A proposed framework for analytical processing of information in the dairy industry using multidimensional data models

By
Aisha Ghaffar
Department of Animal Science
McGill University, Montreal

November, 2012

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirements of the degree of Master of Science

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations

UNIVAC	Universal Automatic Computer
IC	Integrated circuit
PC	Personal computer
DBMS	Database management systems
RDBMS	Relational database management system
OLTP	Online transaction processing
ER	Entity relationship
ODBC	Open database connectivity
SQL	Structured query language
OLAP	Online analytical processing
ADBS	Analytical databases
PDB	Production databases
ROLAP	Relational online analytical processing
MOLAP	Multidimensional online analytical processing
MDX	Multidimensional Expressions
MUN	Milk urea nitrogen
SCC	Somatic cell count
FREQ procedures	Frequency procedures
SAS	Statistical analysis system
API	Application programming interface
OLEDB	Object Linking and Embedding, Database
CDN	Canadian Dairy Network

Abstract

In the dairy industry, datasets pertaining to the milk recording of cows can be extremely large and complex, especially in the Province of Quebec where management and feed information are also collected for on-farm advising. Any subsequent analysis of these data for strategic (or even tactical) decision making is often impeded by the transactional nature of the existing databases, whose main purpose is often to produce regular and routine reports. Since conventional database management systems mostly support simple and short queries and flat views of data, they are less than ideal for the analysis of large datasets, particularly those which contain data of varying dimensions. In recent years, the high value of multidimensional data has been recognized as an important resource in both the academic and business communities. The wider recognition of data warehousing and On-Line Analytical Processing (OLAP) applications has highlighted their importance. The dairy industry is an excellent example of an area where the analysis of its data, and the subsequent decision-making process, could significantly benefit from the implementation of data warehousing and OLAP techniques. While these technologies have already been used to good advantage for the analysis of business data, the unusual nature of dairy data poses certain challenges which are addressed in this study. These include selection of a data model which best suits the hierarchical nature of the data, selection of the highest and lowest hierarchy for data aggregation, and the definition of functions (pre-aggregation) to improve query performance.

In order to investigate the use of an OLAP system for Quebec milk-recording data, a number of multidimensional data models were compared. The star, snowflake and fact-constellation schemes each displayed advantages and disadvantages for the particular data (and their structure) in this study. The star schema did not support many-to-many relationships between fact and dimension tables, and creating combination dimensions (e.g., herd_cow) with a key (such as herd_cow_testdate), resulted in an unmanageable record length in the dimension table, thus rendering the model impractical. Many-to-many relationships were captured by a snowflake schema, by normalizing herd, cow and test day dimensions. In order to achieve an exact aggregation of milk components on each test day and for each cow, a herd_cow bridge dimension was implemented within a snowflake model which had a composite key of herd and cow. The lowest granularity level was test day and the highest was herd, but data could also be rolled up to regions. Queries could subsequently be directly executed on a cube structure, since data were stored in a multidimensional online analytical processing (MOLAP) server. All of the pre-aggregation was typically based on the milk-production test date, but could also support analysis at the individual cow level. The cube structure supports “drill down”, “roll up”, and “slice and dice” operations as an aid to the data analyses. Data could also be exported to Excel pivot tables as a means of simple overview reporting. It is felt that the examination of these technologies, and their future implementation, may lead to increased value for the dairy industry as their large quantities of data are explored for better management and strategic decision making.

Résumé

Dans l'industrie laitière, les ensembles de données relatives au contrôle des bovins laitiers peuvent être extrêmement vastes et complexes, en particulier dans la province de Québec où des données de régie et d'alimentation sont également collectées pour le service conseil. Une fois ces données collectées, leur analyse ultérieure pour la prise de décision stratégique (ou même tactique) est souvent entravée par la nature transactionnelle des bases de données existantes, dont le principal objectif est de produire des rapports réguliers et de routine. Puisque les systèmes classiques de gestion de bases de données servent principalement à des requêtes simples et courtes et à des vues simplifiées de données, ils sont loin d'être idéaux pour l'analyse de grands ensembles de données, en particulier ceux qui contiennent des données avec de multiples dimensions. Dans les dernières années, la valeur élevée de données multidimensionnelles a été reconnue comme une ressource importante tant dans les milieux universitaires que d'affaires. Une plus large reconnaissance de l'entreposage de données et de techniques d’analyse comme le traitement analytique en ligne (traduction de l’anglais de On-Line Analytical Processing ou OLAP) a aussi mis en évidence l’importance de ces données. L'industrie laitière est un excellent exemple d'un domaine où l'analyse de ses données et les prises de décision qui en découlent pourraient grandement bénéficier de la mise en œuvre de l’entreposage de données et des techniques OLAP. Bien que ces technologies aient déjà été utilisées pour l'analyse de données dans certaines autres industries, la nature inhabituelle des données laitières pose certains défis qui sont abordés dans cette étude. Ceux-ci comprennent la sélection d'un modèle de données qui convient le mieux à la nature hiérarchique des données, la sélection des niveaux hiérarchiques les plus élevés et les plus bas pour l'agrégation des données, et la définition des fonctions de pré-agrégation pour améliorer les performances des requêtes.

Afin d'étudier l'utilisation d'un système OLAP pour les données de contrôle laitier québécois, un certain nombre de modèles de données multidimensionnelles ont été comparés. Les modèles en étoile, flocon de neige et constellation ont chacun démontré des avantages et des inconvénients pour afficher les données (et leur structure) dans cette étude. Le schéma en étoile n’a pas supporté les relations plusieurs-à-plusieurs entre les tables de faits et de dimension, et la création de dimensions combinées (e.g., troupeau_vache_date-de-test) avec une clé a entraîné une longueur d'enregistrement dans la table de dimension qui était très difficile à gérer, rendant ainsi le modèle impraticable. Les relations plusieurs-à-plusieurs ont été capturées par un schéma en flocon, en normalisant les dimensions troupeau, vache et date de test. Afin de réaliser une agrégation exacte des composants du lait à chaque jour de test et pour chaque vache, une dimension pont troupeau_vache a été développée au sein d'un modèle en flocon de neige qui avait une clé composée troupeau-vache. Le plus bas niveau de granularité était la date de test et le niveau le plus élevé était le troupeau, mais les données pouvaient également être agrégées au niveau de la région. Une fois le modèle implanté et les données stockées sur le serveur suivant une approche multidimensionnelle OLAP (MOLAP), les requêtes pouvaient ensuite être exécutées directement sur la structure de cube. Toutes les pré-agrégations ont été généralement fondées sur la date de test, mais elles pouvaient également supporter l'analyse au niveau de chaque vache. La structure du cube permettait les opérations d’analyse par désagrégation (drill-down), agrégation (drill-up) et blocs de données (slicing). Les données pouvaient également être analysées dans des tableaux croisés dynamiques à l’intérieur d’un chiffrier électronique. L’utilisation éventuelle de ces technologies par l’industrie laitière pourrait accroître la valeur du grand volume de données disponibles et augmenter leur utilisation pour améliorer la gestion et la prise de décision stratégique.

Acknowledgements

I would like to express my thanks to all of those who made the completion of this work possible. Firstly, I would like to express my sincere gratitude to my thesis supervisor, Dr. Kevin M. Wade, for his guidance, determination and support. Secondly, I wish to express my respectful thanks to Dr. René Lacroix for his precious advice and innumerable hours of discussion. I also express my respectful thankfulness to Dr. Roger Cue for his valuable advice on the available datasets.

This study was made possible with the financial support of the Animal Science Department, McGill University and the Higher Education Commission, Government of Pakistan. I am very thankful to Valacta for access to the data used in this research.

I also want to express my gratitude to the staff, Barbara Stewart and Cinthya Horvath, and the students of the Department of Animal Science for their friendship and their support. Special thanks are offered to my fellows in Dairy Information Systems lab for all the good times and valuable support.

I also wise to express my sincerest thanks to my husband Imran, my son Dawood and my daughter Abeer for their countless support and love during this study. My profound appreciation for my parents, brothers, and sisters for their continuous love, prayers, and encouragement to pursue my studies. Above all I wish to avow my belief in the Almighty who has helped me to complete this thesis.

Chapter 1: Introduction

On August 30, 1890, the popular American magazine, *Scientific American* (v. 63 no. 9),, devoted its cover to a number of pieces of equipment constituting the U.S. Census Service's new punched-card tabulating system for processing census data. The development of the modern electronic computer began in the mid-twentieth century, after the onset of WWII, when governments began to require computers to meet various strategic needs. An engineers' invention, at Harvard University, of an electronic calculator for the American Navy was another large step forward in the field of computers. Although this machine was very complicated and very large, it could perform simple arithmetic calculations. In 1952, UNIVAC (Universal Automatic Computer) was the first computer used to predict a presidential election in the United States. These first generation computers had their own unique machine language, used vacuum tubes, and employed magnetic drums for storage of data.

Subsequent to the invention of transistor and advances in magnetic core memory, the computer world underwent a big change. Second generations computers were smaller, faster and more energy efficient than their predecessors. The ultimate benefit of the discovery of the transistor was the development of the first supercomputers which could handle massive amounts of data. Such computers were used by different organizations (Atomic energy laboratories and United States Navy Research and Development Centre), however they were too expensive for ordinary businesses, and their effectiveness was lost to the business world. In this generation of computers assembly language was used instead of machine language.

Although the invention of the transistor was a big step in itself, heat generation inside computer components remained a considerable problem. This problem was resolved by the use of sheets of non-conductive quartz rock, on which electronic components were combined to achieve an integrated circuit (IC). As time went on attempts were made to put more components on this IC chip, leading to a smaller third generation computer. Third generation computers had a central program (operating system) which managed memory for the application programs run on them.

Later the sizes and price of the computers were greatly reduced. Large scale integration (hundred of components on one chip) led to very large scale integration (hundreds of thousands of components on one chip) which turned to ultra-large scale integration (millions of components on one chip). The Intel chips took IC's one step ahead by locating the central processing unit, memory, input and output controls in one chip, to meet any number of demands. Computers having these chips were equipped with user-friendly software packages. International Business Machine Corporation (IBM) introduced the personal computer (PC) for use in home, schools and offices, started the era of fourth generation computers. Computers have since become smaller, working their way down from desktops to laptops and then to palmtops. The use of screen icons instead of typing instructions and invention of a device (mouse) that imitated the movement of human hand on screen with cursor was another major advancement in this generation of computers. Due to smaller size and greater convenience of computers, these were used to share memory and software while linked to local area network or through telephone lines. These links turned out to be a gigantic network known as Internet. The most popular application of this network was electronic mail which became available across the world.

It is difficult to define the fifth generation computers as this generation is in its early stages. But this era has considerable achievement of artificial intelligent computers which can use visual input, learn from their own experiences and give output. Other new achievements are (i) parallel processing in which more than one central processing unit work together as one, and (ii) superconductor technology which improves information flow by reducing the resistance in flow of electricity. The good example of fifth generation computers can be expert systems which are presently in use in different industries. Expert systems are systems which either entirely execute or assist in the execution of tasks that are otherwise carried out by human experts. Examples of expert systems include systems to forecast stock prices, programme routes for delivery vehicles or diagnose human diseases.

Undoubtedly computers have a great influence in our lives, since computers can now do much more than computing. Scanners at shopping stores help in calculation of grocery bills, automatic teller machines help in conducting banking transactions, video conferencing helps many of us in arranging lectures across the world, and record keeping of tax payers helps governments in tracking public funds. In parallel to the population growth and urbanization of the last century, computer technology has continued to grow alongside the industries. These have all lead to a need for large-scale information collection, processing and communication. As Governments began to have trouble counting their populations, telegraph companies could not keep pace with message traffic and insurance agencies had trouble processing policies for large numbers of workers. Passionate by these increasingly complex problems, driven by advances in understanding and technique, and powered by the emergence of the Internet, today's science is based on computation, data analysis, and teamwork, achieved through the efforts of individual scientists and theorists (Foster and Kesselman, 1999).

Advances in computation in an increasing number of scientific disciplines have lead to large data-collections which are emerging as important resources, serving a wide community of researchers. The communities of researchers that need to access and analyze these data are often owned by either single or multiple institutions distributed geographically. Likewise the computing and storage resources on which these communities depend to store and analyze their data are also present at single or different locations. The increase in data production arising from the availability of sophisticated computer hardware and software, has led to an information explosion presenting unique and complex challenges for analysis, modeling and drawing of conclusions.

Dealing with large datasets may present many challenges, beginning with data collection and design issues and leading to data quality issues such as noise, missing data and erroneous data. Sinha *et al.,* (2009) emphasizing information management issues related to large datasets, including the identification of relevant and accurate information critical for analysis, integration, access control/privacy and performance. They stated that model appropriateness with respect to model selection for large datasets and associated computational complexity issues, were related to the ability to perform computations in a convenient manner.

Huber (1999) offered a classification of datasets according to their size in 10^x bytes: tiny (x = 2), small (x = 4), medium (x = 6), large (x = 8), huge (x = 10), and monster (x = 12). Huber defined massiveness in terms of aggravation. In this regard, he associates visualization aggravation with $x > 6$ and data analysis aggravation with $x > 8$.

As PCs (personal computers) are excellently matched to the requirements of interactive analysis of medium datasets, until now people have developed considerable experience of data analysis with small and medium datasets. If large datasets are analysed on the same PCs several bottlenecks become apparent. Some limitations may be due to human capacities and others may be due to computational complexity or by technological limits. Others may include financial (e.g. memory cost) or lack of software (for parallel processing).

Along with human-machine interaction, storage requirements and computational problems, difficulty in visualizing large datasets is one of the main human limitations addressed by Huber (1999). The conditions for efficient human-machine interaction are violated in case of large datasets. Necessary prerequisites for human-machine interaction include (Huber, 1999): (i), the task is such that a sequence of reasonably straightforward decisions has to be made, (ii) these decisions must be made in relatively quick sequence, and, (iii) each decision has to be based on the results of the preceding step. With large datasets data complexity may result in decisions not being straight, the human response too slow, such that the human side of the comment loop is broken when response time exceeds the order of human think time. Under such conditions it may be difficult to give a sensible basis for the next decision if one cannot visualize the previous results. Moreover, data storage capacity, as measured in bits per unit area continues to increase, making it possible for certain physics experiments to save as much as peta-bytes (Pb) of data per year. In addition, dramatic improvements in microprocessor performance mean that a simple desktop or laptop is now a powerful computational engine.

Memory size frequently represents a bottleneck preventing the full use of fast processors. However, according to Foster (2002), computational capabilities are falling behind relative to storage, “only” doubling every 18 months and so “only” increasing by a single order of magnitude every 5 years. As datasets increase in size their complexity increases, and their homogeneity is lost, so that these techniques need to analyse such datasets can test the limits of our computing capacities.

With large datasets, processing time problems generally have more to do with storage access than processor speed. To overcome this one will have to produce small, derived datasets that selectively contain the required information and can be accessed quickly. Even then parallel processors and distributed memory create additional complications, making analytical operations difficult and of greater magnitude. Consequently there is a need to develop database structures and access techniques which will allow access, exploration and visualization of this era’s large datasets, rather than use a “one size fits all” approach, as in the past (French, 1995).

Digital data are fundamental to all branches of science and engineering now. Such datasets are held in databases, which are logically integrated collections of data maintained in one or more files and organized to ease their efficient storage, modification, and the retrieval of related information (Frawley et al., 1992). The recorded data have implicit meanings and the databases which house them represent some aspects of the real world, sometimes termed the mini-world. The mini-world’s changes are thus reflected in the database (Elmasri and Navathe, 2000).

Databases are designed, built and populated with data for some specific purpose. These have some planned group of users and some preconceived applications in which these users are interested. A database management system (DBMS) is a collection of procedures for retrieving, storing, and manipulating data within databases (Frawley et al., 1992). The DBMS is a general-purpose software that facilitates the process of defining, constructing and operating databases for various applications. Defining a database involves specifying data types, structures and constraints for the data stored in the database. Constructing the database is the process of storing data on some storage medium that is controlled by the DBMS. Operating a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes and generating reports (Elmasri and Navathe, 2000).

Most of the DBMS in organizations are relational database management systems (RDBMS). A relational database consists of tables (also called relations) made up of rows and columns. No ordering is implied among the rows of a table. Each column is atomic, and repeating items within a column is not allowed. A column, or collection of columns for which different rows must have distinct values, is called a table key (Larson, 1983). Relational database management systems were originally designed for mainframe computers and business data processing applications. Moreover, relational systems were optimized for environments with large numbers of users who issue short queries (Hurson et al., 1993), are well established and reliable, and have proved to be flexible platform for the evolution towards new functionalities that meet at least a few of the demands of novel applications (Bordoloi et al., 1994). Because most RDBMS were developed during the 1990s their architecture has rarely been amended. Designers at that time were not aware of the potential growth of the Internet and no one could expect that one day databases would not only serve for operational tasks in an organization but also be used for data analysis.

The current relational database management system architecture was developed keeping in mind the online transaction processing (OLTP), i.e., the architecture was optimized for the hardware available at the time, to efficiently support OLTP (Halawani et al., 2010). Entity relationship (ER) modeling is a standard technique for building an OLTP system; a system which is designed for many simple concurrent requests. This kind of modelling is also useful for showing the flow of data within an organization (Chaturvedi et al., 2008). OLTP systems (*e.g.*, airline reservations, order entries, and banking transaction systems) handle the daily operational features of business dealings (Chaudhuri and Dayal, 1997; Cui, 2003; Stamen, 1993). In daily business dealings comprehensive and current data are required, wherein the data are read or updated a few records at a time, by way of a primary key (Chaudhuri and Dayal, 1997). The data can be deleted from the OLTP systems, *e.g.*, if an order is shipped today then its record will be deleted from the database (Stamen, 1993). The OLTP systems (operational databases) can be expected to be hundreds of megabytes to gigabytes in size. The critical issues are, consistency and recoverability of the database along with maximizing transaction throughput, which is the key performance metric (Chaudhuri and Dayal, 1997).

Relational databases have been a *de facto* standard over the last two decades for both operational and analytical applications. For example, current enterprise systems have utilized relational databases to incorporate applications across operational areas (order entry), whereas analysing enterprise data (*e.g.*, computing the sales history of all stores over a certain period of time) is a resource-intensive job that is better performed off-line and should not affect the daily operations of the business. The need for analysing business data for trends and strategic information is hardly novel. Data analysis can serve to define requirements, set targets, plan interventions, and evaluate growth. In a perfect world, the analyst could simply query the operational database in a school to see the performance of students in Mathematics, which may point to new ways of training Mathematics teachers. In reality, few operational environments have the bandwidth to allow information seekers to create the elaborate queries necessary for complicated analyses of data. The data which are used for operational systems are physically separated from the data used for analytical processing or for information. The supporting technology for operational processing is also different for both systems, as the users are also different (Jones, 1998). Data in the operational applications are simply used for record keeping, whereas data analysis is the process of collection, presentation and summarization of information contained in the data, with the goal of helping in decision support. A number of different mathematical techniques are used in data analysis, *e.g*., frequency counter, percentage, average, ranks, ratio and standard deviation.

Data in operational applications are file-oriented and consist of linear lists of related values each describing an employee, customer, product or other entity. These data are accessed with a single key and viewed as a simple list on the screen. In contrast, applications which are based on some data analysis, are best viewed in multidimensional format because the user wants to see all the possible combinations of entities and their relationships, *e.g*., performance of different products over time in different markets (Stamen, 1993). So data analysis applications place some rather different requirements on database technology compared to traditional OLTP applications.

Many industries around the world, have presented the challenge of analysing large datasets. The Quebec dairy industry faces the same challenge, since gigabytes of data are produced on the dairy farm every year and data analysis is quite important for drawing up strategies at the dairy farm level. Since dairy farming in Quebec constitutes an important component of Quebec's agricultural economy. The Canadian dairy industry ranks third in terms of value in the Canadian agricultural sector following grains and oil seeds, and red meats. Current dairy production in Canada generated a total for net farm receipts of $5.8 billion and generated sales of $13.7 billion, representing 16.4 % of the Canadian food and beverage sector in 2011 (www.dairyinfo.gc.ca).

Compared to former decades, many of today's farmers consider information management to be an important element of their business. This interest has been stimulated by the availability of powerful new computer hardware and software at a reasonable cost. Software has become easier to use and the hardware/software systems are capable of storing more data and doing more useful analysis on these datasets. Similarly, agricultural extension systems of the past are now being replaced with computer-based decision support systems. For instance, the farmer might want to know the peak milk yield of all second lactation cows that calved between August and March, from the available data (an example of data analysis query).

Most of Quebec dairy farms already have computerized information systems, but these are mainly focused on record keeping and primarily support operational management decisions. To support more steps of the decision-making process these systems should be extended. Priority should be given to (i) extension of the record keeping systems with modules for data analysis, and (ii) development of models to support tactical management decisions. These systems are report-oriented systems and the reports are based on the current activities at the farm. If a farmer wants to know the performance of a herd in previous years, the processing of a large quantity of data on existing systems is necessary. Although not impossible, it might take hours to process such a query. This problem results from a number of issues:

(i) The execution of data analysis queries (analytical queries) requires a great deal of system resources as these queries necessitate a lot of scans and joins. Systems used for analytical query execution are not completely separated from transaction systems, so the sharing of a database or data file slows down both the operational and analysis processes.

(ii) Because of the limitations of a relational database, users can only observe the data in flat views, so data visualization from different perspectives is almost impossible.

(iii) Historical data is very important as the improvement of future herds is largely based on the performance efficiency of past herds. An asset for the dairy sector, these datasets, which typically originate from different organizations, need a storage structure that will allow them to be analyzed within a convenient time span and give multidimensional views of data to decision makers. Such a structure is not supported by relational database management systems.

(iv) When a dataset is sufficiently large the human eye cannot encompass it: thus to produce a summary report with relational databases within seconds is a challenge since the person making the query can lose patience or be easily distracted. There is, therefore, a need to devise such tools as can help in summarizing data quickly from the same datasets.

Prompted by the above needs, the basic idea of this research was to propose a methodology to develop a storage place, which could act as a repository of collected data used for data analysis and decision support. The data available on farms today can prove an advantage in improving on-farm decision-making, but only if interpreted and exploited properly through analytical tools. With the help of these tools, dairy-farm advisors can better help producers with respect in decision-making: *e.g.*, keeping animals in different management groups for better feed usage and reduced cost of production, and enhancing efficiency by harmonizing livestock requirements to feed inputs.

Purpose of study

Nowadays, dairy producers and their advisors in the dairy industry have access to an increasing volume of data. This is available at the farm where daily records have been kept. The detailed production data from all participating resources are actually present in operational systems which support day to day operations. To take effective decisions using these data: (i) should be stored separately from operational systems, since the latter store current data, whereas historical data are required for strategic decision-making and trend analysis; and (ii) these should be modeled in a multidimensional manner to support analytical queries which are the basis of decision-support.

Overall objective

The overall objective of this research was to: (i) develop a methodological framework for implementation of multidimensional models to available dairy data, in order to improve the advisors' ability to analyze data and come up with appropriate decision-support recommendations. To achieve this overall objective, the study was divided into four stages.

(i) To acquire, explore and understand the data. At this stage understanding the data is extremely important; for example, what do missing values for an attribute actually indicate (i.e., are they essential), and does it make sense to calculate an average or sum for a given attribute;

(ii) To modify the data, metadata need to be carefully understood, for each selected attribute in the previous stage so that they can be manipulated mathematically. Desired attributes need to be calculated, and keys selected during the process of data transformation;

(iii) To design different data models to store the selected attributes in a structure which could help in analytical query execution. The efficiency of the models needs to be carefully examined to assess their relative advantages with respect to dairy data; and

(iv) To analyse the data by querying schemas of the DBMS, specifically using a front-end graphical user interface that could interpret user information needs to support on-farm decision making.

Chapter 2: Review of literature

This Chapter considers the existence of valuable data in operational data stores, the need for those data to be extracted, cleaned and transformed for the purposes of specific analyses, and their storage in formats that will allow access to be optimized for analytical purposes. Given the desire to make more use of existing data, their use in decision-support systems is discussed, and the differences between online transactional processing (OLTP) – routine reports – versus online analytical processing (OLAP) – query-based analyses – are explained. In the case of the latter – the subject of interest in this thesis – data can be amalgamated from different sources and modeled with the help of multidimensional data models. Their subsequent storage and analysis can be facilitated by a data warehouse (cube structure) and various cube operations for viewing the data. Finally, some applications of multidimensional data modelling and data warehousing are highlighted from different areas, leading to the topic of interest in this study – the dairy industry.

2.1: Data extraction and cleaning

Data which are to be used for analysis purposes should be stored separately from the organization’s operational database. Analysis of data demonstrates altered functional and performance requirements. These requirements are quite different from those of the OLTP applications traditionally supported by operational databases. OLTP applications usually computerize clerical data processing tasks such as banking transactions and order entry that are the basic daily operations of an organization. These are organized and repetitive tasks and consist of scheduled transactions. The data required by these transactions are detailed, current, read and updated in seconds and accessed on a primary key. Operational databases can range from hundreds of megabytes to gigabytes in size. In contrast, data which are used for analysis and aid in decision-making should be historical. Instead of detailed records, summarized and consolidated data is more desirable in the case of data analysis, which explains why data which are to be analyzed should be separated from operational databases. In this way historical datasets can be created at a storage place other than that of operational data storage. Similarly data analyses require millions of scans and joins, while the process of analysis and most of the queries are *ad* *hoc*; this usually means that using the same database for both operational and analytical purposes will slow down the operational database. Data extraction from operational databases for analysis of data is usually implemented via standard interfaces such as ODBC (open database connectivity) and Oracle Open Connect, etc. (Chaudhuri and Dayal, 1997).

The source of data is a vital factor, as data entry and acquisition are naturally prone to errors, both for simple and complex data. A great deal of effort has been given to minimize errors in this front-end procedure, but more often than not, errors in large datasets are frequent. To avoid these errors during the process of data storage for analysis, it is important to make an attempt to clean the data in some way, with the aim of exploring datasets for possible problems, with the intention of correcting the errors. Naturally, in case of large datasets, doing this task "by hand" is entirely out of the question in the real world, and might take hours to months. A manual process of data cleansing is also difficult and can itself create errors. Powerful data cleaning tools that computerize or greatly assist in the data cleansing process are essential and may be the only useful and cost-effective method to attain a rational quality level in an available dataset (Maletic and Marcus, 2000).

Data cleaning is performed to improve the quality. Data quality problems are also present in standalone data sources such as files and databases. The problems can be due to misspellings during data entry, missing values or invalid data (Rahm and Do, 2000). The most important phases in data cleaning are: defining and determining the error types, and searching for and identifying the error instances. Each phase is difficult and requires the input from a domain expert. Errors can be viewed in an outlier detection process. If a large percentage of data elements conform to a general form (99.9%) then the remaining (0.1%) are likely to be outliers. Outlier values for data elements can be identified with automatically computed statistics. For each data element the mean and the standard deviation are calculated and are recognized, based on those elements that have values in a given field beyond a number of standard deviations from the mean (Maletic and Marcus, 2000). Four standard deviation below and above the mean can be used to remove outliers in an iterative approach. A four standard deviation threshold can keep most of the valid values in the datasets (Pietersma et al., 2006).

2.2: Data transformation and de-normalization

The data transformation process turns the data from its present format into the format desired by the target application. For example, if data on values and quantities are present in an operational database, then computation for profits (transformation) can be performed, so that all three values are stored in, and accessible for analysis. Inclusion and organization of related metadata for both original (data coming from operational database) and transformed data are crucial to the reliability of the data storage place used for data analysis (Thornsbury et al., 2003). The transformed data are summarized; this is mostly done to improve of the efficiency of the system (Gray and Watson, 1998). The transformation process can encompass several functions like data reformatting, recalculation of certain data elements, adding elements of time, defining new keys in the database and merging data from multiple files (Simitsis and Theodoratos, 2009).

The data transformation process typically consists of multiple steps, and there are different tools available for this process. An easy and common approach is the use of the standard query language (SQL) to perform data transformations and utilize the possibility of application-specific language extensions. A transformation step specified in SQL is illustrated in Figure 2.1. This example covers part of the necessary data transformations to be applied to the data source. The transformation describes a view on which additional mappings can be made. The transformation carries out a schema reform with added attributes in the view acquired by dividing the name and address attributes of the data source (Rahm and Do, 2000).

```
CREATE VIEW Customer2 (LName, FName, Gender, Street, City, State, ZIP, CID)
AS SELECT LastNameExtract (Name), FirstNameExtract (Name), Sex, Street,
CityExtract (City), StateExtract (City), ZIPExtract (City), CID
FROM Customer
```

Figure 2.1: An example of data transformation mapping using standard query language (SQL) (Rahm and Do, 2000).

Transformation process also includes data denormalization (Simitsis and Theodoratos, 2009). The concept of normalizing data in operational databases does not apply to data stores prepared for data analysis. In operational databases normalization is a well thought-out and significant tool in avoiding redundancy. It typically requires dividing a database table into numerous tables and defining relationships among the tables. Redundant data can create inconsistencies without normalization and the update of irregularities can take place during deletion and insertion procedures. Database normalization is important in operational databases where data modifications occur rapidly and randomly all over a database (Ahmad et al., 2004). Most of the time, operational databases are normalized up to the third normal form which means that: (i) there are no repeating fields in a database table (First Normal Form), (ii) all non-key attributes are fully dependant on a primary key (Second Normal Form), and (iii) there are no dependencies among non-key attributes in a database table (Third Normal Form). On the other hand, denormalization is a technique to move from third to first normal forms of database modeling in order to accelerate database access and, ultimately, improve query processing. Denormalization is suitable for inactive data, which is historical and used for analysis, to expedite query performance (Kimball et al., 1998). A simple process of converting normalized data into denormalized data can be viewed in Figure 2.2. Accordingly, whilst normalization is a process of dividing a database into smaller tables to efficiently work with data in operational databases, denormalization is a procedure of collecting data tables into larger tables for well-organized analytical processing (Ahmad et al., 2004). This is why redundancies are quite acceptable in the case of analytical processing (Gray and Watson, 1998) and, apart from the primary key, all other fields allow NULL values (Wah and Sim, 2009).

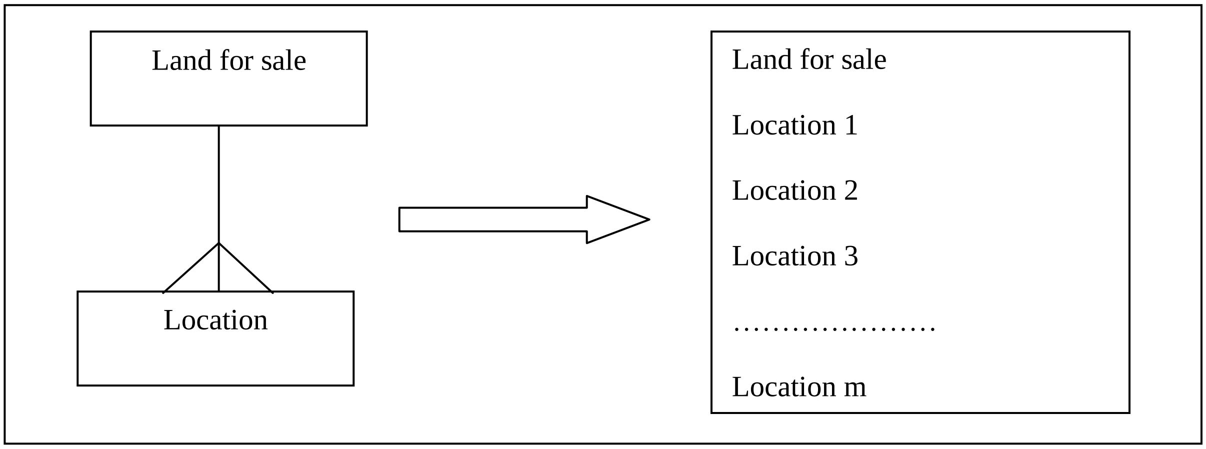


Figure 2.2: A conceptual model for conversion of normalized data into denormalized data (Ahmad et al., 2004)

2.3: Data warehouse

A data warehouse may be defined in different ways; some of them are merely for data, while others may include people, processes, software, tools and data. The global definition is that the data warehouse is a collection of integrated, subject-oriented databases, designed to support decision-support functions where each unit of data is related to some moment in time. The data warehouse is a relatively new concept in response to a major business need; the analysis of extremely large volumes of historical data to answer difficult business questions. These questions can be: What segment of customers buys this product, or with which credit cards do customers pay their bills by the due date? Technology existing before the data warehouse lacked the ability to accurately answer these types of questions. In the past most information systems were designed to create pre-defined reports containing superficial information (AL-Hamami and Hashem, 2009).

