

Joins and Aggregates Are Critical for Achieving Faster Data Retrieval in a Dimensional Data Warehouse

By Craig Abramson and Thomas Mascoli



Data warehouses are becoming overloaded with information. From Web sites alone, a company can generate several gigabytes of data each day. Now combine this with such data as store inventory, monthly sales, and customer addresses, and you've got all the ingredients for a bottleneck. Maintaining aggregate tables and separating descriptive data from factual information are techniques that can speed the data query process in a dimensional data warehouse. Aggregates reduce the amount of source data that must be processed by certain queries, while joins allow you to combine descriptive information with voluminous factual data at the latest possible moment during a processing sequence, thus minimizing storage and throughput requirements. However, in order for these techniques to be effective, you first have to prepare the data in the data warehouse.

Preparing Fact and Dimensional Tables

According to data warehousing expert Ralph Kimball, the only viable way to deliver data efficiently to end users is to employ dimensional modeling in the data warehouse.

Applied properly, this model can enhance user understandability, query performance and flexibility by presenting data in a standard framework. (Kimball, 1998)

Fact and dimensional tables are the main components of the dimensional model. The fact table is the primary table in this model and it contains measurements of the business that are usually quantitative, such as “units sold.” Because fact tables accumulate large amounts of data resulting from individual transactions, they are designed to contain little or no descriptive information. Instead, a composite key that includes a subset of at least two or more component foreign keys identifies each transaction record. Each foreign key refers to a specific entry in a dimensional table. The dimensional table contains descriptive information about some aspect of the transaction, such as details about the product sold. This descriptive information is often the basis for constraining and grouping within data warehouse queries. For example, a product dimension table can include such information as product description, size, and category. (Kimball, 1998)

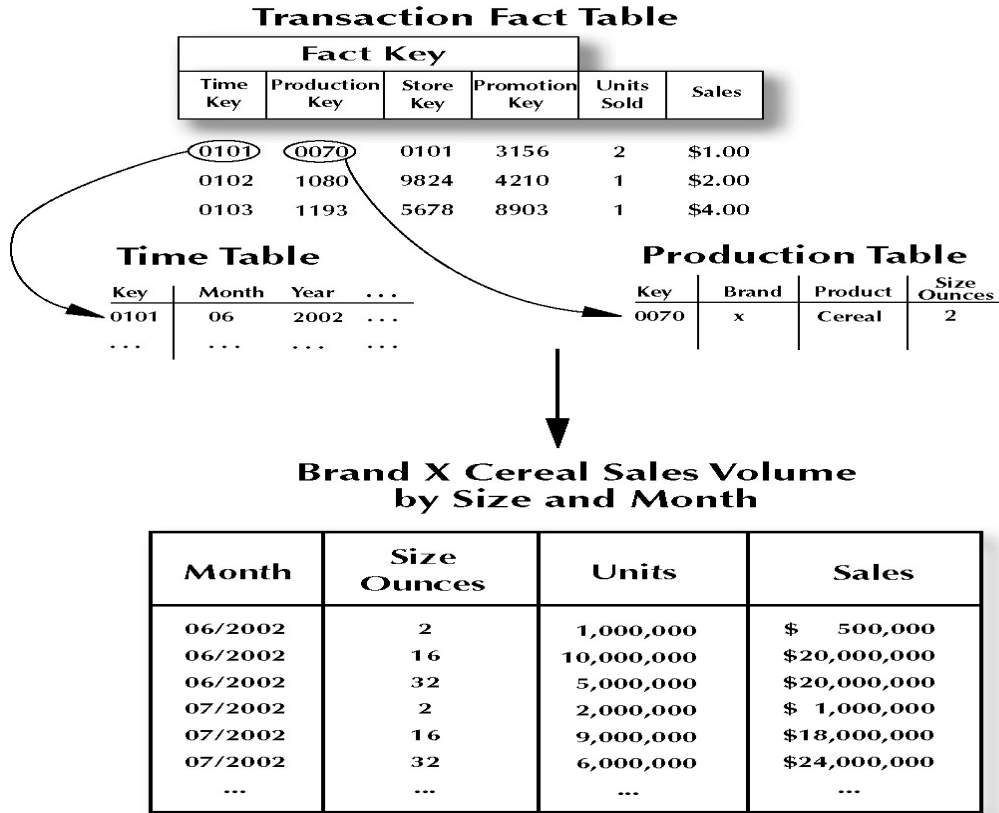
Combining Fact and Dimensional Tables Using a Join

Joins are used to look up descriptive dimensional information about the factual data only when this descriptive information is needed in response to a query. The primary key for each fact record consists of the keys for each of the dimensional records associated with it. In the fact record, each of the dimensional component keys are known as foreign keys because they are a primary key in another table. The dimensional component foreign key of the composite fact record key matches the fact record with the corresponding dimensional record, allowing for the retrieval of the descriptive information.

