The Role of Tools in Development of a Data Warehouse

M.C. McCabe, D. Grossman
Office of Information Technology

Abstract

We discuss the tools required for building a data warehouse. We describe key features of each tool. Software tools for data warehousing include data modeling, database management systems, data extraction and migration, data validation, metadata, and job schedulers. We also describe project management methods we have developed for building a data warehouse. Specific tailoring of traditional software development methodology for data warehouse projects is proposed including dependent work breakdown structures and a method of estimation and prioritization.

Throughout this paper, we present our experience of a 25 gigabyte data warehouse project which has accomplished an initial delivery to over 400 users.

1.0 Introduction

Making good decisions requires timely, accurate and complete information. Today, more and more organizations are turning to data warehousing for this information. Building a data warehouse is a complex development effort requiring numerous software tools. We have built the first components of a data warehouse containing business administration data. We expect it to grow to 100 gigabytes over the next two years. We selected tools from the myriad of products available. We describe what tools we selected and why. We also describe our lessons learned surrounding these tools. These tools include:

A. Query Tools - tools used to query the data warehouse with ad-hoc and canned queries. They may also be used to build applications such as Executive Information Systems.

B. Data Loading - tools used to load data from flat files to the data warehouse.

C. Data Extract - tools used to migrate data from other sources to flat files.

D. Data modeling - tools used to build and maintain a diagram of the data and relationships between the data.

E. Metadata - tools used to store and manage information about the data warehouse and the data contained in it.

F. Scheduling - tools used to automate the extracting and loading of data into the data warehouse.

Additionally, we have identified the need to tailor existing project management techniques in order to make them more applicable to a data warehouse. These techniques include a dependent work breakdown structure for each subject area, a cost estimating technique and prioritization criteria.

Section 2 describes an overview of data warehousing and a brief review of prior work. Section 3 contains descriptions of the data warehouse we built and discussion of the software tools used. Section 4 describes the project management tools. Finally, section 5 contains our conclusions and directions for future work.

2.0 Background and Prior Work

2.1 The Data Warehouse Concept

Data Warehousing is a software approach to decision support, corporate reporting, and data analysis. Industry analysts estimate that software and consulting services for data warehousing is a market that approaches $1 billion and continues to grow rapidly [1]. A data warehouse is essentially a read-only database designed to optimize query capability by business users (rather than computer specialists). Data is extracted from the business applications and loaded into the data warehouse for read-only access. In this way, the data warehouse separates the Online Transaction Processing System (OLTP) from the reporting database or Online Analytic Processing (OLAP). The OLTP is generally updated interactively, immediately while the data warehouse is generally updated in a nightly, weekly or even monthly basis. The OLTP is designed for efficient updates generally in third
normal form or higher. The data warehouse is designed for efficient queries, frequently denormalizing data to prevent joins and to present the data in a way that is meaningful to the end-users.

Thus, the data warehouse is an infrastructure to support better Decision Support (DSS) applications. The data warehouse itself is not a decision support system but an enabling tool. Formerly, DSS platforms frequently consisted of specialized flat files requiring specialized access programs for each DSS application. The data warehouse approach places all relevant data into one repository and makes it available for ad hoc queries as well as specialized applications.

2.2 Prior Work

Although the term ‘data warehouse’ is relatively new, decision support systems have been around for some time. The first reference to ‘information warehouse’ in an IBM environment was in 1988 [2]. In fact, file extracts for report purposes can be considered the forerunner to DSS and DW efforts. Today, challenges of management reporting, integrating large amounts of data, and automating much of the analysis work is addressed through the use of data warehousing. An excellent summary of tools available today for data warehousing is contained in the Data Warehouse Institute’s 1995 Data Warehousing Roadmap. [3]

The data warehouse is defined as a “subject oriented, integrated, nonvolatile, time variant collection of data in support of management decisions.” [4] Where on-line transaction processing systems are focused around business functions and applications, the data warehouse is organized around the data -- subject areas. Thus, the data warehouse integrates data which may cross organizational and application boundaries. Where OLTP systems may update records, actually changing the values stored, the data warehouse captures transactions and snapshots in time -- nonvolatile, time variant.

