Creating and Using Data Warehouses Overview

Organizations collect data in the normal course of business operations. The purpose of a data warehouse is to consolidate and organize this data so it can be analyzed and used to support business decisions. In many cases a data warehouse contains the living history of the organization.

Data warehouses usually contain historical data, often collected from a variety of disparate sources such as online transaction processing (OLTP) systems, legacy systems, text files, or spreadsheets. A data warehouse combines this data, cleanses it for accuracy and consistency, and organizes it for ease and efficiency of querying.

Some definitions of a data warehouse include several elements such as a data preparation area, the cleansing process, the database that holds the data warehouse data, and the tools that organize and present the data to client applications. Other definitions restrict the data warehouse to the database that contains the data warehouse data. In large data warehousing applications, data is often segmented into specialized components, called data marts, that address individual components of the organization. Some definitions consider data marts to be part of the data warehouse; other definitions consider them to be separate entities. The intended meaning of the term data warehouse is usually clear from the context in which it is used. The topics in this section generally use the broadest definition and address individual elements as components within the context of the data warehouse.

Data warehousing is an advanced and complex technology. A complete treatment of data warehousing is not possible in this document, but many excellent books and training courses are available to enhance your understanding. The topics in this section discuss the elements and processes of data warehousing and identify the many powerful tools provided by Microsoft® SQL ServerTM 2000 to help you in the task of creating, using, and maintaining a data warehouse.

Торіс	Description
SQL Server 2000 Tools for	Describes tools provided by SQL Server 2000
<u>Data Warehouses</u>	that are commonly used in data warehouse
	applications.

Parts of a Data Warehouse	Describes the elements that make up a data
	warehouse.
Creating a Data Warehouse	Describes the steps in creating a data
	warehouse.
Using a Data Warehouse	Describes the tools and methods used to
	prepare data for presentation and to provide
	client access to the information.
Maintaining a Data	Describes the processes involved in updating
<u>Warehouse</u>	data in the data warehouse and modifying the
	presentation of information to users.

SQL Server 2000 Tools for Data Warehouses

Microsoft® SQL Server[™] 2000 provides many tools that support database applications. Some of these tools are used more often than others in data warehouse applications, and some are specifically designed to address special needs of data warehouses. The tools listed here are commonly used in data warehouse applications, although most are also applicable to other database applications. Many other tools not mentioned here can often be used to solve specific problems in data warehouse applications.

General descriptions of the tools commonly used in data warehouse applications are provided here with links to more detailed information about the tools themselves. The uses of these tools in data warehouse applications are specifically discussed in other topics in this section.

Relational Database

Data warehouses use relational database technology as the foundation for their design, construction, and maintenance. The core component of SQL Server 2000 is a powerful and full-featured relational database engine. SQL Server 2000 provides many tools for design and manipulation of relational databases, regardless of the applications for which the databases are used. Information about the many relational database management tools is provided throughout the SQL Server 2000 documentation. For more information, see <u>SQL Server 2000</u> <u>Features</u>.

Data Transformation Services

Data warehouse applications require the transformation of data from many sources into a cohesive, consistent set of data configured appropriately for use in data warehouse operations. SQL Server 2000 provides a powerful tool for such tasks, Data Transformation Services (DTS). DTS can access data from a wide variety of sources and transform it using built-in and custom transformation specifications. For more information, see <u>DTS Overview</u>.

Replication

Database replication is a powerful tool with many uses. Often used to distribute data and coordinate updates of distributed data in online transaction processing systems (OLTP), replication can also be used in data warehouses. Some potential data warehouse applications of replication are the distribution of data from a central data warehouse to data marts, and the updating of data warehouse data from the data preparation area. For more information, see <u>Replication Overview</u>.

Analysis Services

Data warehouses collect and organize enterprise data to support organizational decision-making through analysis. SQL Server 2000 Analysis Services provides online analytical processing (OLAP) technology to organize massive amounts of data warehouse data for rapid analysis by client tools, and sophisticated data mining technology to analyze and discover information within the data warehouse data. For more information, see <u>Analysis Services Overview</u>.

English Query

English Query provides access to data warehouse data using English language queries such as "Show me the sales for stores in California for 1996 through 1998." English Query is a development tool for creating client applications that transform English language into the syntax of SQL to query relational databases or the syntax of Multidimensional Expressions (MDX) to query OLAP cubes. You can develop English Query models specific to your data warehouse to reduce sophisticated and complex SQL or MDX queries to simple English questions. For more information, see English Query Overview.

Meta Data Services

Many of the various tools in SQL Server 2000 store meta data in a centralized repository in the **msdb** system database. SQL Server 2000 Meta Data Services provides a browser for viewing this meta data and application interfaces for use in developing custom meta data applications. For more information, see <u>Meta Data Services Overview</u>.

Parts of a Data Warehouse

There are several physical and functional elements that make up a data warehouse. The topics in this section discuss these elements.

Торіс	Description
Data Marts	Describes data marts, which contain portions of
	data warehouse data for specialized purposes.
Relational Databases	Describes the roles and uses of relational
	databases in data warehouses.
Data Sources	Describes various sources of organizational data
	typically used in data warehouses.
Data Preparation Area	Describes the area where data extracted from data
	sources is prepared for use in a data warehouse.
Presentation Services	Describes the services that organize and analyze
	data warehouse information and make it available
	to client applications.
End-User Analysis	Describes the use of client applications to access
	and analyze information in a data warehouse.

Data Marts

In some data warehouse implementations, a data mart is a miniature data warehouse; in others, it is just one segment of the data warehouse. Data marts are often used to provide information to functional segments of the organization. Typical examples are data marts for the sales department, the inventory and shipping department, the finance department, upper level management, and so on. Data marts can also be used to segment data warehouse data to reflect a geographically compartmentalized business in which each region is relatively autonomous. For example, a large service organization may treat regional operating centers as individual business units, each with its own data mart that contributes to the master data warehouse.

Data marts are sometimes designed as complete individual data warehouses and contribute to the overall organization as a member of a distributed data warehouse. In other designs, data marts receive data from a master data warehouse through periodic updates, in which case the data mart functionality is often limited to presentation services for clients.

Regardless of the functionality provided by data marts, they must be designed as components of the master data warehouse so that data organization, format, and schemas are consistent throughout the data warehouse. Inconsistent table designs, update mechanisms, or dimension hierarchies can prevent data from being reused throughout the data warehouse, and they can result in inconsistent reports from the same data. For example, it is unlikely that summary reports produced from a finance department data mart that organizes the sales force by management reporting structure will agree with summary reports produced from a sales department data mart that organizes the sales force by geographical region. It is not necessary to impose one view of data on all data marts to achieve consistency; it is usually possible to design consistent schemas and data formats that permit rich varieties of data views without sacrificing interoperability. For example, the use of a standard format and organization for time, customer, and product data does not preclude data marts from presenting information in the diverse perspectives of inventory, sales, or financial analysis.

Data marts should be designed from the perspective that they are components of the data warehouse regardless of their individual functionality or construction.

This provides consistency and usability of information throughout the organization.

Microsoft® SQL Server[™] 2000 tools used for a data mart may include any of the tools used for data warehouses depending on how the data mart is designed. If the data mart is created and maintained locally and participates in the organization's data warehouse as an independent contributor, its creation and maintenance will involve all the operations of a data warehouse. If the data mart is a local access point for data distributed from a central data warehouse, only a subset of the tools may be involved.

Distributing Data Warehouse Data to Data Marts

If data warehouse data is maintained in a central data warehouse, the data is prepared and loaded into the data warehouse at the central site and then distributed to local data marts.

SQL Server Agent and Data Transformation Services (DTS) can be used to schedule and perform data transfers, including filtering data appropriate to the data mart and updating the appropriate tables in the data mart. DTS packages can also be created and scheduled to update OLAP cubes in the data mart after new data is received from the central data warehouse.

