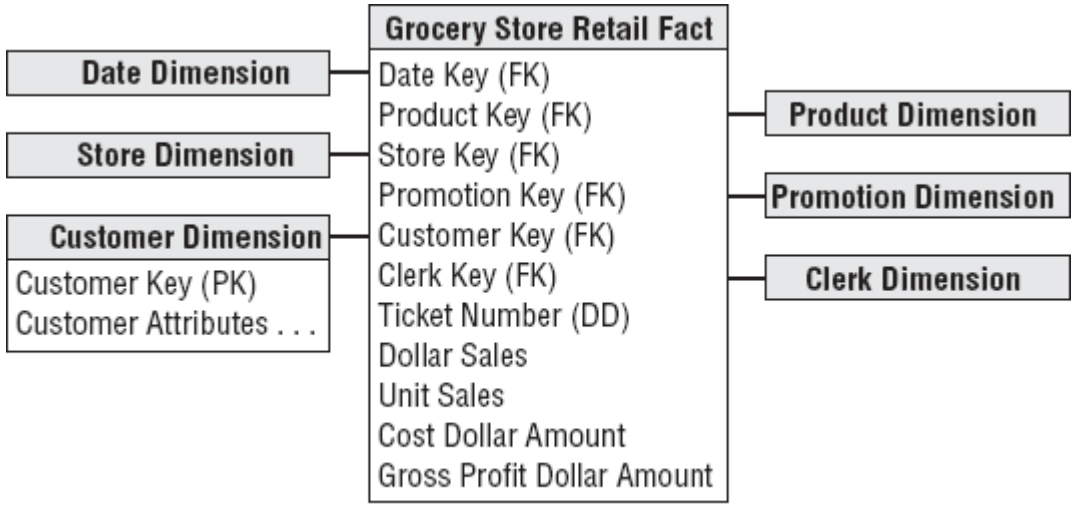


Figure 6-1: A dimensional model for grocery store sales.

Measurements and Context

Dimensional modeling begins by dividing the world into measurements and context. Measurements are usually numeric and taken repeatedly. Numeric measurements are *facts*. Facts are always surrounded by mostly textual context that's true at the moment the fact is recorded. Facts are very specific, well-defined numeric attributes.

Dimensional Keys

If the facts are truly measures taken repeatedly, you find that fact tables always create a characteristic many-to-many relationship among the dimensions. Many customers buy many products in many stores at many times.

Therefore, you logically model measurements as fact tables with multiple foreign keys referring to the contextual entities. And the contextual entities are each dimensions with a single primary key, e.g., for the customer dimension, as in Figure 6-1.

Dimensional modeling – Fact tables & Dimension tables

Kimball &Ross,2016

Star schema

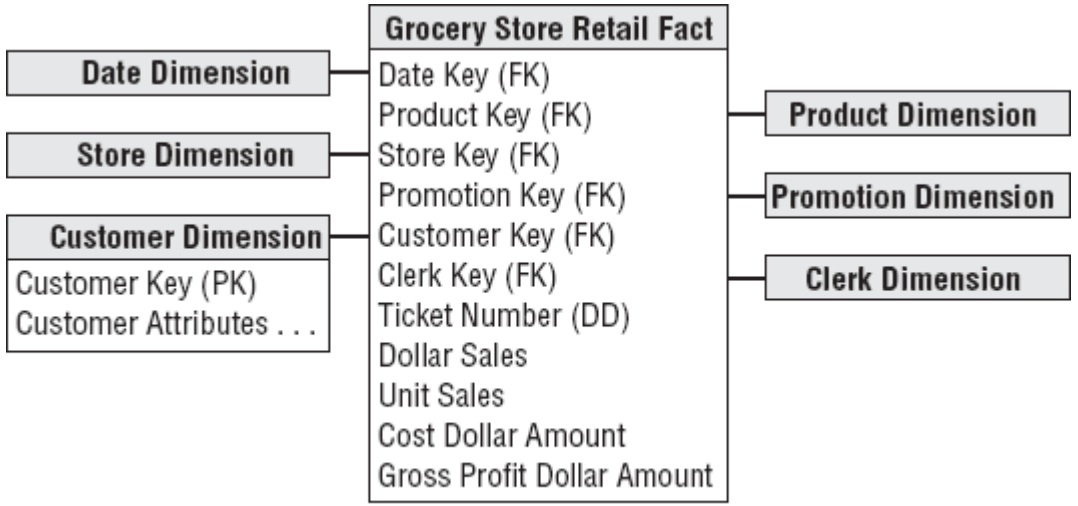


Figure 6-1: A dimensional model for grocery store sales.

Dimensional Keys

I insist that the foreign keys in the fact table obey referential integrity with respect to the primary keys in their respective dimensions. In other words, every foreign key in the fact table has a match to a unique primary key in the respective dimension.

> In the real world, there are many compelling reasons to build the FK-PK pairs as surrogate keys that are just sequentially assigned integers. It's a major mistake to build data warehouse keys out of the natural keys that come from the underlying operational data sources.

Occasionally a perfectly legitimate measurement will involve a missing dimension. Perhaps in some situations a product can be sold to a customer in a transaction without a store defined. In this case, rather than attempting to store a null value in the store FK, you build a special record in the store dimension representing "No Store." Now this condition has a perfectly normal FK-PK representation in the fact table.

Dimensional modeling – Fact tables & Dimension tables

Kimball &Ross,2016

A	B	C	.....
A2	B2	1.23	
A1	B3	2.34	.....
A2	B1	5.56	
A2	B2	8.36	
A1	B3	3.27	
A2	B1	9.45	
A2	B2	6.23	
A2	B1	1.98	
A1	B3	8.23	
A2	B2	0.11	
A3	B1	3.44	
A3	B1	2.08	

