11 Telecommunications

This chapter unfolds a bit differently than preceding chapters. We begin with a case study overview but we won't be designing a dimensional model from scratch this time. Instead, we'll step into a project midstream to conduct a design review, looking for opportunities to improve the initial draft schema. The bulk of this chapter focuses on identifying design flaws in dimensional models.

We'll use a billing vignette drawn from the telecommunications industry as the basis for the case study; it shares similar characteristics with the billing data generated by a utilities company. At the end of this chapter we'll describe the handling of geographic location information in the data warehouse.

Chapter 11 discusses the following concepts:

- Bus matrix snippet for telecommunications company
- Design review exercise
- Checklist of common design mistakes
- Recommended tactics when conducting design reviews
- Retrofitting existing data structures
- Abstract geographic location dimensions

Telecommunications Case Study and Bus Matrix

Given your extensive experience in dimensional modeling (10 chapters so far), you've recently been recruited to a new position as a dimensional modeler on the DW/BI team for a large wireless telecommunications company. On your first day, after a few hours of human resources paperwork and orientation, you're ready to get to work.

The DW/BI team is anxious for you to review its initial dimensional design. So far it seems the project is off to a good start. The business and IT sponsorship committee appreciates that the DW/BI program must be business-driven; as such, the

committee was fully supportive of the business requirements gathering process. Based on the requirements initiative, the team drafted an initial data warehouse bus matrix, illustrated in Figure 11-1. The team identified several core business processes and a number of common dimensions. Of course, the complete enterprise matrix would have a much larger number of rows and columns, but you're comfortable that the key constituencies' major data requirements have been captured.

	Date	Product	Customer	Rate Plan	Sales Organization	Service Line #	Switch	Employee	Support Call Profile
Purchasing	X	X						X	
Internal Inventory	X	X							
Channel Inventory	X	X			X				
Service Activation	X	X	X	X	X	X			
Product Sales	X	X	X	X	X	X			
Promotion Participation	X	X	X	X	X	X		X	
Call Detail Traffic	X	X	X	X	X	X	X		
Customer Billing	X	X	X	X	X	X			
Customer Support Calls	X	X	X	X	X	X		X	X
Repair Work Orders	X	X	X			X		X	X

Figure 11-1: Sample bus matrix rows for telecommunications company.

The sponsorship committee decided to focus on the customer billing process for the initial DW/BI project. Business management determined better access to the metrics resulting from the billing process would have a significant impact on the business. Management wants the ability to see monthly usage and billing metrics (otherwise known as revenue) by customer, sales organization, and rate plan to perform sales channel and rate plan analyses. Fortunately, the IT team felt it was feasible to tackle this business process during the first warehouse iteration.

Some people in the IT organization thought it would be preferable to tackle individual call and message detail traffic, such as every call initiated or received by every phone. Although this level of highly granular data would provide interesting insights, it was determined by the joint sponsorship committee that the associated data presents more feasibility challenges while not delivering as much short-term business value.

Based on the sponsors' direction, the team looked more closely at the customer billing data. Each month, the operational billing system generates a bill for each phone number, also known as a service line. Because the wireless company has millions of service lines, this represents a significant amount of data. Each service

line is associated with a single customer. However, a customer can have multiple wireless service lines, which appear as separate line items on the same bill; each service line has its own set of billing metrics, such as the number of minutes, number of text messages, amount of data, and monthly service charges. There is a single rate plan associated with each service line on a given bill, but this plan can change as customers' usage habits evolve. Finally, a sales organization and channel is associated with each service line to evaluate the ongoing billing revenue stream generated by each channel partner.

Working closely with representatives from the business and other DW/BI team members, the data modeler designed a fact table with the grain being one row per bill each month. The team proudly unrolls its draft dimensional modeling masterpiece, as shown in Figure 11-2, and expectantly looks at you.

What do you think? Before moving on, please spend several minutes studying the design in Figure 11-2. Try to identify the design flaws and suggest improvements before reading ahead.

