8 Customer Relationship Management

Long before the *customer relationship management* (*CRM*) buzzword existed, organizations were designing and developing customer-centric dimensional models to better understand their customers' behavior. For decades, these models were used to respond to management's inquiries about which customers were solicited, who responded, and what was the magnitude of their response. The business value of understanding the full spectrum of customers' interactions and transactions has propelled CRM to the top of the charts. CRM not only includes familiar residential and commercial customers, but also citizens, patients, students, and many other categories of people and organizations whose behavior and preferences are important. CRM is a mission-critical business strategy that many view as essential to an organization's survival.

In this chapter we start with a CRM overview, including its operational and analytic roles. We then introduce the basic design of the customer dimension, including common attributes such as dates, segmentation attributes, repeated contact roles, and aggregated facts. We discuss customer name and address parsing, along with international considerations. We remind you of the challenges of modeling complex hierarchies when we describe various kinds of customer hierarchies.

Chapter 8 discusses the following concepts:

- CRM overview
- Customer name and address parsing, including international considerations
- Handling of dates, aggregated facts, and segmentation behavior attributes and scores in a customer dimension
- Outriggers for low cardinality attributes
- Bridge tables for sparse attributes, along with trade-offs of bridge tables versus a positional design
- Bridge tables for multiple customer contacts
- Behavior study groups to capture customer cohort groups

- Step dimensions to analyze sequential customer behavior
- Timespan fact tables with effective and expiration dates
- Embellishing fact tables with dimensions for satisfaction or abnormal scenarios
- Integrating customer data via master data management or partial conformity during the downstream ETL processing
- Warnings about fact-to-fact table joins
- Reality check on real time, low latency requirements

Because this chapter's customer-centric modeling issues and patterns are relevant across industries and functional areas, we have not included a bus matrix.

CRM Overview

Regardless of the industry, organizations have flocked to the concept of CRM. They've jumped on the bandwagon in an attempt to migrate from a product-centric orientation to one that is driven by customer needs. Although all-encompassing terms such as customer relationship management sometimes seem ambiguous and/or overly ambitious, the premise behind CRM is far from rocket science. It's based on the simple notion that the better you know your customers, the better you can maintain long-lasting, valuable relationships with them. The goal of CRM is to maximize relationships with your customers over their lifetime. It entails focusing all aspects of the business, from marketing, sales, operations, and service, on establishing and sustaining mutually beneficial customer relations. To do so, the organization must develop a single, integrated view of each customer.

CRM promises significant returns for organizations that embrace it, both for increased revenue and operational efficiencies. Switching to a customer-driven perspective can lead to increased sales effectiveness and closure rates, revenue growth, enhanced sales productivity at reduced cost, improved customer profitability margins, higher customer satisfaction, and increased customer retention. Ultimately, every organization wants more loyal, more profitable customers. As it often requires a sizeable investment to attract new customers, you can't afford to have the profitable ones leave.

In many organizations, the view of the customer varies depending on the product line business unit, business function, and/or geographic location. Each group may use different customer data in different ways with different results. The evolution from the existing silos to a more integrated perspective obviously requires organizational commitment. CRM is like a stick of dynamite that knocks down the silo walls. It requires the right integration of business processes, people resources, and application technology to be effective.

Over the past decade, the explosive growth of social media, location tracking technology, network usage monitoring, multimedia applications, and sensor networks

has provided an ocean of customer behavioral data that even Main Street enterprises recognize as providing actionable insights. Although much of this data lies outside the comfort zone of relational databases, the new "big data" techniques can bring this data back into the DW/BI fold. Chapter 21: Big Data Analytics discusses the best practices for bringing this new kind of big data into the DW/BI environment. But setting aside the purely technological challenges, the real message is the need for profound integration. You must step up to the challenge of integrating as many as 100 customer-facing data sources, most of which are external. These data sources are at different grains, have incompatible customer attributes, and are not under your control. Any questions?

Because it is human nature to resist change, it comes as no surprise that people-related issues often challenge CRM implementations. CRM involves brand new ways of interacting with customers and often entails radical changes to the sales channels. CRM requires new information flows based on the complete acquisition and dissemination of customer "touch point" data. Often organization structures and incentive systems are dramatically altered.

In Chapter 17: Kimball DW/BI Lifecycle Overview, we'll stress the importance of having support from both senior business and IT management for a DW/BI initiative. This advice also applies to a CRM implementation because of its cross-functional focus. CRM requires a clear business vision. Without business strategy, buy-in, and authorization to change, CRM becomes an exercise in futility. Neither IT nor the business community can successfully implement CRM on its own; CRM demands a joint commitment of support.

Operational and Analytic CRM

It could be said that CRM suffers from a split personality syndrome because it needs to address both operational and analytic requirements. Effective CRM relies on the collection of data at every interaction you have with a customer and then leveraging that breadth of data through analysis.

On the operational front, CRM calls for the synchronization of customer-facing processes. Often operational systems must either be updated or supplemented to coordinate across sales, marketing, operations, and service. Think about all the customer interactions that occur during the purchase and usage of a product or service, from the initial prospect contact, quote generation, purchase transaction, fulfillment, payment transaction, and on-going customer service. Rather than thinking about these processes as independent silos (or multiple silos varying by product line), the CRM mindset is to integrate these customer activities. Key customer metrics and characteristics are collected at each touch point and made available to the others.

As data is created on the operational side of the CRM equation, you obviously need to store and analyze the historical metrics resulting from the customer

interaction and transaction systems. Sounds familiar, doesn't it? The DW/BI system sits at the core of CRM. It serves as the repository to collect and integrate the breadth of customer information found in the operational systems, as well as from external sources. The data warehouse is the foundation that supports the panoramic 360-degree view of your customers.

Analytic CRM is enabled via accurate, integrated, and accessible customer data in the DW/BI system. You can measure the effectiveness of decisions made in the past to optimize future interactions. Customer data can be leveraged to better identify up-sell and cross-sell opportunities, pinpoint inefficiencies, generate demand, and improve retention. In addition, the historical, integrated data can be leveraged to generate models or scores that close the loop back to the operational world. Recalling the major components of a DW/BI environment from Chapter 1: Data Warehousing, Business Intelligence, and Dimensional Modeling Primer, you can envision the model results pushed back to where the relationship is operationally managed (such as the rep, call center, or website), as illustrated in Figure 8-1. The model output can translate into specific proactive or reactive tactics recommended for the next point of customer contact, such as the appropriate next product offer or anti-attrition response. The model results are also retained in the DW/BI environment for subsequent analysis.

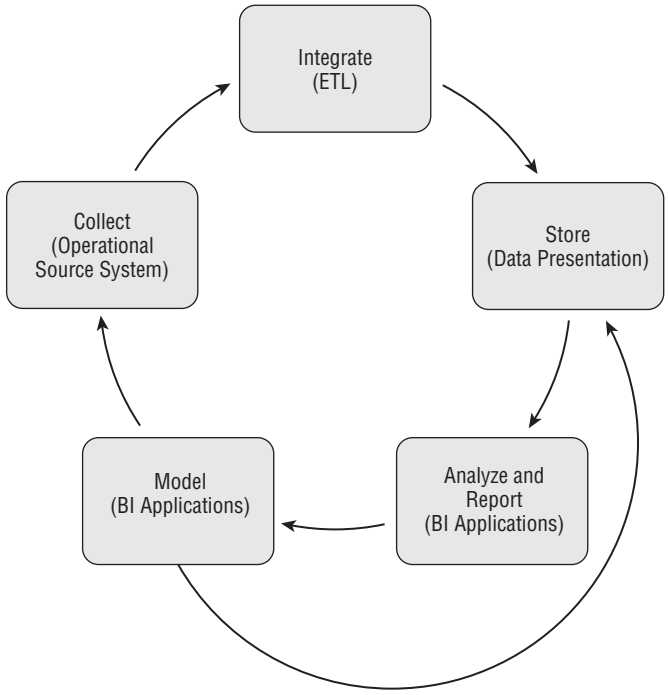


Figure 8-1: Closed loop analytic CRM.

In other situations, information must feed back to the operational website or call center systems on a more real-time basis. In this case, the closed loop is much tighter than Figure 8-1 because it's a matter of collection and storage, and then feedback to the collection system. Today's operational processes must combine the current view with a historical view, so a decision maker can decide, for example, whether to grant credit to a customer in real time, while considering the customer's lifetime history. But generally, the integration requirements for operational CRM are not as far reaching as for analytic CRM.