Some data warehouse distribution scenarios may also use replication to coordinate and maintain data mart data.

See Also

DTS Overview SQLServerAgent Service Replication Overview

Relational Databases

Relational databases are the underlying engines that provide power to data warehouses. Many of the characteristics and features that have been developed and enhanced to make relational databases the workhorses of online transactional processing (OLTP) systems are directly applicable to data warehouses.

Relational databases are used in data warehouse systems to stage, cleanse, and transform incoming data in the data preparation database, contain and manage the massive quantities of data in the data warehouse database, and support data marts.

Data warehouses store, manage, and manipulate huge quantities of data, often on the order of hundreds of millions of rows of historical information. The relational database must provide rapid data transfer and update, flexible and efficient indexing, and sophisticated and effective query capabilities to organize and retrieve data warehouse data. Sophisticated locking mechanisms and high multi-table transaction throughput may be more important in OLTP systems than in data warehouses, but such features are often based on extremely efficient relational engine design, which is very important in data warehouse operations.

Microsoft® SQL Server[™] 2000 provides an extremely powerful relational database for OLTP systems and data warehouse data storage. It also includes many powerful features critical to data warehouses, such as Data Transformation Services (DTS), replication management, SQL Server 2000 Analysis Services with its multidimensional online analytical processing (OLAP) and data mining server and management support, SQL Server 2000 Meta Data Services, and English Query for natural language querying of both relational and multidimensional data.

See Also

<u>Creating and Maintaining Databases Overview</u> <u>Optimizing Database Performance Overview</u>

Data Sources

Data warehouses are intended to provide information to decision makers. To do so, data warehouses must gather and consolidate data from many sources in the organization into a consistent set of data that accurately reflects the organization's business operation and history.

Organizations often have multiple online transaction processing (OLTP) systems to capture daily business operations. These OLTP systems are seldom designed at the same time as data warehouses. They may even be designed by different organizations, which is often the case when organizations grow through acquisitions and mergers. Database schemas and data element identification keys often vary from database to database. For example, the customer table in the OLTP of an acquired company may contain many of the same customers and products as the acquiring company but use a different identification system. Data extracted from these OLTP systems must be transformed into a common representation.

Legacy systems that have been in use for many years often contain denormalized data as well as unusual data identification designs and limited query flexibility.

Data critical for business analysis may even reside on individual desktop computers in personal databases and spreadsheets, especially in organizations that developed and grew without a central information technology group. Such data must also be captured into the data warehouse.

Sources of data to be used in the data warehouse must be identified and techniques developed for extracting the data from them. Data Transformation Services (DTS) provides powerful tools for extracting and transforming data from diverse data sources. For more information, see <u>DTS Overview</u> and <u>DTS Basics</u>.

Data Preparation Area

Data to be used in the data warehouse must be extracted from the data sources, cleansed and formatted for consistency, and transformed into the data warehouse schema. The data preparation area, sometimes called the data staging area, is a relational database into which data is extracted from the data sources, transformed into common formats, checked for consistency and referential integrity, and made ready for loading into the data warehouse database. The data preparation area and the data warehouse database can be combined in some data warehouse implementations as long as the cleansing and transformation operations do not interfere with the performance or operation of serving the end users of the data warehouse data. Performing the preparation operations in source databases is rarely an option because of the diversity of data sources and the processing load that data preparation can impose on online transaction processing systems. The relational database used for data preparation, regardless of where it is performed, must have powerful data manipulation and transformation capabilities such as those provided by Microsoft® SQL Server™ 2000.

After the initial load of a data warehouse, the data preparation area is used in an ongoing basis to prepare new data for updating the data warehouse. In most data warehouse systems, these ongoing operations are performed on a periodic basis, often scheduled to minimize performance impact on the operational data source systems.

The use of a data preparation area that is separated from the data sources and the data warehouse promotes effective data warehouse management. Attempting to transform data in the data source systems can interfere with online transaction processing (OLTP) performance, and many legacy systems do not have effective or easily implemented transformation capabilities. Reconciliation of inconsistencies in data extracted from various sources can rarely be accomplished until the data is collected in a common database, at which time data integrity errors can more easily be identified and rectified.

The data preparation area should isolate raw data from the data warehouse data to preserve the integrity of the data warehouse and permit it to perform its primary function of preparing information for presentation and supporting access by clients. If the data warehouse database is used for data preparation, care should be taken to avoid introducing errors into the data warehouse data and to minimize the effect of data preparation processing on the performance of the data warehouse. Many data warehouse database operations require sophisticated queries and the processing of large amounts of data; data cleansing can interfere with these operations.

The data preparation area is a relational database that serves as a general work area for the data preparation operations. It will contain tables that relate source data keys to surrogate keys used in the data warehouse, tables of transformation data, and many temporary tables. It will also contain the processes and procedures, such as Data Transformation Services (DTS) packages, that extract data from source data systems.

See Also

Creating and Maintaining Databases Overview

Accessing and Changing Relational Data Overview

Presentation Services

The purpose of a data warehouse is to expose business information for use by decision makers in the organization. A data warehouse containing hundreds of millions of pieces of data is of no use to a decision maker without tools to assist in the analysis and evaluation process. These analysis tools can vary from simple reports to sophisticated data mining algorithms. Application programming interfaces (APIs) must also be available to support the development of custom applications that use data warehouse information.

Predefined Reports

Simple predefined summary reports can provide managers with periodic or ondemand snapshots of the state of the business at a point in time. More sophisticated reports can display trends of predetermined business variables. Such reports are useful and have historically been produced from online transaction (OLTP) systems. To capture up-to-the-minute status, snapshot detail and summary reports must continue to be produced from the data source systems. Periodic reports that are coordinated with data warehouse updates can be shifted to the data warehouse to reduce loads on operational systems. Reports that use historical data to evaluate trends should be accomplished in the data warehouse, which contains readily available historical data in appropriate formats, and which is designed to process large quantities of data for summarization.

You can prepare predefined reports in your Microsoft® SQL Server[™] 2000 data warehouse by developing client applications that access either the relational database or multidimensional data cubes prepared by SQL Server 2000 Analysis Services. For more information, see <u>Building SQL Server Applications</u> <u>Overview</u>.

Online Analytical Processing

Predefined reports serve their specific purposes well but are not suited to explorative analysis. Analysts want to discover trends and anomalies in the data and explore various areas of the data to find the sources of these trends and anomalies. Online analytical processing (OLAP) is a tool designed to facilitate this kind of analysis of massive amounts of data warehouse data.

This example illustrates explorative analysis and OLAP. A manager notices a larger than normal sales amount in a predefined summary report and wants to find the cause. The cause may be external to the business, such as an unusual weather condition that drove the sales of related merchandise, or internal, such as a sales promotion. The increased sales may be spread across a geographic region or isolated to a single store, and they may have occurred over a short or relatively long period of time.

To explore data in the data warehouse, the manager asks questions of the data and then asks related or different questions based on the answers. The manager may first ask for the sales data summarized by week for the quarter in question, then drill down to the days of the week to find that the unusual sales amount is a one-time event. Having discovered the time of the event, the manager then explores the sales by product and finds that a large sale of a specific product was made on that day. Based on these answers, the manager then asks for the sales by region and drills down to find that a particular store filled a large order of a specific product on a specific day.

OLAP technology has been developed to facilitate this kind of explorative analysis. Analysis Services includes an Analysis server that uses OLAP technology to prepare large quantities of data warehouse data for exploration in real time. Multidimensional data structures called cubes are predefined and created to organize and summarize data warehouse data in such a way that typical explorative analysis questions can be answered with little or no querying of the relational database. In a typical Analysis Services implementation in a data warehouse, the manager in the example could find the answer in less than a minute because the Analysis server can answer each of the example questions in a second or two, even if there were millions of items in the data being explored. For more information, see <u>Analysis Services Overview</u>.

