A Case Study on a Implementation of Marketing Data Analysis System

Jianhui He

A Project Report

In

The Department

Of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

June 1999

0-612-39113-2

ABSTRACT

A Case Study on a Implementation of
Marketing Data Analysis System

Jianhui He

This report documents the implementation of a data warehousing initiative for the purpose of marketing data analysis. Implementation of this project was divided into two phases. The objective of phase one is to produce a concept-proof prototype. Phase two, of which I took major responsibility, is to generate an actual production system. Major tasks I performed in phase two covered many aspects of the data warehousing life cycle: revised and fine-tuned the conceptual, logical and physical data model; performed database redesign and database sizing; built and rebuilt the database to improve performance; improved data extraction, transformation and loading process; performed database and SQL performance tuning; planned and implemented information presentation with off the shell data access tools.

The first part of the report reviews the data warehousing literature by examining its evolution, conceptual model, major architectural components and some critical issues involved. In the second part of the report, the implementation of a marketing data warehouse is examined in details. A system overview is provided along with the logical data model. It then describes the mainframe component, UNIX components, presentation/end user component and the interaction among them.

The Appendix provides further technical details of the project.

Table of Contents

1 Literature Review

1.1 Evolution of Data Warehousing

1.1.1 Historical Perspective of Data Analysis System

Practically all business system development was done on the IBM mainframe computers using tools such as Cobol, CICS, IMS, DB2, etc during the 1970's. The new mini-computer platforms such as AS/400 and VAX/VMS arrived in the 1980's. UNIX became a popular server platform in the late eighties due to the introduction of client/server architecture.

A remarkably large number of business applications continue to run in the mainframe environment despite all the changes in the platforms, architectures, tools, and technologies. These systems have grown to capture the business knowledge and rules that are incredibly difficult to carry to a new platform or application. It is estimated that more than 70 percent of business data for large corporations still reside in the mainframe environment.

These systems continue to be the largest source of data for analysis systems. They are generically named legacy systems. It is common for an institution to generate countless reports and extracts over the years, each designed to extract requisite information out of these legacy systems. Ordinarily IS/IT groups assume responsibility for designing and developing programs for these reports and extracts. It usually takes a long time to generate and deploy these programs and frequently turns out to be more time consuming than the end users had originally thought.

Other categories of popular analysis systems have been the decision support systems (DSS) and executive information systems(EIS). Decision support systems focus more on detail and are targeted towards lower to mid-level managers. Executive information systems generally provide a higher level of consolidation and a multi-dimensional view of the data, since high level executives need the ability to slice and dice the same data and to drill down to review the data detail. DSS and EIS never entered the mainstream due to the high cost of development and their inherent coordination difficulties in production deployment.

1.1.2 Changing Business Environment

The economic downturn of the late eighties caused many global corporations to experience a remarkable period of consolidation. Business process reengineering and downsizing forced businesses to reevaluate their business practices. The layers of middle management that historically had distilled data were cut and thus such responsibilities were passed upward. Users were then able to be cross-trained to analyze their own information. The desktop PC has made many users comfortable with database manipulation due to the distribution of computing power and expertise generated by it. The demand from such users was to have hands-on access to historical data to analyze trends, exceptions and root causes.

While competing with other companies in vastly different cultures and economic environments corporations have found markets for their products globally. Business and acquisition mergers also crossed country boundaries. This globalization of business has not only increased the need for more continuous analysis, but also to manage data in a centralized location.

Data warehousing is one of the technological solutions to a business environment that is constantly changing.

1.1.3 Recent Technological Advancements

The enormous forward movement in the hardware and software technologies had significantly influenced the quick evolution of the data warehousing discipline. The sharp decline in prices and the increasing power of computer hardware, along with the ease of use of today's software, has made the analysis of hundreds of gigabytes of information and business knowledge possible.

The commodity computer market has profited enormously due to the development of the Pentium II and Alpha processors. Sophisticated processor hardware architectures such as symmetric multi-processing have improved mainstream computing inexpensively. A key component influencing the performance of a data warehouse system which are now available at very low prices are higher capacity memory chips. Computer Bus such as PCI and controller interfaces such as Ultra SCSI have made I/O incredibly fast. Another important note is that the disk drive has been reduced to hold amazing amounts of information.

Windows NT and Unix are server operating systems that have brought mission-critical stability and powerful features to the distributed computing environment. The operating system software has become very feature-rich and powerful while the cost has been going down steadily. This combination has caused sophisticated operating system concepts such as virtual memory, multi-tasking, and symmetric multi-processing to be available on inexpensive operating platforms. Due to innovations such as powerful personal productivity software, easy–to-use graphical interface, and responsive business applications, the personal computer has become the focal point of all computing today. The powerful desktop hardware and software has paved the way for development of the client/server or multi-tier computing architecture. A large amount of data could be accessed by personal computer based tools due to these factors. These tools vary from very simple query capabilities included in most productivity packages to incredibly powerful graphical multi-dimensional analysis tools.

These advancements make data warehousing feasible to the majority of corporations, not only technically but also financially.

1.2 Data Warehousing Conceptual Model

One of the fastest growing client/server applications is data warehousing. It is oriented towards information analysis and decision support, not towards operation or transaction processing. William Inmon, who coined the term "data warehouse" in 1990, defined a data warehouse as a managed database in which the data is:

- Subject oriented: There is a shift from application-oriented data (i.e. data designed to support application processing) to decision-support data(i.e., data designed to aid in decision making). Subject-oriented data provides a stable image of business processes, independent of legacy systems. In other words, it captures the basic nature of the business environment.
- Integrated: The data warehouse consolidates application data from different legacy systems (usually means old-style mainframe databases) which use different encoding, measurement units, and so on, and eliminates inconsistencies in the data.
- Time-variant: Informational data has a time dimension. Each data point is associated with a point in time, and data points can be compared along the time axis unlike operational data which is valid only at the moment of access.
- Nonvolatile : New data is always appended rather than replaced. The database continually absorbs new data, integrating it with the previous data.

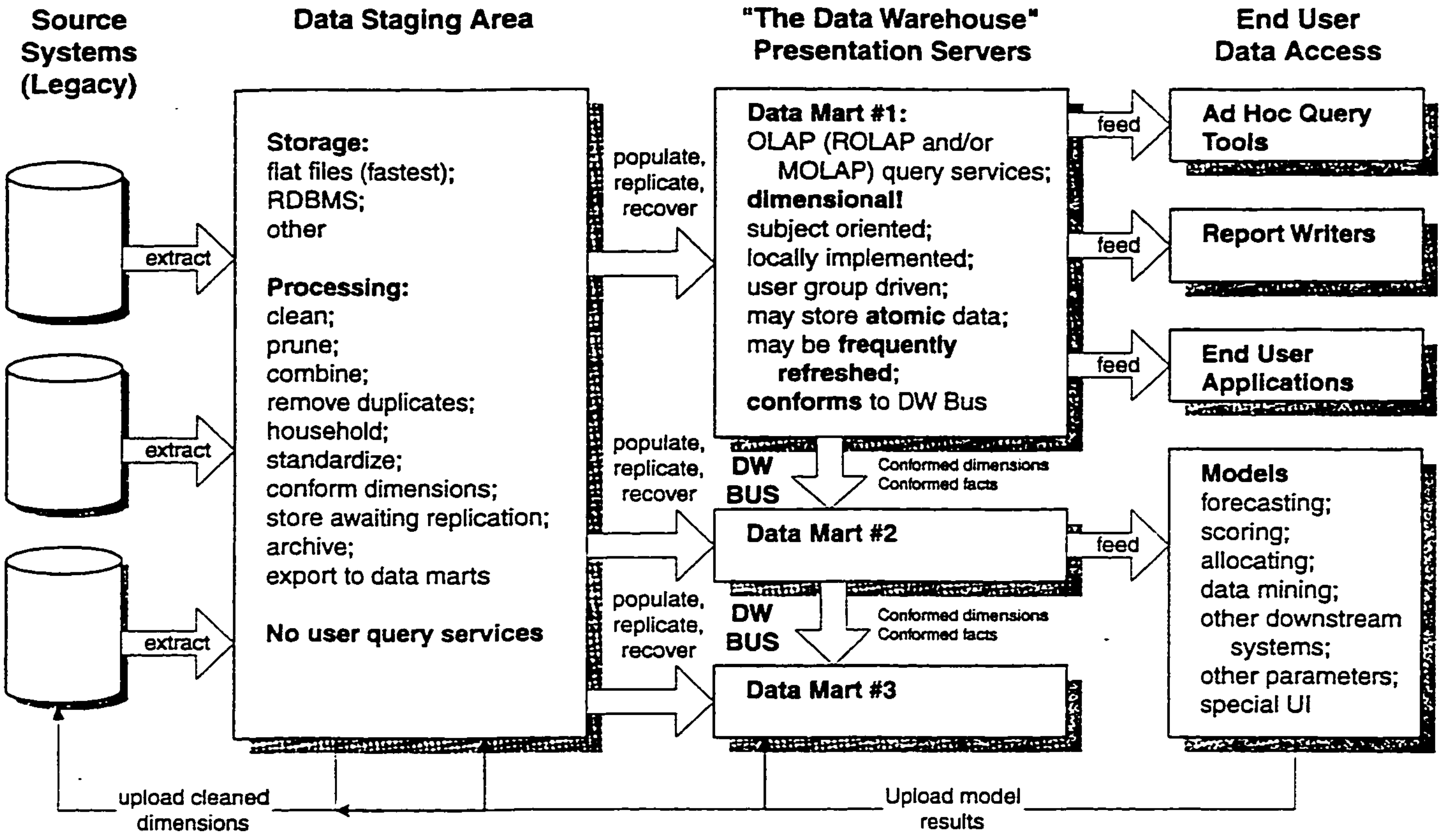
One of the major purposes of data warehousing is to build a database separately from transactional databases since analytic data and transactional data are different in terms of requirements and user communities. The transactional databases are not suitable for analytic purposes because:

- Transactional databases simply contain raw data, and thus, the processing speed will be considerably slower
- Transactional databases are not designed for queries, reports and analyses uses; therefore, performance is poor on those tasks. They also do not store the historic data that is necessary for data analyses.
- Transactional databases are inconsistent in the way that they represent information. For example, different databases may use varying units of measurement for the same attributes.

1.3 Architectural Component of Data Warehousing Systems

1.3.1 Component Diagram

The following diagram shows a schematic overview of the key components of a data warehousing system[1].



[1] The diagram is an extract from "The Data Warehouse Life Cycle Toolkit" by R. Kimball, Wiley, 1998

1.3.2 Component Descriptions

Kimball et al.(1998) provided a good description of the different architectural components of a data warehousing system. Source System is an operational system of recording which functions are to capture the transactions of the business. A source system is often called a "legacy system" in a mainframe environment. The main priorities of the source system are uptime and availability. Queries against source systems are narrow, "account-based" queries that are part of the normal transaction flow and severely restricted in the demands of the legacy system. Data Staging Area is a storage area and set of processes that clean, transform, combine, de-duplicate, household, archive, and prepare source data for use in the data warehouse. The Presentation Server is the target physical machine on which the data warehouse data is organized and stored for direct querying by end users, report writers, and other applications.

The Dimensional Model is a specific discipline for modeling data which is an alternative to entity-relationship modeling. A dimensional model contains the same information as an E/R model but packages the data in a symmetric format which design goals are user understandability, query performance, and resilience to change. Fact tables and dimension tables are the main components of a dimensional model. A fact table is the primary table in each dimensional model that is meant to contain measurements of the business. The most useful facts are numeric and additive. Every fact table represents a many-to-many relationship and a set of two or more foreign keys that join to their respective dimension tables. A dimension table is one of a set of companion tables to a fact table. Each dimension is defined by its primary key that serves as the basis for referential integrity with any given fact table to which it is joined. Most dimension tables contain many textual attributes that are the basis for constraining and grouping within data warehouse queries. A Business Process is a coherent set of business activities that make sense to the business users of our data warehouses.

Data Mart is a logical subset of the complete data warehouse. Data Warehouse is the source of data that may be queryed in the enterprise. The data warehouse is simply the union of all the constituent data marts. A data warehouse is fed from the data staging area. The data warehouse manager is responsible both for the data warehouse and the data staging area. The Operational Data Store serves as the point of integration for operational systems. This was especially important for legacy systems that grew independent of each other.

ROLAP is a set of user interfaces and applications that give dimension to a relational database. MOLAP is a set of user interfaces, applications, and proprietary database technologies that are strongly dimensional. End User Application is a collection of tools that query, analyze, and present information targeted to support a specific business need. A client of the data warehouse is End User Data Access Tool. In a relational data warehouse, such a client maintains a session with the presentation server while sending a stream of separate SQL requests to the server. Eventually the end user data access tool had completed the SQL session and results in the presentation of a screen of data, a report, a graph, or some other higher form of analysis to the user. An end user data access tool can be as simple as an ad hoc query tool, or as complex as a sophisticated data mining or modeling application. Sophisticated data access tools such as modeling or forecasting tools may actually upload their results into special areas of the data warehouse. Ad Hoc Query Tool is a specific kink in the end user data access tool that invites the user to form their own queries by directly manipulating relational tables and their joining matter. The information in the data warehouse environment that is not the actual data itself is referred to as Metadata.

Modeling Application is a sophisticated kink in the data warehouse client with analytic capabilities that transform or digest the output from the data warehouse. Modeling applications include:

- Forecasting models that attempt to predict the future
- Behavior scoring models that cluster and classify customer purchase behavior or customer credit behavior
- Allocation models that take cost data from the data warehouse and spread the costs across product groupings or customer groupings
- Most data mining tools

1.4 Data Warehousing Basic Processes

1.4.1 Data Staging Process

According to Kimball et al.(1998), data staging is a major process that includes extracting, transforming, loading indexing, and quality assurance checking.

- Extracting. Extraction step is the first step of getting data into the data warehouse environment. Extracting means reading and understanding the source data, and copying the parts that are needed to the data staging area for further work.
- Transforming. Once the data is extracted into the data staging area, there are many possible steps in transformation
    - Cleaning the data by correcting misspelled words, resolving domain conflicts (such as a city name that is incompatible with a postal code), dealing with missing data elements, and parsing into standard formats
    - Purging selected fields from the legacy data that are not useful for the data warehouse
    - Combining data sources, by matching exact key values or by performing fuzzy matches on non-key attributes, including looking up textual equivalents of legacy system codes
    - Boosting the performance of common queries in order to build aggregates
- Loading and Indexing. When the transformation process is completed, the data is in the form of loaded record images. Loading in the data warehouse environment usually takes the form of replicating the dimension tables and fact tables and presenting these tables to the bulk loading facilities of each recipient data mart. Record-at-a-time loading is far slower than bulk loading, which is a very important capability.
- Quality Assurance Checking. Running a comprehensive exception report on the entire set of newly loaded data can test quality assurance. Reporting categories must be present and totals must be satisfactory. All reported values must be consistent with the time series of similar values which preceded them. The data mart's end user report writing facility could be used to build the exception report.

1.4.2 Other Data Warehousing Processes

Other data warehousing processes include:

- Release/Publishing. The user community must be notified that the new data is ready once each data mart has been freshly loaded and quality assured. The nature of any changes that have occurred in the underlying dimensions and new assumptions that have been introduced into the measured or calculated facts is communicated through publishing.
- Updating. Modern data marts may well be updated, contrary to the original beliefs surrounding the data warehouse. It is apparent that incorrect data should be corrected. The changes often triggered in the original data stored in the data marts that comprise the data warehouse are labels, hierarchies, status, and corporate ownership. These are known as "managed load updates", not the transactional updates.
- Querying. A broad term that encompasses all the activities of requesting data from a data mart, including ad hoc querying by end users, report writing, complex decision support applications, requests from models, and full-fledged data mining is a query. Querying never takes place in the data staging area. A query takes place on a data warehouse presentation server. Querying is the reason behind using the data warehouse.
- Data Feedback/Feeding in Reverse. First step: It is now possible to upload a cleaned dimension description from the data staging area to a legacy system. This can be done once the legacy system recognizes the values of the improved data. Second step: It is possible to upload the results of a complex query or a model run or a data mining analysis back into a data mart. This would be a way to capture the value of a complex query that takes the form of many rows and columns that satisfies the user's needs.
- Auditing. It is sometimes critical to know where the data came from and which calculations were performed. It is now possible to create special audit records. A user can ask for the audit record of the data at any time due to the fact that the audit records are linked directly to the real data.

- Securing. Every data warehouse is faced with a challenging dilemma: the need to publish the data widely to as many users as possible with the easiest-to-use interfaces, as well as protecting the valuable sensitive data from hackers, snoopers, and industrial spies. The severity of this has increased due to the development of the Internet. Users' accessibility to all the constituent data warehouse must be done through a single sign-on. Warehouse security must be managed from a single console.
- Backing Up and Recovering. Data warehouse data is a flow of data from the legacy systems through the data marts and finally onto the users' desktops. This is where the real question about where to take the necessary snapshots of the data for archival purposes and disaster recovery arises. Furthermore, it may be even more complicated to back up to recover all of the metadata that is the background of the data warehouse operation.

1.5 Data Warehousing Considerations

Sakaguchi, T. et al.(1996) conducted a literature survey on the advantages and disadvantages of data warehousing technology.