Obviously, as the organization becomes more centered on the customer, so must the DW/BI system. CRM will inevitably drive change in the data warehouse. DW/BI environments will grow even more rapidly as you collect more and more information about your customers. ETL processes will grow more complicated as you match and integrate data from multiple sources. Most important, the need for a conformed customer dimension becomes even more paramount.

Customer Dimension Attributes

The conformed customer dimension is a critical element for effective CRM. A well-maintained, well-deployed conformed customer dimension is the cornerstone of sound CRM analysis.

The customer dimension is typically the most challenging dimension for any DW/BI system. In a large organization, the customer dimension can be extremely deep (with many millions of rows), extremely wide (with dozens or even hundreds of attributes), and sometimes subject to rapid change. The biggest retailers, credit card companies, and government agencies have monster customer dimensions whose size exceeds 100 million rows. To further complicate matters, the customer dimension often represents an amalgamation of data from multiple internal and external source systems.

In this next section, we focus on numerous customer dimension design considerations. We'll begin with name/address parsing and other common customer attributes, including coverage of dimension outriggers, and then move on to other interesting customer attributes. Of course, the list of customer attributes is typically quite lengthy. The more descriptive information you capture about your customers, the more robust the customer dimension, and the more interesting the analyses.

Name and Address Parsing

Regardless of whether you deal with individual human beings or commercial entities, customers' name and address attributes are typically captured. The operational handling of name and address information is usually too simplistic to be very useful

in the DW/BI system. Many designers feel a liberal design of general purpose columns for names and addresses, such as Name-1 through Name-3 and Address-1 through Address-6, can handle any situation. Unfortunately, these catchall columns are virtually worthless when it comes to better understanding and segmenting the customer base. Designing the name and location columns in a generic way can actually contribute to data quality problems. Consider the sample design in Figure 8-2 with general purpose columns.

Column	Sample Data Value
Name	Ms. R. Jane Smith, Atty
Address 1	123 Main Rd, North West, Ste 100A
Address 2	PO Box 2348
City	Kensington
State	Ark.
ZIP Code	88887-2348
Phone Number	888-555-3333 x776 main, 555-4444 fax

Figure 8-2: Sample customer name/address data in overly general columns.

In this design, the name column is far too limited. There is no consistent mechanism for handling salutations, titles, or suffixes. You can't identify what the person's first name is, or how she should be addressed in a personalized greeting. If you look at additional sample data from this operational system, you would potentially find multiple customers listed in a single name attribute. You might also find additional descriptive information in the name column, such as Confidential, Trustee, or UGMA (Uniform Gift to Minors Act).

In the sample address attributes, inconsistent abbreviations are used in various places. The address columns may contain enough room for any address, but there is no discipline imposed by the columns that can guarantee conformance with postal authority regulations or support address matching and latitude/longitude identification.

Instead of using a few, general purpose columns, the name and location attributes should be broken down into as many elemental parts as possible. The extract process needs to perform significant parsing on the original dirty names and addresses. After the attributes have been parsed, they can be standardized. For example, Rd would become Road and Ste would become Suite. The attributes can also be verified, such as verifying the ZIP code and associated state combination is correct. Fortunately, there are name and address data cleansing and scrubbing tools available in the market to assist with parsing, standardization, and verification.

A sample set of name and location attributes for individuals in the United States is shown in Figure 8-3. Every attribute is filled in with sample data to make the design clearer, but no single real instance would look like this. Of course, the business data governance representatives should be involved in determining the analytic value of these parsed data elements in the customer dimension.

Column	Sample Data Value
Salutation	Ms.
Informal Greeting Name	Jane
Formal Greeting Name	Ms. Smith
First and Middle Names	R. Jane
Surname	Smith
Suffix	Jr.
Ethnicity	English
Title	Attorney
Street Number	123
Street Name	Main
Street Type	Road
Street Direction	North West
City	Kensington
District	Cornwall
Second District	Berkeleyshire
State	Arkansas
Region	South
Country	United States
Continent	North America
Primary Postal Code	88887
Secondary Postal Code	2348
Postal Code Type	United States
Office Telephone Country Code	1
Office Telephone Area Code	888
Office Telephone Number	5553333
Office Extension	776
Mobile Telephone Country Code	1
Mobile Telephone Area Code	509
Mobile Telephone Number	5554444
E-mail	RJSmith@ABCGenIntl.com
Web Site	www.ABCGenIntl.com
Public Key Authentication	X.509
Certificate Authority	Verisign
Unique Individual Identifier	7346531

Figure 8-3: Sample customer name/address data with parsed name and address elements.

Commercial customers typically have multiple addresses, such as physical and shipping addresses; each of these addresses would follow much the same logic as the address structure shown in Figure 8-3.

International Name and Address Considerations

International display and printing typically requires representing foreign characters, including not just the accented characters from western European alphabets, but also Cyrillic, Arabic, Japanese, Chinese, and dozens of other less familiar writing systems. It is important to understand this is not a font problem. This is a character set problem. A font is simply an artist's rendering of a set of characters. There are hundreds of fonts available for standard English, but standard English has a relatively small character set that is enough for anyone's use unless you are a professional typographer. This small character set is usually encoded in *American Standard Code for Information Interchange* (*ASCII*), which is an 8-bit encoding that has a maximum of 255 possible characters. Only approximately 100 of these 255 characters have a standard interpretation that can be invoked from a normal English keyboard, but this is usually enough for English speaking computer users. It should be clear that ASCII is woefully inadequate for representing the thousands of characters needed for non-English writing systems.

An international body of system architects, the Unicode Consortium, defined a standard known as Unicode for representing characters and alphabets in almost all the world's languages and cultures. Their work can be accessed on the web at `www.unicode.org`. The Unicode Standard, version 6.2.0 has defined specific interpretations for 110,182 possible characters and now covers the principal written languages of the Americas, Europe, the Middle East, Africa, India, Asia, and Pacifica. Unicode is the foundation you must use for addressing international character sets.

But it is important to understand that implementing Unicode solutions is done in the foundation layers of your systems. First, the operating system must be Unicode-compliant. Fortunately, the most current releases of all the major operating systems are Unicode-compliant.

Above the operating system, all the devices that capture, store, transmit, and print characters must be Unicode-compliant. Data warehouse back room tools must be Unicode-compliant, including sort packages, programming languages, and automated ETL packages. Finally, the DW/BI applications, including database engines, BI application servers and their report writers and query tools, web servers, and browsers must all be Unicode-compliant. The DW/BI architect should not only talk to the vendors of each package in the data pipeline, but also should conduct various end-to-end tests. Capture some names and addresses with Unicode characters at the data capture screens of one of the legacy applications, and send them through the system. Get them to print out of a final report or a final browser window from

the DW/BI system and see if the special characters are still there. That simple test will cut through a lot of the confusion. Note that even when you do this, the same character, such as an a-umlaut, sorts differently in different countries such as Norway and Germany. Even though you can't solve all the variations in international collating sequences, at least both the Norwegians and the Germans will agree that the character is an a-umlaut.

Customer geographic attributes become more complicated if you deal with customers from multiple countries. Even if you don't have international customers, you may need to contend with international names and addresses somewhere in the DW/BI system for international suppliers and human resources personnel records.

NOTE Customer dimensions sometimes include a full address block attribute. This is a specially crafted column that assembles a postally-valid address for the customer including mail stop, ZIP code, and other attributes needed to satisfy postal authorities. This attribute is useful for international locations where addresses have local idiosyncrasies.

International DW/BI Goals

After committing to a Unicode foundation, you need to keep the following goals in mind, in addition to the name and address parsing requirements discussed earlier:

- **Universal and consistent.** As they say, in for a penny, in for a pound. If you are going to design a system for international use, you want it to work around the world. You need to think carefully if BI tools are to produce translated versions of reports in many languages. It may be tempting to provide translated versions of dimensions for each language, but translated dimensions give rise to some subtle problems.
    - Sorting sequences will be different, so either the reports will be sorted differently or all reports except those in the "root" language will appear to be unsorted.
    - If the attribute cardinalities are not faithfully preserved across languages, then either group totals will not be the same across reports, or some groups in various languages will contain duplicated row headers that look like mistakes. To avoid the worst of these problems, you should translate dimensions after the report is run; the report first needs to be produced in a single root language, and then the report face needs to be translated into the intended target languages.
    - All the BI tool messages and prompts need to be translated for the benefit of the business user. This process is known as *localization* and is further discussed in Chapter 12: Transportation.

