CHAPTER 24

Data Warehousing

Abstract

This chapter presents an overview database warehouses, large databases used for strategic decision making, rather than day-to-day transaction processing. The chapter discusses data modeling for data warehouses, as well as the extract-translate-load process.

Keywords

data warehouse
data warehousing
extract-translate-load
ETL
dimensional modeling

A *data warehouse* is a repository of transaction and nontransaction data used for querying, reporting, and corporate decision-making. The data typically come from multiple sources. They are not used for day-to-day corporate operations and therefore once data have been stored, tend to change much less than the transactional/operational databases we have used as examples to this point.

Because of the cost associated with creating and maintaining a data warehouse, only very large organizations establish their own. Tables can grow to hold millions of rows; storage capacities have now moved into the petabyte range. Full-fledged data warehouses therefore require mainframe processing capabilities or large clusters of smaller servers.

Note: Who has data warehouses that large? eBay for one. The company has at least two data warehouses. The smaller is a mere 9.2 petabytes. The larger, which stores Web clicks along with other data, was more than 40 petabytes in 2013. Wal-Mart, Verizon, AT&T, and Bank of America also have data warehouses with storage measured in petabytes.

The software that manages a data warehouse typically is a relational DBMS.[1] However, data modeling is somewhat different, because the goals of the data warehouse are different from an operational transaction-based database. The purpose of this chapter is to acquaint you with how data warehouses fit into the information strategy of large organizations, as well as how and why their designs differ from the relational data model as it has been presented throughout this book.

Scope and Purpose of a Data Warehouse

To better understand the difference between a data warehouse (or its smaller sibling, a *data mart*), let's return to SmartMart, the retailer whose operational database was presented in Chapter 15. SmartMart's operational system performs the following data management activities for the organization:

- Tracks the location and stock level of inventory items
- Stores in-store and Web sales data, including promotions applied at the time of purchase
- Handles employee scheduling and work assignments (feeds into the payroll system)

The applications that run against the database answer queries such as

- Where is a specific product in stock and how many are available?
- What promotions apply to a product being purchased?
- What items were ordered on a specific Web order?
- Where and when is a specific employee working?

The queries provide information necessary for the day-to-day operations of the business. However, they aren't intended to provide the types of information that upper-level management needs to make strategic decisions, such as which products sell well in which parts of country, and to evaluate the results of previous decisions, such as which promotions generated a significant rise in sales and should be repeated. Although an operational database can provide summary reports for a district manager showing how the stores in his or her territory performed over various time periods, such reports are typically limited to only a few years and the type of information presented by the reports is fixed when the report application is developed.

Strategic planning and reviews of the implementation of strategic plans need to be able to "slice and dice" the data in a variety of ways. In other words, the queries need to allow data to be grouped by a variety of descriptions—sales by state, sales by promotion type, sales by zip code, sales by date, and so on—in an ad hoc manner. The data in an operational database may be offloaded to archival storage after a year or so to keep the database from becoming too large, but the data in a data warehouse are usually kept indefinitely.

Operational systems are usually accompanied by a variety of prewritten applications, such as those that run on point of sale terminals, and management summary reports, such as that described earlier. Very few users have ad hoc query access to the database. In contrast, there are few (if any) prewritten applications for the data in a data warehouse. Users work with a query tool that makes it possible to explore data groupings however they choose and however the data might lead them.

The primary activity performed using a data warehouse is *data mining*, through which a user analyzes data to look for patterns in the data. For example, data mining can identify which products sell best in which parts of a company's market. The results can be used to tailor marketing example, data mining can identify which products sell best in which parts of a company's market. The results can be used to tailor marketing campaigns to the location or to shift inventory levels to better match demand.

A data mining activity conducted by the 7-Eleven Corporation, for example, indicated that around 8 pm in the evening, sales of beer and diapers went up. When you think about it, the result makes some sense: Fathers are sent to the convenience store to get diapers in the early evening and, while there, pick up a six pack of beer. The corporation moved merchandise in the stores so that diapers were placed next to the beer coolers. As a result, beer sales went up significantly.

You have to be careful when data mining, because statistically you are bound to find "something" sooner or later. For example, a data mining activity discovered that individuals with higher incomes tended to own more expensive houses than those with lower incomes. This type of fact is certainly true, but of little practical use to anyone.

