COVER FEATURE

Database Technology for Decision Support Systems

Creating the framework for an effective decision support system—one that leverages business data from numerous discrete touch points—is a daunting but doable task.

Surajit Chaudhuri
Microsoft Research

Umeshwar Dayal
Hewlett-Packard Laboratories

Venkatesh Ganti
Microsoft Research

Decision support systems are the core of business IT infrastructures because they give companies a way to translate a wealth of business information into tangible and lucrative results. Collecting, maintaining, and analyzing large amounts of data, however, are mammoth tasks that involve significant technical challenges, expenses, and organizational commitment.

Online transaction processing systems allow organizations to collect large volumes of daily business point-of-sales data. OLTP applications typically automate structured, repetitive data processing tasks such as order entry and banking transactions. This detailed, up-to-date data from various independent touch points must be consolidated into a single location before analysts can extract meaningful summaries. Managers use this aggregated data to make numerous day-to-day business decisions—everything from managing inventory to coordinating mail-order campaigns.

DECISION SUPPORT SYSTEM COMPONENTS

A successful decision support system is a complex creation with numerous components. A fictitious business example, the Footwear Sellers Company, helps illustrate a decision support system's various components. FSC manufactures footwear and sells through two channels—directly to customers and through resellers. FSC's marketing executives need to extract the following information from the company's aggregate business data:

- the five states reporting the highest increases in youth product category sales within the past year,
- total footwear sales in New York City within the past month by product family,
- the 50 cities with the highest number of unique customers, and
- one million customers who are most likely to buy the new Walk-on-Air shoe model.

Before building a system that provides this decision support information, FSC's analysts must address and resolve three fundamental issues:

- what data to gather and how to conceptually model the data and manage its storage,
- how to analyze the data, and
- how to efficiently load data from several independent sources.

As Figure 1 shows, these issues correlate to a decision support system's three principal components: a data warehouse server, online analytical processing and data mining tools, and back-end tools for populating the data warehouse.

Data warehouses contain data consolidated from several operational databases and tend to be orders of magnitude larger than operational databases, often hundreds of gigabytes to terabytes in size. Typically, the data warehouse is maintained separately from the organization's operational databases because analytical applications' functional and performance requirements are quite different from those of operational databases. Data warehouses exist principally for decision support applications and provide the historical, summarized, and consolidated data more appropriate

0018-9162/01/$17.00 © 2001 IEEE

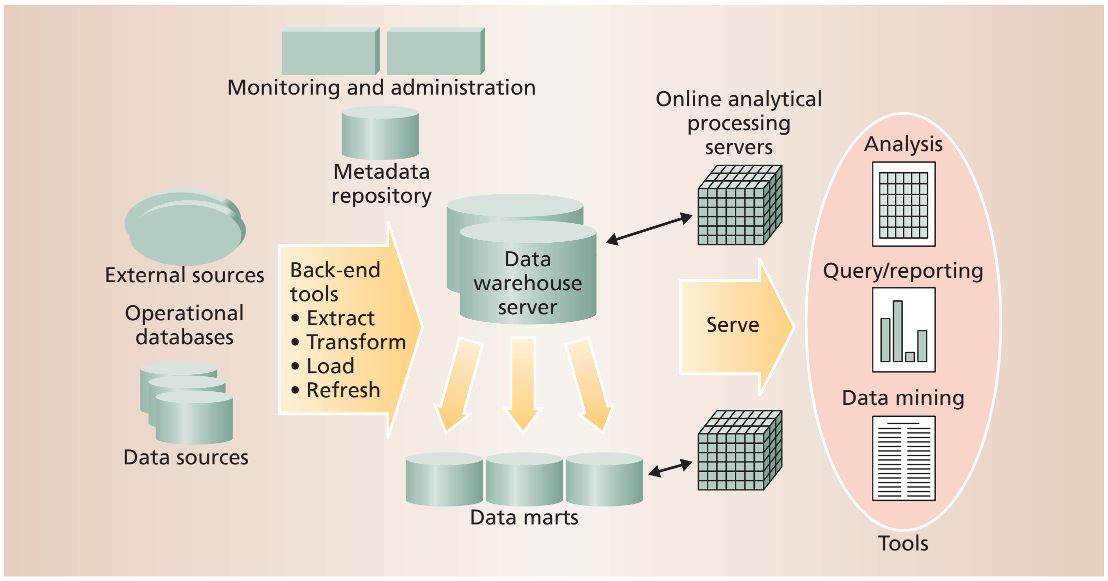

Figure 1. Decision support system architecture, which consists of three principal components: a data warehouse server, analysis and data mining tools, and data warehouse back-end tools.