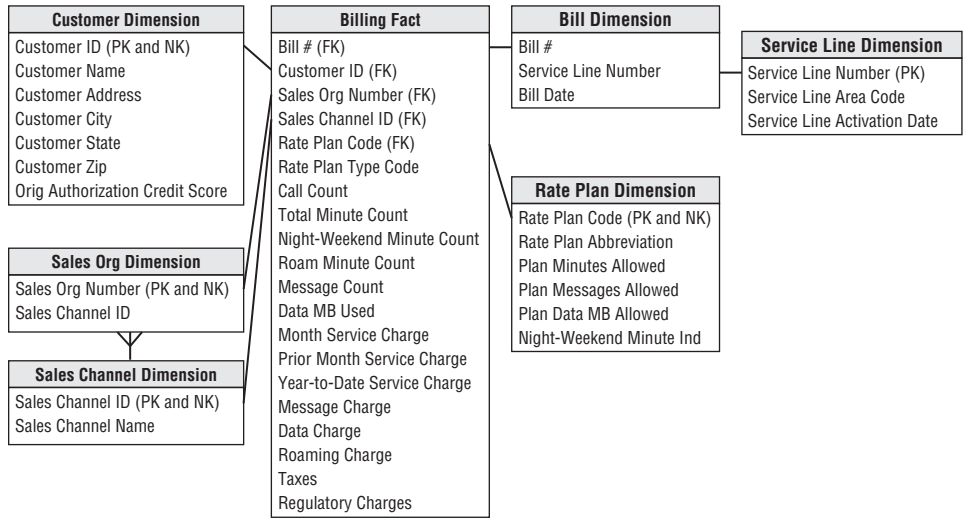

Figure 11-2: Draft schema prior to design review.

General Design Review Considerations

Before we discuss the specific issues and potential recommendations for the Figure 11-2 schema, we'll outline the design issues commonly encountered when conducting design reviews. Not to insinuate that the DW/BI team in this case study has stepped into all these traps, but it may be guilty of violating several. Again, the design review exercise will be a more effective learning tool if you take a moment to jot down your personal ideas regarding Figure 11-2 before proceeding.

Balance Business Requirements and Source Realities

Dimensional models should be designed based on a blended understanding of the business's needs, along with the operational source system's data realities. While requirements are collected from the business users, the underlying source data should be profiled. Models driven solely by requirements inevitably include data elements that can't be sourced. Meanwhile, models driven solely by source system data analysis inevitably omit data elements that are critical to the business's analytics.

Focus on Business Processes

As reinforced for 10 chapters, dimensional models should be designed to mirror an organization's primary business process events. Dimensional models should not be designed solely to deliver specific reports or answer specific questions. Of course, business users' analytic questions are critical input because they help identify which processes are priorities for the business. But dimensional models designed to produce a specific report or answer a specific question are unlikely to withstand the test of time, especially when the questions and report formats are slightly modified. Developing dimensional models that more fully describe the underlying business process are more resilient to change. Process-centric dimensional models also address the analytic needs from multiple business departments; the same is definitely not true when models are designed to answer a single department's specific need.

After the base processes have been built, it may be useful to design complementary schemas, such as summary aggregations, accumulating snapshots that look across a workflow of processes, consolidated fact tables that combine facts from multiple processes to a common granularity, or subset fact tables that provide access to a limited subset of fact data for security or data distribution purposes. Again, these are all secondary complements to the core process-centric dimensional models.

Granularity

The first question to always ask during a design review is, "What's the grain of the fact table?" Surprisingly, you often get inconsistent answers to this inquiry from a design team. Declaring a clear and concise definition of the grain of the fact table is critical to a productive modeling effort. The project team and business liaisons must share a common understanding of this grain declaration; without this agreement, the design effort will spin in circles.

Of course, if you've read this far, you're aware we strongly believe fact tables should be built at the lowest level of granularity possible for maximum flexibility and extensibility, especially given the unpredictable filtering and grouping required by business user queries. Users typically don't need to see a single row at a time,

but you can't predict the somewhat arbitrary ways they'll want to screen and roll up the details. The definition of the lowest level of granularity possible depends on the business process being modeled. In this case, you want to implement the most granular data available for the selected billing process, not just the most granular data available in the enterprise.