Because we continually add data to a data warehouse and rarely delete data, data warehouses tend to be extremely large databases, with tables containing millions and tens of millions of rows. The sheer volume of the data and the processing power needed to perform ad hoc queries against them require a mainframe or a cluster of smaller servers with the power of a mainframe, rather than a single desktop server. Although many desktop servers do rival mainframes in raw processing power, they can't handle the high volume of I/O that data warehouses require.

Data warehouses also support creating reports for a large business. For example, a company that manages many fast-food restaurants loads all transaction data into a data warehouse. The reports include charts of profit and loss for some or all restaurants, as well as summary reports (such as summaries of sales and the average customer check) for the company as a whole.[2]

Note: Data marts, the smaller versions of full-fledged data warehouses, can and often do run on desktop servers.

Most of the large data warehouses in use today use relational DBMSs, such as DB/2 and Oracle, two of the few products able to handle the data volume.

Note: A new type of DBMSs has emerged to handle the large data sets used by data warehouses. These nonrelational databases are particularly good at managing distributed installations. For details, see Chapter 28.

Obtaining and Preparing the Data

Early in the evolution of data warehousing, general wisdom suggested that the data warehouse should store summarized data rather than the detailed data generated by operational systems. Experience has shown, however, that the data warehouse needs as much detail as the operational system. Storing the detail makes the data warehouse more flexible, allowing users to query in any way they choose. You can always produce summaries if you have the detail, but if you only have summarized data, such as totals and averages, you can't recreate the detail should you need it.

Most, but not all, of the data come from operational systems. Data may also come from outside sources. For example, a company such as SmartMart might want to include demographic data about geographic regions and would be more likely to purchase such data from a government entity rather than attempt to collect those data itself.

Although a data warehouse may store much of the same data as an operational database, there are some significant differences in the way the data are handled:

- Operational databases are generally updated in real time. For example, a sales transaction is entered into the database as the sale occurs. In contrast, data warehouses are typically loaded in batches at regular intervals, such as once a day.
- Operational systems are interested in the latest or current values of many data elements, such as a customer's current address and telephone. Data warehouses, however, want to see how data have changed over time and therefore need to keep historical data. This means that there may be multiple values for a customer's address; each value will then be associated with the dates the address was valid. (See the section *Dates and Data* later in this chapter for more information.)
- Missing values are acceptable in an operational database. For example, if the attribute color doesn't apply to an inventory item, then the value for that column in the product's row can simply be left null. However, nulls in a data warehouse can produce unpredictable or inaccurate results. Assume that we want to know the percentage of products sold over the Web that aren't shipped, such as software that is downloaded. Such items have no shipping weight and, in the operational database, produce no problems when the shipping weight column remains null. But when the data warehouse software is counting the number of items that aren't shipped, the nulls aren't specific enough. A null might represent an item that isn't shipped, but might also represent a shipped item for which we don't know the shipping weight. Therefore, nulls in a data warehouse need to be replaced with specific values, such as "doesn't ship" in our example of the nonshipping inventory items.

Data warehouses typically obtain their data from multiple sources, be they operational systems or data obtained from outside the company. This generates a significant problem when the data sources don't represent duplicated data in the same way. For example, two operational systems (say, one from sales and one from marketing) may use different transaction identifiers, although many transactions appear in both databases. The software that loads the data warehouse must recognize that the transactions are the same and merge the data into a single entity.

Before they are loaded into a data warehouse, data must be modified so that they match whatever format is used in the data warehouse. In addition, duplicated data must be identified and coalesced; nulls must be replaced with specific values. These activities, along with procedures for cleaning the data (removing errors), are performed before the data are loaded.

The process of getting data into the data warehouse is known as *extract-transform-load* (ETL). It is virtually impossible to purchase complete software that will perform ETL processing because the sources of data for each data warehouse are so very different. In most cases, such software must be custom-developed for each warehouse. Much of the expense in setting up a data warehouse therefore comes from the writing and testing of the ETL software. Running data through the ETL software and maintaining the ETL software also consumes a large portion of IT staff effort in maintaining the data warehouse. This work takes place out of the users' sight, in the "back room" where the data are prepared.