1.5.1 Advantages of a Data Warehouse

- Simplicity is the main advantage of data warehousing. Data warehousing simplifies business because it provides a single image of business reality by integrating various data. The benefits of data warehouses include allowing existing legacy systems to continue in operation, consolidating inconsistent data from various legacy systems into one coherent set, and reaping benefits from vital information about current operations. Current and past operations may be monitored and compared. In addition, predictions of future operations can be made, new business processes can be developed and new operational systems would multiply greatly to support those processes. Large amounts of historical data and corporate wide data that companies require to turn into vital business information can be stored in data warehouses. One of the benefits offered by data warehouses is a single, centralized data location while maintaining local client/server distribution. Data warehouses are company wide systems; therefore, they improve corporate wide communication.
- Better quality data and improved productivity are other advantages. Data consistency, accuracy and documentation are improvements in productivity. Better decision making through OLAP and data mining analysis made against the data warehouse created the improvements. An increase in data access is another advantage. The workload of IS can be cut since data warehouses allow users to retrieve necessary data by themselves. The systems response time should be reduced since the necessary data is in one place.
- Another advantage is the ease of use. Data warehouse enables easy access to business data without slowing down the operational database by taking some of the operational data and putting it in a separate database which means queries from users do not interfere with the normal operations. They are targeted at end users

since the focus is on subjects and support on-time, ad-hoc queries for fast decision-making as well as regular reporting.

- Separation between decision-support operation and production operation. In order to separate operational, continually updated transaction from historical data required for business analysis data warehouses are built. Managers and analysts can then use historical data for their decision-making activities without slowing down the production operation.
- Data warehouse gives a competitive advantage. Businesses become more competitive, understand customers better, and more rapidly meet market demands since data warehouses better manage and utilize corporate knowledge.
- Data warehouse facilitates distributed database. Data warehouses pull together information from different and potentially incompatible locations throughout the organization and optimize its use. In order to link those disparate data sources, middleware, data transfer software and other client/server tools are used. A data warehouse is an ultimate distributed database.
- Low operation cost. New operational systems can be created due to the fertile ground data warehouses provide. Once the initial investment is covered, the organization's information-technology group generally required fewer resources since it eliminates paper based files.

1.5.2 Disadvantages of A Data Warehouse

- Complexity and anticipation in development. The main disadvantage is complexity in development. A data warehouse cannot just be bought off the shelf. The IS would have to build one because each warehouse has a unique architecture and a set of requirements that evolve from the organizations. Data warehouse construction requires a sense of anticipation about future ways to use the collected records. The constantly changing needs of their company's business and the capabilities of the available and emerging hardware and software need to be taken into consideration by developers. The manner in which to scale the warehouse to meet increasing user demand for both volume and complexity makes its development more complex. Choosing the right products may be difficult in developing such a large database.
- Takes time to build. It takes 2-3 years to build a data warehouse. IS directors or others wishing to develop a warehouse may spend an inordinate amount of time justifying the need when there is no strong executive sponsorship.
- Expensive to build. A data warehouse may cost up to $2 to 3 million to build. Data warehouses are so expensive is because data must be moved or copied from existing databases, sometimes manually, and it needs to be translated into a common format.
- Lack of API. Data warehousing software lacks a set of application programming interfaces or other standards that shuttle data through the entire warehouse process.
- End-user training. Employees must be prepared to capitalize upon the innovative data analysis provided by data warehouses and to create a new way of thinking. Those end users would require extensive training. In order to educate all constituents, it is essential to have a good communication plan.
- Complexity involved in SMP and MPP. The complexity of data warehousing will increase if the warehouses involve symmetrical multiprocessing and massive parallel processing. Achieving synchronization and shared access is difficult.
- Difficulty in distributed database environment. The data warehouse is a method of bringing various data together, and is centralized by its very nature. Since many companies are still in the preliminary stages of putting their data warehouses together,

this centralization means only workers located at the same site as the warehouse have access to the data.

- Time-lag between data warehouses and operation. The data in data warehouses is extracted from operational databases that are in continual change. Real-time replication while maintaining a full-scale data warehouse is impossible. Data warehouse store only a fraction of corporate data that is steadily drifting backward losing relevance until the warehouse is replenished.

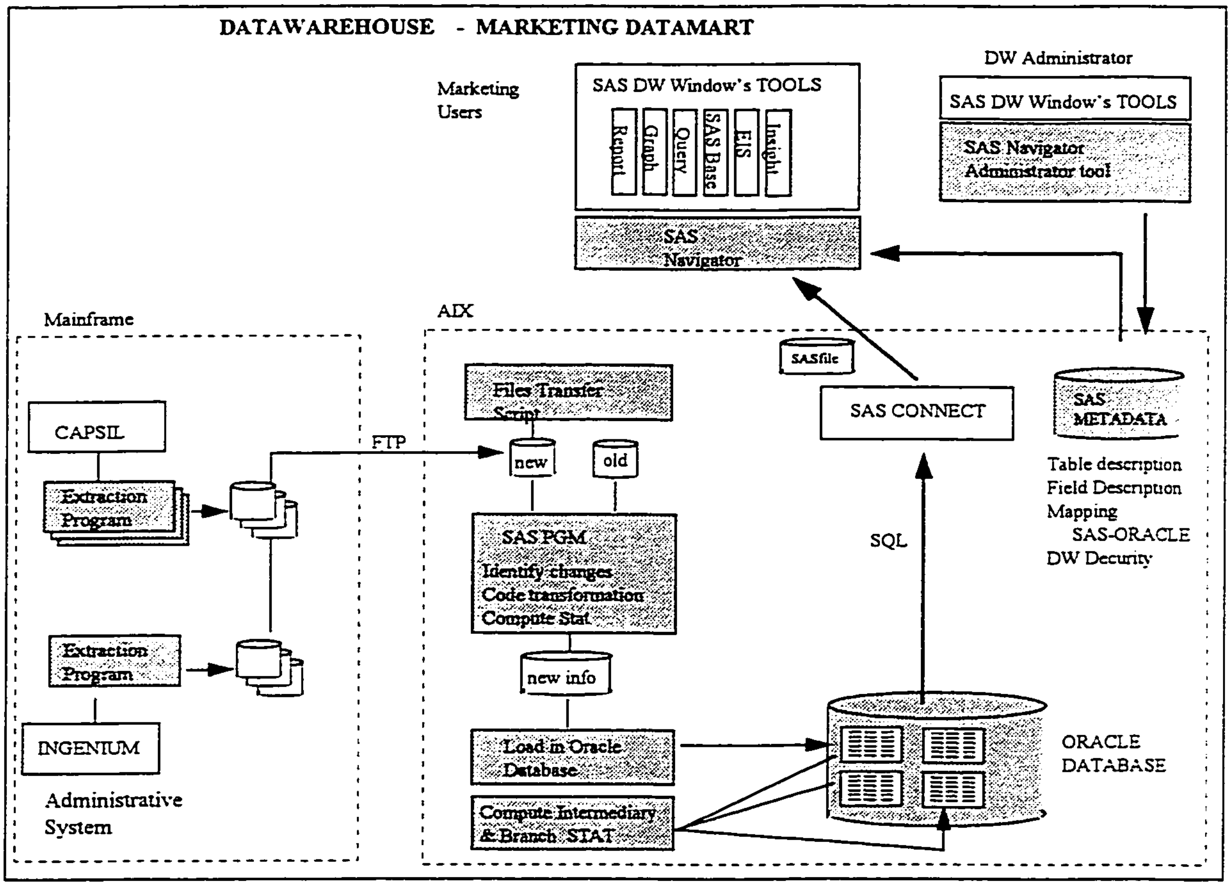
2 Implementation of a Marketing Data Warehouse[2]

2.1 System Description

2.1.1 Global View

This section describes a global view of all components related to a marketing data warehouse. Each component executes on a different platform. First, basic information is contained in operational systems on mainframe. Insurance policy related data are extracted from two mainframe systems[3] twice a month. After the first step is completed, these extracted files are transferred to an AIX platform to be cleaned and loaded in tables of an ORACLE Database. At the end, marketing users access these tables using MS ACCESS/ODBC and SAS tools from their personal computer.

This following diagram shows a global view of all components



[2] this part is an excerpt from a technical project report written by He, Jianhui in 1999 right after the project was finished.

[3] CAPSIL and INGENIUM are the names of the two mainframe systems

The marketing data warehouse is refreshed twice a month. The update cycle is launched the fifteenth and last days of the month (e.g.: 98-01-15,98-01-31,98-02-15,98-02-28...)

2.1.1.1 Mainframe

At each cycle, two independent sets of jobs are launched.

For Capsil 6 jobs are triggered by Zeke to extract and produce 8 files:

- Policy
- Coverage
- Commission
- Branch
- Producer (agent)
- NAAD
- Plan code description
- Province description

For Ingenium 1 job is triggered by Zeke and produces 6 files:

- Policy
- Coverage
- Commission
- NAAD
- Plan code description
- Province description

Those files stay on the mainframe until the next cycle. Files are managed on a GDG base for historical and restart purpose(5 generations are kept). There is no control file produced by the mainframe application to assure that all files are produced correctly. Manual verifications are done on AIX after the transfer, to assure that extract programs had run.

2.1.1.2 AIX Platform

The major transformation runs on the AIX platform. The AutoSyst scheduler is used to synchronize the jobs on mainframe with the AIX Scripts.

The major steps are:

- **Transfer :**

  The extracted files are transferred to the Unix machine. The AIX script used FTP (get) to copy files from mainframe to AIX directory.

- **Changes identification:**

  SAS programs are used to compare previous extracts with new ones to identify only changes in extracted data. This process must be done because Capsil does not keep all time stamps when change occurs.

  > In Release 2 of the marketing datamart, a new strategy was implemented: data snapshot was taken at the end of each refresh cycle and such images kept in flat files. To minimize the impact of such change of strategy, all the SAS programs used to identify changes as the previous paragraph describes remain the same, instead a new step is added in the refresh cycle to delete the previous extracts. Therefore, when the SAS programs run, all data in the new extracts is identified as the changes.

- **Transformation**

  Data from the mainframe is transformed to produce more comprehensible information. Code and type are changed in long description. Derivation rules are applied to field (seniority segment is computed with hire date and actual date). New fields are also computed like the number of coverage within a policy.

- **Database Load**

  Output files from the transformation step are loaded in Oracle database using Oracle SQL Loader.

- **Statistics compute from basic tables**

  SQL procedures (scripts) are used to summarize intermediary and branch statistics from the policy details tables. SQL scripts are also used to generate year to date statistics.

2.1.1.3 User's Accessing Tools

Once the data is stored in the Oracle database, end-users can use the MS ACCESS/ODBC and SAS access tools. The SAS software includes these tools:

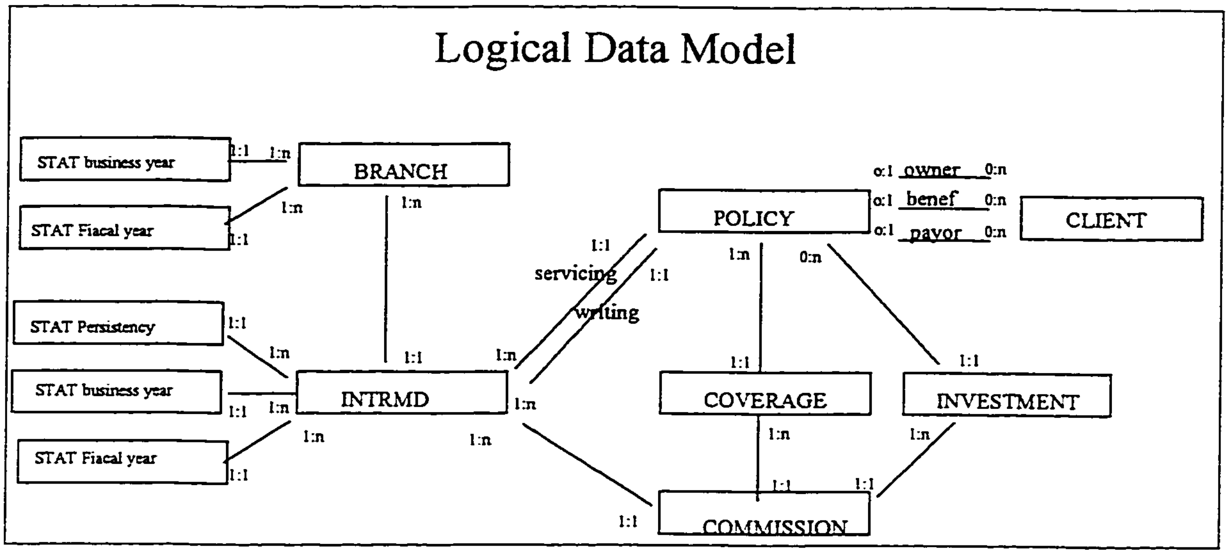
- *Navigator* tool[4]	Description of tables, fields and subjects, data filtering
- Viewer	Spreadsheet like data presenter
- Report	Report generator
- Query	End user tool to build a query without knowledge of SQL
- Graph	Graph Building Utilities
- EIS	Tool to build EIS type application (drill down report, graph, etc)
- Insight (trial)	Data mining tool

After selecting table and fields, users can apply filters and download a subset of the corporate table on their PC. Results can be viewed in the viewer, drop on report, graph, insight, EIS template or export on other PC tool like Excel.

[4] Navigator ***is not a SAS product***, it is a framework first prepared by SAS developers for us to maintain description on table and field. It also provides security management and the capability to organize and structure information by subject. This tool can be modified and adapted by the IT dept or modifications can be done by SAS developers with an extra cost.

2.1.2 Logical Data Model

The following ER diagram lays out the major entities in the marketing data warehouse and the relationship among these entities. Product information is captured by the POLICY, COVERAGE, INVESTMENT and COMMISSION entities. The company information is described by BRANCH, INTRM and five statistics entities. The CLIENT provides information of the company customer base.



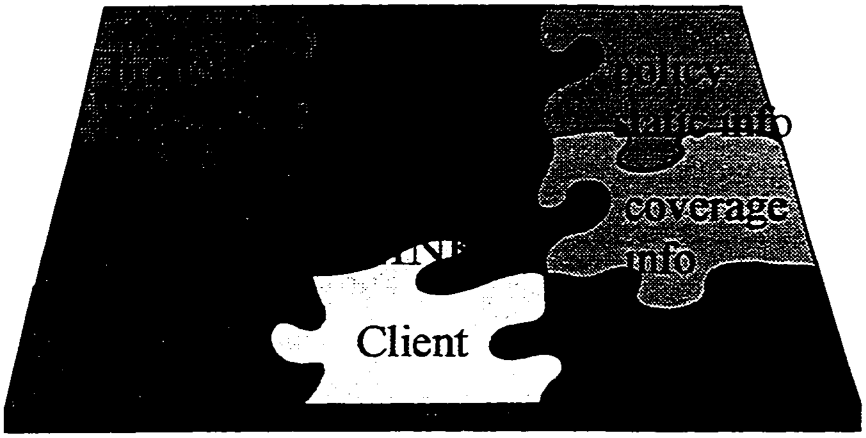
2.1.3 Concept of Keeping Multiple Images of Policy Data

One requirement asked by end-users is to keep periodic snapshots or image of policy data. The actual database design permits to keep 24 images of a policy by year. This feature allows users to go back in the past and produce a report of a specific period. To manage this feature we have to keep each policy 24 times by year. For 160,000 policies, it has a large space requirement. This figure describes all blocs of information related to a policy.

