ISSN:0975-9646

Pervasive SAS Techniques for Designing a Data Warehouse for an Integrated Enterprise: An Approach towards Business Process

Ardhendu Tripathy, Kaberi Das, Tripti Swarnkar

Department of Computer Applications, I.T.E.R, Shiksha 'O' Anusandhan University
Jagamohan Nagar, Bhubaneswar, India

***Abstract*- In an attempt to find a method to create a stable, fast, easily modified replacement for a Microsoft Access database, SAS rapidly became the only choice as a replacement tool. SAS can very quickly be adapted for uses by relative beginners in novel ways to exploit its strengths as a data extraction and manipulation tool. Using PC SAS and SAS CONNECT for ODBC a method was found to convert the most fragile parts of a Rail Car Setoff reporting database. Aimed at the well-informed MS Office user with exposure to interest in SAS, this paper is intended to introduce PC SAS as a tool that does more than just statistical analysis but also is a viable alternate tool to the more traditional programming solutions. This shift from MS Access to SAS cut processing time from over 20 minutes to less than 4 and reduced the complexity of the database from multiple nested macros down to a single file that is in plain text in easily readable SAS® code.**

***Keywords*- Data Warehouse, OLAP, SAS**

I. INTRODUCTION:

Data warehousing is a collection of ***decision support*** technologies, aimed at enabling the ***knowledge worker*** (executive, manager and analyst) to make better and faster decisions. A data warehouse is a "subject-oriented, integrated, time varying, non-volatile collection of data that is used primarily in organizational decision making." Typically, the data warehouse is maintained separately from the organization's operational databases. There are many reasons for doing this. The data warehouse supports on-line analytical processing (OLAP), the functional and performance requirements of which are quite different from those of the on-line transaction processing (OLTP) applications traditionally supported by the operational databases. OLTP applications typically automate clerical data processing tasks such as order entry and banking transactions that are the bread-and-butter day-to-day operations of an organization. These tasks are structured and repetitive, and consist of short, atomic, isolated transactions. Data warehouses, in contrast, are targeted for decision support. Designing a warehouse database for an Integrated Enterprise is a very complicated and iterative process since it needs data from many departmental units, data cleaning, and requires extensive business modeling.

Therefore, some organizations have preferred to develop a *data mart* to meet requirements specific to a departmental or restricted community of users. Of course, the development of data marts entails the lower cost and shorter implementation time.

II. TWO ENDS OF THE SPECTRUM

In the evolution of building data warehouses and data marts, two practices have emerged. One advocates a centralized enterprise data warehouse that serves as the repository of data from all data sources, from which various data marts can be created. The other advocates a purely bottom-up approach where data marts are built as point solutions to departmental or functional needs isolated from the rest of the enterprise. Companies attempt to tie together these data marts into universal data marts. While the bottom-up approach avoids some of the pitfalls of the enterprise data warehouse, it creates a different set of issues. Simon (1998) referred to these two approaches as the "big bang" and the "loose confederation" approaches respectively. Simon further pointed out that the enterprise data warehouse "big bang" approach attains the maximum data extraction and the highest degree of source-neutral information abstraction. However, as pointed out by Simon, there are many pitfalls, which include cross-organization issues, technical complexity, data semantic issues and long delivery time. On the other hand, the bottom-up approach, while providing the point solutions with respect to departmental or functional needs, will result in information silos within an enterprise over time.

III. THE ENTERPRISE MODEL SOLUTION

The Enterprise Model framework is an extension of the " 3-Schema" architecture originally [17]. The 3-Schema addresses the construct of data based on three levels of representation: the conceptual schema represents the logical view of data, the internal schema represents the physical data storage definitions, and the external schema represents the user application views of data. While the 3-Schema provides the foundation of data definition in the development of databases and their applications, the fundamental concept can be extended to enterprise information management. This extended definition of the 3-Schema will be henceforth called the Enterprise Model.

IV. BUILDING DATA WAREHOUSE DETAILS

A. *The Physical Data Warehouse*

The physical data warehouses and all source systems belong to the Technical Enterprise Model (TEM) ([17], [19]). The Enterprise Data Model (EDM) can be physically realized as one or more Operational Data Stores (ODS). The ODS provides data consolidation of source systems. Data are extracted, transformed and loaded into the ODS from different source systems which include the legacy systems, transactional systems and external systems. It should be observed that the total consolidation of data into one data store might not be practical for large enterprises. The key point of the enterprise modeling approach is that the design allows each data warehouse or data mart to source its data from zero to one or more ODS. In the case of zero ODS, the data warehouse or data mart sources data directly from the source systems. When all data warehouses and data marts source from one big ODS, the architecture is similar to the enterprise data warehouse. The optimal design of the number of ODS for an enterprise will take into consideration the number and size of data sources, and their relationships to the target data warehouses.

The phases of a data warehouse project listed below are similar to those of most database projects, starting with identifying requirements and ending with deploying the system:

- Identify and gather requirements
- Design the dimensional model
- Develop the architecture, including the Operational Data Store (ODS)
- Design the relational database and OLAP cubes
- Develop the data maintenance applications
- Develop analysis applications
- Test and deploy the system

B. *Identify and Gather Requirements:*

Identify sponsors: A successful data warehouse project needs a sponsor in the business organization and usually a second sponsor in the Information Technology group. Sponsors must understand and support the business value of the project. Understand the business before entering into discussions with users. Then interview and work with the users, not the data learn the needs of the users and turn these needs into project requirements.