(a) Table R

A1	A2	A3
0	1	0
1	0	0
0	1	0
0	1	0
1	0	0
0	1	0
0	1	0
0	1	0
1	0	0
0	1	0
0	0	1
0	0	1

B1	B2	B3
0	1	0
0	0	1
1	0	0
0	1	0
0	0	1
1	0	0
0	1	0
1	0	0
0	0	1
0	1	0
1	0	0
1	0	0

(b) Bitmap Indices for A, B

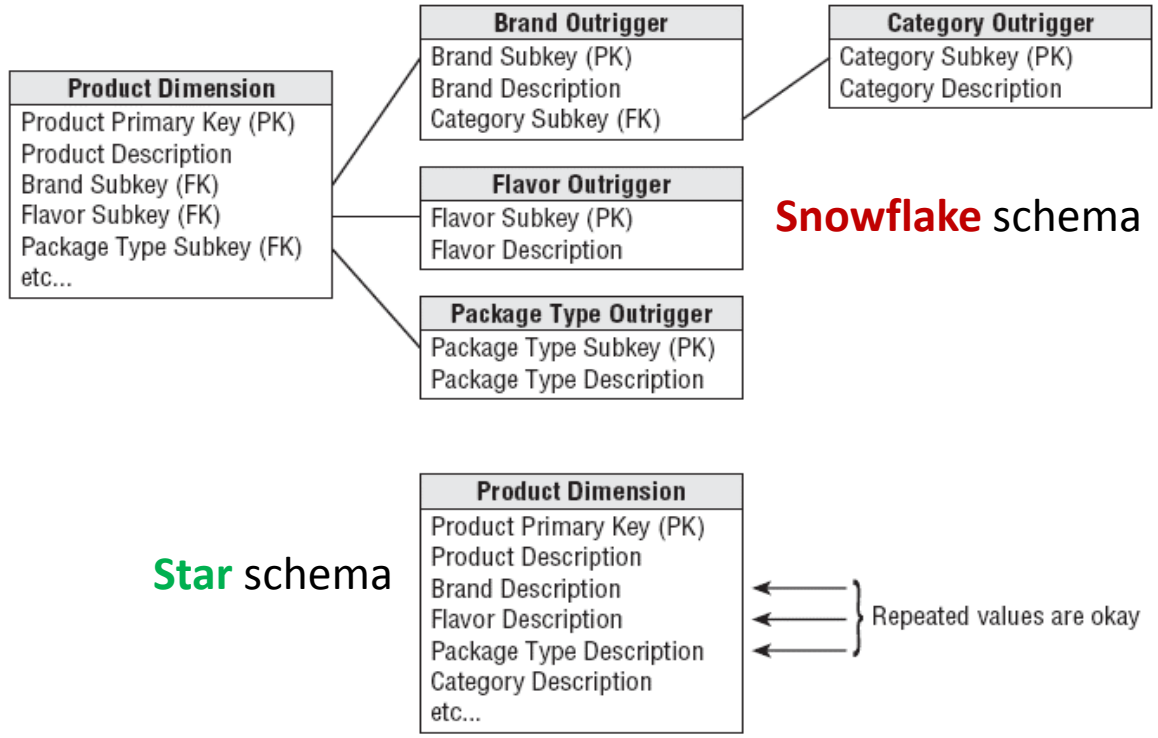


Figure 5-2: Two representations of a product dimension—the top snowflaked version is normalized to 3NF, and the bottom is denormalized into a flat dimension.

I resist the urge to further snowflake the dimension tables and am content to leave them in flat second normal form because the flat tables are much more efficient to query.

Snowflaking a dimension into third normal form, while not incorrect, destroys the ability to use bitmap indexes and increases the user perceived complexity of the design.

Dimensional modeling – Fact tables & Dimension tables

Kimball &Ross,2016

Declaring the Grain

Although theoretically any mixture of measured facts could be shoehorned into a single table, a proper dimensional design allows only facts of a uniform grain (the same dimensionality) to coexist in a single fact table. Uniform grain guarantees that all the dimensions are used with all the fact records (keeping in mind the "No Store" example) and greatly reduces the possibility of application errors due to combining data at different grains.

When you have facts at two different grains, place the facts in separate tables.

FactWeekly

WeekKey	Measure1
1	200
2	200
3	200
4	200
5	200
6	200

DimWeek

WeekKey	Week	StartDate	EndDate
1	Week 1	27 Dec 2010	2 Jan 2011
2	Week 2	3 Jan 2011	9 Jan 2011
3	Week 3	10 Jan 2011	16 Jan 2011
4	Week 4	17 Jan 2011	23 Jan 2011
5	Week 5	24 Jan 2011	30 Jan 2011
6	Week 6	31 Jan 2011	6 Feb 2011

FactMonthly

MonthKey	Measure1
1	4500
2	4000

DimMonth

MonthKey	Month
1	40544
2	40575

Dimensional modeling – Fact tables & Dimension tables

Kimball &Ross,2016

Additive Facts

At the heart of every fact table is the list of facts that represent the measurements. Because most fact tables are huge, with millions or even billions of rows, you almost never fetch a single record into your answer set. Rather, you fetch a very large number of records, which you compress into digestible form by adding, counting, averaging, or taking the min or max. But for practical purposes, the most common choice, by far, is adding.

Some facts, like bank balances and inventory levels, represent intensities that are awkward to express in an additive format. You can treat these semi-additive facts as if they were additive—but just before presenting the results to the business user, divide the answer by the number of time periods to get the right result. This technique is called *averaging over time*.

factless fact tables.

The classic example of a factless fact table is a record representing a student attending a class on a specific day. The dimensions are day, student, professor, course, and location, but there are no obvious numeric facts. The tuition paid and grade received are good facts, but not at the grain of the daily attendance.