Generally, the traditional software development lifecycle is modified for decision support type projects as described in [12] and [4]. [12] proposes the following phases for a “Decision Support Life Cycle”:

Figure 1 Architecture of Initial Implementation
1. Planning
2. Gathering Data Requirements and Modeling
3. Physical Database Design and Development
4. Data mapping and Transformation
5. Data Extraction and Load
6. Automating the Data Management Process
7. Application Development
8. Data Validation and Testing
9. Training
10. Rollout

In [4] it is proposed that any data-driven methodology can be the basis for data warehouse development. Tailoring James Martin’s Information Engineering Methodology [13], [4] eliminates the process-driven portion of the IE approach and defines the following steps as distinguishing the data warehouse development methodology:

1. Data Model Analysis
2. Breadbox Analysis (rough sizing estimates)
3. Technical Assessment (define technical requirements)
4. Technical Environment Preparation (define and configure architecture)
5. Subject Area Analysis (define data requirements)
6. Data Warehouse Design (define physical database)
7. Source System Analysis (identify data sources and conflict resolution rules)
8. Specifications (define code specifications)
9. Programming (code)
10. Population

These approaches are a start but more rigorous application of project management methods is needed to be successful in implementing a data warehouse. In section 4.0, we present our experience of tailoring project management to a data warehouse project.

2.3 The Case Study

The administrative data warehouse (ADW), used as the case study for this paper, originated as a repository for financial management data and has since grown to include budget preparation data and will include personnel data in the next major release. Figure 1 shows the architecture of the initial implementation of the ADW.

The Agency Financial Management System (AFMS) is a mainframe-based accounting software using DB2. Over 600 users enter updates to the system through the application’s 3270-based front-end. The application has over fifteen interfaces with other administrative systems. Almost all reporting is handled through queries and reports against the data warehouse. Because AFMS consists mainly of a commercial product which came with some canned reports, some monthly and annual reports consist of COBOL code against flat files generated by the system.

Since its implementation, the ADW has grown in terms of sources of data and number of users. The second subject area to be added was the budget formulation data. The budgeting system consists of a custom-built application using Integrated Database Management System (IDMS). Over one hundred users enter budgetary data using mainframe online screens. Hundreds of custom reports are available to users through an online menu. Corporate level reporting and trend analysis was conducted through special one-time extracts and reports by a dedicated programmer. Since the implementation of budget data into the data warehouse, high-level analysis has been conducted using interactive, standard Structured Query Language (SQL) queries by the business experts.

Figure 2 shows the sources of data and the loading process for the ADW currently under development. The key goal of growth in the data warehouse is to enable users to combine data from two or more subject areas (i.e. budget and personnel) in a single query. Also, data values are compared across source systems and discrepancies are resolved. For instance, Budget Object Codes are stored redundantly in the financial management application and in the budget formulation application. The data warehouse loading programs compare these values and report on discrepancies. Thus, the data warehouse results in more accurate and extensive reporting. A more detailed description of how to reconcile data discrepancies is found in section 3.2

3.0 Building the Data Warehouse

3.1 Data Modeling

Data Modeling is an important task for a data warehousing project. Because the data warehouse, in essence, is a read-only database, the data model is the foundation for the effort. The differentiating factor among data modeling tools for use in data warehousing is the ability to read data description language (DDL) of the source systems and the target platform. This capability makes it possible to have a data model that is accurate and current.

In our project, we used the Information Engineering Facility (IEF) for modeling the data warehouse. This tool was selected because of its availability to our project, our existing expertise with the product, the existence of many
source application's data models in the IEF and its ability to share the model with other projects through the Central Encyclopedia. The use of the IEF provided these advantages but it did not provide any support for working with source systems that were not in the IEF or had no existing data models. Since it is a CASE tool, the code cannot be optimized for performance. This tool cannot read the data description language (DDL) of our source systems nor our target platform. One ramification of this is that the model has been kept up-to-date at a cost of rekeying changes to the database into both the database and the modeling tool. As the ADW grows, this becomes an unacceptable burden--increasing the operations and maintenance costs of the data warehouse and slowing new development. In the future, we will use a different modeling tool to meet the key criteria of being able to read the DDL of the source systems and the target platform.