for analysis than detailed, individual records. The workloads consist of ad hoc, complex queries that access millions of records and perform multiple scans, joins, and aggregates. Query response times are more important than transaction throughput.

Because data warehouse construction is a complex process that can take many years, some organizations instead build *data marts*, which contain information for specific departmental subsets. For example, a marketing data mart may include only customer, product, and sales information and may not include delivery schedules. Several departmental data marts can coexist with the main data warehouse and provide a partial view of the warehouse contents. Data marts roll out faster than data warehouses but can involve complex integration problems later if the initial planning does not reflect a complete business model.

Online analytical processing and data mining tools enable sophisticated data analysis. Back-end tools—such as extraction, transformation, and load tools—populate the data warehouse from external data sources.

DATA WAREHOUSE

Most data warehouses use relational database technology because it offers a robust, reliable, and efficient approach for storing and managing large volumes of data. The most significant issue associated with data warehouse construction is database design, both logical and physical. Building a logical schema for an enterprise data warehouse requires extensive business modeling.

Logical database design

In the *star schema* design, the database consists of a *fact table* that describes all transactions and a *dimension table* for each entity. For the fictitious FSC, each sales transaction involves several entities—a customer, a salesperson, a product, an order, a transaction date, and the city where the transaction occurred. Each transaction also has *measure* attributes—the number of units sold and the total amount the customer paid.

Each tuple in the fact table consists of a pointer to each entity in a transaction and the numeric measures associated with the transaction. Each dimension table consists of columns that correspond to the entity's attributes. Computing the join between a fact table and a set of dimension tables is more efficient than computing a join among arbitrary relations.

Some entities, however, are associated with *hierarchies*, which star schemas do not explicitly support. A hierarchy is a multilevel grouping in which each level consists of a disjoint grouping of the values in the level immediately below it. For example, all products can be grouped into a disjoint set of *categories*, which are themselves grouped into a disjoint set of *families*.

Snowflake schemas are a refinement of star schemas in which a dimensional hierarchy is explicitly represented by normalizing the dimension tables. In the star schema depicted in Figure 2, a set of attributes describes each dimension and may be related via a relationship hierarchy. For example, the FSC's product dimension consists of five attributes: the product name (Running Shoe 2000), category (athletics), product family (shoe), price ($80), and the profit margin (80 percent).

Physical database design

Database systems use redundant structures such as indices and materialized views to efficiently process complex queries. Determining the most appropriate set of indices and views is a complex physical design problem. While index lookups and scans can be effective for data-selective queries, data-intensive queries can require sequential scans of an entire relation or a relation's vertical partitions. Improving the efficiency of table scans and exploiting parallelism to reduce query response times are important design considerations.[1]

Index structures and their usage

Query processing techniques that exploit indices through index intersection and union are useful for answering multiple-predicate queries. Index intersec-