Note: When all or most of the data that go into a data warehouse come from within an organization, the changes necessary to make data formatting consistent can either be made in the feeder systems or during the ETL process. If the organization has many operational systems—and, in particular, is working with legacy software—then it may not be feasible to modify the operational systems. However, many organizations can benefit from a project that makes data formatting consistent across multiple databases. The effort to make the changes may be worthwhile in and of itself, even without the benefits to the ongoing ETL process.

Data Modeling for the Data Warehouse

Because the purpose of a data warehouse differs from that of an operational system, there are differences in the types of tables that make up the design. The most commonly used data model used in data warehouses is *dimensional modeling*. As you will see, it takes its basic precepts from the relational data model, such as tables and a variety of keys. However, the tables are generally not normalized. In fact, they are really only in the relational data model, such as tables and a variety of keys. However, the tables are generally not normalized. In fact, they are really only in first normal form, because many tables contain data about multiple entities. They are nonetheless legal relations, because they are two-dimensional tables without repeating groups.

Dimensional Modeling Basics

Dimensional modeling uses two major types of tables: *fact tables* and *dimension tables*. Fact tables hold numeric data that can be summarized as needed; dimension tables hold the descriptive criteria by which a user can organize the data. As a first example, consider Figure 24.1. These tables support querying the data warehouse about inventory levels of products on any given date. They illustrate several of the characteristics of dimensional modeling that are different from pure relational design.

- Natural keys, such as UPCs or ISBNs or invoice numbers, are not used as all or part of the primary keys. Instead, each row in a table is given a unique integer key. These keys speed up joins between the fact and dimension tables. For example, the primary key of the *dimension_date* table is an arbitrary integer rather than the date itself. When the natural keys are included in a table, they are known as *deprecated dimensions*. (Although they are natural keys, they are not referenced by any foreign keys in the data warehouse.)
- Fact tables contain foreign keys and data that can be summarized. For example, the *fact_inventory_level* table contains foreign keys to the date, location, and product dimension tables. The summarizable data item is the quantity in stock. The primary key of the table is the concatenation of two or more of the foreign keys (all of which are arbitrary integer keys). Data warehouses use referential integrity to ensure that the arbitrary foreign keys, used in the fact tables, reference existing dimension table rows. However, unlike true relational databases, the dimensional model foreign keys are always meaningless. Data warehouses also enforce nonnull, unique primary keys.
- Dimension tables contain descriptive data. The *dimension_date* table, for example, has its unique integer key, along with attributes for the many ways in which a person might want to use a date in a query. There will be one row in a date dimension table for each date that might be used by any query against the data warehouse.

