IJDCST @Dec Issue- V-1, I-8, SW-17
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Merging and indexing mechanisms for speedy processing in Flat File handling

[1]Kommunuri.Ravi Kumar, [2]Sravanthi.Kasturi, [3]RavindraPrasad.Gundala

[1]Sr.Asst.Professor,CSE dept, Mandava Engineering College,Jaggayya Pet
[2]Asst.Professor,CSE dept, Mandava Engineering College,Jaggayya Pet
[3]Asst.Professor,CSE dept, Mandava Engineering College,Jaggayya Pet

Abstract: Processing of the data in fastest manner is essential part of data warehousing environment. In the data warehousing process Extraction transformation and loading plays a vital role. The paper describes the usage of FLAT file source so as to speed up the ETL process. The benefit of the methodology that we are going to provide will improves the efficiency of data ware housing environment in case of storage and processing time. The IT companies are looking for better performance for the data warehousing processing as the DWH environment holds bulk amount of the data, we concentrate on FLAT file source in ETL processing because it is the second highest source used by the DWH environment.

Index Terms: Flat Files, Etl, Merging, Indexing, DWH.

I. INTRODUCTION

Industry has huge amount of operational data Knowledge worker wants to turn this data into useful information. This information is used by them to support strategic decision making. It is a platform for consolidated historical data for analysis. It stores data of good quality so that knowledge worker can make correct decisions. According to BARRY DEVLIN single, complete and consistent store of data obtained from a variety of different sources made available to end users in what they can understand and use in a business context. Data should be integrated across the enterprise. The source data is taken from legacy and kind of the data is numerical in most of the cases. External data may be included, often purchased from third-party sources. Data are moved from source to target data bases. AETL processing is very costly, time consuming part of data warehousing. Some sample ETL tools are Teradata Warehouse Builder from Terawatt, SAS System from SAS Institute, Power Mart/Power Center from Informatics, and Sagent Solution from Sagest Software [1].

II. IMPORTANCE OF FLAT FILE IN ETL

Improvement of data warehousing process is possible through FLAT files. Compared with data base source FLAT files are better in case of processing. The Extraction time for 2000 records using flat file takes 16.37ms where data base file requires 25ms. The Flat File needs

less storage space on disk compared to data base table. Data extraction using direct data base file requires more time compared to that of the flat file data. Improvement during extraction process is achieved by converting the data base table into flat file and extract [1]. DBMS heterogeneity problem is handled with flat file usage. Here heterogeneity refers to various formats of the data. Reduction of overhead and programming efforts possible through flat file processing [2]. In addition to flat files usage of indices will greatly improve the performance. High through-put experiments such as micro array monitor multiple objects at the same time. With out Indexing, the brute-force method for nested file scanning took 81.59s where as with indices it took 20.89 seconds. Use of Indices on Flat Files Can Improves System's performance and functionality [2] Input Flat File data must be transformed into a uniform format which could be more suitable for analytical purposes. The Advantage of a Flat File is that it takes up less space than a structured file. Flat File Management such as updation/Deletion/Insertion is easy when transform into structured format. Flat File is One of the semi structured plain text file to store it in dwh flat file doesn't store directly in the dwh [3] The index should able to operate with other indexes to filtering out the records before accessing original data. Clustering index is good for range based queries but requires sorted data. If the set of key-value pairs is fixed and known ahead of time, hash based indexing is best for equality selections. Three possible queries are Full table Scan, exact match and statistical type query. Cluster index is better for full table scanning. Hash based indexing is good for exact match query. Bitmap index is good for executing statistical query execution. Bitmap requires less space if data have less cardinality, it can also be compressible if once created. [5]

The following diagram gives the usage percentages of various source data in the ETL processing.

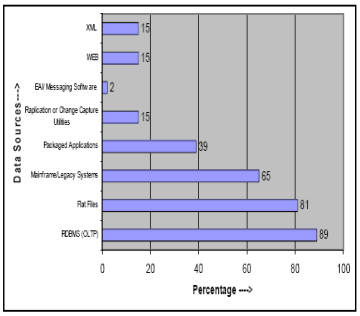

Figure 1: Source usage % in ETL processing.