Management of historical data

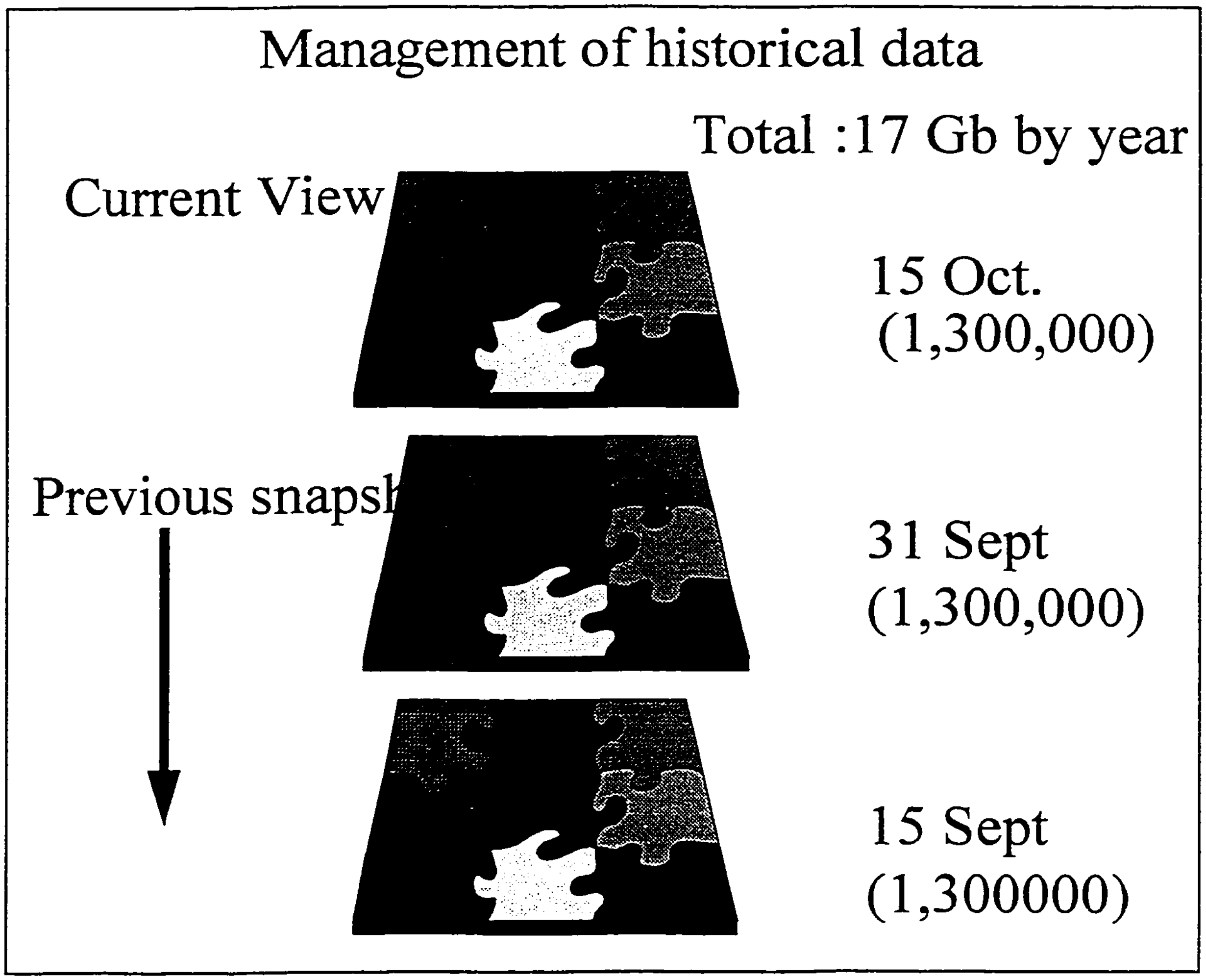
150000 policies
300000 coverages
400000 commissions
450000 clients
TOTAL:1,300,000 rows

24 periods by year



lot of static info in policy data

2.1.3.1 Release I Design



Management of historical data

Total :17 Gb by year

Current View

Previous snaps

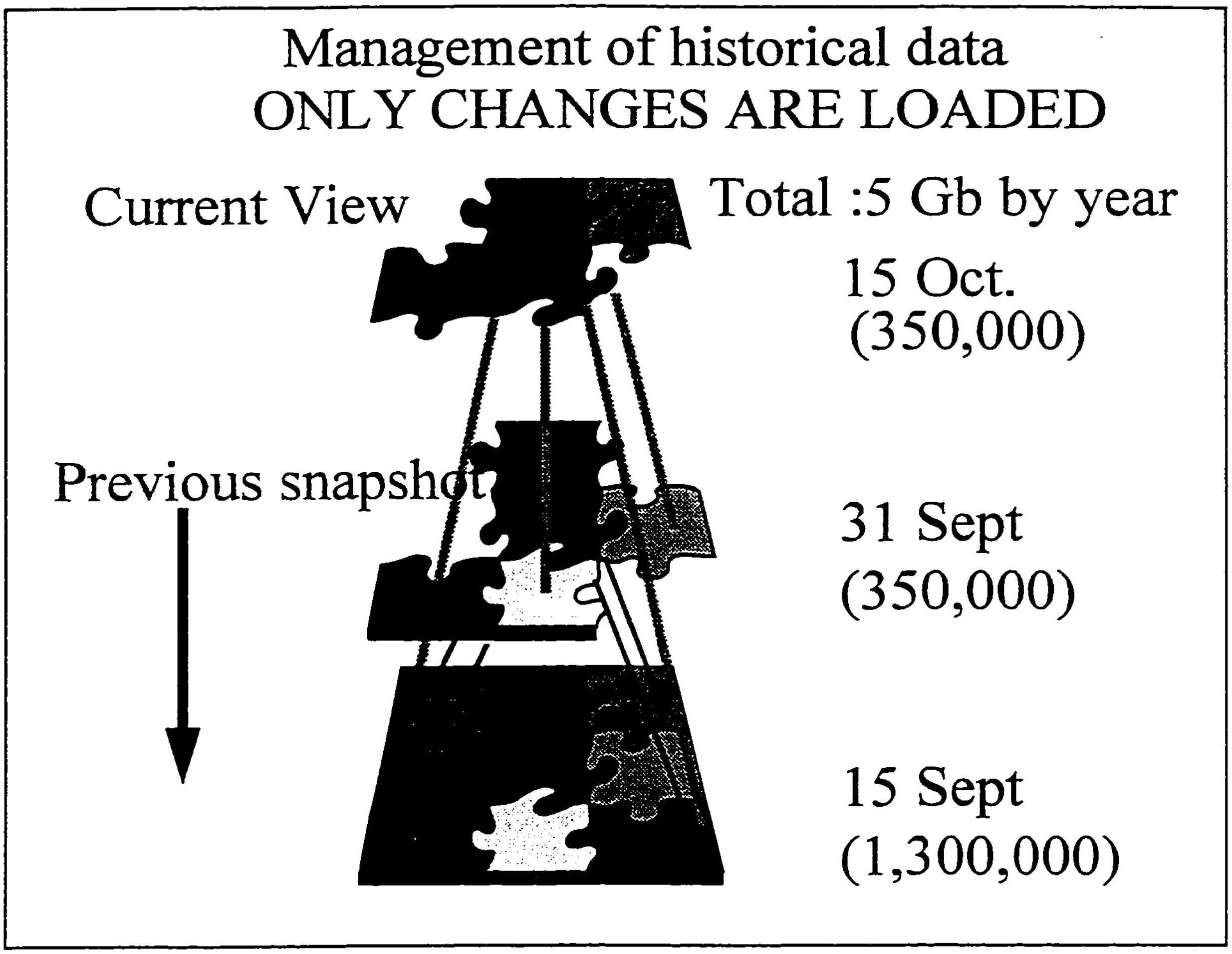
15 Oct.
(1,300,000)

31 Sept
(1,300,000)

15 Sept
(1,300000)

To keep all the historical data, we need around 17G of disk space by year. The actual AIX server rent for the pilot project contained only 5G that can be used by the production database (other 5G are used for temporary space, test environment and software like Unix, Oracle, SAS, Metadata file, etc)

To resolve the space problem, we did a special design that stored only piece of information that had changed between two cycles. This way 24 period can be stored in 5G. The next figure describes the way information is stored in the Database. We kept a special table that contains pointers to the most recent part of a policy.



Management of historical data
ONLY CHANGES ARE LOADED

Current View

Total :5 Gb by year

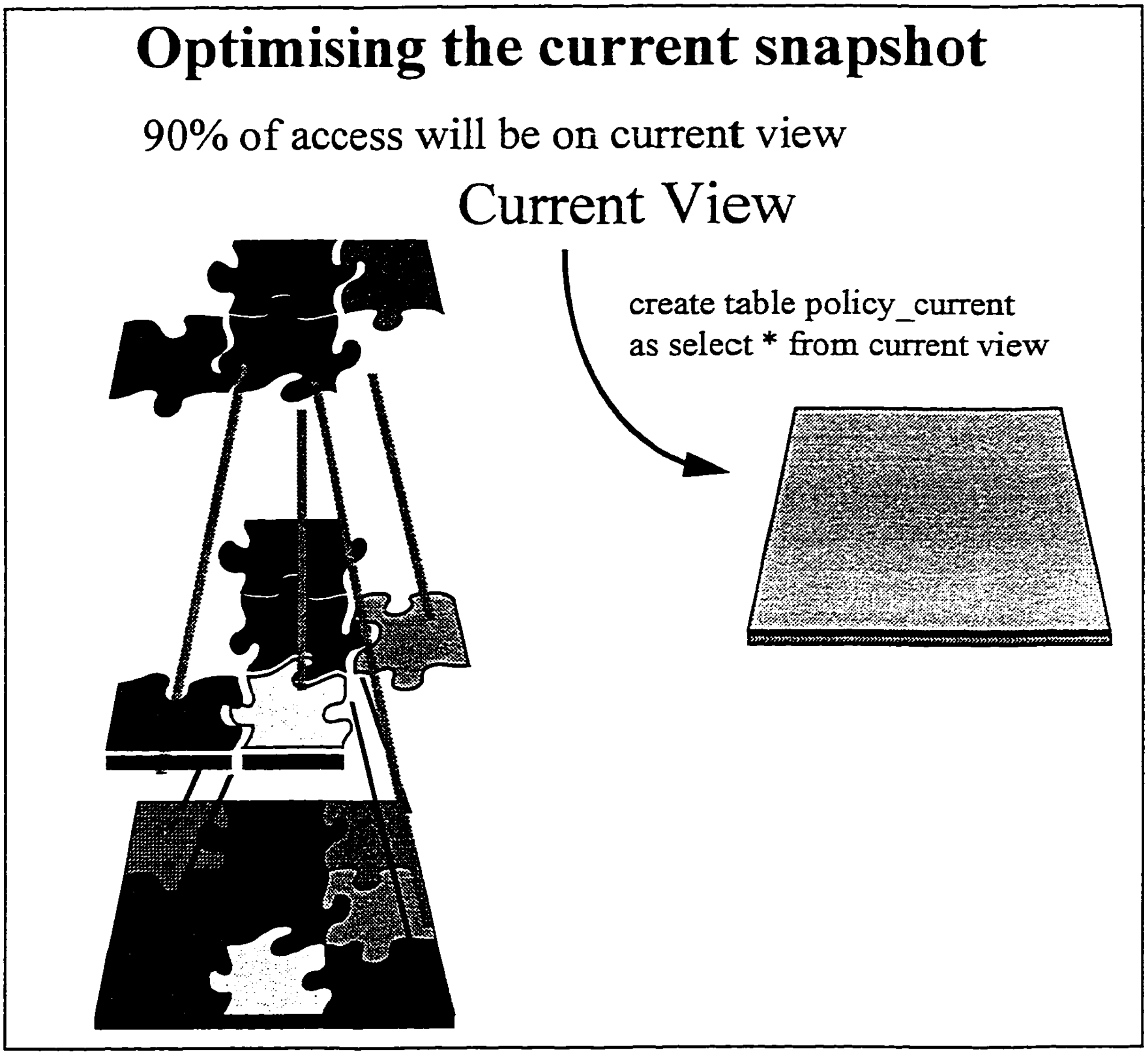
15 Oct.
(350,000)

Previous snapshot

31 Sept
(350,000)

15 Sept
(1,300,000)

Most of the access was done on the most recent snapshot. To optimize access to the current snapshot, we have created a physical table where all the "join" are already done.



Optimising the current snapshot

90% of access will be on current view

Current View

create table policy_current
as select * from current view

2.1.3.2 Modification in Release II Design

Snapshots of data image from all previous periods were produced during each refresh cycle. These snapshots are kept in compressed flat files in the data warehouse. Ad-hoc procedure was put in place to upload the historical data into the marketing data warehouse environment.

2.2 Mainframe Components

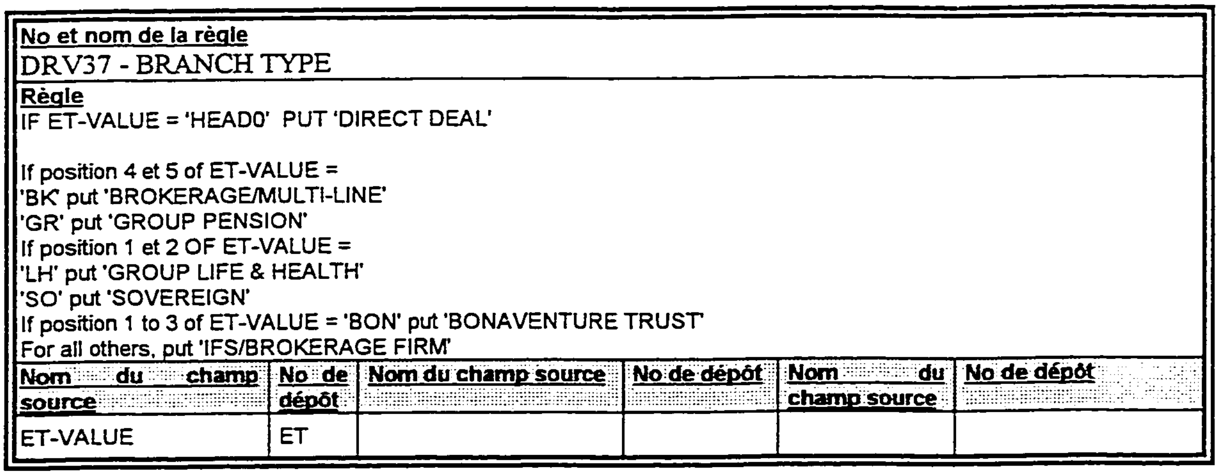
The mainframe programs extracts data from the administrative systems. The various fields can be extracted as is or can be derived from other. Derivation rules has been identified and documented in the Appendix C.

For example:

Deposit name	CASIL.EDITTAB - BRANCH REGION	Deposit number	ET
Extraction program name	EXCAET		
Fields to extract	Fields specific rules	FORMAT	
EXTRACTION DATE		X(10)	
BRANCH CODE	DRV-09	X(05)	
BRANCH NAME	DRV-09	X(30)	
BRANCH REGION	DRV-09	X(10)	
BRANCH TYPE	DRV-37	X(30)	

This is the layout of the Capsil branch extraction file. When a field needs to be computed or transformed, we have indicated a derivation rule that explains code specification.

This is an example of derivation rule documentation:



No et nom de la règle

DRV37 - BRANCH TYPE

Règle

IF ET-VALUE = 'HEAD0' PUT 'DIRECT DEAL'

If position 4 et 5 of ET-VALUE =
'BK' put 'BROKERAGE/MULTI-LINE'
'GR' put 'GROUP PENSION'
If position 1 et 2 OF ET-VALUE =
'LH' put 'GROUP LIFE & HEALTH'
'SO' put 'SOVEREIGN'
If position 1 to 3 of ET-VALUE = 'BON' put 'BONAVENTURE TRUST'
For all others, put 'IFS/BROKERAGE FIRM'

Nom du champ source	No de dépôt	Nom du champ source	No de dépôt	Nom du champ source	No de dépôt
ET-VALUE	ET				

The Capsil and Ingenium jobs are independent jobs. There is no trigger mechanism to assure that jobs run the same day. Frequently extracted dates are not the same. This does not have an effect on the Data warehouse integrity. The two sets of files must be produced before the AIX starts the processing.

All extracted files produces by those jobs are GDG based. We kept 5 generations to assure that we can reload previous files in case of errors.

2.2.1 Capsil Extracts

Six mainframe jobs produce 8 files for the marketing data warehouse. These tables show each job: job name, purpose, extract file, and the parameters for the extraction.

Job Name: PCAPP625	**Job Title:** AGENT DATA EXTRACTION (WAREHOUSE)	**System:** CAPS-I-L
Job Purpose: To extract all active agents and those agents that become inactive within the month range specified as a parameter of extraction.		
FILE: RPRT.CAPSIL.PERI.EXCAAG GDG(5)		
Control card description Section:	N/A □	

Dataset/PDS(member) NAME: CAPSIL.DATA(CAPP625C)			
Line 1			
Field description	**Columns**		**Format/Value:**
	From	**To**	
Program-ID	1	8	character / MS625ETP
Company-literal	10	17	character / COMPANY=
Company-code	18	19	character / SL
Range Date	21	23	Numeric / ### Note: ### represents the number of months to be considered in the extraction range for the inactive agents.

Job Name: PCAPP626	**Job Title:** BRANCH DATA EXTRACTION (WAREHOUSE)	**System:** CAPS-I-L
Job Purpose: To extract the branch name from the Edit Table.		
File : RPRT.CAPSIL.PERI.EXCABR GDG(5)		
Control card description Section:	N/A □	

Dataset/PDS(member) NAME: CAPSIL.DATA(CAPP626C)			
Line 1			
Field description	**Columns**		**Format/Value:**
	From	**To**	
Program-ID	1	8	character / MS626ETP
Company-literal	10	17	character / COMPANY=
Company-code	18	19	character / SL

Job Name: PCAPP627	**Job Title:** : NAME AND ADDRESS EXTRACTION (WAREHOUSE)	**System:** CAPS-I-L
Job Purpose: To extract the NAME and ADDRESS from NAAD file.		

1) RPRT.CAPSIL.PERI.EXCANA GDG(5)	4)
Control card description Section:	N/A □

Dataset/PDS(member) NAME: CAPSIL.DATA(CAPP627C)

Line 1

Field description	**Columns**		**Format/Value:**
	From	**To**	
Program-ID	1	8	character / MS626ETP
Company-literal	10	17	character / COMPANY=
Company-code	18	19	character / SL

Job Name: PCAPP628	**Job Title:** PLAN NAME EXTRACTION (WAREHOUSE)	**System:** CAPS-I-L
Job Purpose: To extract the PLAN NAME from the EDIT table.		