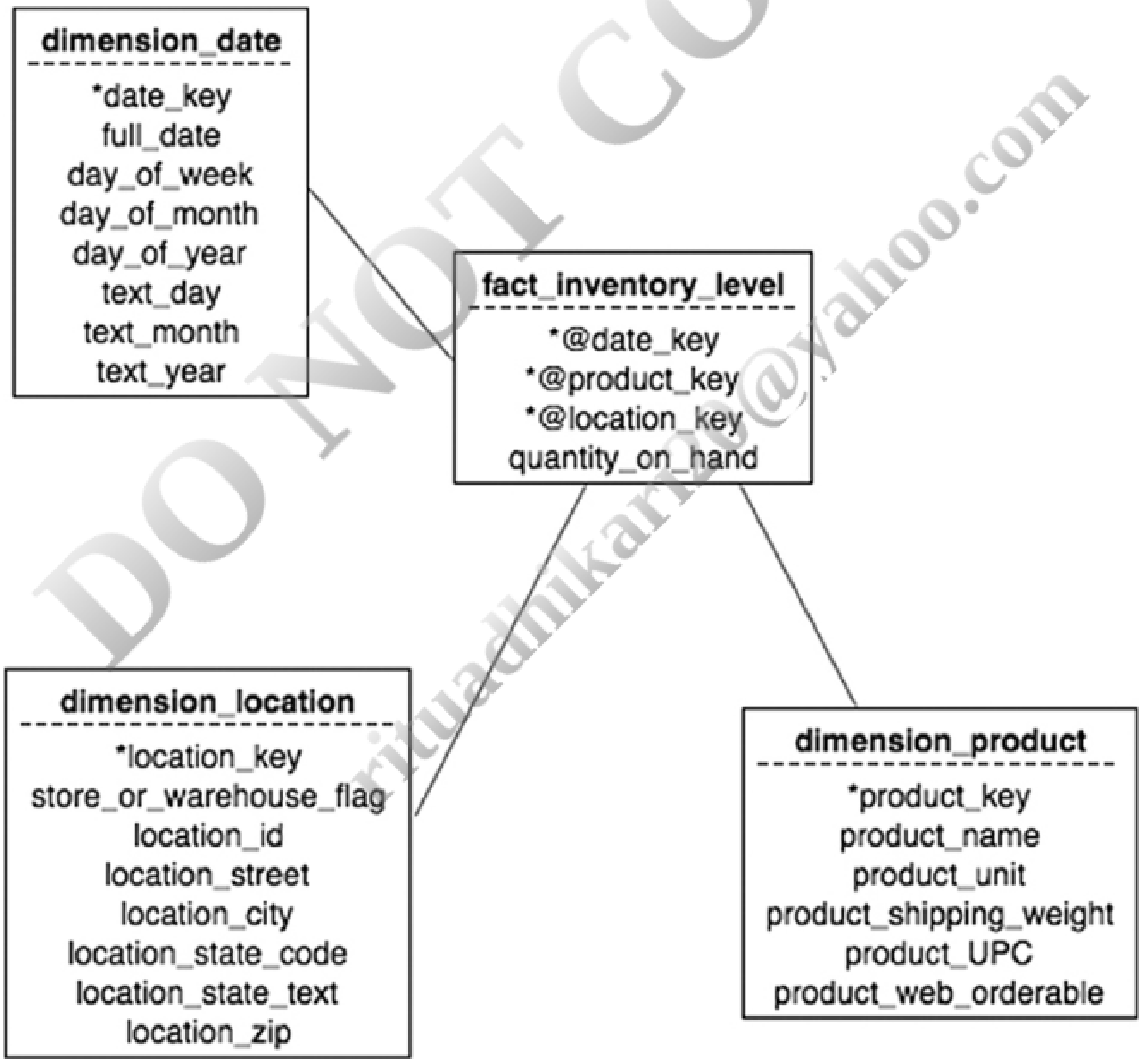


FIGURE 24.1 **Dimension and fact tables to capture inventory levels at a given point in time.**

What can a user do with the dimensional model of the inventory levels? There are a variety of analyses that the model can satisfy, including the following:

- How do the inventory levels of product X vary from month to month?

- How do the inventory levels of product X vary from month to month?
- How do the inventory levels of product X vary from month to month and store to store?
- Which stores show the greatest/least inventory of product X after the winter holiday shopping season?
- Which products had more inventory in the warehouses for product X than in stores during the winter holiday season?
- Which warehouses had the total lowest inventory at the end of each month?

Dates and Data

Unlike operational databases, data warehouses keep track of data changes over time. For example, the SmartMart operational database is concerned about inventory levels at the current time; inventory levels a week ago aren't particularly relevant. However, the design in Figure 24.1 takes a snapshot of inventory levels at some specified date. The data warehouse will therefore contain many snapshots, making the analysis of inventory levels over time possible.

In some circumstances, this can present some problems to the data warehouse designer, especially where human data, such as addresses, are concerned. Consider, for example, the simple customer dimension in Figure 24.2. The problem with this design is that, over time, customers change their addresses and phone numbers. Analysis based on customer location during a specific time period may be incorrect if there are multiple addresses for the same customer.