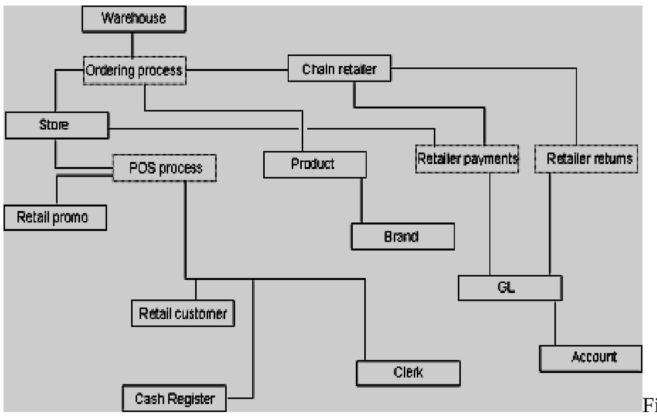
It is the data warehouse designer's job to determine what data is necessary to provide the information. Topics for discussion are the user’s objectives and challenges and how they go about making business decisions. Business users should be closely tied to the design team during the logical design process; they are the people who understand the meaning of existing data. Many successful projects include several business users on the design team to act as data experts and "sounding boards" for design concepts.

C. *Design the Dimensional Model:*

The dimensional model must suit the requirements of the users and support ease of use for direct access ([1], [2]). The model must also be designed so that it is easy to maintain and can adapt to future changes. The model design must result in a relational database that supports OLAP cubes to provide "instantaneous" query results for analysts.

An OLTP system requires a normalized structure to minimize redundancy, provide validation of input data, and support a high volume of fast transactions. A transaction usually involves a single business event, such as placing an order or posting an invoice payment.

An OLTP model often looks like a spider web of hundreds or even thousands of related tables. In contrast, a typical dimensional model uses a star or snowflake design that is easy to understand and relate to business needs, supports simplified business queries, and provides superior query performance by minimizing table joins.



Fi

Fig. 1. Flow Chart

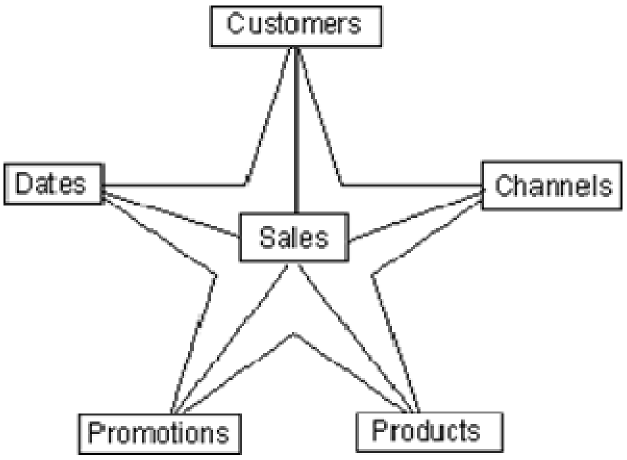


Fig. 2. Star Diagram

1) Dimensional Model Schemas: The principal characteristic of a dimensional model is a set of detailed business facts surrounded by multiple dimensions that describe those facts. When realized in a database, the schema for a dimensional model contains a central fact table and multiple dimension tables. A dimensional model may produce a ***star schema*** or a ***snowflake schema***.

2) Star Schemas: A schema is called a ***star schema*** if all dimension tables can be joined directly to the fact table. The following diagram shows a classic star schema.

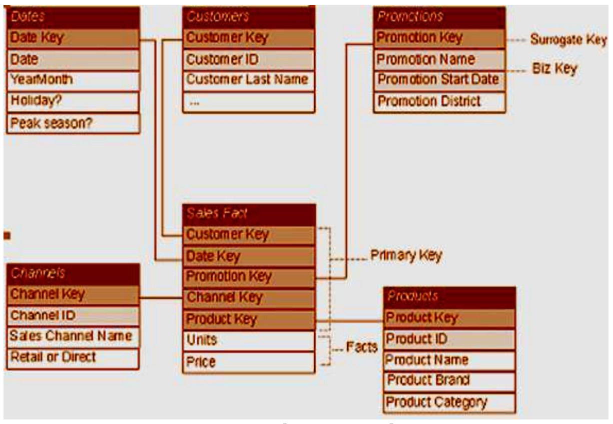


Fig. 3. Classic star schema

3) Snowflake Schemas: A schema is called a ***snowflake schema*** if one or more dimension tables do not join directly to the fact table but must join through other dimension tables. For example, a dimension that describes products may be separated into three tables (*snowflake*) as illustrated in the following diagram.

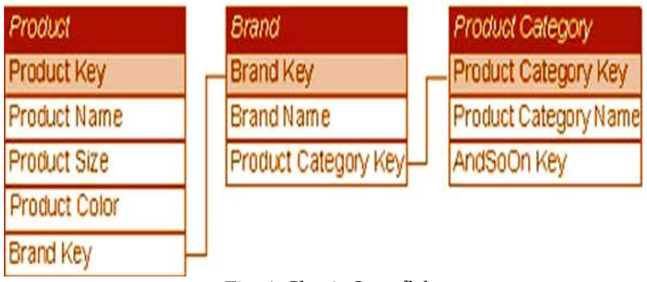


Fig. 4. Classic Snowflake

4) Star or Snowflake: Both star and snowflake schemas are dimensional models; the difference is in their physical implementations. Snowflake schemas support ease of dimension maintenance because they are more normalized. Star schemas are easier for direct user access and often support simpler and more efficient queries. The decision to model a dimension as a star or snowflake depends on the nature of the dimension itself, such as how frequently it changes and which of its elements change, and often involves evaluating tradeoffs between ease of use and ease of maintenance. It is often easiest to maintain a complex dimension by snow flaking the dimension. By pulling hierarchical levels into separate tables, referential integrity between the levels of the hierarchy is guaranteed. Analysis Services reads from a snowflaked dimension as well as, or better than, from a star dimension.