FILE : RPRT.CAPSIL.PERI.EXCAPC GDG(5)	
Control card description Section:	N/A □

Dataset/PDS(member) NAME: CAPSIL.DATA(CAPP628C)

Line 1

Field description	**Columns**		**Format/Value:**
	From	**To**	
Program-ID	1	8	character / MS628ETP
Company-literal	10	17	character / COMPANY=
Company-code	18	19	character / SL

Job Name: PCAPP629	**Job Title:** POLICY,COVERAGE AND COMMISSION EXTRACTION (WAREHOUSE)	**System:** CAPS-I-L

Job Purpose: To extract policy, coverage and commission data for all active agents and the inactive agents with termination date within the rage of months given as a parameter of extraction.

FILE:			4)
1) RPRT.CAPSIL.PERI.EXCAPO	(POLICY)	GDG(5)	
2) RPRT.CAPSIL.PERI.EXCACO	(COVERAGE)	GDG(5)	5)
3) RPRT.CAPSIL.PERI.EXCACOC	(COMMISSION)	GDG(5)	6)

Control card description Section: N/A □

Dataset/PDS(member) NAME: CAPSIL.DATA(CPP629C)

Line 1

Field description	**Columns**		**Format/Value:**
	From	**To**	
Program-ID	1	8	character / MS629ETP
Company-literal	10	17	character / COMPANY=
Company-code	18	19	character / SL
Range Date	21	23	Numeric / ### Note: ### represents the number of months to be considered in extraction range for the inactive agents.

Job Name: PCAPP630	**Job Title:** PROVINCE NAME EXTRACTION (WAREHOUSE)	**System:** CAPS-I-L

Job Purpose: To extract the PROVINCE NAME from the Edit Table.

1) RPRT.CAPSIL.PERI.EXCAPR	GDG(5)	4)
2)		5)
3)		6)

Control card description Section: N/A □

Dataset/PDS(member) NAME: CAPSIL.DATA(CAPP630C)

Line 1

Field description	**Columns**		**Format/Value:**
	From	**To**	
Program-ID	1	8	character / MS630ETP
Company-literal	10	17	character / COMPANY=
Company-code	18	19	character / SL

2.2.2 Ingenium Extracts

For Ingenium, there is only one Job that produces 6 extracted files.

Job Name : PIAMM982	**Job Title:** Produce extract for Data Warehouse	**System:** INGENIUM
Job Purpose: Extract INGENIUM information and create extract files for the Data Warehouse system.		
FILES: 1) INGENIUM.EXT.MLY.CS982A 2) INGENIUM.EXT.MLY.CS982B 3) INGENIUM.EXT.MLY.CS982C 4) INGENIUM.EXT.MLY.CS982D 5) INGENIUM.EXT.MLY.CS982E	6) INGENIUM.EXT-MLY.CS982F 7) 8) 9)	
Control card description Section:	**N/A** □	
Dataset/PDS(member) NAME: ____INGENIUM.DATA (IAM982C)________________________________		

Line 1			
Field description	**Columns**		**Format/Value:**
	From	**To**	
PROGRAM NAME	1	8	CSBM9082
COMPANY	10	19	COMPANY=SL

2.3 Unix Components

2.3.1 Unix Environments

- **the data warehouse Unix components reside on server SXMQ1005**
- **two environments has been implemented:**
  - tdwh environment is for the development, unit and integrated tests
  - pdwh environment is for the production
- **directory structure of the development environment**

Directory	Contents
/tdwh/src	C programs
/tdwh/bin	shell scripts, SAS programs, run time pgm
/tdwh/sql	Sql scripts
/tdwh/ctl	Control file for Oracle SQL Loader
/tdwh/log	Log and trace
/tdwh/saslib	SAS data set(extracted files from previous cycle)
/data/transfer	Files from MVS
/data/update	Result of SAS compare pgm(input of SQL Loader)
/hist	Historical Data

- **directory structure of the production environment**

Directory	Owner	Content
• /Pdwh	pdwh	
• /Dbf • /db1 • /db2 • /db3 • /db4 • /db5 • /db6	oracle	Oracle data files
• /hist	pdwh	Historical snapshots
• /pfl	oracle	Support files for Instance startup
• /dmp • /usr • /bkg • /cor • /aud	oracle	dump depository for user, background and server process
• /exp	oracle	Export depository
• /arc	oracle	Archive log depository
• /dba	oracle	Dba scripts

• /adm • /view • /sqltable • /trigger • /histable • /stview • /bin		
• /bin	pdwh	Ksh scripts and SAS pgm for refresh cycle
• /sql	pdwh	Sql scripts for refresh cycle
• /ctl	pdwh	Control files for ORACLE sql loader
• /log	pdwh	Process log files
• /data • /update • /transfer	pdwh	• depository of results from SAS pgm(input to sql loader) • depository of files from MVS
• /saslib	Pdwh	SAS data set(extracted files from previous cycles)
• /rwtc • /metadata • /rwtcdlmi • /rwtcdlug	Pdwh	SAS setup scripts
• /sasuser	Pdwh	Directory for SAS users
• /saswork	Pdwh	Temporary data directory for SAS users

- **choice of context environment**

The run time context is determined by 2 Unix environment variables set in the profile script.

```
# Korn shell login script
# SAS and Oracle environment variables
export PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:/home/
sas/sas612:/u/ora732/app/oracle/product/7.3.2/bin:.
export ORACLE_HOME=/home/ora732/app/oracle/product/7.3.2
export ORACLE_SID=pdwh
export ORAENV_ASK=no
export SASORA=V7
export EPC_DISABLED=TRUE
export TDWH=/pdwh
```

The ORACLE_SID variable is used to identify the Oracle instance.

The TDWH variable is used is in script to link to specific directories (note that TDWH is not a good name for this variable, it should be changed to DWHDIR)

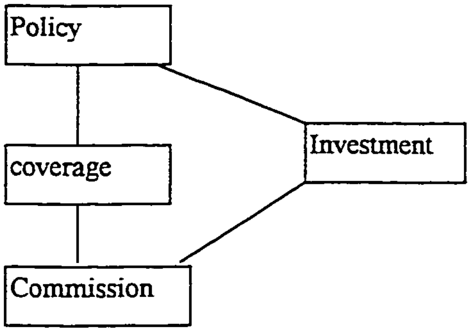
- **UNIX software related to the data warehouse RS/6000 server**

AIX	4.1.2	AIX operating system
Oracle server	7.3.2.1	Oracle DBMS
SAS/Base	6.12	SAS base software
SAS/Share	6.12	Shared access to SAS dataset
SAS/Connect	6.12	Communication middleware (SAS to SAS)
SAS/Access to Oracle	6.12	Access to Oracle databases

2.3.2 Physical Database Model

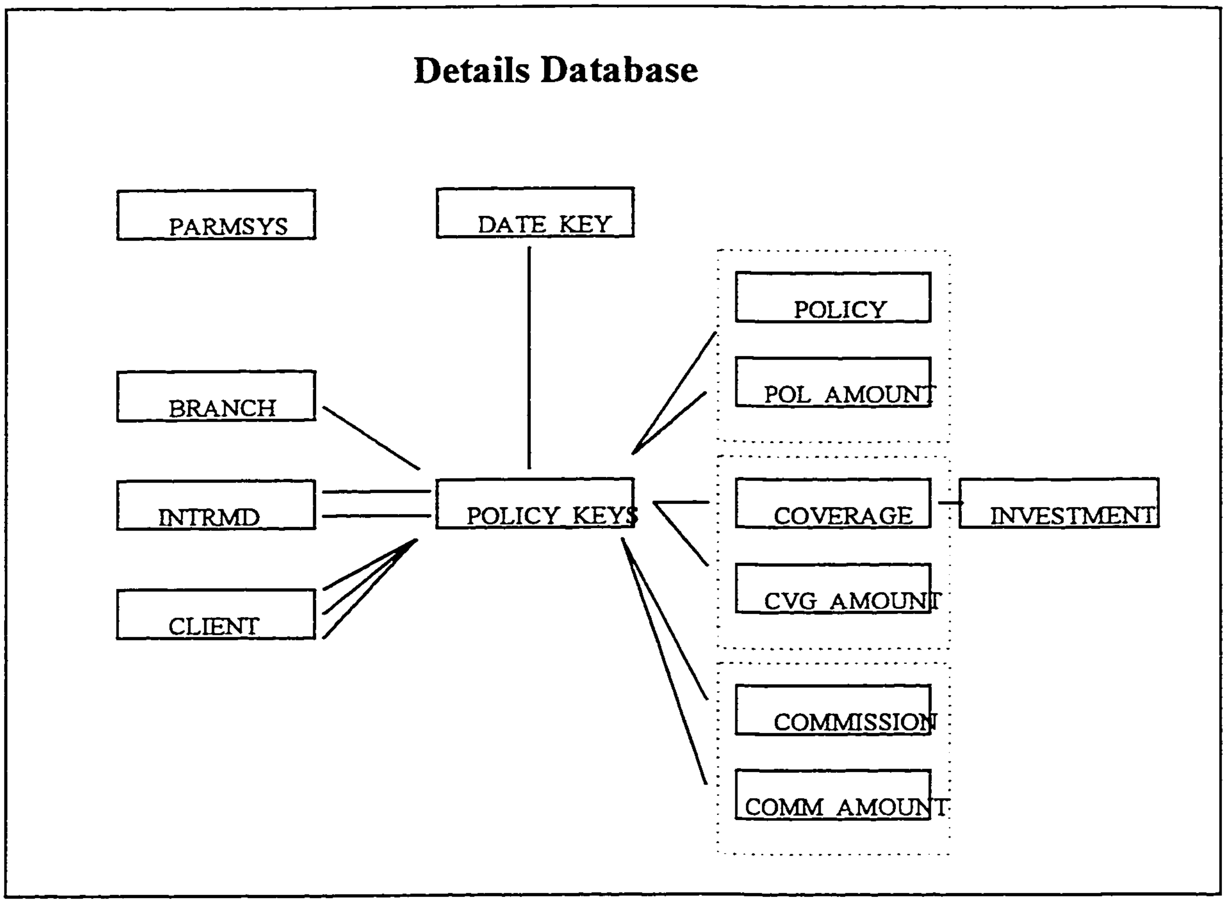
2.3.2.1 Policy Basic Tables

Four tables can be used to store policy information:



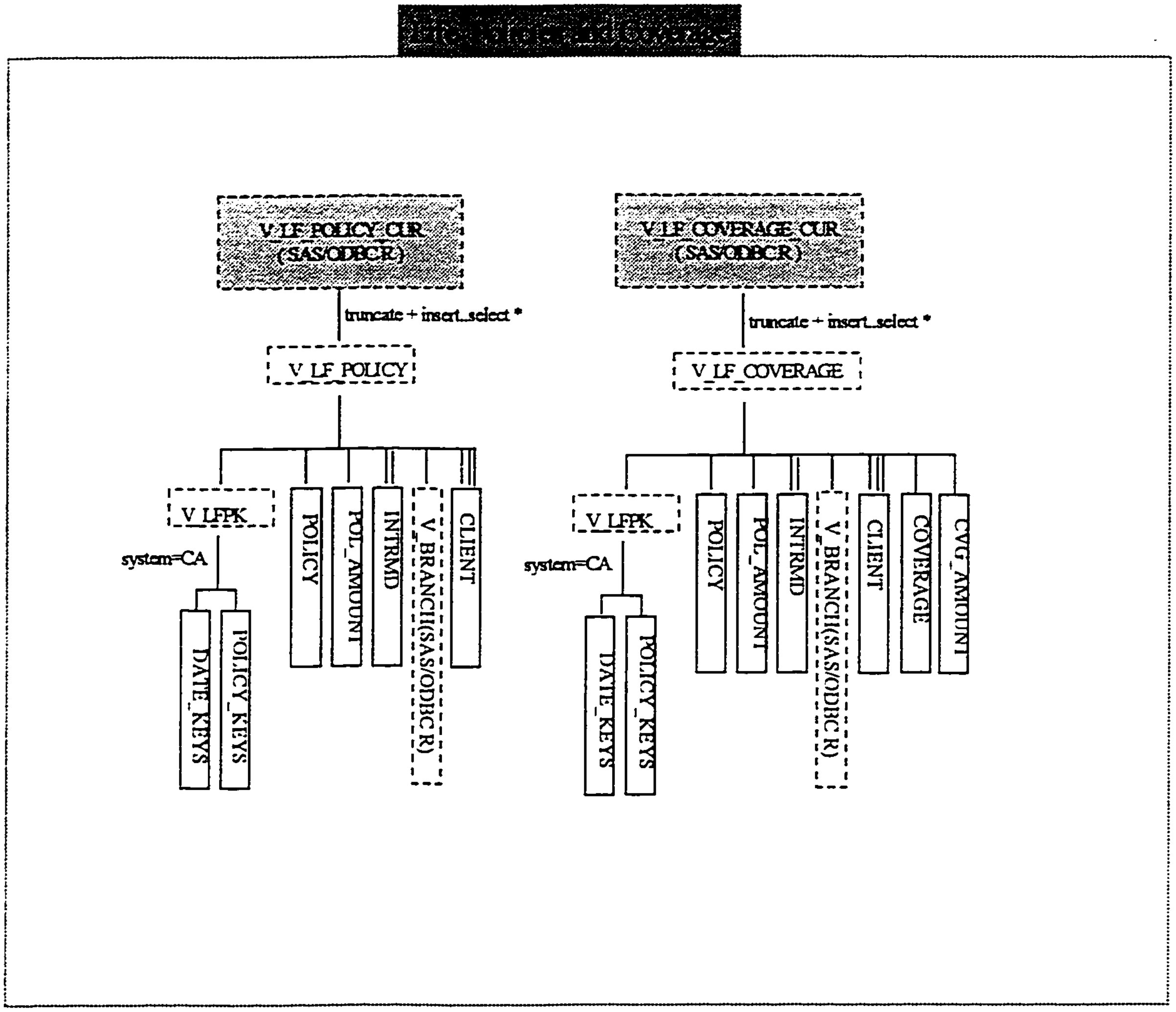
To minimize disk space in release, each table was cut in two parts. The first part contains the static information (less change,) and the second part contains the financial information that changes more frequently.

This diagram describes all tables related to policy information.

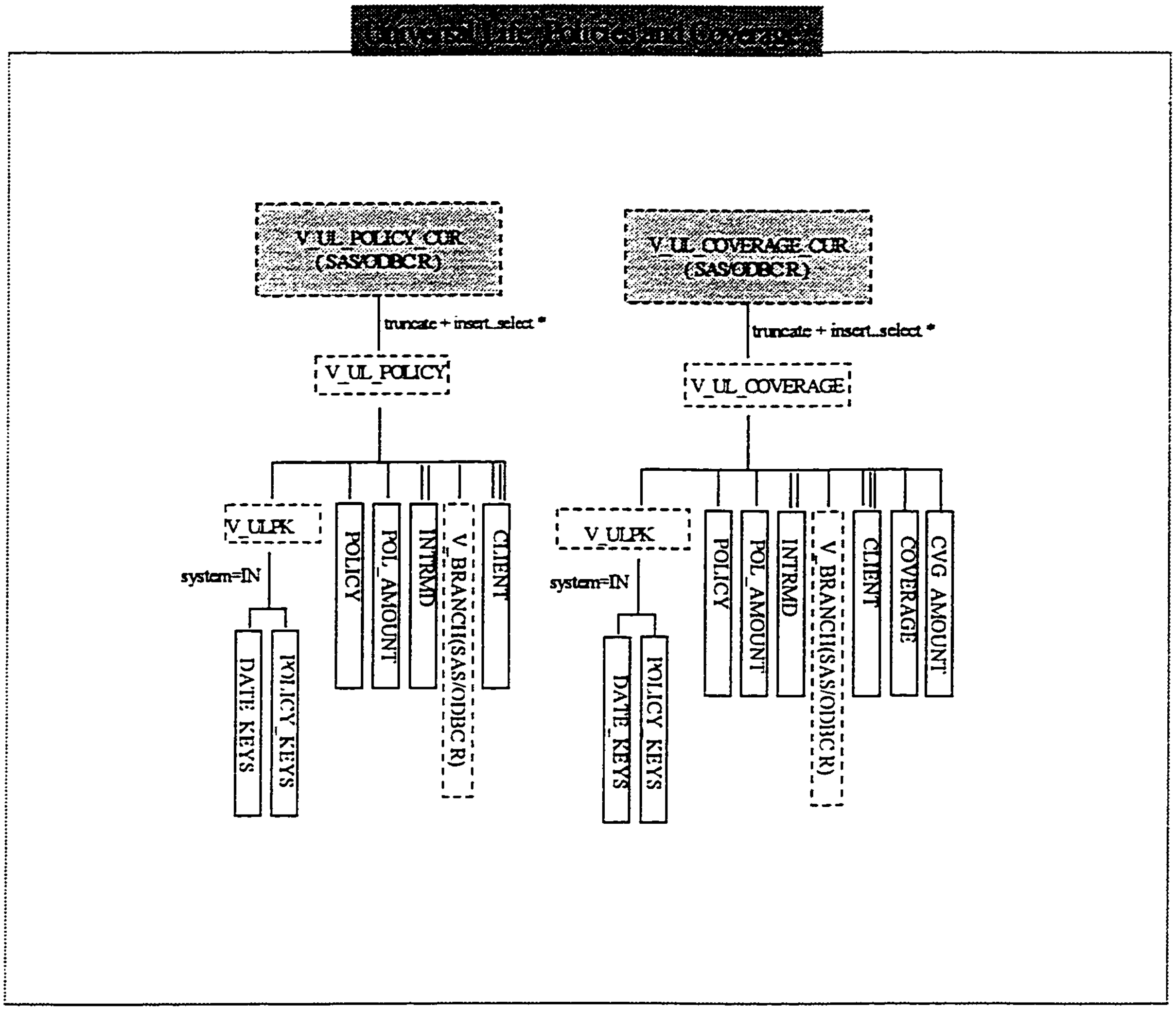


As explained in a previous section, the marketing data warehouse contains historical information on previous policy snapshots and also data on the current period. To provide better response time on the current snapshot, a series of Oracle views as been defined to select the current snapshot. The SQL code of those views contains all the "joins" between tables to provide a completed image of all policies (including intermediary, branch, client description). Results of those big "JOIN" are stored physically in the "current" table(v_policy_cur table).