Suppose a grocery store chain wanted to determine the effectiveness of a marketing campaign that promoted the convenience of individual serving size packages of brand X cereals. They want to see monthly sales over the past year and compare sales of the product in the larger package sizes. Each factual record's composite primary key consists of all of the foreign keys that refer to associated dimension tables including the time dimension table primary key and the product dimension table primary key. The time and product table keys by themselves do not reveal in which month the sale occurred, or the brand and package size of the product sold. To obtain this information, it is necessary to join each factual record with its corresponding time and product dimension record using these keys. Once joined, the factual information is now complemented by the needed time and product details to complete the query, namely the month in which the sale occurred, the product brand name, and the product package size. Now the factual and descriptive information may be summarized in a meaningful form.

[Figure 1]

Determining the Effectiveness of a Marketing Campaign



Preprocessing Fact and Dimension Tables

Prior to loading the fact and dimension tables into the data warehouse, it is critical to preprocess the data to get it into the proper form for the dimensional data warehouse, which will ultimately help you to reduce the elapsed time needed to retrieve the information. The data staging area is used to perform the necessary preprocessing, such as cleaning and merging records. The dimensions are usually processed before the facts in order to maintain referential integrity. Dimensional data processing may

include generation of more compact surrogate keys by a look up (join) operation using the existing production key in the dimensional record. A surrogate key will be generated for new dimensional entries. Dimensional data from different sources is often merged, converted to a uniform format and summarized to eliminate duplicates before it is loaded. Cleansing operations may also be performed to remove erroneous data before loading. For example, sorting dimensional records on a textual attribute may reveal variations in spellings. In this way, erroneous records may be detected and deleted. (Kimball, April 1998)

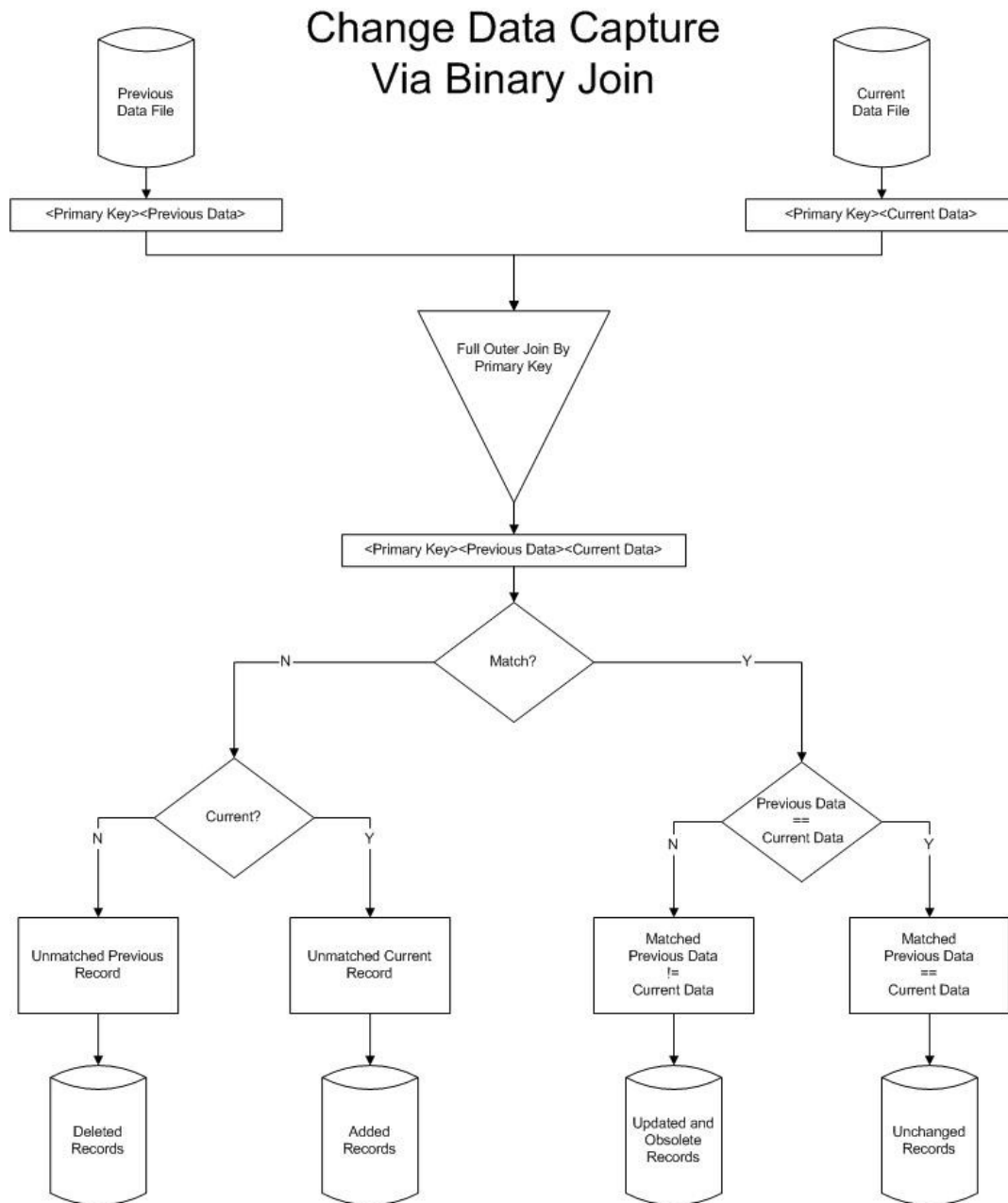
Once dimensional processing is complete, that attention shifts to fact processing. Factual records may also contain a production key that corresponds to a more compact data warehouse surrogate key. A lookup operation will be required to obtain the correct data warehouse surrogate key. Factual data should be summarized to the minimal level of granularity required by the data warehouse. For example, if data warehouse queries refer to time increments of no less than one day, and the raw factual data consisted of individual transactions that occurred during the day, it would be appropriate to aggregate the data by day before loading it. (Kimball, April 1998) When factual information entered into the warehouse consists of a period snapshot of data from online transaction processing (such as membership records), and if a relatively small percentage of these records are added, changed or deleted between data warehouse load times, it may be more efficient to compare the current and previous data snapshot to identify and load only added and changed records.