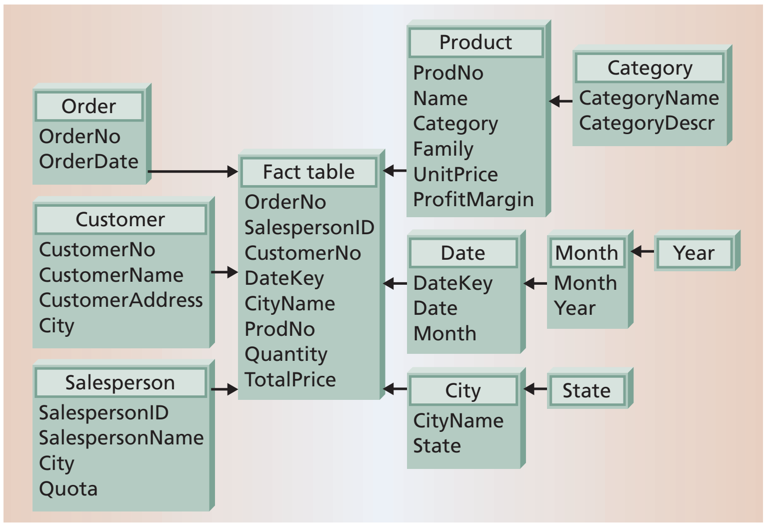

Figure 2. Snowflake schema for a the hypothetical Footwear Sellers Company. A set of attributes describes each dimension and is related via a relationship hierarchy.

tions exploit multiple-condition selectivities and can significantly reduce or eliminate the need to access base tables if all projection columns are available via index scans.

The specialized nature of star schemas makes master-detail join indices especially attractive for decision support. While indices traditionally map the value in a column to a list of rows with that value, a join index maintains the relationship between a foreign key and its matching primary keys. In the context of a star schema, a join index can relate the values of one or more attributes of a dimension table to matching rows in the fact table. The schema in Figure 2, for example, can support a join index on City that maintains, for each city, a list of the tuples' record identifiers in the fact table that correspond to sales in that city. Essentially, the join index precomputes a binary join.

Multikey join indices can represent precomputed *n*-way joins. For example, a multidimensional join index built over the sales database can join City.CityName and Product.Name to the fact table. Consequently, the index entry for *Seattle, Running Shoe* points to the record identifiers of tuples in the Sales table with that combination.

Materialized views and usage

Many data warehouse queries require summary data and therefore use aggregates. Materializing summary data can accelerate many common queries. In the FSC example, two data views—total sales grouped by product family and city, and total number of customers grouped by city—can efficiently answer three of the marketing department's queries: states reporting the highest increases in youth product category sales, total footwear sales in New York City by product family, and the 50 cities with the highest number of unique customers.

The challenges in exploiting materialized views are similar to the index challenges:

- identify the views to materialize,
- exploit the materialized views to answer queries, and
- update the materialized views during load and refresh.

Because materialized views require extremely large amounts of space, currently adopted solutions only support a restricted class of structurally simple materialized views.

ONLINE ANALYTICAL APPLICATIONS

In a typical *online analytical application*, a query aggregates a numeric measure at higher levels in the dimensional hierarchy. An example is the first FSC marketing query that asks for a set of aggregate hierarchical measures—the five states reporting the highest increases in youth product category sales within the past year. State and year are ancestors of the city and date entities.

In the context of the FSC data warehouse, a typical OLAP session to determine regional sales of athletic shoes in the last quarter might proceed as follows:

- The analyst issues a *select sum(sales) group by country* query to view the distribution of sales of athletic shoes in the last quarter across all countries.
- After isolating the country with either the highest or the lowest total sales relative to market size, the analyst issues another query to compute the total sales within each state of that country to understand the reason for its exceptional behavior.

The analyst traverses down the hierarchy associated with the city dimension. Such a downward traversal of the hierarchy from the most summarized to the most detailed level is called *drill-down*. In a *roll-up* operation, the analyst goes up one level—perhaps from the state level to the country level—in a dimensional hierarchy.

Key OLAP-related issues include the conceptual data model and server architectures.

OLAP conceptual data model

The *multidimensional model* shown in Figure 3 uses numeric measures as its analysis objects. Each numeric measure in this aggregation-centric conceptual data model depends on dimensions that describe the entities in the transaction. For instance, the dimensions associated with a sale in the FSC example are the customer, the salesperson, the city, the product name, and the date the sale was made. Together, the dimensions uniquely determine the measure, so the multidimensional data model treats a measure as a value in the dimensions' *multidimensional space*.

With a multidimensional data view, drill-down and roll-up queries are logical operations on the cube depicted in Figure 3. Another popular operation is to compare two measures that are aggregated by the same dimensions, such as sales and budget.

