

# Capturing Changes: Using SyncSort to Reduce the Elapsed Time of a Delta Processing Application

By Craig Abramson and Thomas Mascoli



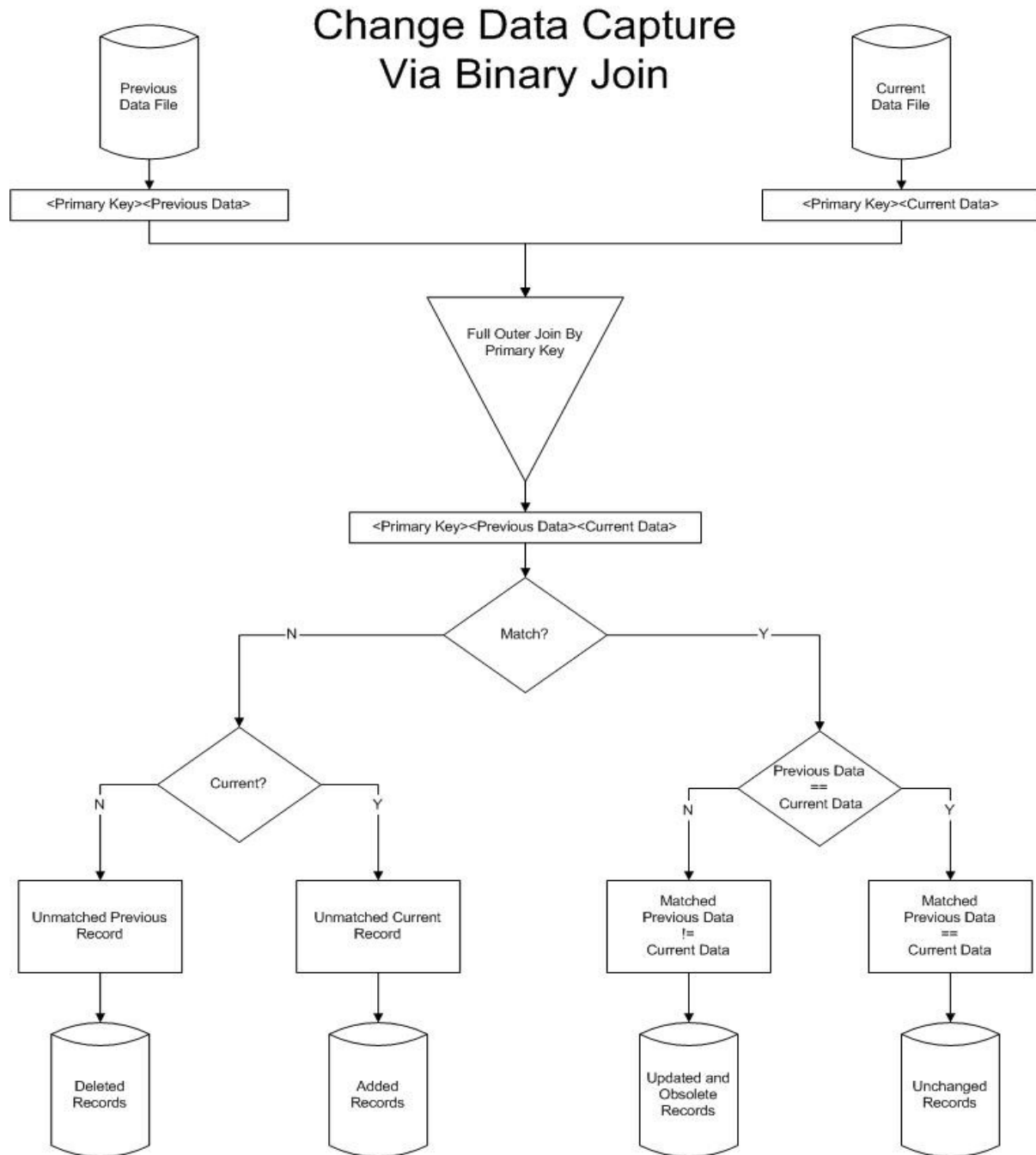
Imagine if on top of all your other responsibilities at work, your boss asked you to manage a vending machine in the office. Hoping to get a boost in pay, you take on the task and decide that you're going to check the inventory in the machine every Friday to replenish supplies. As you count the inventory one day, you notice that 10 chocolate bars, 12 bags of chips, and 3 packs of gum were purchased during the week. No one bought the protein bars or the cookies in this period. Naturally, you're only concerned about the inventory that has changed and needs to be replaced. Now imagine that instead of a vending machine, you're managing a data warehouse with over 100 gigabytes of data. Each week you receive a snapshot of your entire OLTP database, containing new, changed, and unchanged records. Instead of taking the time to replace the entire 100 gigabytes of data in your data warehouse, you can compare the last snapshot file with the current one to identify the changes that have occurred. Only the data that has changed is updated in the data warehouse. This procedure is called change data capture or delta processing.