Joins and Merges Are Effective for Capturing Changes to Large Operational Data Stores

Data warehouses are updated at scheduled times with new data from the online transactional database. One way to perform this update is to replace the information in the data warehouse with all of the data in the online transactional database. However, if only a small fraction of the records in the database change during the scheduled updates, then it would be much more efficient to modify the data warehouse information with only the changed or added records.

Using a join is an effective technique for comparing the previously loaded database with the current database to capture deleted, updated or changed records. A join will match the primary key of the previously loaded record with its corresponding new record. The data portions of the record can be compared to determine if they have changed. For instance, if a previously loaded record has no matching record in the new file, then it is deleted. If a new record has no match in the previously loaded data, then it is added as a new record. If a previous record matches with a new record but the corresponding data has changed, then the record is updated in the data warehouse. This technique will help reduce significant amounts of elapsed time when performing change data capture or delta processing.

[Figure 2]

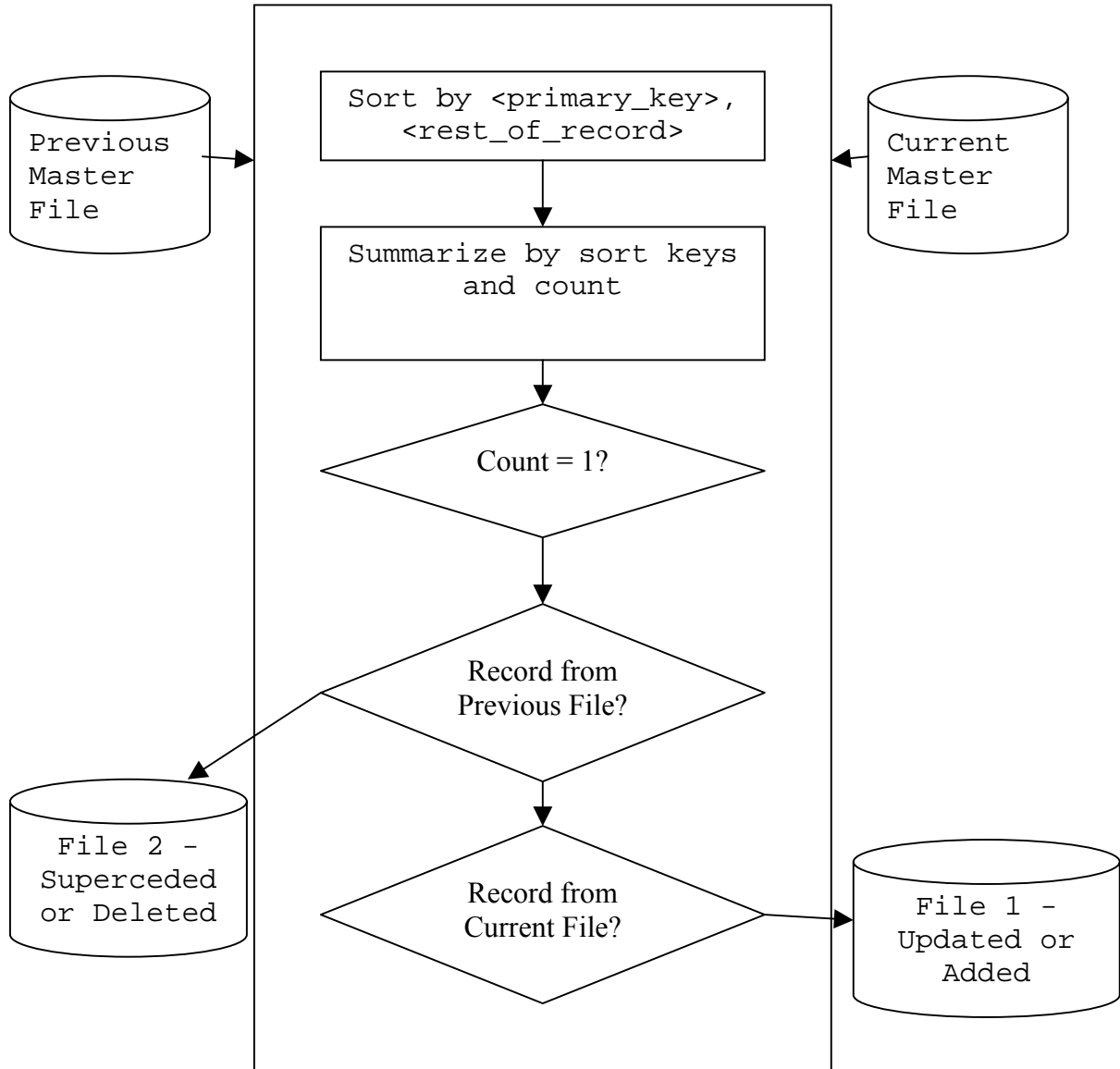


A sort, merge and summarize process is also effective in capturing changes. Previous snapshot data is summarized with current snapshot data using the entire record as the summary key. A counter is initialized to '1' and appended to each record before the

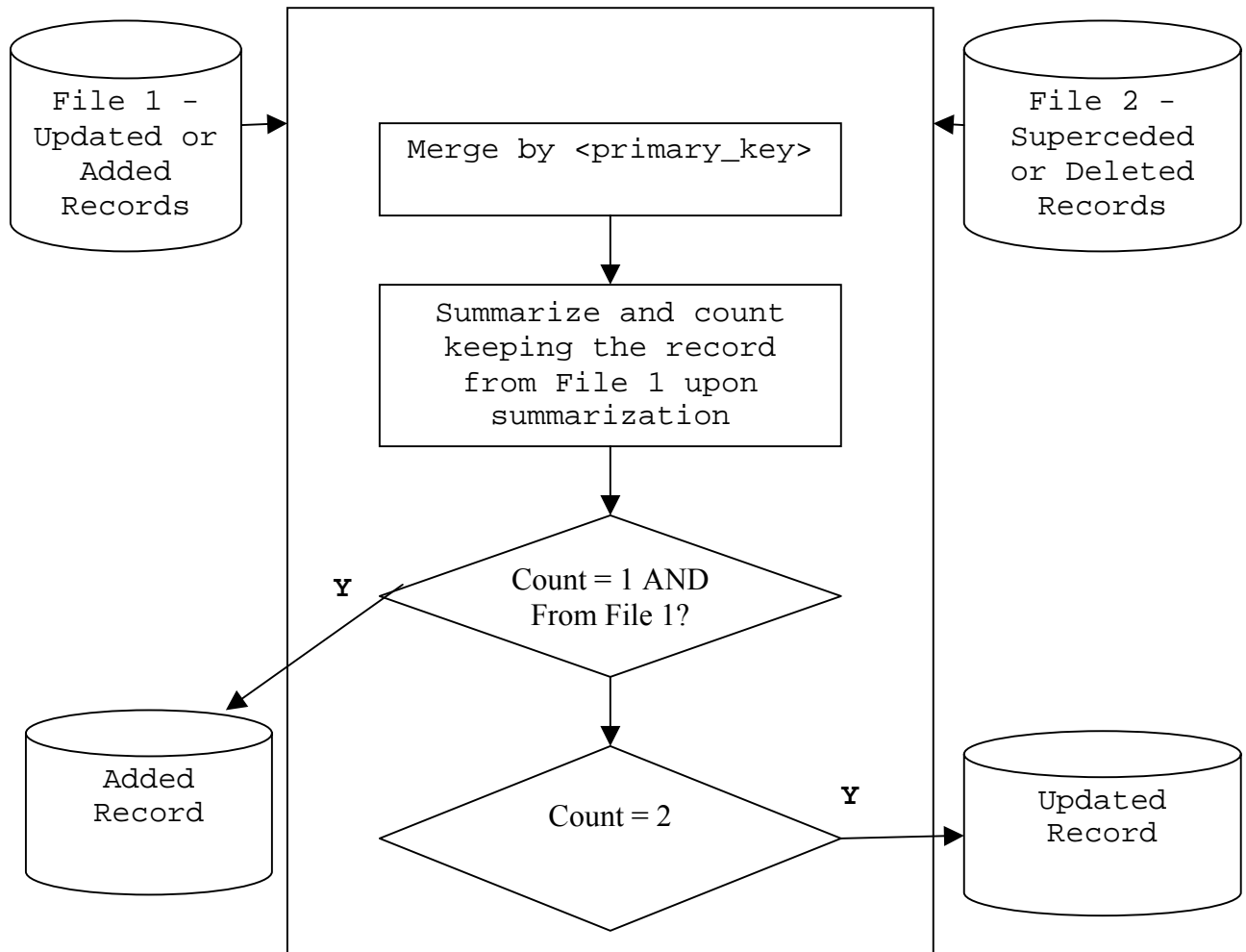
summarization, and totaled during the summarization process. Summarized records with a count of '2' are unchanged between the previous and current snapshots and are usually eliminated from further processing. Summarized records originating from the current file with a count of '1' are updated or added records. Summarized records with a count of '1' originating from the previous snapshot are either deleted records or obsolete records (records from the previous snapshot with a corresponding updated record from the current snapshot). A final processing step can partition the added, changed and deleted record types to individual targets.

