An Accounts Receivable Star Schema For A Data Mart

Jian Guan, (Email: jeff.guan@louisville.edu), University of Louisville
Alan S. Levitan, (Email: levitan@louisville.edu), University of Louisville

Abstract

Data warehouses, or smaller data marts, are becoming crucial for analyzing the enormous amount of data generated by a typical business. This article describes a useful design for organizing accounts receivable data to facilitate analysis. While the relational system might be ideal for processing transactions, a star schema is proposed to support the ad hoc queries common in a data warehouse environment.

I. Introduction

Over the past few decades our capabilities to generate and collect data have been increasing very rapidly. This wealth of data poses a unique challenge for organizations that want to extract timely information from the data. The challenge is to organize the data so that useful information and knowledge may be derived from the data in a timely fashion. The traditional approach is to derive information or business intelligence directly from on-line-transaction-processing systems or OLTP systems (Grabski and Marsh 1994; McCarthy 1982; Levitan 1999). It is now commonly recognized that this traditional approach is inadequate (Granof 1999; Kimball 1996; Inmon 1996; O'Leary 1999; Chen and Han 1999; Srivastava and Chen 1999). In the past few years the data warehouse technology has become the preferred method to address information delivery and analytical needs and the most popular data modeling technique used in data warehouses is the star schema (O'Leary 1999; Kimball 1996; Inmon 1996). This paper presents the design of an accounts receivable data warehouse using the star schema and discusses some common issues

and benefits in such a design. The star schema based approach to model accounts receivable data represents a very different approach than used in the past and relatively little such work has been seen.

II. Data Warehouse Systems

The explosive growth in data and databases has generated an urgent need for tools and techniques to transform the wealth of data into information and knowledge. The most common channel for data generation and collection is the operational database systems in an organization. However, most operational data are stored in relational databases in which the structures (tables) tend to be highly normalized. Operational data storage is optimized to support transactions that represent daily activities. For example, a simple sales transaction might be recorded in one or two tables (invoice, invoice-line) while referencing data in several other tables (customer, salesperson, store, etc.). Although such an arrangement is excellent in an operational database, it is not query-friendly. There are three main areas in which operational data differ from the type of data that support decision-making: timespan, granularity, and dimensionality (Kimball 1996; Inmon 1996).

Readers with comments or questions are encouraged to contact the authors via email.

Timespan. Operational data represent current (atomic) transactions whereas Decision Support System (DSS) data tend to cover a much longer time frame, such as 1 to 10 years.

Granularity. Operational data represent specific transactions that occur at a given time. In contrast DSS data may be presented at different levels of aggregation, from highly summarized to near-atomic. This allows decision makers to drill down or roll up the data in data analysis.

Dimensionality. From the data analysis point of view, the data are always related in many ways. Data analysts tend to require that the data be presented in many dimensions. This is very different from the single-transaction view typical of operational data.

In response to the need to provide timely and accurate information for both tactical and strategic information, organizations have turned to a different type of data storage architecture, called data warehouse (Chen and Han 1999; Srivastava and Chen 1999). A data warehouse is typically a collection of data in support of management decision making with the following characteristics (Inmon 1996):

Subject-oriented. Data warehouse data are arranged and optimized to provide answers to questions coming from diverse functional areas within a company. Therefore, the data warehouse contains data organized and summarized by topic, such as sales, marketing, finance, distribution and transportation. For each one of these topics the data warehouse contains specific subjects of interest—products, customers, departments, regions, promotions, and so on. This form of data organization is quite different from the more functional or process-oriented organization of typical transaction systems.

Integrated. The data warehouse is a centralized, consolidated database that integrates data derived from the entire organization. Thus the data warehouse consolidates data from multiple and

diverse sources with diverse formats.

Time-variant. All data warehouse data contain a time element. In contrast to the operational data that focus on current transactions, the warehouse data represent the flow of data through time. The data warehouse can even contain projected data generated through statistical and other models. It is also time-variant in the sense that, once data are periodically uploaded to the data warehouse, all time-dependent aggregations are recomputed. For example, when data for previous weekly sales are uploaded to the data warehouse, the weekly, monthly, yearly, and other time-dependent aggregates for products, customers, stores, and other variables are also updated.