Thus, a data warehouse is a collection of technologies that provides ideas and techniques which give users valuable information for faster decision-making (Anil et al., 2008; Chaudhuri and Dayal, 1997; Manole and Gheorghe, 2007). The difference between a data warehouse and a traditional operational database is the volatility of the data. The information in an operational database is constantly changing whereas, in a data warehouse, the information is stable and updated at standard intervals (monthly or weekly). A data warehouse can be updated to add values for the new time period only, without changing values which were previously stored in the warehouse (Alkharouf et al., 2005). So the data warehouse contains summarized data and data having the lowest level of detail (Inmon, 1996). Another difference between the two systems is that operational databases provide an answer to operational requirements, while data warehouses provide an answer to analytical requirements (Manole and Gheorghe, 2007), thus offering the possibility for high quality analyses and complex *ad hoc* queries through user-friendly interfaces (Berndt et al., 2003; Manole and Gheorghe, 2007). A subject which is the field of activity is the basic criterion for the organisation of data in data warehouses, while application is the basic criterion for an operational database (Manole and Gheorghe, 2007).

Managers of organizations have realized that the data stored in databases, represent informational gold mines if properly exploited. The data warehousing approach solves the problem of complex data analysis which could not be achieved with operational databases (Manole and Gheorghe, 2007). The need to build a data warehouse starts from the requirement of quality information within the organization. The data which come from different internal and external sources, having different formats are filtered according to business rules and incorporated into a large single data collection (Manole and Gheorghe, 2007; McFadden, 2002). The contents of a data warehouse may be a copy of a part of operational data or may be the result of preprocessed queries or both (Samtani et al., 1999).

In the process of data warehousing, the raw data fed into the integrated system may also contain "metadata" that define and describe that particular source data. Metadata normally include descriptions of data elements, data types and attributes, and processes and methods of collection. Source data often contain a high percentage of "dirty data" with inconsistent, missing, or incomplete values. Thornsbury *et al.*, (2003) emphasize that data must be "cleaned" and organized, and metadata must be synchronized. Performing data compilation and cleaning functions, including metadata organization, on a regular basis is critical for reducing response time and adding value to the operational database. Such an operational/archival database provides storage for large volumes of data and maintains the historical record.

A data warehouse is part of a larger client/server environment and does not exist in isolation. There are three types of components that comprise the architecture (Inmon, 1996):

i) The data acquisition software (or back end), which extracts data from the heterogeneous data sources (legacy systems and external data), consolidates and summarizes the data, and loads it into the data warehouse.

ii) The data warehouse presentation server, *i.e*., the platform and software (including the repository) that houses the data warehouse. Data marts which are logical subsets of data warehouses will be developed instead of one centralized data warehouse; then a bus architecture will be used for communication within the data warehouse

iii) The client (or front-end) software which allows decision support users to access and analyze data in the warehouse.

In designing a data warehouse, each unit of data is relevant to some moment in time. The process of data warehousing is a collection of decision-support technologies which support the knowledge worker (executive, manager, and analyst) to make better and faster decisions (McFadden, 2002). The characteristics of a data warehouse are illustrated in Table 2.1.

CHARACTERISTIC	DESCRIPTION OF CHARACTERISTIC
Subject oriented	Data are organized by how users mention and apply them
Integrated	Inconsistencies are removed in inconsistent information (i.e., the data are 'cleaned’).
Non-volatile	Read only data. Cannot be altered by users.
Time series	Data are based on time series rather than current status
Summarized	Operational data are recorded into decision-usable form
Larger	Maintenance of time series data, which is why much more data are reserved.
Metadata	Metadata are data about the data.
Not normalized	Data in the warehouse can be redundant
Input	Operational databases

Table 2.1: Characteristics of a data warehouse (Gray and Watson, 1998) .

At present, developmental processes for data warehousing lack an established scheme. Existing data warehousing development methods fall within three fundamental approaches, clearly defined by List *et al.*, (2002): data-driven, user-driven and goal-driven.

(i) The data driven approach explores the data and derives a potential abstract schematic for the data warehouse. This abstract schema is developed after the process of transformation of the operational data sources. The analytical needs of an organization are rarely identified and explained after implementation of a data warehouse (Inmon, 1996).

(ii) The user-driven approach primarily brings out the requirements of the decision-making process by group interviews or by separate interviews of users and experts. Then a model is developed after recognition of facts and dimensions (Kimball et al., 1998).

(iii) The goal-driven approach encloses numerous considerations like the selection of the business process. This allows one to decide the smallest piece of information in the process and then select dimensions and facts. Each business process is considered as a major transaction (Kimball et al., 1998).

There are two ways to build the warehoused data: a bottom-up approach or top-down approach. In the case of the former, the data are obtained from the operational databases relevant to the data warehouse applications. Queries are normally identified in advance, and data are then selected, transformed, and integrated by data extraction tools. Using this bottom-up approach, user queries can be answered immediately as data are present in the data warehouse. For this reason, this approach is deemed realistic, and enhances the performance of the system. In a top-down approach, the data are acquired from the operational databases whenever a query is created. In this case, the warehouse system accesses the operational databases in order to answer the query. Another approach is a hybrid approach, which unites aspects of the bottom-up and top-down approaches. In this approach, some data are stored in a warehouse, and other data can be acquired from operational databases as needed (Hull and Zhou, 1996).

2.4: Decision-support systems

An application which is currently in high demand today with respect to management decision-making is the decision-support system. Management takes information from these systems and make decisions about business growth, levels of stock on hand, etc. It is a challenge to derive answers to business questions from the available data, so that decision-makers at all levels can react quickly to changes in the business environment. The standard transactional query might ask, "When did order x ship?" whereas a classic decision-support question might ask, "How do sales in the South-western region for this quarter compare with sales a year ago?” (Halawani et al., 2010).

Decision-support systems are computer technology solutions that can be used to support complex decision-making and problem-solving. Typical DSS tool design consist of components for: (i) sophisticated database management capabilities with access to internal and external data, information, and knowledge; (ii) powerful modeling functions accessed by a model management system; and (iii) a powerful but simple user-interface design which enables interactive queries, reporting, and graphing functions. Much research and practical design effort has been conducted in each of these fields (Shim et al., 2002).

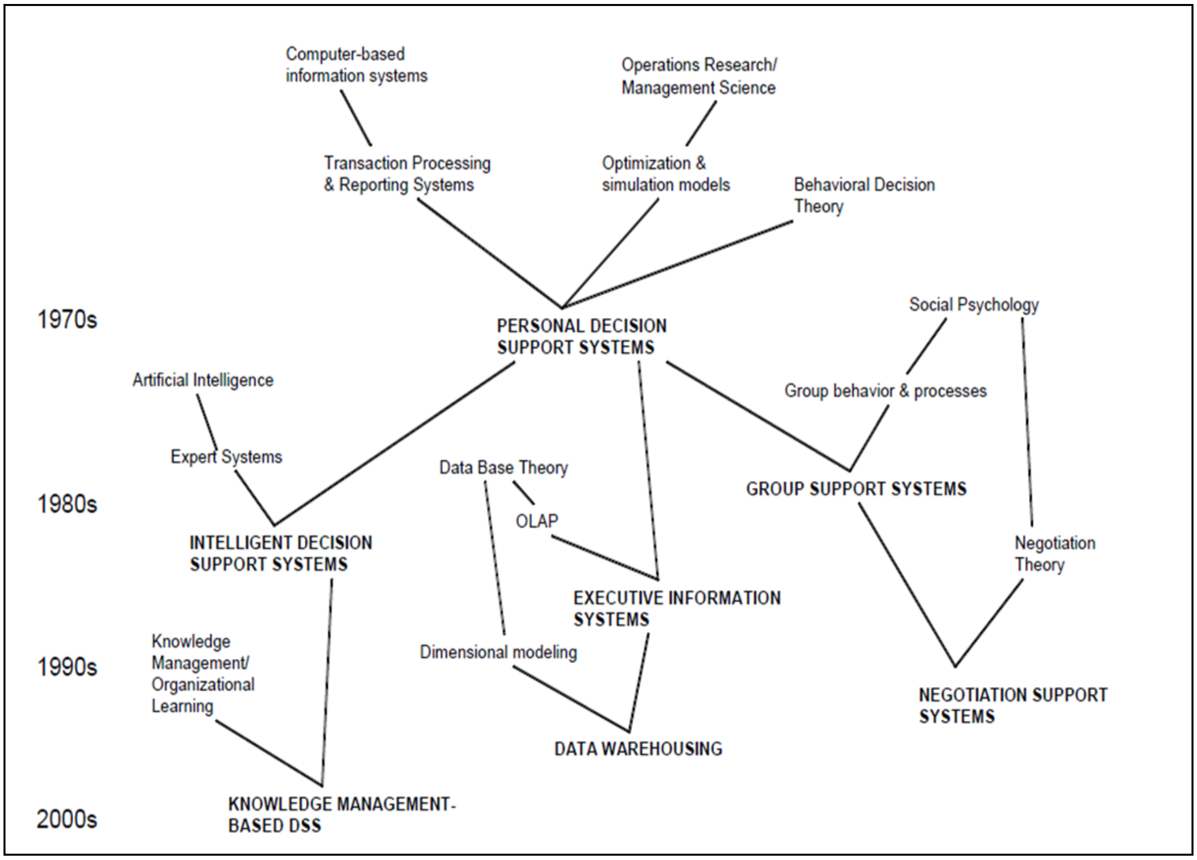

Computer-based information systems
Operations Research/ Management Science
Transaction Processing & Reporting Systems
Optimization & simulation models
Behavioral Decision Theory
1970s
PERSONAL DECISION SUPPORT SYSTEMS
Social Psychology
Artificial Intelligence
Group behavior & processes
Expert Systems
Data Base Theory
GROUP SUPPORT SYSTEMS
1980s
OLAP
INTELLIGENT DECISION SUPPORT SYSTEMS
Negotiation Theory
EXECUTIVE INFORMATION SYSTEMS
Knowledge Management/ Organizational Learning
Dimensional modeling
1990s
NEGOTIATION SUPPORT SYSTEMS
DATA WAREHOUSING
KNOWLEDGE MANAGEMENT-BASED DSS
2000s

Figure 2.3: The evolution of decision support systems (Arnott and Pervan, 2005)

Decision support systems are the hub of business IT infrastructures since they give companies a way to translate a wealth of business information into factual and productive results. It is a massive task to collect, to maintain, and to analyze large amounts of data, which presents significant technical challenges, expense, and organizational commitment. Before building a system that provides this decision support information, analysts must address and resolve three fundamental issues (Chaudhuri et al., 2001): (i) what data to gather and how to conceptually model the data and deal with their storage, (ii) how to analyze the data, and (iii) how to efficiently load data from several independent sources. Arnott and Pervan (2005) schematically show the evolution of decision support system research (Figure 2.3).

2.5: Analysis of business data

The need for analysing business data for trends and strategic information is not new. In a perfect world, the analyst could simply query the operational database in-company to determine the long-term effects of a policy or the fiscal inference of a marketing decision. In reality, few operational environments have the resources to allow information seekers to create the elaborate queries necessary for complicated data analysis. The data which are used for operational systems are physically separated from data used for analytical processing or for information. The supporting technology for operational processing is also different for both systems, so the users are also different (Jones, 1998).

Data in transaction applications are file-oriented and consist of linear lists of related values, each describing an employee, customer, product or other entity. These data are accessed with a single key and viewed as a simple list on the screen. End-user applications, in contrast, are best viewed in multidimensional format because the user wants to see various possible combinations of entities and their relationships, *e.g.*, performance of different products over time in different markets (Stamen, 1993). Toward this purpose, data from various operational sources are reconciled and stored in a repository database using a multidimensional data model. The multidimensional modeling is a conceptual modeling technique used by OLAP applications or data warehouses (Chaturvedi et al., 2008).

2.5.1: Online analytical processing (OLAP)

On-line analytical processing, introduced by Codd *et al*. (1993), is capable of capturing the structure of real world data in the form of multidimensional tables. OLAP tools are well-suited for complex data analysis such as multidimensional data analysis and to assist in decision-support activities (Chaudhuri and Dayal, 1997), informally referred to as end-user computing (Stamen, 1993). The data used for OLAP are obtained from a data warehouse, established independently of the current operational data, and can consist of data collected from heterogeneous, operational and legacy data sources

(Chaudhuri and Dayal, 1997). OLAP allows for the fast analysis of shared multidimensional information: most system responses occur within 5 seconds, with the simplest analysis taking no more than 1 second and a very few more complex queries taking more than 20 seconds. However, speeds vary, depending on the OLAP vendor and system hardware (Alkharouf et al., 2005). What the decision-maker needs, and what OLAP must provide, are the following functions (Codd *et al.*, 1993):

i. Access to the data in the database management system;
ii. Data and data consolidation paths or dimensions that can be defined according to user requirements;
iii. Accommodation of the variety of ways or different contexts in which the user may wish to view, manipulate and animate the data analyses; and
iv. Accessibility to these functions via the end-user's interface.

2.5.2: OLAP versus OLTP

Database technology is at the center of most information systems. Among these information systems are decision-support systems and executive information systems. OLTP environments use database technology to execute and query data, and support the daily operational needs of business executives. On the other hand, OLAP environments use database technology to support analysis and to provide decision-makers with a platform from which to generate decision-making information. Because the process by which data are extracted, transformed, and loaded into the OLAP environment can be relatively slow by transactional data standards, the ability to achieve "real-time" data analysis is lost. OLTP database structures are characterized by storage of "atomic" values, and they are also transaction-oriented as the name implies. Besides, OLAP database structures are generally aggregated and summarized. There is an analytical orientation to the nature of the data, and the values represent a historical view of the entity (Conn, 2005).

In the past, corporate data were usually stored in production databases: it is only recently that information specialists have realised the need to analyse the data and store these in a different form which can be better utilised for decision-making purposes. The work of Codd *et al.* (1993) has led to a distinction between analytic databases (ADBs), and operational or production databases (PDBs). A summary of these differences is presented by Thomsen (2002) which provides a clear explanation of the different functionalities between OLTP and OLAP. This difference can be illustrated with a simple example: consider a customer order processing application in which the production database stores information such as: the customer's identification number, the name of the customer, the customer's order reference number, the value and the date of the order, with links to other tables holding other related data e.g., quantity and quality of items ordered, mode of payment, date of dispatch, etc. A manager, however, needs to analyse the data and requires the resulting information in a more concise form, such as: how many orders were posted without a delay, or total number of orders per day, or total number of deliveries per day etc. It is much more important for the manager to have this information in a timely manner, especially to understand trends in customer behaviour. In traditional PDBs, such appropriate information is not available, because the production databases update information after every transaction. As a result, trends in customer behaviour cannot be identified, which, in turn, denies the business decision-maker from a valuable insight. If this information from the PDB is stored in an ADB, the time to analyse the data then become suitable for decision-making.

2.5.3: Multidimensional Data Modeling

Data warehousing provides a framework for integrating data from multiple sources in a logical and integrated fashion (Inmon, 1996). It provides users with easy access to enterprise data in a consistent manner and a method for storing historical, summarized and data aggregations for easy decision-making. Organizational data warehouses are projected to be hundreds of gigabytes or terabytes in size; therefore, a basic design and implementation issue of a data warehouse is query performance. In order to improve the query performance, the data in a data warehouse are typically modeled from a multidimensional perspective (Anil et al., 2008).

A multidimensional-data-model-based data warehouse for agriculture farm histories has been developed, and is helping experts to generate quality advice by providing crop-related information in an integrated manner (Reddy et al., 2007). The expert advice has helped farmers to achieve significant savings in capital investments and improvements in yield. The most important analysis capability that has emerged for executive information systems and for the data warehouse environment in general is multidimensional analysis. A 1996 survey (McFadden, 2002) indicated that 90% of CIOs claimed that their organizations were developing data warehouses. Of these, 65% said that using multidimensional analysis was a high priority in their organization. The basic feature of this model is that it allows the user to visualise data from different perspectives (Mohania et al., 1999). Multidimensional analysis allows end-users without extensive mathematical or statistical training to perform operations such as drill-down, roll-up, cross-tabulations, ratios and trends, slice and dice, and data pivoting (McFadden, 2002). The multidimensional aspect of enterprise data can be better viewed with OLAP software which enables analysts, managers, and executives to gain insight into an enterprise’s performance through fast interactive access to a wide variety of views of multidimensional-organized data (Colliat, 1996) and data analysis, through interactive querying of data. The multidimensional view of data conceives of information as stored in a multidimensional array sometimes called a hyper cube (Vassiliadis, 1998).

A cube is defined by any number of data dimensions; it is not limited to three; and sometimes a cube may have fewer than three dimensions. In relational database systems, cubes are constructed from a fact table and one or more dimension tables (Alkharouf et al., 2005). A fact table is the relational table in a data warehouse which contains numeric data items used to satisfy all calculation options that are of interest to the end user. The facts can be additive, semi-additive or non-additive (Anil et al., 2008). The dimension tables however are more abstract, containing only one row for each leaf member in the fact table. They are used to create summaries and aggregates of the data in the fact table. The data dimensions describe a cube just as width, height, and depth, where it is appropriate, dimensions can be organized into any number of levels or hierarchies (Alkharouf et al., 2005).

2.5.4: Analysis of data with a Multidimensional Data Model

The multidimensional data model is an integral part of the OLAP. Both on-line and analytical, the OLAP must provide quick answers to complex queries. Designed to solve complex queries in real time, the multidimensional data model is important because it enforces simplicity (Kimball, 1996). In contrast to previous technologies, these databases view data as multidimensional cubes that are particularly well-suited for data analysis (Pedersen and Jensen, 2002). To provide decision-support with this model, developers must decide how to structure data and what type of database management system has to be used in implementation of the data warehouse or data marts. A developer can follow three strategies (McFadden, 2002):

i) **Use an RDBMS server**. Organize the data in the form of relations (either normalized or de-normalized). A client interface which can interact with an RDBMS can provide multidimensional viewing and analysis of these data.

ii) **Use a Relational OLAP server**. Organize the data in the form of a "star" structure. Client interfaces which can interact with the star structure provide multidimensional viewing and analysis of the data. Data warehouses can be implemented on standard or extended relational DBMSs, called Relational OLAP (ROLAP) servers. These servers assume that data are stored in relational databases, and support extensions to SQL and special access and implementation methods to efficiently implement the multidimensional data model and operations (Chaudhuri and Dayal, 1997).

iii) **Use a multidimensional database (MDB) server**. Organize the data physically in the form of multidimensional arrays. Provide multidimensional viewing and analysis through client interfaces that interface with the multidimensional database (Chaudhuri and Dayal, 1997; Pedersen and Jensen, 2002). This is also known as multidimensional online analytical processing (MOLAP).

The first option is unsuitable for multidimensional analysis for two reasons: performance and reliability. Since data in tabular form are not pre-processed, query responses needs to be computed dynamically which requires not only resources but also considerable time (Halawani *et al.*, 2010). Also, inconsistent responses may be generated when an attempt is made to analyze multidimensional data against a relational structure. The second option, in which the developer can use the star data structure, means that data are stored in a relational format that is specially designed for multidimensional analysis. Two types of tables are used: fact tables, and dimension tables (McFadden, 2002). The third option is to use a multidimensional database (MDB) server or MOLAP servers. With an MDB, data are pre-processed and stored in the form of arrays for fast and flexible retrieval in multidimensional analysis. In this way, it is possible to implement front-end multidimensional queries on the storage layer through direct mapping (Chaudhuri and Dayal, 1997).

The most essential performance-improving techniques in multidimensional databases are pre-computation and pre-aggregation, which enable fast, interactive data analysis for potentially large amounts of data. As an example, computing and storing, or materializing total crop production by region and by year quarter is one application of pre-aggregation. These answers can be derived entirely from the pre-computed results without needing to access large bulks of data in the data warehouse. This method can save time and resources. The latest versions of commercial relational database products, as well as dedicated multidimensional systems, offer query optimization based on pre-computed aggregates and automatic maintenance of stored aggregates during updating of the base data (Winter, 1998).

2.5.5: Star schema

In ROLAP architecture, data are organized in a star schema, the standard schema for dimensional modeling, which is divided into two main structures (Correa et al., 2009): (i) the additive numerical data which is being stored in a unique table or fact table (Abdullah, 2009); and (ii) dimension which is a set of similar entities, e.g., set of all employees or all products. The fact table is linked to all the dimension tables by one-to-many relationships (Stamen, 1993). This is illustrated in Figure 2.4. Abdullah (2009) defined dimension as a business parameter that defines a transaction (or record) and usually consists of hierarchies. Each dimension in a data warehouse has one or more defined hierarchies, *e.g.*, the weather dimension has multiple levels of possible hierarchies like weather > humidity > humidity at 8:00 AM and weather > temperature > minimum temperature.

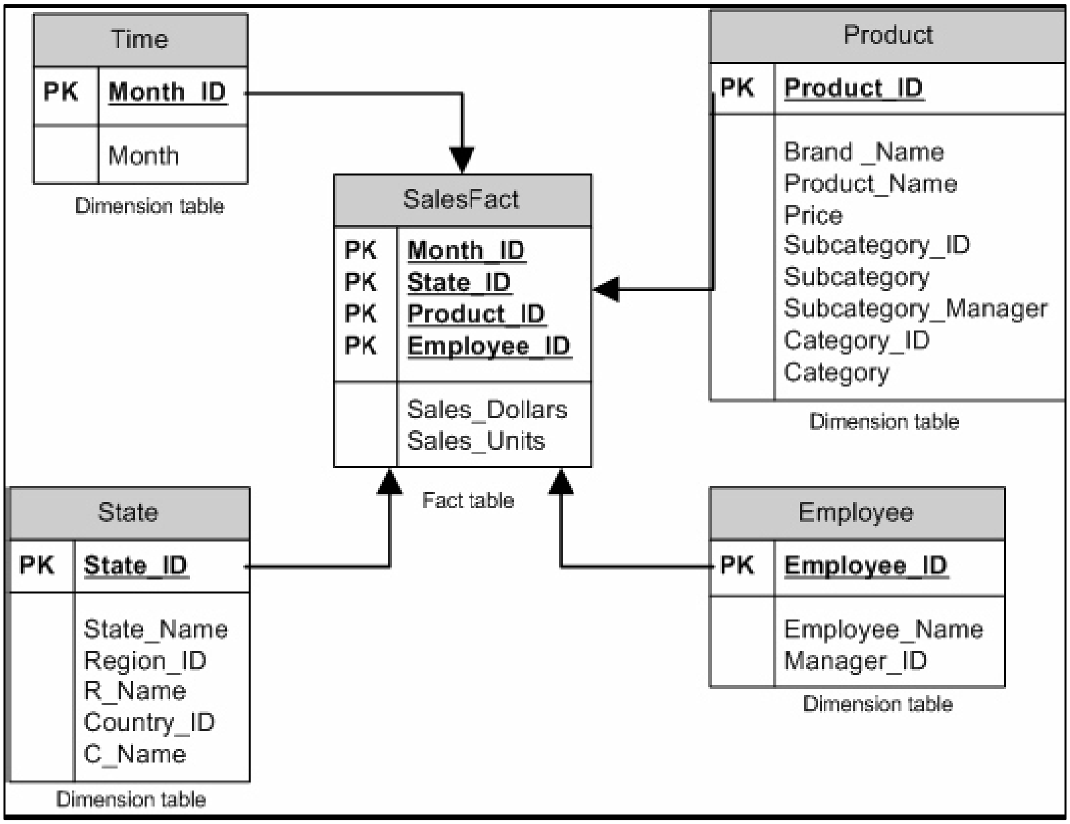


Figure 2.4: Star schema for sales subsystem (data model) (Mishra et al., 2008).

2.5.6: Snowflake schema

The snowflake schema can be used in those cases where many-to-many relationships exist among the data of a dimension table and a fact table (Ballard et al., 1998). In case of the snowflake schema, all dimensional information is stored in the third normal form, while maintaining the same fact table structure. To protect the hierarchy, the dimension tables are linked with sub-dimension tables using many-to-one relationships. This accounts for why this model generates a modification of the star model where the dimensional hierarchy is clearly characterized by normalizing the dimension tables

(Ahmad et al., 2004). A snowflake schema for decision-support to assist builders is illustrated in Figure 2.5.

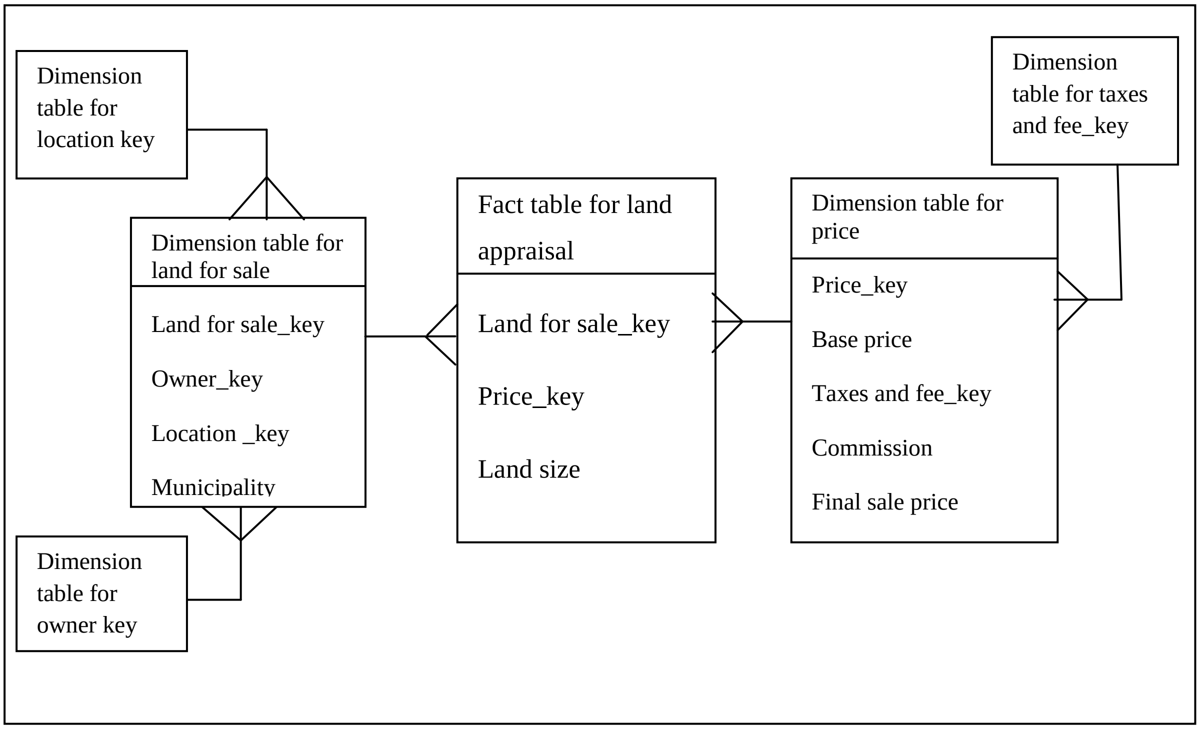

Dimension table for location key
Dimension table for taxes and fee_key
Fact table for land appraisal
Land for sale_key
Price_key
Land size
Dimension table for price
Price_key
Base price
Taxes and fee_key
Commission
Final sale price
Dimension table for land for sale
Land for sale_key
Owner_key
Location _key
Municipality
Dimension table for owner key

Figure 2.5: Snowflake schema for decision support system to assist builders (data model) (Ahmad et al., 2004).

2.5.7: Fact Constellation schema

Normally it is impossible for all measures and dimensions to be captured in a single model. Usually, a data warehouse consists of several fact tables explained by several (shared or non- shared) dimensions (Eder et al., 2006). A fact constellation schema has more complex structures in which multiple fact tables share dimensional tables (Chaudhuri and Dayal, 1997). For example, a sale item and the sale fee may form a fact constellation since they share many dimensions (Figure 2.6).

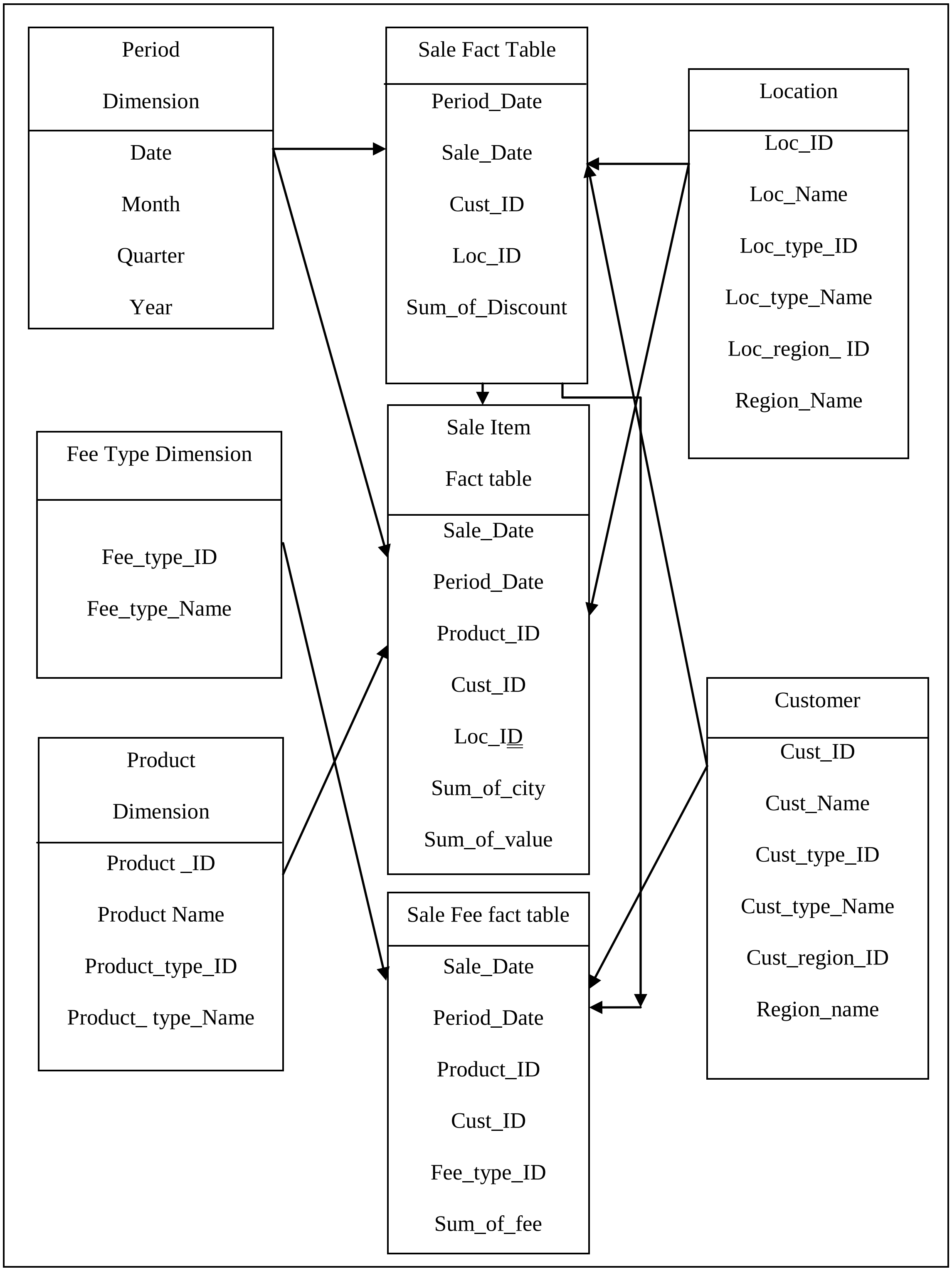


Figure 2.6: Fact Constellation Schema for Sales subsystem (data model) (Mishra et al., 2008).