- **End-to-end data quality and downstream compatibility.** The data warehouse cannot be the only step in the data pipeline that worries about the integrity of international names and addresses. A proper design requires support from the first step of capturing the name and the address, through the data cleaning and storage steps, to the final steps of performing geographic and demographic analysis and printing reports.
- **Cultural correctness.** In many cases, foreign customers and partners will see the results from your DW/BI system in some form. If we don't understand which name is a first name and which is a last name, and if you don't understand how to refer to a person, you run the risk of insulting these individuals, or at the very least, looking stupid. When outputs are punctuated improperly, or misspelled, your foreign customers and partners will wish they were doing business with a local company, rather than you.
- **Real-time customer response.** DW/BI systems can play an operational role by supporting real-time customer response systems. A customer service representative may answer the telephone and may have 5 seconds or less to wait for a greeting to appear on the screen that the data warehouse recommends using with the customer. The greeting may include a proper salutation and a proper use of the customer's title and name. This greeting represents an excellent use of a *hot response cache* that contains precalculated responses for each customer.
- **Other kinds of addresses.** We are in the midst of a revolution in communication and networking. If you are designing a system for identifying international names and addresses, you must anticipate the need to store electronic names, security tokens, and internet addresses.

Similar to international addresses, telephone numbers must be presented differently depending on where the phone call originates. You need to provide attributes to represent the complete foreign dialing sequence, complete domestic dialing sequence, and local dialing sequence. Unfortunately, complete foreign dialing sequences vary by origin country.

Customer-Centric Dates

Customer dimensions often contains dates, such as the date of the first purchase, date of last purchase, and date of birth. Although these dates initially may be SQL date type columns, if you want to summarize these dates by your unique calendar attributes, such as seasons, quarters, and fiscal periods, the dates should be changed to foreign key references to the date dimension. You need to be careful that all such dates fall within the span of the corporate date dimension. These date dimension roles are declared as semantically distinct views, such as a First Purchase

Date dimension table with unique column labels. The system behaves as if there is another physical date table. Constraints on any of these tables have nothing to do with constraints on the primary date dimension table. This design, as shown in Figure 8-4, is an example of a dimension outrigger, which is discussed in the section "Outrigger for Low Cardinality Attribute Set."

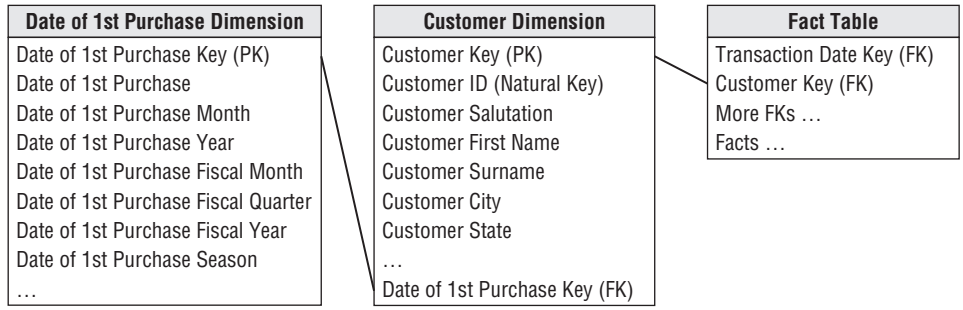


Figure 8-4: Date dimension outrigger.

Aggregated Facts as Dimension Attributes

Business users are often interested in constraining the customer dimension based on aggregated performance metrics, such as filtering on all customers who spent more than a certain dollar amount during last year. Or to make matters worse, perhaps they want to constrain based on how much the customer has purchased in a lifetime. Providing aggregated facts as dimension attributes is sure to be a crowd-pleaser with the business users. They could issue a query to identify all customers who satisfied the spending criteria and then issue another fact query to analyze the behavior for that customer dimension subset. But rather than all that, you can instead store an aggregated fact as a dimension attribute. This allows business users to simply constrain on the spending attribute just like they might on a geographic attribute. These attributes are meant to be used for constraining and labeling; they're not to be used in numeric calculations. Although there are query usability and performance advantages of storing these attributes, the main burden falls on the back room ETL processes to ensure the attributes are accurate, up-to-date, and consistent with the actual fact rows. These attributes can require significant care and feeding. If you opt to include some aggregated facts as dimension attributes, be certain to focus on those that will be frequently used. Also strive to minimize the frequency with which these attributes need to be updated. For example, an attribute for last year's spending would require much less maintenance than one providing year-to-date behavior. Rather than storing attributes down to the specific dollar value, they are sometimes

replaced (or supplemented) with more meaningful descriptive values, such as High Spender as discussed in the next section. These descriptive values minimize your vulnerability that the numeric attributes might not tie back to the appropriate fact tables. In addition, they ensure that all users have a consistent definition for high spenders, for example, rather than resorting to their own individual business rules.

Segmentation Attributes and Scores

Some of the most powerful attributes in a customer dimension are segmentation classifications. These attributes obviously vary greatly by business context. For an individual customer, they may include:

- Gender
- Ethnicity
- Age or other life stage classifications
- Income or other lifestyle classifications
- Status (such as new, active, inactive, and closed)
- Referring source
- Business-specific market segment (such as a preferred customer identifier)

Similarly, many organizations score their customers to characterize them. Statistical segmentation models typically generate these scores which cluster customers in a variety of ways, such as based on their purchase behavior, payment behavior, propensity to churn, or probability to default. Each customer is tagged with a resultant score.

Behavior Tag Time Series

One popular approach for scoring and profiling customers looks at the recency (R), frequency (F), and intensity (I) of the customer's behavior. These are known as the *RFI measures*; sometimes intensity is replaced with monetary (M), so it's also known as RFM. Recency is how many days has it been since the customer last ordered or visited your site. Frequency is how many times the customer has ordered or visited, typically in the past year. And intensity is how much money the customer has spent over the same time period. When dealing with a large customer base, every customer's behavior can be modeled as a point in an RFI cube, as depicted in Figure 8-5. In this figure, the scales along each axis are quintiles, from 1 to 5, which spread the actual values into even groups.

If you have millions of points in the cube, it becomes difficult to see meaningful clusters of these points. This is a good time to ask a data mining professional where the meaningful clusters are. The data mining professional may come back with a list of behavior tags like the following, which is drawn from a slightly more complicated scenario that includes credit behavior and returns:

A: High volume repeat customer, good credit, few product returns
B: High volume repeat customer, good credit, many product returns
C: Recent new customer, no established credit pattern
D: Occasional customer, good credit
E: Occasional customer, poor credit
F: Former good customer, not seen recently
G: Frequent window shopper, mostly unproductive
H: Other

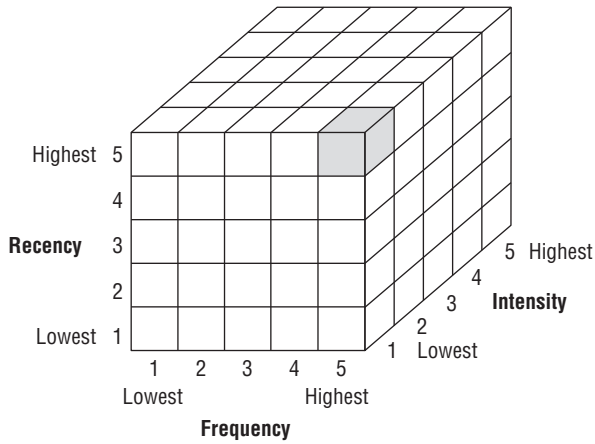


Figure 8-5: Recency, frequency, intensity (RFI) cube.

Now you can look at the customers' time series data and associate each customer in each reporting period with the nearest cluster. The data miner can help do this. Thus, the last 10 observations of a customer named John Doe could look like:

John Doe: C C C D D A A A B B

This time series of behavior tags is unusual because although it comes from a regular periodic measurement process, the observed "values" are textual. The behavior tags are not numeric and cannot be computed or averaged, but they can be queried. For example, you may want to find all the customers who were an A sometime in the fifth, fourth, or third prior period and were a B in the second or first prior period. Perhaps you are concerned by progressions like this and fear losing a valuable customer because of the increasing number of returns.

Behavior tags should not be stored as regular facts. The main use of behavior tags is formulating complex query patterns like the example in the previous paragraph. If the behavior tags were stored in separate fact rows, such querying would be extremely difficult, requiring a cascade of correlated subqueries. The recommended way to handle behavior tags is to build an explicit time series of attributes in the customer dimension. This is another example of a positional design. BI interfaces

are simple because the columns are in the same table, and performance is good because you can build bitmapped indexes on them.