Nonvolatile. Once data enter the data warehouse, they are never removed. Because the data in the data warehouse represent the company's entire history, the operational data, representing the near-term history, are always added to it. Because data are never deleted and new data are always added, the data warehouse is always growing.

While a data warehouse tends to be a centralized location for DSS data for the entire enterprise, a data mart is usually a small data warehouse designed to meet the need of a specific business area, such as marketing, sales, etc. Data warehouses can take years to develop, but data marts can be created and become operational in a few months.

Figure 1 shows a typical data warehouse system. The data from the data sources are first extracted and then cleansed, integrated, and aggregated, and finally fed to the warehouse. Data acquisition applications are developed based on rules defined by the warehouse developer. These rules define the data sources from which the warehouse data will be obtained and the data cleanup and enhancement to be done to this data before it is applied to warehouse databases. The data warehouse data store is typically a relational database management system (RDBMS) such as

Figure 1 - Data Warehouse System

Microsoft Access, Microsoft SQL Server 7.0, and Oracle 8i. The most common database modeling approach used in data warehouse databases is the star schema. The data access component provides the data access tools that enable end users to access and analyze warehouse data. Many tools are available today to help with this process.

This paper will focus on the design of an accounts receivable data mart using the star schema model. The paper also presents examples of multidimensional analysis using the resulting data mart.

III. Star Schema as a Dimensional Modeling Technique

The most common platform for implementing data warehouses has been the tried and tested relational technology. However, the traditional relational modeling techniques and normalization do not yield a database structure that serves the

advanced, multidimensional data analysis requirements in a data warehouse. The main reasons include high computational demands and contention for resources required by day-to-day business operations. A multidimensional modeling technique called the star schema has been developed to address the different database structure requirements in a data warehouse (Kimball 1996; Inmon 1996).

Star schemas yield an easily implemented model for multidimensional data analysis while still preserving the relational structures on which the operational database is built. The basic star schema contains a large central table called the fact table and several smaller tables joined to the fact table. These smaller tables are called dimension tables. The fact table contains numeric values (facts) that represent a specific business activity such as sales. Facts commonly used in business data analysis are units, costs, prices, and revenues. Dimension tables contain data that provide qualifying characteristics for the facts in

the fact table. They describe the facts. Typical dimensions include the time dimension, the product dimension, and the location dimension. Figure 2 depicts a star schema for sales with product, location, and time dimensions. Unlike the entity relation model in a relational database, the star schema is very asymmetric. There is one large dominant fact table in the center of the schema. The dimensional tables all have only a single join attaching them to the central fact table.

It is important to note that the facts in a fact table are typically numeric measurements that are continuously valued and additive. For the sales star schema given in Figure 2 unit sales would be a good fact to capture. The reason is that virtually every query made against the fact table could require thousands of millions of records to be processed. This huge number of records will be compressed to yield an answer set that contains a few rows of data. The most useful and sensible way to compress these records is to add them.

Each dimension table contains attributes that are often used to search, filter, or classify facts. Dimensions provide descriptive characteristics about the facts through their attributes. For the star schema given in Figure 2 it is easy to iden-

tify the following attributes:

- Product dimension: product ID, description, category, manufacturer
- Location dimension: country, region, state, city, store number
- Time dimension: year, quarter, month, week, date

These dimension attributes add a business perspective to the sales facts. The attributes can serve as headers in a user's answer set. Figure 3 expands the sales star schema by adding attributes to each dimension. Each dimension table has a one-to-many relationship with the fact table. In other words each row in a dimension table may be associated with many rows in the fact table. For example a particular product may be associated with many sales facts in the fact table.

Such a design supports multidimensional analysis. For example a typical query might return a data set as follows:

Category	Location	Unit Sales
PC	Central	110
PC	Eastern	200
PC	Western	45
Printer	Central	112
Printer	Eastern	167
Printer	Western	23

Attributes within each dimension often form a well-defined hierarchy. For example the location dimension attributes can be organized in a hierarchy by country, region, state, city, store. An attribute hierarchy allows aggregation of facts and drill-down/roll-up data analysis.