2.5.8: Comparison of Logical Design Models

Efficiency is the most important factor in data warehouse modeling because many queries access large quantities of data that may possibly engage multiple join operations (Martyn, 2004). A star schema is usually the most proficient design for two reasons: (i) a design with denormalized data needs fewer joins between tables; and (ii) most software have star schema and can make efficient “star join” operations. In fact, constellation schema may require more join operations on fact tables. While a snowflake schema needs additional joins on dimension tables, in specific situations where the denormalized dimension tables in the star schema become extensively large, it may be the most adequate design methodology.

Although the star schema is the simplest structure among the three schemas, it has the smallest number of tables, and users need to execute fewer join operations which makes it easier to create analytical queries. It is easier to learn the star schema than either of the other two schemas. Compared to the star and fact constellation schemas, the snowflake schema can share dimension tables and can be reused in a data warehouse. Dimension tables in a snowflake schema do not contain denormalized data. This makes dimension tables in the snowflake schema more reusable in a data warehouse. In star and fact constellation schemas design approaches, dimension tables are denormalized, making it less suitable to share dimension tables among schemas. Other advantages of the snowflake schema include storage of data in normalized tables, which reduces redundancy and results in fewer data inconsistency problems arising (Mishra et al., 2008).

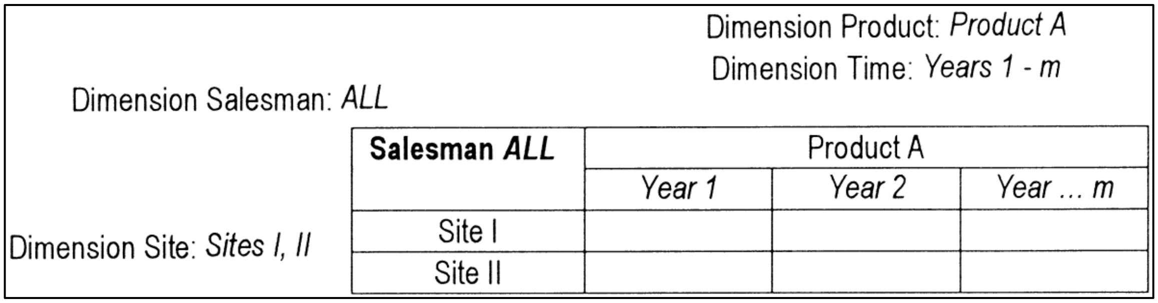
2.5.9: Design of a data warehouse

The design of a data warehouse consists of three basic steps: (i) identifying facts and dimensions; (ii) designing fact and dimension tables; and (iii) designing data warehouse schemas. Identification of facts and dimensions include the following procedures: (i) facts represent quantitative data about a business transaction (*e.g.*, land available for sale, price of land etc.) while dimensions reflect description of that fact (land owner, location etc.), (ii) design fact and dimension tables include the procedures to define primary keys in the dimension tables and introduce related foreign keys in the fact table. As a result a fact table contains facts and keys foreign to the dimension tables. In a query, the system first accesses dimension tables and then the fact table. Designing the data warehouse schema includes the process of establishing the relationships between fact and dimension tables. The resultant schema will contain a central fact table and surrounding dimension tables. The three main types of data warehouse design schemas are star, snowflake and fact constellation (Ahmad et al., 2004).

2.5.10: Accessing data from the data warehouse for analysis

The main reason for using multidimensional data models for business data analysis is spread sheet programs. These are still a popular front end tool for OLAP. But the main thing in using a spread sheet as a front end tool is that either it supports the queries of OLAP or not, as OLAP queries are complex and summarize the data coarsely. The most accepted operation of spread sheet is “pivoting” (Chaudhuri and Dayal, 1997). The pivot operator transposes a spread sheet by aggregating values in the cells of a spread sheet. If pivot is created on two columns containing N and M values, the resulting table would be N*M values (Gray et al., 1997).

The simplest view of a pivot is that it selects two dimensions that are used to aggregate a measure. In Figure 2.7, for all salesmen, the sale of product A is aggregated for years one, two and three in dimension for site one and two (Koutsoukis et al., 1999). Other operators associated with pivot table are roll-up or drill down. As in Figure 2.7 the pivot table is rolled up to all salesmen. The drill down operation is the converse of roll-up, i.e., it serves to explore data for an individual salesman. Slice and dice corresponds to reducing the dimensionality of the data by taking a projection of the data on a subset of dimensions (Chaudhuri and Dayal, 1997). In Figure 2.7, the pivot table (product A) is actually a slice of the large pivot table of all the products (A, B, ... , Z).



Dimension Product: *Product A*
Dimension Time: *Years 1 - m*
Dimension Salesman: *ALL*
Dimension Site: *Sites I, II*

Salesman *ALL*	Product A		
	Year 1	*Year 2*	*Year … m*
Site I			
Site II			

Figure 2.7: Pivot table (Koutsoukis et al., 1999)

Since pivot tables allow for the nesting of multiple dimensions within the same axis, they are adequate for displaying the query results in a straightforward fashion. However, they fail in showing selected values in a larger context and are thus a rather poor option for complex data exploration. Advanced OLAP tools overcome these limitations through visual alternatives for retrieving, displaying, and interactively exploring the data (Vinnik and Mansmann, 2006). Vassiliadis (1998) classifies OLAP cube operations as level-climbing, packing, function-application, projection, dicing, as well as more complex operations, such as navigation and slicing, which are defined on top of the simple ones. He (Vassiliadis, 1998) does not use pivoting in this classification since he claims that pivoting is simply a reorganization of the presentation of the data, rather than a modification of their value or structure. When applying these functions to the existing cube, each operation resulted in a new cube. Aggregate functions were available through slicing and navigation.

2.6: Application of data warehousing

Comprehensive Assessment for Tracking Community Health (CATCH) provides methods for community-level assessment that are very useful in resource allocation and health care strategy formulation. A community-level focus is proposed to support local decision-makers by providing a clear methodology for organizing and understanding relevant health care data. Data warehousing technology, has led to an innovative application of information technology in the health care arena along with extensive field experience with CATCH methods. The data warehouse allows a core set of reports to be produced at a reasonable cost for community use. In addition, OLAP functionality can be used to gain a deeper understanding of specific health care issues (Berndt, 2003).

Data warehousing helps organizations in analyzing patient populations by geographic location, diagnosis, and service consumption to determine which disease management programs will be most beneficial to the patient and the organization. The purpose of this plan is to improve member health status, lower medical expenses and increase physician participation. Implementation of disease management programs allow providers and organizations to actively manage patient care and inform patients so they can monitor their own condition, which will effect a decreased use of high-cost services. The organization should be financially successful, and the outcome is a healthier patient. Disease management efforts which are well developed may even lower disease occurrence in people (Ramick, 2001).

In the field of education, the goal of the data warehousing project at Stanford University was to develop algorithms and tools for the efficient collection and integration of information from heterogeneous and self-governing sources, including legacy sources. The warehousing approach was particularly useful when high query performance is desired, or when information sources were expensive or transitory (Hammer et al., 1995).

Chaudhary *et al.* (2004) proposed a data warehouse architecture which had the potential to satisfy information requirements for effective decision-making in the agricultural domain. Challenges related to administering the collection of a vast amount of transactional data, their processing, analysis and display were addressed by three capable technologies: sensor web enablement, web services and OLAP. Analysis services provided support to build OLAP cubes of varying dimensions. With the support of Multidimensional Query [MDX] technology, complex OLAP analysis operations like slice, dice, and drilldown, were possible on a simple web browser. This capability has the scope to help farmers and policy-makers to make operational and strategic decisions.

The value is added to data when it can be used as information to improve the decision-making capability of individuals. Valuable data are collected and reported, but difficulty in finding, accessing, and processing the information lessens its value to agricultural decision-makers. A data warehouse structure provides the additional features of data management, query, and decision-support systems, thus moving users towards the specificity in data that they desire. This structure would allow decision-makers to more resourcefully access and process the assets of available information, thus reducing exploration and overall transaction costs in decision-making (Thornsbury et al., 2003).

The agricultural advisory system called eSagu (a data warehouse), has been developed to improve the performance and utilization of agriculture technology and help Indian farmers. With the help of this data warehouse, the agricultural expert delivers expert advice at regular (weekly) intervals to each farm by getting the crop status in the form of digital photographs and other information, rather than visiting the crop in person. During 2004-06, agricultural experts' advice was delivered to about 6000 farms growing six crops with the help of this data warehouse. The results show that the experts' advice helped the farmers to achieve savings in capital investment and improved crop yield. The data warehouse was based on farm histories providing crop-related information to the agricultural experts in an integrated manner, allowing them to generate quality agricultural advice (Reddy et al., 2007).

In another example, a data warehouse based on multidimensional data models was used for simulation purposes. Urbanization, climate change and deforestation are complex situations and a lot of data is collected on these dynamics. Consequently, these dynamics may be better viewed with the help of multidimensional data models, to identify and predict the evolution of the environment in response to potential value changes in a large number of influence variables. Data warehousing systems provide tools for supervision of simulation results originating from different sources. Besides, OLAP technologies allow analysis and evaluation of these results and their resultant models. The users can use this methodology to design specific data warehouses, and an adaptation of an OLAP client tool to provide an adequate visualization of data (Mahboubi et al., 2010).

It is the objective of this study to consider the various sources available for improving the decision-making process in the Québec dairy industry, demonstrate the process of acquisition, cleaning and de-normalization, and explore the methods for modelling these multidimensional data. Their storage in a data warehouse, as well as their viewing through an appropriate schema, should facilitate the analytical process of querying these data for the benefit of producers and their advisors in the dairy industry.

Chapter 3: Materials and Methods

3.1: Data extraction and cleaning

The Online Transaction Processing (OLTP) Systems at Valacta are the basic element in the IT-infrastructure of Quebec Dairy Industry. These are the common data storage used by all producers enrolled in the dairy milk recording system. They represent the point of data entry for each cow, and provide access to the data collected by other systems, e.g., Breed Associations and the Canadian Dairy Network. In spite of these characteristics, the databases at Valacta cannot be considered as data warehouses *per se*. Data in these systems are used and organized according to operational purposes, where many kinds of data about one cow are presented to get an overview of the milk components, feed information, breeding dates, health status, etc. On the other hand data in a data warehouse would be stored with respect a specific subject like milk production for a herd in a specific year in a specific season in a specific region. Therefore a data warehouse needs historical data by which trends can be analyzed. Thus extraction of historical data from the operational databases at Valacta was an important component of this research. All of the files which were extracted from the operational databases were in the form of SAS (Statistical analysis system) datasets, which were denormalized. The number of test day records, herds and cows are given in the following table which were based on years 2000 to 2009 (Table 3.1).

Name of attribute	Number of records
Herds	6,917
Cows	1,203,134
Test day records	25,398,435

Table 3.1: Record details

The test day records were extracted in four SAS datasets. Among the four, the dataset which was selected for study had 8,295,154 test day records, 1,943 herds and 447,414 animals. A typical test day record consist of information such as unique herd identification number (hrd_id), unique cow identification number (anm_id), unique herd test period identification number (htp_id), parity, milk in kilograms, percentage of fat, percentage of protein, percentage of lactose, MUN (milk urea nitrogen (mg/dl), SCC (somatic cell count(*1000)), and days in milk (DIM). An anm_id and hrd_id also pointed to the records in other files, e.g., lactation file, feed file, breeding file, etc. A change in the status of the cow, e.g., shift from one lactation to the next lactation, change in weight gain, change in health status, was present in these files. Thus, once a record is entered, it is conserved to see the life time history of a cow. Therefore "time" at which the data are included in the operational data stores is very important to see the history of a cow, which accounts for why there are gigabytes of data stored in operational data stores every year.

A step by step process of warehousing data is shown in Figure 3.1. The SAS datasets were studied carefully for type of variables, count of different variables, missing values, and minimum and maximum values. Then data were "cleaned" (missing values and outliers) according to some accepted rules, and classified by using SAS univariate and FREQ procedures (Figure 3.2).

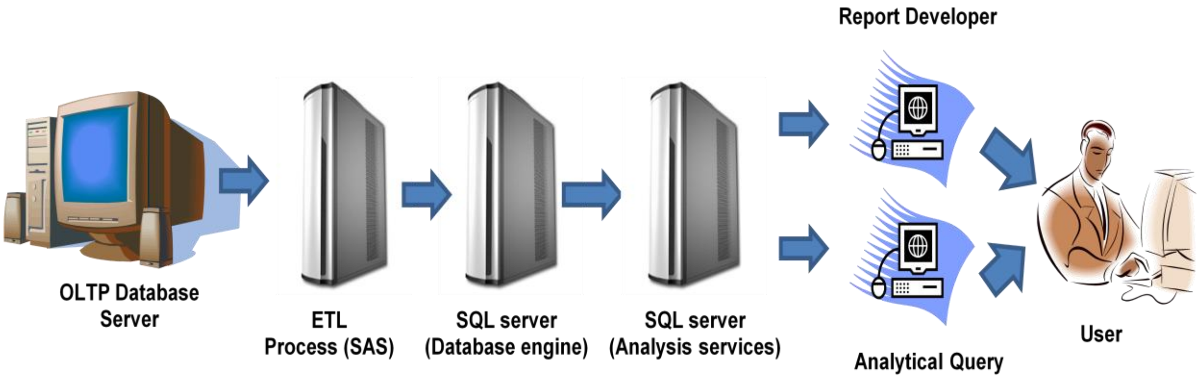


Figure 3.1: Process of data warehousing.

After the removal of outliers with ± 4 standard deviation, 7,911,659 records remained. A detailed study was conducted on their attributes with an emphasis on their data type and length, since in operational databases; catalogues contain fundamental data-type information such as whether an attribute is real, string, integer or a date. From this information of fundamental data types certain properties were useful to determine which attributes could be derived from other attributes. Attributes were also classified as to whether they had continuous or discrete values. This information was very significant with respect to the selection of dimensions and facts for the multidimensional data model. Data were denormalized so that attributes having null values were also considered, since in dimensional data modeling, null values were tackled carefully to avoid wrong calculations.

```
/*Procedure univariate to see the distribution of milk quantity and fat percentage in the data*/
proc univariate data= parity1;
var fat;
HISTOGRAM fat/NORMAL (COLOR=RED W=5);
RUN;

proc univariate data = chk_lact;

var milk305;
by parity;
HISTOGRAM  milk305/NORMAL (COLOR=RED W=5);
RUN:
```

Figure 3.2: SAS univariate procedure to see the distribution of fat and milk in the data.

3.2: Data transformation

Data transformation actually encompasses all those features which transform data from operational databases to a data warehouse. Different required attributes were calculated from the existing attributes, where necessary and data were carefully inspected to determine if the output was reasonable or not. In this way values could be easily distinguished between the extremes of reasonable and unreasonable. This type of analysis dealt with obvious cases of data inaccuracy. An example of the code is given in the Figure 3.3 in which different new columns were created from the existing attributes.

```
create view feed_dm1 as Select hrd_id,anm_id,htp_id,qty,ab,FDC3, descrp,DM,nel,cp,
(Case when (FDC1='7' or  FDC1='8' )
then qty/1000 else qty  end) as qty_kg from feed_daytest;
```

Figure 3.3: Calculation of the new columns from the existing columns

```
/* Classification of data for parity groups*/
data a;
set MkCom.test_day_scc;
if parity = 1 then parity_gp="pgp_1";
if parity =2 then parity_gp="pgp_2";
if parity>= 3 then parity_gp="pgp_3";
run;
```

Figure 3.4: Classification of the data for defining dimensional hierarchies

In the data, variables were identified for defining dimensions (dimension table: which give the perspective for the measure) and measures (fact table: the objects of analysis) for the development of a multidimensional data model. The primary key and foreign key relationships were studied to see how one attribute relates to other attributes in other files. For the definition of the dimension structures data were broken into separate dimensions, such as a dimension for herd, for cow, for parity groups (parity group 1 in which first parity cows were present, parity group 2 in which second parity cows were present and parity group 3 in which third and above parity cows were present (Figure 3.4), dimension for season of calving, etc.

What the granularity for each dimension would be was also determined. Like in case of test day granularity was considered each test date. In the case of region, breeds, herds and cows, region and herds were kept in the one dimension table and breed with cows in one dimension table. The level of granularity for region was to the herd, and to the cow for the breed table. The reason behind keeping herds and cows in separate dimension tables was the hierarchical characteristic, since it was a classic ragged or non-covering hierarchy. A ragged hierarchy or non- covering hierarchy is a hierarchy in which children do not have parents at the same next level. In other words, not all leaf level (or parent level) entries have the same depth (Not all cows have information on all parities, similarly not all herds have the same number of cows). After designing the multidimensional models on paper, it was decided to transform them to software, so data from SAS were loaded to SQL server management studio 2008 with the help of the SQL server native client for the ODBC driver[1] (Figure 3.5).

```
libname sqlsvr odbc datasrc="myodbc" user= password=;
Data sqlsvr.daytest;
set MkCom.test1;
run;
```

Figure 3.5: Connection of SAS with SQL server for data loading

After data were in the SQL server, views were created on the data or the required tables were created to transform the data into a multidimensional model (Figure 3.6, 3.7 and 3.8).

```
/* daytest the fact table */
create table daytest (hrd_id float,anm_id float,parity float,test_date date,
milk float,fat float, protein float,mun float,lactose float, scc float
Primary key (hrd_id,anm_id,parity,test_date));

insert into daytest (hrd_id,anm_id,parity,test_date,milk,fat,protein,mun,lactose,scc)
select hrd_id,anm_id,parity,test_date,milk,fat,protein,mun,lactose,scc from daytest;
create view dim_con_rec as select * from testday;
```

Figure 3.6: Fact table

[1] ODBC is a standard definition of an application programming interface (API) used to access data in relational databases. SQL Server supports ODBC, via the SQL Server Native Client ODBC driver to communicate with the SQL Server.

```
/* Create dimension for date of calving*/
create table dim_doc_daytest (doc Date, season_doc varchar(max));
insert into dim_doc_daytest(doc,season_doc) select distinct doc,(case when
month(doc)>9 then 'Fall'
when month(doc)>3 and month(doc)<=6 then 'Spring'
when month(doc)>6 then 'Summer'
else 'Winter' end) as season_doc from daytest;

/* Creating Dimensions and fact table for test day file*/


/*Create cow dimension*/
create view dim_anm_daytest as
select distinct anm_id from daytest;
/*Create herd and cow bridge dimension*/
create view dim_hrdanm_daytest as
select distinct hrd_id,anm_id from daytest;
/*Create herd dimension*/
create view dim_hrd_daytest as
select distinct hrd_id from daytest;
/* Create dimension for breed*/
create VIEW dis_anb_test AS
select distinct anb_cd from hrd_anm_test;
/* Create diemsion for region*/
CREATE VIEW DIS_REGION_TEST AS
select distinct region from hrd_anm_test;
/* Create diemsion for parity*/
create table dim_parity_daytest (parity float, parity_gp varchar(MAX));
insert into dim_parity_daytest (parity,parity_gp) Select distinct parity,(case when
parity=1 then 'First_Parity'
when parity=2 then 'Second_Parity' else 'Thrid_Parity+' end) as parity_gp from daytest;
```

Figure 3.7: Creation of dimension tables

```
/* Create dimension for condition record (disease information)*/
create table dim_car_daytest (car_1 float , record_condition varchar (max));
insert into dim_car_daytest (car_1, record_condition)
select distinct car_1, (case when car_1=1 then 'Sick' when car_1=2 then 'Bloat' when car_1=
3 then 'Dysentry'
when car_1=4 then 'off_feed' when car_1=5 then 'Ketosis' when car_1=6 then 'Peritonitis'
when car_1=7 then 'Mastitis'
when car_1=8 then 'Sick_lame' when car_1=9 then 'other_helth_problem' when car_1=10
then 'Udeer_injury'
when car_1=11 then 'other_injury' when car_1=12 then 'Milk_fever' when car_1=13 then
'Metritis' when car_1=14 then 'Displaced_abomasum'
when car_1=15 then 'Et_flush' when car_1=16 then 'In_heat' when car_1= 17 then 'Nervous'
when car_1=18 then 'Aborted'
when car_1=19 then 'Oxcytocine' when car_1=20 then 'Rectal_palpation' when car_1=21
then 'Growth_hormone'
when car_1=22 then 'Fever' else 'no_record' end) as record_condition from daytest;
```

Figure 3.8: Creation of dimension table.

Microsoft SQL Server Analysis services provides a graphical user interface to communicate with the SQL server through standard API OLEDB (Object Linking and Embedding, Database). With the help of this API, data source views can be created on the tables and views, present in SQL server database engine. The facilities provided by Microsoft in analysis services are actually an interface for online analytical processing (OLAP). OLAP (Codd et al., 1993) is an area of active business and research interest and these tools focus on providing prompt response to *ad hoc* queries that summarize the warehouse data. With the help of this quick response by OLAP tools, clients can analyse the data and make informed decisions. OLAP applications are not supported by entity relationship and relational models, therefore those models which support multidimensional view of data become known. The models which support a multidimensional view of data have measurable facts like milk production, percentage of fat, percentage of protein, MUN (mg/dl), percentage of lactose and SSC on a specific test day and dimensions which characterize facts such as herd (in a hierarchy of herd and cow), date of calving (in a hierarchy of seasons, *i.e.*, winter, spring, summer and fall), time for test day (in a hierarchy of year, month and date), parity (parity group 1, 2, and 3+), region (there were seventeen regions in the test day file), and breed (there were eight groups for breed code).

3.3: Data storage in a warehouse with a multidimensional schema

The development of multidimensional data models of these data was quite complex since missing data as well as attributes in dimensions which had different levels of granularity had to be accommodated. As discussed earlier, several approaches exist in the literature, e.g., separate star schema for each path (Bauer et al., 2000), by separate tables for each level of hierarchies based on different paths (Jagadish et al., 1999), and using null values for absent levels of an attribute (Lehner et al., 1998). In this dimensional model, separate dimension tables were created for herds and cows (snowflaking), and null values were used for the absent level of an attribute. Many-to-many relationships between facts and dimensions were also captured by using a snowflake schema. For performance and study, star and constellation schemas were also considered. The details of these schemas are presented in the results and discussion section. There was another issue of using the same time dimension for test day and calving in a year. Use of same dimension of date for test date (in a hierarchy of year, month and date) and calving interval (ideally calving interval is 305 days lactation length and 60 days dry period) was impractical. Using the same dimension of time degrades the quality of the data – arguably one of the most important potential erroneous conclusions based on following OLAP queries. The hierarchies in dimensions (Figure 3.9 are actually used for drilling down and rolling up the aggregated data, so that test date is rolled up to month and then month is rolled up to year, but the calving in a year was based on season of calving. Figure 3.9 depicts the dimensional hierarchies in the case of test day, calving interval, herd, and herd-cow bridge dimension.

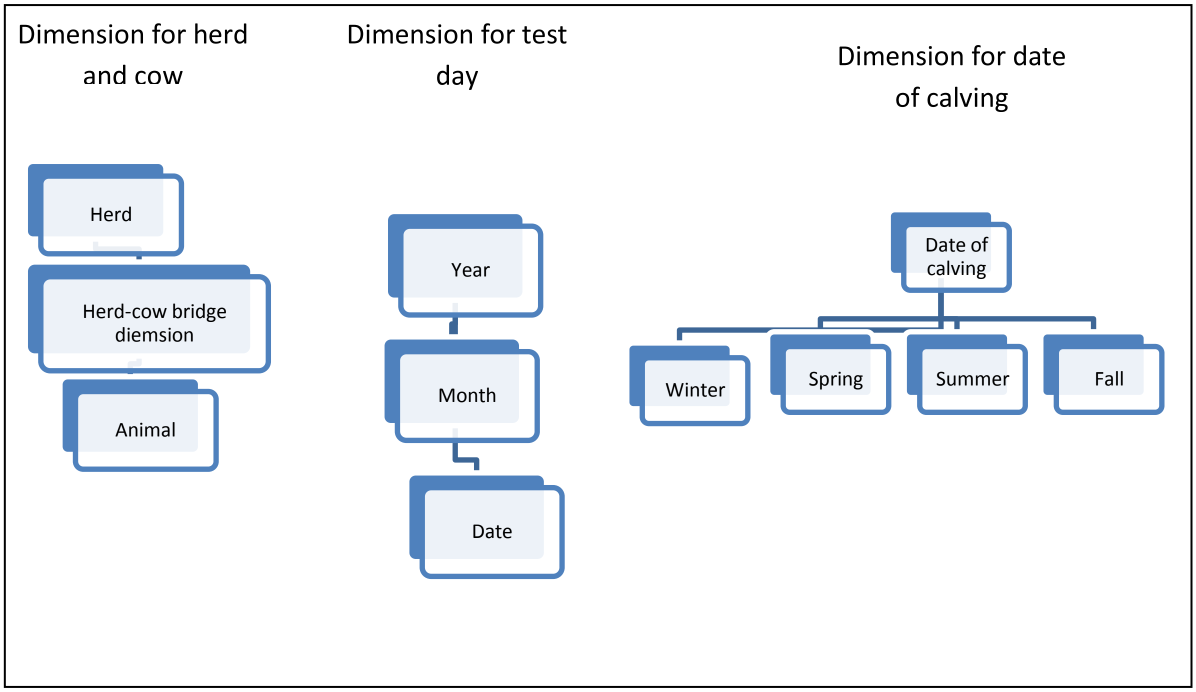
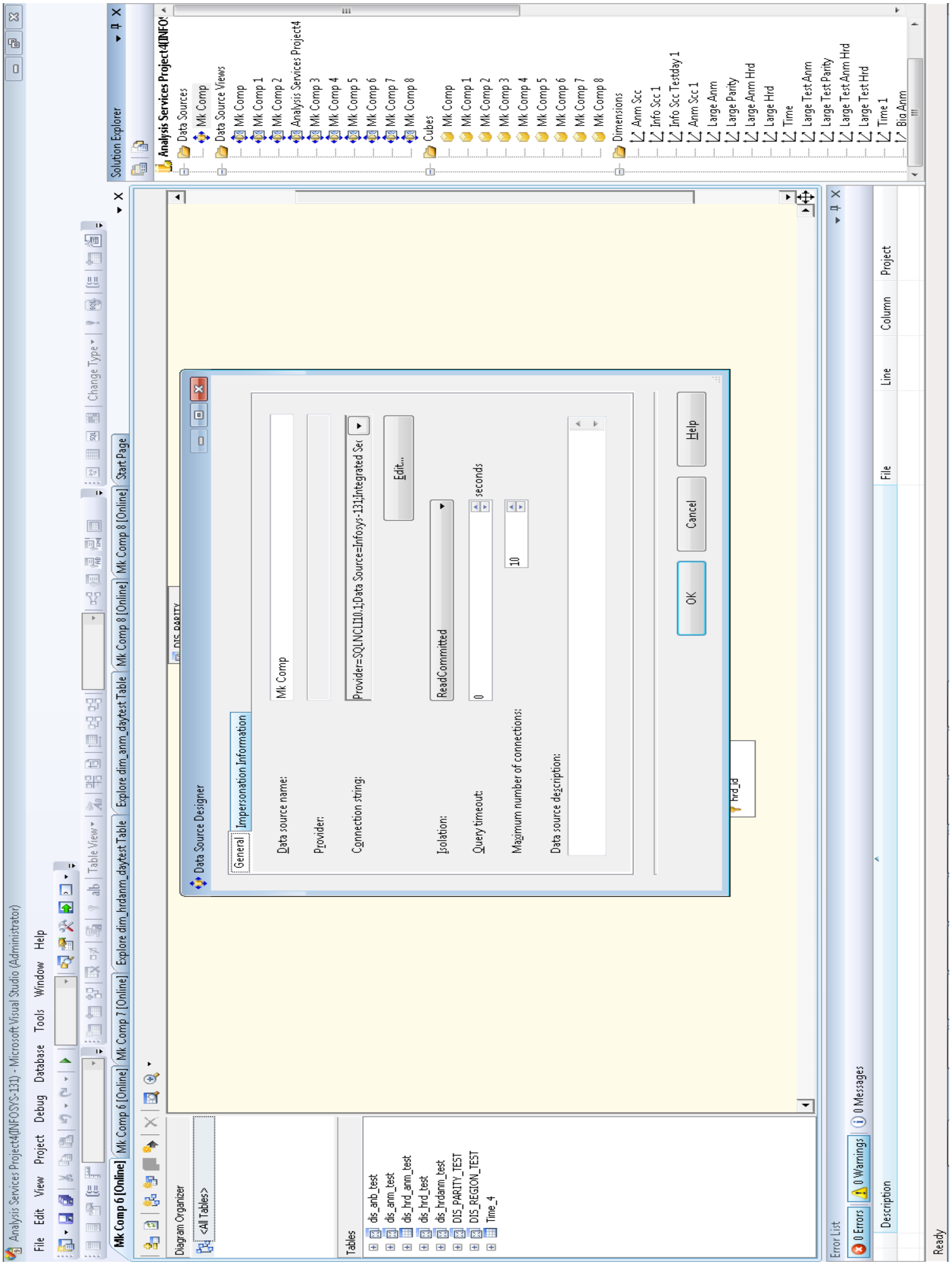


Figure 3.9: Dimensional hierarchies in multidimensional data model.

For the creation and deployment of multidimensional (OLAP) cubes, data were loaded into Microsoft Visual Studio 2008 (Analysis services) with the help of the SQL server Native Client driver for OLEDB. To create a new Analysis Services project, in the environment of Visual Studio 2008, the Analysis Services Project template was selected. A new folder was created for this project by default in the directory of My Documents in Windows 7.[2] After creating this new project a data source was created which was a data source object, represented by a connection to the data source from which data were imported. The connection was established with Native OLEDB/SQL Server Native Client 10.0. Through this native client of the SQL server, a connection was established between analysis services and the SQL server database engine, after entering the name of the server and other authentication credentials. This data source wizard had options of different databases presented in a list, so the desired database was selected from the list (Figure 3.10).