In addition to the separate columns for each behavior tag time period, it would be a good idea to create a single attribute with all the behavior tags concatenated together, such as CCCDDAAABB. This column would support wild card searches for exotic patterns, such as "D followed by a B."

NOTE In addition to the customer dimension's time series of behavior tags, it would be reasonable to include the contemporary behavior tag value in a mini-dimension to analyze facts by the behavior tag in effect when the fact row was loaded.

Relationship Between Data Mining and DW/BI System

The data mining team can be a great client of the data warehouse, and especially great users of customer behavior data. However, there can be a mismatch between the velocity that the data warehouse can deliver data and the velocity that the data miners can consume data. For example, a decision tree tool can process hundreds of records per second, but a big drill-across report that produces "customer observations" can never deliver data at such speeds. Consider the following seven-way drill across a report that might produce millions of customer observations from census, demographic, external credit, internal credit, purchases, returns, and website data:

```
SELECT Customer Identifier, Census Tract, City, County, State,
  Postal Code, Demographic Cluster, Age, Sex, Marital Status,
  Years of Residency, Number of Dependents, Employment Profile,
  Education Profile, Sports Magazine Reader Flag,
  Personal Computer Owner Flag, Cellular Telephone Owner Flag,
  Current Credit Rating, Worst Historical Credit Rating,
  Best Historical Credit Rating, Date First Purchase,
  Date Last Purchase, Number Purchases Last Year,
  Change in Number Purchases vs. Previous Year,
  Total Number Purchases Lifetime, Total Value Purchases Lifetime,
  Number Returned Purchases Lifetime, Maximum Debt,
  Average Age Customer's Debt Lifetime, Number Late Payments,
  Number Fully Paid, Times Visited Web Site,
  Change in Frequency of Web Site Access,
  Number of Pages Visited Per Session,
  Average Dwell Time Per Session, Number Web Product Orders,
  Value Web Product Orders, Number Web Site Visits to Partner Web
  Sites, Change in Partner Web Site Visits
FROM *** WHERE *** ORDER BY *** GROUP BY ***
```

Data mining teams would love this data! For example a big file of millions of these observations could be analyzed by a decision tree tool where the tool is "aimed" at the Total Value Purchases Lifetime column, which is highlighted above. In this analysis, the decision tree tool would determine which of the other columns "predict the variance" of the target field. Maybe the answer is Best Historical Credit Rating and Number of Dependents. Armed with this answer, the enterprise now has a simple way to predict who is going to be a good lifetime customer, without needing to know all the other data content.

But the data mining team wants to use these observations over and over for different kinds of analyses perhaps with neural networks or case-based reasoning tools. Rather than producing this answer set on demand as a big, expensive query, this answer set should be written to a file and given to the data mining team to analyze on its servers.

Counts with Type 2 Dimension Changes

Businesses frequently want to count customers based on their attributes without joining to a fact table. If you used type 2 to track customer dimension changes, you need to be careful to avoid overcounting because you may have multiple rows in the customer dimension for the same individual. Doing a `COUNT DISTINCT` on a unique customer identifier is a possibility, assuming the attribute is indeed unique and durable. A current row indicator in the customer dimension is also helpful to do counts based on the most up-to-date descriptive values for a customer.

Things get more complicated if you need to do a customer count at a given historical point in time using effective and expiration dates in the customer dimension. For example, if you need to know the number of customers you had at the beginning of 2013, you could constrain the row effective date <= '1/1/2013' and row expiration date >= '1/1/2013' to restrict the result set to only those rows that were valid on 1/1/2013. Note the comparison operators are dependent on the business rules used to set the row effective/expiration dates. In this example, the row expiration date on the no longer valid customer row is 1 day less than the effective date on the new row.

Outrigger for Low Cardinality Attribute Set

In Chapter 3: Retail Sales, we encouraged designers to avoid snowflaking where low cardinality columns in the dimension are removed to separate normalized tables, which then link back into the original dimension table. Generally, snowflaking is not recommended in a DW/BI environment because it almost always makes the user presentation more complex, in addition to negatively impacting browsing performance. In spite of this prohibition against snowflaking, there are some special

situations in which it is permissible to build a dimension outrigger that begins to look like a snowflaked table.

In Figure 8-6, the dimension outrigger is a set of data from an external data provider consisting of 150 demographic and socio-economic attributes regarding the customers' county of residence. The data for all customers residing in a given county is identical. Rather than repeating this large block of data for every customer within a county, opt to model it as an outrigger. There are several reasons for bending the "no snowflake" rule. First, the demographic data is available at a significantly different grain than the primary dimension data and it's not as analytically valuable. It is loaded at different times than the rest of the data in the customer dimension. Also, you do save significant space in this case if the underlying customer dimension is large. If you have a query tool that insists on a classic star schema with no snowflakes, the outrigger can be hidden under a view declaration.

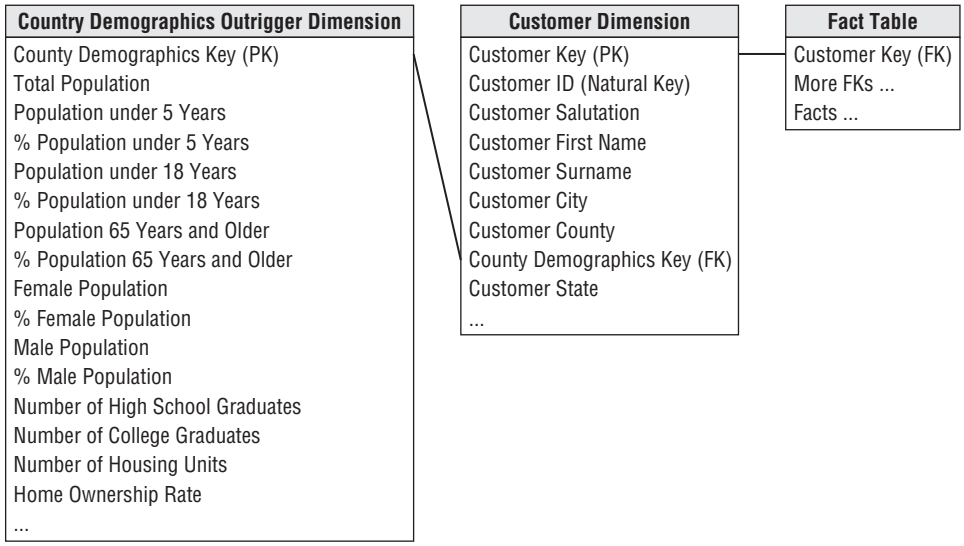


Figure 8-6: Dimension outrigger for cluster of low cardinality attributes.

WARNING Dimension outriggers are permissible, but they should be the exception rather than the rule. A red warning flag should go up if your design is riddled with outriggers; you may have succumbed to the temptation to overly normalize the design.

Customer Hierarchy Considerations

One of the most challenging aspects of dealing with commercial customers is modeling their internal organizational hierarchy. Commercial customers often have a

nested hierarchy of entities ranging from individual locations or organizations up through regional offices, business unit headquarters, and ultimate parent companies. These hierarchical relationships may change frequently as customers reorganize themselves internally or are involved in acquisitions and divestitures.

NOTE In Chapter 7: Accounting, we described how to handle fixed hierarchies, slightly variable hierarchies, and ragged hierarchies of indeterminate depth. Chapter 7 focuses on financial cost center rollups, but the techniques are exactly transferrable to customer hierarchies. If you skipped Chapter 7, you need to backtrack to read that chapter to make sense of the following recommendations.

Although relatively uncommon, the lucky ones amongst us sometimes are confronted with a customer hierarchy that has a highly predictable fixed number of levels. Suppose you track a maximum of three rollup levels, such as the ultimate corporate parent, business unit headquarters, and regional headquarters. In this case, you have three distinct attributes in the customer dimension corresponding to these three levels. For commercial customers with complicated organizational hierarchies, you'd populate all three levels to appropriately represent the three different entities involved at each rollup level. This is the fixed depth hierarchy approach from Chapter 7.

By contrast, if another customer had a mixture of one, two, and three level organizations, you'd duplicate the lower-level value to populate the higher-level attributes. In this way, all regional headquarters would sum to the sum of all business unit headquarters, which would sum to the sum of all ultimate corporate parents. You can report by any level of the hierarchy and see the complete customer base represented. This is the slightly variable hierarchy approach.