Dimensional modeling – Fact tables & Dimension tables

Kimball &Ross,2016

Star schema

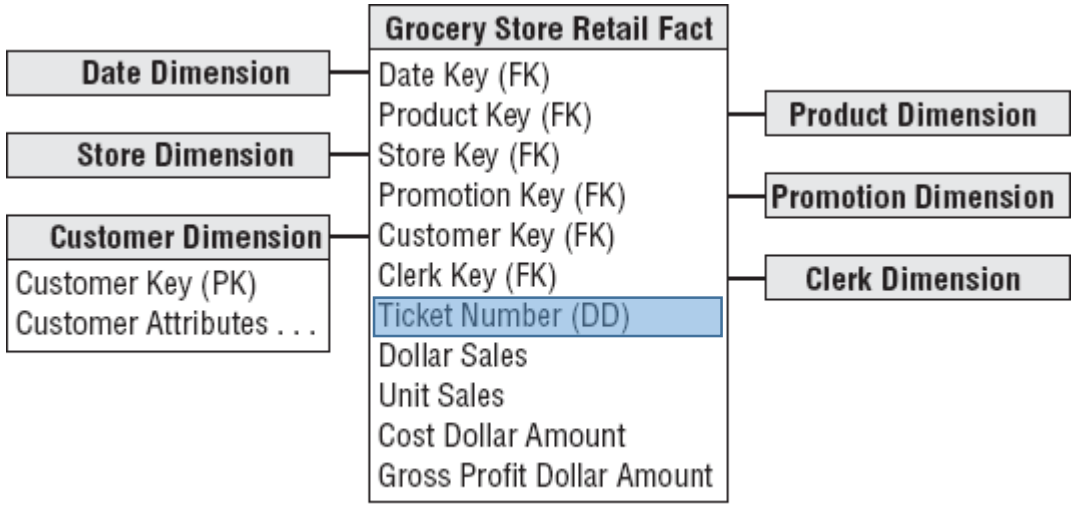
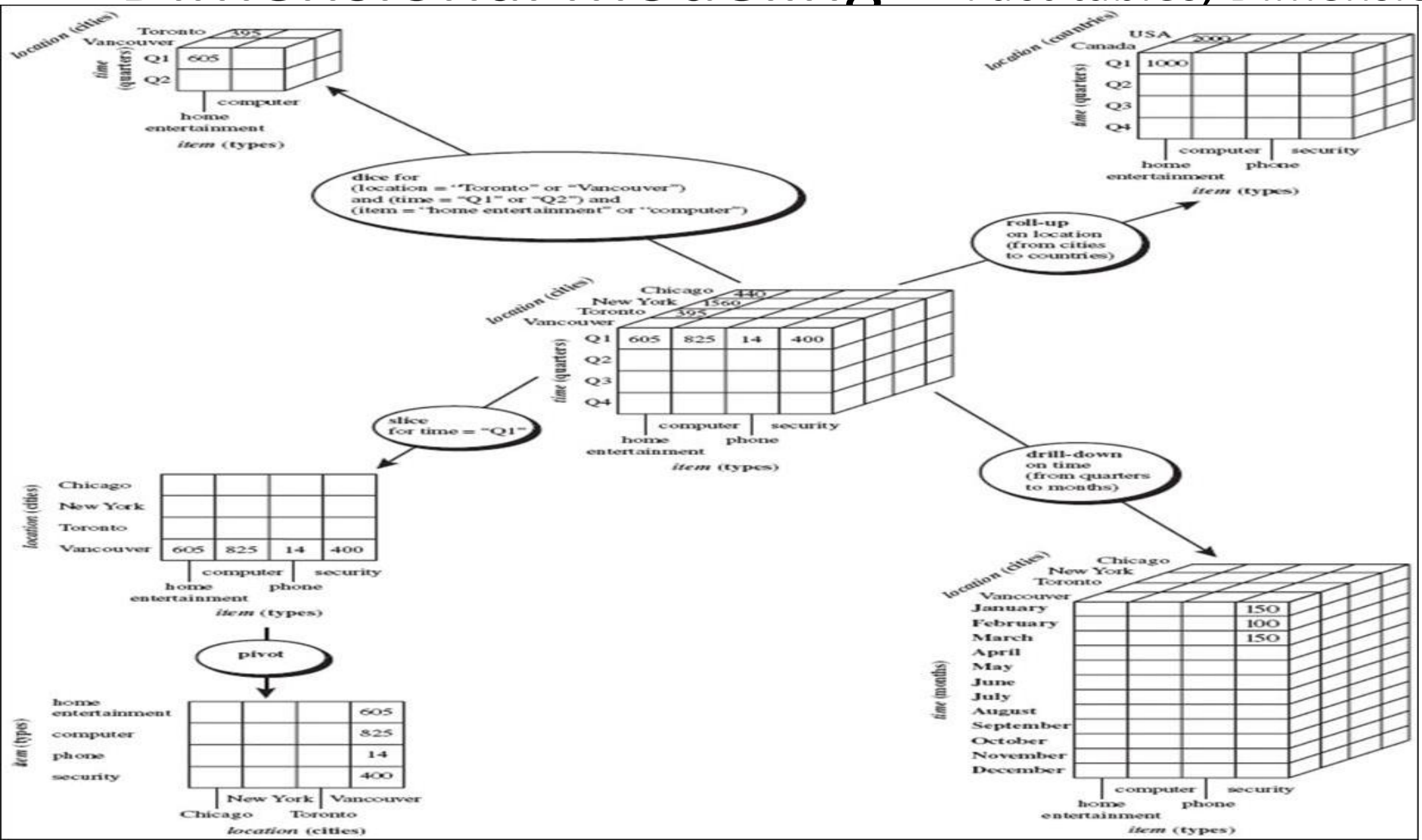


Figure 6-1: A dimensional model for grocery store sales.

Degenerate Dimensions

In many modeling situations where the grain is a child, the natural key of the parent header winds up as an orphan in the design. In the Figure 6-1 grocery example, the grain is the line item on a sales ticket, but the ticket number is the natural key of the parent ticket. Because you have systematically stripped off the ticket context as dimensions, the ticket number is left exposed without any attributes of its own. You model this reality by placing the ticket number by itself right in the fact table. We call this key a *degenerate dimension*. The ticket number is useful because it's the glue that holds the child records together.

Dimensional modeling – Fact tables, Dimension tables & OLAP operations



Dimensional modeling – Fact and Dim.Tables&OLAP ops.

Kimball &Ross,2016

Drilling Down, Up, and Across

Drilling Down

Drilling down is the oldest and most venerable kind of drilling in a data warehouse. Drilling down means nothing more than "give me more detail." In our standard dimensional schema, the attributes in the dimension tables play a crucial role. These attributes are textual (or behave like text), take on discrete values, and are the source of application constraints and grouping columns in the final reports. In fact, you can always imagine creating a grouping column in a report by opportunistically dragging a dimension attribute from any of the dimension tables down into the report, thereby making it a grouping column, as shown in Figure 6-2. The beauty of the dimensional model is that all dimension attributes can become grouping columns. The process of adding grouping columns can be compounded with as many grouping columns from as many dimension tables as the user wishes. The great strength of SQL is that these grouping columns simply get added to the `SELECT` list and the `GROUP BY` clause and the right thing happens. Usually you add these grouping columns to the `ORDER BY` clause also so that you get the grouping in a prescribed order.