[2] Operating system on the machine was Windows 7



Data Source Designer

General | Impersonation Information

Data source name: Mk Comp

Provider:

Connection string: Provider=SQLNCLI10.1;Data Source=Infosys-131;Integrated Sec

Isolation: ReadCommitted

Query timeout: 0 seconds

Maximum number of connections: 10

Data source description:

Figure 3.10: Analysis services data source

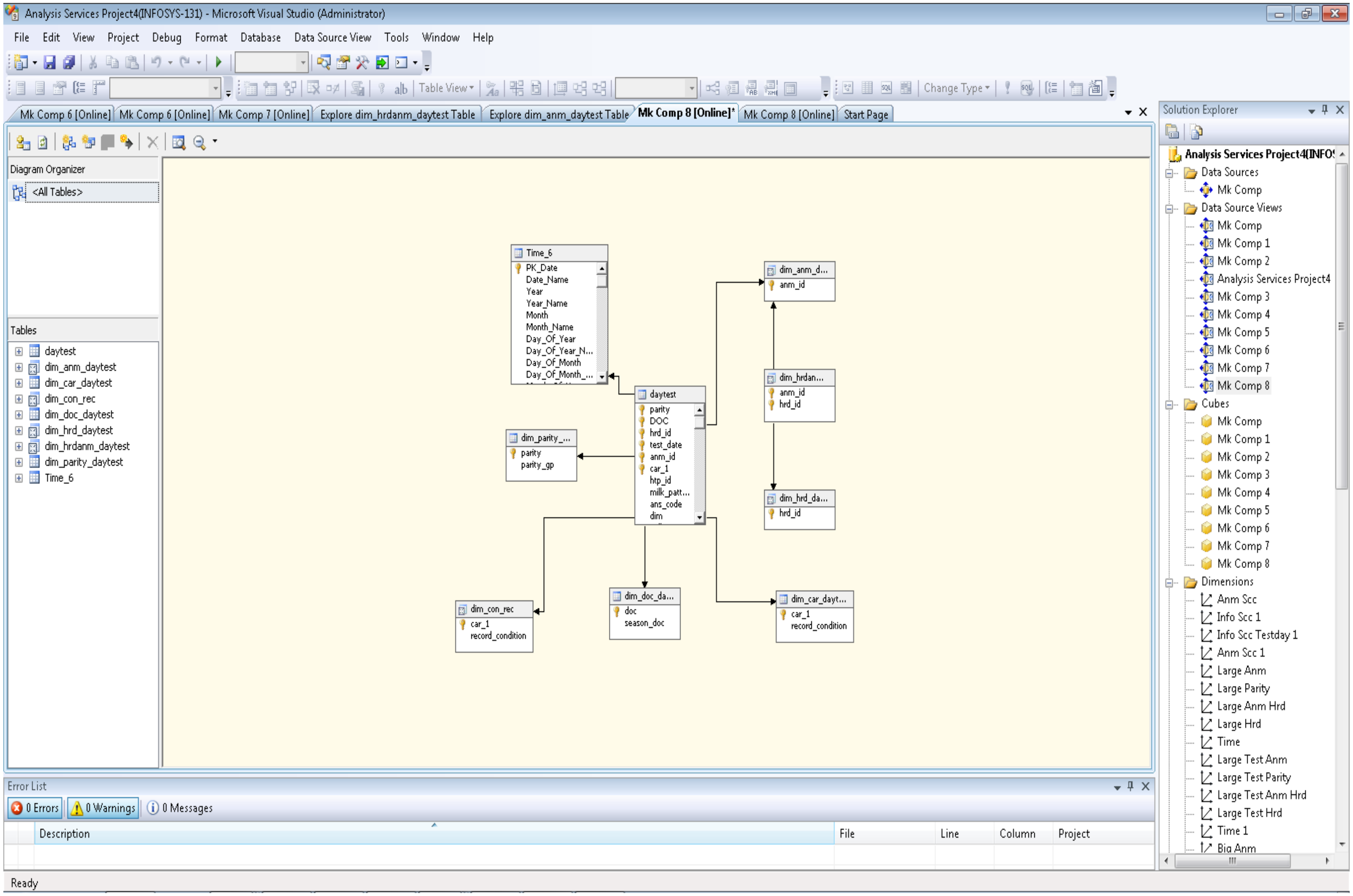


Figure 3.11: Analysis services data source view

This object was used to create data source view. “A data source view contains the logical model of the schema used by Analysis Services database objects namely cubes and dimensions" (http://msdn.microsoft.com). A data source view (Figure 3.11) had metadata that were used to generate an underlying relational data store. It also had relationships; primary keys, calculated columns, and queries that were not present in an underlying data source and which were separate from the underlying data sources, *i.e.*, if there were any changes made to the data source view, there was no change in the original tables in the SQL server.

After the definition of a data source view, the cube was ready to be build. The cube was defined with the help of the cube wizard, thereby establishing the primary foreign key relation between dimensions and fact. However, in the case of bridge dimensions (many-to-many relationship between dimension table and fact table) dimensions were defined using the dimension wizard, processed, and then added to the cube (Figure 3.12). The fundamental components of a cube are dimensions and measures. Dimensions are called categorical attributes and measures are called numerical or summary attributes. According to Agrawal *et al.* (1997) there is no formal way of deciding which attributes must be made dimensions and which attributes must be made measures; it is left as a database-design decision.

In order to navigate the data, dimensions explained the organization of the cube that was used to slice and dice, and measures provide aggregated numerical values of interest to the client. At the junction of dimension members, a cell in the cube is defined as containing the aggregated values of the measures at that specific junction; thus cells are defined for every possible summarized value. As a coherent arrangement, a cube permits a client request to extract values or measures, as if they were contained in cells in the cube. In the Cube designer window, the cube structure (Figure 3.13) tab gives one the opportunity to deal with measure properties and to define new measures and edit existing measures. Because analysis services does not support simple average calculations, counts of non-empty rows were created for each milk component to divide the sum of each milk component value for the calculation of averages.

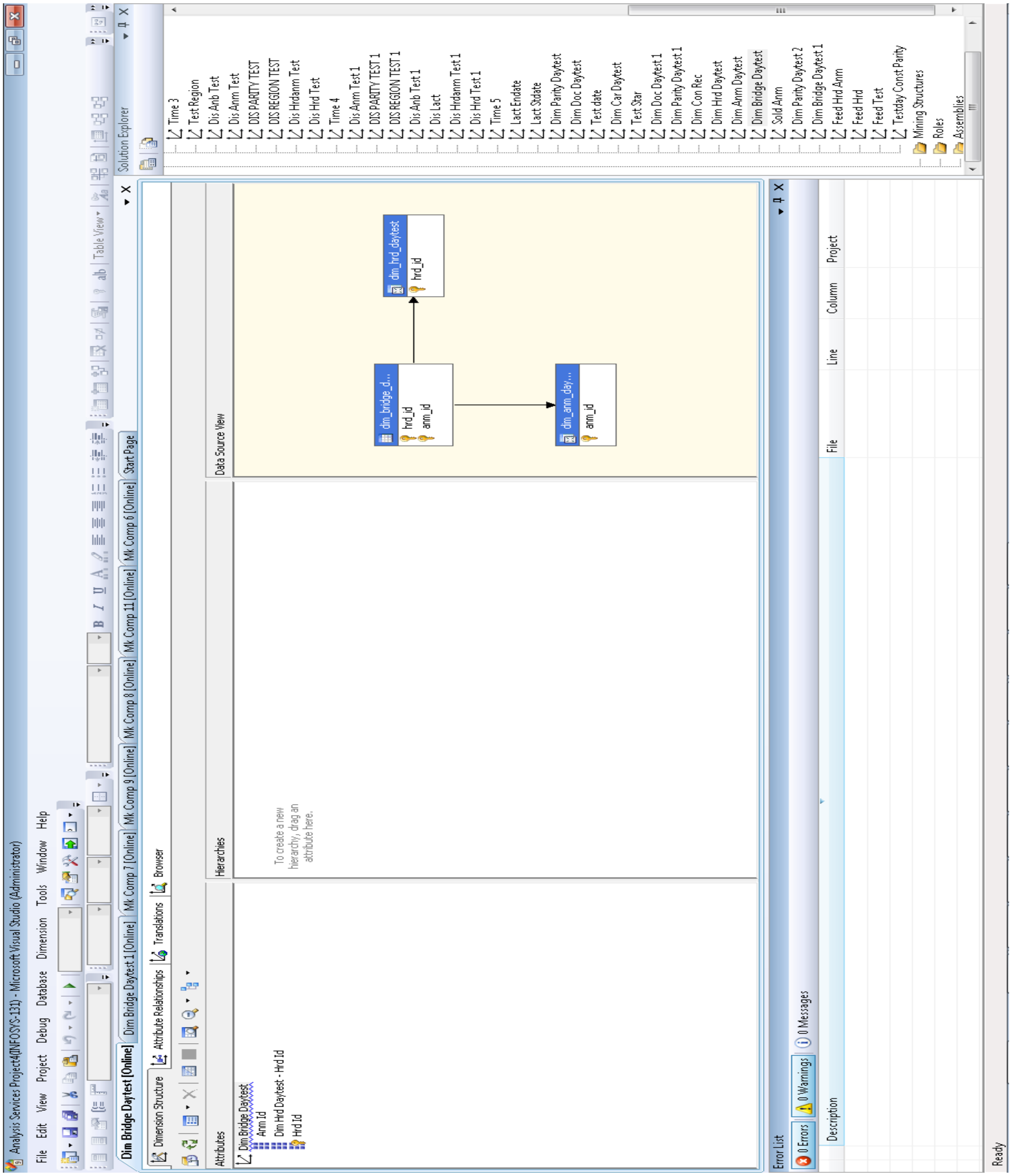


Figure 3.12: Herd_Bridge_Cow (snowflaking for many-to-many relationship between herd and cow and fact table).

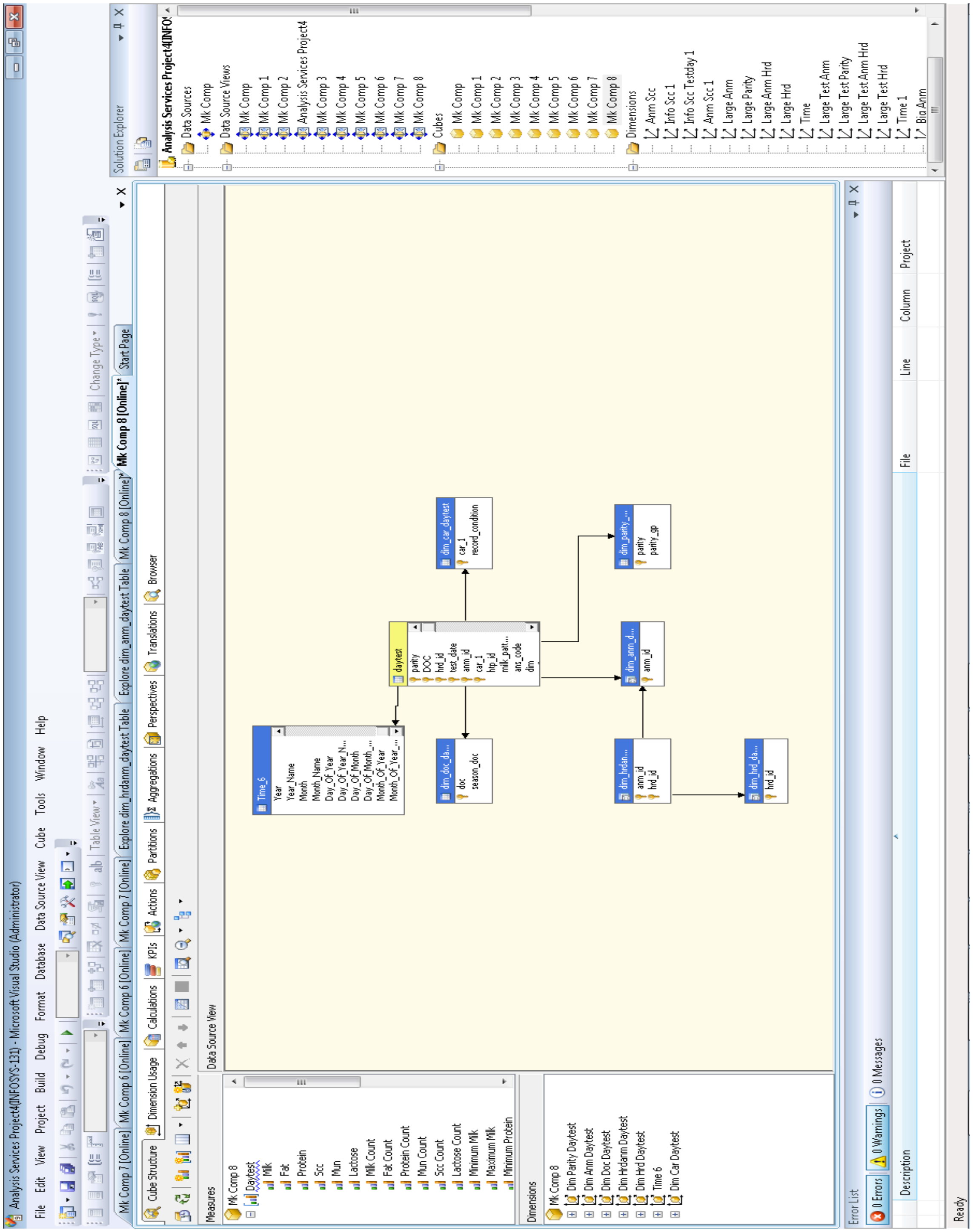


Figure 3.13: Cube structure

Dimensions appeared in the dimensions pane of the cube structure tab in the cube designer window; seven dimensions were created at the database level, as displayed in solution explorer; however, there are eight dimensions in the cube shown in the Figure 3.13. The time dimension in the cube was not present in SQL server, but the built-in dimension in analysis services was used. The data which were used were based on a ten-year time period (2000-2009) so dates were mapped to the dates of a built-in time dimension and a hierarchy of year, month and date was used (Figure 3.14).

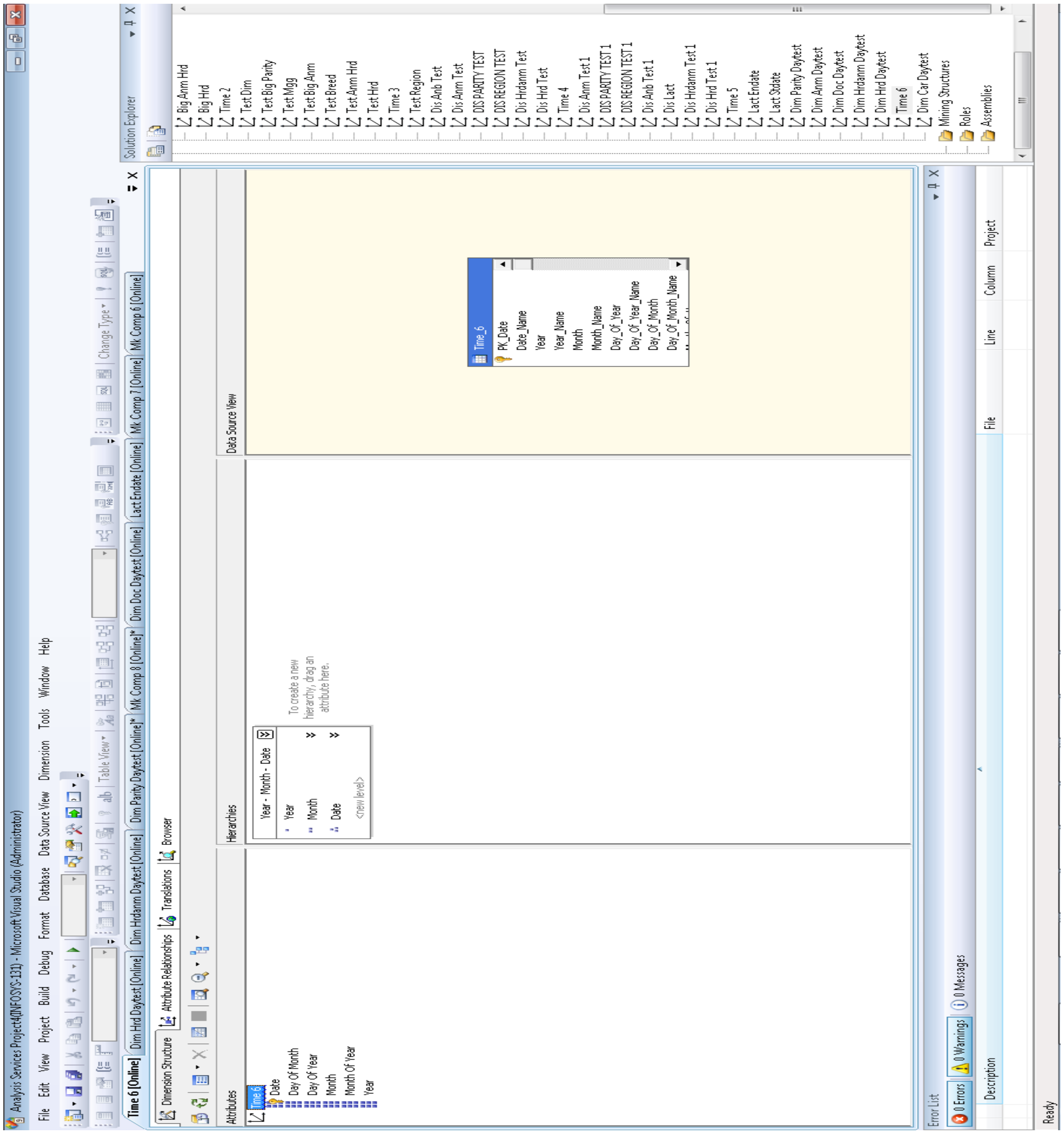
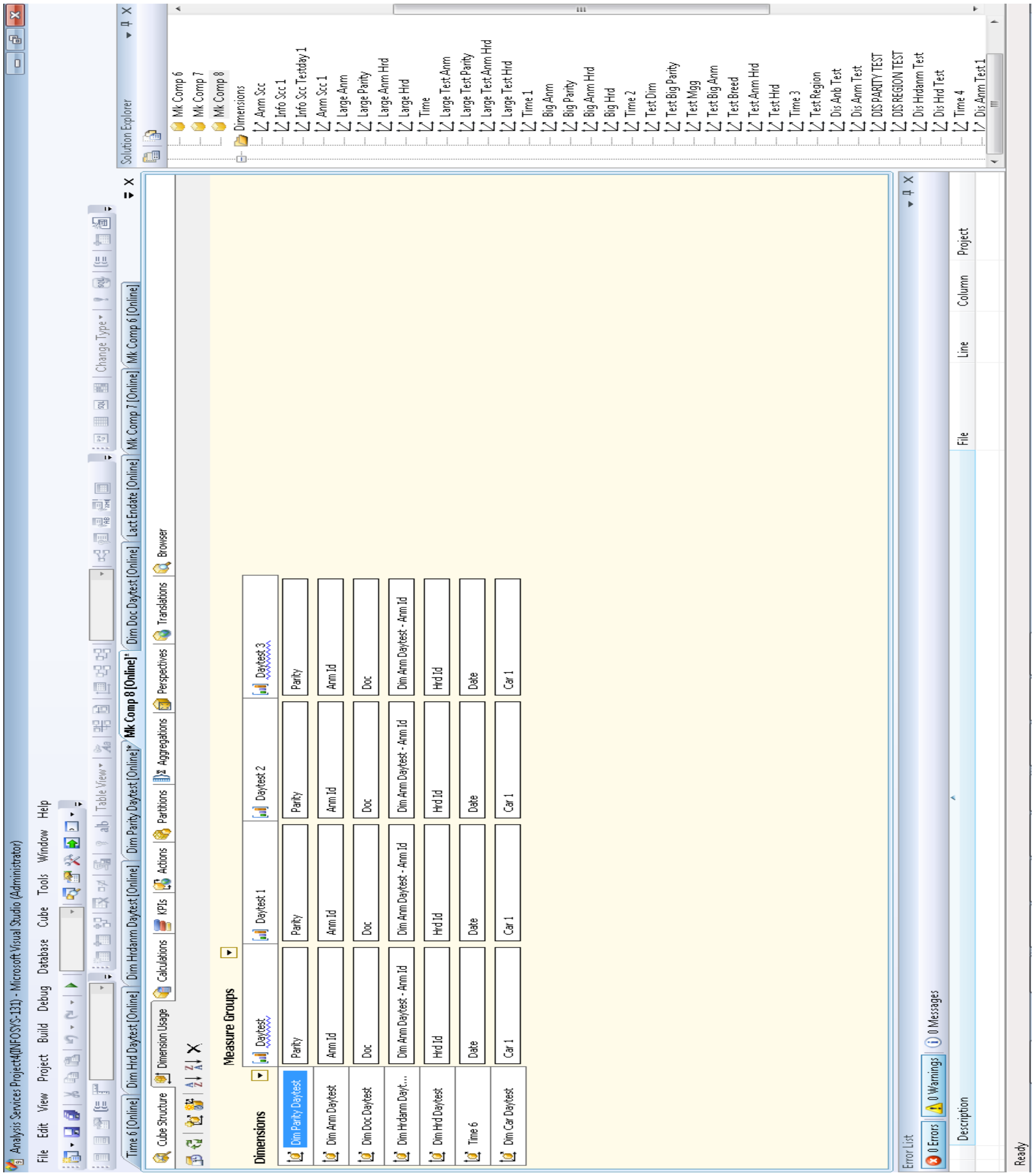


Figure 3.14: Dimension structure for time dimension

Before deploying and processing the cube each dimension was processed and hierarchies were set.

In the solution explorer pane, the options of the dimension designer included four tabs for dimension structure, attribute relationships, translations, and browser. The dimension structure tab included three panes: attributes, hierarchies, and data source view. Those attributes which were present in the dimension table appeared in the attribute pane. The hierarchies were defined in the "hierarchies" pane of the dimension structure tab. In the attribute relationship tab, Microsoft SQL Server Analysis Services automatically defines the relationship between the key attribute and each non-key attribute of the dimension in the case of the star schema. However, in the snowflake schema where dimension attributes are derived from multiple related tables (cow → herd-cow bridge table →herd), attribute relationships are automatically defined (i) between the key attribute (anm_id-dimension cow) and foreign key attribute (anm_id in fact table); (ii) between the key attribute (anm_id, hrd_d id composite key in the herd-cow bridge table) and the foreign key attribute (anm_id dimension cow) in the dimension tables; and (iii) between the key attribute (hrd_id dimension herd) and the foreign key in the bridge table (anm_id, hrd_d id composite key in the herd-cow bridge table). The dimension usage tab was important given that it was the place where the relationship between each dimension and fact table had to be defined and the granularity level was set for calculations (Figure 3.15).



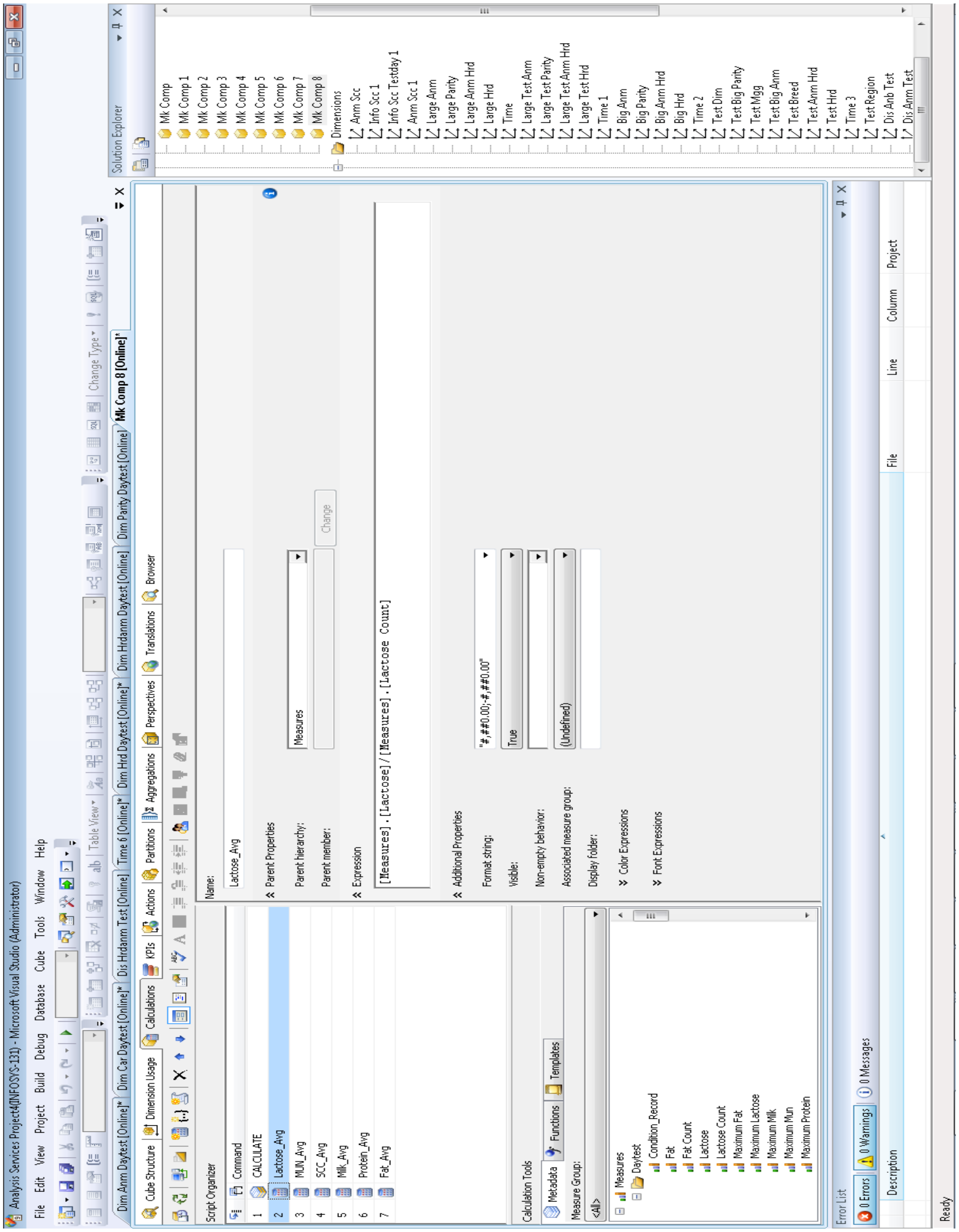
Dimensions	Daytest	Daytest 1	Daytest 2	Daytest 3
Dim Parity Daytest	Parity	Parity	Parity	Parity
Dim Anm Daytest	Anm Id	Anm Id	Anm Id	Anm Id
Dim Doc Daytest	Doc	Doc	Doc	Doc
Dim Hrdanm Dayt...	Dim Anm Daytest - Anm Id	Dim Anm Daytest - Anm Id	Dim Anm Daytest - Anm Id	Dim Anm Daytest - Anm Id
Dim Hrd Daytest	Hrd Id	Hrd Id	Hrd Id	Hrd Id
Time 6	Date	Date	Date	Date
Dim Car Daytest	Car 1	Car 1	Car 1	Car 1

Figure 3.15: Dimension usage in cube structure tab

Before calculated members were created in the cube it was deployed and processed. The "deployment" started with the build option and the actual aim of deployment was to compile the cube and create an analysis services database and update the existing database after each deployment. On the other hand "processing" the cube, copied the data from the underlying data sources into the cube objects; however, without processing the cube it could not be browsed. Thus “deployment” gave a structure to the analysis services database and “process” gave the data for analysis. Whilst the browser tab was used to view both cube and dimension data, this tab provided special capabilities based on the objective of browsing. For dimensions, this tab offered a way to navigate a hierarchy all the way down to the leaf node (Year-Month-Date). The browser tab also offered a built-in MDX query designer by which data were filtered according to requirements. Data from the cube were also analyzed using Excel which can point to the cube with the help of a predefined connection in analysis services.

3.4: Analysis of data with an OLAP cube

Once the cube was processed, the measures which were created in the cube structure measures pane were available for calculations or query with a multidimensional expression (MDX). The calculations tab of the cube design MDX was used to calculate the averages for milk components. Format string property was used to define formatting settings ("#, ##0.00;-#, ##0.00") that control how measures are displayed to clients (Figure 3.16). After the calculations cube was processed again, different options for processing the cube like process default, process full, process data, process structure, unprocess, process index and process incremental were available (Figure 3.17). The “process default” option identified the process state of the database tables, and performed the processing necessary to a fully-processed state. When there were changes in the data, this option implemented “Process Full” on the affected table. “Process full” was performed on a cube that had previously been processed. Analysis Services discarded all data in the cube, and then processed the cube. When a structural change was made to the cube this processing was used. “Process data” was only useful when there were no aggregations in the cube. “Process structure” was only used for cube structure to be processed, not for relevant structures related to cube-like mining models. “Unprocess” was used to delete all structures, data indexes and aggregations from the cube. “Process index” was used to generate or reconstruct indexes and aggregations for a processed cube. “Process incremental” was used to send SQL queries to read the entire dimension table and apply the changes. This worked in a different way than “Process full,” as this option did not discard the dimension storage contents. However, “Process full,” did an implicit process clear on the cube. “Process incremental” was slower than “Process full” since it had added work to apply the changes to the cube.