3.2 Extracting, Transforming and Loading Data

Some industry analysts believe that the problem of data extraction is the weakest area of data warehousing due to reliance on custom software, difficulty crossing platforms and the complexity of data and statistical analysis required [5]. We have found tools for extraction and transforming data to be sufficient and improving each year. So, although it is true that the business analysis and political battles surrounding the data make this task difficult, the technical difficulties are overcome with current tools. In this section, we explore the challenges of data extraction and transformation.

One of the features of a data warehouse is that it consolidates data from many subject areas. Therefore, the data warehouse must pull data from a variety of source systems. Because the data warehouse is optimized for query and the sources are not, transformation and summarization of the data is frequently needed. Finally, efficient loading of the data is critical so that the load process will fit into the time constraints of the nightly window. This is especially important when dealing with large volumes of data. There are many software tools available to assist in this task. First we describe the role of each tool and then present our tool selections and experience in section 3.2.3.

3.2.1 Extracting the data

Our Administrative Data Warehouse must read from commercial database management systems such as DB2, Integrated Data Management System (IDMS), and NOMAD. It must also read from home-grown hierarchical DBMS--GIMS (Generalized Information Management System). Tools used for extracting the data are frequently also used for transforming the data. Our project uses different tools depending on the source and the transformation required. Thus, these tools will be
discussed in section 3.2.3 along with transformation tools.

3.2.2 Transforming the data

There are four main kinds of transformations required:
1. integrating values from disparate sources
2. breaking out embedded information
3. computing summaries
4. comparing values for data assurance.

An example of integrating values from disparate sources is loading information about people from the training system and from the personnel system. Both systems store the social security number. However, one stores it as an eleven character alphanumeric field with hyphens and the other stores it as a nine digit numeric. The data warehouse must transform both values to the data warehouse standard representation for this field.

A more complex transformation is required when two source systems use different units of measure. One may store quantity in ounces and another in pounds. In our data warehouse, the financial management system stores amounts in thousands of dollars. We adopted the standard of storing US monetary amounts to the penny in the data warehouse. Thus, the budgeting system figures are transformed before loading into the data warehouse.

Breaking out embedded information is needed when the source system has more than one piece of information stored in the same field. For instance, the first two digits of our financial project code is the organization designator. The organization is extracted from the project code and stored separately in the data warehouse to permit more efficient queries.

Summarization's are an important means of optimizing retrieval in the data warehouse. Much of the queries against the data warehouse are interested in summaries and trends. While the source system stores the detailed transaction level, the data warehouse must compute the summaries and store those values. In our application, the customer is interested in reporting monthly spending totals by organization and cost category. These summaries are computed and stored monthly so that they may be queried directly without recomputing them with each access.

3.2.3 Extract, Transform and Load Tools Lessons

Our Administrative Data Warehouse uses several tools for extracting and transforming the data depending on the source and the transformation required.

Where the source system is DB2, a simple DB2 unload and load may be used. This is possible if there is no transformation of the data needed and a complete replacement of all rows in the table is acceptable. Generally reference tables (look-up values) are good candidates for this approach. For example, in our system, a table of all current, valid office codes with less than one thousand records is replaced every night using DB2 unload and load utilities. We favor the DB2 load and unload for its simplicity, when it is possible to use it. However, most applications require at least some validation or transformation of the data and thus straight DB2 is insufficient.

Another approach we use is IEF generated code. This is appropriate when complex transformation logic is needed or when routines already implemented in IEF are available. For instance, the updates to the budget are dumped from the source system into a flat file and then loaded into the data warehouse using IEF. One IEF program loads the detail tables and another computes summarization values and loads the summary table. For capability not supported by the CASE tool, we write external COBOL or PL/I programs. The IEF CASE tool has proven to have many handicaps in the data warehouse environment such as contention with other IEF users and the need for external action blocks when reading outside the IEF. Because the data warehouse programs are entirely batch, with no need for online screens or an interactive application, the benefits of the CASE are limited. In the future, our IEF load programs will be converted to ETI's Extract product or a third generation programming language.