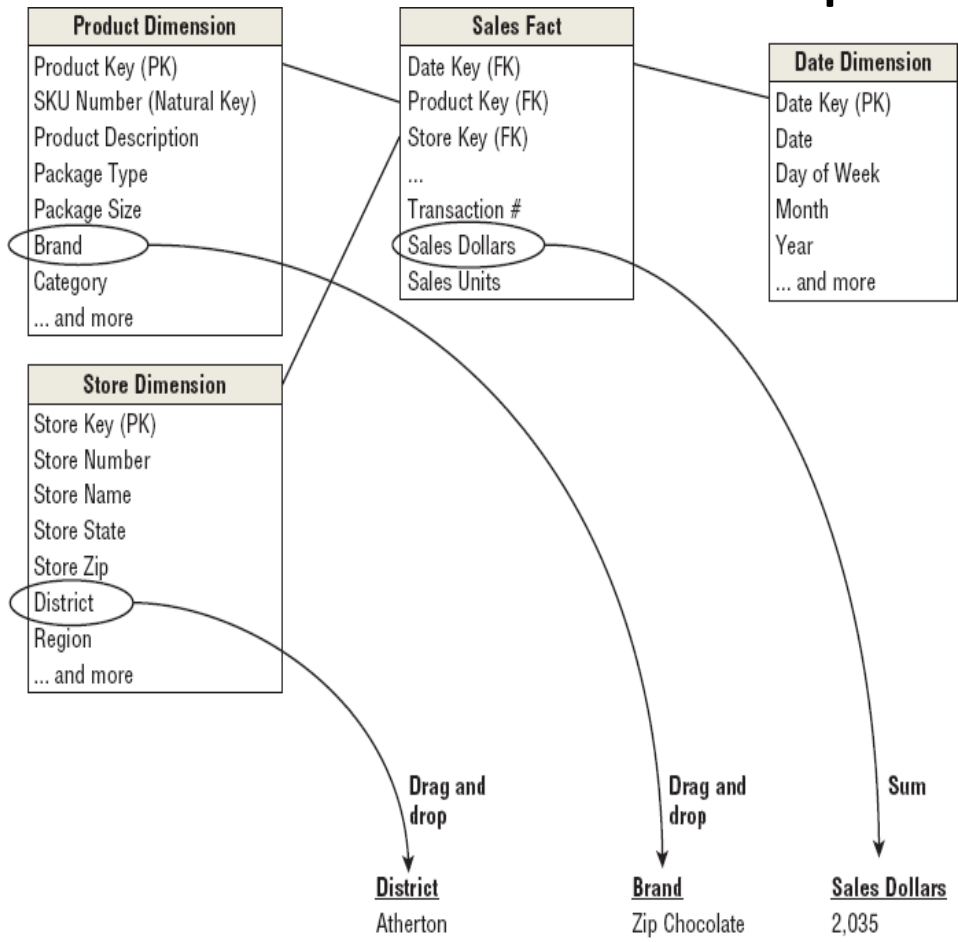


Figure 6-2: Dragging dimension attributes and facts into a report.

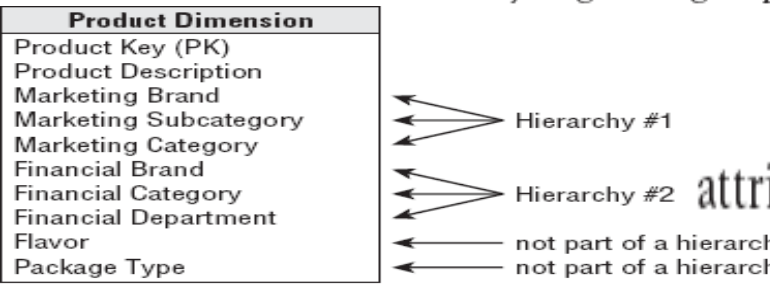


Figure 6-3: A product dimension table with both marketing and finance attributes.

The user must be allowed to traverse any hierarchy and choose unrelated attributes that are not part of the hierarchy.

Dimensional modeling – Fact and Dim.Tables&OLAP ops.

Drilling Down

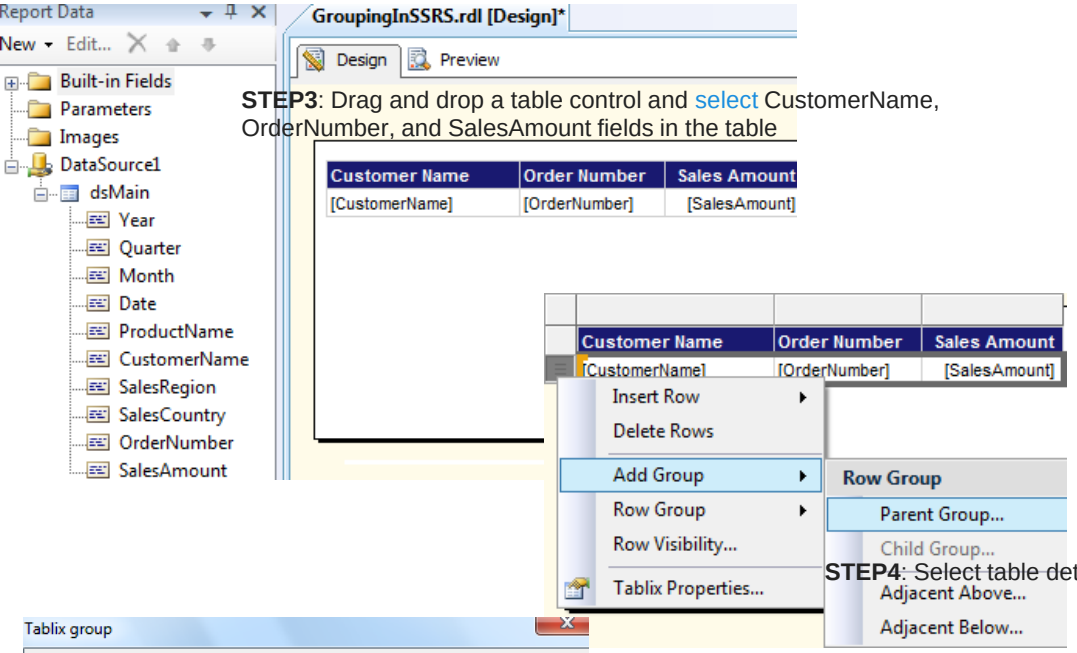
*Translate Drill Down and Up into **colors** with SQL Server Reporting Services*

Particulars	Sales Amount
2005	$3,266,373.66
2006	$6,530,343.53
1	$1,791,698.45
2	$2,014,012.13
April	$663,692.29
4/1/2006 12:00:00 AM	$12,133.01
4/2/2006 12:00:00 AM	$10,556.53
4/3/2006 12:00:00 AM	$29,121.98
Australia	$14,109.80
Canada	$7,156.54
Northwest	$4,277.37
United States	$4,277.37
Road-150 Red, 48	$3,578.27
Road-650 Red, 52	$699.10
AustinFoster	$699.10
United Kingdom	$3,578.27
4/4/2006 12:00:00 AM	$24,641.33
4/5/2006 12:00:00 AM	$17,688.07