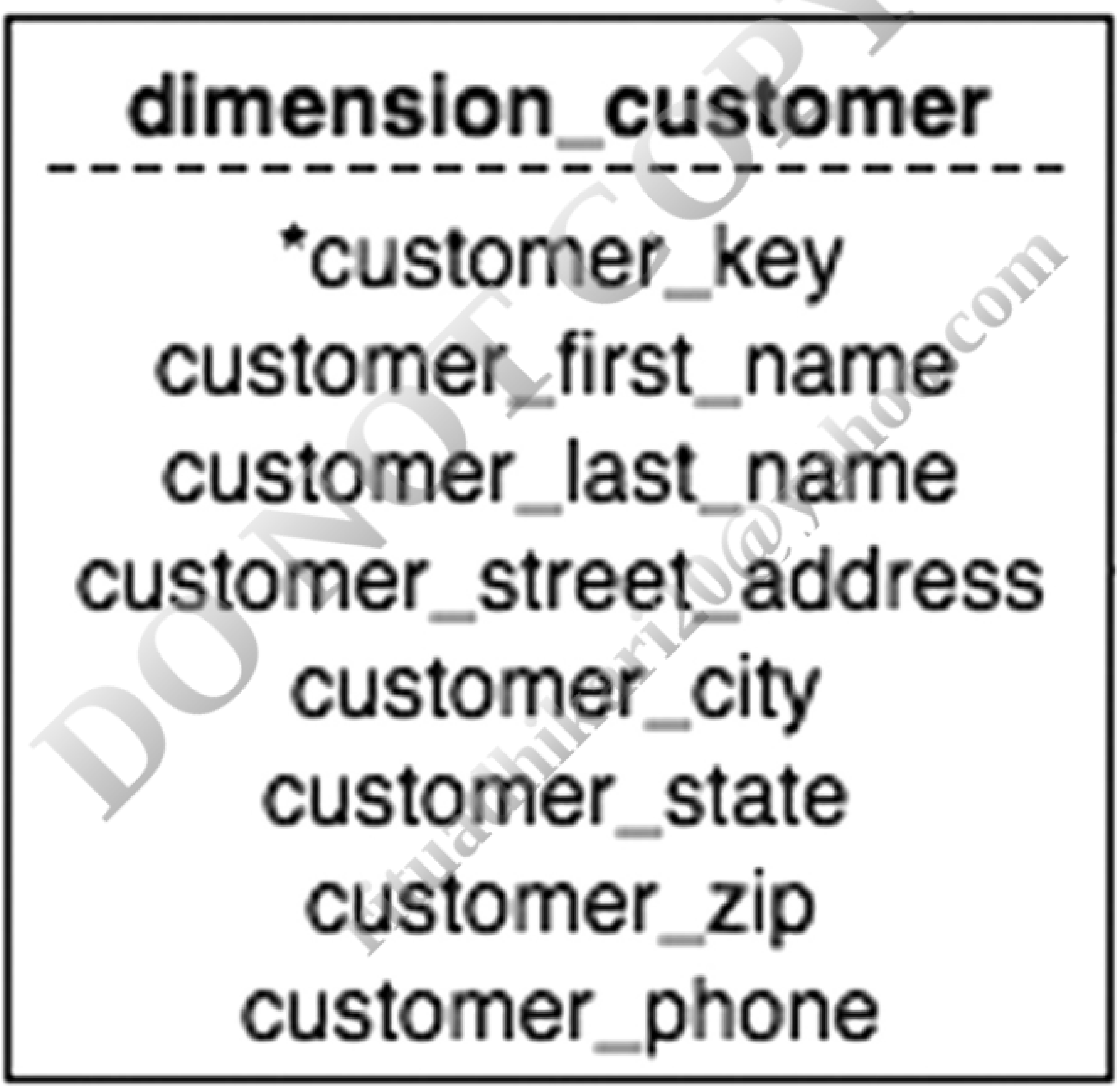