## **Techniques for Capturing Changes to Large Operational Data Stores**

### **Using Joins**

Dimensional data warehouses are updated at scheduled times with new data from the online transactional database. 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 by loading 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 some of the data in the record 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]

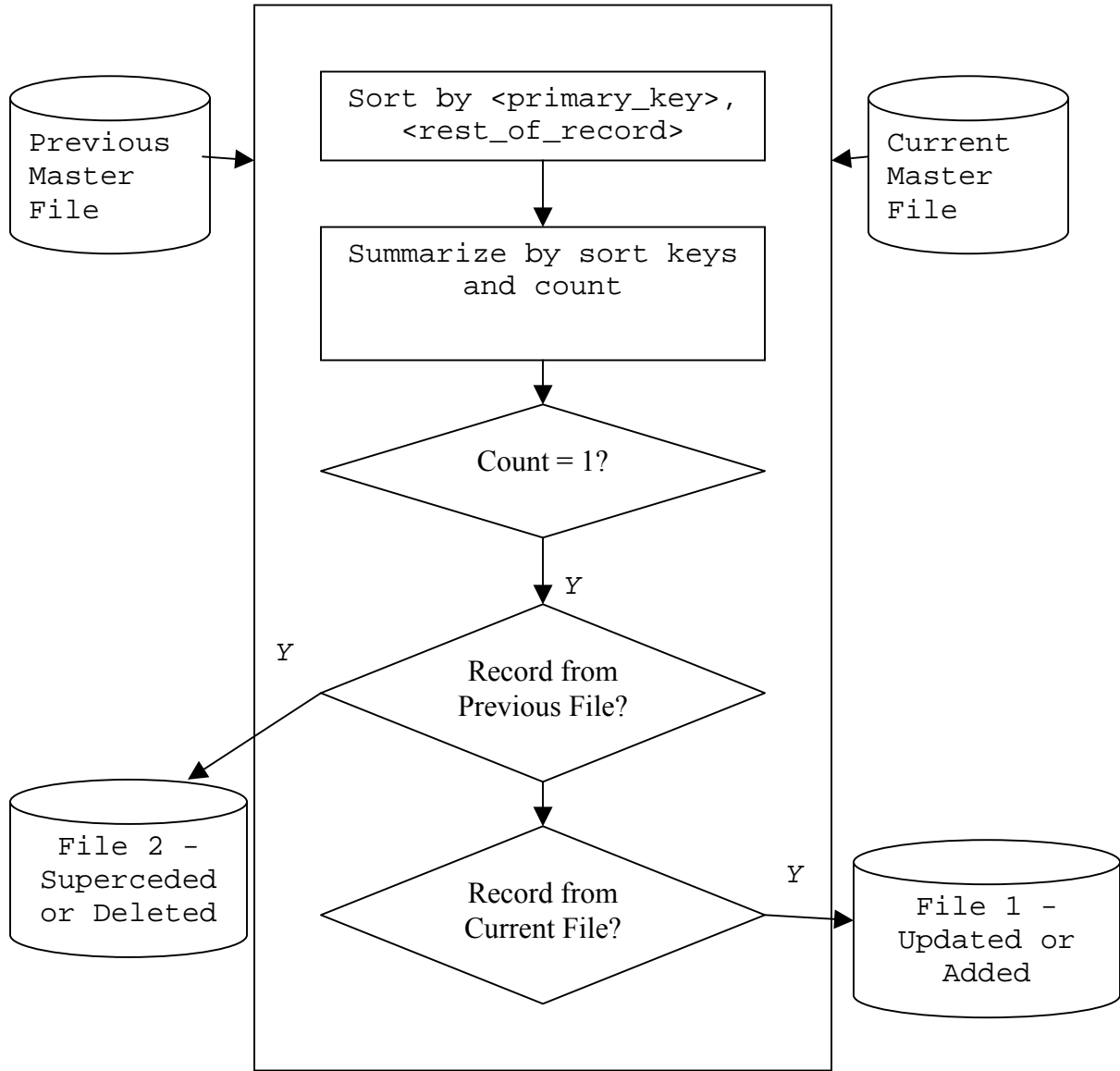


### ***Using a sort, merge and summarize process***

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.