But in many cases, complex commercial customer hierarchies are ragged hierarchies with an indeterminate depth, so you must use a ragged hierarchy modeling technique, as described in Chapter 7. For example, if a utility company is devising a custom rate plan for all the utility consumers that are part of a huge customer with many levels of offices, branch locations, manufacturing locations, and sales locations, you cannot use a fixed hierarchy. As pointed out in Chapter 7, the worst design is a set of generic levels named such as Level-1, Level-2, and so on. This makes for an unusable customer dimension because you don't know how to constrain against these levels when you have a ragged hierarchy of indeterminate depth.

Bridge Tables for Multivalued Dimensions

A fundamental tenet of dimensional modeling is to decide on the grain of the fact table, and then carefully add dimensions and facts to the design that are true to the grain. For example, if you record customer purchase transactions, the grain of

the individual purchase is natural and physically compelling. You do not want to change that grain. Thus you normally require any dimension attached to this fact table to take on a single value because then there's a clean single foreign key in the fact table that identifies a single member of the dimension. Dimensions such as the customer, location, product or service, and time are always single valued. But you may have some "problem" dimensions that take on multiple values at the grain of the individual transaction. Common examples of these multivalued dimensions include:

- Demographic descriptors drawn from a multiplicity of sources
- Contact addresses for a commercial customer
- Professional skills of a job applicant
- Hobbies of an individual
- Diagnoses or symptoms of a patient
- Optional features for an automobile or truck
- Joint account holders in a bank account
- Tenants in a rental property

When faced with a multivalued dimension, there are two basic choices: a positional design or bridge table design. Positional designs are very attractive because the multivalued dimension is spread out into named columns that are easy to query. For example, if modeling the hobbies of an individual as previously mentioned, you could have a hobby dimension with named columns for all the hobbies gathered from your customers, including stamp collecting, coin collecting, astronomy, photography, and many others! Immediately you can see the problem. The positional design approach isn't very scalable. You can easily run out of columns in your database, and it is awkward to add new columns. Also if you have a column for every possible hobby, then any single individual's hobby dimension row will contain mostly null values.

The bridge table approach to multivalued dimensions is powerful but comes with a big compromise. The bridge table removes the scalability and null value objections because rows in the bridge table exist only if they are actually needed, and you can add hundreds or even thousands of hobbies in the previous example. But the resulting table design requires a complex query that must be hidden from direct view by the business users.

WARNING Be aware that complex queries using bridge tables may require SQL that is beyond the normal reach of BI tools.

In the next two sections, we illustrate multivalued bridge table designs that fit with the customer-centric topics of this chapter. We will revisit multivalued bridges in Chapter 9: Human Resources Management, Chapter 10: Financial Services, Chapter 13: Education, Chapter 14: Healthcare, and Chapter 16: Insurance. We'll then describe how to build these bridges in Chapter 19: ETL Subsystems and Techniques.

Bridge Table for Sparse Attributes

Organizations are increasingly collecting demographics and status information about their customers, but the traditional fixed column modeling approach for handling these attributes becomes difficult to scale with hundreds of attributes.

Positional designs have a named column for each attribute. BI tool interfaces are easy to construct for positional attributes because the named columns are easily presented in the tool. Because many columns contain low cardinality contents, the query performance using these attributes can be very good if bitmapped indexes are placed on each column. Positional designs can be scaled up to perhaps 100 or so columns before the databases and user interfaces become awkward or hard to maintain. Columnar databases are well suited to these kinds of designs because new columns can be easily added with minimal disruption to the internal storage of the data, and the low-cardinality columns containing only a few discrete values are dramatically compressed.

When the number of different attributes grows beyond your comfort zone, and if new attributes are added frequently, a bridge table is recommended. Ultimately, when you have a very large and expanding set of demographics indicators, using outriggers or mini-dimensions simply does not gracefully scale. For example, you may collect loan application information as a set of open ended name-value pairs, as shown in Figure 8-7. Name-value pair data is interesting because the values can be numeric, textual, a file pointer, a URL, or even a recursive reference to enclosed name-value pair data.

Over a period of time, you could collect hundreds or even thousands of different loan application variables. For a true name-value pair data source, the value field itself can be stored as a text string to handle the open-ended modality of the values, which is interpreted by the analysis application. In these situations whenever the number of variables is open-ended and unpredictable, a bridge table design is appropriate, as shown in Figure 8-8.

Loan Application Name-Value Pair Data
Photograph: <image>
Primary Income: $72345
Other Taxable Income: $2345
Tax-Free Income: $3456
Long Term Gains: $2367
Garnished Wages: $789
Pending Judgment Potential: $555
Alimony: $666
Jointly Owned Real Estate Appraised Value: $123456
Jointly Owned Real Estate Image: <image>
Jointly Owned Real Estate MLS Listing: <URL>
Percentage Ownership Real Estate: 50
Number Dependents: 4
Pre-existing Medical Disability: Back Injury
Number of Weeks Lost to Disability: 6
Employer Disability Support Statement: <document archive>
Previous Bankruptcy Declaration Type: 11
Years Since Bankruptcy: 8
Spouse Financial Disclosure: <name-value pair>
... 100 more name-value pairs...

Figure 8-7: Loan application name-value pair data.

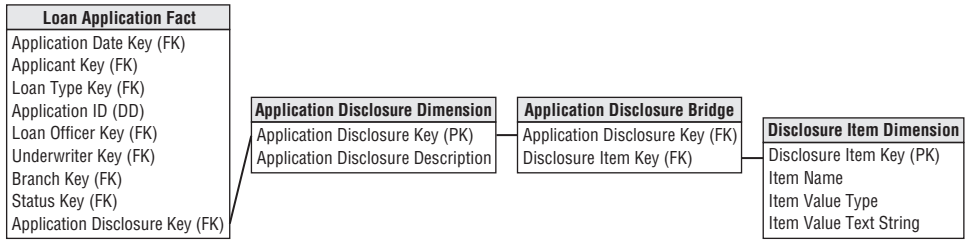


Figure 8-8: Bridge table for wide and sparse name-value pair data set.

Bridge Table for Multiple Customer Contacts

Large commercial customers have many points of contact, including decision makers, purchasing agents, department heads, and user liaisons; each point of contact is associated with a specific role. Because the number of contacts is unpredictable but possibly large, a bridge table design is a convenient way to handle this situation, as shown in Figure 8-9. Some care should be taken not to overdo the contact dimension and make it a dumping ground for every employee or citizen or salesperson or human being the organization interacts with. Restrict the dimension for this use case of contacts as part of the customer relationship.

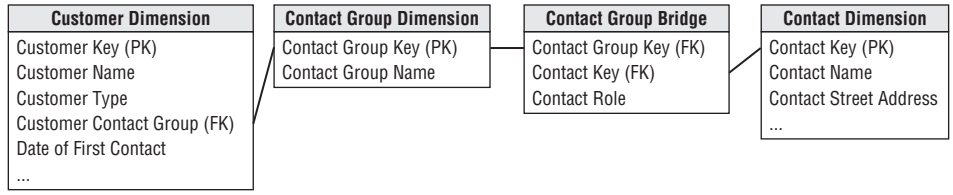


Figure 8-9: Bridge table design for multiple contacts.

Complex Customer Behavior

Customer behavior can be very complex. In this section, we'll discuss the handling of customer cohort groups and capturing sequential behavior. We'll also cover precise timespan fact tables and tagging fact events with indicators of customer satisfaction or abnormal scenarios.

Behavior Study Groups for Cohorts

With customer analysis, simple queries such as how much was sold to customers in this geographic area in the past year rapidly evolve to much more complex inquiries, such as how many customers bought more this past month than their average monthly purchase amount from last year. The latter question is too complex for business users to express in a single SQL request. Some BI tool vendors allow embedded subqueries, whereas others have implemented drill-across capabilities in which complex requests are broken into multiple select statements and then combined in a subsequent pass.

In other situations, you may want to capture the set of customers from a query or exception report, such as the top 100 customers from last year, customers who spent more than $1,000 last month, or customers who received a specific test solicitation, and then use that group of customers, called a *behavior study group*, for subsequent analyses without reprocessing to identify the initial condition. To create a behavior study group, run a query (or series of queries) to identify the set of customers you want to further analyze, and then capture the customer durable keys of the identified set as an actual physical table consisting of a single customer key column. By leveraging the customers' durable keys, the study group dimension is impervious to type 2 changes to the customer dimension which may occur after the study group members are identified.

NOTE The secret to building complex behavioral study group queries is to capture the keys of the customers or products whose behavior you are tracking. You then use the captured keys to subsequently constrain other fact tables without having to rerun the original behavior analysis.

You can now use this special behavior study group dimension table of customer keys whenever you want to constrain any analysis on any table to that set of specially defined customers. The only requirement is that the fact table contains a customer key reference. The use of the behavior study group dimension is shown in Figure 8-10.