Data Mining

In contrast to OLAP, which organizes data into predefined multidimensional structures to facilitate exploration, data mining performs the explorative analysis and identifies interesting nuggets of information such as groupings of data for the analyst or manager to examine. Data mining can also create decision trees that can be used to predict future data based on attributes of existing data elements.

Analysis Services incorporates sophisticated data mining algorithms that can be used to analyze data in the relational database or in OLAP cubes. The results of the data mining analysis can also be used with OLAP cubes to enhance explorative analysis. For example, you can let data mining find groupings of customers according to their attributes and then use these groupings to create an additional dimensional view of OLAP data cubes and explore the data from the perspective of these groupings. For more information, see <u>Analysis Services</u> <u>Overview</u>.

Application Programming Interfaces

SQL Server 2000 provides a number of APIs that can be used to develop client applications tailored to your data warehouse needs. Some APIs provide access to database and tool object models so that custom administrative applications can be developed. Other APIs provide access to data through standard interfaces such as OLE DB for use by custom end-user applications. For more information, see <u>Building SQL Server Applications Overview</u>.

End-User Analysis

Microsoft® SQL Server[™] 2000 and its components provide an open environment for access to data warehouse data. This offers opportunities for independent software vendors to develop sophisticated data analysis and presentation applications for end users. Many third-party client applications that work with SQL Server 2000 and its components are available.

SQL Server 2000 also integrates well with Microsoft Office 2000 to provide end users with easy to use tools for analyzing data warehouse data. Using components of Microsoft Office 2000 you can query SQL Server 2000 databases to incorporate data warehouse data into Microsoft Excel spreadsheets, Microsoft Access databases, or other documents. Excel 2000 PivotTables® can connect directly to SQL Server 2000 Analysis Services cubes to explore data, and users can create local cubes to take with them when offline from the data warehouse.

Creating a Data Warehouse

Creating a data warehouse is a significant project with a number of steps. The topics in this section address these steps.

Торіс	Description
<u>Designing a Data</u> Warehouse	Describes considerations specific to designing data warehouses and the use of dimensional
	modeling.
<u>Creating the Data</u>	Describes the creation of the relational
Preparation Area	database used to prepare data for the data
	warehouse.
<u>Creating the Data</u>	Describes the creation of the relational
<u>Warehouse Database</u>	database that holds the data warehouse data.
Extracting Data from	Describes the process of extracting data from
Operational Systems	operational systems into the data preparation
	area.
<u>Cleansing and</u>	Describes the process of cleansing and
<u>Transforming Data</u>	transforming data in the data preparation area
	before loading the data into the data
	warehouse.
Loading Data into the Data	Describes the process of loading data into the
<u>Warehouse Database</u>	data warehouse database from the data
	preparation area.
Preparing Presentation	Describes the process of preparing data in the
<u>Information</u>	data warehouse for presentation to users.
Distributing Data to Data	Describes the process of distributing data from
<u>Marts</u>	the data warehouse to data marts.

Designing a Data Warehouse

Designing a data warehouse is very different from designing an online transaction processing (OLTP) system. In contrast to an OLTP system in which the purpose is to capture high rates of data changes and additions, the purpose of a data warehouse is to organize large amounts of stable data for ease of analysis and retrieval. Because of these differing purposes, there are many considerations in data warehouse design that differ from OLTP database design.

Data warehouse data must be organized to meet the purpose of the data warehouse, which is rapid access to information for analysis and reporting. Dimensional modeling is used in the design of data warehouse databases to organize the data for efficiency of queries that are intended to analyze and summarize large volumes of data. The data warehouse schema is almost always very different and much simpler than the schema of an OLTP system designed using entity-relation modeling.

Verification tables used in OLTP systems to validate data entry transactions are not necessary in the data warehouse database. This is because the data warehouse data has been cleansed and verified before it is posted to the data warehouse database, and historical data is not expected to change frequently once it is in the data warehouse.

Transaction locking considerations, and transactions themselves, play very small roles in data warehouse databases. OLTP systems specialize in large volumes of data update transactions. In contrast, data warehouses specialize in rapid retrieval of information from stable data, and data updates consist primarily of periodic additions of new data.

Backup and restore strategies also differ in a data warehouse from those necessary for an OLTP system. Much of the data in a data warehouse is unchanging history and does not need repetitive backup. Backup of new data can be accomplished at the time of update, and in some situations it is feasible to do these backups from the data preparation database to minimize performance impact on the data warehouse database. Restore policies for a data warehouse might also differ from those for an OLTP, depending on how critical it is for an organization to have uninterrupted access to data warehouse data. There are some considerations to take into account when designing the data warehouse if you are planning to use Microsoft® SQL Server[™] 2000 Analysis Services for OLAP and data mining. For more information, see <u>Analysis</u> <u>Services Overview</u> and <u>OLAP and Data Warehouses</u>.

Data Mart Design

There are two approaches to creating a data warehouse system for an organization. A central data warehouse can be developed and implemented first with data marts created later, or data marts can be implemented such that they make up the data warehouse when their information is joined. In either approach, design must be centralized so that all of the organization's data warehouse information is consistent and usable. Data marts that adhere to central design specifications produce reports that are consistent even though the data resides in different places. For example, a sales data mart must use the same product table arranged in the same way as the inventory data mart or summary information will be inconsistent between the two.

See Also

Using Dimensional Modeling Fact Tables Aggregation Tables Dimension Tables Indexes

Using Dimensional Modeling

Entity-relation modeling is often used to create a single complex model of all of the organization's processes. This approach has proven effective in creating efficient online transaction processing (OLTP) systems. In contrast, dimensional modeling creates individual models to address discrete business processes. For example, sales information may go to one model, inventory to another, and customer accounts to yet another. Each model captures facts in a fact table and attributes of those facts in dimension tables linked to the fact table. The schemas produced by these arrangements are called star or snowflake schemas, and have been proven effective in data warehouse design.

Dimensional modeling organizes information into structures that often correspond to the way analysts want to query data warehouse data. For example, the question, "What were the sales of food items in the northwest region in the third quarter of 1999?" represents the use of three dimensions (product, geography, time) to specify the information to be summarized.

A Data Warehouse Model

A simple dimensional model of sales information might include a fact table named **Sales_Fact** that contains one record for each line item of each sale, capturing the quantity sold, the unit cost, and the sale value. Varieties of information about a sales record might include the customer, the store where the sale occurred, the time and date of the sale, and the product sold. Each of these categories of information is organized into its own dimension table. Customer information is placed in a **Customer** dimension table, store information in a **Store** dimension table, time and date information in a **Time** dimension table, and product information in a **Product** dimension table.

In a star schema, each dimension table has a single-part primary key that links to one part of the multipart primary key in the fact table. In a snowflake schema, one or more dimension tables are decomposed into multiple tables with the subordinate dimension tables joined to a primary dimension table instead of to the fact table. In most designs, star schemas are preferable to snowflake schemas because they involve fewer joins for information retrieval and are easier to manage.

Fact Tables

Each data warehouse or data mart includes one or more fact tables. Central to a star or snowflake schema, a fact table captures the data that measures the organization's business operations. A fact table might contain business sales events such as cash register transactions or the contributions and expenditures of a nonprofit organization. Fact tables usually contain large numbers of rows, sometimes in the hundreds of millions of records when they contain one or more years of history for a large organization.

A key characteristic of a fact table is that it contains numerical data (facts) that can be summarized to provide information about the history of the operation of the organization. Each fact table also includes a multipart index that contains as foreign keys the primary keys of related dimension tables, which contain the attributes of the fact records. Fact tables should not contain descriptive information or any data other than the numerical measurement fields and the index fields that relate the facts to corresponding entries in the dimension tables.