Another important consideration in the design of any star schema database is the granularity of data or the grain of the fact table. In the sales star schema in Figure 3 the decision has been made to represent the total unit sales of each product in each city on a daily basis. For example for a retail chain with 500 stores and 5,000 products, measuring the daily movement for one year alone would result in a fact table approaching one billion rows.

A star schema design such as the one given in Figure 3 can be easily implemented in a rela-

tional DBMS. Facts and dimensions are normally represented by physical tables in a relational database. The fact table is related to each dimension table in a one-to-many relationship. Each row in a dimension table will be represented by a unique dimensional primary key value such as a customer id or product id. The primary key in a dimensional table tends to be an integer or whole number as integers are processed much faster than other types of data. Because the fact table is related to many dimension tables, the primary key of the fact table is a composite primary key. Figure 4 shows a star schema database in Microsoft Access. Figure 5 shows the design view of the Sales fact table and Figure 6 shows the design view of the Time dimension table. The composite key of the Sales fact table is composed of TIME_KEY, LOCATION_KEY, and PRODUCT_KEY.

Figure 4 Sales Star Schema in MS Access

It is noted that even though the fact table can get very large the dimension tables are typically much smaller. For example a time dimension table with three years of dates still has only about one thousand rows. Compare this to the possible millions of rows in a corresponding fact table. This characteristic of the star schema facilitates data retrieval functions, because most of the time the user will look at the facts through the dimension's attributes. This allows a DBMS to optimize the queries against a star schema data warehouse by first searching the smaller dimension tables before accessing the large fact tables.

IV. An Accounts Receivable Star Schema

This section presents a star schema design to capture transactions related to accounts receivables. The data mart based on this star schema will allow analysis of accounts receivable and

transactions by time, customer, employee, and/or transaction type. Figure 7 shows the star schema for the accounts receivable data mart, which has been implemented in MS Access.

The choice of dimension tables to include in a star schema design depends to a large degree on the query requirements for the database. The time dimension is the one dimension virtually guaranteed to be present in every data warehouse/data mart. Unlike most other dimensions the time dimension table can be built in advance. Even ten years' worth of days is only about 3,650 records. Though this is a fairly simple time dimension, other time attributes can be easily added. For example a number-of-day attribute could be added so that the user can compare the 1st day of one month with the 1st day of another or other months.

sales : Table

Field Name	Data Type
location_key	Number
time_key	Number
product_key	Number
quantity	Number
price	Currency

Field Properties

General | Lookup

Field Size: Long Integer

Format: Auto

Decimal Places: Auto

Input Mask:

Caption:

Default Value: 0

Validation Rule:

Validation Text:

Required: No

Indexed: Yes (Duplicates OK)

Figure 5 Design of Sales Fact Table in MS Access

time_dimension : Table

Field Name	Data Type	Description
time_dimension	AutoNumber	
year	Number	
quarter	Number	
month	Date/Time	
week	Number	
date	Date/Time	

Field Properties

General | Lookup

Field Size: Long Integer

New Values: Increment

Format:

Caption:

Indexed: Yes (No Duplicates)

Figure 6 Design of Time Dimension Table in MS Access

Figure 7 Design of Account Receivables Star Schema

The customer dimension is an important one, as a common use for an accounts receivable data mart is to track and analyze customers' account balance behavior. This dimension contains a minimal set of customer attributes, but other possible attributes to include in the customer dimension might be (1) credit rating, (2) Credit limit, (3) Date of first sale, and (4) Industry code.

The transaction dimension stores the different types of transactions. The most common transaction applied to an invoice is payment. However, other types of transactions are also found. The most common transaction types are: allowance, payment, return, write-off, and invoice.