Single Granularity for Facts

After the fact table granularity has been established, facts should be identified that are consistent with the grain declaration. To improve performance or reduce query complexity, aggregated facts such as year-to-date totals sometimes sneak into the fact row. These totals are dangerous because they are not perfectly additive. Although a year-to-date total reduces the complexity and run time of a few specific queries, having it in the fact table invites double counting the year-to-date column (or worse) when more than one date is included in the query results. It is important that once the grain of a fact table is chosen, all the additive facts are presented at a uniform grain.

You should prohibit text fields, including cryptic indicators and flags, from the fact table. They almost always take up more space in the fact table than a surrogate key. More important, business users generally want to query, constrain, and report against these text fields. You can provide quicker responses and more flexible access by handling these textual values in a dimension table, along with descriptive rollup attributes associated with the flags and indicators.

Dimension Granularity and Hierarchies

Each of the dimensions associated with a fact table should take on a single value with each row of fact table measurements. Likewise, each of the dimension attributes should take on one value for a given dimension row. If the attributes have a many-to-one relationship, this hierarchical relationship can be represented within a single dimension. You should generally look for opportunities to collapse or denormalize dimension hierarchies whenever possible.

Experienced data modelers often revert to the normalization techniques they've applied countless times in operational entity-relationship models. These modelers often need to be reminded that normalization is absolutely appropriate to support transaction processing and ensure referential integrity. But dimensional models support analytic processing. Normalization in the dimensional model negatively impacts the model's twin objectives of understandability and performance. Although normalization is not forbidden in the extract, transform, and load (ETL) system where data integrity must be ensured, it does place an additional burden on the dimension change handling subsystems.

Sometimes designers attempt to deal with dimension hierarchies within the fact table. For example, rather than having a single foreign key to the product dimension, they include separate foreign keys for the key elements in the product hierarchy, such as brand and category. Before you know it, a compact fact table turns into an unruly centipede fact table joining to dozens of dimension tables. If the fact table has more than 20 or so foreign keys, you should look for opportunities to combine or collapse dimensions.

Elsewhere, normalization appears with the snowflaking of hierarchical relationships into separate dimension tables linked to one another. We generally also discourage this practice. Although snowflaking may reduce the disk space consumed by dimension tables, the savings are usually insignificant when compared with the entire data warehouse environment and seldom offset the disadvantages in ease of use or query performance.

Throughout this book we have occasionally discussed outriggers as permissible snowflakes. Outriggers can play a useful role in dimensional designs, but keep in mind that the use of outriggers for a cluster of relatively low-cardinality should be the exception rather than the rule. Be careful to avoid abusing the outrigger technique by overusing them in schemas.

Finally, we sometimes review dimension tables that contain rows for both atomic and hierarchical rollups, such as rows for both products and brands in the same dimension table. These dimensions typically have a telltale "level" attribute to distinguish between its base and summary rows. This pattern was prevalent and generally accepted decades ago prior to aggregate navigation capabilities. However, we discourage its continued use given the strong likelihood of user confusion and the risk of overcounting if the level indicator in every dimension is not constrained in every query.

Date Dimension

Every fact table should have at least one foreign key to an explicit date dimension. Design teams sometimes join a generic date dimension to a fact table because they know it's the most common dimension but then can't articulate what the date refers to, presenting challenges for the ETL team and business users alike. We encourage a meaningful date dimension table with robust date rollup and filter attributes.

Fixed Time Series Buckets Instead of Date Dimension

Designers sometimes avoid a date dimension table altogether by representing a time series of monthly buckets of facts on a single fact table row. Legacy operational systems may contain metric sets that are repeated 12 times on a single record to represent month 1, month 2, and so on. There are several problems with this approach. First, the hard-coded identity of the time slots is inflexible. When you fill up all the buckets, you are left with unpleasant choices. You could alter the table to expand the row. Otherwise, you could shift everything over by one column, dropping the oldest data,

but this wreaks havoc with existing query applications. The second problem with this approach is that all the attributes of the date are now the responsibility of the application, not the database. There is no date dimension in which to place calendar event descriptions for constraining. Finally, the fixed slot approach is inefficient if measurements are taken only in a particular time period, resulting in null columns in many rows. Instead, these recurring time buckets should be presented as separate rows in the fact table.