In the **FoodMart 2000** sample database provided with Microsoft® SQL Server[™] 2000 Analysis Services, one fact table, **sales_fact_1998**, contains the following columns.

Column	Description
product_id	Foreign key for dimension table product .
time_id	Foreign key for dimension table time_by_day .
customer_id	Foreign key for dimension table customer .
promotion_id	Foreign key for dimension table promotion .
store_id	Foreign key for dimension table store .
store_sales	Currency column containing the value of the sale.
store_cost	Currency column containing the cost to the store of the
	sale.
unit_sales	Numeric column containing the quantity sold.

In this fact table, each entry represents the sale of a specific product on a specific day to a specific customer in accordance with a specific promotion at a specific store. The business measurements captured are the value of the sale, the cost to

the store, and the quantity sold.

The most useful measures to include in a fact table are numbers that are additive. Additive measures allow summary information to be obtained by adding various quantities of the measure, such as the sales of a specific item at a group of stores for a particular time period. Nonadditive measures such as inventory quantityon-hand values can also be used in fact tables, but different summarization techniques must then be used.

Aggregation in Fact Tables

Aggregation is the process of calculating summary data from detail records. It is often tempting to reduce the size of fact tables by aggregating data into summary records when the fact table is created. However, when data is summarized in the fact table, detailed information is no longer directly available to the analyst. If detailed information is needed, the detail rows that were summarized will have to be identified and located, possibly in the source system that provided the data. Fact table data should be maintained at the finest granularity possible. Aggregating data in the fact table should only be done after considering the consequences.

Mixing aggregated and detailed data in the fact table can cause issues and complications when using the data warehouse. For example, a sales order often contains several line items and may contain a discount, tax, or shipping cost that is applied to the order total instead of individual line items, yet the quantities and item identification are recorded at the line item level. Summarization queries become more complex in this situation, and tools such as Analysis Services often require the creation of special filters to deal with the mixture of granularity.

There are two approaches that can be used in this situation. One approach is to allocate the order level values to line items based on value, quantity, or shipping weight. Another approach is to create two fact tables, one containing data at the line item level, the other containing the order level information. The order identification key should be carried in the detail fact table so the two tables can be related. The order table can then be used as a dimension table to the detail table, with the order-level values considered as attributes of the order level in the dimension hierarchy.

Aggregation Tables

Aggregation tables are tables that contain summaries of fact table information. These tables are used to improve query performance when SQL is used as the query mechanism. OLAP technology, such as that provided by Microsoft® SQL Server™ 2000 Analysis Services, eliminates the need for such tables. Analysis Services creates OLAP cubes that contain preaggregated summaries so that queries can be answered quickly, regardless of the level of summarization required to answer the query. It is not necessary to create aggregation tables in the data warehouse when Analysis Services is used to provide presentation services. Analysis Services creates aggregations as necessary and stores them in tables in the data warehouse database or in internal multidimensional structures. For more information, see <u>Analysis Services Overview</u>.

Dimension Tables

Dimension tables contain attributes that describe fact records in the fact table. Some of these attributes provide descriptive information; others are used to specify how fact table data should be summarized to provide useful information to the analyst. Dimension tables contain hierarchies of attributes that aid in summarization. For example, a dimension containing product information would often contain a hierarchy that separates products into categories such as food, drink, and nonconsumable items, with each of these categories further subdivided a number of times until the individual product SKU is reached at the lowest level.

Dimensional modeling produces dimension tables in which each table contains fact attributes that are independent of those in other dimensions. For example, a customer dimension table contains data about customers, a product dimension table contains information about products, and a store dimension table contains information about stores. Queries use attributes in dimensions to specify a view into the fact information. For example, a query might use the product, store, and time dimensions to ask the question "What was the cost of nonconsumable goods sold in the northeast region in 1999?" Subsequent queries might drill down along one or more dimensions to examine more detailed data, such as "What was the cost of kitchen products in New York City in the third quarter of 1999?" In these examples, the dimension tables are used to specify how a measure (cost) in the fact table is to be summarized.

Columns in a dimension table can be used to categorize the information into hierarchical levels. For example, a dimension table for stores in the **FoodMart 2000** sample database includes the following columns that specify the hierarchy levels.

Column	Description
store_country	Specifies the country in which the store is located. This
	is the country level of the hierarchy.
store_state	Specifies the state in which the store is located. This is
	the state level of the hierarchy.
store_city	Specifies the city in which the store is located. This is

	the city level of the hierarchy.	
store_id	Specifies the individual store. This is the lowest level of the hierarchy. This field contains the primary key of the	
	store dimension table and is used to join the dimension table to the fact table.	
store_name	Specifies the name of the store. The values in this column are used to identify the store to users in a readable form.	

Other columns not shown provide additional attribute information. For information about how dimension tables are used in OLAP cubes built using Microsoft® SQL Server[™] 2000 Analysis Services, see <u>Dimensions</u>.

Varieties of Dimension Tables

The preceding example illustrates a dimension table that contains a balanced hierarchy that is separated into regular levels. Other types of dimension tables contain less balanced information, such as part-breakdown structures or organization charts in which the hierarchy is represented by parent-child relationships instead of an array of levels.

Surrogate Keys

It is important that primary keys of dimension tables remain stable. It is strongly recommended that surrogate keys be created and used for primary keys for all dimension tables. Surrogate keys are keys that are maintained within the data warehouse instead of keys taken from source data systems. There are several reasons for the use of surrogate keys:

• Data tables in various source systems may use different keys for the same entity.

Legacy systems that provide historical data might have used a different numbering system than a current online transaction processing system. A surrogate key uniquely identifies each entity in the dimension table regardless of its source key. A separate field can be used to contain the key used in the source system.

Systems developed independently in company divisions may not use the

same keys, or they may use keys that conflict with data in the systems of other divisions. This situation may not cause problems when each division independently reports summary data, but it cannot be permitted in the data warehouse where data is consolidated.

• Keys may change or be reused in the source data systems.

This situation is usually less likely than others, but some systems have been known to reuse keys belonging to obsolete data. However, the key may still be in use in historical data in the data warehouse, and the same key cannot be used to identify different entities.

• Changes in organizational structures may move keys in the hierarchy.

This can be a common situation. For example, if a salesperson is transferred from one region to another, the company may prefer to track two things: sales data for the salesperson with the person's original region for data prior to the transfer date, and sales data for the salesperson in the person's new region after the transfer date. To represent this organization of data, the salesperson's record must exist in two places in the sales force dimension table, which is not possible if the salesperson's company employee identification number is used as the primary key for the dimension table. A surrogate key allows the same salesperson to participate in different locations in the dimension hierarchy.

In this case, the salesperson will be represented twice in the dimension table with two different surrogate keys. These surrogate keys are used to join the salesperson's records to the sets of facts appropriate to the various locations in the hierarchy occupied by the salesperson.

The employee's identification number should be carried in a separate column in the table so information about the employee can be reviewed or summarized regardless of the number of times the employee's record appears in the dimension table.

Dimensions that exhibit this type of change are called slowly changing dimensions.

Another example of a situation that causes this type of change is the creation of a new version of a product, such as a reduced-fat version of

a food item. The item will receive a new SKU or Uniform Product Code (UPC), but may retain most of the same attributes of the original item, which is still manufactured and sold. The appropriate use of surrogate keys can allow the two versions of the item to be summarized together or separately.

The implementation and management of surrogate keys is the responsibility of the data warehouse. OLTP systems are rarely affected by these situations, and the purpose of these keys is to accurately track history in the data warehouse. Surrogate keys are maintained in the data preparation area during the data transformation process.