OLAP analysis can involve more complex statistical calculations than simple aggregations such as sum, count, and average. Examples include functions such as moving averages and the percentage change of an aggregate within a certain period compared with a different time period. Many commercial OLAP tools provide such additional functions.

The time dimension is particularly significant for decision support processes such as trend analysis. For example, FSC's market analysts may want to chart sales activity for a class of athletic shoes before or after major national athletic contests. Sophisticated trend analysis is possible if the database has built-in knowledge of calendars and other sequential characteristics of the time dimension. The OLAP Council (http://www.olapcouncil.org) has defined a list of other such multidimensional cube operations.

OLAP server and middleware architectures

Although traditional relational servers do not efficiently process complex OLAP queries or support multidimensional data views, three types of relational DBMS servers—relational, multidimensional, and hybrid online analytical processing—now support OLAP on data warehouses built with relational database systems.

ROLAP servers. ROLAP middleware servers sit between the relational back-end server where the data warehouse is stored and the client front-end tools. ROLAPs support multidimensional OLAP queries and typically optimize for specific back-end relational servers. They identify the views to be materialized, rephrase user queries in terms of the appropriate materialized views, and generate multistatement SQL for the back-end server. They also provide additional services such as query scheduling and resource assignment. ROLAP servers exploit the scalability and transactional features of relational systems, but intrinsic mismatches between OLAP-style querying and SQL can create performance bottlenecks in OLAP servers.

Bottlenecks are becoming less of a problem with the OLAP-specific SQL extensions implemented in relational servers such as Oracle, IBM DB2, and Microsoft SQL Server. Functions such as median, mode, rank, and percentile extend the aggregate functions. Other feature additions include aggregate computations over moving windows, running totals, and breakpoints to enhance support for reporting applications.

Multidimensional spreadsheets require grouping by different sets of attributes. Jim Gray and colleagues[2] proposed two operators—*roll-up* and *cube*—to augment SQL and address this requirement. Roll-up of a list of attributes such as product, year, and city over a data set results in answer sets with the following applications:

- group by product, year, and city;
- group by product and year; and
- group by product.

Given a list of k columns, the cube operator provides a group-by for each of the 2^k combinations of columns. Such multiple group-by operations can be executed efficiently by recognizing commonalities among them.[3] When applicable, precomputing can enhance OLAP server performance.[4]

MOLAP servers. MOLAP is a native server architecture that does not exploit the functionality of a relational back end but directly supports a multidimensional data view through a multidimensional storage engine. MOLAP enables the implementation of multidimensional queries on the storage layer via direct mapping. MOLAP's principal advantage is its excellent indexing properties; its disadvantage is poor storage utilization, especially when the data is sparse. Many MOLAP servers adapt to sparse data sets through a two-level storage representation and extensive compression. In a two-level storage representation, either directly or by means of design tools, the user identifies a set of one- or two-dimensional subarrays that are likely to be dense and represents them in the array format. Traditional indexing structures can then index these smaller arrays. Many of the techniques devised for statistical databases are relevant for MOLAP servers. Although MOLAP servers offer good performance and functionality, they do not scale well for extremely large data sizes.

HOLAP servers. The HOLAP architecture combines ROLAP and MOLAP technologies. In contrast to MOLAP, which performs better when the data is reasonably dense, ROLAP servers perform better when the data is extremely sparse. HOLAP servers identify sparse and dense regions of the multidimensional space and take the ROLAP approach for sparse regions and the MOLAP approach for dense regions. HOLAP servers split a query into multiple queries, issue the queries against the relevant data portions, combine the results, and then present the result to the user. HOLAP's selective view materialization, selective index building, and query and resource scheduling are similar to its MOLAP and ROLAP counterparts.