[Figure 3 and 4]

Using the Sort/Merge/Summarize Process to Capture Changes Step 1



Using the Sort/Merge/Summarize Process to Capture Changes
Step 2



General Parts Slashes Sales and Inventory Data Processing Time by 96%

By using joins and other ETL utility functions to facilitate its data warehouse processing, General Parts, Inc., was able to reduce the elapsed time of their data warehouse update process by 96%. General Parts is a distributor of CARQUEST replacement automotive parts, supplies and tools for every make and model of foreign and domestic car, truck, bus, and farm or industrial vehicle. At their corporate headquarters in Raleigh, NC, the company receives inventory and sales data from its 40 distribution centers and approximately 2800 stores in the U.S. and Canada. Processing all of this data was taking a full month, which meant that current inventory and sales information was not available in a timely manner.

Mark Walley, the Data Warehouse Manager at General Parts, reported that the warehouse was receiving approximately 3500 to 4000 files a day, ranging in size from 5MB up to 50-60MB. At the time, they were pre-processing the data using custom Perl and C programs and then loading it into their database for analysis. Walley brought in Mike Ames, a consultant, who suggested that they try SyncSort UNIX (from Syncsort Inc., Woodcliff Lake, NJ) to speed up the inventory processing so that the data would be available to company management as soon as possible. Ames stated, "We looked at several ETL tools, but given our past experience, we knew that they weren't going to scale for that volume of data. It really came down to how much data we could push through SyncSort." Walley continued, "We're looking at between 250 and 300 gigabytes a day that requires transformation for loading into the data warehouse." Ames added, "Essentially what we do is use SyncSort to prepare the files for loading."

The procedure now begins with several servers using modems to dial into the individual stores and distribution centers in order to pull down the raw database, which includes sales and inventory files. This process allows them to mirror the database in flat files, which are loaded into the local storage system on a Sun E450 with four processors. SyncSort processes the raw data, performing all of the company's dimension management and fact preparation. For the dimension management processing, Walley first orders the files, then uses a join to perform their change data capture procedure (delta processing), and then formats the data to prepare the files for loading. The join adds the appropriate surrogate keys that are missing in the raw data, converting it into dimensional form. Walley then formats the data and prepares the files for loading.

Ames explains, "We perform all of our dimension management first and then handle our facts preparation. For the fact preparation, we'll have a part number and we'll use SyncSort to join it with our parts file in order to add the dimension keys. Once the dimension management and fact preparation is complete, we'll have inherent referential integrity in all of our flat files. We'll have separate parts, customer, line item and inventory flat files ready for loading into the Oracle 8i database."

Ames continued, "With the database, we do a big load, which may take an hour or less. We're loading some inactive tables and creating indexes. Then we do an analyze to gather statistics on the database, which takes the most time. Using this process, the database is a 22 by 7 operation." Overall, General Parts was able to reduce the

processing time by 96%. “The original system that we replaced was doing all of this processing in a month,” Ames stated. “We brought in SyncSort and did everything in 22 hours.”