STEP1: Add new report in Report Server project

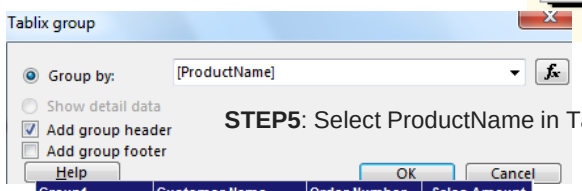
STEP2: Create a new dataset

```
SELECT
D.CalendarYear AS [Year]
,D.CalendarQuarter AS [Quarter]
,D.EnglishMonthName AS [Month]
,D.FullDateAlternateKey AS [Date]
,P.EnglishProductName AS [ProductName]
,C.FirstName + LastName AS [CustomerName]
,ST.SalesTerritoryRegion AS [SalesRegion]
,ST.SalesTerritoryCountry AS [SalesCountry]
,F.SalesOrderNumber AS [OrderNumber]
,F.SalesAmount
FROM FactInternetSales F
JOIN DimProduct P
  ON P.ProductKey = F.ProductKey
JOIN DimCustomer C
  ON C.CustomerKey = F.CustomerKey
JOIN DimDate D
  ON D.DateKey = F.OrderDateKey
JOIN DimSalesTerritory ST
  ON ST.SalesTerritoryKey = F.SalesTerritoryKey
```



STEP3: Drag and drop a table control and select CustomerName, OrderNumber, and SalesAmount fields in the table

STEP4: Select table detail row --> Right click --> select Add Group --> Row Group --> Parent Group



STEP5: Select ProductName in Tablix group

Group1	Customer Name	Order Number	Sales Amount
[ProductName]			
	[CustomerName]	[OrderNumber]	[SalesAmount]

… more details at: goo.gl/0q4VGg

Particulars	Order Number	Sales Amount
[Year]		[Sum(SalesAmount)]
[Quarter]		[Sum(SalesAmount)]
[Month]		[Sum(SalesAmount)]
[Date]		[Sum(SalesAmount)]
[SalesRegion]		[Sum(SalesAmount)]
[SalesCountry]		[Sum(SalesAmount)]
[ProductName]		[Sum(SalesAmount)]
[CustomerName]	[OrderNumber]	[SalesAmount]

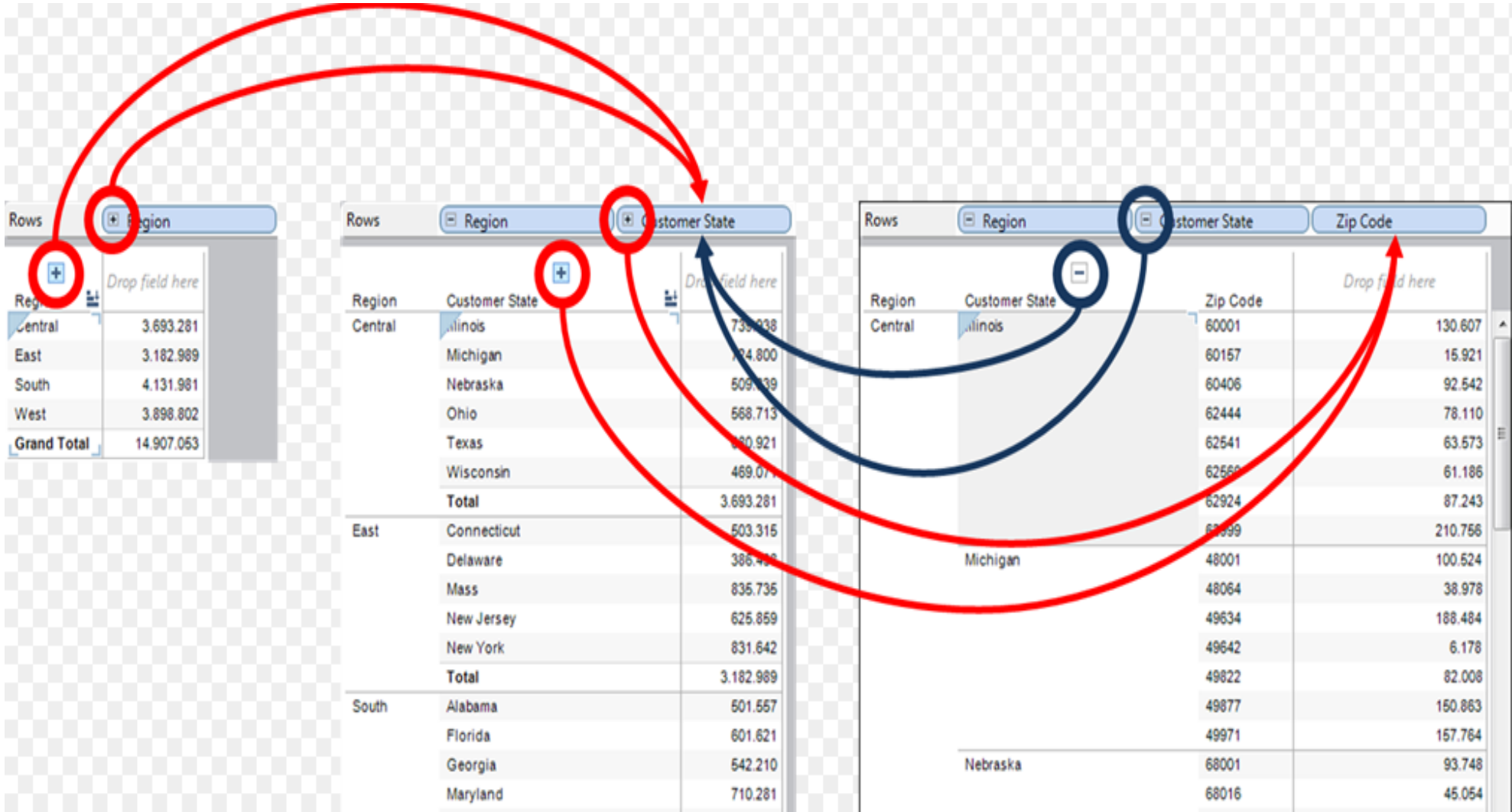
STEP9: Format Groups

Group	Left Indent	BackgroundColor
Year	2pt	#7c73c8
Quarter	12pt	#9894ca
Month	22pt	#b4b4c8
Date	32pt	#c7c7d8
SalesRegion	42pt	#dadaeb
SalesCountry	52pt	#e7e7f0
ProductName	62pt	#f4f4fc
CustomerName (detail row)	72pt	White

Dimensional modeling – Fact and Dim.Tables&OLAP ops.