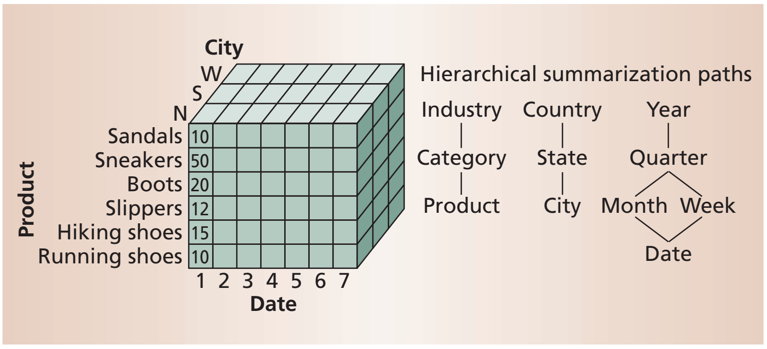

Figure 3. A sample multidimensional database. Each numeric measure in this aggregation-centric conceptual data model depends on dimensions that describe the entities in the transaction.

Data mining iteratively builds models on a prepared data set and then deploys one or more models.

DATA MINING

Suppose that FSC wants to launch a catalog mailing campaign with an expense budget of less than $1 million. Given this constraint, the marketing analysts want to identify the set of customers most likely to respond to and buy from the catalog. Data mining tools provide such advanced predictive and analytical functionality by identifying distribution patterns and characteristic behaviors within a data set.

Knowledge discovery—the process of specifying and achieving a goal through iterative data mining—typically consists of three phases:

- data preparation,
- model building and evaluation, and
- model deployment.

Data preparation

In the data preparation phase, the analyst prepares a data set containing enough information to build accurate models in subsequent phases. To address FSC's information requirement, an accurate model will predict whether a customer is likely to buy products advertised in a new catalog. Because predictions are based on factors potentially influencing customers' purchases, a model data set would include all customers who responded to mailed catalogs in the past three years, their demographic information, the 10 most expensive products each customer purchased, and information about the catalogs that elicited their purchases.

Data preparation can involve complex queries with large results. For instance, preparing the FSC data set involves joins between the customer relation and the sales relation as well as identifying the top 10 products for each customer. All the issues related to efficiently processing decision support queries are equally relevant in the data mining context. In fact, data mining platforms use OLAP or relational servers to meet their data preparation requirements.

Data mining typically involves iteratively building models on a prepared data set and then deploying one or more models. Because building models on large data sets can be expensive, analysts often work initially with data set samples. Data mining platforms, therefore, must support computing random samples of data over complex queries.

Building and evaluating mining models

Only after deciding which model to deploy does the analyst build the model on the entire prepared data set. The goal of the model-building phase is to identify patterns that determine a *target attribute*. A target attribute example in the FSC data set is whether a customer purchased at least one product from a past catalog.

Several classes of data mining models help predict both explicitly specified and hidden attributes. Two important issues influencing model selection are the accuracy of the model and the efficiency of the algorithm for constructing the model over large data sets. Statistically, the accuracy of most models improves with the amount of data used, so the algorithms for inducing mining models must be efficient and scalable to process large data sets within a reasonable amount of time.

Model types

Classification models are predictive. When given a new tuple, classification models predict whether the tuple belongs to one of a set of target classes. In the FSC catalog example, a classification model would determine, based on past behavior, whether a customer is or is not likely to purchase from a catalog. Decision trees and naïve Bayes models are two popular types of classification models.[5-7]

Regression trees and logistic regression are two popular types of *regression models*, which predict numeric attributes, such as a customer's salary or age.[5]

With some applications, the analyst does not explicitly know the set of target classes and considers them hidden. The analyst uses *clustering models* such as K-Means and Birch to determine the appropriate set of classes and to classify a new tuple into one of these hidden classes.[6,7]

Analysts use *rule-based models* such as the association rules model to explore whether the purchase of a certain set of footwear products is indicative, with some degree of confidence, of a purchase of another product.

Additional model considerations

No single model class or algorithm will always build the ideal model for all applications. Data mining platforms must therefore support several types of model construction for evaluation and provide additional functionality for extensibility and interchangeability.

In some cases, analysts may want to build a unique correlation model that the data mining platform does not support. To handle such requirements, mining platforms must support extensibility.

Many commercial products build models for specific domains, but the actual database on which the model must be deployed may be in a different database system. Data mining platforms and database servers therefore must also be capable of interchanging models.