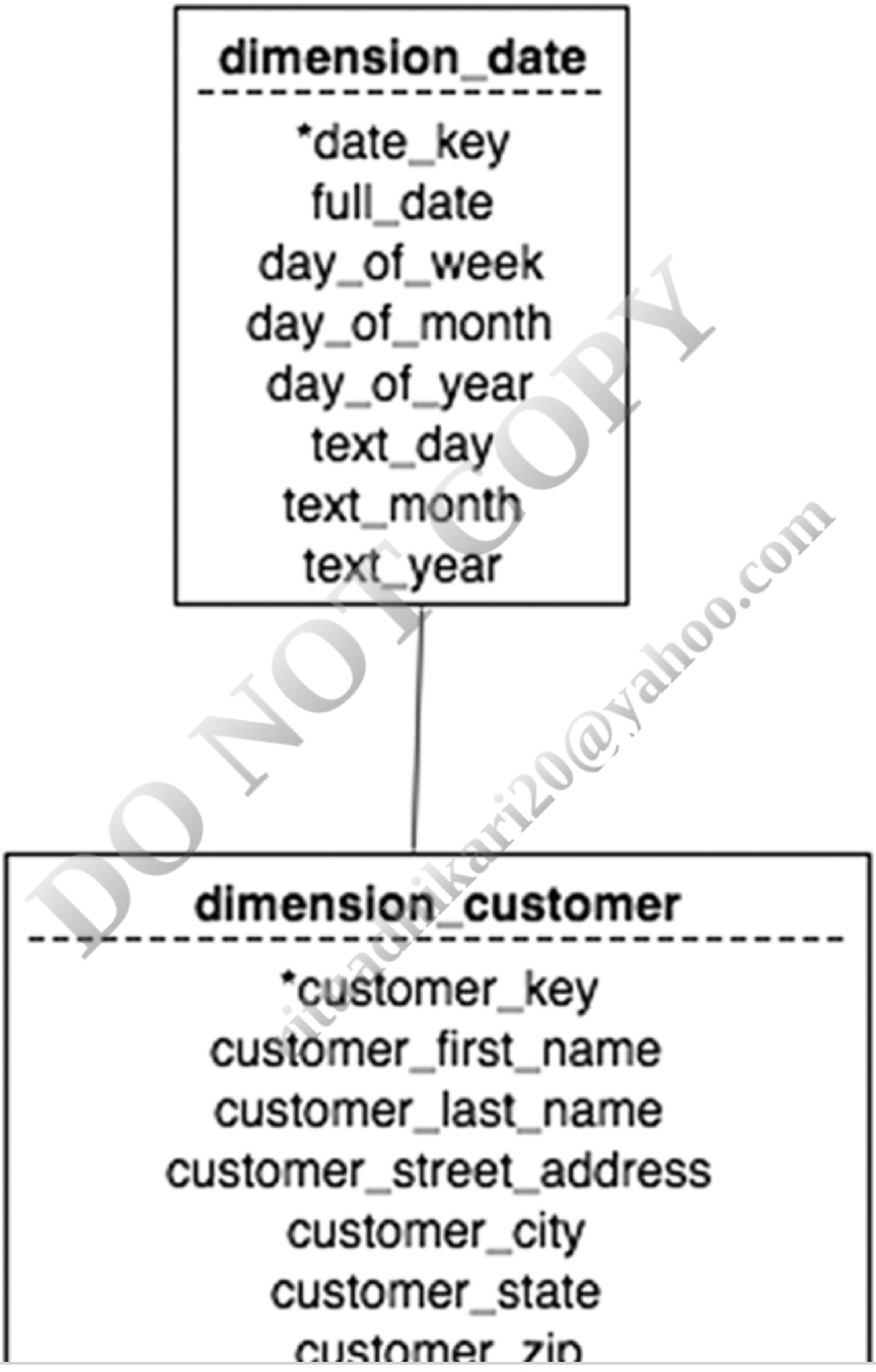