Finally, ETI's Extract tool was used for developing many of the extract, transformation and load programs. It was selected over comparable tools because it was already available within our organization. This tool is well suited for moving and transforming data. The data mapping capability is designed to support this sort of application and the tool is customizable through the use of templates in order to support various platforms and unique requirements. We use this approach to load the personnel data. An extract program is written to read the source system and create a flat file. Another extract program reads that flat file, transforms the data, and loads the data warehouse. We have found that the ETI tool has been a good choice.

We have found the key features of the extraction and transformation tools include: ability to read the source files, ability to perform calculations on fields including look-ups based on values and summarization's and the ability to reformat fields.
3.3 Data Validation

Comparing data warehouse data values with data from disparate sources is an important part of assuring data accuracy. The data warehouse project must identify which source is the final authoritative source -- which one wins. If the employee's address is stored in the payroll system and in the personnel management system, the data warehouse will only load it from one of those sources. Sometimes, the address originates in one system, is passed to the other and then updates are allowed in the second location but not automatically sent back. In this case, the data warehouse can report on discrepancies to the sources for correction.

Determining which values “win” and get stored in the data warehouse is one challenge faced in building a data warehouse. Sometimes it is clear which system is the originator or “owner” of the data. In our case, spending/budget categories originate during the budgeting process and are first input into the budgeting system. However, the finance application does permit changes to these values and conflicts may result. In this case, it is clear that the budgeting system “wins” and is stored in the data warehouse while the discrepancy is reported to both source system teams. Other times, the winner is not as easy to determine. For instance, what organizational component an employee is assigned to is recorded in the personnel system, but the payroll system also reports this data. In the event of a conflict, should the assignment be the unit that the payroll was actually expensed from or the unit that the personnel assignment states? This can only be decided by agreement from the managers of the source systems and the data warehouse team. In our implementation, we chose the personnel assignment as the official source for this information. Partly, this was chosen because the decision-makers want their reports accurately reflecting the assignments as recorded in the Personnel system. Another factor was that the Personnel system is currently being migrated to the data warehouse payroll data will not be available in the data warehouse until later. It is important that the customers of the data understand where the data originated. Metadata provides this insight. See Section 3.6 for more discussion of metadata.

Automated tools for data integrity such as Vality Technology's Integrity product and many metadata tools exist. However, we have used SQL to run comparisons within the data warehouse. We have found SQL to provide most of the capabilities needed for verification, sanity checks, etc. In some cases, it was necessary to write 3GL programs to read the source systems and run comparisons against the data warehouse to report discrepancies.

3.4 Storing the Data

3.4.1 The DBMS

The DBMS is the foundation of the data warehouse. Relational DBMS products are the most common choices today for data warehousing. However, many multidimensional database products are marketed specifically for data warehousing and include many special features. The key capabilities in the DBMS for a data warehouse include features to optimize read-only access such as a parallel engine, special indexing capability and any read-only facilities.

For this project, DB2 was selected as the DBMS. Primarily, DB2 was chosen because of the predominantly MVS environment we were building to and because of the installed base of source systems and the in-house experience with DB2. DB2 provides database administrators with the capability to terminate long-running queries which is vitally important in an environment where users are permitted to perform ad hoc queries of their own construction.

In our experience, DB2 has been an excellent choice. The start-up learning curve was low because of existing skills-base with DB2. DB2 has been able to handle our workload of less than 25 GB of data with several hundred users executing canned and ad hoc reports. Response time remains at an acceptable level for most queries.

Our future plans include migrating to a small machine for savings in storage costs and simplification of the architecture for server-based source applications. To this end, we migrated 2GB of the data warehouse to Unix and NT platforms for performance evaluation. We found performance on a sample set of queries to be acceptable.