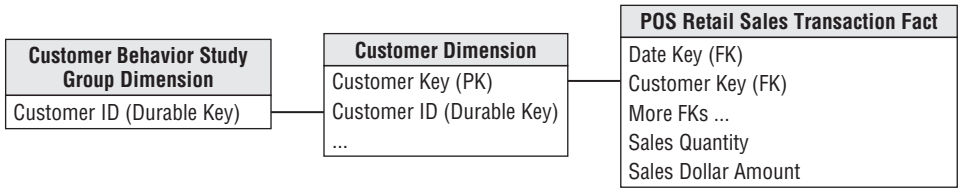


Figure 8-10: Behavior study group dimension joined to customer dimension's durable key.

The behavior study group dimension is attached with an equijoin to the customer dimension's durable key (refer to Customer ID in Figure 8-10). This can even be done in a view that hides the explicit join to the behavior dimension. In this way, the resulting dimensional model looks and behaves like an uncomplicated star. If the special dimension table is hidden under a view, it should be labeled to uniquely identify it as being associated with the top 100 customers, for example. Virtually any BI tool can now analyze this specially restricted schema without paying syntax or user-interface penalties for the complex processing that defined the original subset of customers.

NOTE The exceptional simplicity of study group tables allows them to be combined with union, intersection, and set difference operations. For example, a set of problem customers this month can be intersected with the set of problem customers from last month to identify customers who were problems for two consecutive months.

Study groups can be made even more powerful by including an occurrence date as a second column correlated with each durable key. For example, a panel study of consumer purchases can be conducted where consumers enter the study when they exhibit some behavior such as switching brands of peanut butter. Then further purchases can be tracked after the event to see if they switched brands again. To get this right, these purchase events must be tracked with the right time stamps to get the behavior in the right sequence.

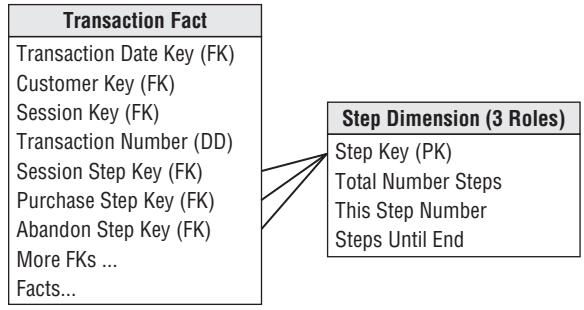
Like many design decisions, this one represents certain compromises. First, this approach requires a user interface for capturing, creating, and administering

real physical behavior study group tables in the data warehouse. After a complex exception report has been defined, you need the ability to capture the resulting keys into an applet to create the special behavior study group dimension. These study group tables must live in the same space as the primary fact table because they are going to be joined directly to the customer dimension table. This obviously affects the DBA's responsibilities.

Step Dimension for Sequential Behavior

Most DW/BI systems do not have good examples of sequential processes. Usually measurements are taken at a particular place watching the stream of customers or products going by. Sequential measurements, by contrast, need to follow a customer or a product through a series of steps, often measured by different data capture systems. Perhaps the most familiar example of a sequential process comes from web events where a session is constructed by collecting individual page events on multiple web servers tied together via a customer's cookie. Understanding where an individual step fits in the overall sequence is a major challenge when analyzing sequential processes.

By introducing a step dimension, you can place an individual step into the context of an overall session, as shown in Figure 8-11.



Sample Step Dimension Rows:

Step Key	Total Number Steps	This Step Number	Steps Until End
1	1	1	0
2	2	1	1
3	2	2	0
4	3	1	2
5	3	2	1
6	3	3	0
7	4	1	3
8	4	2	2
9	4	3	1
10	4	4	0

Figure 8-11: Step dimension to capture sequential activities.

The step dimension is an abstract dimension defined in advance. The first row in the dimension is used only for one-step sessions, where the current step is the first step and there are no more steps remaining. The next two rows in the step dimension are used for two-step sessions. The first row (Step Key = 2) is for step number 1 where there is one more step to go, and the next row (Step Key = 3) is for step number 2

where there are no more steps. The step dimension can be prebuilt to accommodate sessions of at least 100 steps. In Figure 8-11 you see the step dimension can be associated with a transaction fact table whose grain is the individual page event. In this example, the step dimension has three roles. The first role is the overall session. The second role is a successful purchase subsession, where a sequence of page events leads to a confirmed purchase. The third role is the abandoned shopping cart, where the sequence of page events is terminated without a purchase.

Using the step dimension, a specific page can immediately be placed into one or more understandable contexts (overall session, successful purchase, and abandoned shopping cart). But even more interestingly, a query can constrain exclusively only to the first page of successful purchases. This is a classic web event query, where the "attractant" page of successful sessions is identified. Conversely, a query could constrain exclusively to the last page of abandoned shopping carts, where the customer is about to decide to go elsewhere.

Another approach for modeling sequential behavior takes advantage of specific fixed codes for each possible step. If you track customer product purchases in a retail environment, and if each product can be encoded, for instance, as a 5 digit number, then you can create a single wide text column for each customer with the sequence of product codes. You separate the codes with a unique non-numeric character. Such a sequence might look like

11254|45882|53340|74934|21399|93636|36217|87952|…etc.

Now using wild cards you can search for specific products bought sequentially, or bought with other products intervening, or situations in which one product was bought but another was never bought. Modern relational DBMSs can store and process wide text fields of 64,000 characters or more with wild card searches.

Timespan Fact Tables

In more operational applications, you may want to retrieve the exact status of a customer at some arbitrary instant in the past. Was the customer on fraud alert when denied an extension of credit? How long had he been on fraud alert? How many times in the past two years has he been on fraud alert? How many customers were on fraud alert at some point in the past two years? All these questions can be addressed if you carefully manage the transaction fact table containing all customer events. The key modeling step is to include a pair of date/time stamps, as shown in Figure 8-12. The first date/time stamp is the precise moment of the transaction, and the second date/time stamp is the exact moment of the next transaction. If this is done correctly, then the time history of customer transactions maintains an unbroken sequence of date/time stamps with no gaps. Each actual transaction enables you to associate

both demographics and status with the customer. Dense transaction fact tables are interesting because you potentially can change the demographics and especially the status each time a transaction occurs.

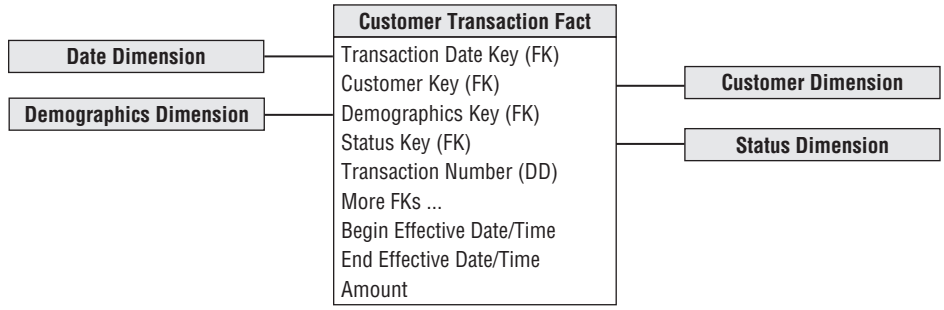


Figure 8-12: Twin date/time stamps in a timespan fact table.

The critical insight is that the pair of date/time stamps on a given transaction defines a span of time in which the demographics and the status are constant. Queries can take advantage of this "quiet" span of time. Thus if you want to know what the status of the customer "Jane Smith" was on July 18, 2013 at 6:33 am, you can issue the following query:

```
Select Customer.Customer_Name, Status
From Transaction_Fact, Customer_dim, Status_dim
Where Transaction_Fact_Customer_Key = Customer_dim.Customer_key
    And Transaction_Fact.Status_key = Status_dim.Status_key
    And Customer_dim.Customer_Name = 'Jane Smith'
    And #July 18, 2013 6:33:00# >= Transaction_Fact.Begin_Eff_
DateTime
    And #July 18, 2013 6:33:00# < Transaction_Fact.End_Eff_DateTime
```

These date/time stamps can be used to perform tricky queries on your customer base. If you want to find all the customers who were on fraud alert sometime in the year 2013, issue the following query:

```
Select Customer.Customer_Name
From Transaction_Fact, Customer_dim, Status_dim
Where <joins>
   And Status_dim Status_Description = 'Fraud Alert'
   And Transaction_Fact.Begin_Eff_DateTime <= 12/31/2013:23:59:59
   And Transaction_Fact.End_Eff_DateTime >= 1/1/2013:0:0:0
```

Amazingly, this one query handles all the possible cases of begin and end effective date/times straddling the beginning or end of 2013, being entirely contained with 2013, or completely straddling 2013.