Referential Integrity

Referential integrity must be maintained between all dimension tables and the fact table. Each fact record contains foreign keys that relate to primary keys in the dimension tables. Every fact record must have a related record in every dimension table used with that fact table. Missing records in a dimension table can cause facts to be ignored when the dimension table is joined to the fact table to respond to queries or for the population of OLAP cubes. Queries can return inconsistent results if records are missing in one or more dimension tables. Queries that join a defective dimension table to the fact table will exclude facts whereas queries that do not join the defective dimension table will include those facts.

Shared Dimensions

A data warehouse must provide consistent information for similar queries. One method to maintain consistency is to create dimension tables that are shared and used by all components and data marts in the data warehouse. Candidates for shared dimensions include customers, time, products, and geographical dimensions such as the store dimension in the example earlier in this topic. For example, requiring that all OLAP cubes and data marts use the same shared time dimension enforces consistency of results summarized by time.

Indexes

Indexes play an important role in data warehouse performance, as they do in any relational database. Every dimension table must be indexed on its primary key. Indexes on other columns such as those that identify levels in the hierarchical structure can also be useful in the performance of some specialized queries.

The fact table must be indexed on the composite primary key made up of the foreign keys of the dimension tables.

These are the primary indexes needed for most data warehouse applications because of the simplicity of star and snowflake schemas. Special query and reporting requirements may indicate the need for additional indexes.

See Also

Creating and Maintaining Databases Overview

Creating the Data Preparation Area

You will need to create tables and other database objects to support the data extraction, cleansing, and transformation operations required to prepare the data for loading into the data warehouse. You can create a separate database for the data preparation area, or you can create these items in the data warehouse database.

The data preparation area should include tables to contain the incoming data, tables to aid in implementing surrogate keys, and tables to hold transformed data. Other tables may be required for reconciling data from diverse data sources; such tables may contain cross-reference information to identify common entities such as customer records from systems that use different keys. A variety of temporary tables may also be needed for intermediate transformations.

The specific design of the data preparation area will depend on the diversity of data sources, the degree of transformation necessary to organize the data for data warehouse loading, and the consistency of the incoming data.

Data that is ready to load into the data warehouse should be in tables that have schemas identical to the target tables in the data warehouse. If not, the data should be ready to load into the data warehouse tables through a transformation that can be accomplished in a single step as it is loaded.

The data preparation area should also contain the processes that are used to extract the data from the data sources, the processes that transform and cleanse the data, and the processes that load the data to the data warehouse. These processes may be in the form of SQL queries, stored procedures, Data Transformation Services (DTS) packages, or documents of manual instructions. As in the development of any database system, the objective is to automate as much of the process as possible and to manage and maintain the automated tools developed. Storing and maintaining the transformation processes in the data preparation area permits the use of standard database backup and restore mechanisms to preserve them.

Regardless of whether a separate database is used, creating the data preparation area involves creating tables, views, indexes, DTS packages, and other elements

common to relational databases.

See Also

Data Preparation Area

Creating and Maintaining Databases Overview

Creating the Data Warehouse Database

You can create the data warehouse database after the data warehouse schema has been designed. You will need to create tables for facts and dimensions, and establish indexes on key fields in all tables.

The data warehouse database schema is often quite simple compared to those of OLTP databases or the data preparation area. A star schema consists of a single fact table and a number of dimension tables. A snowflake schema adds secondary dimension tables. More complex data warehouses may contain multiple fact tables and a number of dimension tables, some of which are common to all fact tables and others that are specific to a single fact table.

For example, a data warehouse may contain both sales information and inventory information. Because sales data and inventory data are different in nature, they should be stored in different fact tables. Some dimension tables, such as a product dimension table, might be common to both sales and inventory, whereas others, such as sales force or warehouse location, might be specific to individual fact tables.

The **FoodMart 2000** sample database provided with Microsoft® SQL Server[™] 2000 Analysis Services illustrates a data warehouse that contains both inventory and sales data. For more information, see <u>Analysis Services Overview</u>.

See Also

Designing a Data Warehouse

Creating and Maintaining Databases Overview

Extracting Data from Operational Systems

Data that will be used in a data warehouse must be extracted from the operational systems that contain the source data. Data is initially extracted during the data warehouse creation, and ongoing periodic extractions occur during updates of the data warehouse. Data extraction can be a simple operation, if the source data resides in a single relational database, or a very complex operation, if the source data resides in multiple heterogeneous operational systems. The goal of the data extraction process is to bring all source data into a common, consistent format so it can be made ready for loading into the data warehouse.

It is better if data in the source operational systems does not contain validation errors. For example, purchase records for which there are no corresponding customer records to identify the purchasers are clearly errors in the source data, and you should correct them in the source operational system before the data is extracted for loading into the data warehouse. You may be able to implement error checking in the source operational system so such errors can be detected before extracting data for the data warehouse. If such errors are frequent, you may need to have the operational system examined and modified to reduce such errors because such errors may affect the organization's business as well as its data warehouse.

You may not be able to identify validation errors until the data has been extracted from the operational systems. This situation can occur when data is extracted from multiple data sources. For example, reconciling data extracted from separate sales tracking, shipping, and billing systems may uncover discrepancies that must be addressed in one or more of the source systems.

You may also identify inconsistencies other than errors in data after it has been extracted. For example, different data sources may use different coding systems for the same kind of data. You can often use translation tables to reconcile these differences during the extraction operation or later during transformation operations. For example, a legacy system may code state or province names using a three-character code, whereas another system may use a two-character code. The data from one or both of these systems must be translated into a single set of codes before loading the data into the data warehouse. In other cases, you may discover inconsistencies if source systems permit freeform entry of text information. Such data is often internally inconsistent because different data-entry personnel may enter the same data in different ways. Inconsistent representations of the same data must be reconciled if such data is to be used for analysis. For example, in a data source that permits free-form text entry for the state or province portion of an address, the state of Florida may be entered as FL, Fla, Florida, or even Flor. It may be difficult to modify legacy source systems to implement a standard coding validation. Manual transformation adjustments may be necessary to reconcile such differences if the contributing source systems cannot be modified.

You can use the powerful transformation capabilities of Data Transformation Services (DTS) in Microsoft® SQL Server[™] 2000 during the extraction process to reconcile many formatting, data encoding, and other inconsistencies. Other transformations must be accomplished after the data has been extracted from the source systems.

Some of the tools available in SQL Server 2000 for extracting data are:

- Transact-SQL
- Distributed queries
- DTS
- Command line applications
- **bcp** utility
- BULK INSERT statement for loading from text files
- ActiveX scripts

In some data warehouse implementations, you may also find that you can use Replication to copy data from source systems to the data preparation area.

See Also

Accessing and Changing Relational Data Overview Importing and Exporting Data DTS Overview Replication Overview

Cleansing and Transforming Data

You can accomplish many data transformations during the process of extracting data from the source systems. However, there are often additional tasks to complete before you can load data into the data warehouse. For example, you must reconcile inconsistent data from heterogeneous data sources after extraction and complete other formatting and cleansing tasks. You should also wait until after the extraction process to incorporate surrogate keys. Some transformations that you might technically accomplish during the extraction process may interfere with the performance or operation of the online source system; you should defer these tasks until after extraction is complete.

After extraction from the source systems, the data should reside in a data preparation area where the cleansing and transformations can be completed before the data is loaded into the data warehouse. The data preparation area can be a separate database or separate tables in the data warehouse database. During the cleansing and transformation phase, you can execute procedures to validate and verify data consistency, transform data into common formats, and incorporate surrogate keys.

You may need to perform manual operations to reconcile data inconsistencies or to resolve ambiguous text field entries. Each time a manual operation is required, you should try to identify a way to eliminate the manual step in future data transformation operations. In some cases, you may be able to modify the source data systems to eliminate the cause at the source. In other cases, you may be able to establish an automated process that will set aside unresolved data for later manual exception processing so the bulk of the data can be loaded into the data warehouse without delay for manual intervention.