Degenerate Dimensions

Rather than treating operational transaction numbers such as the invoice or order number as degenerate dimensions, teams sometimes want to create a separate dimension table for the transaction number. In this case, attributes of the transaction number dimension include elements from the transaction header record, such as the transaction date and customer.

Remember, transaction numbers are best treated as degenerate dimensions. The transaction date and customer should be captured as foreign keys on the fact table, not as attributes in a transaction dimension. Be on the lookout for a dimension table that has as many (or nearly as many) rows as the fact table; this is a warning sign that there may be a degenerate dimension lurking within a dimension table.

Surrogate Keys

Instead of relying on operational keys or identifiers, we recommend the use of surrogate keys as the dimension tables' primary keys. The only permissible deviation from this guideline applies to the highly predictable and stable date dimension. If you are unclear about the reasons for pursuing this strategy, we suggest backtracking to Chapter 3: Retail Sales to refresh your memory.

Dimension Decodes and Descriptions

All identifiers and codes in the dimension tables should be accompanied by descriptive decodes. This practice often seems counterintuitive to experienced data modelers who have historically tried to reduce data redundancies by relying on look-up codes. In the dimensional model, dimension attributes should be populated with the values that business users want to see on BI reports and application pull-down menus. You need to dismiss the misperception that business users prefer to work with codes. To convince yourself, stroll down to their offices to see the decode listings filling their bulletin boards or lining their computer monitors. Most users do not memorize the codes outside of a few favorites. New hires are rendered helpless when assaulted with a lengthy list of meaningless codes.

The good news is that decodes can usually be sourced from operational systems with relatively minimal additional effort or overhead. Occasionally, the descriptions are not available from an operational system but need to be provided by business partners. In these cases, it is important to determine an ongoing maintenance strategy to maintain data quality.

Finally, project teams sometimes opt to embed labeling logic in the BI tool's semantic layer rather than supporting it via dimension table attributes. Although some BI tools provide the ability to decode within the query or reporting application, we recommend that decodes be stored as data elements instead. Applications should be data-driven to minimize the impact of decode additions and changes. Of course, decodes that reside in the database also ensure greater report labeling consistency because most organizations ultimately utilize multiple BI products.

Conformity Commitment

Last, but certainly not least, design teams must commit to using shared conformed dimensions across process-centric models. Everyone needs to take this pledge seriously. Conformed dimensions are absolutely critical to a robust data architecture that ensures consistency and integration. Without conformed dimensions, you inevitably perpetuate incompatible stovepipe views of performance across the organization. By the way, dimension tables should conform and be reused whether you drink the Kimball Kool-Aid or embrace a hub-and-spoke architectural alternative. Fortunately, operational master data management systems are facilitating the development and deployment of conformed dimensions.

Design Review Guidelines

Before diving into a review of the draft model in Figure 11-2, let's review some practical recommendations for conducting dimensional model design reviews. Proper advance preparation increases the likelihood of a successful review process. Here are some suggestions when setting up for a design review:

- **Invite the right players.** The modeling team obviously needs to participate, but you also want representatives from the BI development team to ensure that proposed changes enhance usability. Perhaps most important, it's critical that folks who are very knowledgeable about the business and their needs are sitting at the table. Although diverse perspectives should participate in a review, don't invite 25 people to the party.
- **Designate someone to facilitate the review.** Group dynamics, politics, and the design challenges will drive whether the facilitator should be a neutral resource or involved party. Regardless, their role is to keep the team on track toward a

common goal. Effective facilitators need the right mix of intelligence, enthusiasm, confidence, empathy, flexibility, assertiveness (and a sense of humor).