An allowance would be a credit to a customer's account granted at the company's discretion within its policy guidelines. Generally this would be in response to a customer complaint. Maybe the product is defective in some minor way, or a wrong quantity was sent. In these

cases, the customer agrees to accept the product, as long as some allowance (reduction) is made to the customer's balance owed. A payment should be the most common transaction type to relieve the accounts receivable balance. It would recognize the receipt of a check, or electronic transfer of funds, from the customer. A return would be a credit granted based upon the receipt by the company of goods returned from the customer. A write-off is a recognition of the uncollectability of a debt from a customer. This usually means that the customer is in considerable financial distress. An invoice is a bill charged to the customer. This transaction type is introduced to facilitate the computation of accounts receivable. It will be discussed in greater detail in connection with the fact table. This small dimension can be very useful. For example the user could find the total amount of returns in any given time period or the average percentage of allowance per invoice.

As shown in Figure 7 the employee dimension is a simple one but other attributes such as

department and employee_type could be added. This dimension allows analysis of transactions by employee, such as comparison by department or performance of employees over time.

Noticeably absent from the star schema is the product dimension. This is due to the fact that each transaction may involve multiple products and it may not be possible to distribute the amount of a particular transaction over a number of products. Another popular dimension that is absent is location. Location is not considered in most accounts receivable oriented analyses. However, it would be relatively easy to add the location dimension.

The facts in this star schema design are the daily individual transactions (dollar amounts) for existing invoices. Thus the grain of the fact table is by transaction by customer by employee by day. In other words each record in the fact table represents a transaction from a specific customer for a new or an existing invoice through a particular employee on a particular date. This transaction level granularity allows analysis at, or drill-down to, the transaction level. For example it is possible, with transaction level granularity, to determine the average number of transactions per invoice during a particular period of time or the average invoice balance after 30 days.

A major consideration in the design of an accounts receivable data warehouse/data mart has to be the ease with which customer balances can be computed. The approach taken in this design is to represent the invoice amount as a type of transaction and store the invoice amount as a positive value. Other types of transaction are represented as negative amounts. This structure allows the customer balance to be computed simply through adding all the amounts associated with the customer.

Another interesting field in the fact table is the invoice-number field. This field is necessary to facilitate any analysis by invoice, matching the sale with its credits. It is noted that the invoice-

number is not a fact and it is not related to a dimension. It is not uncommon to include fields such as invoice-number in a fact table without an associated dimension.

The design given above can be used to support very interesting analysis of accounts receivable data. A sales manager, treasurer, CFO, credit manager, or controller might query the data mart for analytical support for a variety of decisions. A sales manager might look for trends by customer, or by salesperson, to identify problems, or to recommend increases in credit limits. The manager might find that some salespersons are more likely than others to grant allowances, or that some customers (or customer types) are more likely to demand them. Patterns in returns could suggest problems in quality, or propensities in certain salespersons to overload customers with inventory.

Financially, a treasurer might look for seasonal patterns in A/R, to project financing needs. More sophisticated analysis might break down these patterns by customer industry code, and then refine the projection with the latest customer industry breakdowns. Financial analysts might be interested in the average A/R balance by customer, or by specific type of customer, to recognize an implicit cost in taking on a new customer. Time series analysis could reveal disturbing trends in allowances, returns, or write-offs, either in total or by any of the dimensions in the database. Days' sales in receivables could be calculated, and customers could be sorted from highest to lowest, exposing those customers who are costing the company the most to carry them, as compared to those who might deserve special appreciation.

A credit manager might look for precursors to write-offs. He or she might learn that write-offs are often preceded by suddenly climbing A/R balances, or by an unusually large number of requests for allowances. A controller might need prior A/R balances to prepare historical balance sheets by industry, in anticipation of a

spin-off of a business segment.

V. Multidimensional Analysis of Accounts Receivable Data

The most attractive feature of a star schema is its support for multidimensional analysis. The star schema presented above permits data analysis by any combination of the four dimensions, employee, customer, transaction type, and time. This section presents examples of multidimensional analysis on the star schema database.