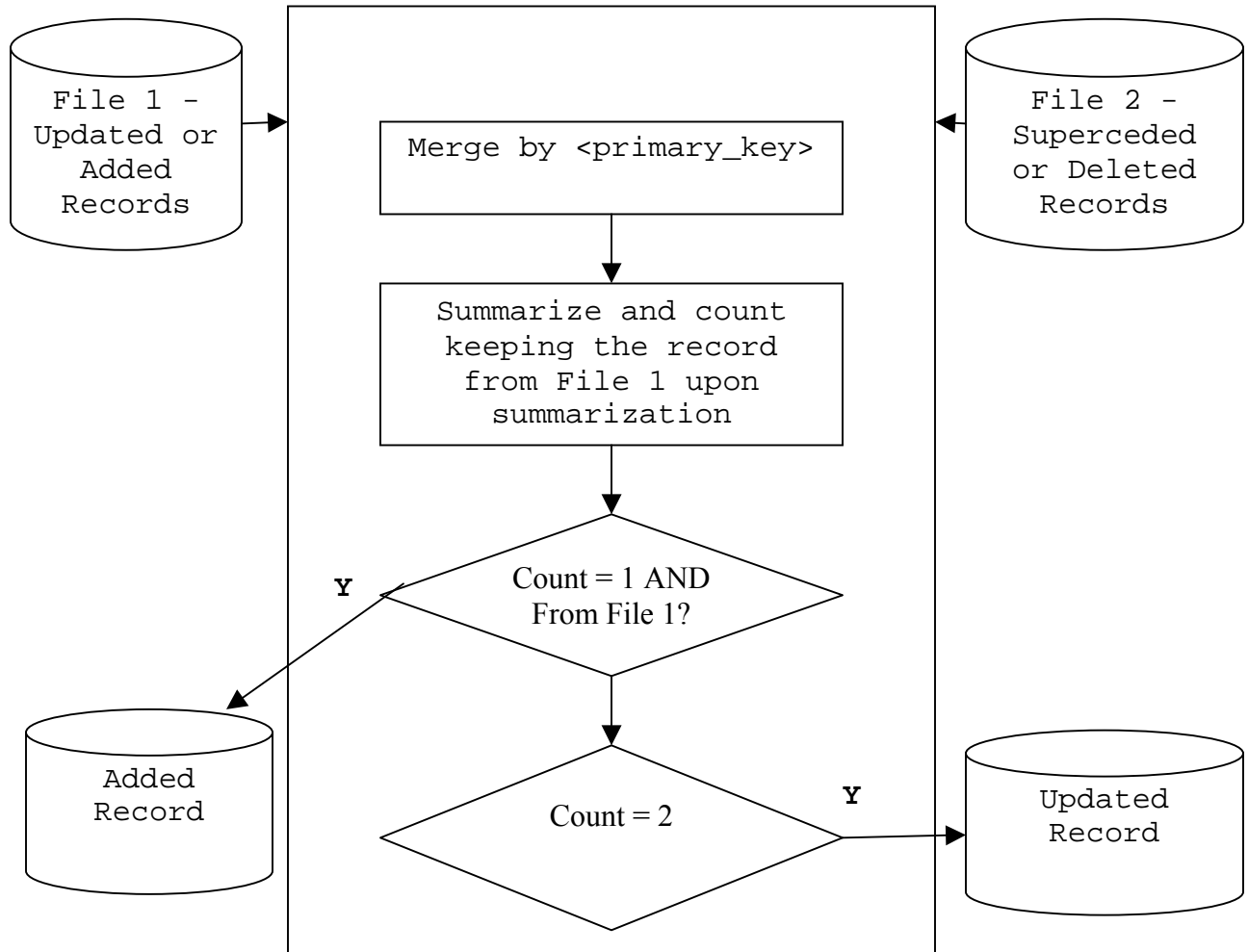
[Figure 3]

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



[Figure 4]

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



## **AT&T Utilizes Both Summarization and Joins to Improve Delta Processing Application**

AT&T is among the world's premier voice, video and data communications companies with annual revenues of nearly \$66 billion in 2000. The company runs the largest, most sophisticated communications network, and is the largest cable operator in the U.S.

AT&T is also the leading supplier of data and Internet services for businesses.

Receiving an enormous amount of new and revised customer and inventory data each day, technical developer Hyon Park was charged with the task of processing the inventory information for the facilities at AT&T on a project called LDD (Local Data Distribution). For this project, he needed to compare two database files and identify deleted, new, identical and updated records. In order to perform the necessary delta processing, Hyon decided to use SyncSort.

The summarization and join features of SyncSort were used to create a script for this project. Summarization aggregates records with duplicate sort keys into a single record, counting records and totaling numeric fields. It also eliminates duplicate-keyed records. The join allows AT&T to take records from two different sources and create target records containing data from both. It quickly and easily matches keys in multiple, differently-formatted files/tables in order to create the joined output file/table in a user-specified format. SyncSort supports left, right, inner, and outer joins.