D. Developing the Architecture:

The data warehouse architecture reflects the dimensional model developed to meet the business requirements [8]. Dimension design largely determines dimension table design and fact definitions determine fact table design. Whether to create a star or snowflake schema depends more on implementation and maintenance considerations than on business needs. Information can be presented to the user in the same way regardless of whether a dimension is snowflaked. Data warehouse schemas are quite simple and straightforward, in contrast to OLTP database schemas with their hundreds or thousands of tables and relationships. However, the quantity of data in data warehouses requires attention to performance and efficiency in their design.

E. Design the Relational Database and OLAP Cubes:

The star or snowflake schema is created in the relational database, surrogate keys are defined and primary and foreign key relationships are established. Views, indexes, and fact table partitions are also defined. OLAP cubes are designed that support the needs of the users.

F. Develop the Data Maintenance Applications:

The data maintenance applications, including extraction, transformation, and loading processes, must be automated, often by specialized custom applications. Data Transformation Services (DTS) in SQL Server 2000 is a powerful tool for defining many transformations. Other tools are Transact-SQL and applications developed using scripting such as Microsoft Visual Basic® Scripting Edition (VBScript) or Microsoft JScript®, or languages such as Visual Basic.

G. Develop Analysis Application:

The applications that support data analysis by the data warehouse users are constructed in this phase of data warehouse development. OLAP cubes and data mining models are constructed using Analysis Services tools, and client access to analysis data is supported by the Analysis Server. Other analysis applications, such as Microsoft PivotTables®, predefined reports, Web sites, and digital dashboards, are also developed in this phase, as are natural language applications using English Query. Specialized third-party analysis tools are also acquired and implemented or installed. Details of these specialized applications are determined directly by user needs.

H. Test and Deploy the System:

It is important to involve users in the testing phase. After initial testing by development and test groups, users should load the system with queries and use it the way they intend to after the system is brought on line. Substantial user involvement in testing will provide a significant number of benefits. Among the benefits are:

- Discrepancies can be found and corrected.
- Users become familiar with the system.
- Index tuning can be performed.

Data warehousing approaches and techniques are well established, widely adopted, successful, and not controversial ([4], [7], [9]). Dimensional modeling, the foundation of data warehouse design, is not an arcane art or science; it is a mature methodology that organizes data in a straightforward, simple, and intuitive representation of the way business decision makers want to view and analyze their data. The key to data warehousing is data design. In this paper we are going to describe how SAS tool will help us to design a data warehouse for an integrated enterprise.

SAS is an excellent tool to replace either the entire system or key parts of it. Making changes like this can lead to more stability, better performance, and can simplify internal structure of the system. Faced with the challenge of maintaining a legacy Access database that is key to CPR's business a decision was made to re-write a number of key sections. This database had passed through the hands of a number of business developers in several different departments leaving the database complicated and fragile. On reviewing the issues, the most fragile section seemed to be the macros and queries used to collect and manipulate the data weekly and monthly. Two options were open at that point:

- Rewrite the process using VBA and redevelop the queries for stability.
- Re-develop the process in a different tool.
-

Due to the complicated relationships inside the Access database, it seemed simpler to develop a new reporting process.

SAS is driven by SAS programs that define a sequence of operations to be performed on data stored as tables. SAS enabled the redevelopment of the extraction routines and then synthesis of the data in to its end report ready state. SAS components expose their functionalities via application programming interfaces, in the form of statements and procedures. SAS Library Engines and Remote Library Services allow access to data stored in external data structures and on remote computer platforms.

V. DESCRIPTION OF SAS:

The DATA step section of a SAS program, like other database-oriented fourth-generation programming languages such as SQL or Focus, assumes a default file structure, and automates the process of identifying files to the operating system, opening the input file, reading the next record, opening the output file, writing the next record, and closing the files [16]. This allows the user/programmer to concentrate on the details of working with the data within each record, in effect working almost entirely within an implicit program loop that runs for each record. Typical tasks include printing or performing statistical analysis, and may just require the user/programmer to identify the data set.

VI. COMPONENTS OF SAS:

SAS consists of a number of components, which organizations separately license and install as required:

A. *SAS Add-In for Microsoft Office:*

A component of the SAS Enterprise Business Intelligence Server is designed to provide access to data, analysis, reporting and analytics for non-technical workers (such as business analysts, power users, domain experts and decision makers) via menus and toolbars integrated into Office applications.

B. *Base SAS:*

The core of SAS, the so-called **Base SAS Software**, manages data. **SAS procedures software** analyzes and reports the data. The SQL procedure allows SQL (Structured Query Language) programming in lieu of data step and procedure programming. Library Engines allow transparent access to common data structures such as Oracle, as well as pass-through of SQL to be executed by such data structures. The **Macro facility** is a tool for extending and customizing SAS software programs and reducing overall program verbosity. The **DATA step debugger** is a programming tool that helps find logic problems in DATA step programs. The **Output Delivery System** (ODS) is an extendable system that delivers output in a variety of formats, such as SAS data sets, listing files, RTF, PDF, XML, or HTML. The **SAS windowing environment** is an interactive, graphical user interface used to run and test SAS programs.

C. *BI Dashboard:*

It is a plugin for Information Delivery Portal. It allows the user to create various graphics that represent a broad range of data. This allows a quick glance to provide a lot of information, without having to look at all the underlying data.

D. SAS Enterprise Business Intelligence Server: Includes both a suite of business intelligence (BI) tools and a platform to provide uniform access to data. The goal of this product is to compete with Business Objects and Cognos' offerings.