- **Agree on the review's scope.** Ancillary topics will inevitably arise during the review, but agreeing in advance on the scope makes it easier to stay focused on the task at hand.
- **Block time on everyone's calendar.** We typically conduct dimensional model reviews as a focused two day effort. The entire review team needs to be present for the full two days. Don't allow players to float in and out to accommodate other commitments. Design reviews require undivided attention; it's disruptive when participants leave intermittently.
- **Reserve the right space.** The same conference room should be blocked for the full two days. Optimally, the room should have a large white board; it's especially helpful if the white board drawings can be saved or printed. If a white board is unavailable, have flip charts on hand. Don't forget markers and tape; drinks and food also help.
- **Assign homework.** For example, ask everyone involved to make a list of their top five concerns, problem areas, or opportunities for improvement with the existing design. Encourage participants to use complete sentences when making their list so that it's meaningful to others. These lists should be sent to the facilitator in advance of the design review for consolidation. Soliciting advance input gets people engaged and helps avoid "group think" during the review.

After the team gathers to focus on the review, we recommend the following tactics:

- **Check attitudes at the door.** Although it's easier said than done, don't be defensive about prior design decisions. Embark on the review thinking change is possible; don't go in resigned to believing nothing can be done to improve the situation.
- **Ban technology unless needed for the review process.** Laptops and smartphones should also be checked at the door (at least figuratively). Allowing participants to check e-mail during the sessions is no different than having them leave to attend an alternative meeting.
- **Exhibit strong facilitation skills.** Review ground rules and ensure everyone is openly participating and communicating. The facilitator must keep the group on track and ban side conversations and discussions that are out of scope or spiral into the death zone.
- **Ensure a common understanding of the current model.** Don't presume everyone around the table already has a comprehensive perspective. It may be worthwhile to dedicate the first hour to walking through the current design and reviewing objectives before delving into potential improvements.

- **Designate someone to act as scribe.** He should take copious notes about both the discussions and decisions being made.
- **Start with the big picture.** Just as when you design from a blank slate, begin with the bus matrix. Focus on a single, high-priority business process, define its granularity and then move out to the corresponding dimensions. Follow this same "peeling back the layers of the onion" method with a design review, starting with the fact table and then tackling dimension-related issues. But don't defer the tough stuff to the afternoon of the second day.
- **Remind everyone that business acceptance is critical.** Business acceptance is the ultimate measure of DW/BI success. The review should focus on improving the business users' experience.
- **Sketch out sample rows with data values.** Viewing sample data during the review sessions helps ensure everyone has a common understanding of the recommended improvements.
- **Close the meeting with a recap.** Don't let participants leave the room without clear expectations about their assignments and due dates, along with an established time for the next follow-up.

After the team completes the design review meeting, here are a few recommendations to wrap up the process:

- **Assign responsibility for any remaining open issues.** Commit to wrestling these issues to the ground following the review, even though this can be challenging without an authoritative party involved.
- **Don't let the team's hard work gather dust.** Evaluate the cost/benefit for the potential improvements; some changes will be more painless (or painful) than others. Action plans for implementing the improvements then need to be developed.
- **Anticipate future reviews.** Plan to reevaluate models every 12 to 24 months. Try to view inevitable changes to the design as signs of success, rather than failure.

Draft Design Exercise Discussion

Now that you've reviewed the common dimensional modeling mistakes frequently encountered during design reviews, refer to the draft design in Figure 11-2. Several opportunities for improvement should immediately jump out at you.

The first thing to focus on is the grain of the fact table. The team stated the grain is one row for each bill each month. However, based on your understanding from the source system documentation and data profiling effort, the lowest level of billing data would be one row per service line on a bill. When you point this out, the team initially directs you to the bill dimension, which includes the service line number. However, when reminded that each service line has its own set of billing metrics, the team agrees the more appropriate grain declaration would be one row per service line per bill. The service line key is moved into the fact table as a foreign key to the service line dimension.