The Data Mining Group (http://www.dmg.org) recently proposed using the Predictive Model Markup Language, an XML-based standard, to interchange several popular predictive model classes. The idea is that any database supporting the standard can import and deploy any model described in the standard.

Mining model deployment

During the model deployment phase, the analyst applies the selected model to data sets to predict a target attribute with an unknown value. For each current set of customers in the FSC example, the prediction is whether they will purchase a product from the new catalog. Deploying a model on an input data set—a subset or partitioning of the input data set—may result in another data set. In the FSC example, the model deployment phase identifies the subset of customers that will be mailed catalogs.

When input data sets are extremely large, the model deployment strategy must be very efficient. Using indexes on the input relation to filter out tuples that will not be in the deployment result may be necessary, but this requires tighter integration between database systems and model deployment. Unfortunately, the research community has devoted less attention to deployment efficiency than to model-building scalability.

ADDITIONAL OLAP AND DATA MINING ISSUES

Other important issues in the context of OLAP and data mining technology include packaged applications, platform application program interfaces and the impact of XML, approximate query processing, OLAP and data mining integration, and Web mining.

Packaged applications

To develop a complete OLAP or data mining analysis solution, the analyst must perform a series of complex queries and build, tune, and deploy complex models. Several commercial tools try to bridge the gap between actual solution requirements for well-understood domains and the support from a given OLAP or data mining platform. Packaged applications and reporting tools can exploit vertical-domain knowledge to make the analyst's task easier by providing higher-level, domain-specific abstractions. The Data Warehousing Information Center (http://www.dwinfocenter.org/) and KDnuggets (http://www.kdnuggets.com/solutions/index.html) provide comprehensive lists of domain-specific solutions.

Businesses can purchase such solutions instead of developing their own analysis, but specific vertical-domain solutions are limited by their feature sets and therefore may not satisfy all of a company's analysis needs as its business grows.

Platform APIs and XML's impact

Several OLAP and data mining platforms provide APIs so analysts can build custom solutions. However, solution providers typically have had to program for a variety of OLAP or data mining engines to provide an engine-independent solution. New Web-based XML services offer solution providers a common interface for OLAP engines. Microsoft and Hyperion have published the XML for Analysis specification (http://www.essbase.com/downloads/XML_Analysis_spec.pdf), a simple object access protocol-based XML API designed specifically for standardizing the data access interaction between a client application and an analytical data provider (OLAP and data mining) working over the Web. With this specification, solution providers can program using a single XML API instead of multiple vendor-specific APIs.

Specific vertical-domain solutions are limited by their feature sets, and they may not satisfy all of a company's analysis needs as its business grows.

Approximate query processing

Complex aggregate query processing typically requires accessing large amounts of data in the warehouse. For example, computing FSC's average sales across cities requires a scan of all warehouse data. In many cases, however, approximate query processing is an option for obtaining a reasonably accurate estimate very quickly. The basic idea is to summarize the underlying data as concisely as possible and then answer aggregate queries using the summary instead of the actual data. The Approximate Query Processing project (http://www.research.microsoft.com/dmx/ApproximateQP) and the AQUA Project (http://www.bell-labs.com/project/aqua) provide additional information about this approach.

OLAP and data mining integration

OLAP tools help analysts identify relevant portions of the data, while mining models effectively enhance this functionality. For example, if the FSC product sales increase does not meet the targeted rates, the marketing managers will want to know the *anomalous regions* and product categories that failed to meet the target. An exploratory analysis that identifies anomalies uses a technique that marks an aggregate measure at a higher level in a dimensional hierarchy with an *anomaly score*. The anomaly score computes the overall deviation of the actual aggregate values from corresponding expected values over all its descendants.[8,9] Analysts can use data mining tools such as regression models to compute expected values.

Web mining

Most major businesses maintain a Web presence where customers can browse, inquire about, and purchase products. Because each customer has one-to-one contact with the business through the Web site, companies can personalize the experience. For example, the Web site may recommend products, services, or articles within the customer's interest category. Amazon.com has pioneered the deployment of such personalization systems.