E. Enterprise Guide: SAS Enterprise Guide is a Microsoft Windows client application that provides a guided mechanism to use SAS and publish dynamic results throughout an organization in a uniform way. It is marketed as the default interface to SAS for business analysts, statisticians, and programmers. Though Data Integration Studio is the true ETL tool of SAS, Enterprise Guide can be used for the ETL of smaller projects.

F. OLAP Cube Studio: A client application that helps with building OLAP Cubes.

G. SAS/ACCESS: Provides the ability for SAS to transparently share data with non-native data sources.

H. SAS/ACCESS for PC Files: Allows SAS to transparently share data with personal computer applications including MS Access and Microsoft Office Excel.

I. SAS/CONNECT: Provides ability for SAS sessions on different platforms to communicate with each other.
SAS/Warehouse Administrator: Superseded in SAS 9 by SAS ETL Server.

VII. FEATURES OF SAS:

- Read and write many different file formats.
- Process data in many different formats.
- SAS programming language, a 4th generation programming language.
- DATA steps are written in a 3rd-generation procedural language very similar to PL/I; SAS PROCS.
- SAS AF/SCL is a fifth generation programming language is similar in syntax to Java.
- Many built-in statistical and random number functions.
- Hundreds of built-in functions for manipulating character and numeric variables.
- System of formats and informats. These control representation and categorization of data.
- Comprehensive date and time-handling functions.
- Interaction with database products through a subset of SQL (and ability to use SQL internally to manipulate SAS data sets).
- SAS/ACCESS modules allow communication with databases (including databases accessible via ODBC.
- Direct output of reports to CSV, HTML, PCL, PDF, PostScript, RTF, XML, and more using Output Delivery System. Templates, custom tag sets, styles incl. CSS and other markup tools available and fully programmable.
- Fast development time, particularly from the many built-in procedures, functions, in/formats, the macro facility, etc.
- An integrated development environment.
- Dynamic data-driven code generation using the SAS Macro language.
- Can process files containing millions of rows and thousands of columns of data.

VIII. DESIGN ENVIRONMENT REQUIREMENT FOR ENTERPRISE DATA WAREHOUSE

In addition to meeting the specific project requirements, a set of needs is projected onto the design environment and tools being used to implement and manage the project itself ([17], [18], [19]). Since the enterprise data warehouse will have a wide range of uses and therefore higher complexity, an underlying object orientation of design is key. These objects are considered metadata, or information about the data, its processing, and use in the enterprise. From a design standpoint, the warehouse development team needs to bring sources together, transform and merge them as needed, and ultimately arrive at a target or endpoint configuration.

A design environment is used to perform these tasks, and depends on components to implement each type of operation related to sources, transformations, and targets. We can use the following terminology for these areas:

A. *Source Designers:*

These components support connection to source systems, whether they are databases, operational systems, web information, or any other regularly-used type of data. Wizard-based panels guide the user through the exploration and discovery process to locate the needed information.

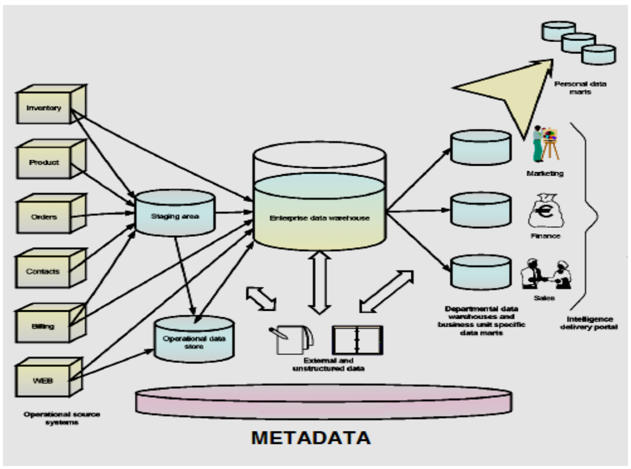

Inventory
Product
Orders
Contacts
Billing
WEB
Operational source systems
Staging area
Operational data store
Enterprise data warehouse
External and unstructured data
Personal data marts
Marketing
Finance
Sales
Departmental data warehouses and business unit specific data marts
Intelligence delivery portal
METADATA

Fig.4. a Typical large Data Warehouse with metadata underlying the entire structure

B. *Transformation Components:*

These make it easy to perform common operations like joining multiple data sources in various ways, sorting or sub setting data based on some criteria, or other data-based operations.

C. *Target Designers:*

These components support the population of a specific endpoint for the data flow. This can refer to a data organization such as a star schema, or a particular destination requirement like writing analytic results back into a specific location in an operational system.

IX. RELATED WORK

Our related work includes designing a data warehouse for an integrated enterprise and here we will begin with designing a warehouse for a banking sector using SAS as a tool for development, So before going into detail design of our work let us have a briefing regarding benefits of implementing SAS for detail data store for banking also regarding the industry version of banking DDS.

A. *Describing SAS Detail Data Store for Banking:*

- Expanded data model coverage with additional tables and fields. The majority of the additional data is for additional functionality in the latest versions of SAS Credit Risk Management for Banking and SAS Credit Scoring for Banking.
- DDL and metadata for a DB2 version of the banking DDS (Base SAS software, SAS Scalable Performance Data Server, and Oracle continue to be included in the banking DDS).

- The banking DDS is a data store that serves as the single version for the SAS Banking Intelligence Solutions.
- It contains the atomic-level data and historical information that is needed to populate the solution data marts.
- A huge range of valuable business analysis is possible utilizing the Banking Data Warehouse, spanning Customer Relationship Management, profitability analysis, risk, etc.