This diagram describes all views and "stored views" links for Life products policies and coverage.



This diagram describes all views and “stored views” links for Universal Life products policies and coverage.



2.3.2.2 Intermediary and branch statistics

Three kind of statistics are kept by the intermediary:

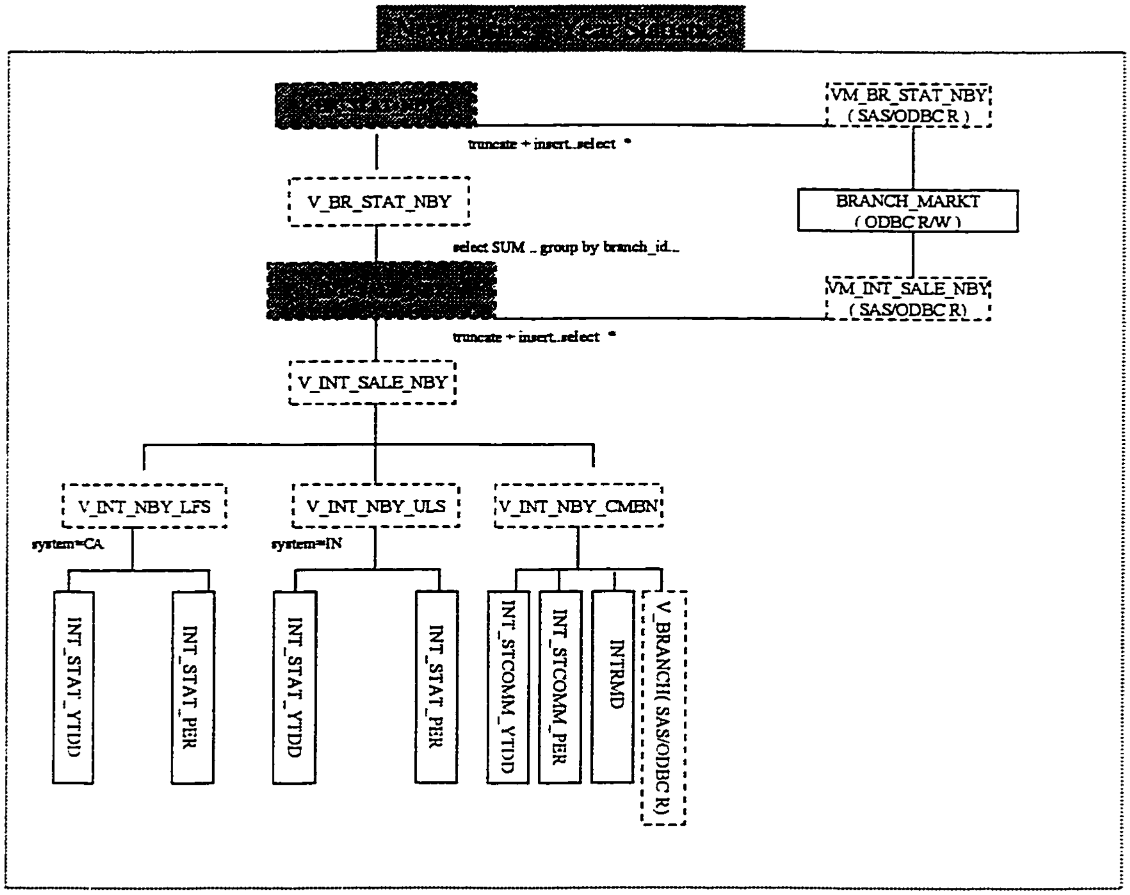
- Statistics on commission and bonus (COMM)
- Statistics on policies (#-$ submit, accept, reject...) (STAT)
- Statistics on persistency (INTPERS)

Statistics are also kept by period (twice a month) and year to date (fiscal and business year)

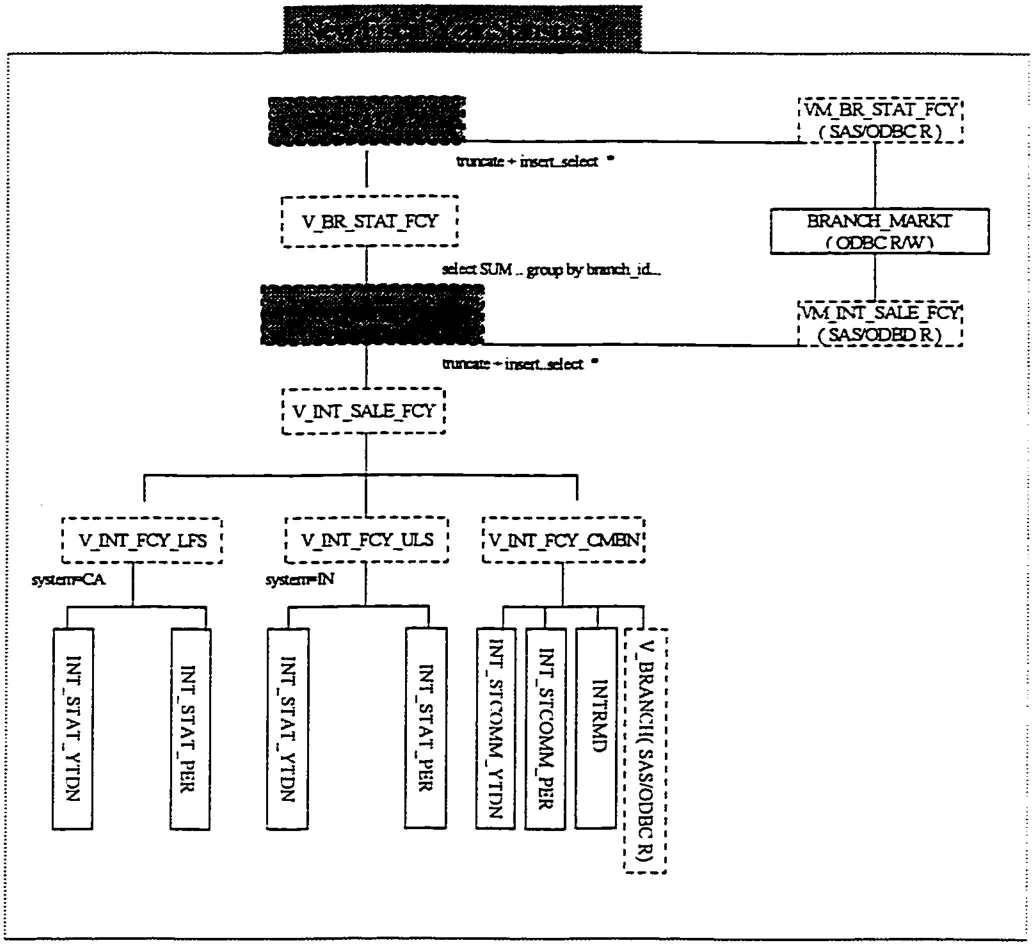
This schema describes basics statistical tables

To provide better performance and better response time during access, a set of tables has been defined to store the physical view to all the joins between the tables. Those joins permit us to have on the same row all period and year to date field and also all the intermediary and branch descriptions. SQL procedure has been implemented to summarize intermediary statistics to produce branch statistics.

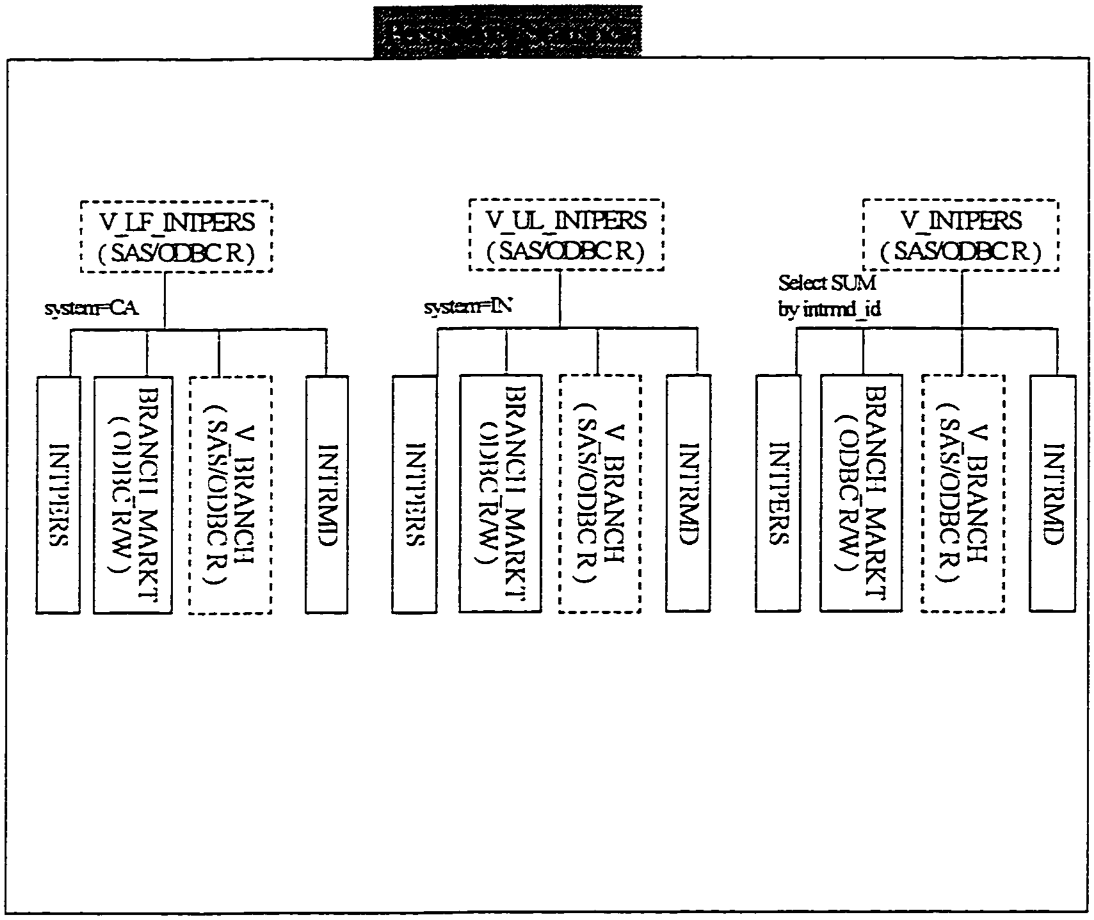
This diagram shows links between components to produce final tables related to new business year that users can access.



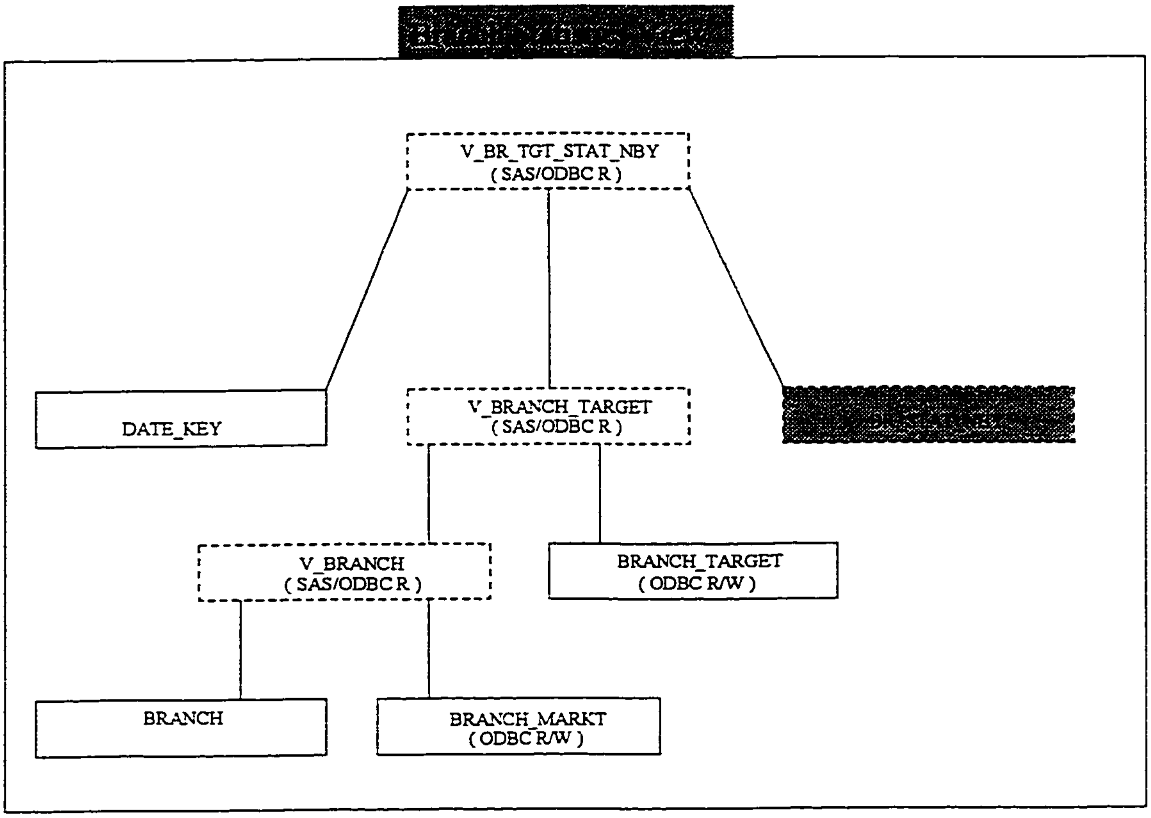
This diagram shows links between components to produce final tables related to fiscal year that users can access.



For persistency statistics, Oracle views has been defined to do all the joins to provide intermediary and branch descriptions.



Views are also provide to facilitate the comparison of targeted and actual sales by each branch.



2.3.2.3 DBA Scripts

The DBA scripts are in the /pdwh/dba/adm directories on the DW UNIX server. This directory contains scripts to create the Oracle instance (table space, users...)