Analysis Services Project4(INFOSYS-131) - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Database Cube Tools Window Help
Table View
Change Type
Dim Arm Daytest [Online]* | Dim Car Daytest [Online]* | Dis Hrdanm Test [Online] | Time 6 [Online] | Dim Hrd Daytest [Online]* | Dim Hrdanm Daytest [Online] | Dim Parity Daytest [Online] | Mk Comp 8 [Online]*
Cube Structure | Dimension Usage | Calculations | KPIs | Actions | Partitions | Aggregations | Perspectives | Translations | Browser
Script Organizer
Calculation Tools
Metadata | Functions | Templates
Measure Group:
<All>
Change
Solution Explorer
Error List
0 Errors | 0 Warnings | 0 Messages
Description | File | Line | Column | Project
Ready

Script Organizer

	Command
1	CALCULATE
2	Lactose_Avg
3	MUN_Avg
4	SCC_Avg
5	Milk_Avg
6	Protein_Avg
7	Fat_Avg

Name: Lactose_Avg

« Parent Properties

Parent hierarchy: Measures

Parent member:

« Expression

[Measures].[Lactose]/[Measures].[Lactose Count]

« Additional Properties

Format string: "#,##0.00;-#,##0.00"

Visible: True

Non-empty behavior:

Associated measure group: (Undefined)

Display folder:

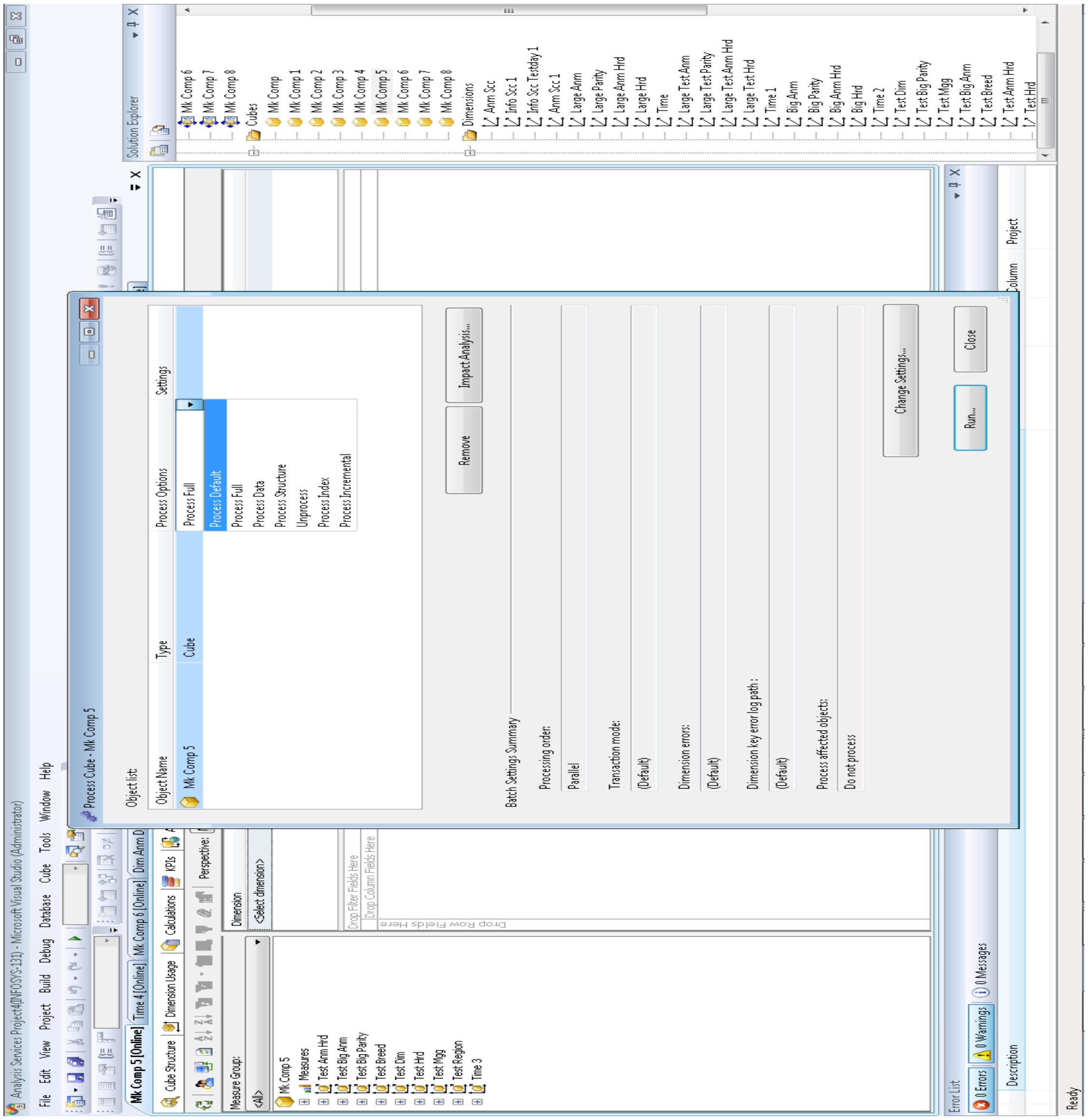
» Color Expressions

» Font Expressions

Measures
- Daytest
  - Condition_Record
  - Fat
  - Fat Count
  - Lactose
  - Lactose Count
  - Maximum Fat
  - Maximum Lactose
  - Maximum Milk
  - Maximum Mun
  - Maximum Protein

Solution Explorer
- Mk Comp
- Mk Comp 1
- Mk Comp 2
- Mk Comp 3
- Mk Comp 4
- Mk Comp 5
- Mk Comp 6
- Mk Comp 7
- Mk Comp 8
- Dimensions
  - Anm Scc
  - Info Scc 1
  - Info Scc Testday 1
  - Anm Scc 1
  - Large Anm
  - Large Parity
  - Large Anm Hrd
  - Large Hrd
  - Time
  - Large Test Anm
  - Large Test Parity
  - Large Test Anm Hrd
  - Large Test Hrd
  - Time 1
  - Big Anm
  - Big Parity
  - Big Anm Hrd
  - Big Hrd
  - Time 2
  - Test Dim
  - Test Big Parity
  - Test Mgg
  - Test Big Anm
  - Test Breed
  - Test Anm Hrd
  - Test Hrd
  - Time 3
  - Test Region
  - Dis Anb Test
  - Dis Anm Test

Figure 3.16: Calculated measures



Process Cube - Mk Comp 5

Object list:

Object Name	Type	Process Options	Settings
Mk Comp 5	Cube	Process Full	

Batch Settings Summary

Processing order:
Parallel

Transaction mode:
(Default)

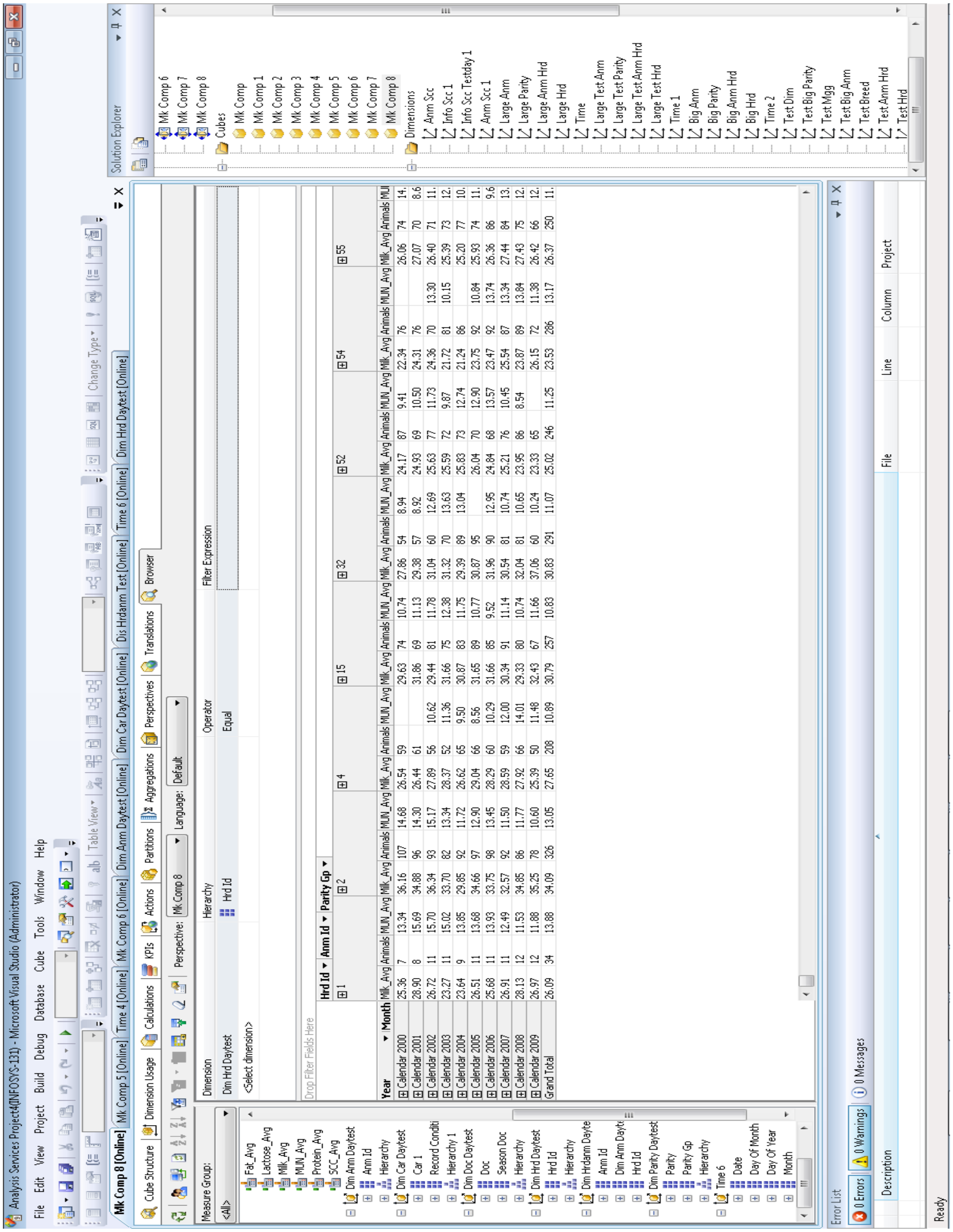
Dimension errors:
(Default)

Dimension key error log path :
(Default)

Process affected objects:
Do not process

Figure 3.17: Processing options of a cube

After processing, the cube was ready for data analysis, and for the first time the cube was available for data analysis in cube editor (Figure 3.18). Data analysis is the procedure of drawing out information from large databases. The strong feature of the OALP supporting software is the drag and drop option for drill down and roll-up navigation inside the cube data. When operations (such as level-climbing, packing, function-application, projection, dicing) were applied to the existing cube, each operation resulted in a new cube. Aggregate functions were available through slicing and navigation.



Perspective: Mk Comp 8 Language: Default

Measure Group: <All>

Dimension	Hierarchy	Operator	Filter Expression
Dim Hrd Daytest	Hrd Id	Equal	
<Select dimension>			

Drop Filter Fields Here

Hrd Id / Anm Id / Parity Gp	1			2			4			15			32			52			54			55		
Year / Month	Milk_Avg	Animals	MUN_Avg	Milk_Avg	Animals	MUN_Avg	Milk_Avg	Animals	MUN_Avg	Milk_Avg	Animals	MUN_Avg	Milk_Avg	Animals	MUN_Avg	Milk_Avg	Animals	MUN_Avg	Milk_Avg	Animals	MUN_Avg	Milk_Avg	Animals	MUI
Calendar 2000	25.36	7	13.34	36.16	107	14.68	26.54	59		29.63	74	10.74	27.86	54	8.94	24.17	87	9.41	22.34	76		26.06	74	14.
Calendar 2001	28.90	8	15.69	34.88	96	14.30	26.44	61		31.86	69	11.13	29.38	57	8.92	24.93	69	10.50	24.31	76		27.07	70	8.6
Calendar 2002	26.72	11	15.70	36.34	93	15.17	27.89	56	10.62	29.44	81	11.78	31.04	60	12.69	25.63	77	11.73	24.36	70	13.30	26.40	71	11.
Calendar 2003	23.27	11	15.02	33.70	82	13.34	28.37	52	11.36	31.66	75	12.38	31.32	70	13.63	25.59	72	9.87	21.72	81	10.15	25.39	73	12.
Calendar 2004	23.64	9	13.85	29.85	92	11.72	26.62	65	9.50	30.87	83	11.75	29.39	89	13.04	25.83	73	12.74	21.24	86		25.20	77	10.
Calendar 2005	26.51	11	13.68	34.66	97	12.90	29.04	66	8.56	31.65	89	10.77	30.87	95		26.04	70	12.90	23.75	92	10.84	25.93	74	11.
Calendar 2006	25.68	11	13.93	33.75	98	13.45	28.29	60	10.29	31.66	85	9.52	31.96	90	12.95	24.84	68	13.57	23.47	92	13.74	26.36	86	9.6
Calendar 2007	26.91	11	12.49	32.57	92	11.50	28.59	59	12.00	30.34	91	11.14	30.54	81	10.74	25.21	76	10.45	25.54	87	13.34	27.44	84	13.
Calendar 2008	28.13	12	11.53	34.85	86	11.77	27.92	66	14.01	29.33	80	10.74	32.04	81	10.65	23.95	86	8.54	23.87	89	13.84	27.43	75	12.
Calendar 2009	26.97	12	11.88	35.25	78	10.60	25.39	50	11.48	32.43	67	11.66	37.06	60	10.24	23.33	65		26.15	72	11.38	26.42	66	12.
Grand Total	26.09	34	13.88	34.09	326	13.05	27.65	208	10.89	30.79	257	10.83	30.83	291	11.07	25.02	246	11.25	23.53	286	13.17	26.37	250	11.

0 Errors 0 Warnings 0 Messages

Figure 3.18: Cube browser

The set of allowed aggregate functions are average, count, sum, min, rank (n), and no-operation. All of them were well-known relational aggregate functions, except for no-operation which means that no function is applied on the data of the cube and rank (n) which returns the first n-components of an aggregated set of values which can be ordered. Detail of these cube operations are discussed in the results and discussion section.

Chapter 4: Results and Discussion

4.1: Data Extraction and cleaning

The extraction of data from the operational data stores was important as these stores did not support analytical query execution. Analytical topics like ‘number of animals in different parities in a herd,’ ‘rank of cows’ milk production, and ‘feed intake needs,’ require significant data sorting and the joining of multiple tables. As a result such queries really slow down the working of operational databases. Therefore it is advisable to separate computer systems; *i.e.*, an operational database for operational tasks and a data warehouse for analytical query execution (Chaudhuri and Dayal, 1997). Operational databases are relational databases; therefore their structure of relational databases does not support the analytical query execution. Relational databases support short queries due to their highly normalized data. Analytical queries are *ad hoc* and access millions of records at a time, which requires many table scans, such that normalized tables cannot support them.

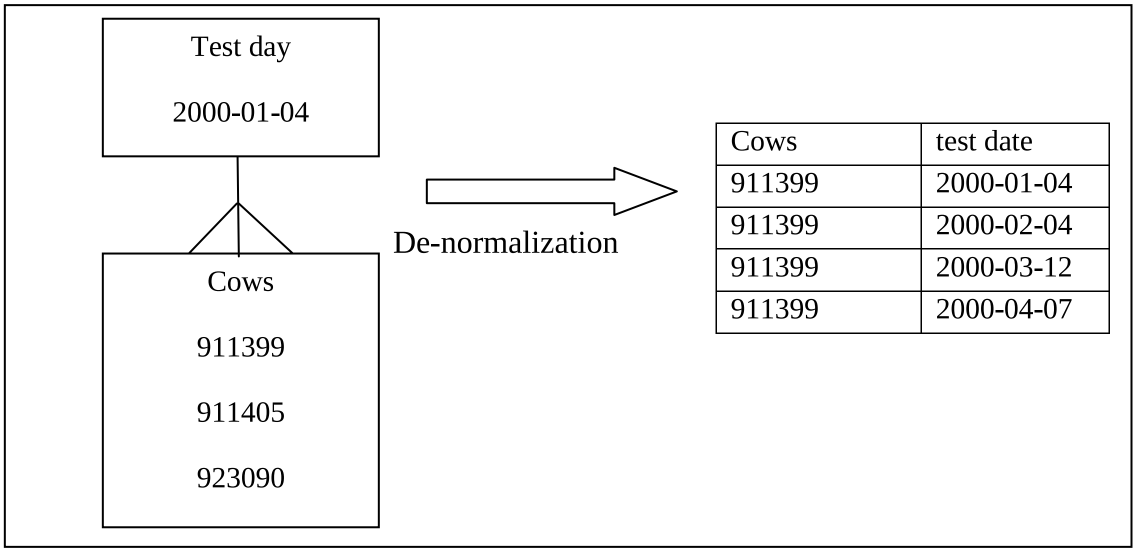
The extraction of data from the operational databases at Valacta was not easy as these datasets differed from conventional datasets present in business environments. The major challenge was to extract data in chronological order as these datasets are available for a period of more than 10 years. The main reason to keep historical data is to observe the performance of cows; however, an ordinary business would only keep data for six to ten years. None of this data had been deleted, allowing one to see the performance of past cows. In a business environment, relevant data can change due to the introduction/discontinuation of new/old products, price changes, changes in customer preferences, so the results of queries can often become outdated. However, in the case of dairy data, historical data are as important as data pertaining to human medical issues, which are collected but seldom deleted. In this perspective, the design of a data warehouse was not easy, as historical data had to be stored in a chronological order so that whenever they was extracted for analysis they would give reasonable results.

The main objective of collecting data at dairy farms is to see the performance of the dairy herds and to take effective decisions at the farm level. Decision-making depends upon the analysis of data. If data are entered incorrectly and then used for analysis they will not show a clear picture of the cows and ultimately the herds. Thus, to get the desired results from the data and to take effective decisions, data were cleaned by taking input from the domain experts and consulting the literature. Outliers were removed and missing values were also treated while designing the multidimensional data model. Data were cleaned in a different staging area before being loaded to the data warehouse. Almost everywhere data warehouses have been developed researchers (Ahmad et al., 2004; Chaudhuri and Dayal, 1997; Gray and Watson, 1998) have emphasised the importance of cleaning the data. Therefore data warehouse was not only a storage place for historical data used for analytical purpose, but also contained cleaned data.

4.2. Data transformation and de-normalization

The data transformation process was a very important phase of project. In this process all the variables which were desired by the analytical queries were calculated, *e.g.*, dry matter intake by the cows on a specific test date. Attributes were selected for defining dimensions (herd, cow, test date etc.) as well as facts (quantity of milk, fat and protein percentages on a specific test date, etc.). The primary key and foreign keys were declared while views were created on the base tables. Data were subdivided into, data about a cow on individual test day, summarized data on all test days for each cow, and information about variables. The lowest level of detail of each record was test day on which milk was recorded for each cow. Test day records were considered as a special case because these records are unique on Quebec dairy farms, and available for an individual cow. If the cow was considered as an entity then it had certain characteristics like a registration number, date of birth, weight, body condition score, date of conception, lactation start date, lactation end date, information related to milk production and milk components, information on calves born, information on the sire to which this cow was bred, information related to its feeding and about the herd in which the cow was present. The test day records were available from the year 2000 to the year 2009. Milk components like fat and protein percentages, SCC and MUN were recorded in year 2000 but records on lactose were only available from 2001 onwards. Percentages of available MUN and lactose records were 37% and 30%, respectively, among other milk component records, as these were recorded in a subset of herds. Almost all transformations were conducted with the help of SQL.

An important part of the data transformation process is de-normalization which actually improves query performance. The de-normalized data were used to develop multidimensional models which provide: (i) a database structure that are easy for end users to understand and submit queries to these databases; and (ii) database structures which maximise the efficiency of queries, which is achieved by minimising the number of tables and relationships among them. These structures also decrease the complexity of the database and lessen the number of joins obligatory in user queries. Therefore de-normalization is appropriate for a data warehouse (Ahmad et al., 2004). The de-normalization of cow and test day table into one table is illustrated in Figure 4.1.



Cows	test date
911399	2000-01-04
911399	2000-02-04
911399	2000-03-12
911399	2000-04-07

Figure 4.1: Transformation of normalized data into de-normalized data.

The normalization is the main cause for the complexity of transactional databases; it tends to increase the number of tables needed to keep functionally-dependent attributes in separate tables. Although this helps in updating of data, it seriously thwarts the retrieval of data. In data warehousing, redundancy is a minor problem because data is not updated on-line. Consequently, data in the data warehouse is de-normalized, a technique whereby one moves from higher to lower normal forms of database modeling in order to speed up database access and hence query processing (Kimball, 1996).

4.3: Data warehouse

After the identification of dimensions and facts, dimension and fact tables were created during the data transformation process, make them available for the design of a data warehouse schema. A data warehouse schema is a database design which can show the relationships between different tables containing data. A schema for a data warehouse is a multidimensional data model having a central fact table and surrounding dimension tables. To design a data warehouse, a data-driven approach was used. In this approach data were first explored for identification of dimension and fact tables and then an abstract schema was developed for a data warehouse. This approach was also used by Mahboubi *et al.,* (2010) in their design of a data warehouse for simulation results. Designing a data warehouse with this approach was found to: (i) be the fastest method for the development of a data warehouse; and (ii) provided a good opportunity to understand, explore and analyze the data. After the completion of the data warehousing process the queries were executed on the warehouse, as suggested by Inmon (1996).

The use of a data warehouse requires dimensional models, which represents a different modelling approach from operational databases. The basic schema of operational databases is one of entity relationship modelling. Entity relationship models cannot allow users to easily navigate in the database and execute analytical queries, as this modeling technique is used to remove redundancy in the data and is coupled with normalization. The data in the data warehouse is mostly denormalized; therefore, entity relation models could not be used as the basis for data warehouses. On the other hand multidimensional data models allowed quick access to the data and were built on denormalized data. A possible entity relationship model which depicts the difference between this model and a multidimensional data model is shown in Figure 4.2. Where an entity is something which can be distinctly identified, such as herd, cow, test day (on which milk is collected), milk components (milk yield, percentage of fat, protein and lactose, MUN and SCC). A relationship is an association among entities, *e.g.*, the relationships between herd and cows, and cows and test days and test days and milk components. Studies also specify that traditional database design techniques, *i.e.*, entity relationship modeling and normalized tables, are not appropriate for data warehouse applications (Pedersen et al., 2001); as a result, new techniques, *e.g.*, star schemas, have emerged. These better support the data warehouse needs of data analysis (Kimball, 1996). Some other schemas include the snowflake schema, or a hybrid of the two, the star-flake schema (Ahmad et al., 2004).

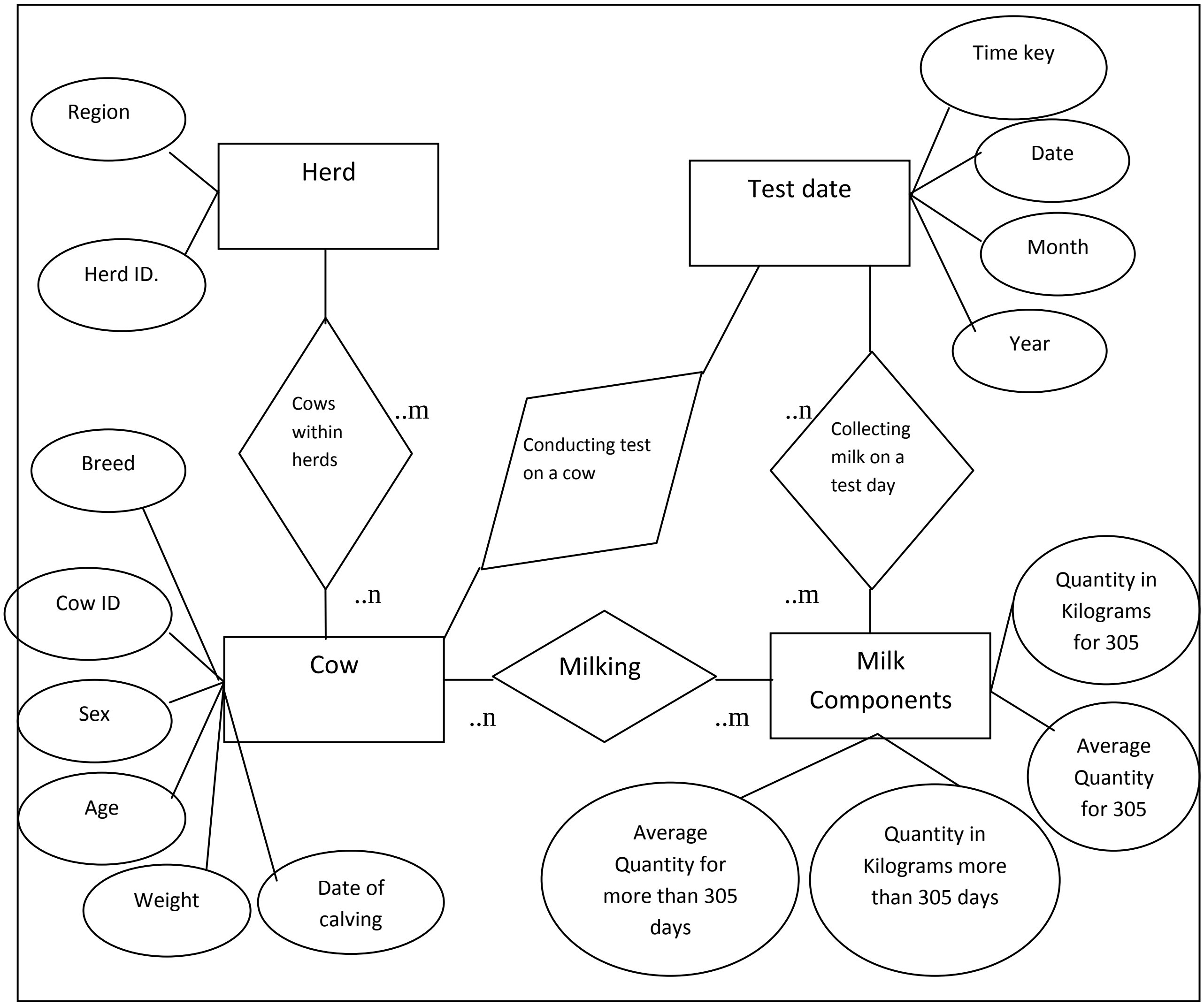


Figure 4.2: An entity relationship diagram of test day records.

Multidimensional data models emerged during the last decade to exploit data warehouses when the objective was to analyze the data rather than to perform online transactions. Multidimensional models categorize data either as facts with associated numerical measures or as textual dimensions that characterize the facts. It was found that these models were helpful in (i) viewing certain features of the data, (ii) summarizing the data, and (iii) providing a quick response to analytical queries. Need of all three factors is most obvious in databases that are used for decision-support. Models were visualized in such a way that future users can draw information from these databases even when implemented on a larger scale.

4.3.1: Selection of schema (data model)

Among the three schemas discussed, the star schema had the simplest database structure, containing a fact table in the center which is surrounded by the dimension tables (Figure 4.3).

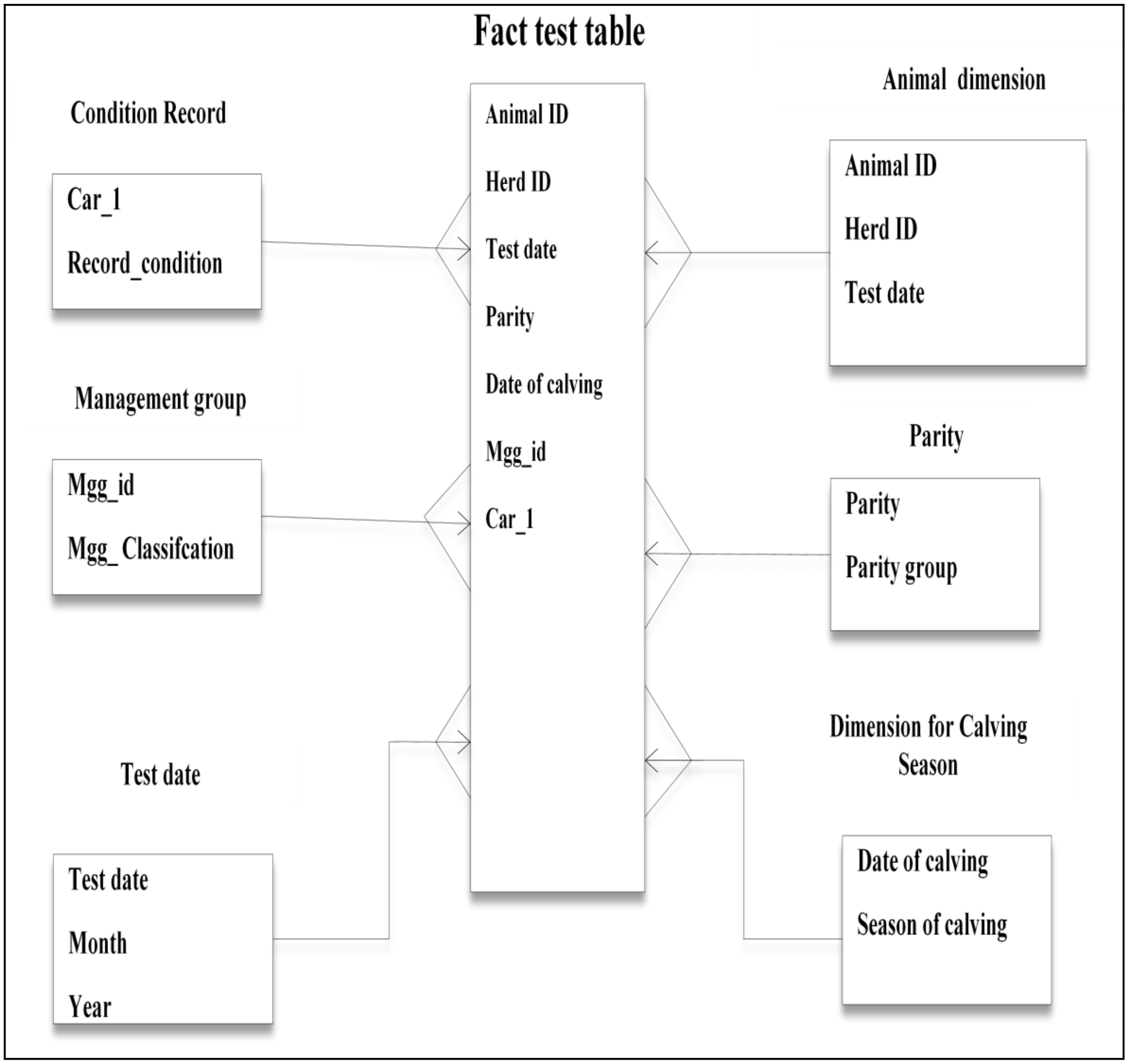
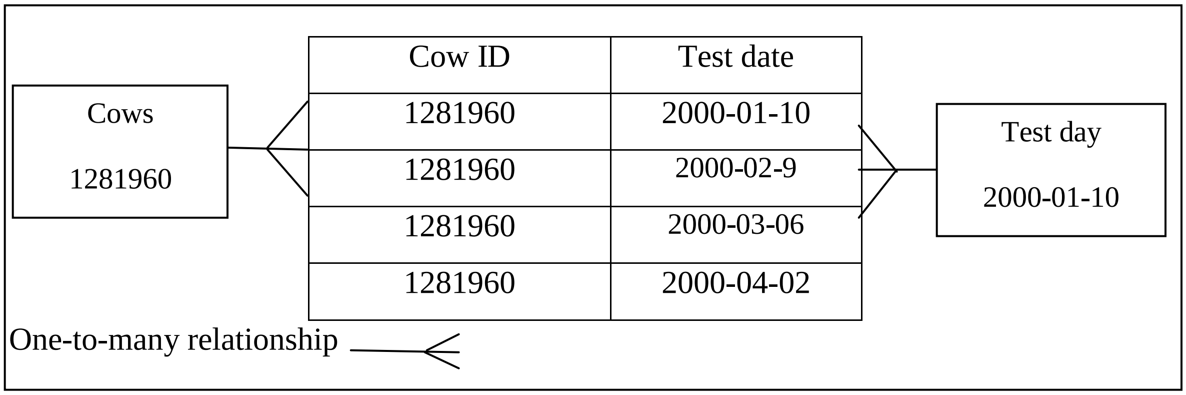


Figure 4.3: Star schema (data model)

Most data warehouses used a star schema to represent the multidimensional data model (Chaudhuri and Dayal, 1997). A simple star schema for a test day file (Figure 4.4) was arranged such that the dimensions have an exactly one-to-many relationship with the fact table. The model is composed of a single fact table surrounded by a single table for each dimension. The fact table contains measurements of the mean yields of milk, milk fat and milk protein. The dimension tables provide the basis for the summarization of the measurements in the fact table. The primary key of the fact table is the combination of the primary keys of all the dimension tables that provide its multidimensional match, and stores the numeric measures for those matches. In the case of a star schema, dimension tables have embedded hierarchies which are the result of denormalization of the data in a data warehouse. A detailed star model is depicted in Figure 4.5.



Cow ID	Test date
1281960	2000-01-10
1281960	2000-02-9
1281960	2000-03-06
1281960	2000-04-02

Figure 4.4: Example of relationship between a fact table and dimension tables (data model of star schema).

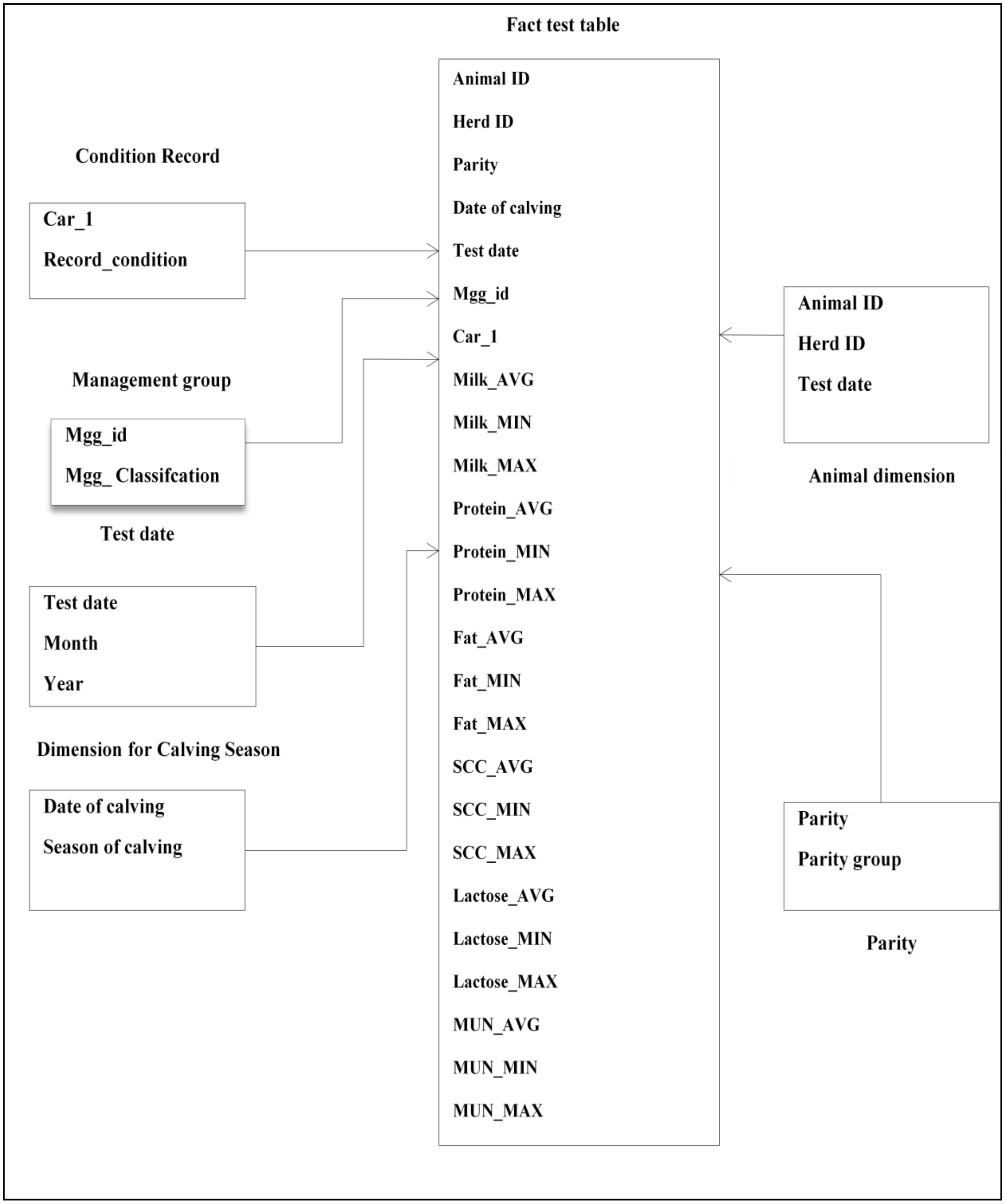


Figure 4.5: Detailed star schema.