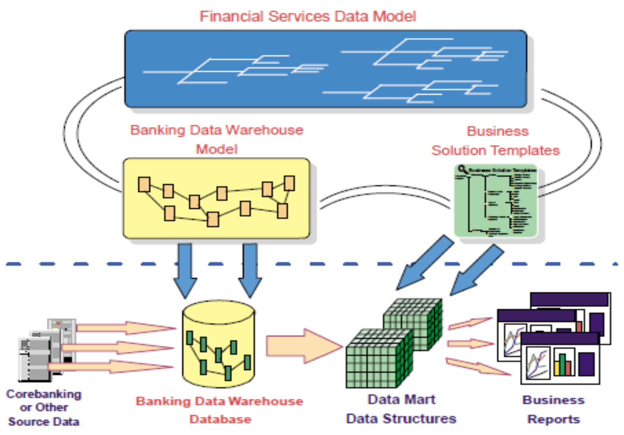

Financial Services Data Model
Banking Data Warehouse Model
Business Solution Templates
Corebanking or Other Source Data
Banking Data Warehouse Database
Data Mart Data Structures
Business Reports

Fig. 5. Integrated Solution Architecture for Banking Data Warehouse

B. *Benefits of Implementing the SAS Detail Data Store for Banking:*

- The banking DDS provides a single data target for loading data.
- Because the definition of the banking DDS table is known, the extract, transform, and load (ETL) process from the banking DDS to the solution data marts will be pre-built.
- SAS Banking Intelligence Solutions can more easily share system data with each other. For example, the definition of a customer table is the same for SAS Credit Scoring for Banking as it is for SAS Credit Risk Management for Banking. Therefore, populating a single definition of a customer table ensures that both of these solutions have a single version of the truth.
- Data that is created from SAS Banking Intelligence Solutions can be stored in a central location and shared with other solutions. For example, the credit scores from SAS Credit Scoring for Banking are written back to the banking DDS and shared with SAS Credit Risk Management for Banking.

C. *Industry Version Of Banking DDS:*

The banking DDS is part of a larger group of industry data models. Although it would be convenient to have a single data model that covers all industries, in reality, different industries have different data needs.
For example, customer, supplier, product, and segment tables share similar data attributes across industry.

Data more likely to differ considerably across industry is the customer-facing or front office data. For example, in banking, there are accounts; in retail, there are transactions; and in insurance, there are premiums and claims. Because of this difference in data, the industry versions of the banking DDS contain tables that are part of the base, cross-industry data model, and contain tables that are part of the industry-specific data model.

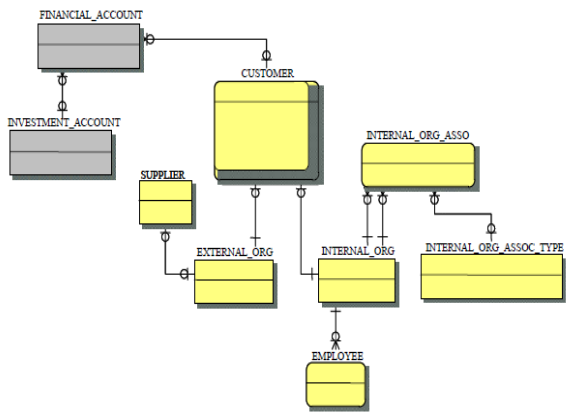


Fig. 6. Sample of a Base Data Model with Industry-Specific Tables

D. *Organizing Tables in the SAS Detail Data Store for Banking Sector:*

At a high level, the banking DDS can be grouped into subject areas such as:

1) *Parties:*
   - This subject area includes information on the parties that are involved in banking, such as customers and counterparties.
   - Customer information includes details of individuals, organizations, and corporate customers, associated addresses and contact information, household information for individuals, organization information for corporate customers etc.
2) *Financial Accounts:*
   - This subject area includes all of the types of financial accounts.
   - The information includes attributes that are common to all financial accounts, such as date opened and account balance.
   - These accounts include loan, mortgage, core banking, credit card, investment, Bonds, Share Trading, Gold, life insurance, auto insurance, property insurance, protection insurance, travel insurance, and retirement.
3) *Banking Accounts:*
   - **Mortgage Account:** Includes information such as the account branch.

- **Loan Account:** Includes information such as the interest rate and the amount of the loan.
- **Core Banking Account:** Includes information such as the overdraft charge amount and the interest rate.
- **Credit Card Account:** Includes information such as the payment protection status and the payoff date.

4) *Banking Transactions:*

- It includes transactions that are related to traditional banking accounts, such as withdrawals and deposits.
- Includes information that is related to the nature of the transaction such as the transaction amount or the channel of transaction.

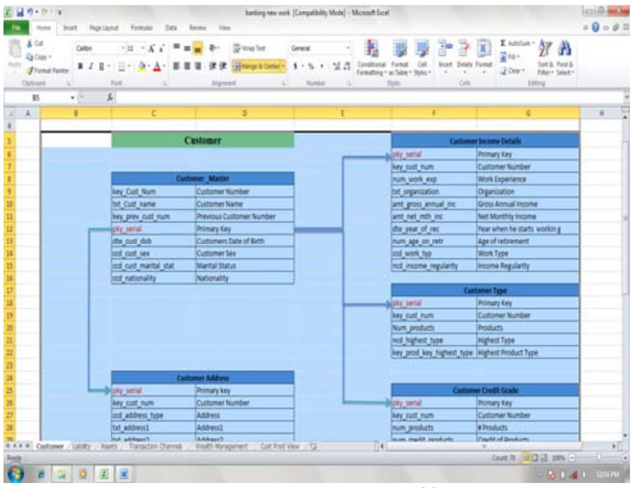
5) *Financial Reporting:* This subject area includes tables that relate to the financial reporting area of an organization. The financial reporting area contains critical data, including information on annual revenues, net sales revenue, annual interest charges and profit before and after tax, extraordinary income, and other financial measures. The logical model report provides a drill-down feature.