Subdirectories are:

sqltable: Scripts to generate tables, primary keys and indexes

stview : Scripts to generate stored views

histable: Scripts to generate historical tables

view: Scripts to generated views, Stored views tables primary keys and indexes on stored views

trigger: Scripts to create triggers code used in the SQL load process

2.3.3 Database Refreshing Components

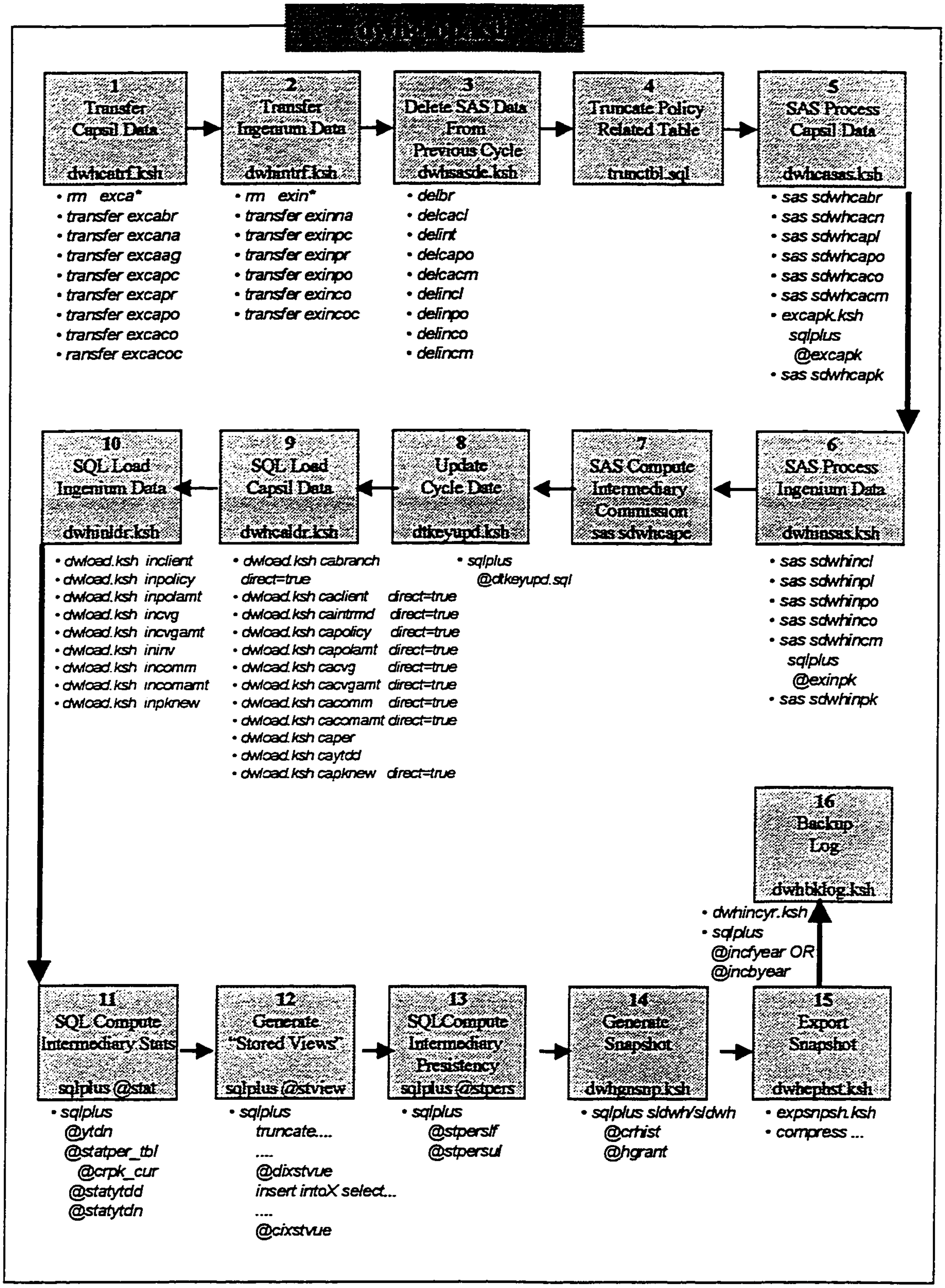
The refresh cycle can be started on Unix by a single script. The “dwhglob.ksh” script is located in the "pdwh/bin" directory. This script contains 16 steps:

1. Transfer Capsil Data
2. Transfer Ingenium Data
3. Delete SAS Data From Previous Cycle
4. Truncate Policy Related Table
5. SAS Process Capsil Data
6. SAS Process Ingenium Data
7. SAS Compute Intermediary Commission
8. Update Cycle Date
9. SQL Load Capsil Data
10. SQL Load Ingenium Data
11. SQL Compute Intermediary Statistics
12. Generate "Stored Views"
13. SQL Compute Intermediary Persistency Statistics
14. Generate Snapshot
15. Export Snapshot
16. Backup Log and Update the Parmsys table

This script can be restarted by specifying a restart parameter (step number). In major cases, each step starts another script that contains few sub steps.

The entry into the parmsys table needs to be adjusted annually. The column: NOV_YTD_BEGIN_DT needs to be manually incremented one year after the mid-November refresh cycle is completed, while the column DEC_YTD_BEGIN_DT needs to be incremented after the mid-December refresh cycle.

This diagram describes a detailed view of all steps related to the process.



2.4 Interactions Between Mainframe and UNIX Components

2.4.1 Putting Branch Marketing Information Into Mainframe

2.4.1.1 File Transfer Process and Scheduling

A UNIX cron job is scheduled to run at mid night of the 10th and 25th, every month of the year to transfer the information from the branch_markt table into CAPSIL. Details of this cron table could be found in the file "/pdwh/cron/crontable".

The actual transfer is accomplished by the following scripts

- **/pdwh/bin/Dwhbrmkt.ksh**

  (launch the transfer process)

- **Sqlplus /pdwh/sql/exbrmkt**

  (Extract the branch_markt info into an ASCII file "/pdwh/data/brmkt/exbrmkt.lst")

- **/pdwh/bin/Dwhbmput.ksh**

  (Transfer "/pdwh/data/brmkt/exbrmkt.lst" from UNIX to "CAPSIL.DWTAB" on CAPSIL)

- **/pdwh/bin/Brmkt_transfer**

  (file transfer engineer tailored to branch marketing data)

2.4.1.2 Format of the ASCII File

Column Order	Column Name	Space	Example Content
1	Current Year	11	1998-12-01
2	BR_ID	6	VANCO
3	BR_NM	31	VANCOUVER
4	BR_REG	11	WESTERN
5	BR_TYP	31	IFS
6	BR_SORT_ORDER_EW	6	31
7	TARGET_YEAR	5	1998
8	BR_IFS_TARGET_LIFE	15	6000
9	BR_IFS_TARGET_MONEY_GRP_TTL	15	200000
10	BR_IFS_TARGET_MF	15	100000
11	BR_IFS_REP_HIRE	7	4
12	BR_BRK_TARGET_LIFE	15	10000
13	BR_BRK_TARGET_MONEY_GRP_TTL	15	50000
14	BR_BRK_TARGET_MF	15	20000
15	BR_BRK_REP_HIRE	7	5

2.5 SAS Administrative Components

This tool is used by the Data warehouse administrator to :

- create the metadata
- maintain tables and fields descriptions
- define remote database definition
- define subjects and structure of subjects
- create icons for tools or applications available from the user tool bar
- manage security (user, group and privilege)

2.6 User's components

This section describes all softwares used by DW users

Software	Version	Comments
Windows 95	SP 1	Windows operating system
MS Office	4.3C	MS Office suite 16 bits
Netware client	?	Netware 32 bit client
Navigator	1.02	Data Warehouse Navigator application
SAS/Base	6.12	SAS base software
SAS/Connect	6.12	Communication middleware (SAS to SAS)
SAS Client/server ECO	6.12	Data Warehouse modules
SAS/Access PC file format	6.12	Export/import of SAS dataset to other file formats (ie. Excel)
SAS/AF	6.12	Tools to develop GUI application
SAS/Assist	6.12	User interface and SQL generator
SAS/Calc	6.12	Electronic spreadsheet
SAS/EIS	6.12	Object oriented development tool
SAS/FSP	6.12	SAS dataset full screen access and editor
SAS/Graph	6.12	Graphics generator
SAS/Tutor	6.12	On-line SAS tutorial

SAS /insight was also used in a trial mode.

3 Appendix

3.1 Database Volume

Table Name	Number	avg row	Table	Table	Initial	avg idx	Table	Index	Initial
	of rows	length	space name	size	extent	length	Space name	size	extent
BRANCH	200	66	DWHINTRM	17k	500k	23	DWHINTRMIX	6k	100k
BRANCH_MARKT	200	100	DWHMKDEPT	26k	500k	23	DWHMKDEPT	6k	100k
BRANCH_TARGET	800	150	DWHMKDEPT	153k	500k	23	DWHMKDEPT	24k	500k
DATE_KEY	100	17	DWHINTRM	3k	100k	18	DWHINTRMIX	3k	100k
INTRMD	21000	101	DWHINTRM	2 710k	2 980k	24	DWHINTRMIX	642k	710k
CLIENT	600000	40	DWHCLIEN	30 001k	33 000k	34	DWHCLIENIX	26 087k	28 700k
POLICY	250000	137	DWHPOLIC	43 479k	47 830k	31	DWHPOLICIX	9 901k	10 890k
POL_AMOUNT	250000	48	DWHPOLAM	15 152k	16 670k	31	DWHPOLAMIX	9 901k	10 890k
COVERAGE	625000	131	DWHCOVER	104 167k	114 580k	34	DWHCOVERIX	27 174k	29 890k
CVG_AMOUNT	625000	46	DWHCVGAM	36 232k	39 860k	34	DWHCVGAMIX	27 174k	29 890k
COMMISSION	625000	70	DWHCOMMI	55 556k	61 110k	37	DWHCOMMIIX	29 412k	32 360k
COMM_AMOUNT	625000	44	DWHCOMMA	34 723k	38 200k	37	DWHCOMMAIX	29 412k	32 360k
INVESTMENT	40000	31	DWHINTRM	1 554k	1 710k	34	DWHINTRMIX	1 740k	1 920k
POLICY_KEYS	250000	165	DWHPOLKEY	52 632k	57 900k	31	DWHPLKEYIX	9 901k	10 890k
PARMSYS	1	19	DWHINTRM	1k	16k	0	none	0k	
PK_CUR	250000	165	DWHINTRM	52 632k	57 900k	0	none	0k	
INTPERS	108000	41	DWHSTINT	5 539k	6 100k	27	DWHSTINTIX	3 725k	4 100k
INT_STCOM_PER	1512000	43	DWHSTINT	81 730k	89 900k	24	DWHSTINTIX	46 168k	50 790k
INT_STCOM_YTDD	1512000	44	DWHSTINT	84 001k	92 400k	24	DWHSTINTIX	46 168k	50 790k
INT_STCOM_YDTN	1512000	45	DWHSTINT	85 184k	93 700k	24	DWHSTINTIX	46 168k	50 790k
INT_STAT_PER	360000	22	DWHSTINT	9 932k	10 930k	27	DWHSTINTIX	12 414k	13 660k
INT_STAT_YTDD	360000	25	DWHSTINT	11 251k	12 380k	27	DWHSTINTIX	12 414k	13 660k
INT_STAT_YTDN	360000	25	DWHSTINT	11 251k	12 380k	27	DWHSTINTIX	12 414k	13 660k
V_LF_POLICY_CUR	250000	288	DWHSTVIEW1	90 910k	100 000k	1	none	0k	
V_LF_COVERAGE_CUR	625000	404	DWHSTVIEW1	357 143k	392 860k	1	none	0k	
V_UL_POLICY_CUR	30000	331	DWHSTVIEW1	13 334k	14 670k	1	none	0k	
V_UL_INVESTMENT_CUR	30000	389	DWHSTVIEW1	15 001k	16 500k	1	none	0k	
V_UL_COVERAGE_CUR	30000	438	DWHSTVIEW1	17 143k	18 860k	1	none	0k	
H_LF_POLICY	250000	287	DWHSTHIST	90 910k	100 000k	1	none	0k	
H_LF_COVERAGE	625000	403	DWHSTHIST	357 143k	392 860k	1	none	0k	
H_UL_POLICY	30000	331	DWHSTHIST	13 334k	14 670k	1	none	0k	
H_UL_INVESTMENT	30000	389	DWHSTHIST	15 001k	16 500k	1	none	0k	
H_UL_COVERAGE	30000	437	DWHSTHIST	17 143k	18 860k	1	none	0k	
INT_SALE_NBY	1512000	166	DWHSTVIEW2	318 316k	350 150k	0	none	0k	
INT_SALE_FCY	1512000	167	DWHSTVIEW2	318 316k	350 150k	0	none	0k	
BR_STAT_FCY	14400	157	DWHSTVIEW2	2 881k	3 170k	0	none	0k	
BR_STAT_NBY	14400	161	DWHSTVIEW2	3 032k	3 340k	0	none	0k	

Table Name	Number	avg row	Table	Table	Initial	avrg idx	Table	Index	Initial
Index	of rows	length	space name	size	extent	length	Space name	size	extent
V_LF_Policy_Cur(pol_id)00	250 000					15	DWHPOLICIX	4 785k	5 260k
V_LF_Policy_Cur(pol_plan_cd)02	250 000					15	DWHPOLICIX	4 785k	5 260k
V_LF_Policy_Cur(pol_sta)03	250 000					19	DWHPOLICIX	6 061k	6 670k
V_LF_Policy_Cur(pol_sub_sta)04	250 000					29	DWHPOLICIX	9 260k	10 190k
V_LF_Policy_Cur(iss_prov_nm)05	250 000					49	DWHPOLICIX	15 626k	17 190k
V_LF_Policy_Cur(app_recv_dt)06	250 000					17	DWHPOLICIX	5 406k	5 950k
V_LF_Policy_Cur(app_cmplt_dt)07	250 000					17	DWHPOLICIX	5 406k	5 950k
V_LF_Policy_Cur(infc_dt)08	250 000					17	DWHPOLICIX	5 406k	5 950k
V_LF_Policy_Cur(iss_eff_dt)09	250 000					17	DWHPOLICIX	5 406k	5 950k
V_LF_Policy_Cur(pol_ceas_dt)10	250 000					17	DWHPOLICIX	5 406k	5 950k
V_LF_Policy_Cur(pd_to_dt)11	250 000					17	DWHPOLICIX	5 406k	5 950k
V_LF_cvg_Cur(cvg_plan_cd)01	625 000					15	DWHCOVERIX	11 962k	13 160k
V_LF_cvg_Cur(cvg_sub_stat)02	625 000					19	DWHCOVERIX	15 152k	16 670k
V_LF_cvg_Cur(cvg_app_recv_dt)03	625 000					17	DWHCOVERIX	13 514k	14 870k
V_LF_cvg_Cur(cvg_app_cmplt_dt)04	625 000					17	DWHCOVERIX	13 514k	14 870k
V_LF_cvg_Cur(cvg_inv_xbit_dt)05	625 000					17	DWHCOVERIX	13 514k	14 870k
V_LF_cvg_Cur(cvg_infc_dt)06	625 000					17	DWHCOVERIX	13 514k	14 870k
V_LF_cvg_Cur(cvg_ceas_dt)07	625 000					17	DWHCOVERIX	13 514k	14 870k
									0k
Polkey(writ_intrmd_id)01	250 000					15	DWHPLKEYIX	4 785k	5 260k
Polkey(serv_intrmd_id)02	250 000					15	DWHPLKEYIX	4 785k	5 260k
Polkey(br_id)03	250 000					14	DWHPLKEYIX	4 465k	4 910k
polkey(pol_id)04	250 000					19	DWHPLKEYIX	6 061k	6 670k
									0k
intrmd(intrmd_nm)02	21 000					60	DWHINTRMIX	1 618k	1 780k
intrmd(intrmd_typ)03	21 000					21	DWHINTRMIX	564k	620k
Intrmd(intrmd_ble)04	21 000					29	DWHINTRMIX	778k	860k
Intrmd(intrmd_sta)05	21 000					21	DWHINTRMIX	564k	620k
Intrmd(intrmd_hire_dt)06	21 000					17	DWHINTRMIX	455k	500k
Intrmd(intrmd_ceas_dt)07	21 000					17	DWHINTRMIX	455k	500k
Intrmd(intrmd_ceas_rea)08	21 000					19	DWHINTRMIX	510k	560k

2 347 533k	2 583 736k		+	*543 529k*	598 740k
=	2 971 296k		+		688 551k
=	2 902 Mb		+		673 Mb
		TOTAL =	3 575 Mb		

TABLESPACES DISTRIBUTION

TABLE			
Table space Name	Total	Total * TS Free factor	Disk Id
DWHINTRM	63 206k	75 800k	DB1
DWHCLIEN	33 000k	39 600k	DB6
DWHPOLIC	47 830k	57 400k	DB3
DWHPOLAM	16 670k	20 000k	DB1
DWHCOVER	114 580k	137 500k	DB6
DWHCVGAM	39 860k	47 800k	DB1
DWHCOMMI	61 110k	73 300k	DB4
DWHCOMMA	38 200k	45 800k	DB1
DWHPLKEY	57 900k	69 500k	DB4
DWHSTINT	317 790k	381 300k	DB2
DWHSTVIEW1	542 890k	651 500k	DB1
DWHSTVIEW2	706 810k	848 200k	DB5
DWHSTHIST	542 890k	651 500k	DB6
DWHMKDEPT	1 000k	1 200k	DB3
2 LOG	40 000k	40 000k	DB5
2 LOG	40 000k	40 000k	DB4
RBS	550 Mb	563 200k	DB2
SYSTEM	50 Mb	51 200k	DB1
TEMP	750 Mb	768 000k	DB3
TOTAL	4 046 136k	4 562 800k	
	3 951 Mb	4 456 Mb	

	DISK A	
DB1 =	892 100k	1 024 Mb
DB2 =	944 500k	1 024 Mb
	1 836 600k	
	DISK B	
DB3 =	827 300k	1 024 Mb
DB4 =	900 500k	1 024 Mb
	1 727 800k	
	DISK C	
DB5=	888 200k	1 024 Mb
DB6=	828 600k	1 024 Mb
	1 716 800k	
	DISK D	
f3	0k	o
f4	0k	o
	0k	
	DISK E	
f5	0k	0 Mb
f6	0k	0 Mb
	0k	
Total =	5281200k	6 144 Mb

INDEX			
Disk Id	Table space name	Total	Total * TS free factor
DB4	DWHINTRMIX	8 270k	9 900k
DB4	DWHCLIENIX	28 700k	34 400k
DB4	DWHPOLICIX	91 160k	109 400k
DB4	DWHPOLAMIX	10 890k	13 100k
DB4	DWHCOVERIX	134 070k	160 900k
DB4	DWHCVGAMIX	29 890k	35 900k
DB4	DWHCOMMIIX	32 360k	38 800k
DB4	DWHCOMMAIX	32 360k	38 800k
DB4	DWHPLKEYIX	32 990k	39 600k
DB4	DWHSTINTIX	197 450k	236 900k
DB3	DWHMKDEPT	600k	700k
	TOTAL	598 740k	718 400k
		585 Mb	702 Mb

3.2 Derivation Rules

Extract file description are documented in the s:\dataware\docum\rule\

extrcaps.doc

extring.doc

Derivation rules are documented in the s:\architec\dataware\docum\rule\drvmark.doc