You can even count the number of days each customer was on fraud alert in 2013:

```
Select Customer.Customer_Name,
    sum( least(12/31/2013:23:59:59, Transaction_Fact.End_Eff_
DateTime)
       - greatest(1/1/2013:0:0:0, Transaction_Fact.Begin_Eff_
DateTime))
From Transaction_Fact, Customer_dim, Status_dim
Where <joins>
   And Status_dim Status_Description = 'Fraud Alert'
   And Transaction_Fact.Begin_Eff_DateTime <= 12/31/2013:23:59:59
   And Transaction_Fact.End_Eff_DateTime >= 1/1/2013:0:0:0
Group By Customer.Customer_Name
```

Back Room Administration of Dual Date/Time Stamps

For a given customer, the date/time stamps on the sequence of transactions must form a perfect unbroken sequence with no gaps. It is tempting to make the end effective date/time stamp be one "tick" less than the beginning effective date/time stamp of the next transaction, so the query SQL can use the BETWEEN syntax rather than the uglier constraints shown above. However, in many situations the little gap defined by that tick could be significant if a transaction could fall within the gap. By making the end effective date/time exactly equal to the begin date time of the next transaction, you eliminate this risk.

Using the pair of date/time stamps requires a two-step process whenever a new transaction row is entered. In the first step, the end effective date/time stamp of the most current transaction must be set to a fictitious date/time far in the future. Although it would be semantically correct to insert NULL for this date/time, nulls become a headache when you encounter them in constraints because they can cause a database error when you ask if the field is equal to a specific value. By using a fictitious date/time far in the future, this problem is avoided.

In the second step, after the new transaction is entered into the database, the ETL process must retrieve the previous transaction and set its end effective date/time to the date/time of the newly entered transaction. Although this two-step process is a noticeable cost of this twin date/time approach, it is a classic and desirable trade-off between extra ETL overhead in the back room and reduced query complexity in the front room.

Tagging Fact Tables with Satisfaction Indicators

Although profitability might be the most important key performance indicator in many organizations, customer satisfaction is a close second. And in organizations without profit metrics, such as government agencies, satisfaction is (or should be) number one.

Satisfaction, like profitability, requires integration across many sources. Virtually every customer facing process is a potential source of satisfaction information, whether the source is sales, returns, customer support, billing, website activity, social media, or even geopositioning data.

Satisfaction data can be either numeric or textual. In the Chapter 6: Order Management, you saw how classic measures of customer satisfaction could be modeled both ways simultaneously. The on-time measures could be both additive numeric facts as well as textual attributes in a service level dimension. Other purely numeric measures of satisfaction include numbers of product returns, numbers of lost customers, numbers of support calls, and product attitude metrics from social media.

Figure 8-13 illustrates a frequent flyer satisfaction dimension that could be added to the flight activity fact tables described in Chapter 12. Textual satisfaction data is generally modeled in two ways, depending on the number of satisfaction attributes and the sparsity of the incoming data. When the list of satisfaction attributes is bounded and reasonably stable, a positional design is very effective, as shown in Figure 8-13.

Satisfaction Dimension
Satisfaction Key (PK)
Delayed Arrival Indicator
Diversion to Other Airport Indicator
Lost Luggage Indicator
Failure to Get Upgrade Indicator
Middle Seat Indicator
Personnel Problem Indicator

Figure 8-13: Positional satisfaction dimension for airline frequent flyers.

Tagging Fact Tables with Abnormal Scenario Indicators

Accumulating snapshot fact tables depend on a series of dates that implement the "standard scenario" for the pipeline process. For order fulfillment, you may have the steps of order created, order shipped, order delivered, order paid, and order returned as standard steps in the order scenario. This kind of design is successful when 90 percent or more of the orders progress through these steps (hopefully without the return) without any unusual exceptions.

But if an occasional situation deviates from the standard scenario, you don't have a good way to reveal what happened. For example, maybe when the order

was shipped, the delivery truck had a flat tire. A decision was made to unload the delivery to another truck, but unfortunately it began to rain and the shipment was water damaged. Then it was refused by the customer, and ultimately there was a lawsuit. None of these unusual steps are modeled in the standard scenario in the accumulating snapshot. Nor should they be!

The way to describe unusual departures from the standard scenario is to add a delivery status dimension to the accumulating snapshot fact table. For the case of the weird delivery scenario, you tag this order fulfillment row with the status Weird. Then if the analyst wants to see the complete story, the analyst can join to a companion transaction fact table through the order number and line number that has every step of the story. The transaction fact table joins to a transaction dimension, which indeed has Flat Tire, Damaged Shipment, and Lawsuit as transactions. Even though this transaction dimension will grow over time with unusual entries, it is well bounded and stable.

Customer Data Integration Approaches

In typical environments with many customer facing processes, you need to choose between two approaches: a single customer dimension derived from all the versions of customer source system records or multiple customer dimensions tied together by conformed attributes.

Master Data Management Creating a Single Customer Dimension

In some cases, you can build a single customer dimension that is the "best of breed" choice among a number of available customer data sources. It is likely that such a conformed customer dimension is a distillation of data from several operational systems within your organization. But it would be typical for a unique customer to have multiple identifiers in multiple touch point systems. To make matters worse, data entry systems often don't incorporate adequate validation rules. Obviously, an operational CRM objective is to create a unique customer identifier and restrict the creation of unnecessary identifiers. In the meantime, the DW/BI team will likely be responsible for sorting out and integrating the disparate sources of customer information.

Some organizations are lucky enough to have a centralized *master data management* (*MDM*) system that takes responsibility for creating and controlling the single enterprise-wide customer entity. But such centralization is rare in the real world. More frequently, the data warehouse extracts multiple incompatible customer data

files and builds a "downstream" MDM system. These two styles of MDM are illustrated in Figure 8-14.

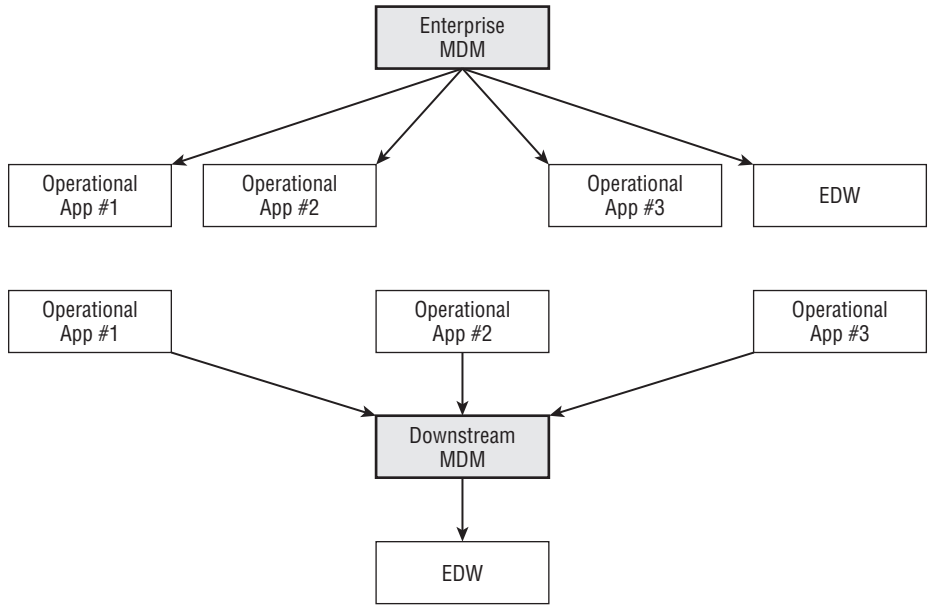


Figure 8-14: Two styles of master data management.

Unfortunately, there's no secret weapon for tackling this data consolidation. The attributes in the customer dimension should represent the "best" source available in the enterprise. A *national change of address* (*NCOA*) process should be integrated to ensure address changes are captured. Much of the heavy lifting associated with customer data consolidation demands customer matching or deduplicating logic. Removing duplicates or invalid addresses from large customer lists is critical to eliminate the costs associated with redundant, misdirected, or undeliverable communication, avoid misleading customer counts, and improve customer satisfaction through higher quality communication.