Ames believes the impact of this new process on General Parts’ business goes far beyond saving processing time. “The company went from not knowing what stores had what inventory to being able to quickly check inventory. They also are able to improve their negotiations with vendors because they can find out the exact number of parts they’ve ordered in the past. Now the data warehouse also provides an up-to-date snapshot of sales. For example, we can go in and see how much sales a particular store has had in the past three months. The new process has had a huge impact.”

Building Aggregates in the Dimensional Model

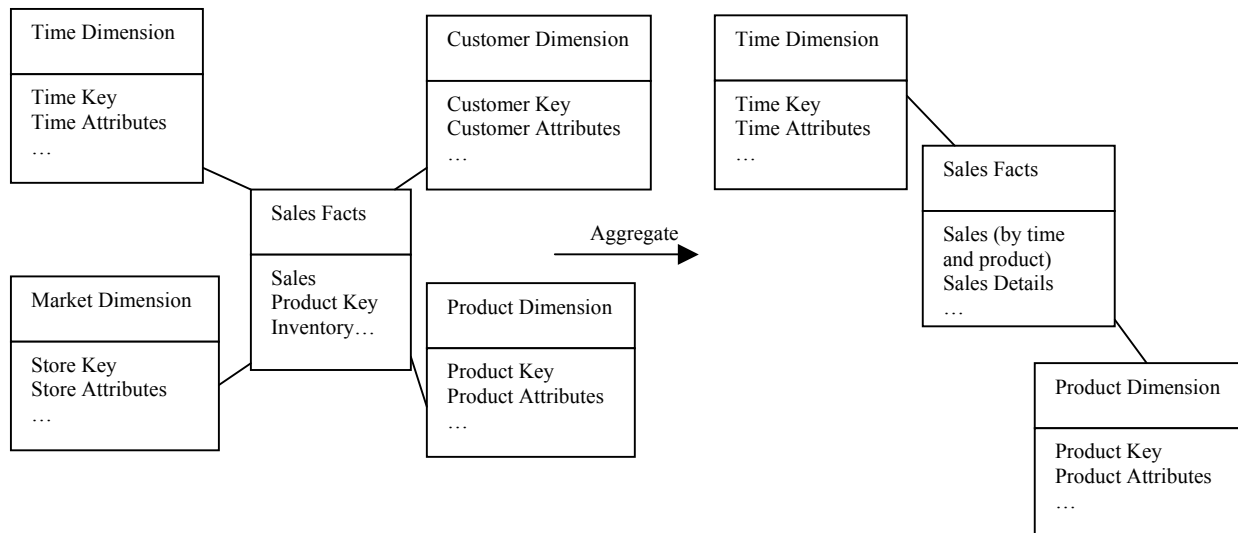
Another way to improve performance when processing large amounts of data in a data warehouse is to build aggregates. A query answered from base-level data can take hours and involve millions of data records and millions of calculations. With pre-calculated aggregates, the same query can be answered in seconds with just a few records and calculations.

Building aggregates from dimensional models is a simple approach because each large fact table can provide numerous aggregates with a predictable structure and a predictable relationship to the base fact table. There are three basic ways to build aggregates from the fact table in this type of model. The first approach is to exclude one

or more dimensions when summarizing a fact table. This is the easiest type of aggregate to create because the dimensional data doesn't need to be accessed. The aggregate can also provide significant performance advantages over the base fact table. (Corr, 2002)

[Figure 5]

Figure 5: Creating Aggregates by Excluding One or More Dimensions When Summarizing a Fact Table