Drilling Down

example:



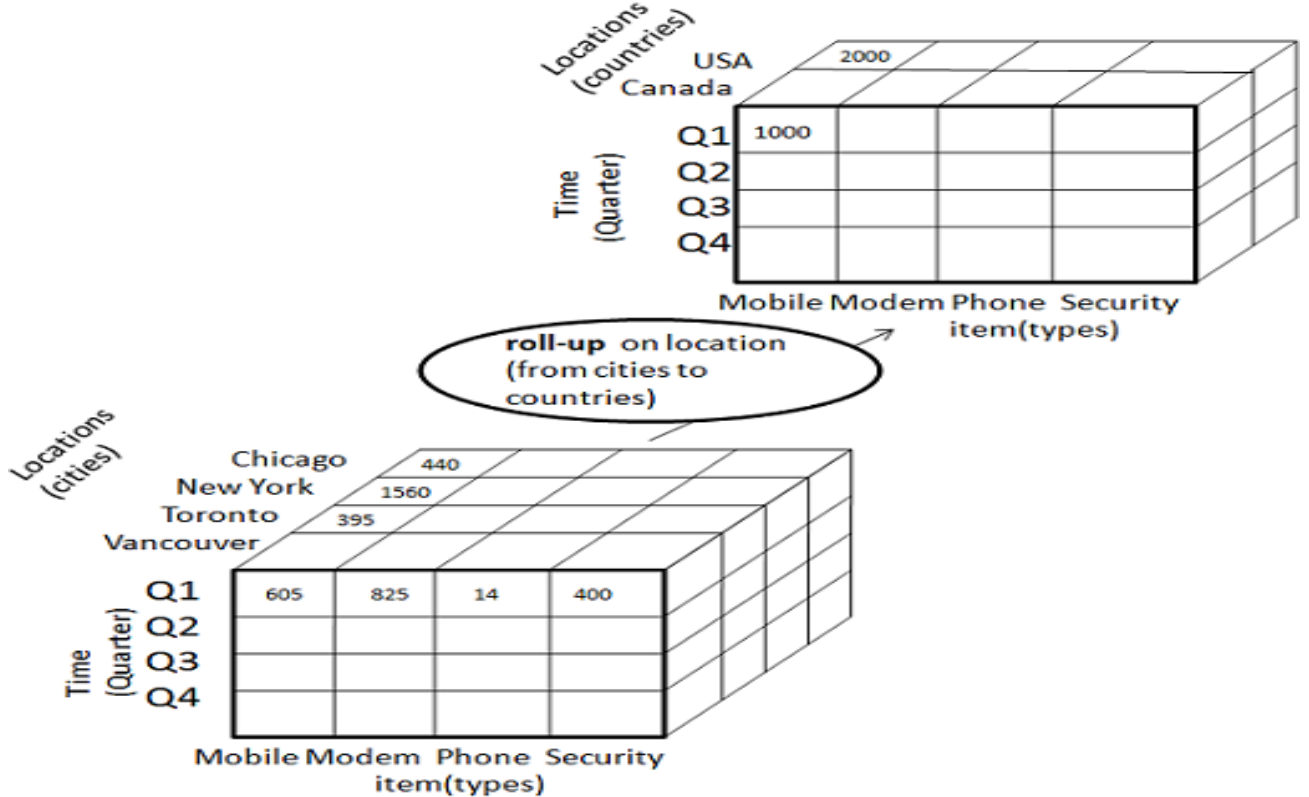
Dimensional modeling – Fact and Dim.Tables&OLAP ops.

Kimball &Ross,2016

Drilling Down, Up, and Across

Drilling Up

If drilling down is adding grouping columns from the dimension tables, then drilling up is subtracting grouping columns. Of course, it is not necessary to subtract the grouping columns in the same order that they were added. In general, each time the user adds or subtracts a grouping column, a new multi-table join query must be launched.



Dimensional modeling – Fact and Dim.Tables&OLAP ops.

Kimball &Ross,2016

Drilling Down, Up, and Across

Drilling Across

If drilling down is requesting ever finer and more granular data from the same fact table, then drilling across is the process of linking two or more fact tables at the same granularity, or, in other words, tables with the same set of grouping columns and dimensional constraints. Drilling across is a valuable technique whenever a business has several fundamental business processes that can be arranged in a value chain. Each business process gets its own separate fact table. For example, almost all manufacturers have an obvious value chain representing the demand side of their businesses consisting of finished goods inventory, orders, shipments, customer inventory, and customer sales, as shown in Figure 6-4. The product and time dimensions thread through all of these fact tables. Some dimensions, such as customer, thread through some, but not all of the fact tables. For instance, customer does not apply to finished goods inventory.

A drill-across report can be created by using grouping columns that apply to all the fact tables used in the report. Thus in our manufacturing value chain example, attributes may be freely chosen from the product and time dimension tables because they make sense for every fact table.