FIGURE 24.2 **A simple customer dimension.**

One solution is to include only the most recent address for a customer in the data warehouse. However, this makes it impossible to analyze sales by location, given that the address in the related customer row may not have been the address when the sale was made. Another solution is to add the dates during which an address was valid to the customer dimension table, as was done in Figure 24.3. There will then be one row in the table for each address used by a customer when making purchases. Because each sale is related to the customer based on an arbitrary customer key rather than a concatenation of customer data, there will be no problem determining the correct address for a specific sale (see Figure 24.4). Queries based on location and date will then need to include logic to ensure that an address was valid during the time period of the analysis.


dimension_date
*date_key
full_date
day_of_week
day_of_month
day_of_year
text_day
text_month
text_year
dimension_customer
*customer_key
customer_first_name
customer_last_name
customer_street_address
customer_city
customer_state
customer_zip

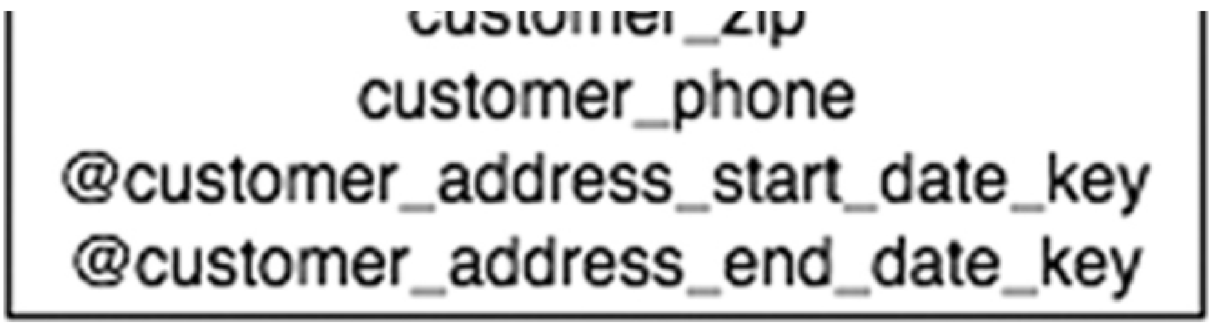

customer_zip
customer_phone
@customer_address_start_date_key
@customer_address_end_date_key

FIGURE 24.3 Including valid dates for an address.