Some typical data transformations include:

- Combining multiple name fields into one field.
- Breaking down date fields into separate year, month, and day fields.
- Mapping data from one representation to another, such as TRUE to 1

and FALSE to 0 or postal codes from numeric to text.

- Mapping data from multiple representations to a single representation, such as a common format for telephone numbers, or different credit rating codes to a common "Good, Average, Poor" representation.
- Creating and applying surrogate keys for dimension table records.

Some of the tools available in Microsoft® SQL Server[™] 2000 for transforming data are:

- Transact-SQL queries
- DTS packages
- Command line applications
- ActiveX scripts

See Also

Accessing and Changing Relational Data Overview

DTS Overview

Automating Administrative Tasks

Loading Data into the Data Warehouse Database

After the data has been cleansed and transformed into a structure consistent with the data warehouse requirements, data is ready for loading into the data warehouse. You may make some final transformation during the loading operation, although you should complete any transformations that could identify inconsistencies before the final loading operation.

The initial load of the data warehouse consists of populating the tables in the data warehouse schema and then verifying that the data is ready for use. You can use various methods to load the data warehouse tables, such as:

- Transact-SQL
- DTS
- **bcp** utility

When you load data into the data warehouse, you are populating the tables that will be used by the presentation applications that make the data available to users. Loading data often involves the transfer of large amounts of data from source operational systems, a data preparation area database, or preparation area tables in the data warehouse database. Such operations can impose significant processing loads on the databases involved and should be accomplished during a period of relatively low system use.

After the data has been loaded into the data warehouse database, verify the referential integrity between dimension and fact tables to ensure that all records relate to appropriate records in other tables. You should verify that every record in a fact table relates to a record in each dimension table that will be used with that fact table. For example, if a fact table of product sales is to be used with dimension tables for customers, products, time, and stores, then for each sale record in the fact table there must be a record in each dimension table that relates to the sale record through correspondence of primary keys. This verification ensures that for every sale, the customer who made the purchase, the product sold, and the time and location of the sale are identified.

Data integrity in the reverse order is not necessary. That is, it is not necessary for every record in a dimension table to relate to a record in the fact table. For example, dimensions in a sales data warehouse typically are shared dimensions, which contain the full sets of customers, products, stores, and so on. A fact table may contain sales records for a specific time period during which some customers did not make any purchases and some products were not sold.

Most queries that retrieve data from the data warehouse use inner joins between the fact and dimension tables. Such queries will ignore facts for which at least one of the joined dimension tables does not contain a matching record, causing retrieved data to be inaccurate and possibly inconsistent among different queries. For example, if a customer record is missing for a particular sales fact record, any query that includes the customer dimension table will ignore the sales fact record, but any query that does not include the customer dimension table will contain the sales fact record. A query that computes the sum of sale amounts by customer will yield a different grand total than a query that computes the sum of sale amounts by product, because the first query ignores the sale for which there is no customer and the second query includes it.

If you use a dimension table containing data that does not apply to all facts, you must include a record in the dimension table that can be used to relate to the remaining facts. For example, in a table of sales promotion records, you can include a generic record that you can use to relate to any sales fact for which there is no applicable sales promotion. Without this generic promotion record any query that joins the promotion table to the sales fact table will not include sales for which there is no corresponding promotion.

To verify referential integrity in a star schema you can use a simple SQL query that counts the rows returned when all appropriate dimension tables are joined to the fact table using inner joins. The number of rows returned by this query should match the number of rows in the fact table. If you are using a snowflake schema, you should also verify referential integrity between dimension tables and the subordinate tables to which they are linked to verify that no records in any table are eliminated by inner joins to subordinate tables. You should perform this verification by starting with the tables at the lowest level of the snowflake dimension and joining them to the tables at the next higher level, continuing until the primary dimension table has been verified. This is an important step because there can be situations in which the dimension may verify correctly against the current fact table, even though some dimension records are missing;

these records will be needed when new facts are added.

See Also

Accessing and Changing Relational Data Overview

Preparing Presentation Information

Because access to data warehouse data is often provided through client applications, there are often tasks that must be performed in the data warehouse to prepare the information for presentation to end users. Part of the data warehouse design effort is to identify any special data configuration requirements necessary for these applications and often to configure the applications themselves as described in the application documentation. For more information, see <u>Using a Data Warehouse</u>.

Distributing Data to Data Marts

Your data warehouse design may include data marts for specific business areas such as sales, warehouse, and financial departments. Each such data mart contains a subset of the data warehouse data, but it should use common shared dimensions to prevent inconsistencies in analysis and reporting. For example, a **Products** dimension table may be used in all data marts, but a financial business planning dimension table may only be appropriate for financial data marts. Other data marts may contain region-specific data to augment region-specific operational database systems.

The initial load of a data mart from the data warehouse copies all applicable shared dimension tables and the fact data appropriate to the data mart. Special dimension tables unique to a data mart may be created locally if they are to be used only within the department or group served by the data mart; dimensions that are used to create reports for comparison to reports from other data marts should be shared dimensions that are administered centrally at the data warehouse and loaded from data warehouse tables. You can use many or all of the same tools to load data marts that you use to load the data warehouse database.

See Also

Updating Data Marts

Using a Data Warehouse

The traditional role of a data warehouse is to collect and organize historical business data so it can be analyzed to assist management in making business decisions. Until recently, access to data warehouses was limited to database experts who could create the sophisticated queries necessary to retrieve, summarize, and format information for use by analysts and executive decision makers. As data warehouses become more common and businesses involve lower levels of management in the decision-making process, the need has become greater for direct end-user access to data warehouse data by people with minimal database expertise.

The data warehouse must accommodate the requirements of a continually increasing variety of applications that access data warehouse data. Most applications must be set up and initially configured before they can work effectively with a data warehouse, and this work is often performed or managed by the data warehouse administrator. In some cases the data warehouse must incorporate modifications in order to meet the requirements of a new application.

In addition to end-user applications for data access, other applications continue to be developed that execute within the data warehouse environment to configure and analyze data in new and powerful ways. Such applications require administration and maintenance by the data warehouse administrator.

New uses for data warehouse technology are continually being developed. Some organizations now collect, analyze, and package data for sale to customers. Real-time data warehouses, once a term with no meaning, are now emerging for use in online commerce.

The topics in this section describe various technologies that can be used with data warehouses.

Торіс	Description
SQL Queries	Describes the uses of SQL queries in a data
	warehouse.
OLAP and Data Mining	Describes online analytical processing (OLAP)
	and data mining technologies used with data

	warehouses.
English Query	Describes the component of Microsoft® SQL
	Server™ 2000 that enables querying a data
	warehouse using English words and phrases.
Microsoft Office 2000	Describes how Microsoft Office 2000
	components can use data in a data warehouse.
Web Access and	Describes access to data warehouse data and
<u>Reporting</u>	reports over the Web.
Offline OLAP Cubes	Describes the use of offline cube for data analysis
	when users are not connected to SQL Server
	2000 Analysis Services.
Third-Party Applications	Describes the use of third-party applications with
	data warehouses.
Custom Applications	Describes the availability of application
	programming interfaces (APIs) that can be used
	to create custom applications that administer or
	use data warehouses.

SQL Queries

End users seldom access data warehouse data directly using Structured Query Language (SQL) queries. Analytical SQL queries can be quite complex, requiring database expertise to create correctly. The volume of data in a data warehouse is often so large that sophisticated SQL techniques are needed to achieve useful performance. A SQL query that joins three or four dimension tables to a fact table containing millions of rows and uses aggregate functions such as SUM to summarize and group the results can impose a significant load on any relational database and often yields performance that is not acceptable for online analysis.