SAS aggregate derivation rule are documented in the s:\architec\dataware\docum\rule\

aginpers.doc

aginprod.doc

agrbr.doc

agrint.doc

aglicvg.doc

aglife.doc

aglipol.doc

agulife.doc

agsomr.doc

agulcvg.doc

agulvo.doc

agulpol.doc

3.3 Steps to Reinitialize the Test Environment From Production Environemnt

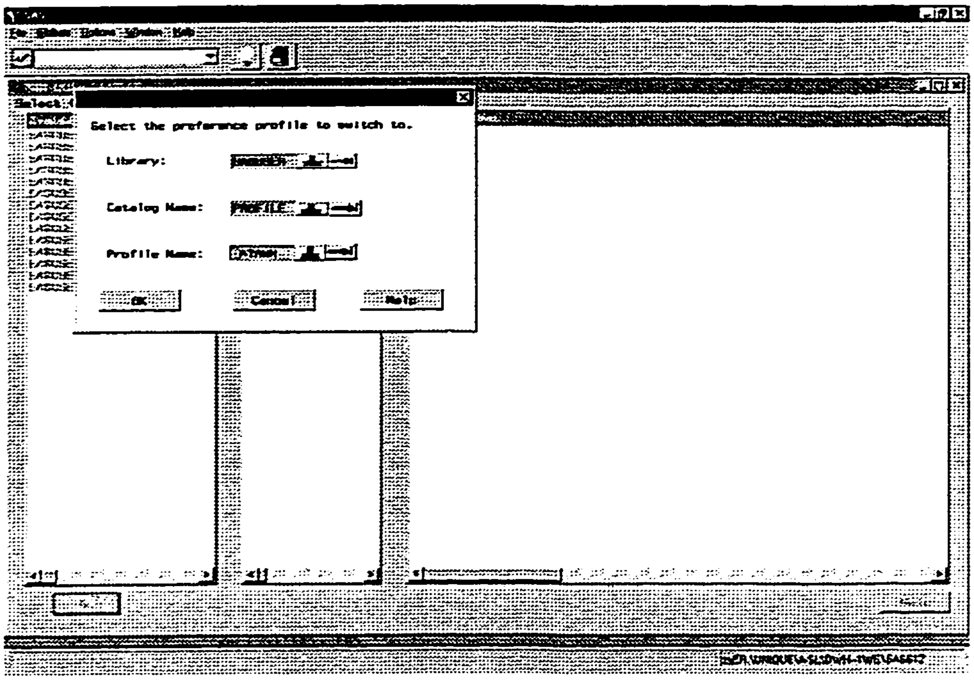
Steps	UNIX	PL/SQL
Take Full Export of the Production Environment /pdwh/exp/expfull.ksh	Oracle(pdwh)	
Copy the Export File From Production Environment into the Test Environment cp /pdwh/exp/pdwh.dmp /tdwh/exp/pdwh.dmp	oracle	
Truncate All Tables /tdwh/dba/adm/emptytbl.sql	Oracle(tdwh)	dba
Import All Permanent Tables /tdwh/dba/adm/impbastb.ksh	Oracle(tdwh)	
Rebuild one Temporary Table /tdwh/sql/crpk_cur.sql	tdwh	tdwh

3.4 SAS Setup

3.4.1 Create Report Definition Using ORACLE Profile

1. Switch to the designated profile

SAS612(menu) → Globals(menu) → Desktop(Menu) → Query and Reporting(icon) → Query Data(icon) →Action(menu) → Switch to New Profile(menu) → Profile Name : <Designated Profile Name>



Select the preference profile to switch to.

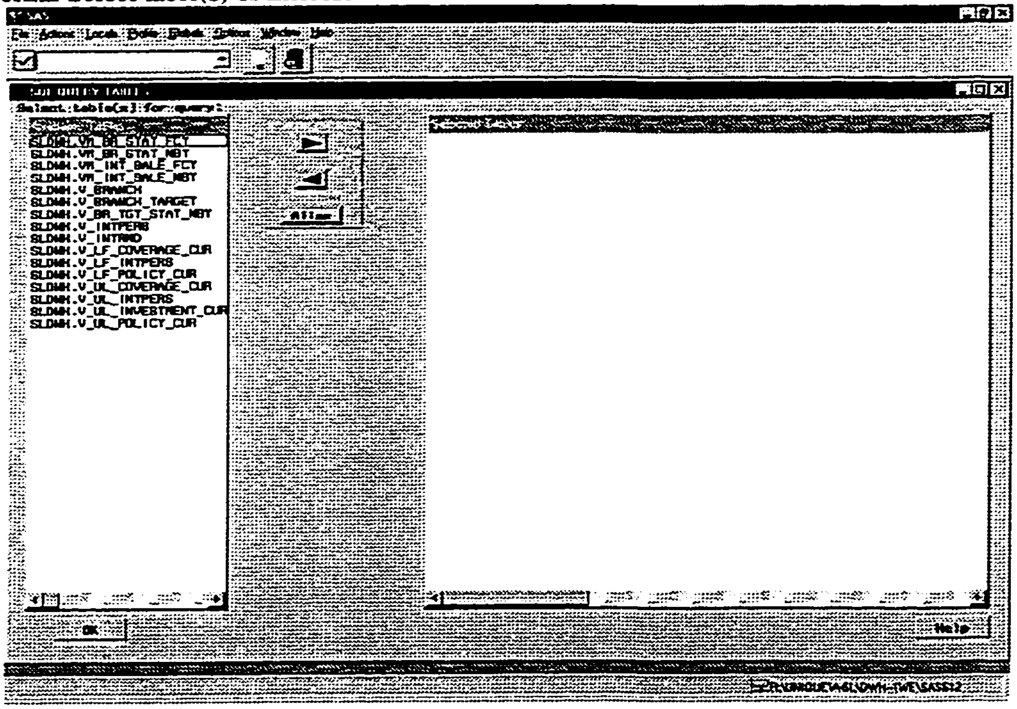
Library: [illegible]

Catalog Name: PROFILE

Profile Name: [illegible]

2. Enter the data warehouse

OK(Profile dialogue box) → Userid?:< UNIX user name> → Password?< UNIX password >
Note: the An ORACLE User name same as the UNIX must exist with ORACLE password set to external Select table(s) of interest

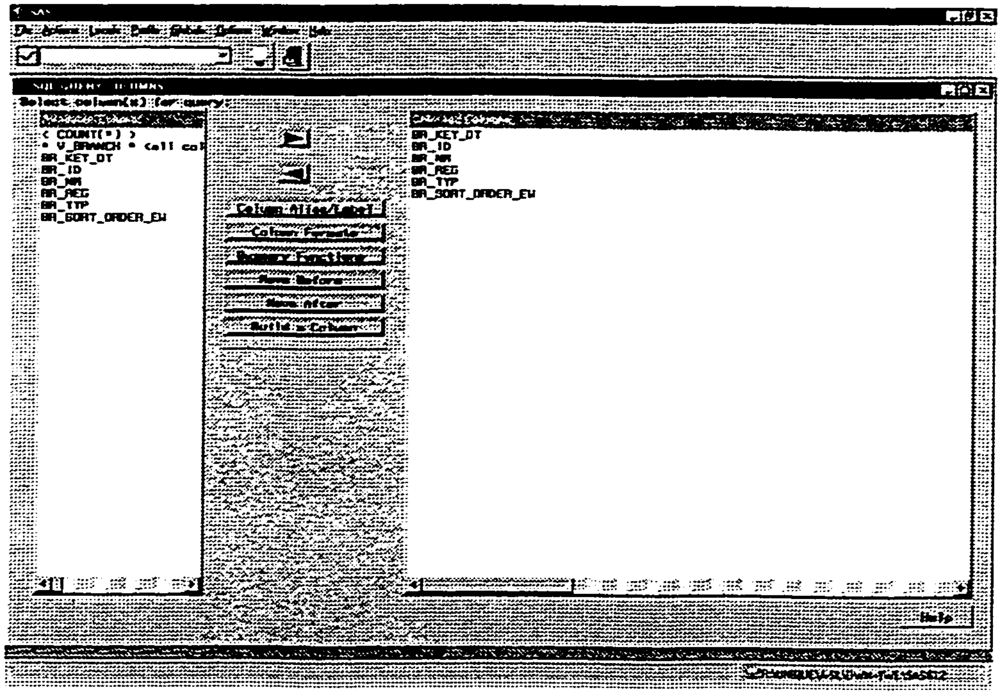


SQL QUERY TABLES

Select table(s) for query:

SLDWH.VR_BR_STAT_FCT
SLDWH.VR_BR_STAT_MBT
SLDWH.VR_INT_SALE_FCT
SLDWH.VR_INT_SALE_MBT
SLDWH.V_BRANCH
SLDWH.V_BRANCH_TARGET
SLDWH.V_BR_TGT_STAT_MBT
SLDWH.V_INTPERS
SLDWH.V_INTRMD
SLDWH.V_LF_COVERAGE_CUR
SLDWH.V_LF_INTPERS
SLDWH.V_LF_POLICY_CUR
SLDWH.V_UL_COVERAGE_CUR
SLDWH.V_UL_INTPERS
SLDWH.V_UL_INVESTMENT_CUR
SLDWH.V_UL_POLICY_CUR

3. Select columns of interest

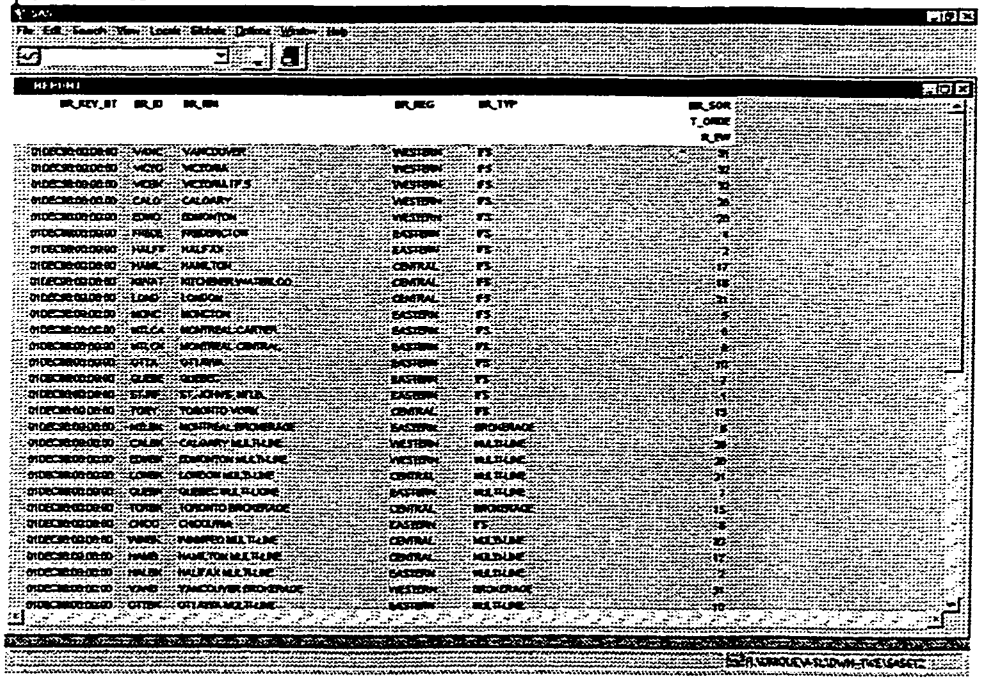


Select column(s) for query:

```
< COUNT(*) >
* V_BRANCH * (all col
BR_KEY_DT
BR_ID
BR_NM
BR_REG
BR_TYP
BR_SORT_ORDER_EW
```

```
BR_KEY_DT
BR_ID
BR_NM
BR_REG
BR_TYP
BR_SORT_ORDER_EW
```

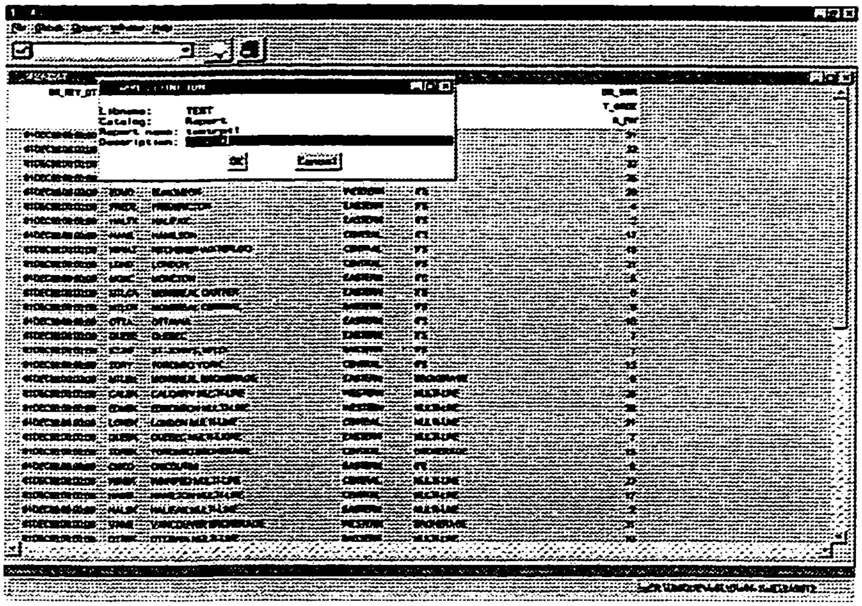
4. Build A Report
Action(menu) → Run Query → Design a Report with Report Viewer → Begin with default report → Yes



BR_KEY_DT	BR_ID	BR_NM	BR_REG	BR_TYP	BR_SOR T_ORDE R_EW
01DEC98:00:00:00	VANC	VANCOUVER	WESTERN	FS	31
01DEC98:00:00:00	VICTO	VICTORIA	WESTERN	FS	32
01DEC98:00:00:00	VICBK	VICTORIA I.F.S	WESTERN	FS	33
01DEC98:00:00:00	CALG	CALGARY	WESTERN	FS	26
01DEC98:00:00:00	EDMO	EDMONTON	WESTERN	FS	20
01DEC98:00:00:00	FREDE	FREDERICTON	EASTERN	FS	4
01DEC98:00:00:00	HALFX	HALIFAX	EASTERN	FS	2
01DEC98:00:00:00	HAML	HAMILTON	CENTRAL	FS	17
01DEC98:00:00:00	KINAT	KITCHENER/WATERLOO	CENTRAL	FS	18
01DEC98:00:00:00	LOND	LONDON	CENTRAL	FS	21
01DEC98:00:00:00	MONC	MONCTON	EASTERN	FS	5
01DEC98:00:00:00	MTLCA	MONTREAL-CARTIER	EASTERN	FS	6
01DEC98:00:00:00	MTLCN	MONTREAL CENTRAL	EASTERN	FS	8
01DEC98:00:00:00	OTTA	OTTAWA	EASTERN	FS	10
01DEC98:00:00:00	QUEBE	QUEBEC	EASTERN	FS	7
01DEC98:00:00:00	STJNF	ST.JOHNS, NFLD.	EASTERN	FS	1
01DEC98:00:00:00	TORY	TORONTO-YORK	CENTRAL	FS	13
01DEC98:00:00:00	MTLBK	MONTREAL BROKERAGE	EASTERN	BROKERAGE	8
01DEC98:00:00:00	CALBK	CALGARY MULTILINE	WESTERN	MULTILINE	26
01DEC98:00:00:00	EDMBK	EDMONTON MULTILINE	WESTERN	MULTILINE	20
01DEC98:00:00:00	LONBK	LONDON MULTILINE	CENTRAL	MULTILINE	21
01DEC98:00:00:00	QUEBK	QUEBEC MULTILINE	EASTERN	MULTILINE	7
01DEC98:00:00:00	TORBK	TORONTO BROKERAGE	CENTRAL	BROKERAGE	15
01DEC98:00:00:00	CHCO	CHICOUTIMI	EASTERN	FS	8
01DEC98:00:00:00	WINBK	WINNIPEG MULTILINE	CENTRAL	MULTILINE	22
01DEC98:00:00:00	HAMBK	HAMILTON MULTILINE	CENTRAL	MULTILINE	17
01DEC98:00:00:00	HALBK	HALIFAX MULTILINE	EASTERN	MULTILINE	2
01DEC98:00:00:00	VANBK	VANCOUVER BROKERAGE	WESTERN	BROKERAGE	31
01DEC98:00:00:00	OTTBK	OTTAWA MULTILINE	EASTERN	MULTILINE	10