Using a star schema approach seems quite simple, and the best practice to design a data warehouse (*e.g*., Kimball *et al*.,(1998)). The formation of star schemas is based on an analysis of user query requirements and then identification of facts that need to be aggregated and the dimensional attributes to aggregate them by. Nevertheless there are a number of practical challenges with this approach, including the existence of relationships in the data leading to ultimately incorrect results (Moody and Kortink, 2000). As a result the approach of Inmon (1996) was used in the preset study, by first exploring and understanding the data and then trying to develop a model. The following discussion comprehensively explains the characteristics of datasets available for this research and also highlights the need to develop a snowflake schema instead of a star schema.

In the case of multidimensional data models, measures (facts in the fact table) are in exactly *n*-1 relationship with the elements of the dimension tables, which, in turn, set strict hierarchies from lower level to upper level, *e.g*., milk is specifically recorded for one cow on one test day, and on a specific test day the cow belongs to a specific management group, in a specific parity in one specific herd. But cows belong to different management groups according to their stage of lactation within a single parity, and cows could also move from one herd to another herd in the same parity or different parities. Likewise the relationship between dimension tables and a fact table is also many-to-many, as a cow normally has multiple test day records in lactation and each cow may have has several locations. So a data model in which test day records for each cow were counted only once was needed. However this would not be easy using conventional multidimensional data models.

This problem was solved by snowflaking the dimensions having many-to-many relations with the data in the fact table. Although Pederson *et al.,* (2001) discussed three ways to achieve this – traditional dimensions, mini-dimensions and snow flaking – it was found that, in the case of first approach, a primary key was needed for our research, based on the combination of herd ID, cow ID and test date, to uniquely identify a record in the fact table. The name of this dimension table was herd_cow_testday. The total records exceeded one million in our case. The length of each variable in the herd_cow_testday dimension table was eight, resulting in a table 24 Mb in size. This was estimated by roughly multiplying the record length with the number of records. The size of this table on the disk was 219 Mb on the SQL database server including the index size. If the same approach were used for the 117,068,136 rows which are present in the feed file, the disk size for this table would be 2.7 Gb. This would be a very wide dimension table and simply unusable for the client. It would also cause difficulties in writing MDX (multidimensional expression) queries as this dimension structure would not provide implicit attribute hierarchies (herd → cow).

In a second approach, the use of mini dimensions was chosen, but it proved not be very useful as there were 6,917 herds; if we took cows in each herd as a dimension table then we would have 6,917 mini dimension tables, which would also be impractical.

The third approach was to use snowflaking dimensions, which provided a refinement of the star schema, where the dimensional hierarchy is clearly represented by normalizing the dimension tables (separate herd and cow dimension tables). This advantage was also reported by Chaudhuri and Dayal (1997); however, they emphasised that the denormalized structure of the dimensional table in a star schema may be more suitable for browsing dimensions. In the case of this research, although it was easy to browse the dimensions in the star schema, the star schema did not support hierarchies of attributes within a dimension table, so they needed to be defined. This concept is known as a user-defined hierarchy that is a hierarchy of attribute, used to facilitate browsing cube data by users, without adding to cube space.

The snowflake schema (Figure 4.6) actually appeared by decomposing one or more dimensions in which the hierarchy was already present.

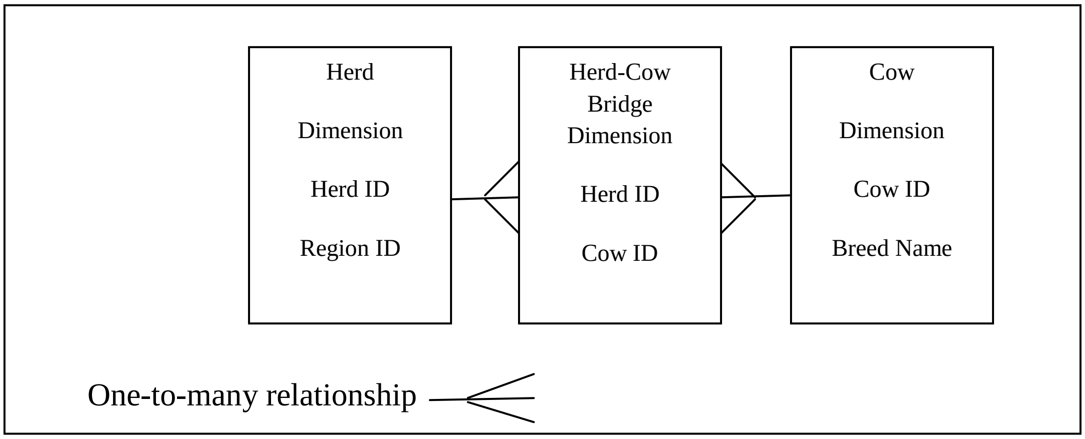
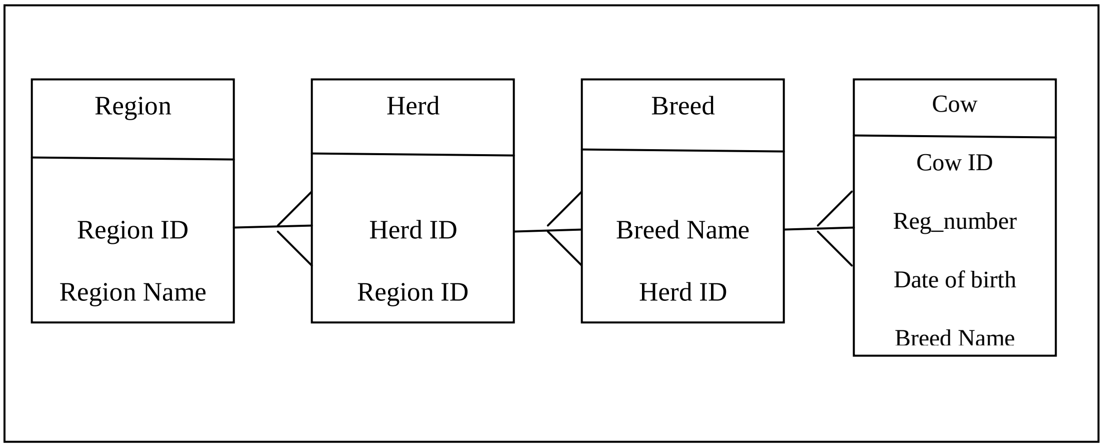


Figure 4.6: Example of hierarchy for herd dimension, herd-cow bridge dimension and cow dimension

According to Moody and Kortink (2000) the snowflake schema can be formed from a star schema by normalising the hierarchies in each dimension. They also drew upon an entity relationship model to discover dimensions and fact tables for the snowflake schema. It also proved a good opportunity to design a snowflake schema from an entity relationship model; these steps would help database users to understand the schema design. The steps were the following:

- A transaction entity (transaction entities record details about particular events that occur at the farm, *e.g.*, milk yield, fat yield, protein yield, etc…) can be used to form a fact table. The key to the table is the combination of the keys of the associated component entities that is an entity which is directly related to a transaction entity via a one-to-many relationship. Component entities describe the information or "components" of each business transaction at the farm. These entities can answer what, who, when, where and how questions in business events. As an example of "what": What is the production of this herd; "when": When did this cow last calve; "which": which are the top ten milk producing herds; "where": Where is mastitis most prevalent; and "who": who achieved the highest milk production and a disease-free farm?
- A component entity can be converted into a dimension table. Numerical attributes within transaction entities can be combined by the key attributes. Where

hierarchical relationships exist between transaction entities, the child entity inherits all of the relationships to component entities (and key attributes) from the parent entity. For example Region, herd, breed, cows are all “ancestors” for milk yield (Figure 4.7).



One-to-many relationship

Figure 4.7: Example of hierarchy for region, herd, breed and cow

Another way of defining hierarchy could occur from the herd level but in a different way: within herds there were different management groups; in different management groups there were different parities; and in different parities there were different number of cows. Cows transfer from their present parity to their next parity, and can move between different management groups and between different herds. So there was hierarchy of herd → management group → parity → cow. To solve the problem of many-to-many relationship between herd and cow dimension; a herd-cow bridge dimension was used which has a composite of herd ID and cow ID.

This was one advantage to the fact that the size of dimensions on the disk was only 20 Mb for herd dimension, herd-cow bridge dimension and cow as a dimension. Also, the hierarchical structure of the snowflake schema helped in writing easy MDX (multidimensional data expression) queries. Separate dimensions were defined for parity and management group. The reason for defining them separately will be discussed below.

In the present era hardware is available at a low cost and the disk space required by the star schema could be overlooked, but even then the snowflake schema is better in MDX query execution, due to its support for hierarchies. Execution time for an MDX query is also very important with respect to user interaction through OLAP tools.

The management group (assigned to a cow according to its status of lactation) and the lactation number (parity) in which the cow was, identified a specific cow state at the herd level. A management group and parity were good examples of typical OLAP dimensions, as these characterize the state of the cow. Since a cow could transfer from one herd to another within a parity this was quite an important issue, which was resolved by defining separate dimensions of parity and management group to avoid the double count of measures in summarization processes like roll-up to next level in a dimension. Likewise it was inappropriate to take parity in management group in the same dimension, since 60% of test day records had management group information on them. So defining either parity under management group or management group over parity created equal problems due to insufficient information on management group. Another very important aspect was time. The test date was considered the lowest granularity on which summarization of different facts were calculated. The test day records show records of milk yield and milk components for each cow at approximately each month in her productive life. So dealing with the chronological order of recorded data was quite an important task, allowing one to show trends in milk yield and milk components over time. The condition record dimension had information about the health status of the cow. In these data, there were twenty one conditions related to health status of the cow. Dimension of date of calving determines the season of calving which is very important with respect to milk yield. Figure 4.8 depicts the snowflake schema in detail.

This study provided an excellent opportunity to observe and design the hierarchical structure in the data. Equal hierarchies are fundamental to OLAP tools since they allow clients to query summarized data at any level of correctness within the hierarchy, drilling down to a more specific analysis or rolling up to a more concise analysis when needed. Three features of the hierarchy in cows deserved special attention.

1) The cow hierarchy was non-strict: a lower-level cow could be a member of several herds at a higher-level, on the other hand, strict hierarchies exist where

every lower-level cow belongs to a single higher-level parity (cow belongs to a specific parity once in a life).

2) When there was movement of cow from one herd to other herd there was a change in hierarchy.
3) The hierarchy was not balanced: *e.g.*, under the hierarchies of herds there were different numbers of cows in each herd.

It was important to deal with these types of hierarchies to permit drilling down from higher levels to lower levels or vice versa and to get the correct summarized results from the facts.

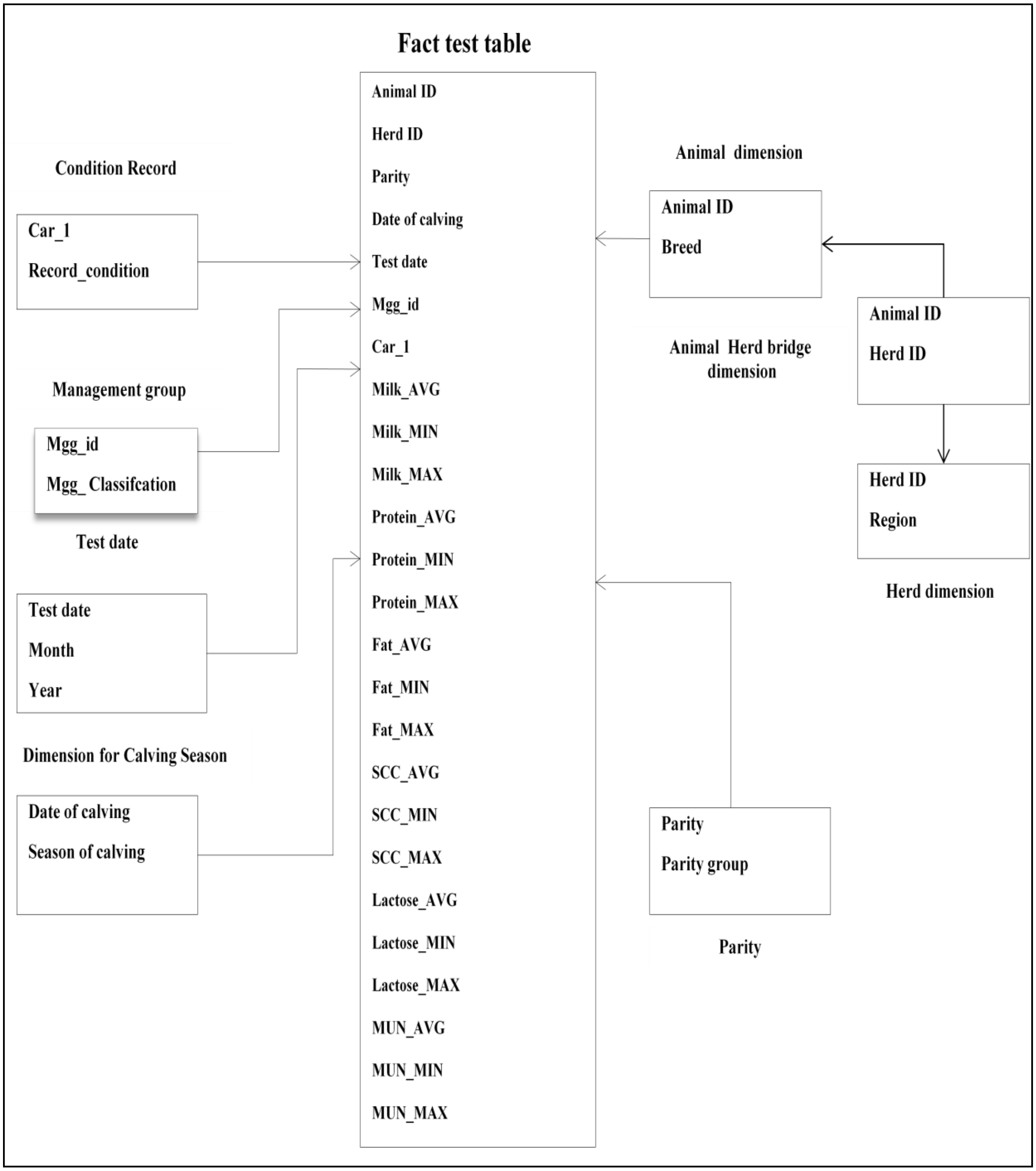


Figure 4.8: Snowflake schema (data model).

The capture of hierarchies and other schema-related issues are discussed below:

- The schema which was developed typically captured non-strict hierarchies in the dimensions. This permitted drill-down and roll-up. In our example, the hierarchies were arranged cow < cow-herd bridge dimension < herd. If the bridge dimension was not defined with a composite key of herd ID and cow ID the data were double counted. The same problem was reported by Pedersen *et al.*, (2001) and they proposed an extended multidimensional data model to address this issue.

- Different dimensions for test date and date of calving. As an example, the test date dimension days roll up to months and months roll up to year. However, for date of calving, year was divided into four seasons.
- Occasionally the hierarchies in a dimension were not balanced, i.e., the path from the root to the leaves had varying lengths. In this case, which occurs in the cow hierarchy, each parity had a varying number of management groups and each management group had a varying number of cows. This problem was solved by defining separate dimensions for both the parity and management group. In this way these dimensions had normalized data and data were counted just once for these dimensions. Pedersen *et al.,* (2001) also suggested the normalization for these types of unbalanced hierarchies.
- The schema included aggregation semantics of the data to provide a phenomenon which helps users execute only those queries which are meaningful, *i.e.*, avoiding addition of non-additive data. As an illustration, it might not be meaningful to add % fat together, but performing average calculations on them would make sense.
- The schema allowed measures to be treated as dimensions and vice versa. In our case, the attribute 'milk yield' of cows was typically treated as a measure, to allow for computations for average milk yields in a herd etc.; however, we could define a dimension which allowed grouping cows into low, middle and high milk-yielders.
- The schema allowed change in the hierarchy. *e.g.*, entrance of cow into a herd and culling of a cow.
- For some types of data, *e.g*., test day records, aggregation involves counting the number of occurrences. So aggregation functions were carefully studied and applied to prevent users from performing wrong calculations.
- The number of cows in a specific herd varied at a specific time, meaning that the total number of cows varied in the overall dataset from time to time. In essence,

this means that it was meaningless to count cows over years in herds, but in a year or in a month, that information might be useful.

- The milk yield can be added over cow or over herds for a specific year or over years but most temporal milk averages are considered for a cow or for a herd for a specific year.

The fact constellation schema was also designed; this is an example of a set of complex structures in which multiple fact tables share dimensional tables. The advantage of using a fact constellation schema was to combine attributes from two different fact tables having the same dimensions; there was provision to access the data from each fact table separately or in combination, linked with same dimension. The snowflaking affect in herd-herd_cow_bridge-cow dimensions was the same in this schema as when applied in a snowflake schema. In the literature two star schemas are combined to form a constellation schema (Moody and Kortink, 2000). This schema would provide an advantage; if data sources were located at different places where these sources could be accessed by common dimensions having different measures in the fact table. In this case, the feed file had a record about feed consumption of a cow on a specific test day. This information includes all feed ingredients offered to that cow on that test day. This file had a variable named HTP_ID (herd test period id) which was also present in the test day file, whereas the feed file did not contain the variable test date. Consequently, it was decided to introduce a test date column in the feed file during the transformation process. However the test day file already had test day and htp_id. This information in the test day file helped us in combining records on the basis of htp_id which is same for a cow in both files. But the count of htp_id is less than that of the feed file, since the rest of the htp_ids for that cow were present in the DRY COW file. The feed file (amount of feed intake in kilograms) and the test day file (milk components) formed fact tables of feed intake and milk components respectively. They shared dimensions of herd, cow, parity, and management group and test date. Figure 4.9 shows a constellation schema.

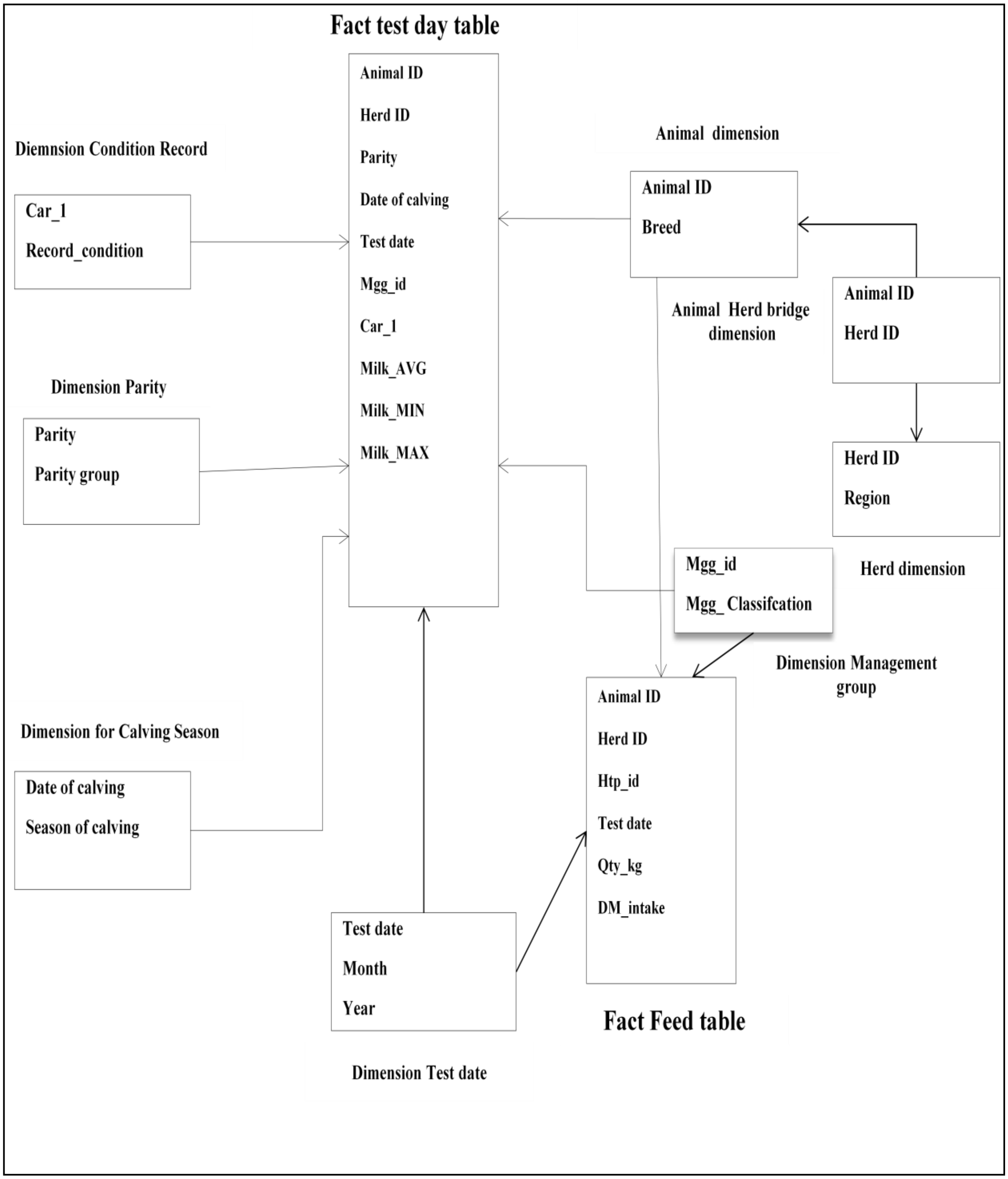


Figure 4.9: Fact Constellation schema

4.3.2: Identified problems with existing OLAP tools

- While the discussion about hierarchies in the previous section is fully or partially supported by the available OLAP tools, there are still potential problems with non-strict hierarchies. Summarization of milk yields on those cows which transferred from one herd to the other herd counted double if a bridge dimension table, having a combination of primary key of cow (cow id) and herd (herd id), had not been defined in the SQL server database, *i.e.*, before loading data to analysis server. As analysis server supported the definition of a primary key/foreign key relationship which did not affect the underlying data stored in the SQL server database. The issue of a non-strict hierarchy was also discussed by Pedersen and Jensen (1998) regarding the designing of a data warehouse for medical clinics.
- Calculations based on aggregation functions such as average and percentage transformed into new queries. For example an average function is transformed into “sum” and “count” functions. Percentages were calculated by dividing selected divided and “0%” format string, a function built into the analysis server. Undeniably on-the-fly execution of extended queries with percentage and average functions lowered the performance of the cube, even if the multidimensional model was accurately developed. It appeared that the problem was with the OLAP aggregation process, not with the hardware, since the same hardware previously calculated “sum” and “count” values for calculation of average within seconds for the same datasets. Therefore pre-aggregation in materialized views showed quick data upload in the cube and response of query was much faster, but was unfortunately very slow in the case of on-the-fly query execution. So the developers and researchers in the field of OLAP should consider those situations where the query may not already be known. Toja *et al.,* (2007) discussed on-the-fly query execution and suggested pre-aggregation and the use of sophisticated hardware which can support parallel processing. Examples of queries are shown in the Figure 4.10 A, B & C.

```
   /* Code for percentage parity in a herd in a year */
   WITH MEMBER [Measures].[Denominator] AS
    SUM([Dim Parity Daytest].[Parity].[Parity],
     IIF([Measures].[Cows]= 0, null, [Measures].[Cows]))
     MEMBER [Measures].[percent parity1]AS
    IIF([Measures].[Denominator] = null, 0,
    ([Dim Parity Daytest].[Parity].&[1],[Measures].[Cows])/ [Measures].[Denominator])
    ,FORMAT_STRING = '0%'

    MEMBER [Measures].[percent parity2] AS
    IIF([Measures].[Denominator] = null, 0,
    ([Dim Parity Daytest].[Parity].&[2],[Measures].[Cows]) / [Measures].[Denominator])
    ,FORMAT_STRING = '0%'
    MEMBER [Measures].[percent parity3] AS
    IIF([Measures].[Denominator] = null, 0,
    ([Dim Parity Daytest].[Parity].&[3],[Measures].[Cows]) / [Measures].[Denominator])
    ,FORMAT_STRING = '0%'
    MEMBER [Measures].[percent parity4] AS
    IIF([Measures].[Denominator] = null, 0,
    ([Dim Parity Daytest].[Parity].&[4],[Measures].[Cows]) / [Measures].[Denominator])
    ,FORMAT_STRING = '0%'
   MEMBER [Measures].[percent parity5] AS
    IIF([Measures].[Denominator] = null, 0,
    ([Dim Parity Daytest].[Parity].&[5],[Measures].[Cows]) / [Measures].[Denominator])
    ,FORMAT_STRING = '0%'

SELECT  {[Measures].[percent parity1],[Measures].[percent parity2],[Measures].[percent
parity3],[Measures].[percent parity4],[Measures].[percent parity5]} ON COLUMNS
 ,{[Dim Hrd Daytest].[Hrd Id].[Hrd Id]*[Time 6].[Year].[Year]} ON ROWS
FROM [Mk Comp 8]
```

Figure 4.10 (A): Code for cows in different parities in different herds.

```
/* Code for count of cows per parity basis in a herd in a specific year*/



SELECT ON COLUMNS, ON ROWS
FROM [Mk Comp 8]
/* Code for average milk in a specific herd in a specific herd on a specific cow */

[Measures].[Milk]/[Measures].[Milk Count]
```

Figure 4.10 (B): Code for cows in different parities in different herds.

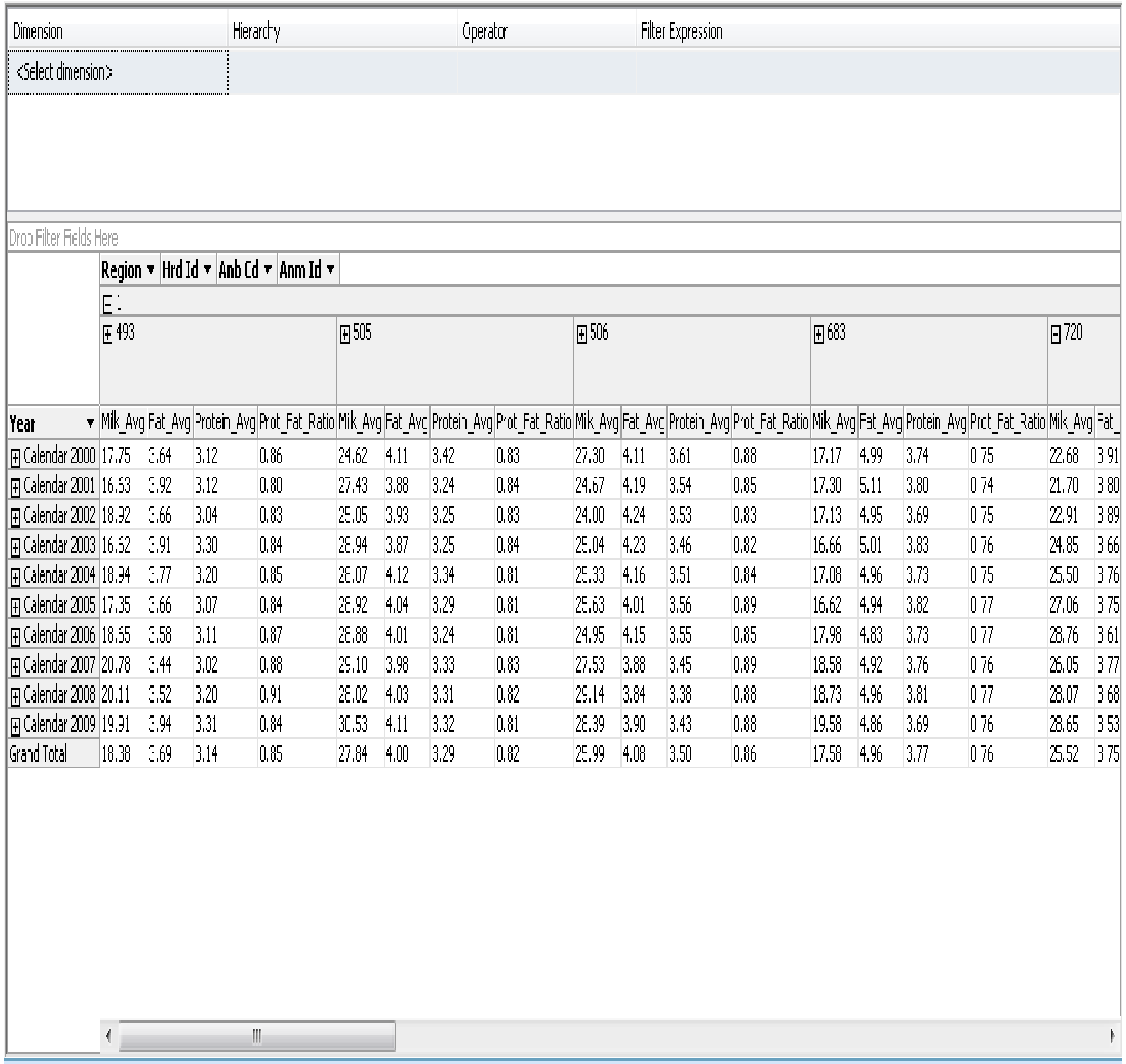
4.3.3: MOLAP server for query processing

OLAP systems are typically implemented by using two technologies: ROLAP (relational online analytical processing), where data are stored in relational database management systems, and MOLAP (multidimensional online analytical processing) where a multidimensional database management system is used, and a query can be executed on the multidimensional structure, *i.e.*, directly on the cube. In this research a MOLAP server was used to store the pre-computation data after modeling them with multidimensional data models. MOLAP servers actually store data in a multidimensional format, the cube structure and data were queried directly from this structure which was not possible with a ROLAP server. It was also found that MOLAP helped in fast and flexible retrieval of multidimensional data. Similar results with MOLAP were found by Chaudhuri and Dayal (1997).

4.4: Cube Operations

The multidimensional view of data considers that information is stored in a multidimensional array sometimes known as a hypercube. This is a group of data cells arranged according to the dimensions of the data. The dimensions of a cube are a list of members by which the user wishes to view the data, *e.g.*, according to herd, cow, year, calving season, parity, etc. The value present at each intersection of dimension, which can be called a cell, has a real measured value like quantity of milk produced or number of cows in a specific parity. Navigation is the term which is used to explore the data within a cube. Cube operations support navigation functions such as: (i) Roll-up, which involves the summarization of data for a higher level of hierarchy, with results that are suitable for strategic decision making, (ii) Roll-down, (a.k.a. Drill down or drill through) which involves navigation to the lowest granularity of the data for tactical and operational decision-making, (iii) Selection, wherein a condition is assessed against the data or elements of a dimension in order to limit the set of recovered data, (iv) Slicing, which allows for the selection of all data satisfying a condition along a particular dimension), and (v) pivoting, which involves changing the dimensional orientation of a cube. According to Vassiliadis (1998) cube operations include level_climbing, packing, function_applications, projection, dicing while complex functions like navigation and slicing are defined on top of the simple functions. Pivoting is the re-organization of the data rather than modifications of the values.