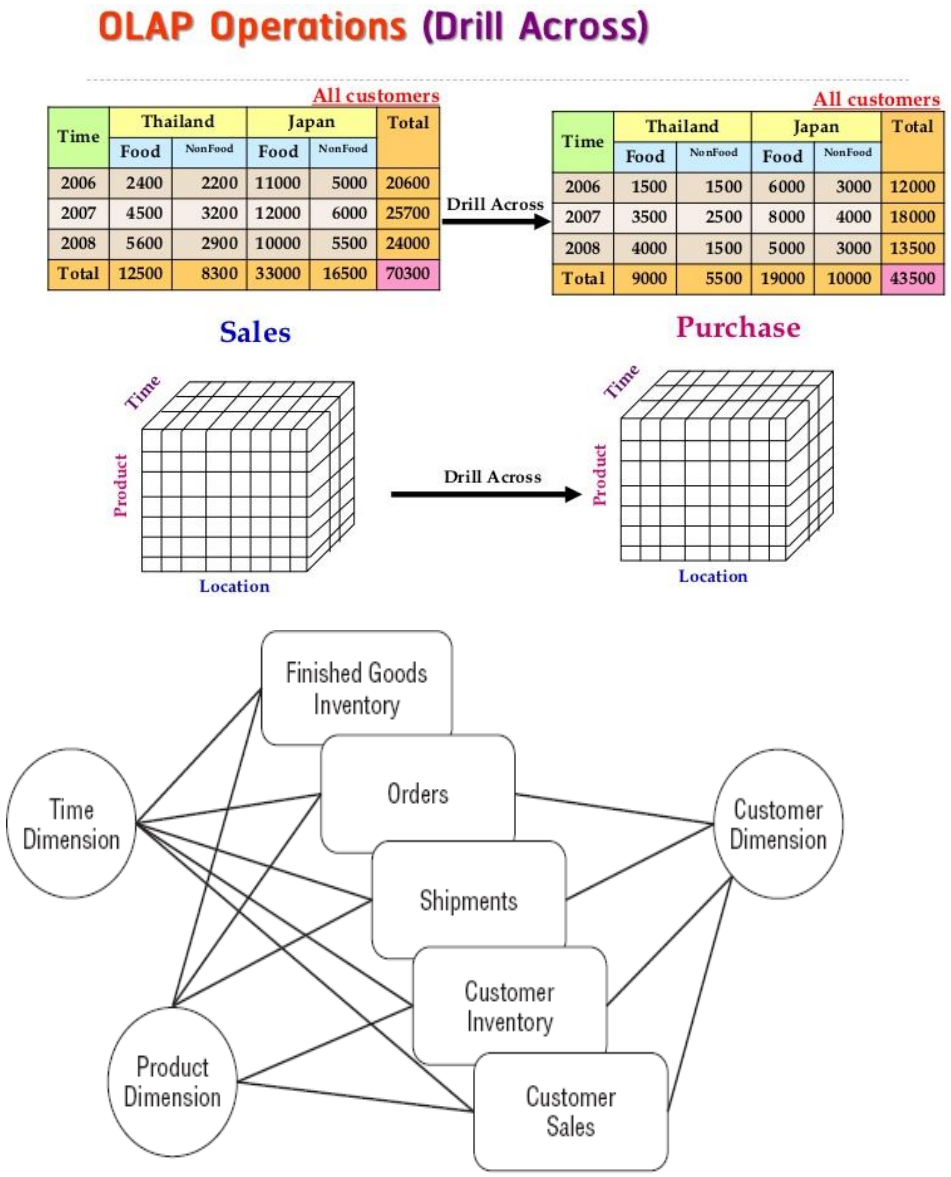


Figure 6-4: Separate fact tables for each business process share common dimensions.

Dimensional modeling

Ten Essential Rules of Dimensional Modeling

Kimball &Ross,2016

Rule #1: Load detailed atomic data into dimensional structures. Dimensional models should be populated with bedrock atomic details to support the unpredictable filtering and grouping required by business user queries. Users typically don't need to see a single record at a time, but you can't predict the somewhat arbitrary ways they'll want to screen and roll up the details. If only summarized data is available, then you've already made assumptions about data usage patterns that will cause users to run into a brick wall when they want to dig deeper into the details.

Rule #2: Structure dimensional models around business processes. Business processes are the activities performed by your organization; they represent measurement events, like taking an order or billing a customer. Business processes typically capture or generate unique performance metrics associated with each event. These metrics translate into facts, with each business process represented by a single atomic fact table. In addition to single process fact tables, consolidated fact tables are sometimes created that combine metrics from multiple processes into one fact table at a common level of detail; consolidated fact tables are a complement to the detailed single process fact tables, not a substitute for them.

Rule #3: Ensure every fact table has a date dimension table associated with it. The measurement events described in rule #2 always have a date stamp of some variety associated with them, whether it's a monthly balance snapshot or a monetary transaction captured to the hundredth of a second. Every fact table should have at least one foreign key to an associated date dimension table, whose grain is a single day, with calendar attributes and nonstandard characteristics about the measurement event date, such as the fiscal month and corporate holiday indicator.

Dimensional modeling

Ten Essential Rules of Dimensional Modeling

Kimball &Ross,2016

Rule #4: Ensure that all facts in a single fact table are at the same grain or level of detail. There are three fundamental grains to categorize all fact tables: transactional, periodic snapshot, or accumulating snapshot. Regardless of its grain type, every measurement within a fact table must be at the exact same level of detail.

Rule #5: Resolve many-to-many relationships in fact tables. Because a fact table stores the results of a business process event, there's inherently a many-to-many (M:M) relationship between its foreign keys, such as multiple products being sold in multiple stores on multiple days. These foreign key fields should never be null.

Rule #6: Resolve many-to-one relationships in dimension tables. Hierarchical, fixed depth many-to-one (M:1) relationships between attributes are typically denormalized or collapsed into a flattened dimension table. If you've spent most of your career designing normalized entity-relationship models for transaction processing systems, you'll need to resist your instinctive tendencies to normalize or snowflake the M:1 relationship into smaller subdimensions; dimension denormalization is the name

Rule #7: Store report labels and filter domain values in dimension tables. The codes and, more importantly, associated decodes and descriptors used for labeling and query filtering should be captured in dimension tables. Avoid storing cryptic code fields or bulky descriptive fields in the fact table itself; likewise, don't just store the code in the dimension table and assume that users don't need descriptive decodes or that they'll be handled in the BI application.

Dimensional modeling

Ten Essential Rules of Dimensional Modeling

Kimball &Ross,2016

Rule #8: Make sure dimension tables use a surrogate key. Meaningless, sequentially assigned surrogate keys (except for the date dimension where chronologically assigned and even more meaningful keys are acceptable) deliver a number of operational benefits, including smaller keys, which mean smaller fact tables, smaller indexes, and improved performance. Surrogate keys are absolutely required if you're tracking dimension attribute changes with a new dimension record for each profile change;

Rule #9: Create conformed dimensions to integrate data across the enterprise. Conformed dimensions (otherwise known as common, master, standard, or reference dimensions) are essential for enterprise data warehousing. Managed once in the ETL system and then reused across multiple fact tables, conformed dimensions deliver consistent descriptive attributes across dimensional models and support the ability to drill across and integrate data from multiple business processes.

Rule #10: Continuously balance the requirements and realities to deliver a DW/BI solution that's accepted by the business users and supports their decision making. Dimensional modelers must constantly straddle the business users' requirements along with the underlying realities of the associated source data to deliver a design that is both reasonably implementable, and more importantly, stands a reasonable chance of business adoption. The requirements versus realities balancing act is a fact of life for DW/BI practitioners, whether you're focused on the dimensional model, project strategy, technical/ETL/BI architectures, or deployment/maintenance plan.

Dimensional modeling

What Not to Do

The 12 dimensional modeling techniques to avoid

Kimball &Ross,2016

Mistake 12: Place text attributes in a fact table if you mean to use them as the basis of constraining and grouping. Creating a dimensional model is a kind of triage. Start by identifying the numeric measurements delivered from an operational source; those go in the fact table. Then identify the descriptive textual attributes from the context of the measurements; these go in the dimensions. Finally, make a case-by-case decision about the leftover codes and pseudo numeric items, placing them in the fact table if they are more like measurements, and in the dimension table if they are more like physical descriptions of something.

Mistake 11: Limit the use of verbose descriptive attributes in dimensions to save space. You might think that you're being a good, conservative designer by keeping the size of your dimensions under control. But in virtually every data warehouse, the dimension tables are geometrically smaller than the fact tables. So what if you have a 100MB product dimension table, if the fact table is 100 times as large! To design an easy-to-use data warehouse, you must supply as much verbose descriptive context in each dimension as you can. Make sure every code is augmented with readable descriptive text. Remember that the textual attributes in the dimensions "implement" the user interface for browsing your data, provide the entry points for constraining, and supply the content for the row and column headers in the final reports.