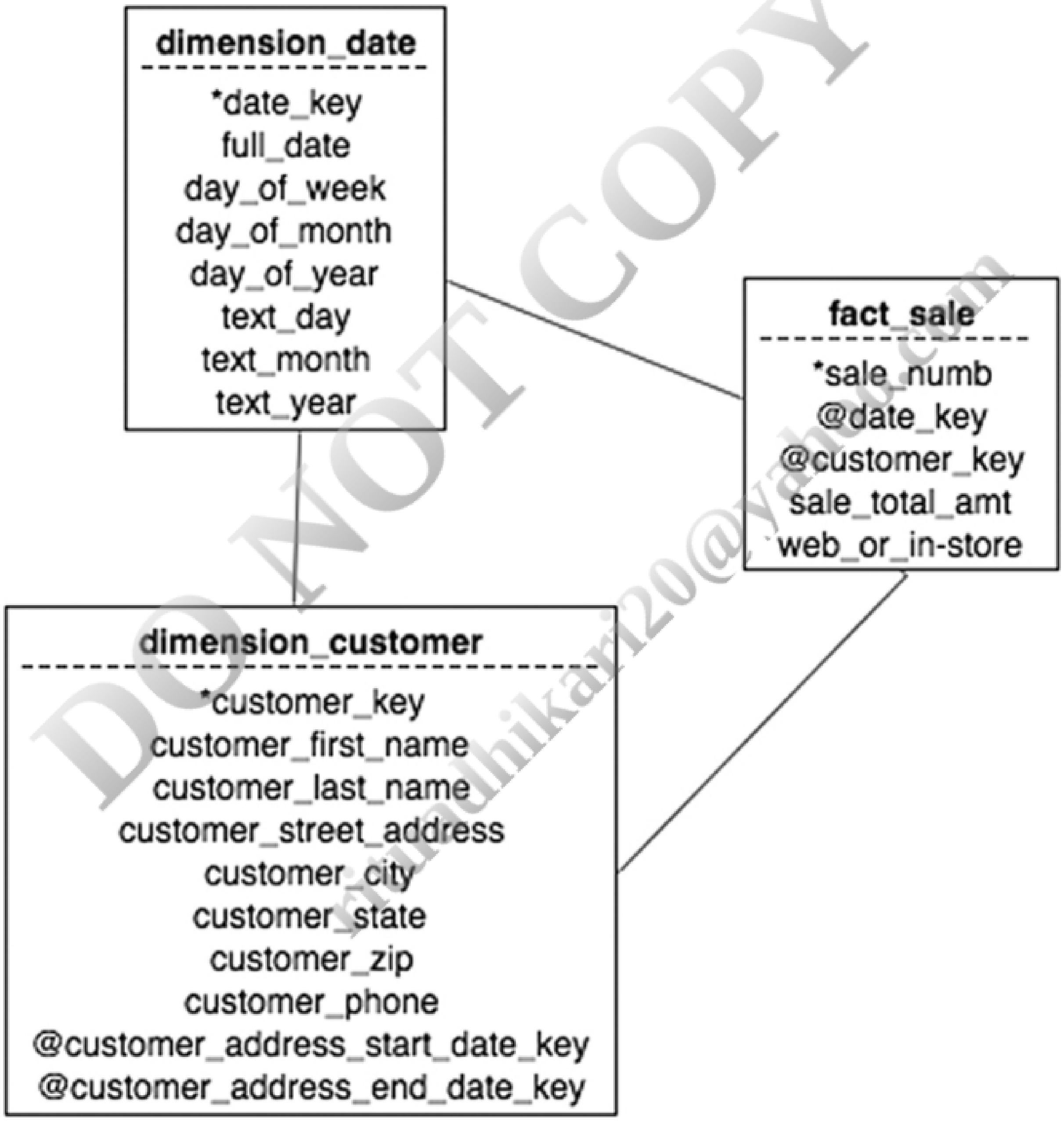


FIGURE 24.4 Relating a customer with a time-sensitive address to a sale.

Data Warehouse Appliances

During the early 1980s, when relational databases were supplanting those based on older data models, several hardware vendors were attempting to sell special-purpose computers called *database machines*. The idea was to take a minicomputer and use it to run just a DBMS (with an OS that to sell special-purpose computers called *database machines*. The idea was to take a minicomputer and use it to run just a DBMS (with an OS that was specifically tailored to that purpose). It would be connected to another computer in a master–slave relationship. Requests for database processing came first to the master machine, which passed them to the database machine. The database machine completed all database activity and sent the results back to the master computer, which, in turn, sent the data to the user. In theory, by offloading database processing to a dedicated machine overall system performance (including database performance) would improve. In practice, however, only database applications that were severely CPU-bound showed significant performance improvements when running on a database machine. The overhead needed to move queries and data to and from the database machine erased any performance gains that occurred from relieving the master computer's CPU of database work. By 1990, almost no one had heard of a database machine.

The rise of data warehouses has seen the reappearance of computers dedicated to database work. Today, the situation is very different because a dedicated database computer is really a special-purpose server that is connected directly to a network.

One such hardware configuration, for example, is being offered by Teradata.[3] The system is focused on high performance access, using a “shared-nothing” architecture. In other words, each processor has access to its own storage and pathways to reach that storage.

The benefit of such a preconfigured solution is that it simplifies setting up the data warehouse. An appliance such as Teradata’s also means that the entire data warehouse infrastructure is supported by a single vendor, which many organizations prefer.

For Further Reading

Aggarwal C. *Data Mining: The Textbook*. Springer; 2015.
Clampa B. *The Data Warehouse Workshop: Providing Practical Experience to the Aspiring ETL Developer*. CreateSpace Independent Publishing; 2014.
Inmon WH, Linstedt D. *Data Architecture: A Primer for the Data Scientist: Big Data, Data Warehouse and Data Vault*. Morgan Kaufmann; 2014.
Kimball R, Ross M. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. third ed. Wiley; 2013.
Kimball R, Ross M, Thornwaite W, Mundy J, Becker B. *The Data Warehouse Lifecycle Toolkit*. second ed. Wiley; 2008.
Laberge R. *The Data Warehouse Mentor: Practical Data Warehouse and Business Intelligence Insights*. McGraw-Hill Osborne Media; 2011.
Tan P-N, Steinbach M, Kumar V. *Introduction to Data Mining*. Addison-Wesley; 2005.
Vaisman A, Zimanyi E. *Data Warehouse Systems: Design and Implementation (Data-Centric Systems and Applications)*. Springer; 2014.
Westphal C. *Data Mining for Intelligence, Fraud & Criminal Detection: Advanced Analytics & Information Sharing Techniques*. CRC; 2008.
Zaki MJ, Meira Jr W. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press; 2014.

[1] Most data warehouse software is commercial. However, Infobright has released an open-source product. For details, see http://pcworld.about.com/od/businesscenter/Infobright-Releases-Open-sourc.htm.

[2] For details, see http://www.xore.com/casestudies/case_study_fastfood.pdf.

[3] http://www.teradata.com/News-Releases/2015/Newest-Teradata-Data-Warehouse-Appliance-is-a-Powerhouse-for-the-Most-Demanding-Analytics/