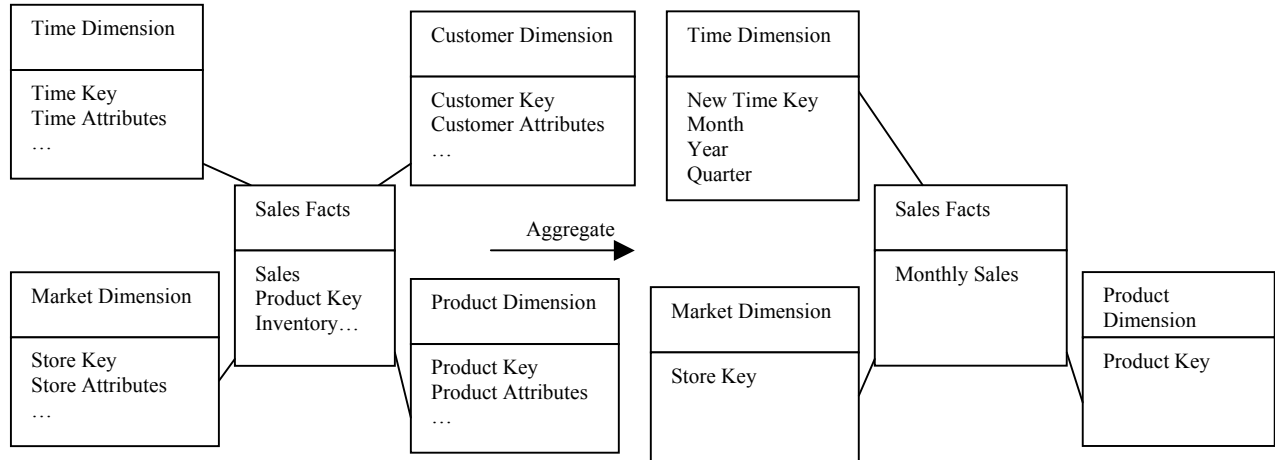


Keys				Sales	Cost	Units
Customer #	Product #	Store #	Time #			
3	4	8	133	10	9	2
4	4	10	133	20	18	4

Keys		Sales	Cost	Units
Production #	Time #			
4	133	30	27	7

The second approach is to have one or more dimensions replaced by rolled up versions of themselves. For instance, you can use this approach to create a monthly-product-sales-by-store summary of the original fact table. You're aggregating all of the daily sales records into monthly records and reducing the size of the fact table, but you're still able to do dimensional analysis by time at the month, quarter, and year levels. While this helps achieve a balance between performance and usability, it will require maintenance of the physical dimensions and keys. The third approach eliminates this maintenance, replacing dimensional keys with high-level dimensional attributes. This type of aggregate should only be used for high-level summaries where the dimensional attributes are limited and short. An example of this would be a quarterly-product line-unit sales summary. This approach also accelerates query performance by eliminating the need to create joins for query processing. (Corr, 2002)

[Figure 6]
 Creating Aggregates by Replacing Dimensions with Rolled Up Versions of Themselves



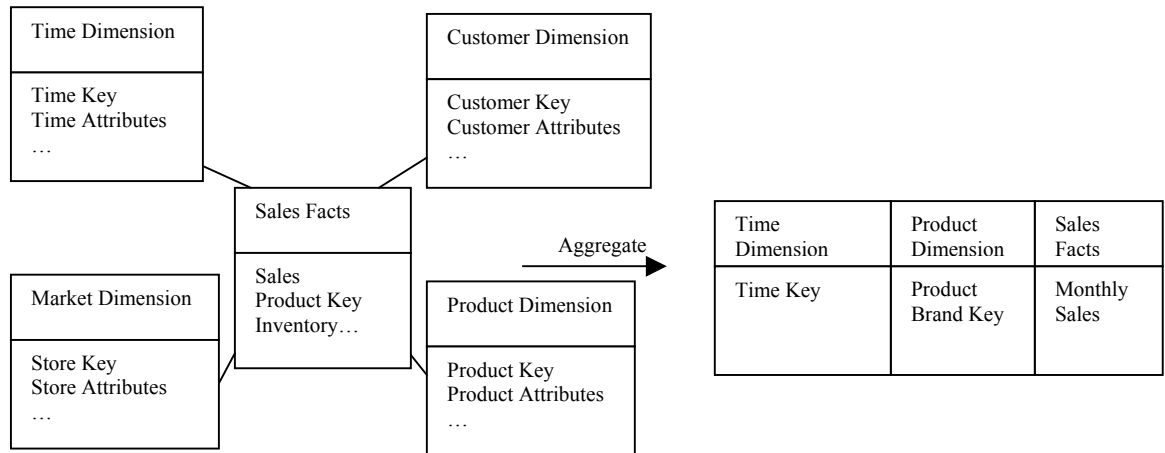
Keys				Sales	Cost	Units
Customer #	Product #	Store #	Time #			
1	5	6	10	10	9	2
2	5	6	11	20	18	4
3	5	6	12	30	27	6

Time Details				
Time Key	Day	Month	Year	Quarter
10	9	06	2003	2
11	10	06	2003	2
12	11	06	2003	2
...				

New Time Key			
Time Key	Month	Year	Quarter
20	6	2003	2

Product #	Store #	Month/Year	Sales	Cost	Units
5	6	06/2003	60	54	12

[Figure 7]
 Creating Aggregates by Replacing Dimensional Keys with High-Level Dimensional Attributes.



Keys						
Customer #	Product #	Store #	Time #	Sales	Cost	Units
1	5	6	10	10	9	2
2	5	6	11	20	18	4
3	5	6	12	30	27	6

New Time Key	Product Line	Unit Sales
20	5	12

Nabisco Builds Aggregates to Dramatically Improve Data Warehouse Query

Performance

One company that has benefited from the use of data aggregates is the Nabisco Biscuit & Snacks Group, based in Northern New Jersey. Comprised of three divisions -- the Nabisco Biscuit Division, the Confections Division, and the Planters Division -- the company generates an enormous amount of customer, manufacturing and sales data. After all, Nabisco sells such icon brands as Oreo cookies, Ritz crackers, Life Savers candies, Altoids intense mints, and Planters nuts.