X. DATA MODEL SAMPLE DESIGN OF OUR WORK

Here we begin our work with designing various data models using Microsoft Excel in which we started creating various table and tried to derive the relationship among these tables.

A. Customer Master:

- It contains Customer_Master table.
- Customer Address Details table.
- Customer Income Details table.
- Customer Type table.
- Customer Credit Grade table.



Customer

Customer_Master	
key_Cust_Num	Customer Number
txt_Cust_name	Customer Name
key_prev_cust_num	Previous Customer Number
pky_serial	Primary Key
dte_cust_dob	Customers Date of Birth
ccd_cust_sex	Customer Sex
ccd_cust_marital_stat	Marital Status
ccd_nationality	Nationality

Customer Address	
pky_serial	Primary key
key_cust_num	Customer Number
ccd_address_type	Address
txt_address1	Address1

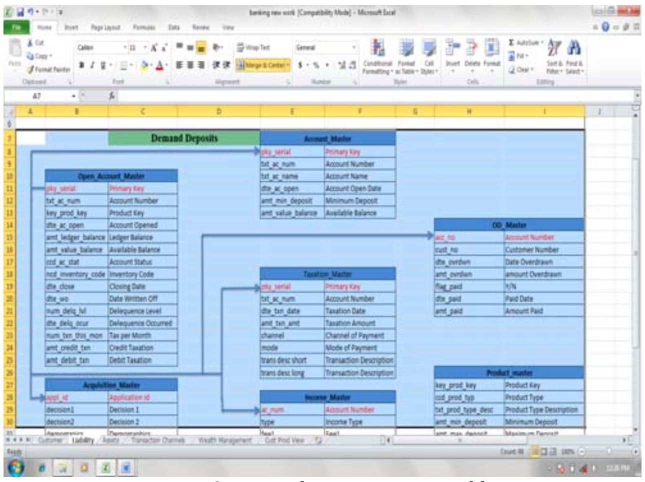
Customer Income Details	
pky_serial	Primary Key
key_cust_num	Customer Number
num_work_exp	Work Experience
txt_organization	Organization
amt_gross_annual_inc	Gross Annual Income
amt_net_mth_inc	Net Monthly Income
dte_year_of_rec	Year when he starts workin g
num_age_on_retr	Age of retirement
ccd_work_typ	Work Type
ncd_income_regularity	Income Regularity

Customer Type	
pky_serial	Primary Key
key_cust_num	Customer Number
Num_products	Products
ncd_highest_type	Highest Type
key_prod_key_highest_type	Highest Product Type

Customer Credit Grade	
pky_serial	Primary Key
key_cust_num	Customer Number
num_products	# Products

Fig. 7. Customer Master Table

B. Liability Master: It contains Demand Deposit and Term Deposit.



Demand Deposits

Open_Account_Master	
pky_serial	Primary Key
txt_ac_num	Account Number
key_prod_key	Product Key
dte_ac_open	Account Opened
amt_ledger_balance	Ledger Balance
amt_value_balance	Available Balance
ccd_ac_stat	Account Status
ncd_inventory_code	Inventory Code
dte_close	Closing Date
dte_wo	Date Written Off
num_delq_lvl	Delequence Level
dte_delq_ocur	Delequence Occured
num_txn_this_mon	Tax per Month
amt_credit_txn	Credit Taxation
amt_debit_txn	Debit Taxation

Acquisition_Master	
appl_id	Application Id
decision1	Decision 1
decision2	Decision 2

Account_Master	
pky_serial	Primary Key
txt_ac_num	Account Number
txt_ac_name	Account Name
dte_ac_open	Account Open Date
amt_min_deposit	Minimum Deposit
amt_value_balance	Available Balance

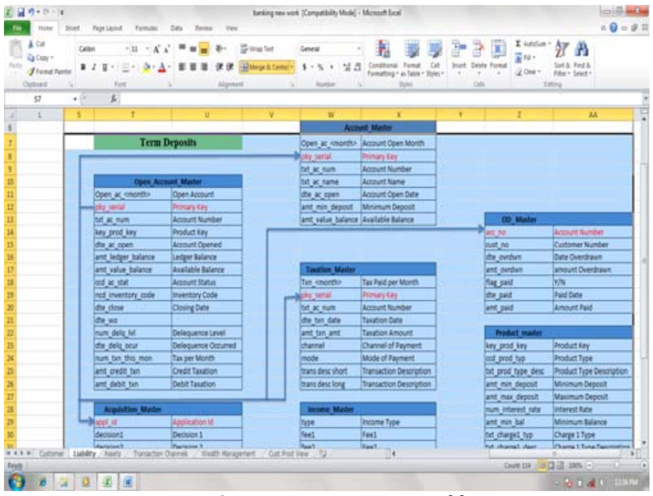
Taxation_Master	
pky_serial	Primary Key
txt_ac_num	Account Number
dte_txn_date	Taxation Date
amt_txn_amt	Taxation Amount
channel	Channel of Payment
mode	Mode of Payment
trans desc short	Transaction Description
trans desc long	Transaction Description

Income_Master	
ac_num	Account Number
type	Income Type

OD_Master	
acc_no	Account Number
cust_no	Customer Number
dte_ovrdwn	Date Overdrawn
amt_ovrdwn	amount Overdrawn
flag_paid	Y/N
dte_paid	Paid Date
amt_paid	Amount Paid

Product_master	
key_prod_key	Product Key
ccd_prod_typ	Product Type
txt_prod_type_desc	Product Type Description
amt_min_deposit	Minimum Deposit