SQL queries are often created by database experts for use with predefined reports that are executed on a regular basis during periods of low activity. Auxiliary summarization tables can be created and used to optimize the performance of these queries; such tables must be initially designed and populated when the data warehouse is loaded, and then updated every time the data warehouse is updated.

The use of SQL queries is one of the oldest methods of accessing data warehouse data. There are many books and training courses available that offer very sophisticated techniques for configuring data warehouse information and designing effective SQL queries that address complex analytical tasks.

Microsoft® SQL Server[™] 2000 provides sophisticated query processing and optimization techniques and a powerful language, Transact-SQL, to address the needs of the data warehouse implementation. For more information, see <u>Accessing and Changing Relational Data Overview</u>.

OLAP and Data Mining

Online analytical processing (OLAP) is a technology that uses multidimensional data representations, called cubes, to provide rapid access to data warehouse data. Cubes model data in the dimension and fact tables in the data warehouse and provide sophisticated query and analysis capabilities to client applications.

Data mining uses sophisticated algorithms to analyze data and create models that represent information about the data. Data mining models can be used to predict characteristics of new data or to identify groups of data entities that have similar characteristics.

Microsoft® SQL Server[™] 2000 Analysis Services provides a powerful server and administrative tools to create and manage OLAP data and serve online client applications. Analysis Services also incorporates data mining algorithms that can analyze relational data in the data warehouse database and multidimensional data in cubes.

Cubes and data mining models must be designed, configured, and processed before they can be used by client applications, and they usually require updating when the data warehouse data is updated. For more information, see <u>Analysis</u> <u>Services Overview</u>.

English Query

English Query provides a system for developing client applications that enable end users to access data using English words and phrases. English Query can be used to access data in the data warehouse database or in cubes created by Microsoft® SQL ServerTM 2000 Analysis Services.

To develop an English Query application, a model must first be created that relates database tables, fields, cubes, and data to English words and phrases. An English Query application can then be generated and incorporated into custom Web or client applications and made available to end users. For more information, see English Query Overview.

Microsoft Office 2000

Data warehouse data in a Microsoft® SQL Server[™] 2000 database can be accessed by Microsoft Office components such as Microsoft Excel or Microsoft Access. However, the volume of data in most data warehouses often dictates that special queries or data tables be created and maintained to support the use of these components by end users. Such special queries and tables must be created and maintained as part of the data warehouse.

One exception is the integration of Excel PivotTables® with SQL Server 2000 Analysis Services. When Analysis Services is used to create and manage OLAP data, end users can easily connect to cubes through an Analysis server and analyze data online or create cubes on their local computer for offline use. For more information, see the Microsoft Office 2000 documentation.

Web Access and Reporting

Web applications that provide end-user access to data warehouse data are popular because the client can use a standard Web browser instead of an application that must be installed, configured, and maintained. Initially limited to simple viewing of data presented on static Web pages, current technology now enables the creation of sophisticated interactive applications that allow users to query and update data in data warehouses and cubes.

Microsoft® SQL Server[™] 2000 and its components, such as Analysis Services and English Query, offer a number of ways to query and update data over the Web when used with Microsoft Internet Information Services (IIS). SQL Server 2000 introduces support for XML functionality for storing, retrieving, and updating information using XML, XML-Data Reduced (XDR) schemas, and XPath queries over HTTP connections. The PivotTable® Service component of Analysis Services can be used with IIS to provide Web access to cubes using Multidimensional Expressions (MDX) syntax for querying. English Query applications can be embedded into Active Server Pages (ASP) or COM-based applications to support Web queries in English.

Web data access applications are developed using APIs provided by SQL Server 2000 and its components. Web applications can be as simple as displaying a predefined report or executing predefined queries against the data warehouse database or OLAP cubes, or they can be as complex as any directly connected client-server application. The impact of a Web application on data warehouse design or maintenance is determined by the application.

See Also

XML and Internet Support Overview

PivotTable Service

Developing and Deploying English Query Applications

Offline OLAP Cubes

Cubes used in online analytical processing (OLAP) provide a multidimensional view of data warehouse data that end users find easy to use and explore as they search for answers to business questions. Microsoft® SQL ServerTM 2000 Analysis Services provides the capability through its PivotTable® Service component for client applications to create subsets of data warehouse cubes and save them locally for offline analysis. End-user applications can also use PivotTable Service in an offline mode to create offline cubes directly from relational databases.

Third-party applications and custom applications can use PivotTable Service to create and manage offline cubes. One end-user application that provides offline cube support is Microsoft Excel 2000. For more information, see the Excel documentation.

Offline cubes are created and managed by end-user applications and generally have little impact on data warehouse or cube design. Maintenance of offline cube data is the responsibility of the end user, who can refresh data from online cubes or update offline cubes created from local databases as necessary. Offline cubes do not interfere with normal data warehouse and cube management and maintenance.

See Also

PivotTable Service

Third-Party Applications

Many applications have been commercially developed for use with data warehouses and OLAP cubes. Each application has unique requirements that may or may not require design changes to a data warehouse for effective operation of the application. Some applications operate on the data warehouse to provide additional analysis, management, or maintenance capabilities. Others are client applications that provide analysis capabilities for end users. Commercial applications usually require setup and configuration before they can use data warehouse data effectively. Applications may also need configuration adjustments in order to accommodate changes in the data warehouse and updates to data.

Custom Applications

Microsoft® SQL Server[™] 2000 and its components provide a rich set of application programming interfaces (APIs) that can be used to develop custom applications to enhance and automate data warehouse administration, or to create client applications tailored to your business needs. For more information, see <u>Building SQL Server Applications Overview</u>.

Maintaining a Data Warehouse

Data warehouses collect and organize historical business data so it can be analyzed to assist management in making business decisions. To achieve this purpose, the data warehouse is created and initially loaded with the existing historical business data. It is then periodically updated with new data from operational data systems. Much of the effort in data warehouse maintenance is involved with updating the data in the data warehouse, adjusting data presentation applications to incorporate new data, and updating data marts.

Topics in this section describe common tasks performed to maintain data warehouses.

Торіс	Description
Updating Data Warehouse Data	Describes the process and tasks
	involved in updating a data warehouse.
Administering a Data Warehouse	Describes common data warehouse
	administration tasks.
Tuning Data Warehouse	Describes ways to improve data
Performance	warehouse performance.

Updating Data Warehouse Data

Updating data warehouse data includes periodically extracting data from operational systems, cleansing and transforming the data, and loading the new data into the data warehouse. Each data update also includes tasks that must be accomplished to synchronize cubes if Microsoft® SQL Server[™] 2000 Analysis Services is used for online analytical processing (OLAP), and to update any data marts that are part of the data warehouse.

The process of extracting, cleansing, and transforming data for a periodic update is essentially the same as the process used in the initial loading of the data warehouse, although the update process is often much less complex and more automated than the initial load process. Procedures and automated tasks developed during the initial load process can reduce the amount of manual effort required during updates. Corrections to source operational systems identified and implemented during the initial load also reduce the number of inconsistencies and errors that must be addressed during updates. However, it is often the case that manual intervention is required during updates to ensure the data is ready for loading into the data warehouse.

One difference between the initial data load and data updates is that verifying the referential integrity should be performed incrementally on update data before it is loaded into the data warehouse and made available to users. Updates often include additions and changes to dimension tables as well as the addition of rows to the fact tables. The new and changed data should be checked for internal consistency as well as verified against existing data in the data warehouse before it is loaded into the data warehouse.

After the update data has been made ready for loading into the data warehouse, you can use Transact-SQL, Data Transformation Services (DTS), or the **bcp** utility to update the data warehouse tables. Depending on the design and implementation of the presentation applications that provide access to data warehouse data for end users, you may need to take the data warehouse offline during the update to prevent inconsistencies in query results.