While discussing the granularity, the bill dimension is scrutinized, especially because the service line key was just moved into the fact table. As the draft model was originally drawn in Figure 11-2, every time a bill row is loaded into the fact table, a row also would be loaded into the bill dimension table. It doesn't take much to convince the team that something is wrong with this picture. Even with the modified granularity to include service line, you would still end up with nearly as many rows in both the fact and bill dimension tables because many customers are billed for one service line. Instead, the bill number should be treated as a degenerate dimension. At the same time, you move the bill date into the fact table and join it to a robust date dimension playing the role of bill date in this schema.

You've probably been bothered since first looking at the design by the double joins on the sales channel dimension table. The sales channel hierarchy has been unnecessarily snowflaked. You opt to collapse the hierarchy by including the sales channel identifiers (hopefully along with more meaningful descriptors) as additional attributes in the sales organization dimension table. In addition, you can eliminate the unneeded sales channel foreign key in the fact table.

The design inappropriately treats the rate plan type code as a textual fact. Textual facts are seldom a sound design choice. In this case study, the rate plan type code and its decode can be treated as rollup attributes in the rate plan dimension table.

The team spent some time discussing the relationship between the service line and the customer, sales organization, and rate plan dimensions. Because there is a single customer, sales organization, and rate plan associated with a service line number, the dimensions theoretically could be collapsed and modeled as service line attributes. However, collapsing the dimensions would result in a schema with just two dimensions: bill date and service line. The service line dimension already has millions of rows in it and is rapidly growing. In the end, you opt to treat the customer, sales organization, and rate plan as separate entities (or mini-dimensions) of the service line.

Surrogate keys are used inconsistently throughout the design. Many of the draft dimension tables use operational identifiers as primary keys. You encourage the team to implement surrogate keys for all the dimension primary keys and then reference them as fact table foreign keys.

The original design was riddled with operational codes and identifiers. Adding descriptive names makes the data more legible to the business users. If required by the business, the operational codes can continue to accompany the descriptors as dimension attributes.

Finally, you notice that there is a year-to-date metric stored in the fact table. Although the team felt this would enable users to report year-to-date figures more easily, in reality, year-to-date facts can be confusing and prone to error. You opt to remove the year-to-date fact. Instead, users can calculate year-to-date amounts on-the-fly by using a constraint on the year in the date dimension or leveraging the BI tool's capabilities.

After two exhausting days, the initial review of the design is complete. Of course, there's more ground to cover, including the handling of changes to the dimension attributes. In the meantime, everyone on the team agrees the revamped design, illustrated in Figure 11-3, is a vast improvement. You've earned your first week's pay.

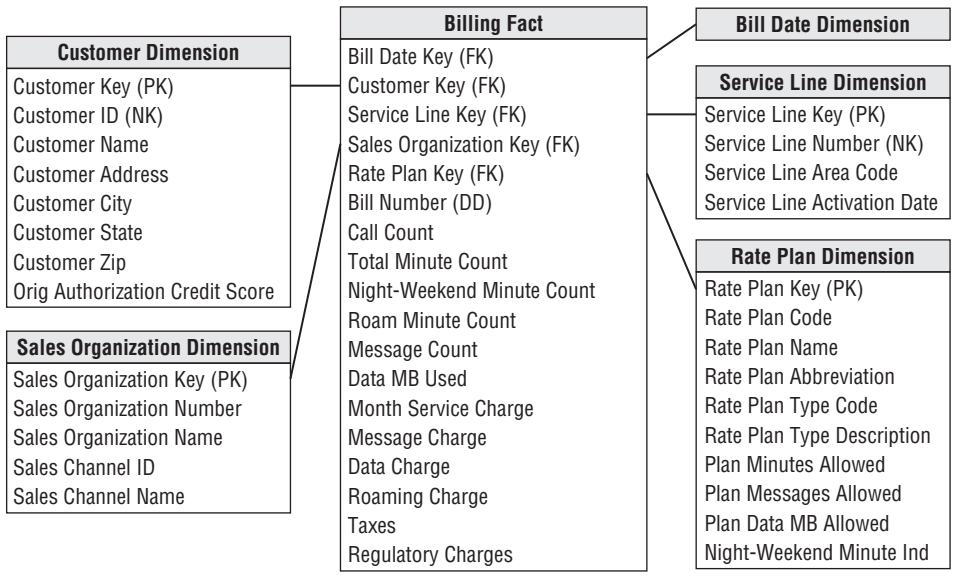

Figure 11-3: Draft schema following design review.