Fig. 8. Demand Deposit Master Table



Term Deposits

Open_Account_Master	
Open_ac_<month>	Open Account
pky_serial	Primary Key
txt_ac_num	Account Number
key_prod_key	Product Key
dte_ac_open	Account Opened
amt_ledger_balance	Ledger Balance
amt_value_balance	Available Balance
ccd_ac_stat	Account Status
ncd_inventory_code	Inventory Code
dte_close	Closing Date
dte_wo	
num_delq_lvl	Delequence Level
dte_delq_ocur	Delequence Occured
num_txn_this_mon	Tax per Month
amt_credit_txn	Credit Taxation
amt_debit_txn	Debit Taxation

Acquisition_Master	
appl_id	Application Id
decision1	Decision 1

Account_Master	
Open_ac_<month>	Account Open Month
pky_serial	Primary Key
txt_ac_num	Account Number
txt_ac_name	Account Name
dte_ac_open	Account Open Date
amt_min_deposit	Minimum Deposit
amt_value_balance	Available Balance

Taxation_Master	
Txn_<month>	Tax Paid per Month
pky_serial	Primary Key
txt_ac_num	Account Number
dte_txn_date	Taxation Date
amt_txn_amt	Taxation Amount
channel	Channel of Payment
mode	Mode of Payment
trans desc short	Transaction Description
trans desc long	Transaction Description

Income_Master	
type	Income Type
fee1	Fee1

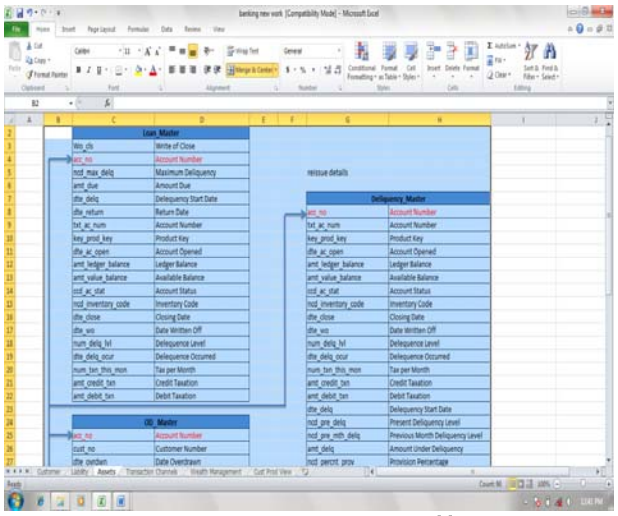
OD_Master	
acc_no	Account Number
cust_no	Customer Number
dte_ovrdwn	Date Overdrawn
amt_ovrdwn	amount Overdrawn
flag_paid	Y/N
dte_paid	Paid Date
amt_paid	Amount Paid

Product_master	
key_prod_key	Product Key
ccd_prod_typ	Product Type
txt_prod_type_desc	Product Type Description
amt_min_deposit	Minimum Deposit
amt_max_deposit	Maximum Deposit
num_interest_rate	Interest Rate
amt_min_bal	Minimum Balance
txt_charge1_typ	Charge 1 Type

Fig. 9. Term Deposit Master Table

C. Assests Master:

- It contains Loan_ Master Table.
- OD_Master Table.
- Delequency_Master Table.



Loan_Master	
Wo_cls	Write of Close
acc_no	Account Number
ncd_max_delq	Maximum Deliquency
amt_due	Amount Due
dte_delq	Delequency Start Date
dte_return	Return Date
txt_ac_num	Account Number
key_prod_key	Product Key
dte_ac_open	Account Opened
amt_ledger_balance	Ledger Balance
amt_value_balance	Available Balance
ccd_ac_stat	Account Status
ncd_inventory_code	Inventory Code
dte_close	Closing Date
dte_wo	Date Written Off
num_delq_lvl	Delequence Level
dte_delq_ocur	Delequence Occured
num_txn_this_mon	Tax per Month
amt_credit_txn	Credit Taxation
amt_debit_txn	Debit Taxation

OD_Master	
acc_no	Account Number
cust_no	Customer Number
dte_ovrdwn	Date Overdrawn

reissue details

Deliquency_Master	
acc_no	Account Number
txt_ac_num	Account Number
key_prod_key	Product Key
dte_ac_open	Account Opened
amt_ledger_balance	Ledger Balance
amt_value_balance	Available Balance
ccd_ac_stat	Account Status
ncd_inventory_code	Inventory Code
dte_close	Closing Date
dte_wo	Date Written Off
num_delq_lvl	Delequence Level
dte_delq_ocur	Delequence Occured
num_txn_this_mon	Tax per Month
amt_credit_txn	Credit Taxation
amt_debit_txn	Debit Taxation
dte_delq	Delequency Start Date
ncd_pre_delq	Present Deliquency Level
ncd_pre_mth_delq	Previous Month Deliquency Level
amt_delq	Amount Under Deliquency
ncd_percnt_prov	Provision Percentage

Fig. 10. Assests Master Table

D. *Implementation Work:*

After successful design of the data model we moved ahead with the designing of ETL (extract, transform, load). For this we used Microsoft Access which would work as a database to store data for our work. We started creating files for each of the Master Tables where the sample datas could be stored. We created a database called Bank which consisted of datas of each Master Table. After designing the database we proceeded further with the ETL work. For this we took the help of SAS Application.