The science of customer matching is more sophisticated than it might first appear. It involves fuzzy logic, address parsing algorithms, and enormous look-up directories to validate address elements and postal codes, which vary significantly by country. There are specialized, commercially available software and service offerings that perform individual customer or commercial entity matching with remarkable accuracy. Often these products match the address components to standardized census codes, such as state codes, country codes, census tracts, block groups, metropolitan statistical areas (MSAs), and latitude/longitude, which facilitate the merging

of external data. As discussed in Chapter 10, there are also householding capabilities to group or link customers sharing similar name and/or address information. Rather than merely performing intrafile matching, some services maintain an enormous external reference file of everyone in the United States to match against. Although these products and services are expensive and/or complex, it's worthwhile to make the investment if customer matching is strategic to the organization. In the end, effective consolidation of customer data depends on a balance of capturing the data as accurately as possible in the source systems, coupled with powerful data cleansing/merging tools in the ETL process.

Partial Conformity of Multiple Customer Dimensions

Enterprises today build customer knowledge stores that collect all the internal and external customer-facing data sources they can find. A large organization could have as many as 20 internal data sources and 50 or more external data sources, all of which relate in some way to the customer. These sources can vary wildly in granularity and consistency. Of course, there is no guaranteed high-quality customer key defined across all these data sources and no consistent attributes. You don't have any control over these sources. It seems like a hopeless mess.

In Chapter 4: Inventory, we laid the groundwork for conformed dimensions, which are the required glue for achieving integration across separate data sources. In the ideal case, you examine all the data sources and define a single comprehensive dimension which you attach to all the data sources, either simultaneously within a single tablespace or by replicating across multiple tablespaces. Such a single comprehensive conformed dimension becomes a wonderful driver for creating integrated queries, analyses, and reports by making consistent row labels available for drill-across queries.

But in the extreme integration world with dozens of customer-related dimensions of different granularity and different quality, such a single comprehensive customer dimension is impossible to build. Fortunately, you can implement a lighter weight kind of conformed customer dimension. Remember the essential requirement for two dimensions to be conformed is they share one or more specially administered attributes that have the same column names and data values. Instead of requiring dozens of customer-related dimensions to be identical, you only require they share the specially administered conformed attributes.

Not only have you taken the pressure off the data warehouse by relaxing the requirement that all the customer dimensions in your environment be equal from top to bottom, but in addition you can proceed in an incremental and agile way to plant the specially administered conformed attributes in each of the customer-related dimensions. For example, suppose you start by defining a fairly high-level

categorization of customers called customer category. You can proceed methodically across all the customer-related dimensions, planting this attribute in each dimension without changing the grain of any target dimension and without invalidating any existing applications that depend on those dimensions. Over a period of time, you gradually increase the scope of integration as you add the special attributes to the separate customer dimensions attached to different sources. At any point in time, you can stop and perform drill-across reports using the dimensions where you have inserted the customer category attribute.

When the customer category attribute has been inserted into as many of the customer-related dimensions as possible, you can then define more conformed attributes. Geographic attributes such as city, county, state, and country should be even easier than the customer category. Over a period of time, the scope and power of the conformed customer dimensions let you do increasingly sophisticated analyses. This incremental development with its closely spaced deliverables fits an agile approach.

Avoiding Fact-to-Fact Table Joins

DW/BI systems should be built process-by-process, not department-by-department, on a foundation of conformed dimensions to support integration. You can imagine querying the sales or support fact tables to better understand a customers' purchase or service history.

Because the sales and support tables both contain a customer foreign key, you can further imagine joining both fact tables to a common customer dimension to simultaneously summarize sales facts along with support facts for a given customer. Unfortunately, the many-to-one-to-many join will return the wrong answer in a relational environment due to the differences in fact table cardinality, even when the relational database is working perfectly. There is no combination of inner, outer, left, or right joins that produces the desired answer when the two fact tables have incompatible cardinalities.

Consider the case in which you have a fact table of customer solicitations, and another fact table with the customer responses to solicitations, as shown in Figure 8-15. There is a one-to-many relationship between customer and solicitation, and another one-to-many relationship between customer and response. The solicitation and response fact tables have different cardinalities; in other words, not every solicitation results in a response (unfortunately for the marketing department) and some responses are received for which there is no solicitation. Simultaneously joining the solicitations fact table to the customer dimension, which is, in turn, joined to the responses fact table, does not return the correct answer in a relational DBMS due to the cardinality differences. Fortunately, this problem is easily avoided. You simply issue the drill-across technique explained in Chapter 4 to query the

solicitations table and responses table in separate queries and then outer join the two answer sets. The drill-across approach has additional benefits for better controlling performance parameters, in addition to supporting queries that combine data from fact tables in different physical locations.

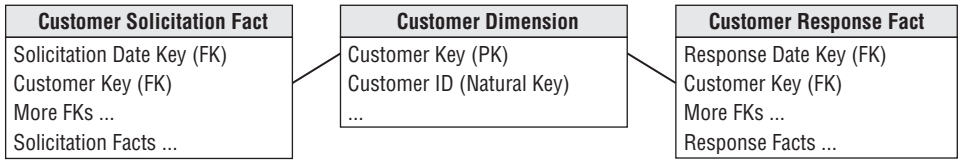


Figure 8-15: Many-to-one-to-many joined tables should *not* be queried with a single SELECT statement.

WARNING Be very careful when simultaneously joining a single dimension table to two fact tables of different cardinality. In many cases, relational engines return the "wrong" answer.

If business users are frequently combining data from multiple business processes, a final approach is to define an additional fact table that combines the data once into a consolidated fact table rather than relying on users to consistently and accurately combine the data on their own, as described in Chapter 7. Merely using SQL to drill across fact tables to combine the results makes more sense when the underlying processes are less closely correlated. Of course, when constructing the consolidated fact table, you still need to establish business rules to deal with the differing cardinality. For example, does the consolidated fact table include all the solicitations and responses or only those where both a solicitation and response occurred?

Low Latency Reality Check

The behavior of a customer in the last few hours or minutes can be extremely interesting. You may even want to make decisions while dealing with the customer in real time. But you need to be thoughtful in recognizing the costs and limitations of low latency data. Generally, data quality suffers as the data is delivered closer to real time.

Business users may automatically think that the faster the information arrives in the DW/BI system, the better. But decreasing the latency increases the data quality

problems. Figure 20-6 summarizes the issues that arise as data is delivered faster. In the conventional batch world, perhaps downloading a batch file once each 24 hours, you typically get complete transaction sets. For example, if a commercial customer places an order, they may have to pass a credit check and verify the final commitment. The batch download includes orders only where all these steps have taken place. In addition, because the batch download is processed just once each 24 hours, the ETL team has the time to run the full spectrum of data quality checks, as we'll describe in Chapter 19: ETL Subsystems and Techniques.

If the data is extracted many times per day, then the guarantee of complete transaction sets may have to be relinquished. The customer may have placed the order but has not passed the credit check. Thus there is the possibility that results may have to be adjusted after the fact. You also may not run the full spectrum of data quality checks because you don't have time for extensive multitable lookups. Finally, you may have to post data into the data warehouse when all the keys have not been resolved. In this case, temporary dimensional entries may need to be used while waiting for additional data feeds.

Finally, if you deliver data instantaneously, you may be getting only transaction fragments, and you may not have time to perform any data quality checks or other processing of the data.

Low latency data delivery can be very valuable, but the business users need to be informed about these trade-offs. An interesting hybrid approach is to provide low latency intraday delivery but then revert to a batch extract at night, thereby correcting various data problems that could not be addressed during the day. We discuss the impact of low latency requirements on the ETL system in Chapter 20: ETL System Design and Development Process and Tasks.

Summary

In this chapter, we focused exclusively on the customer, beginning with an overview of customer relationship management (CRM) basics. We then delved into design issues surrounding the customer dimension table. We discussed name and address parsing where operational fields are decomposed to their basic elements so that they can be standardized and validated. We explored several other types of common customer dimension attributes, such as dates, segmentation attributes, and aggregated facts. Dimension outriggers that contain a large block of relatively low-cardinality attributes were described.

This chapter introduced the use of bridge tables to handle unpredictable, sparsely populated dimension attributes, as well as multivalued dimension attributes.

We also explored several complex customer behavior scenarios, including sequential activities, timespan fact tables, and tagging fact events with indicators to identify abnormal situations.

We closed the chapter by discussing alternative approaches for consistently identifying customers and consolidating a rich set of characteristics from the source data, either via operational master data management or downstream processing in the ETL back room with potentially partial conformity. Finally, we touched on the challenges of low latency data requirements.