Remodeling Existing Data Structures

It's one thing to conduct a review and identify opportunities for improvement. However, implementing the changes might be easier said than done if the design has already been physically implemented.

For example, adding a new attribute to an existing dimension table feels like a minor enhancement. It is nearly pain-free if the business data stewards declare it to be a slowly changing dimension type 1 attribute. Likewise if the attribute is to be populated starting now with no attempt to backfill historically accurate values beyond a Not Available attribute value; note that while this tactic is relatively easy to implement, it presents analytic challenges and may be deemed unacceptable. But if the new attribute is a designated type 2 attribute with the requirement to capture historical changes, this seemingly simple enhancement just got much more complicated. In this scenario, rows need to be added to the dimension table to capture the historical changes in the attribute, along with the other dimension attribute changes. Some fact table rows then need to be recast so the appropriate dimension table row is associated with the fact table's event. This most robust approach consumes surprisingly more effort than you might initially imagine.

Much less surprising is the effort required to take an existing dimensional model and convert it into a structure that leverages newly created conformed dimensions. As discussed in Chapter 4: Inventory, at a minimum, the fact table's rows must be completely reprocessed to reference the conformed dimension keys. The task is obviously more challenging if there are granularity or other major issues.

In addition to thinking about the data-centric challenges of retrofitting existing data structures, there are also unwanted ripples in the BI reporting and analytic applications built on the existing data foundation. Using views to buffer the BI applications from the physical data structures provides some relief, but it's typically not adequate to avoid unpleasant whipsawing in the BI environment.

When considering enhancements to existing data structures, you must evaluate the costs of tackling the changes alongside the perceived benefits. In many cases, you'll determine improvements need to be made despite the pain. Similarly, you may determine the best approach is to decommission the current structures to put them out of their misery and tackle the subject area with a fresh slate. Finally, in some situations, the best approach is to simply ignore the suboptimal data structures because the costs compared to the potential benefits don't justify the remodeling and schema improvement effort. Sometimes, the best time to consider a remodeling effort is when other changes, such as a source system conversion or migration to a new BI tool standard, provide a catalyst.

Geographic Location Dimension

Let's shift gears and presume you work for a phone company with land lines tied to a specific physical location. The telecommunications and utilities industries have a very well-developed notion of location. Many of their dimensions contain a precise geographic location as part of the attribute set. The location may be resolved to a physical street, city, state, ZIP code, latitude, and longitude. Latitude and longitude geo-coding can be leveraged for geospatial analysis and map-centric visualization.

Some designers imagine a single master location table where address data is standardized and then the location dimension is attached as an outrigger to the service line telephone number, equipment inventory, network inventory (including poles and switch boxes), real estate inventory, service location, dispatch location, right of way, and customer entities. In this scenario, each row in the master location table is a specific point in space that rolls up to every conceivable geographic grouping.

Standardizing the attributes associated with points in space is valuable. However, this is a back room ETL task; you don't need to unveil the single resultant table containing all the addresses the organization interacts with to the business users. Geographic information is naturally handled as attributes within multiple dimensions, not as a standalone location dimension or outrigger. There is typically little overlap between the geographic locations embedded in various dimensions. You would pay a performance price for consolidating all the disparate addresses into a single dimension.

Operational systems often embrace data abstraction, but you should typically avoid generic abstract dimensions, such as a generalized location dimension in the DW/BI presentation area because they negatively impact the ease-of-use and query performance objectives. These structures are more acceptable behind the scenes in the ETL back room.

Summary

This chapter provided the opportunity to conduct a design review using an example case study. It provided recommendations for conducting effective design reviews, along with a laundry list of common design flaws to scout for when performing a review. We encourage you to use this laundry list to review your own draft schemas when searching for potential improvements.