1) *Coding for Account Master Table:*

```
data dic;
input source_table $ source $ target_table $ target
$ label $ informat $ format $ conv $;
datalines;
DDACM DDACN dd_ac_master txt_ac_num ACCOUNT 8. 8.
NILL
;
data dic;
set dic;
if conv = 'NILL' then conv = '';
run;

filename test catalog 'work.test.cet.source';

// catalog will be created

if _n_ then put "Data dd_ac_master; set DDACM;";
code = trim(target)||' =
'||trim(source_table)||'.'||trim(source)||';';
put code;
run;

filename test catalog 'work.test.cet.source' mod;
data _null_;
set dic end = _last_;
file test;
{

}
put code;
run;

filename test catalog 'work.test.cet.source' mod;
data _null_;
set dic end = _last_;
file test;
{

}
put code;
if _last_ then put "run;";
run;
```

2) *Coding for Customer Master*

```
data dic;
input source_table $ source $ target_table $ target
$ label $ informat $ format $ conv $;
datalines;
DDCSM DDCSID dd_cs_master txt_ac_num;
data dic;
set dic;
if conv = 'NILL' then conv = '';
run;

filename test catalog 'work.test.cet.source';

data _Null_;

//catalog will be created

code = trim(target)||' =
'||trim(source_table)||'.'||trim(source)||';';
put code;
run;

filename test catalog 'work.test.cet.source' mod;
{;


}
set dic end = _last_;
file test;
code = "Format "||trim(target)||' =
"'||trim(format)||'";';
put code;
run;

filename test catalog 'work.test.cet.source' mod;
```

XI. CONCLUSION

This report presents an experimental evaluation of the speedup obtained by using the DWS technique. This technique takes advantage of the specific characteristics of star schemas and typical data warehouse query profile to guarantee optimal load balance of query execution and assure high scalability. In data warehouse striping fact tables are distributed over an arbitrary number of computers and the queries are executed in parallel by all the computers, guaranteeing a nearly optimal speedup. We are now on the verse of designing our ETL process and are on the process of designing a warehouse for a Banking Sector.

REFERENCES

[1] Jens Albrecht, Holger Gunzel, Wolfgang Lehner, "An Architecture for Distributed OLAP", Int. Conf. PDPTA 1998.

[2] APB-1 Benchmark, Olap Council, November 1998. Available at www.olpacouncil.org.

[3] P.M. G. Apers, C. A. van den Berg, J. Flokstra, P. W. P.J. Grefen, M. L. Kersten, A. N. Wilschut, "PRISM WDB: A parallel, main-memory, relational DBMS," IEEE Transactions on Knowledge and Data Engineering, December 1992,541 -554.

[4] H. Boral, et al, "Prototyping Bubba, A highly parallel database system", IEEE Transactions on Knowledge and Data Engineering, Vol. 2, March 1990, pp.4-24.

[5] Chee-Yong Chan and Yannis E. Ioannidis, "Bitmap index design and evaluation", Proc. of ACM SIGMOD Intemational Conference on Management of Data, 1998, pp.355 - 366.

[6] D. **J.** DeWitt et al., ***"The*** Gamma Database Machine Project", IEEE Transactions on Knowledge and Data Engineering, Vol. 2, March 1990, pp.44-62.

[7] D. J. DeWitt and Jim Gray, "Parallel Database Systems: The future of high performance database systems", Communications of the ACM, ***35(6),*** June 1992, pp.85-98.

[8] Pedro Furtado and H. Madeira, "Analysis of Accuracy of Data Reduction Techniques", First International Conference, DaWaK'99, Florence, Italy, pp.377-388.

[9] G. Graefe, "Query evaluation techniques for large databases", ACM Computing Surveys, 25(2):73-170, 1993.

[10] Joseph M. Hellerstein, "Online Processing Redux", Data Engineering Bulletin, Vol. 20, N"3, September 1997, pp. 20-29.

[11] Ralph Kimball. The'Data Warehouse Toolkit. Ed. J. Wiley & Sons, Inc, 1996.
[12] Richard J. Lipton, Jeffrey F. Naughton and Donovan A. Schneider; "Practical selectivity estimation through adaptive sampling", Proc. of ACM SIGMOD Intemational Conference on Management of Data, 1990, pp.1 - 11.
[13] Hongjun Lu, Beng Chin. Ooi, and Kian Lee Tan. Query Processing in Parallel Relational Database Systems. IEEE Computer Society, May 1994.
[14] Patrick E. ONeil, Dallan Quass, "Improved Query Performance with Variant Indexes", Proc. of ACM SIGMOD Int. Conference on Management of Data, 1997, pp.38-49.
[15] Oracle Corporation, Oracle RDBMS Version 7 Parallel Server Administrators Guide.
[16] Mehler, G. (2006). Data warehousing for the Enterprise, SAS Institute Inc., Cary North Carolina, 143-27.
[17] C. Adamson, M. Venerable. *Data Warehouse Design Solutions.* J. Wiley & Sons, Inc.1998.
[18] R. Agrawal, A. Gupta, S. Sarawagi. *Modeling Multidimensional Databases.* ICDE 1997.
[19] C. Ballard. *Data Modeling Techniques for Data Warehousing.* SG24-2238-00. IBM Red Book. ISBN number 0738402451. 1998.
[20] M. A. R. Kortnik, D. L. Moody. *From Entities to Stars, Snowflakes, Clusters, Constellations and Galaxies: A Methodology for Data Warehouse Design.* 18th. International Conference on Conceptual Modelling. Industrial Track Proceedings. ER'99.
[21] S. Chaudhuri, U. Dayal. *An overview of Data Warehousing and OLAP Technology.* SIGMOD Record 26(1). 1997.
[22] M. S. Hacid, U. Sattler (DWQ project). *An Object-Centered Multi-dimensional Data Model with Hierarchically Structured Dimensions.* Proc. of the IEEE Knowledge and Data Engineering Workshop. 1997.