Examples of different cube operations are shown in the following figures. In this operation dates climb up to months and months climb up to years. The lowest level of granularity, which is cow in this case, can climb up to breed, breed to herds, and herd to region (Figure 4.11).



Dimension	Hierarchy	Operator	Filter Expression
<Select dimension>			

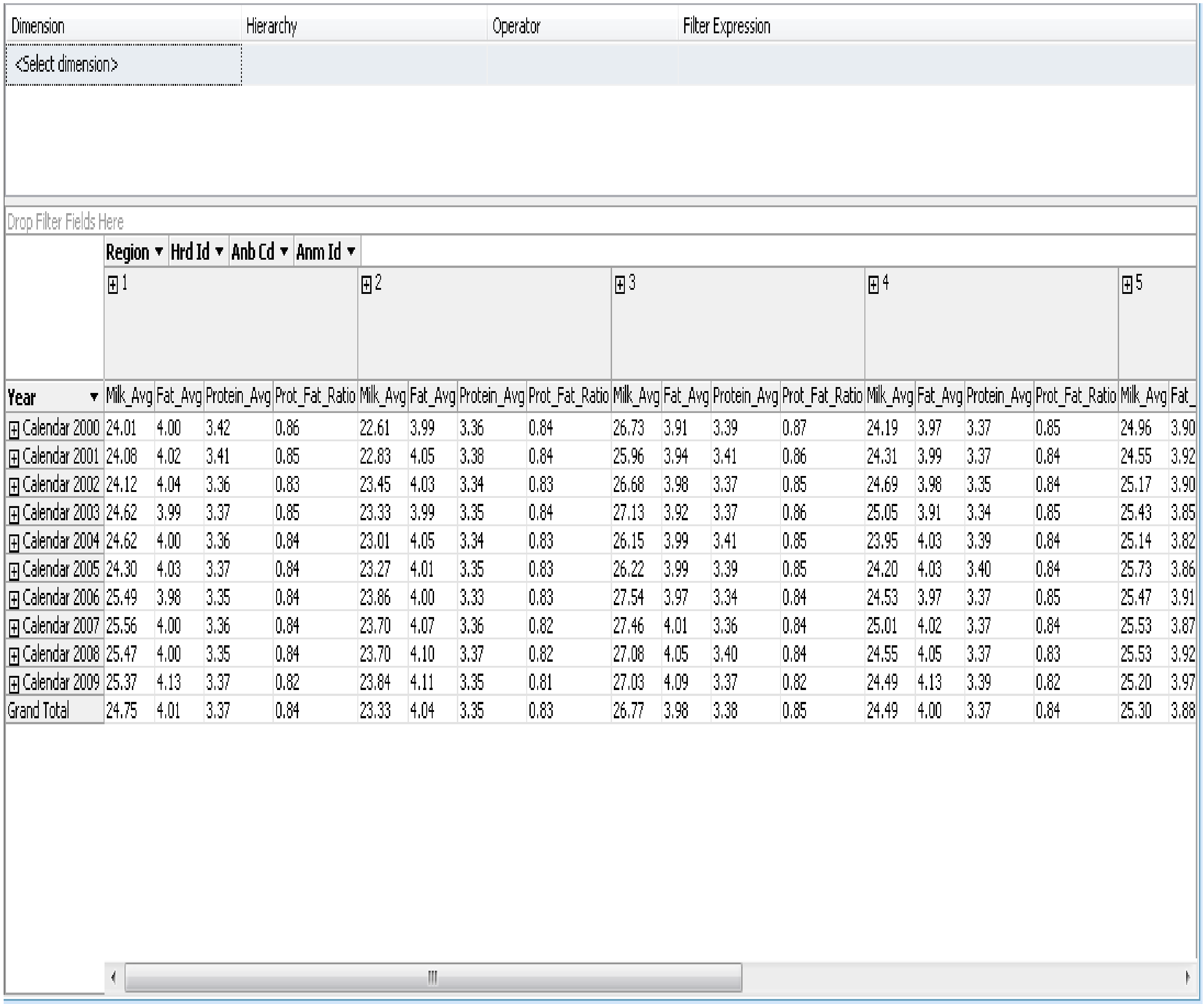
Drop Filter Fields Here

Region ▾ Hrd Id ▾ Anb Cd ▾ Anm Id ▾

	⊟ 1																				
	⊞ 493				⊞ 505				⊞ 506				⊞ 683				⊞ 720	
Year ▾	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_
⊞ Calendar 2000	17.75	3.64	3.12	0.86	24.62	4.11	3.42	0.83	27.30	4.11	3.61	0.88	17.17	4.99	3.74	0.75	22.68	3.91
⊞ Calendar 2001	16.63	3.92	3.12	0.80	27.43	3.88	3.24	0.84	24.67	4.19	3.54	0.85	17.30	5.11	3.80	0.74	21.70	3.80
⊞ Calendar 2002	18.92	3.66	3.04	0.83	25.05	3.93	3.25	0.83	24.00	4.24	3.53	0.83	17.13	4.95	3.69	0.75	22.91	3.89
⊞ Calendar 2003	16.62	3.91	3.30	0.84	28.94	3.87	3.25	0.84	25.04	4.23	3.46	0.82	16.66	5.01	3.83	0.76	24.85	3.66
⊞ Calendar 2004	18.94	3.77	3.20	0.85	28.07	4.12	3.34	0.81	25.33	4.16	3.51	0.84	17.08	4.96	3.73	0.75	25.50	3.76
⊞ Calendar 2005	17.35	3.66	3.07	0.84	28.92	4.04	3.29	0.81	25.63	4.01	3.56	0.89	16.62	4.94	3.82	0.77	27.06	3.75
⊞ Calendar 2006	18.65	3.58	3.11	0.87	28.88	4.01	3.24	0.81	24.95	4.15	3.55	0.85	17.98	4.83	3.73	0.77	28.76	3.61
⊞ Calendar 2007	20.78	3.44	3.02	0.88	29.10	3.98	3.33	0.83	27.53	3.88	3.45	0.89	18.58	4.92	3.76	0.76	26.05	3.77
⊞ Calendar 2008	20.11	3.52	3.20	0.91	28.02	4.03	3.31	0.82	29.14	3.84	3.38	0.88	18.73	4.96	3.81	0.77	28.07	3.68
⊞ Calendar 2009	19.91	3.94	3.31	0.84	30.53	4.11	3.32	0.81	28.39	3.90	3.43	0.88	19.58	4.86	3.69	0.76	28.65	3.53
Grand Total	18.38	3.69	3.14	0.85	27.84	4.00	3.29	0.82	25.99	4.08	3.50	0.86	17.58	4.96	3.77	0.76	25.52	3.75

Figure 4.11: Level_climbing operation

Packing is the cube operation in which cube is consolidated, such as would be the case if the information in Figure 4.12 were packed to years and regions that are the highest level hierarchies.



Dimension	Hierarchy	Operator	Filter Expression
<Select dimension>			

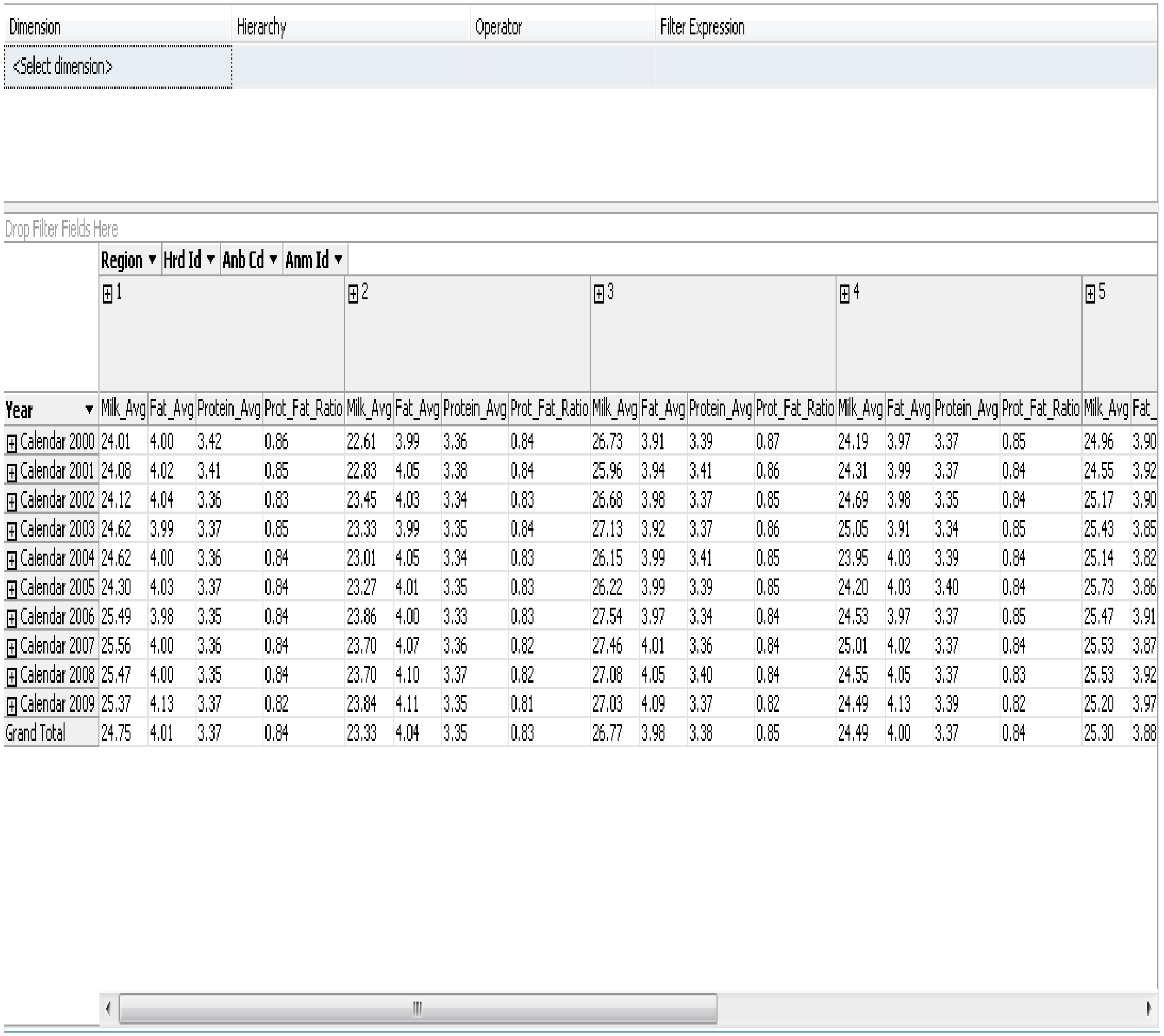
Drop Filter Fields Here

Region ▼ Hrd Id ▼ Anb Cd ▼ Anm Id ▼

	⊞ 1				⊞ 2				⊞ 3				⊞ 4				⊞ 5	
Year ▼	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_
⊞ Calendar 2000	24.01	4.00	3.42	0.86	22.61	3.99	3.36	0.84	26.73	3.91	3.39	0.87	24.19	3.97	3.37	0.85	24.96	3.90
⊞ Calendar 2001	24.08	4.02	3.41	0.85	22.83	4.05	3.38	0.84	25.96	3.94	3.41	0.86	24.31	3.99	3.37	0.84	24.55	3.92
⊞ Calendar 2002	24.12	4.04	3.36	0.83	23.45	4.03	3.34	0.83	26.68	3.98	3.37	0.85	24.69	3.98	3.35	0.84	25.17	3.90
⊞ Calendar 2003	24.62	3.99	3.37	0.85	23.33	3.99	3.35	0.84	27.13	3.92	3.37	0.86	25.05	3.91	3.34	0.85	25.43	3.85
⊞ Calendar 2004	24.62	4.00	3.36	0.84	23.01	4.05	3.34	0.83	26.15	3.99	3.41	0.85	23.95	4.03	3.39	0.84	25.14	3.82
⊞ Calendar 2005	24.30	4.03	3.37	0.84	23.27	4.01	3.35	0.83	26.22	3.99	3.39	0.85	24.20	4.03	3.40	0.84	25.73	3.86
⊞ Calendar 2006	25.49	3.98	3.35	0.84	23.86	4.00	3.33	0.83	27.54	3.97	3.34	0.84	24.53	3.97	3.37	0.85	25.47	3.91
⊞ Calendar 2007	25.56	4.00	3.36	0.84	23.70	4.07	3.36	0.82	27.46	4.01	3.36	0.84	25.01	4.02	3.37	0.84	25.53	3.87
⊞ Calendar 2008	25.47	4.00	3.35	0.84	23.70	4.10	3.37	0.82	27.08	4.05	3.40	0.84	24.55	4.05	3.37	0.83	25.53	3.92
⊞ Calendar 2009	25.37	4.13	3.37	0.82	23.84	4.11	3.35	0.81	27.03	4.09	3.37	0.82	24.49	4.13	3.39	0.82	25.20	3.97
Grand Total	24.75	4.01	3.37	0.84	23.33	4.04	3.35	0.83	26.77	3.98	3.38	0.85	24.49	4.00	3.37	0.84	25.30	3.88

Figure 4.12: Packing

To see the number of cows in each parity, in a specific herd, first all the cows in a specific parity were counted and then their count was divided by the total number of cows present in a herd to get percent of cows in each parity. The basic idea of function application was to build a new cube on an existing one. Another example, with protein to fat ratio can be seen in Figures 4.13 A & B.

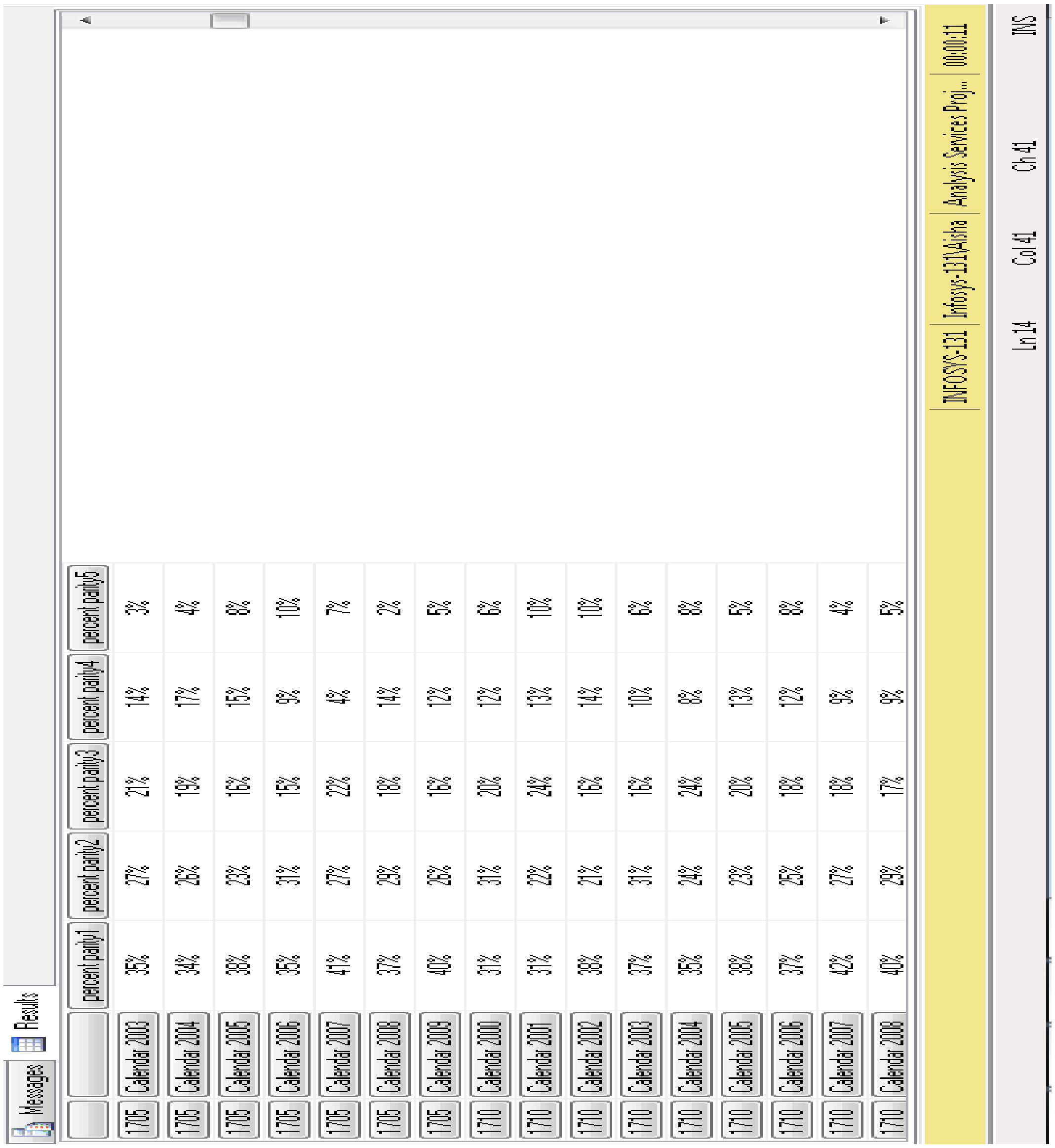


Dimension	Hierarchy	Operator	Filter Expression
<Select dimension>			

Drop Filter Fields Here

	Region ▾ Hrd Id ▾ Anb Cd ▾ Anm Id ▾																				
	⊞ 1				⊞ 2				⊞ 3				⊞ 4				⊞ 5	
Year ▾	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_
⊞ Calendar 2000	24.01	4.00	3.42	0.86	22.61	3.99	3.36	0.84	26.73	3.91	3.39	0.87	24.19	3.97	3.37	0.85	24.96	3.90
⊞ Calendar 2001	24.08	4.02	3.41	0.85	22.83	4.05	3.38	0.84	25.96	3.94	3.41	0.86	24.31	3.99	3.37	0.84	24.55	3.92
⊞ Calendar 2002	24.12	4.04	3.36	0.83	23.45	4.03	3.34	0.83	26.68	3.98	3.37	0.85	24.69	3.98	3.35	0.84	25.17	3.90
⊞ Calendar 2003	24.62	3.99	3.37	0.85	23.33	3.99	3.35	0.84	27.13	3.92	3.37	0.86	25.05	3.91	3.34	0.85	25.43	3.85
⊞ Calendar 2004	24.62	4.00	3.36	0.84	23.01	4.05	3.34	0.83	26.15	3.99	3.41	0.85	23.95	4.03	3.39	0.84	25.14	3.82
⊞ Calendar 2005	24.30	4.03	3.37	0.84	23.27	4.01	3.35	0.83	26.22	3.99	3.39	0.85	24.20	4.03	3.40	0.84	25.73	3.86
⊞ Calendar 2006	25.49	3.98	3.35	0.84	23.86	4.00	3.33	0.83	27.54	3.97	3.34	0.84	24.53	3.97	3.37	0.85	25.47	3.91
⊞ Calendar 2007	25.56	4.00	3.36	0.84	23.70	4.07	3.36	0.82	27.46	4.01	3.36	0.84	25.01	4.02	3.37	0.84	25.53	3.87
⊞ Calendar 2008	25.47	4.00	3.35	0.84	23.70	4.10	3.37	0.82	27.08	4.05	3.40	0.84	24.55	4.05	3.37	0.83	25.53	3.92
⊞ Calendar 2009	25.37	4.13	3.37	0.82	23.84	4.11	3.35	0.81	27.03	4.09	3.37	0.82	24.49	4.13	3.39	0.82	25.20	3.97
Grand Total	24.75	4.01	3.37	0.84	23.33	4.04	3.35	0.83	26.77	3.98	3.38	0.85	24.49	4.00	3.37	0.84	25.30	3.88

Figure 4.13 (A): Function application (protein to fat ratio)



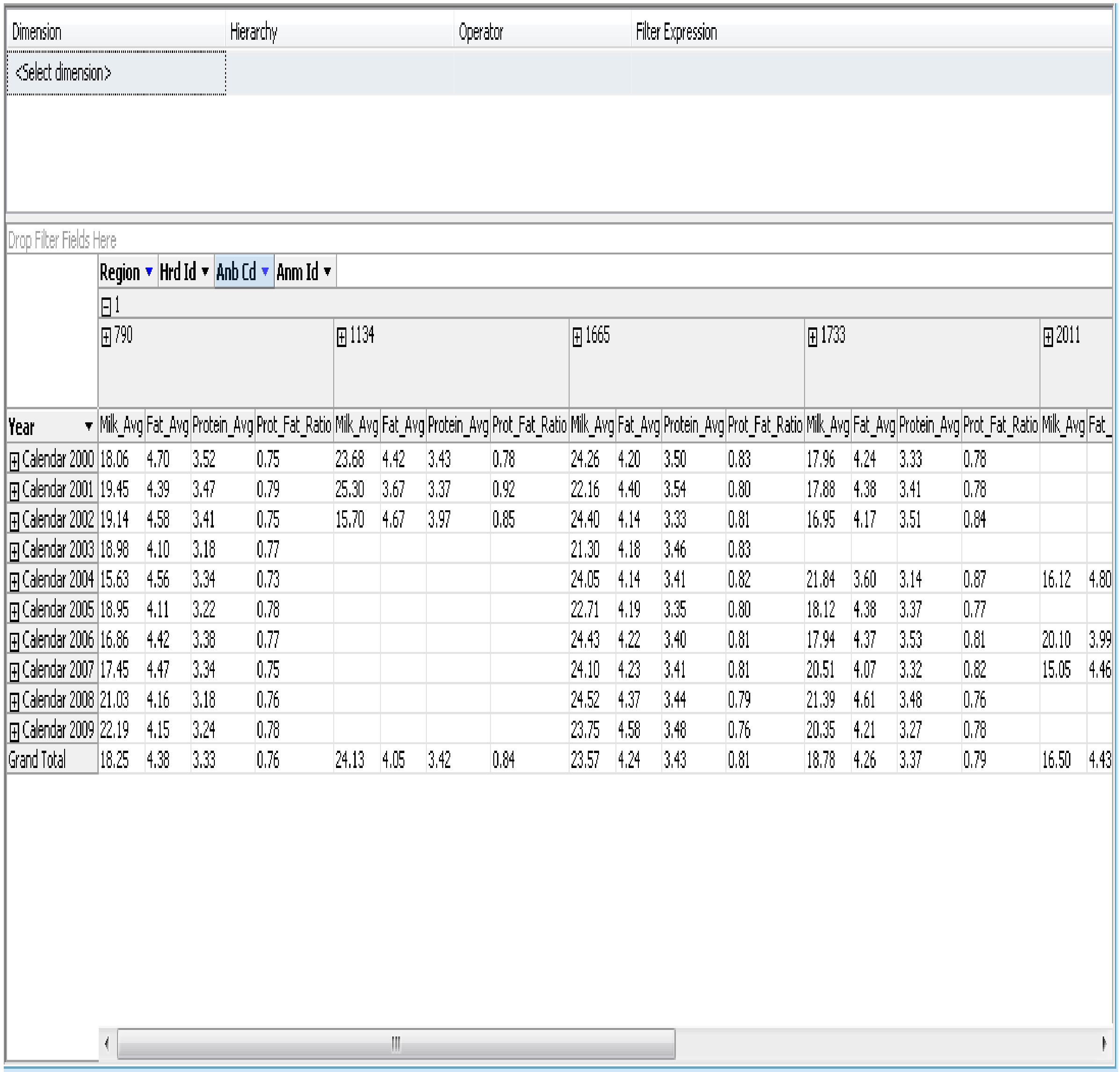
		percent parity1	percent parity2	percent parity3	percent parity4	percent parity5
1705	Calendar 2003	35%	27%	21%	14%	3%
1705	Calendar 2004	34%	26%	19%	17%	4%
1705	Calendar 2005	38%	23%	16%	15%	8%
1705	Calendar 2006	35%	31%	15%	9%	10%
1705	Calendar 2007	41%	27%	22%	4%	7%
1705	Calendar 2008	37%	29%	18%	14%	2%
1705	Calendar 2009	40%	26%	16%	12%	5%
1710	Calendar 2000	31%	31%	20%	12%	6%
1710	Calendar 2001	31%	22%	24%	13%	10%
1710	Calendar 2002	38%	21%	16%	14%	10%
1710	Calendar 2003	37%	31%	16%	10%	6%
1710	Calendar 2004	35%	24%	24%	8%	8%
1710	Calendar 2005	38%	23%	20%	13%	5%
1710	Calendar 2006	37%	25%	18%	12%	8%
1710	Calendar 2007	42%	27%	18%	9%	4%
1710	Calendar 2008	40%	29%	17%	9%	5%

INFOSYS-131 | Infosys-131\Aisha | Analysis Services Proj... | 00:00:11

Ln 14 Col 41 Ch 41 INS

Figure 4.13(B): Function application

The cube illustrated in Figure 4.14 has values on Region 1 having all herds with cows of the Ayrshire breed. In this case, the purpose of the operators was to form a cube which was based on some specific dimensions and aggregate over the rest of the dimensions by use of aggregation function.



Dimension	Hierarchy	Operator	Filter Expression
<Select dimension>			

Drop Filter Fields Here

Region ▾ Hrd Id ▾ Anb Cd ▾ Anm Id ▾

⊟ 1

	⊞ 790				⊞ 1134				⊞ 1665				⊞ 1733				⊞ 2011	
Year ▾	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_Avg	Protein_Avg	Prot_Fat_Ratio	Milk_Avg	Fat_
⊞ Calendar 2000	18.06	4.70	3.52	0.75	23.68	4.42	3.43	0.78	24.26	4.20	3.50	0.83	17.96	4.24	3.33	0.78		
⊞ Calendar 2001	19.45	4.39	3.47	0.79	25.30	3.67	3.37	0.92	22.16	4.40	3.54	0.80	17.88	4.38	3.41	0.78		
⊞ Calendar 2002	19.14	4.58	3.41	0.75	15.70	4.67	3.97	0.85	24.40	4.14	3.33	0.81	16.95	4.17	3.51	0.84		
⊞ Calendar 2003	18.98	4.10	3.18	0.77					21.30	4.18	3.46	0.83						
⊞ Calendar 2004	15.63	4.56	3.34	0.73					24.05	4.14	3.41	0.82	21.84	3.60	3.14	0.87	16.12	4.80
⊞ Calendar 2005	18.95	4.11	3.22	0.78					22.71	4.19	3.35	0.80	18.12	4.38	3.37	0.77		
⊞ Calendar 2006	16.86	4.42	3.38	0.77					24.43	4.22	3.40	0.81	17.94	4.37	3.53	0.81	20.10	3.99
⊞ Calendar 2007	17.45	4.47	3.34	0.75					24.10	4.23	3.41	0.81	20.51	4.07	3.32	0.82	15.05	4.46
⊞ Calendar 2008	21.03	4.16	3.18	0.76					24.52	4.37	3.44	0.79	21.39	4.61	3.48	0.76		
⊞ Calendar 2009	22.19	4.15	3.24	0.78					23.75	4.58	3.48	0.76	20.35	4.21	3.27	0.78		
Grand Total	18.25	4.38	3.33	0.76	24.13	4.05	3.42	0.84	23.57	4.24	3.43	0.81	18.78	4.26	3.37	0.79	16.50	4.43

Figure 4.14: Slice and dice

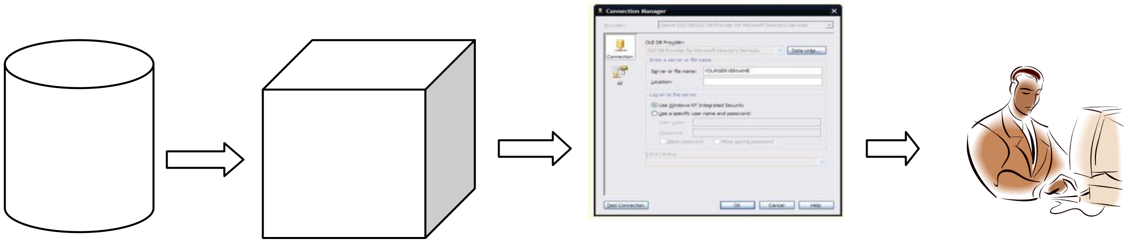
The data could be viewed at the lowest level of granularity *i.e.*, single cow and the test date without any effect on aggregate functions. A sample of a few queries which were executed on the cube is shown in Table 4.1.

	Query Examples
1	Average milk quantity of a cow in a herd on specific test day
2	Fat percentage in the milk of a cow in a herd on specific test day
3	Number of herds in a region
4	Number of cows within a herd and within a region
5	Maximum and minimum average milk quantity for all herds
6	Maximum and minimum fat percentage for all herds
7	Number of cows within a parity
8	Protein to fat ratio for all cows
9	Cows move from one herd to the other herd
10	Average values for MUN for different herds
11	Feed intake and milk production of each cow
12	Average milk production for cows in different calving seasons

Table 4.1: Query Examples

4.4.1: Front end tools to access the data from the cube

In modern database technology the front-end is separated from back-end database server by a presentation layer. These technologies consist of an OLAP client that handles the client interface and an OLAP server that administers the data and process queries. The client communicates with the server using a standardized application programming interface (API), e.g., Microsoft's OLEDB for OLAP (Figure 4.15).



Server or file name: YOURSERVERNAME

Location:

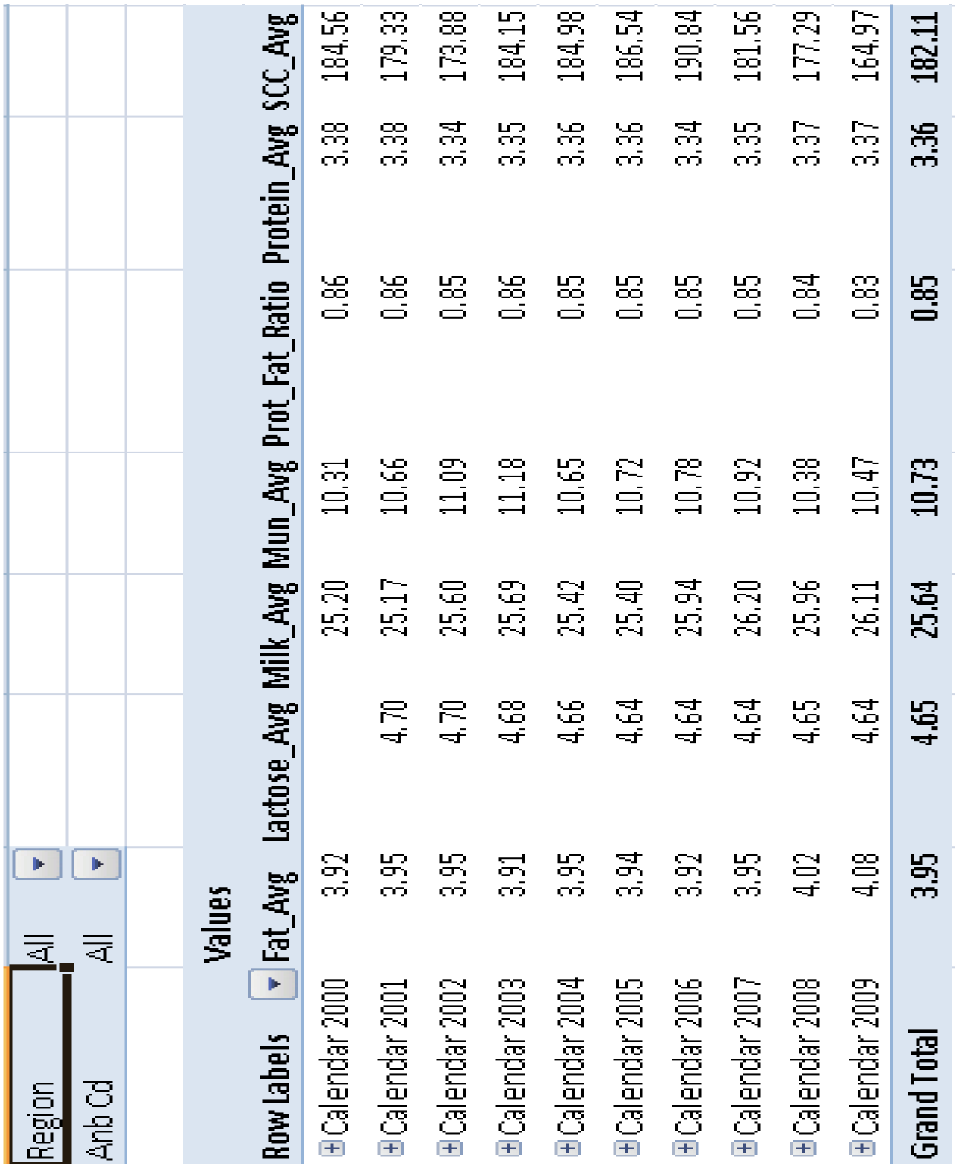
◉ Use Windows NT Integrated Security

○ Use a specific user name and password:

Figure 4.15: OALP server communicate with the user through Microsoft OLEDB

Studies have shown that queries on a data warehouse consist 80% in navigational queries that discover the dimension hierarchies and 20% aggregation queries that aggregate or summarize the data at various levels of detail (Kimball, 1996). Examples of navigational and aggregation queries are “different herds in different regions having various breeds” and correspondingly “count of cows, count of herds grouped by parities and season of calving.”