The data aggregates are used by Nabisco's sales, finance and marketing staffs to analyze both customer SKU and manufacturing data. They consist of current year to date as well as the previous two years' sales information. To create these aggregates, Nabisco used a combination of UNIX joins and SyncSort UNIX running on an IBM RISC 6000 Silver Node with four gigabytes of memory and four processors. The computer system stored Nabisco's extensive fact table that consisted of 750 gigabytes of data.

Although this process was providing the results the company needed, the Information Services team at Nabisco realized they had a potential problem. They were limited in the amount of time they had to create the aggregates but were receiving a growing number of requests. "With a batch window of only five to six hours a night, we needed a faster methodology to build the multiple new aggregates that are continually required by our users," said Neil Gurwitz, Project Manager of Information Services. After an

extensive review of the products available, the team decided that Sigma (from Syncsort Inc., Woodcliff Lake, NJ) offered the best solution. This high-performance utility allowed the team to create and manage data aggregates rapidly, easily, and accurately.

“We chose Sigma because of its demonstrated capabilities and because of Syncsort’s reputation for high quality products and service, which we’ve experienced firsthand during years of using their data management products.” The team runs Sigma on an IBM RISC 6000. The computer system utilizes AIX 4.4.3, and the warehouse database features Informix Red Brick Release 6.0.2.

Nabisco uses the product to build a total of 13 aggregates that consist of current and two prior years of sales history. These base tables are rebuilt every four weeks due to store realignments and are updated with daily sales information six times a week.

“Specifically, the aggregates are rolling up from the store or market level and consist of: team territory, team class of trade, region class of trade, area class of trade, branch and account / sub account on both the SKU and manufacturing. In addition, we have just completed one aggregate off of the account / sub account SKU aggregate which rolls up to a product grouping.” Approximately 1,100 users generate 45,000 reports per month using this information. The application is written in Powerbuilder and has dynamic online reporting and dynamic batch reporting.

Since its implementation, Sigma has helped Nabisco to reduce by up to 40 percent the amount of time it takes to define and maintain aggregates and has dramatically cut

response time for database queries. With this approach to creating new aggregates, Nabisco is confident that they have the ability to provide the answers that their employees need in the fastest and most efficient way.

Getting the Speed You Need

The experiences of Nabisco and General Parts demonstrate that the process of preparing data for a dimensional data warehouse can be significantly reduced by the effective use of aggregation and join processing performed by a tool optimized for fast sequential processing. Properly prepared factual and dimensional data can significantly improve query performance in a data warehouse by minimizing the amount of data that must be processed in response to a query. In many cases, processing performed in the data warehouse staging area is sequential. Because of this, you may want to test a third-party tool that is designed to handle sequential processing. The tool can help reduce the elapsed time needed to process large amounts of data, which in turn can dramatically improve the performance of the database.

References

Corr, Lawrence. "Lost, Shrunken, And Collapsed. Why and how to create the three types of aggregates in a dimensional data warehouse," www.intelligententerprise.com (2002), 1-2

Kimball, Ralph, Laura Reeves, Margy Ross, and Warren Thornthwaite, "The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses," John Wiley & Sons, (1998), 17, 140

Kimball, Ralph, "Is Data Staging Relational? Or Does It Have More to Do with Sequential Processing?," DBMS, (April, 1998)

Biographies

Craig Abramson is a technical analyst at Syncsort Incorporated, focusing on the latest data warehouse performance solutions. He has over 7 years experience in the field working on projects dealing with data warehousing, database management and Web log processing.

Craig Abramson

Technical Analyst

Syncsort Incorporated

50 Tice Boulevard

Woodcliff Lake, NJ 07677

201-930-9700 ext. 308

Email: cabramson@syncsort.com

Thomas Mascoli is a Senior Software Development Engineer in Technical Support at Syncsort Incorporated. His responsibilities include implementing product fixes and enhancements, and assisting customers with high performance data processing solutions utilizing Syncsort's suite of products for UNIX and Windows.

Thomas Mascoli

Senior Software Development Engineer

Syncsort Incorporated

50 Tice Boulevard

Woodcliff Lake, NJ 07677

201-930-9700 ext. 296

Email: tmascoli@syncsort.com

For more information
or to arrange a free trial,
call Syncsort at
(201) 930-8200
or visit the Syncsort
web site at
www.syncsort.com