- Relational Usage—89%
- Flat Files Usage—81%
- Mainframe/Legacy—65%
- Packaged Applications—39%
- XML & WEB—15%

Flat files are used not only as data storage tools in DB, but also as data transfer tools to remote servers. A flat file can be a plain text file or a binary file. There are usually no structural relationships between the records. The Most Commonly used kind of OLTP in DWH is Flat file (The second highest percentage usage). So better handling of the flat files in DWH will result best processing of data.[6]

III. PROPOSED APPROACH

Storing huge number of small files results in high memory usage and unacceptable access cost. The Basic idea is to merge small files into larger ones to reduce file number, and to build index or use key-value pair for each original file. A File merging method among files belonging to is proposed to improve the storing efficiency of small files, and a local index file is established for each merged file [7]. A two- level pre-fetching mechanism, which comprises local index file pre-fetching and correlated file pre-fetching, is utilized to improve the accessing efficiency of the small files. It mainly consists of mapping the required file to a merged file, reading a local index file, splitting a target block and pre-fetching. We can expect better performance by combining small files into larger ones and building hash index for each small file. We can create multiple threads to read the flat file. We can use Hash Partitioning. We can also use Key Range Partitioning. Loading the flat file into a temp table. And then we can create indexes on the required columns.[8]. The merging mechanism of Flat files can be handled by the following convention.

	Merging Intermediate Files	Merging Directly
Reading in file1	2.86	----
Reading in file2	3.25	----
Reading in file3	2.99	----
Merging	4.18	6.16
Total Seconds:	**13.28**	**6.16**

Note: timings were taken from SAS version 8 running on a PC.

Figure 2: Merging Task Comparison.

So the final result shows that merging of flat files or any other source directly will require 6.16 seconds where as in case of Merging with intermediate files requires 13.28. Creating and merging intermediate SAS files vs. Merging flat files directly File Name Type Position Length Driver CUSTNUM Numeric 1 1

ADDRESS Char 3 6

Support1 KEY1 Numeric 1 1

INCOME Numeric 3 3

Support2 KEY2 Numeric 1 1

SALES Numeric 3 3

The actual data for this example is:

Driver Support1 Support2

Obs#(CUSTNUM,ADDRESS)
(KEY1,INCOME) (KEY2,SALES)

1 1, MAIN 2,100 1, 20

2 2, FIRST 3,60 3,40

3 5, SECOND 4,140 4,100

4 --- 5,240 ---

Example Code

The following is the actual code that is needed to merge these files.

** Merging flat files: SUGI presentation

1 data final(keep=CUSTNUM ADDRESS INCOME SALES);
2 retain DONE1 DONE2 0 KEY1 KEY2 ADVANCE;

/* input driver file information */

3 infile 'driver.txt';

```
 4 input @1 CUSTNUM 1.

 5 @3 ADDRESS $6.;

/* input each supporting file information */

/* income file first */

 6 if not DONE1 then do;

 7 ADVANCE=0;

 8 infile 'income.txt' end=LAST1;

 9 do until (KEY1>=CUSTNUM or LAST1);

10 if ADVANCE then input;

11 input @1 KEY1 1. @@;

12 ADVANCE=1;

13 end;

14 if KEY1=CUSTNUM then

15 input @3 INCOME 3.;

16 if LAST1 then if KEY1<=CUSTNUM then DONE1=1;

17 end;

/* sales file next */

18 if not DONE2 then do;

SUGI 27 Coders' Corner2

19 ADVANCE=0;

20 infile 'sales.txt' end=LAST2;

21 do until (KEY2>=CUSTNUM or LAST2);

 22 if ADVANCE then input;
23 input @1 KEY2 1. @@;

24 ADVANCE=1;

25 end;

26 if KEY2=CUSTNUM then

27 input @3 SALES 3.;

28 if LAST2 then if KEY2<=CUSTNUM then DONE2=1;

29 end;

30 run;
```

Another requirement to speed up processing of the Flat files is process of Indexing.