The two key issues involved in developing and deploying such Web systems are data collection and personalization techniques. Analysis of Web log data—

Data mining models can exploit behavioral data to personalize the Web pages the customer sees with appropriate advertisements.

automatically collected records of customer behavior at the Web site—can reveal typical patterns. For example, this kind of analysis would allow FSC to offer a special on athletic socks to a customer who purchases shoes. Data mining models can exploit such behavioral data—particularly when it is combined with the demographic data the customer enters during registration or checkout—to personalize the Web pages the customer sees with appropriate advertisements. Over time, when a large user community develops, the business can recommend additional products based on similarities among users' behavioral patterns. Data mining models can identify such similar user classes.

DATA WAREHOUSE TOOLS

Building a data warehouse from independent data sources is a multistep process that involves extracting the data from each source, transforming it to conform to the warehouse schema, cleaning it, and then loading it into the warehouse. The Data Warehousing Information Center provides an extensive list of ETL (extract, transform, load) tools for use in this operations sequence.

Extraction and transformation

The goal of the data extraction step is to bring data from disparate sources into a database where it can be modified and incorporated into the data warehouse. The goal of the subsequent data transformation step is to address discrepancies in schema and attribute value conventions. A set of rules and scripts typically handles the transformation of data from an input schema to the destination schema.

For example, an FSC distributor may report sales transactions as a flat file in which each record describes all entities and the number of units involved in the transaction. The distributor might split each customer name into three fields: first name, middle initial, and last name. To incorporate the distributor's sales information into the FSC data warehouse with the schema shown in Figure 2, the analyst must first extract the records and then, for each record, transform all three name-related source columns to derive a value for the customer name attribute.

Data cleaning

Data entry errors and differences in schema conventions can cause the customer dimension table to have several corresponding tuples for a single customer, resulting in inaccurate query responses and inappropriate mining models. For example, if the customer table has multiple tuples for several FSC customers in New York, New York may incorrectly appear in the list of top 50 cities with the highest number of unique customers. Tools that help detect and correct data anomalies can have a high payoff, and a significant amount of research addresses the problems of duplicate elimination[10] and data-cleaning frameworks.[11]

Loading

After its extraction and transformation, data may still require additional preprocessing before it is loaded into the data warehouse. Typically, batch load utilities handle functions such as checking integrity constraints; sorting; summarizing, aggregating, and performing other computations to build derived tables stored in the warehouse; and building indices and other access paths. In addition to populating the warehouse, a load utility must allow the system administrator to monitor status; to cancel, suspend, and resume a load; and to restart after failure with no data integrity loss. Because load utilities for data warehouses handle much larger data volumes than operational databases, they exploit pipelined and partitioned parallelism.[1]

Refreshing

Refreshing a data warehouse consists of propagating updates on source data that correspondingly update the base tables and the derived data—materialized views and indexes—stored in the warehouse. The two issues to consider are when to refresh and how to refresh.

Typically, data warehouses are refreshed periodically according to a predetermined schedule, such as daily or weekly. Only if some OLAP queries require current data such as up-to-the-minute stock quotes is it necessary to propagate every update. The data warehouse administrator sets the refresh policy depending on user needs and traffic. Refresh schedules may differ for different sources. The administrator must choose the refresh cycles properly so that the data volume does not overwhelm the incremental load utility.

Most commercial utilities use incremental loading during refresh to reduce data volume, inserting only the updated tuples if the data sources support extracting relevant portions of data. However, the incremental load process can be difficult to manage because the update must be coordinated with ongoing transactions.

METADATA MANAGEMENT

Metadata is any information required to manage the data warehouse, and metadata management is an essential warehousing architecture element. Administrative metadata includes all information required to set up and use a warehouse. Business metadata includes business terms and definitions, data ownership, and charging policies. Operational metadata includes information collected during warehouse operation such as the lineage of migrated and transformed data; the data currency (active, archived, or purged); and monitoring information such as usage statistics, error reports, and audit trails. Warehouse metadata often resides in a repository that enables metadata sharing among tools and processes for designing, setting up, using, operating, and administering a warehouse.