3.4.2 Design Considerations

Although there are cases of data warehouses being implemented with a normalized design [6], the denormalized approach, introduced in [7] and [8] is generally considered more appropriate and a better performer. The star schema is the most popular structure, in which one large “fact” table is surrounded by a series of related tables. Typically, the most frequently accessed information is stored redundantly in the fact table to prevent frequent joins with the related, reference tables. This approach yields the following benefits [13]:

- creates a database design providing fast response times
- can be easily modified or added to
- parallels how end-users generally think of and use the data
• simplifies the understanding and navigation of metadata for users and developers
• broadens the choices of front-end data access tools, as some products require a star design

We used a star schema for the administrative data warehouse. An example of this approach is the budget_fact table which stores records containing budget line-items. Each line item is related to several “dimension” tables which give additional attributes of codes and abbreviations stored in the “fact” tables. For instance, the “fact” record specifies that this budget line item is for the organizational unit called “research department”. It is related to a record in the organizational unit table which contains a lot of information about the research department such as where it is located, how many positions it is permitted, etc.

3.5 Query Tools

Important features to look for in a query tool include: powerful query ability, intuitive query interface (i.e. Query By Example and point-and-click selection), computational capability, complete reporting and graphing, export/interoperate with other tools, and open connectivity (API) with email, applications, etc. One query tool may not meet all query requirements. For instance, it makes sense to select one tool for building executive information systems and another for ad hoc query users. There are hundreds of query tools on the market today ranging from connecting an Excel spreadsheet to a database to using a SAS analytic engine.

We selected Trinzic Corporation’s Forest and Trees product for querying and presenting financial data. This product was selected for its strengths in representing financial data in tabular and graphical formats, its capability to compute functions on the data and its ability to interact with other Windows’ applications such as exporting data to Excel. An unknown benefit of this product has turned out to be its interoperability with Lotus Notes which is installed in over 90% of our organization. One may send a Lotus Notes which includes query results from within the Forest and Trees application. In addition, one may trigger the note to be sent based on the results of the query. For instance, the note might be sent only if the amount returned exceeded one hundred dollars.

Figure 3 displays the opening menu for the budget analysts application. The Windows interface is a big advantage for our finance officers who must use a mainframe-based accounting system.

3.6 Repository/Metadata

Metadata is frequently emphasized as an element that can differentiate data warehousing from previous reporting mechanisms. Metadata has three roles in the data warehouse:
1. a directory of contents (the structure of the data)
2. a mapping back to originating sources (the winners)
3. a guide to algorithms used for summarizing the data

The metadata technology allows one to navigate the data warehouse and understand the locations and structures of the data [9]. The key to an effective metadata tool is it’s ability to interrelate with all the components of the data warehouse environment. We strive to eliminate manual upkeep of metadata to ensure current, accurate information. The tool must import data from a variety of sources including: the DBMS, the extract and load programs which may be another tool, a fourth generation language, or simply database utilities, and in some cases, documentation.

Rudimentary metadata is provided through query tools that provide capability to read the data dictionary and sometimes other columns of the System Catalog tables of the DBMS. However, access to these tables is frequently restricted to developers only. In addition, more information is required to adequately provide metadata. A repository tool is needed to store and control (or enable) access to this information. Prism’s Directory Manager and Reltek’s (now PLATINUM’s) Repository are leading tools for metadata implementation.

For the ADW project, we analyzed metadata requirements and designed a custom solution for metadata implementation. However, due to the manual-intensive maintenance required, the implementation of metadata was postponed to a later phase of the project when an automated tool could be selected and used. Other organizations have opted to postpone implementation of metadata in favor of making raw data available quickly and gaining a better understanding of the data warehouse environment [11]. However, it is important to have robust metadata to enable unsophisticated end-users to effectively explore the system. And metadata is considered a critical success factor for very large data warehouse efforts.

3.7 Automating the Load Process (CA7)

Automating the load is critical when dealing with many source systems. Each system has its own nightly processing to complete before availability each day. The loading of the data warehouse must not impact availability of the OLTP systems and it must complete