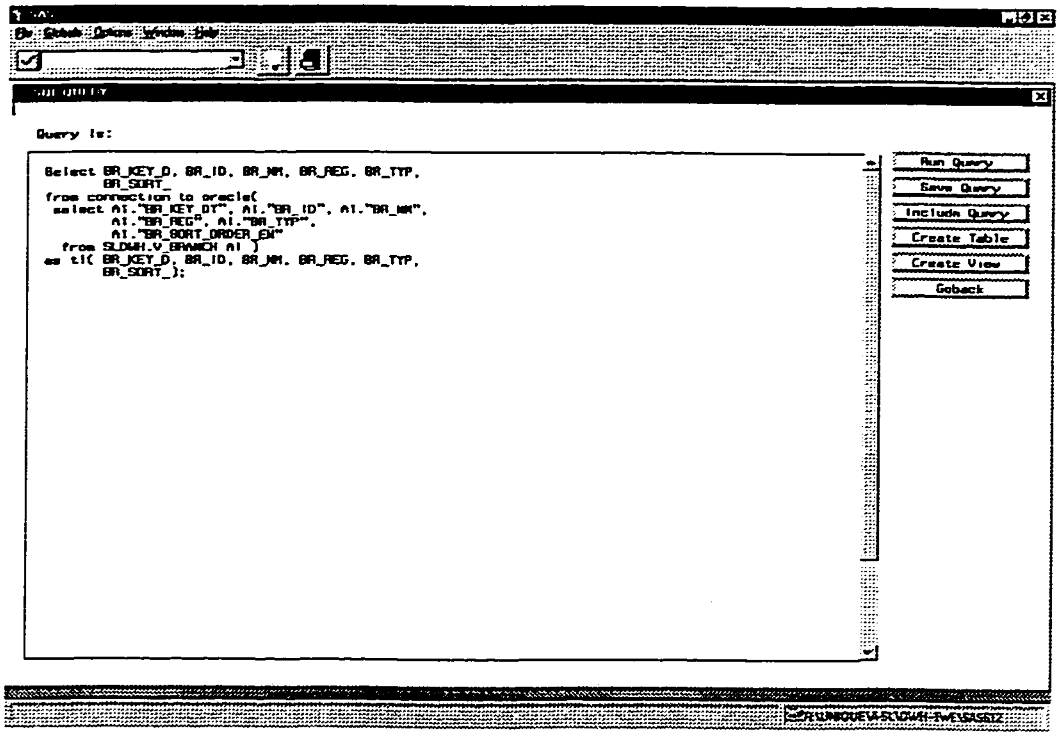
5. Save a Report Definition

File → Save → Report Definition→Library: < Library name > →Catalog: < Catalog Name > → Report Name: < Report Name>



Libname: TEST
Catalog: Report
Report name: testrpt1
Description:

6. View the Report Definition Query
Action(Menu) → Show Query



```
Select BR_KEY_D, BR_ID, BR_NM, BR_REG, BR_TYP,
       BR_SORT_
from connection to oracle(
 select A1."BR_KEY_DT", A1."BR_ID", A1."BR_NM",
        A1."BR_REG", A1."BR_TYP",
        A1."BR_SORT_ORDER_EN"
  from SLDWH.V_BRANCH A1 )
as t1( BR_KEY_D, BR_ID, BR_NM, BR_REG, BR_TYP,
       BR_SORT_);
```

7. Save the SQL Report Definition Query
Save Query(Menu) → Save As External File

3.4.2 Create Report Icon

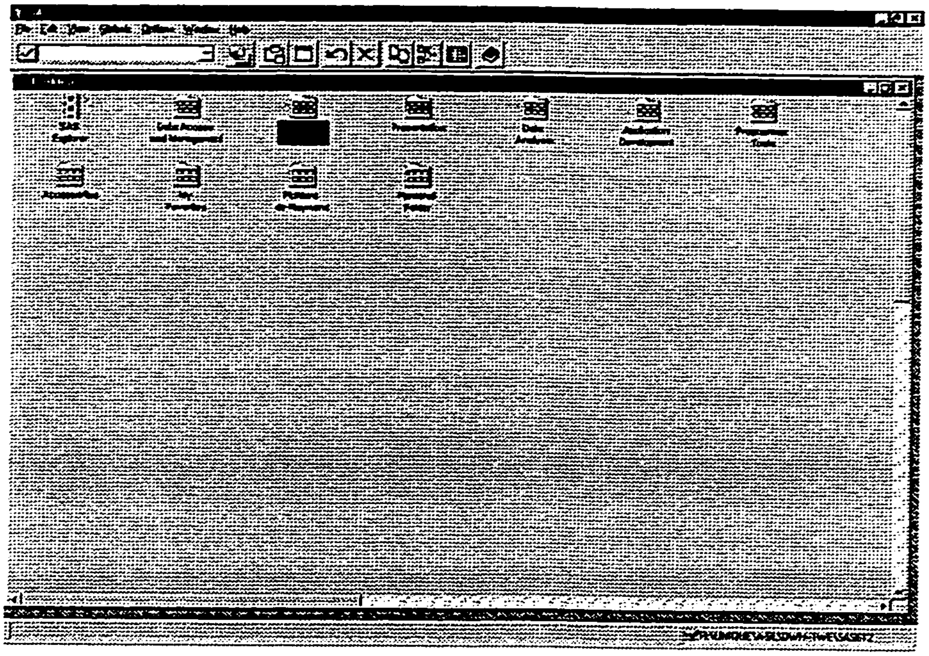
- Prepare a SAS program with ".sas" file extension in any directory accessible to the user with content as follows

```
option remote=sxmq1005
comamid=tcp;
filename rlink "r:\unique\a-sl\dwh-
lwe\sas612\connect\saslink\tcpunix.scr";
signon;

rsubmit;

proc sql;
connect to oracle (user=oracleusername orapw=oraclepasswd
buffsize=35);

create table tmptable as
Select * from connection to oracle
( select * from SLDWH.V_BRANCH );
quit;

proc download data=tmptable
     out=tmptable;
run;

endrsubmit;

proc report data=tmptable report=test.report.testrpt1
windows;
run;
```

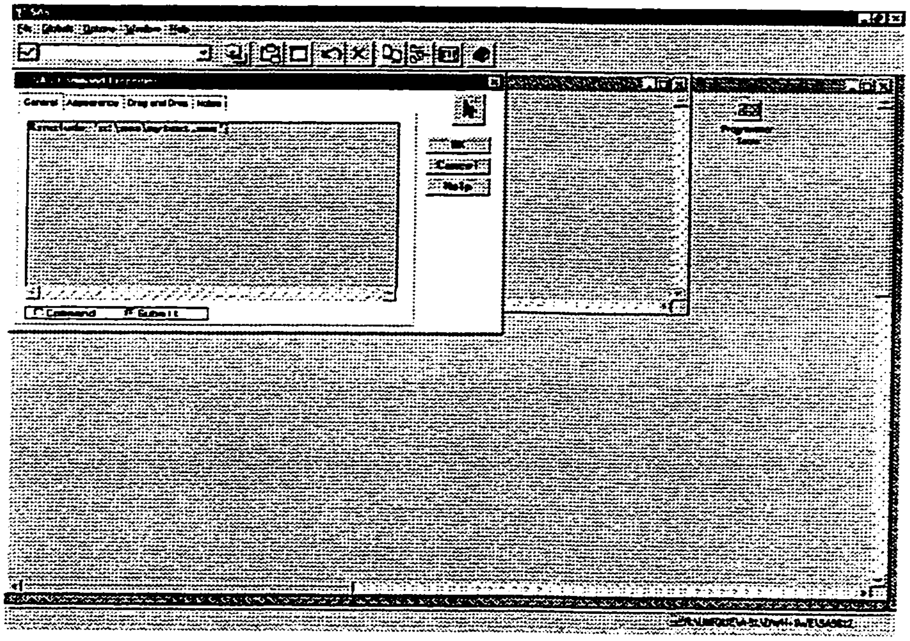
note:

1. " sxmq1005 " is the name of the host machine where ORACLE data warehouse resides
2. "r:\unique\a-sl\dwh-lwe\sas612\connect\saslink\tcpunix.scr" is the SAS tcpunix file
3. " oracleusername " and " oraclepasswd " is the ORACLE user name and password
4. " Select * from connection to oracle (select * from SLDWH.V_BRANCH); " is the SQL statement used in creating the report definition
5. "test.report.testrpt1" is the name of the report definition together with its library name and catalogue name

- Open a Personal Folder



- Right mouse click inside a personal folder → add item(menu) → SAS command



- Enter "%include 'u:\sas\test.sas' "(one assumption is made here that the SAS program created in step 1 is stored as "u:\sas\test.sas".
- Choose the "submit" radio button.

3.5 Complete Reorganization Strategy with tdwh example

Steps	Owned by (UNIX)	Executed by (database)
1. **Edit startup file in " /tdwh/pfl "** • Inittdwh.ora • Configtdwh.ora 2. **Create symbolic links from "$ORACLE_HOME/dbs"** 3. **Comment out the rb statement in confiftdwh.ora**	Oracle	
4. **Shut down the instance**		Internal(svrmgrl)
5. **Start instance and create database** • Comment out the rb segment • /tdwh/dba/adm/crdb.sql	oracle	Internal(svrmgrl)
6. **Create rollback segments and others** • /tdwh/dba/adm/crrb.sql	oracle	Internal(svrmgrl)
7. **Create tablespaces, including datafiles** • /tdwh/dba/adm/crts.sql	oracle	Internal(svrmgrl)
8. Create dba(/tdwh/dba/adm/crdba.sql) • guduch1<dana> • hejian1<test99> • omouma1<omouma1> • thtan91<thtan91> • backtrack<trackback>	oracle	Internal(svrmgrl)
9. Shut down instance/database **10. Uncomment the rb segment** **11. Start up the instance/database**		Internal(svrmgrl)
12. **Create role** • /tdwh/dba/adm/crole.sql • dwh_user • dwh_adm • dwh_upd	oracle	Dba(sqlplus)
13. **Create schema owner and database users** • /tdwh/dba/adm/crowner.sql • sldwh<sldwh> • /tdwh/dba/adm/cruser.sql • tdwh(external) • dwread<read68> • jigran2(external) • jopers2(external) • madixo1(external) • rothor2(external)	oracle	Dba(sqlplus)

• sas96(external)		
14. **Create 22 basic table definition in /tdwh/dba/sqltable/** (crbastab.sql) • ctbpsys.sql • ctbbra.sql • ctbint.sql • ctbcli.sql • ctbdate.sql • ctbpolk.sql • ctbpol.sql • ctbpola.sql • ctbcvg.sql • ctbcvga.sql • ctbinv.sql • ctbcomm.sql • ctbcomma.sql • ctbipers.sql • ctbintpe.sql • ctbintyd.sql • ctbintyn.sql • ctbistpe.sql • ctbistyd.sql • ctbistyn.sql • ctbbrmkt.sql • ctbbrtgt.sql	oracle	Sldwh(sqlplus)
15. **Create stored views definition** (/tdwh/dba/stview/crstview.sql)	oracle	Sldwh(sqlplus)
16. **Create historical table definition** (/tdwh/dba/histable/crhistab.sql)	oracle	Sldwh(sqlplus)
17. **Create 21 unique index(primary key) for tables** (/tdwh/dba/sqltable/crprmkey.sql) • cpkbra.sql • cpkint.sql • cpkcli.sql • cpkdate.sql • cpkpolk.sql • cpkpol.sql	oracle	Sldwh(sqlplus)

• cpkpola.sql • cpkcvg.sql • cpkcvga.sql • cpkinv.sql • cpkcomm.sql • cpkcomma.sql • cpkipers.sql • cpkintpe.sql • cpkintyd.sql • cpkintyn.sql • cpkistpe.sql • cpkistyd.sql • cpkistyn.sql • cpkbrmkt.sql • cpkbrtgt.sql		
18. **Create indexes on intrmd and polkey tables** (/tdwh/dba/sqltable/cridxall.sql)	oracle	Sldwh(sqlplus)
19. **Create triggers for branch_markt and branch_target tables** • /tdwh/dba/trigger/ctrgbrmk.sql • /tdwh/dba/trigger/ctrgbrtg.sql	oracle	Sldwh(sqlplus)
20. **Create views**(/tdwh/dba/view/crviewal.sql) • vintrmd.sql • vlfpk.sql • vlfpol.sql • vlfcvg.sql • vulpk.sql • vulpol.sql • vulcvg.sql • vulinv.sql • vnbylfs.sql • vnbyuls.sql • vnbycmbn.sql • vnby.sql • vbrnby.sql • vnsalemk.sql • vnstatmk.sql • vfcylfs.sql • vfcyuls.sql	oracle	Sldwh(sqlplus)

• vfcycmbn.sql • vfcy.sql • vbrfcy.sql • vfsalemk.sql • vfstatmk.sql • vperslf.sql • vpersul.sql • vpers.sql • vbranch.sql • vbrtgt.sql • vtgtstat.sql		
21. **Grant view privileges to roles(dwh_user & dwh_adm)** • /tdwh/dba/view/grtorole.sql	oracle	Sldwh(sqlplus)
22. Create public synonyms • /tdwh/dba/adm/crsynonm.sql • V_BRANCH • V_INTRMD • V_LF_POLICY_CUR • V_UL_POLICY_CUR • V_LF_COVERAGE_CUR • V_UL_COVERAGE_CUR • V_UL_INVESTMENT_CUR • V_LF_INTPERS • V_UL_INTPERS • V_INTPERS • VM_BR_STAT_NBY • VM_BR_STAT_FCY • VM_BR_SALE_NBY • VM_BR_SALE_FCY • BRANCH_MARKT • BRANCH_TARGET • V_BRANCH_TARGET • V_BR_TGT_STAT_NBY • H_LF_POLICY • H_LF_COVERAGE • H_UL_POLICY • H_UL_COVERAGE • H_UL_INVESTMENT	oracle	Sldwh(sqlplus)
23. Revoke privileges over granted to DWH_USR	Oracle	Sldwh(sqlplus)

• /tdwh/dba/adm/revokpvl.sql		
24. **Import all basic table data** • Uncompress /exp/pdwh.dmp.Z • /tdwh/dba/adm/impbastb.ksh • /tdwh/dba/adm/importpar.txt	oracle	Any valid UNIX user
25. **Create indexes for v_lf_policy_cur and lf_coverage_cur during each refresh cycle**(new storage and unrecoverable clause in create indexs) • /tdwh/sql/cixstvue.sql	tdwh	Tdwh(sqlplus)
26. **Populate stored views during each refresh cycle** • /tdwh/sql/stview.sql	tdwh	Tdwh(sqlplus)
27. **Populate historical tables during each refresh cycle** • /tdwh/sql/crhist.sql	tdwh	Tdwh(sqlplus)
28. Create ad-hoc table pk_cur during each refresh cycle • /tdwh/sql/crpk_cur.sql	tdwh	Tdwh(sqlplus)
29. Adjust the export path of historical snapshots to /tdwh/hist • /tdwh/bin/dwhephst.ksh • /tdwh/bin/expsnpsh.ksh	tdwh	Tdwh(sqlplus)

3.6 Oracle Users

Type	User Name	Roles	System privileges
dba	• Guduch1 • Hejian1 • Omouma1 • Thtan91 • Backtrack	DBA	UNLIMITED Table Space
Application	• Jigran2 • Jopers2 • Madixo1 • Rothor2 • Sas96 • Rabove1	DWH_USER	Create Session
Schema owner	• Sldwh	CONNECT	Create public synonym Create Trigger Drop public synonym
Production user	• Tdwh	DWH_ADM	Create Session
ODBC	• Dwread	DWH_USER DWH_UPD CONNECT	
Other users	• Slacadm • Slacmon • Slacsec • Dbsnmp	*created by system DBA	

4 Bibliography

1. Adhikari, Richard. "Migrating legacy data", Software Magazine, Vol. 16, No. 1(January 1996), 75 –80
2. Bort, Julie. "Can Message brokers deliver?" Software magazine, Vol. 16, No.6(June 1996), 70-76
3. Brown, Scott. "Try slice and dice", Computing Canada, Vol.21, No.22(October 1995), 44
4. Chen, Minder. "A model-driven approach", Journal of Management Information System, Vol. 11, No.4(April 1995), 33-63
5. Cole, Barb. "Data warehouse demands straining corporate", Network World, Vol.12, No.35(August 1995), 1-83
6. Fahey, Bill G. "Building an ABC data warehouse", Management Accounting, Vol. 77, No.9(March 1996), 33-36
7. Francett, Barbara. "Warehousing", Software Magazine, Vol.14, No.8(August 1994), 68
8. Greenberg, Ilan. "OLAP or ROLAP?" InfoWorld, Vol.18, No.24(June 1996a), 1,69+
9. Gupta, Vivek R., "An Introduction to Data Warehousing", System Services Corporation White Paper, 1997
10. He, Jianhui. "Technical Report for the Marketing Data Warehouse", Project Technical Report, 1999
11. Hoffman, Thomas, and Nash, KimS. "Going outside for data warehousing", Computerworld, Vol.30, No.17(April 1996a), 6
12. Inmon, William. "Building the Data Warehouse", Wiley, New York, NY, 1992
13. Kimball, Ralph et al. "The Data Warehouse Life Cycle Toolkit", Wiley, New York, NY, 1998
14. LaMonica, Martin. "SAS to beef up object support in SAS system", InfoWorld, Vol.17, No.45(November 1995a)
15. Raden, Neil. "Maximizing your warehouse(PART 1)", Information Week, No.571(March 1996c), 42-48
16. Richman, Dan. "Parallelism takes aim at time/space quandary", Computerworld, Vol.30, No.23(June 1996c), 51
17. Sakaguchi, Toru et al. "A Review of the Data Warehousing Literature", University of Memphis White Paper, 1996
18. Teach, Edward. "Plumbing the data", CFO:The Magazine for Senior Financial Executives, Vol.12, No.5(May 1996c), 55
19. White, Colin. "Data delivery", Network World, Vol.12, No.20(May 1995), 39-42
20. Whitten, et al. "Data Warehousing", Communications Week, No.612(June 1996), 104