Concerted efforts within industry and academia have brought substantial technological progress to the data warehousing task, reflected by the number of commercial tools that exist for each of the three major operations: populating the data warehouse from independent operational databases, storing and managing the data, and analyzing the data to make intelligent business decisions. Despite the plethora of commercial tools, however, several interesting avenues for research remain.

Data cleaning is related to heterogeneous data integration, a problem that has been studied for many years. To date, much emphasis has centered on data inconsistencies rather than schema inconsistencies. Although data cleaning is the subject of recent attention, more work needs to be done to develop domain-independent tools that solve the variety of data-cleaning problems associated with data warehouse development.

Most data mining research has focused on developing algorithms for building more accurate models or for building models faster. The other two stages of the knowledge discovery process—data preparation and mining model deployment—have largely been ignored. Both stages present several interesting problems specifically related to achieving better synergy between database systems and data mining technology. Ultimately, new tools should give analysts more efficient ways to prepare a good data set for achieving a specific goal and more efficient ways to deploy models over the results of arbitrary SQL queries. ✷

More work is required to develop domain-independent tools that solve the data-cleaning problems associated with data warehouse development.

References

1. T. Barclay et al., "Loading Databases Using Dataflow Parallelism," *SIGMOD Record*, Dec. 1994, pp. 72-83.
2. J. Gray et al., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub Totals," *Data Mining and Knowledge Discovery J.*, Apr. 1997, pp. 29-54.
3. S. Agrawal et al., "On the Computation of Multidimensional Aggregates," *Proc. VLDB Conf.*, Morgan Kaufmann, San Francisco, 1996, pp. 506-512.
4. V. Harinarayan, A. Rajaraman, and J.D. Ullman, "Implementing Data Cubes Efficiently," *Proc. SIGMOD Conf.*, ACM Press, New York, 1996, pp. 205-216.
5. L. Breiman et al., *Classification and Regression Trees*, Chapman & Hall/CRC, Boca Raton, Fla., 1984.
6. V. Ganti, J. Gehrke, and R. Ramakrishnan, "Mining Very Large Data Sets," *Computer*, Aug. 1999, pp. 38-45.
7. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco, 2001.
8. S. Sarawagi, "User Adaptive Exploration of OLAP Data Cubes," *Proc. VLDB Conf.*, Morgan Kaufmann, San Francisco, 2000, pp. 307-316.
9. J. Han, "OLAP Mining: An Integration of OLAP with Data Mining," *Proc. IFIP Conf. Data Semantics*, Chapman & Hall/CRC, Boca Raton, Fla., 1997, pp. 1-11.
10. M. Hernandez and S. Stolfo, "The Merge/Purge Problem for Large Databases," *Proc. SIGMOD Conf.*, ACM Press, New York, 1995, pp. 127-138.
11. H. Galhardas et al., "Declarative Data Cleaning: Model, Language, and Algorithms," *Proc. VLDB Conf.*, Morgan Kaufmann, San Francisco, 2001, pp. 371-380.

Surajit Chaudhuri *is a senior researcher and manager of the Data Management, Exploration, and Mining Group at Microsoft Research. His research interests include self-tuning database systems, decision support systems, and integration of text, relational, and semi-structured information. He received a PhD in computer science from Stanford University. Chaudhuri is a member of the IEEE Computer Society and the ACM. Contact him at surajitc@microsoft.com.*

Umeshwar Dayal *is a principal laboratory scientist at Hewlett-Packard Laboratories. His research interests are data mining, business process management, distributed information management, and decision support technologies, especially as applied to e-business. He received a PhD in applied mathematics from Harvard University. Dayal is a member of the ACM and the IEEE. Contact him at dayal@hpl.hp.com.*

Venkatesh Ganti *is a researcher in the Data Management, Exploration, and Mining Group at Microsoft Research. His research interests include mining and monitoring large evolving data sets and decision support systems. He received a PhD in computer science from the University of Wisconsin–Madison. Ganti is a member of the ACM. Contact him at vganti@microsoft.com.*