Hyon uses SyncSort to perform the comparisons between AT&T's Local Network Inventory data with another day of data and outputs the difference between the two

files. The daily processing file size is around three gigabytes (the two files are 1.5 gigabytes each). Seven different jobs are run using this data and take between one and 30 minutes to complete. The largest job is a join between a 500MB file and another 500MB file, creating a one gigabyte output file. By using SyncSort in their process, AT&T is able to quickly maintain the most current information in their database.

### **SyncSort Speeds Delta Processing for Mortgage Company**

A mortgage company needed to perform delta processing on its existing loans and loan applications database. The company had a Sun 6800 with 32 gigabytes, and eight CPUs per domain, with an EMC array for storage. The consultant working on the project anticipated detecting deltas between a 22 gigabyte current file and a previous master file. He estimated that there would be 15,000 changes and 5,000 new records per day. Data would come in from the mainframe in EBCDIC format and would be converted to ASCII using Informatica and a COBOL copybook. The application was taking too much time so the company turned to SyncSort to improve performance.

There were two approaches that could have been taken using SyncSort to detect the updates and added records. One would detect them collectively using a single-pass application while the other would detect them separately using a two-pass application. Both approaches required only one sort of the data, which would be the most time-consuming operation, and used SyncSort's stable sort and summarization features. The effect of using summarize and stable sort together was that the first record encountered

in the source was kept among summarized records. Since the company needed to differentiate between the updated or added records, the second approach was used.

The process would begin by running two SyncSort copy applications that would reformat the previous and current data files (fixed / EBCDIC) to the fields of interest and output the results through pipes. A SyncSort sort / summarize would detect and eliminate fields in the previous and current data streams that were unchanged, and create a file with superceded and deleted records, and a file with added and updated records. The application required that a workspace area be identified in the file system via the /workspace option. This was the only step that required a working file storage. This ended pass one.

Pass two began with the outputs from the previous step being read from the file into a SyncSort / merge. Resorting was not required because the previous application sorted the data by the primary key first (loan number). This saved considerable processing resources. The file separated the added records from the updated records through named pipes to FilePort applications. Two FilePort applications were then used to replace Informatica and perform the translation from EBCDIC to ASCII. Each record was terminated by a linefeed, numeric fields were converted to displayable form and fields were pipe delimited.

Given the preponderance of spaces in numeric fields, it was best to do the FilePort translation last. Otherwise, the application would incur the overhead of detecting invalid

numeric fields twice during the initial FilePort translation and when SyncSort converted the packed fields to displayable. Also, FilePort could automatically insert the delimiters. Placeholder fields were next inserted using a copy step and then SyncSort performed an entire record reformat using the reformat feature.

## **Retailer Turns to Visual SyncSort to Reduce the Elapsed Time of Its Delta**

### **Processing Application**

A rapidly expanding national retail chain was using a Perl script to perform delta processing and the process required over three hours to complete. Loading the deltas to the target took another hour. In order to reduce this window, the company decided to integrate SyncSort into the process to achieve the necessary performance gains. Six applications were created using Visual SyncSort, the graphical user interface for SyncSort. The applications were executed using a Korn shell script.

During the process, updates and additions were to be captured from two files representing product order status for a previous and current time period. The records in each file had the same structure - UNIX text format with pipe-delimited fields. Each record included the order number with descriptive and status information. SyncSort would execute two sort applications in parallel, each reading the input files and sorting the data piping to two different merge applications, which detected and partitioned duplicate records within each source file. The data was sorted such that it would not have to be re-sorted in subsequent steps, thus reducing resource requirements.

The de-duplicated data from each input file was then read by one merge application, which created three output files. The first contained records that existed in both files. These would be the unchanged records between periods. The second contained records that existed only in the previous files. These records would be either obsolete versions of records that were updated (the "before" version of a changed order record) or deleted records no longer present in the current file (an order that had been cancelled or fulfilled and purged from the current order status). The third output would contain records that existed only in the current file. This would include the current version of updated order records and orders initiated in the current cycle that therefore did not exist in the previous file. The final SyncSort merge processed the second and third outputs from the previous step to produce an output containing the updated order records and the added order records. These were then loaded into an Oracle table. SyncSort reduced the delta processing time from over three hours to less than 30 minutes, an 83% improvement.

## **Summary**

As shown by the experience of the three companies, SyncSort provides a broad range of functionality and a significant reduction in elapsed time for delta processing applications. There are numerous approaches that can be taken with SyncSort, including using the sort, merge and summarization features. These approaches will help you identify new, changed or unchanged records so that you can quickly update your database and analyze the data.

## **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](mailto: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](mailto: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](http://www.syncsort.com)**