See Also

Extracting Data from Operational Systems Cleansing and Transforming Data Loading Data into the Data Warehouse Database Scheduling Data Updates Synchronizing OLAP Cubes Updating Data Marts

Scheduling Data Updates

Data warehouses used for the analysis of historical business data are better served by periodic updates than they are by online data updates. Constantly changing data can interfere with the analysis process, which requires time and iterative querying to refine and verify results. The frequency of data warehouse updates depends on the needs of the organization and the uses of the data warehouse. Typical update periods may be monthly, weekly, or daily. There are several items to consider when scheduling data warehouse updates.

Although the amount of data involved in a data warehouse update is usually much less than the amount of data initially loaded, the process of extracting, cleansing and transforming, and loading the data can still adversely affect the operational systems involved. When you load data into the data warehouse, you are populating the tables that will be used by the presentation applications that make the data available to users. Loading data often involves the transfer of large amounts of data from source operational systems, a data preparation area database, or preparation area tables in the data warehouse database. Such operations can impose significant processing loads on the databases involved and should be accomplished during periods of relatively low system use.

You must also coordinate data warehouse update operations with the operation of the presentation services that make the data available to users. If you are loading large amounts of data, you may want to perform bulk loads without using transactions or logging. In this case, you may need to take the data warehouse offline during the loading operation to prevent users from accessing data that could be inconsistent while it is being loaded to various tables.

Presentation applications that provide access to the data warehouse by end users may require adjustment to accommodate new data. For example, Microsoft® SQL Server[™] 2000 Analysis Services cubes often require updating or reprocessing to incorporate new data in the data warehouse. For more information, see <u>Synchronizing OLAP Cubes</u>.

Synchronizing OLAP Cubes

If you are using Microsoft® SQL Server[™] 2000 Analysis Services with your data warehouse to analyze and prepare OLAP data for presentation to users, you must be familiar with the effects of data changes on Analysis Services cubes before updating data in the data warehouse. In some situations, users can receive inconsistent results after data warehouse data changes until summary information is updated to incorporate new detail data.

The effects of changes in underlying data on cubes vary with the types of cubes in use. For example, multidimensional OLAP (MOLAP) cubes are unaffected by changes in underlying data until the cubes are reprocessed. Depending on how they are defined, relational OLAP (ROLAP) cubes and hybrid OLAP (HOLAP) cubes may be immediately affected and require updating or reprocessing to regain consistency. Real-time cubes are designed to incorporate new data warehouse data immediately. For more information, see <u>Analysis Services</u> <u>Overview</u> and <u>Maintaining OLAP Data</u>.

Updating Data Marts

In some data warehouse implementations, data marts receive their data from the master data warehouse. In other implementations the data marts are updated locally and contribute data to the data warehouse database. Moving updated data to or from data marts can be straightforward if the data marts are designed to be consistent with the master data warehouse. The use of standardized schemas, shared dimensions, and common fact table formats greatly contribute to the ease of data mart maintenance.

If the data marts receive data updates from a master data warehouse, you should be able to design automated tasks that filter the data warehouse update data and post the appropriate data subset to each data mart during the update of the data warehouse database. Presentation applications used by the data mart may need to be adjusted to accommodate the new data.

If a data mart collects data locally and contributes data to the master data warehouse, you may need to treat the data mart as if it were an operational data source and bring the data into a preparation area for cleansing and verification against the data warehouse before posting it to the data warehouse. If the data mart is designed and managed as an integral part of the data warehouse, the verification process may not be necessary and the data can be loaded directly from the data mart into the data warehouse database. In either case, you may need to adjust presentation applications to accommodate the new data.

You can use Transact-SQL, Data Transformation Services (DTS), the **bcp** utility, or custom applications to update the data mart or data warehouse tables. You may also be able to use replication to perform data mart updates. For more information, see <u>Replication Overview</u>.

Administering a Data Warehouse

Administering a data warehouse is both similar to and different from administering an online transaction processing (OLTP) system. It is similar in that data warehouse data is stored and maintained in a relational database, so the tools used to administer relational databases can be used with data warehouses. It is different in that OLTP systems are generally characterized by high-volume transaction updates to volatile data, whereas data warehouses are generally characterized by massive amounts of stable historical data. These differences call for different approaches to data warehouse administrative tasks such as backing up data and automating recurring tasks.

See Also

Administering SQL Server Overview Administering Analysis Services Backing Up Data Warehouse Data Automating Data Warehouse Tasks

Backing Up Data Warehouse Data

An online transaction processing (OLTP) system captures incoming data and updates a database. To ensure against loss of data, the system logs transactions as they are performed, and administrators develop backup strategies that include periodic full and incremental backups of database. These strategies are designed to prevent loss of data, to minimize interference with operational transaction processing, and to provide for rapid recovery from system malfunctions.

In contrast, a data warehouse stores massive amounts of stable historical data that is updated on a managed periodic schedule. For data warehouses, you should design backup strategies to minimize full backups and to use incremental backups for data updates.

Recovery time constraints are often more flexible and less restrictive for data warehouse malfunctions than for OLTP malfunctions. More permissive recovery time constraints usually permit full data warehouse backups to be made much less frequently than are required for OLTP systems. For example, a table of sales facts may contain hundreds of millions of rows that reflect sales for ten years of history. It is quite unlikely that changes will be made to sales data after the business has performed a year-end closing process.

Repeated backups of data that has not changed are unnecessary, and backup strategies should take this into account. Depending on recovery time constraints and data volume, a strategy may be created that backs up data added during data warehouse updates using incremental backups, and then creates a backup of only the data added during the current year after year-end closing. To recover from a complete failure of the data warehouse database would require loading multiple backups, one for each year prior to the current year, then incremental backups for the current year updates.

Microsoft® SQL Server[™] 2000 Analysis Services maintains OLAP data in special-purpose Analysis server databases, which can be archived and restored separately from data warehouse database backups.

See Also

Backing Up and Restoring Databases

Archiving and Restoring Databases

Automating Data Warehouse Tasks

There are numerous tasks involved in administering and maintaining a data warehouse. Many of these tasks can be automated using a variety of tools available in Microsoft® SQL Server[™] 2000. Appropriate tasks can be scheduled for periodic accomplishment.

A data warehouse must be updated on a periodic basis to incorporate new data from ongoing business operations. You can combine various SQL Server 2000 processing features and methods to define jobs that perform the data extractions. You can schedule the periodic execution of such jobs and provide automatic notification of job completion and status.

In addition to the built-in tools, SQL Server 2000 provides a number of programming object models and interfaces that can be used to create custom applications that perform administrative tasks. You can use programming languages such as C/C++, Microsoft Visual Basic®, or scripting to create these applications. For more information, see <u>Building SQL Server Applications</u> <u>Overview</u> and other topics in the Building SQL Server Applications section of Books Online.

See Also

Automating Administrative Tasks

Overview of the SQL Server Tools

Updating Data Warehouse Data

Tuning Data Warehouse Performance

A data warehouse must provide rapid evaluation of queries that analyze and summarize huge numbers of rows of data from multiple joined tables. Microsoft® SQL Server[™] 2000 provides information you can use to optimize the performance of the relational database that contains the data warehouse data. Database performance can be affected by many choices you make in the logical design of the database, its physical implementation, index tuning, query tuning, and so on. For more information, see <u>Optimizing Database Performance</u> <u>Overview</u>.

Although the performance of SQL Server 2000 Analysis Services depends to a large extent on the performance of the data warehouse database, its performance is also influenced by the design of the data warehouse database and the Analysis Services cubes. You can also tune the performance of Analysis Services by using tools that analyze usage patterns by adjusting the amount of aggregations that are precalculated when cubes are processed, optimizing cube schemas to avoid unnecessary joins, and so on. Computer hardware configurations also affect the performance of Analysis servers. For more information, see <u>Analyzing and Optimizing Performance</u>.