First some ad hoc queries will be examined. The total AR for each customer by the end of a specific month, such as October 1999, can be easily generated by the following query:

```
select      customer_dimension.lastname,
            sum(f.amount) as ar
from fact, customer_dimension, time_dimension
where      fact.customer_key=customer_di-
            mension.customer_key and
            fact.time_key = time_dimension.
            time_key and time_dimension.year
            = 1999 and time_dimension.month
            <= 10
group by   customer_dimension.lastname;
```

It is noted that the query obtains its result by joining the fact table with dimension tables. The rows returned by the fact table are restricted through delimiting the associated dimensional rows. The above query can be slightly changed to return total AR for all customers by removing the group by customer name clause.

Similar questions or queries can be asked about the types of transactions such as allowances. The following query returns the total amount of allowances by customer in 1999:

```
select      customer_dimension.lastname,
            sum(fact.amount) as total
from fact, transaction_dimension, time_dim-
            ension, customer_dimension
```

```
where      f.transaction_key = t.transaction_
            key and f.time_key = t1.time_key
            and t1.year = 1999 and t.trans-
            action_type = 'allowance'
group by   c.lastname
```

The above query gets its result by joining the fact table with the time dimension table, the customer dimension table, and the transaction dimension table and grouping by customer.

Both of the above queries are fairly straightforward and are supported on any RDBMS product. However, far greater insight can be derived by comparing data across different periods of time, for example, the trend in allowances, returns, or write-offs by customer in a specific time period. The cross-tab query feature in MS Access can be used to generate the required data. The query given below returns the total amount transaction by transaction type.

```
transform  sum(fact.amount) as amount
select     transaction_dimension.transaction
            _type
from       transaction_dimension inner join
            (time_dimension inner join fact on
            time_dimension.time_key =
            fact.time_key) on transac-
            tion_dimension.transaction_key =
            fact.transaction_key
where      time_dimension.year=1999 and
            transaction_dimension.transaction
            _type <> 'invoice'
group by   transaction_dimension.transaction
            _type
order by   transaction_dimension.transaction
            _type
pivot      time_dimension.month
```

A similar report on AR trend by customer would not be as simple because the invoice and the associated transactions may not be in the same time period being grouped. It is possible to extract customer AR balances for multiple months using a single query or in one step. But such an approach leads to a high level of com-

plexity while the resulting query could also suffer from performance problems if the fact table is large. A much simpler approach is to generate the report in steps. A closer examination of the report reveals that the report really consists of several sets of data. The first set of data is the monthly customer balance of the first month, the second set of data is the customer balance of the second month, and so on. The best approach is to use a sequence of steps. The following is an example showing how a customer AR trending report for a given period may be produced.

- Create a table to hold the report data if it does not already exist
- Obtain the beginning and ending months representing the period to be reported.
- For each month in the period find the report month customer balances and insert the data into the report table.
- Use a cross-tab query to present the data gathered in the report table.
- Query the report table as desired.

The approach outlined in the above set of steps has been implemented in Access as a subroutine shown below.

```
Private Sub cmdGO_Click()
    Dim REC As Recordset
    Dim DB As Database
    Dim SQLSTR As String
    Dim AMonth As Date
    SQLSTR = "delete from AR;"
    DoCmd.RunSQL (SQLSTR)
    AMonth = Format(Me!txtBegin,
        "mm/dd/yyyy")
    Do While AMonth <= Format(Me!txtEnd,
        "mm/dd/yyyy")
        SQLSTR = "insert into AR (customer,
            ar, month) " _
        & " SELECT c.lastname,
            sum(f.amount), " _
        & Format(AMonth, "yyyymm") & "' " _
        & " FROM fact AS f, customer_dimen-
            sion AS c, time_dimension AS t " _
        & " WHERE f.customer_key =
```

```
        c.customer_key And f.time_key =
            t.time_key " _
        & " and t.month <= " & Month(AMonth)
        & " " _
        & " and t.year = " & Year(AMonth)
        & " " _
        & " GROUP BY c.lastname;"
    AMonth = DateAdd("m", 1, AMonth)
    DoCmd.RunSQL (SQLSTR)
```

```
    Loop
End Sub
```

Figure 8 shows the design of the Account-
Receivables table in MS Access and Figure 9
shows a cross tab query to return customer AR
balance by month for the year 1999. Finally Fig-
ure 10 shows a sample report in MS Access
based on the query. The report was created using
the Report Wizard in MS Access. In fact since a
data warehouse/data mart represents historic
data, there is no need to recompute each month's
AR. Only the current month AR needs to be
computed every month.