Mistake 10: Split hierarchies and hierarchy levels into multiple dimensions. A hierarchy is a cascaded series of many-to-one relationships. Many products roll up to a single brand. Many brands roll up to a single category, and so on. If your dimension is expressed at the lowest level of granularity (such as product), then all the higher levels of the hierarchy can be expressed as unique values in the product record. Users understand hierarchies, and your job is to present them in the most natural and efficient way. A hierarchy belongs in a single, physical flat dimension table. Resist the urge to "snowflake" a hierarchy by generating a set of progressively smaller subdimension tables.

Dimensional modeling

Kimball &Ross,2016

What Not to Do

The 12 dimensional modeling techniques to avoid

Mistake 9: Delay dealing with a slowly changing dimension (SCD). Too many data warehouses are designed to regularly overwrite the most important dimensions, such as customer and product, from the underlying data sources. This goes against a basic data warehouse oath: The data warehouse will represent history accurately, even if the underlying data source does not. SCDs are an essential design element of every data warehouse.

Mistake 8: Use smart keys to join a dimension table to a fact table. Beginning data warehouse designers tend to be somewhat too literal minded when designing the primary keys in dimension tables that must necessarily connect to the foreign keys of the fact table. It is counterproductive to declare a whole suite of dimension attributes as the dimension table key and then use them all as the basis of the physical join to the fact table. All sorts of ugly problems eventually arise. Replace the smart physical key with a simple integer surrogate key that is numbered sequentially from 1 to N (the number of records in the dimension table).

Mistake 7: Add dimensions to a fact table before declaring its grain. All dimensional designs should start with the numeric measurements and work outward. First, identify the source of the measurements. Second, specify the exact granularity and meaning of the measurements. Third, surround these measurements with dimensions that are true to that grain. Staying true to the grain is a crucial step in the design of a dimensional data model.

Mistake 6: Declare that a dimensional model is "based on a specific report." A dimensional model has nothing to do with an intended report! A dimensional model is a model of a measurement process. A numeric measurement is a solid physical reality. Numeric measurements form the basis of fact tables. The dimensions appropriate for a given fact table are the physical context that describe the circumstances of the measurements. A dimensional model is solidly based on the physics of a measurement process and is quite independent from how a business user chooses to define a report.

Dimensional modeling

Kimball &Ross,2016

What Not to Do

The 12 dimensional modeling techniques to avoid

Mistake 5: Mix facts of differing grain in the same fact table. A serious error in a dimensional design is to add "helpful" facts to a fact table, such as records that describe totals for an extended time span or rolled up geographic area. Although these extra facts are well known at the time of the individual measurement and would seem to make some applications simpler, they cause havoc because all the automatic summations across dimensions double and triple count these higher level facts, producing incorrect results. Each different measurement grain demands its own fact table.

Mistake 4: Leave lowest level atomic data in normalized format. The lowest level data is the most dimensional and should be the physical foundation of your dimensional design. Aggregated data has been deprived of some of its dimensions. If you build a dimensional model from aggregated data and expect your user query tools to drill down to normalized atomic data that you have left in your staging area, then you're dreaming. Build all your dimensional models on the most atomic data. Make all atomic data part of the presentation portion of your data warehouse. Then your user tools will gracefully resist the "ad hoc attack."

Mistake 3: Eschew aggregate fact tables and shrunken dimension tables when faced with query performance concerns; solve performance problems by adding more parallel processing hardware. Aggregates (such as Oracle's materialized views and IBM DB2's automatic summary tables) are the single most cost-effective way to improve query performance. Most query tool vendors explicitly support aggregates, and all of these depend on dimensional modeling constructs. The addition of parallel processing hardware, which is expensive, should be done as part of a balanced program that consists also of building aggregates, choosing query-efficient DBMS software, building lots of indexes, increasing real memory size, and increasing CPU speed.

Mistake 2: Fail to conform facts across separate fact tables. It would be a shame to get this far and then build stovepipes. This is called snatching defeat from the jaws of victory. If you have a numeric measured fact called, for example, revenue, in two or more of your dimensional models sourced from different underlying systems, then you need to take special care to make sure that the technical definitions of these facts match exactly. You want to be able to add and divide these separate revenue facts freely in your applications. This act is called *conforming the facts*.

Mistake 1: Fail to conform dimensions across separate fact tables. This is the biggest mistake because the single most important design technique in the dimensional modeling arsenal is conforming your dimensions. If two or more fact tables have the same dimension, you must be a fanatic about making these dimensions identical or carefully chosen subsets of each other. When you conform your dimensions across fact tables, you will be able to drill across separate data sources because the constraints and row headers will mean the same thing and match at the data level.

Bibliography

-Airinei, D., Depozite de date, Polirom, Iaşi, 2002
portal.feaa.uaic.ro/Master/sia/Pages/default.aspx
-Airinei, D., Dospinescu, O., Huiban, A., Aplicatii practice cu sisteme OLAP si Depozite de date, Editura Sedcom Libris, Iaşi, 2008
-Homocianu, D., Sistemele de asistare a deciziilor in contextual societatii cunoasterii, Editura UAIC, Iasi, 2009 (ssrn.com/abstract=2384380)
-Inmon,W.H.,Linstedt, D., Data Architecture: A primer for the data scientist, MK, MA, 2015 (tinyurl.com/h7dcq66)
-Kimball, R., Ross, M., The Data Warehouse Toolkit Third Edition. The Definitive Guide to Dimensional Modeling, John Wiley & Sons, New York, 2013 (tinyurl.com/jogo5uy)
-Kimball, R., Ross, M., The Kimball Group Reader. Relentlessly Practical Tools for Data Warehousing and Business Intelligence – Remastered Collection, Second Edition, Wiley, New York, 2016 (tinyurl.com/johox4v)
-Krishnan, K, Data Warehousing in The Age of Big Data, Morgan Kaufmann (MK), MA, 2013 (tinyurl.com/zrjo75j)
-Reeves, L.L., A Manager's Guide to Data Warehousing, Wiley, New York, 2009 (tinyurl.com/htsslk8)
-Sarka, D., et. al., Implementing a Data Warehouse with Microsoft SQL Server 2012. Training Kit, O'Reilly Media, Sebastopol, 2012 (tinyurl.com/jk6lclk)
-Sheldon, B., et. al., Professional Visual Basic 2012 and .NET 4.5 Programming, John Wiley & Sons, Indianapolis, 2013 (tinyurl.com/hrb6j9u)