The OLAP client sends off a query to the query handler, the primary task of which is to determine whether the query is a navigational query (inner to a dimension) or an aggregation query (linking the facts) (Pedersen et al., 2001). After the status of query has been resolved, the navigational queries are passed to the OLAP server which handles the original (navigational) data, while aggregation queries are passed to another OLAP server that manages the transformed (aggregation) data. The result is that the OLAP provides an interface to the data warehouse based on a multidimensional structure. It should be noted that the main reason behind the popularity of the multidimensional data model are the spread sheets in which clients can view data with different dimensions. The spread sheet is still a very popular front-end tool for OLAP tools: they have the ability to summarize the data by applying queries that are processed on the OLAP servers. The fundamental operation supported by these spread sheets is “pivoting”. If two of the dimensions were selected then, at the intersection of each dimension, a value is present which has actually appeared due to both dimensions. For example, if regions and year were selected, then in the grid at each cell a value would show the summarized value of average milk quantity in that specific region in that specific year. Figure 4.16 shows the pivot operation in Microsoft Excel 2007.

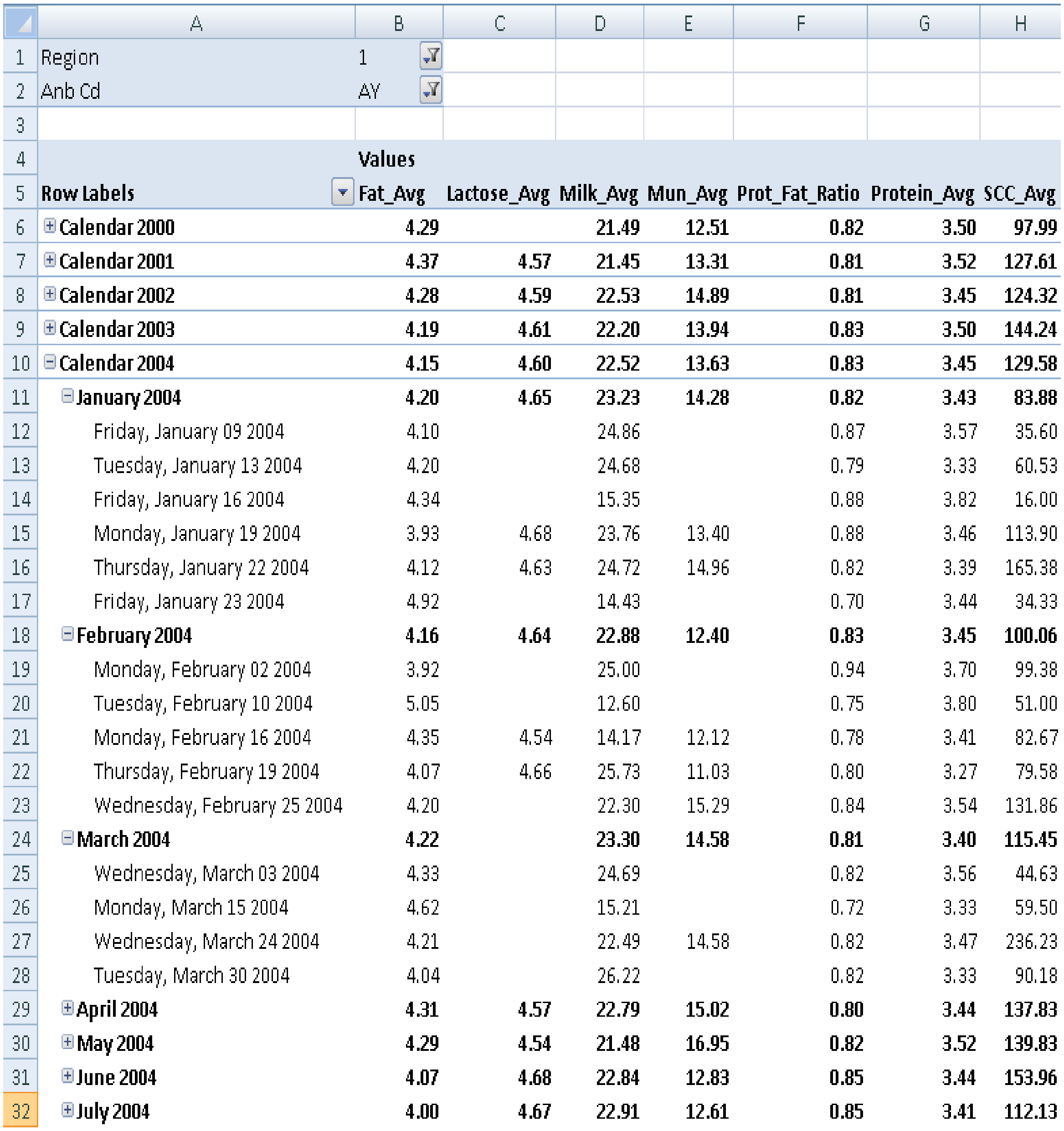


Region	All
Anb Cd	All

	Values						
Row Labels	Fat_Avg	Lactose_Avg	Milk_Avg	Mun_Avg	Prot_Fat_Ratio	Protein_Avg	SCC_Avg
⊞ Calendar 2000	3.92		25.20	10.31	0.86	3.38	184.56
⊞ Calendar 2001	3.95	4.70	25.17	10.66	0.86	3.38	179.33
⊞ Calendar 2002	3.95	4.70	25.60	11.09	0.85	3.34	173.88
⊞ Calendar 2003	3.91	4.68	25.69	11.18	0.86	3.35	184.15
⊞ Calendar 2004	3.95	4.66	25.42	10.65	0.85	3.36	184.98
⊞ Calendar 2005	3.94	4.64	25.40	10.72	0.85	3.36	186.54
⊞ Calendar 2006	3.92	4.64	25.94	10.78	0.85	3.34	190.84
⊞ Calendar 2007	3.95	4.64	26.20	10.92	0.85	3.35	181.56
⊞ Calendar 2008	4.02	4.65	25.96	10.38	0.84	3.37	177.29
⊞ Calendar 2009	4.08	4.64	26.11	10.47	0.83	3.37	164.97
Grand Total	**3.95**	**4.65**	**25.64**	**10.73**	**0.85**	**3.36**	**182.11**

Figure 4.16: Pivot table.

Pivot tables (Figure 4.17) also support operations of roll-up, drill down, slice and dice. In this figure (4.17) the milk-component averages can be seen at the test date level in region one for the Ayrshire breed. This shows how a pivot table is a two-dimensional spreadsheet which supports viewing complex data with related subtotals and totals by nesting several dimensions on the *x*- or *y*-axis (Pedersen and Jensen, 1998).



	A	B	C	D	E	F	G	H
1	Region	1						
2	Anb Cd	AY						
3								
4		**Values**						
5	**Row Labels**	**Fat_Avg**	**Lactose_Avg**	**Milk_Avg**	**Mun_Avg**	**Prot_Fat_Ratio**	**Protein_Avg**	**SCC_Avg**
6	⊞ **Calendar 2000**	**4.29**		**21.49**	**12.51**	**0.82**	**3.50**	**97.99**
7	⊞ **Calendar 2001**	**4.37**	**4.57**	**21.45**	**13.31**	**0.81**	**3.52**	**127.61**
8	⊞ **Calendar 2002**	**4.28**	**4.59**	**22.53**	**14.89**	**0.81**	**3.45**	**124.32**
9	⊞ **Calendar 2003**	**4.19**	**4.61**	**22.20**	**13.94**	**0.83**	**3.50**	**144.24**
10	⊟ **Calendar 2004**	**4.15**	**4.60**	**22.52**	**13.63**	**0.83**	**3.45**	**129.58**
11	⊟ **January 2004**	**4.20**	**4.65**	**23.23**	**14.28**	**0.82**	**3.43**	**83.88**
12	Friday, January 09 2004	4.10		24.86		0.87	3.57	35.60
13	Tuesday, January 13 2004	4.20		24.68		0.79	3.33	60.53
14	Friday, January 16 2004	4.34		15.35		0.88	3.82	16.00
15	Monday, January 19 2004	3.93	4.68	23.76	13.40	0.88	3.46	113.90
16	Thursday, January 22 2004	4.12	4.63	24.72	14.96	0.82	3.39	165.38
17	Friday, January 23 2004	4.92		14.43		0.70	3.44	34.33
18	⊟ **February 2004**	**4.16**	**4.64**	**22.88**	**12.40**	**0.83**	**3.45**	**100.06**
19	Monday, February 02 2004	3.92		25.00		0.94	3.70	99.38
20	Tuesday, February 10 2004	5.05		12.60		0.75	3.80	51.00
21	Monday, February 16 2004	4.35	4.54	14.17	12.12	0.78	3.41	82.67
22	Thursday, February 19 2004	4.07	4.66	25.73	11.03	0.80	3.27	79.58
23	Wednesday, February 25 2004	4.20		22.30	15.29	0.84	3.54	131.86
24	⊟ **March 2004**	**4.22**		**23.30**	**14.58**	**0.81**	**3.40**	**115.45**
25	Wednesday, March 03 2004	4.33		24.69		0.82	3.56	44.63
26	Monday, March 15 2004	4.62		15.21		0.72	3.33	59.50
27	Wednesday, March 24 2004	4.21		22.49	14.58	0.82	3.47	236.23
28	Tuesday, March 30 2004	4.04		26.22		0.82	3.33	90.18
29	⊞ **April 2004**	**4.31**	**4.57**	**22.79**	**15.02**	**0.80**	**3.44**	**137.83**
30	⊞ **May 2004**	**4.29**	**4.54**	**21.48**	**16.95**	**0.82**	**3.52**	**139.83**
31	⊞ **June 2004**	**4.07**	**4.68**	**22.84**	**12.83**	**0.85**	**3.44**	**153.96**
32	⊞ **July 2004**	**4.00**	**4.67**	**22.91**	**12.61**	**0.85**	**3.41**	**112.13**

Figure 4.17: Drill down and slice operation in pivot table.

Pivot tables usually support a collection of data subsets and alteration of the presentation level of detail. But spreadsheets are an insufficient tool for administration and storing multidimensional data because they bind data storage exceptionally strongly to the presentation. Spreadsheets do not separate the information about the organization of the data from the required analysis of the information (Pedersen and Jensen, 2002). Consequently adding a dimension or grouping data in the spreadsheets is quite difficult.

Likewise the ways hierarchies are supported by multidimensional databases are not supported by spread sheets. On the other hand a database management system recommends significant simplicity in organizing data. The many-to-many relations which are not supported satisfactorily by the star model can be handled adequately by snowflaking the dimensions which is ultimately helpful in managing hierarchies in dimensions.

However, a simple SQL-based relational model does not handle hierarchical dimensions satisfactorily. Nevertheless, creating numerous desired computations such as totals and subtotals, or determining rankings (top 10 herds in milk production) is complicated if not impractical in standard SQL, but is supported by spreadsheets. Also transposing columns and rows involves manual writing of several views in SQL but is possible with spreadsheets. Spreadsheets have also been used to view the raw data in multidimensional perspective, but spreadsheets and relational databases, although they provide sufficient support for small datasets with a few non-hierarchical dimensions, they do not fully support the needs for sophisticated data analysis. These observations regarding the limitations of spread sheets have been discussed by Pederson and Jenson (2002). Therefore, after performing data analysis in Microsoft Excel pivot tables the data or reports or graphs can be generated and viewed in Excel, but actual data should be stored with a database technology that offers in-built support for the full range of multidimensional data modeling for large datasets.

4.5: Updating the data warehouse

Update the data in the data warehouse would follow similar steps to those by which it was first implemented, *i.e.*, selection of data required to warehouse, cleaning of the data, transformation to get the desired variables and loading the data into the warehouse. However, these steps would be faster than before since rules would have already been set up to prepare the data to be warehoused. If some new data was added then rules can be defined for these data also. Although the first process of data preparation made later steps in updating the data easier, even then data must be careful observed to avoid any mistakes. Updates would include the addition of variables to the dimension tables as well as the addition of variables and rows to the fact tables.

All the constraints regarding referential integrity (primary key-foreign key relationship) were carefully considered and new data were verified against existing data. There were also two main things to consider: (i) how many times the data warehouse will be updated and (ii) how the data is updated and reflected in the cube. As cow's lactation length and dry period is ideally a year, so to see an effect on a cow's or a herd's performance; the data warehouse would have to be updated after a year. The cube update was also based on data being refreshed; in this case the data inside the cube was changed (new aggregations may need to be calculated at some time) but not the underlying structure. However, the possibility of introducing a new dimension exists, so in that case the full process of the cube was required, based on restructuring and recalculation of the data. It should be mentioned that the Analysis server did not introduce the combine primary key relationship in the dimensions according to our needs, so the process of introducing a combine primary key was done through the SQL server. Although the Analysis server did not produce any error message, it did double count the average milk quantity for all those cows which were transferred from one herd to another.

Conclusion

Most computer-oriented techniques have found their way into the dairy sector (Lazarus et al., 1990), and a lot of data are produced daily on dairy farms. Operational databases such as those at Valacta are not designed to support the required feedback for analytical queries, important to strategic decision-making, in an optimal manner. Therefore a methodology was developed to store these large historical datasets for analytical query execution in order to give advisors a tool that should help them to improve the decision-making process. The methodology comprises extraction, cleaning, selection and transformation of data, storing of data in a data warehouse in a multidimensional format and access to this data with the help of OLAP cubes. It is assumed that data warehouse implementation can provide benefits for storage of large historical datasets for decision-making at the operational, tactical and strategic level on dairy farms. The use of applications like data warehousing have few parallels in the dairy industry, which may be due to the complexity of software available for warehousing and lack of technical personnel in the field. The other strong reason is the lack of sufficient budgets in the public sector. However, mega datasets require customized data analysis systems, modified specifically for them, first for the analysis, and then for the presentation of results. Attention can subsequently be given to the heterogeneity and computational complexity of the data. Therefore, it is potentially possible to increase productivity in dairy research through the analysis of large datasets, by using central databases (Data warehouses) in which many data definitions and pre-processing are already implemented.

Detailed data was stored in the data warehouse for future use; the lowest granularity of the data were cow and test date. A data driven approach was preferred, given that it is the fastest method to develop a data warehouse. This method helped in data exploration and pilot analysis. Models were visualized in such a way that future users should be able to obtain helpful information if this methodology were to be implemented on a larger scale. A MOLAP server was used to store pre-computation data and store the information derived after modeling the data with multidimensional data models. The queries were executed directly on the cube structure supported by MOLAP servers. Multidimensional data models were constructed, as these models support analytical query execution (what and why queries). Comparatively, entity relationship models could not support such queries, as these models support an OLTP application, where queries are mostly restricted to insert, delete and update. Denormalized data were used to develop multidimensional models which provide: (i) a database structure that is easy for end-users to understand and write queries to these databases; and (ii) database structures which maximise the efficiency of queries by minimizing the number of tables and relationships among them. It was concluded that the snowflake schema was better than the star schema since, with the latter, denormalized data with the herd_cow_testdate (composite key of herd and cow dimension) dimension were very large, rendering the system difficult to understand and impractical for handling large sets of records. The efficiency of the MDX was found to be better with the snowflake schema. It was also concluded that any future amalgamation of data from different organizations could be facilitated by joining data with similar dimensions through the use of a fact-constellation schema.

OLAP tools are mainly developed for business applications like banking and retail. In the dairy field, data show different characteristics, so we were faced with certain challenges while developing our query procedures for certain situations. These situations were either tackled within the SQL server or with the help of MDX, since there was no built-in help for these situations in OLAP tools. For example, the highest level to which we wished to aggregate was herd, so it was necessary to define a herd-cow-bridge dimension where all the distinct records were present within a single occurrence for each herd and cow. If a cow was selected for aggregation then it was double-counted for all those appearances where a cow was transferred from one herd to the other. In this way for highest level of aggregation – herd – was selected, so that for the appearance of each cow, this was present once on that specific test date in that herd. This was not handled optimally by the software, so a composite key was defined within the SQL server as the herd_cow key, and then a named column property was set to herd in the analysis service to stop the double count of measures. Calculations based on aggregation functions such as average and percentage were transformed into new queries. Undeniably on-the-fly execution of extended queries with percentage and average functions lowered the performance of the cube, even when the multidimensional model was accurately developed. It appeared that the problem was associated with the OLAP aggregation process, and not with the hardware, since the same hardware had previously calculated the "sum" and "count" values efficiently from the same data set, for the calculation of averages. This means that if all the required aggregations were calculated in materialized views, the cube data upload response would be much faster, which otherwise is unfortunately very slow in the case of on-the-fly query execution. Developers and researchers in the field of OLAP should consider those situations where the query may not already be known.

Most of those queries performed should aid farmers in decision-making at all levels of management: strategic, tactical or operational. These queries were supported either by drill-down and roll-up operations of the cube, or by filtering options. If not handled through simple drag and drop operations, a special MDX helped in the extract of required information from the data warehouse. The OLAP tools provided multidimensional views of the data through pivot tables, but, if graphs were required, the data could be downloaded to Excel spread sheets. Although Excel spread sheets support pivoting of raw data, they do not support more complex queries or the storage of data in a multidimensional format. For a complete multidimensional storage structure, and multidimensional views of data, a sophisticated system like multidimensional databases (MOLAP) would be required.

Future work

This application is an important step towards more advanced implementation of data mining, grid computing and cloud computing which meet the needs associated with growing datasets. Data mining techniques have already been used in the dairy sector to investigate disease risk classification, as a proxy for compromised bio-security of cattle herds in Wales (Ángel and Dirk, 2008). However if these techniques were used in looking at the weather data along with feed management, they could provide benefits with respect to dairy cows' nutrient uptake during severe winter weather. Moreover, with the implementation of a national cow identification system, the Quebec dairy sector could conceivably derive benefits from grid computing to process massive amounts of cow tracking data. This type of grid would not only serve future researchers, but also government agencies, educators and public users. Although grid computing projects are long-term projects which can bring benefits, the initial investment can be quite high (Călin et al., 2011), so both government and private agencies would need to contribute funds and equipment to build an appropriate infrastructure. The grid would reduce bottlenecks that occur during data loading in the data warehouse, making the data processing faster. Speeding up the data processing means the data would be immediately available to business users for analysis, so that business decisions could arrived at in a timely manner. Also in the case of grid technology, the cost remains relatively similar since pre-existing resources are being used in a more efficient way. As in any other field, the agricultural sector evolves with the implementation of new, cutting-edge computing technologies. Consequently it is likely that the Quebec dairy sector could soon become such a hub of information where all stake holders can access data from a private cloud, with servers housed at different physical locations, all operating in a manner similar to an information junction. Analytical applications (which are a main reason for developing a data warehouse) are perhaps better suited to cloud environments (Abadi, 2009) since they require large amounts of data to be stored for long periods, but be processed infrequently. Furthermore, the process of data analysis consists of many large scans, and fact-constellation schema joins, all of which could conceptually be facilitated through parallel processing across nodes in a cloud setting.

References

1. Abadi, D.J., 2009. Data management in the cloud: Limitations and opportunities. IEEE Data Eng. Bull 32, 3-12.
2. Abdullah, A., 2009. Analysis of mealybug incidence on the cotton crop using ADSS-OLAP (Online Analytical Processing) tool. Computers and Electronics in Agriculture 69, 59-72.
3. Agrawal, R., Gupta, A., Sarawagi, S., 1997. Modeling multidimensional databases. IEEE, pp. 232-243.
4. Ahmad, I., Azhar, S., Lukauskis, P., 2004. Development of a decision support system using data warehousing to assist builders/developers in site selection. Automation in Construction 13, 525-542.
5. AL-Hamami, A.a.H., Hashem, S.H., 2009. An Approach for Facilating Knowledge Data Warehouse. International Journal of Soft Computing Applications © EuroJournals Publishing, Inc. 2009, 35-40.
6. Alkharouf, N.W., Jamison, D.C., Matthews, B.F., 2005. Online Analytical Processing (OLAP): A Fast and Effective Data Mining Tool for Gene Expression Databases. Journal of Biomedicine and Biotechnology 2005, 181-188.
7. Ángel, O.P., Dirk, P., 2008. Use of data mining techniques to investigate disease risk classification as a proxy for compromised biosecurity of cattle herds in Wales. BMC Veterinary Research 4.
8. Anil, R., Vipin, D., Chaturvedi, K.K., Malhotra, P.K., 2008. Design and development of data mart for animal resources. computers and electronics in agriculture 64, 111-119.
9. Arnott, D., Pervan, G., 2005. A critical analysis of decision support systems research. Journal of Information Technology 20, 67-87.
10. Ballard, C., Herreman, D., Schau, D., Rhonda Bell, Kim, E., Internatio, A.V., 1998. Data Modeling Techniques for Data Warehousing. International Business Machines Corporation.
11. Bauer, A., Hümmer, W., Lehner, W., 2000. An alternative relational OLAP modeling approach. Data Warehousing and Knowledge Discovery, 189-198.

12. Berndt, D., 2003. The Catch data warehouse: support for community health care decision-making. Decision Support Systems 35, 367-384.
13. Berndt, D.J., Hevner, A.R., Studnicki, J., 2003. The Catch data warehouse: support for community health care decision-making. Decision Support Systems 35, 367-384.
14. Bordoloi, B., Agarwal, A., Sircar, S., 1994. Relational or Object-oriented or Hybrid?: A Framework for Selecting an Appropriate Database Management System Type in a Computer Integrated Manufacturing Setting. International Journal of Operations & Production Management 14, 32-44.
15. Călin, M., Craus, M., Filipov, F., Chiru , C., 2011. Involving grid computing in agricultural research. Research Journal of Agricultural Science 39, 655-660.
16. Chaturvedi, K.K., Rai, A., K.Dubey, V., Malhotra, P.K., 2008. On-line Analytical Processing in Agriculture using Multidiemnsioanl Cubes. J.ind.SOc.Agril.Statist 62, 56-64.
17. Chaudhary, S., Sorathia, V., Laliwala, Z., 2004. Architecture of sensor based agricultural information system for effective planning of farm activities. IEEE, pp. 93-100.
18. Chaudhuri, S., Dayal, U., 1997. An Overview of Data Warehousing and OLAP Technology. SIGMOD 26.
19. Chaudhuri, S., Dayal, U., Ganti, V., 2001. Database technology for decision support systems. Computer, 48-55.
20. Codd, E., Codd, S., Salley, C., Codd, Date, I., 1993. Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. Codd & Date, Inc.
21. Colliat, G., 1996. OLAP, Relational, and Multidimensional Database Systems. SIGMOD Record, 25, 64-69.
22. Conn, S.S., 2005. OLTP and OLAP data integration: a review of feasible implementation methods and architectures for real time data analysis. IEEE, pp. 515-520.
23. Correa, F.E., Corrêa, P.L.P., Junior, J.R.A., Alves, L.R.A., Saraiva, A.M., 2009. Data warehouse for soybeans and corn market on Brazil, EFITA conference ’09, pp. 675-681.

24. Cui, X., 2003. A Capacity Planning Study of Database Management Systems with OLAP Workloads, School of Computing. Queen's University, Kingston, Ontario, Canada.

25. Eder, J., Koncilia, C., Morzy, T., 2006. The COMET metamodel for temporal data warehouses. Springer, pp. 83-99.

26. Elmasri, R., Navathe, S.B., 2000. Fundamentals of database systems. 3 ed. Addison Wesley.

27. Foster, I., 2002. The grid: A new infrastructure for 21st century science. Wiley Online Library, pp. 51-63.

28. Foster, I., Kesselman, C., 1999. The Grid: Blueprint for a New Computing Architecture.

29. Frawley, W., Piatetsky-Shapiro, G., Matheus, C., 1992. Knowledge discovery in databases: An overview. Ai Magazine 13, 57.

30. French, C.D., 1995. “One size fits all” database architectures do not work for DSS. ACM, pp. 449-450.

31. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H., 1997. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data Mining and Knowledge Discovery 1, 29-53.

32. Gray, P., Watson, H.J., 1998. Present and future directions in data warehousing. ACM SIGMIS Database 29, 83-90.

33. Halawani, S.M., Albidewi, I.A., Alam, J., Khan, Z., 2010. A Critical Evaluation of Relational Database Systems for OLAP Applications. International Journal of Computer and Network Security 2, 120-127.

34. Hammer, J., Garcia-Molina, H., Widom, J., Labio, W., Zhuge, Y., 1995. The Stanford DataWarehousing Project. Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society 18, 40-47.

35. Huber, P.J., 1999. Massive Datasets Workshop: Four Years after. Journal of Computational and Graphical Statistics 8, 635-652.

36. Hull, R., Zhou, G., 1996. A framework for supporting data integration using the materialized and virtual approaches. ACM.

37. Hurson, A.R., Pakzad, S.H., Cheng, J.B., 1993. Object-oriented database management systems: evolution and performance issues. Computer 26, 48-58, 60.

38. Inmon, W., 1996. Building the Data Warehouse. John Wiley & Sons, Inc., New York.

39. Jagadish, H., Lakshmanan, L.V.S., Srivastava, D., 1999. What can Hierarchies do for Data Warehouses8.

40. Jones, D.K., 1998. An Introduction to Data Warehousing: What Are the Implications for the Network? International Journal of Network Management 8, 42-56.

41. Kimball, R., 1996. The data warehouse toolkit: practical techniques for building dimensional data warehouses. John Wiley & Sons, Inc. New York, NY, USA.

42. Kimball, R., Reeves, L., Thornthwaite, W., Ross, M., Thornwaite, W., 1998. The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses with CD Rom. John Wiley & Sons, Inc., New York, NY.

43. Koutsoukis, N., Mitra, G., Lucas, C., 1999. Adapting on-line analytical processing for decision modelling: the interaction of information and decision technologies. Decision Support Systems 26, 1-30.

44. Larson, J., 1983. Bridging the gap between network and relational database management systems. Computer 16, 82-83.

45. Lazarus, W.F., Streeter, D., Jofre-Giraudo, E., 1990. Management information systems: impact on dairy farm profitability. North Central Journal of Agricultural Economics 12, 267.

46. Lehner, W., Albrecht, J., Wedekind, H., 1998. Normal forms for multidimensional databases. IEEE, pp. 63-72.

47. List, B., Bruckner, R., Machaczek, K., Schiefer, J., 2002. A comparison of data warehouse development methodologies case study of the process warehouse. Springer, pp. 203-215.

48. Mahboubi, H., Faure, T., Bimonte, S., Deffuant, G., Chanet, J.P., Pinet, F., 2010. A Multidimensional Model for Data Warehouses of Simulation Results.

International Journal of Agricultural and Environmental Information Systems (IJAEIS) 1, 1-19.

49. Maletic, J.I., Marcus, A., 2000. Data cleansing: Beyond integrity analysis. Citeseer, pp. 200-209.
50. Manole, V., Gheorghe, M., 2007. Database Vs Data Warehouse. Revista Informatica Economică 43, 91-95.
51. Martyn, T., 2004. Reconsidering Mutli-Dimensional Schemas. SIGMOD RECORD 33, 83-88.
52. McFadden, F., 2002. Data warehouse for EIS: some issues and impacts. IEEE, pp. 120-129.
53. Mishra, D., Yazici, A., Basaran, B., 2008. A casestudy of data models in data warehousing. IEEE, pp. 314-319.
54. Mohania, M., Samtani, S., Roddick, J., Kambayashi, Y., 1999. Advances and Research Directions in Data-Warehousing Technology. AJIS 7.
55. Moody, D.L., Kortink, M.A.R., 2000. From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design.
56. Pedersen, T., Jensen, C., 2002. Multidimensional database technology. Computer 34, 40-46.
57. Pedersen, T.B., Jensen, C.S., 1998. Research issues in clinical data warehousing. IEEE, pp. 43-52.
58. Pedersen, T.B., Jensen, C.S., Dyreson, C.E., 2001. A foundation for capturing and querying complex multidimensional data. Information Systems 26, 383-423.
59. Pietersma, D., Lacroix, R., Lefebvre, D., Cue, R., Wade, K., 2006. Trends in growth and age at first calving for Holstein and Ayrshire heifers in Quebec. Canadian Journal of Animal Science 86, 325-336.
60. Rahm, E., Do, H.H., 2000. Data cleaning: Problems and current approaches. Bulletin of the Technical Committee on, 3.
61. Ramick, D.C., 2001. Data warehousing in disease management programs. Journal of Healthcare Information Management 15, 99-106.
62. Reddy, P.K., GV Ramaraju, Reddy, G.S., 2007. eSaguTM: A Data Warehouse Enabled Personalized Agricultural Advisory System. SIGMOD’07, 910-914.

63. Samtani, S., Mohania, M., Kumar, V., Kambayashi, Y., 1999. Recent Advances and Research Problems in Data Warehousing. Advances in Database Technologies, 1942-1944.

64. Shim, J.P., Warkentin, M., Courtney, J.F., Power, D.J., Sharda, R., Carlsson, C., 2002. Past, present, and future of decision support technology. Decision Support Systems 33 111-126.

65. Simitsis, A., Theodoratos, D., 2009. Data Warehouse Back-End Tools. Encyclopedia of Data Warehousing and Mining, 572-579.

66. Sinha, A., Hripcsak, G., Markatou, M., 2009. Large datasets in biomedicine: a discussion of salient analytic issues. Journal of the American Medical Informatics Association 16, 759-767.

67. Stamen, J.P., 1993. Structuring databases for analysis. Spectrum, IEEE 30, 55-58.

68. Thomsen, E., 2002. OLAP solutions: building multidimensional information systems. John Wiley & Sons, Inc. New York, NY, USA.

69. Thornsbury, S., Davis, K., Minton, T., 2003. Adding value to agricultural data: a golden opportunity. Review of Agricultural Economics 25, 550-568.

70. Tjoa, A.M., Rauber, A., Tomsich, P., Wagner, R., 2007. OLAP of the Future. Informationssysteme: Daten-Information-Wissen}, 153-166.

71. Vassiliadis, P., 1998. Modeling multidimensional databases, cubes and cube operations. IEEE, pp. 53-62.

72. Vinnik, S., Mansmann, F., 2006. From analysis to interactive exploration: Building visual hierarchies from OLAP cubes. Advances in Database Technology-EDBT 2006, 496-514.

73. Wah, T.Y., Sim, O.S., 2009. Development of a data warehouse for Lymphoma cancer diagnosis and treatment decision support. WSEAS Transactions on Information Science and Applications 6, 530-543.

74. Winter, R., 1998. Databases: Back in the OLAP game. Intelligent Enterprise Magazine 1, 60-64.