VI. Conclusions

This paper presents the design and possible
uses for an accounts receivable star schema. The
star schema, along with some example queries
and reports, was implemented in MS Access. As
methods of data collection become easier and the
cost of data collection is reduced, the challenge
is to derive timely and useful information from
the vast accumulation of data. Data ware-
houses/data marts have now become a common
solution for hosting historic data for intelligent
analysis. The star schema design discussed in
this paper represents the most popular design
method used in today's data warehouses/data
marts.

VII. Suggestions for Future Research

This paper has focused on the design of an
accounts receivable data mart. Such a data mart
may very well be part of a decision support sys-
tem where there are several data marts, each fo-
cusing on a different subject area, such as mar-

	Field Name	Data Type	Description
▶	customer	Text	
	month	Text	
	account_receivables	Currency	

Field Properties

General | Lookup

Field Size: 50

Format:

Input Mask:

Caption:

Default Value:

Validation Rule:

Validation Text:

Required: No

Allow Zero Length: No

Indexed: No

Figure 8 Design of Account_Receivables Table

AR:

- *
- customer
- month
- ar

Field:	customer	month	AR: ar	Left([month],4)
Table:	ar	ar	ar	
Total:	Group By	Group By	Sum	Where
Crosstab:	Row Heading	Column Heading	Value	
Sort:				
Criteria:				'1999'
or:				

Figure 9 Design of Cross Tab Query for Customer AR by Month

Customer AR Balance 1999

customer	199910	199911	199912
Anderson	\$500.00	\$250.00	\$250.00
Guan	\$400.00	\$350.00	\$0.00

Page: 1

Figure 10 Customer AR Balance 1999

keting or sales. A very interesting future research topic is to look at the design of dimensions so that the dimensions can be shared among different data marts with different subject areas. Another interesting area for future research is to examine the issues of data extraction, transformation, and loading for an accounting-oriented data mart or data warehouse. As has been shown in the paper, the data in the data mart consist of several different types of data. Often these different data types come from different application systems that employ different data structures and similar data may be defined differently on different systems. It is very important to examine the common issues in data cleansing, data integration, and data transformation for an accounts receivable data mart/data warehouse. □

References

1. Granof, Michael H., "Government Accounting and Reporting: A Data Warehousing Approach," *The Government Accountants Journal*, Vol. 48, Issue 2, pp. 30-38, 1999.
2. Chen, Ming-Syan, Jiawei Han and Philip S. Yu, "Data Mining: An Overview from a Database Perspective," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp.866-883, 1996.
3. Grabski, Severin V. and Robert J. Marsh, "Integrating Accounting and Manufacturing Information Systems: An ABC and REA-Based Approach," *Journal of Information Systems*, Vol. 8, No. 2, pp. 61-80, 1994.
4. Inmon, William, *Building the Data Warehouse*, John Wiley & Sons, New York, NY, 1996.
5. Kimball, Ralph, *The Data Warehouse Toolkit*, John Wiley & Sons, New York, NY, 1996.
6. Levitan, Alan S., "Deriving Accounts Receivable In An Events-Based, Relational Database System," *The Review of Accounting Information Systems*, Vol. 3, No. 3, pp. 47-52, 1999.
7. McCarthy, William E., "The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment," *The Accounting Review*, Vol. 57, No. 3, pp.554-578, 1982.
8. O'Leary, Daniel E., "REAL-D: A Schema for Data Warehouses," *Journal of Information Systems*, Vol. 13, No. 1, pp.49-62, 1999.
9. Srivastava, Jaideep and Ping-Yao Chen, "Warehouse Creation-A Potential Roadblock to Data Warehousing," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 1, pp. 118-126, 1999.

Notes