No. records	No . of condition	Experiment1		Experiment 2		Experiment3	
		Bit map	B-tree	Bit map	B-tree	Bit map	B-tree
1000,000	Three condition	13.15	20.67	15.89	21.81	10.60	17.88
	Two condition	5.63	12.24	4.43	10.52	4.87	8.54
	One condition	3.58	9.94	3.68	8.06	3.57	5.17
800,000	Three condition	10.86	21.17	5.94	11.04	9.28	11.26
	Two condition	5.02	13.68	4.87	8.13	4.48	7.97
	One condition	3.81	8.88	3.37	7.98	3.22	5.01
600,000	Three condition	7.72	13.28	7.55	12.24	5.04	11.24
	Two condition	3.15	6.32	3.32	7.33	2.80	8.93
	One condition	3.34	6.02	3.74	6.93	3.74	6.39
300,000	Three condition	4.74	7.02	3.12	5.51	3.47	7.00
	Two condition	3.38	6.33	2.68	4.56	2.19	5.18
	One condition	1.48	7.79	2.57	5.76	1.95	5.34

Figure 3: Time required for Each Indexing.

To observe the efficiency of each indexing the following diagram will shows the experimental results

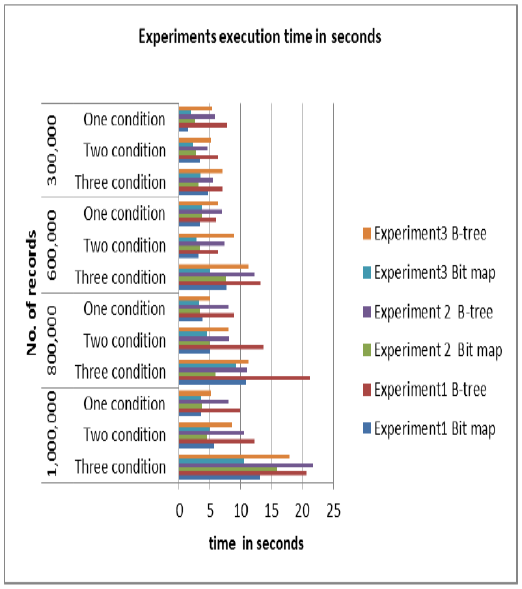

Figure 4: Bitmap and B-Tree Indexing Time Requirements.

As shown in the figure 4 the experimental results show efficient process in reading big amount of data in real time application with relative data efficiency in accessing services, that includes sufficient process generation in construction of ETL and handling those data sets with their requirement specification in real time application process. In this application we will include technical aspects of relevant task management operations in real time data handling applications for progressive environment specifications. These results are accessed modularity with event management operations. A File merging method among files belonging to is proposed to improve the storing efficiency of small files, and a local index file is established for each merged file [7]. A two- level pre-fetching mechanism, which comprises local index file pre-fetching and correlated file pre-fetching, is utilized to improve the accessing efficiency of the small files. It mainly consists of mapping the required file to a merged file, reading a local index file, splitting a target block and pre-fetching. We can expect better performance by combining small files into larger ones and building hash index for each small file. Process of developing these applications is as follows:

Input: Datasets related to technical issues of data representation

Output: Skyline Computational results of each data set.

Step1: Import datasets from i=1……..n.

Step 2: Data aggregative operations in each data set using Hash function generation process.

Step 3: Perform Early pruning with noise data

Step 4: Perform Late Pruning with noise data.

Step 5: Updated results can be stored in repository using ETL techniques.

Step 5: Experimental result are stored in semantic data representation.

Algorithm 1: Skyline computation results using ETL operations.

By using above algorithm we perform and calculate efficient processing applications with including processing of data with sufficient experimental results for analyzing and modifying data restrictions with procedure oriented processing real time data applications.

IV. CONCLUSION

The paper describes the usage of FLAT file source so as to speed up the ETL process. The benefit of the methodology that we are going to provide will improves the efficiency of data ware housing environment in case of storage and processing time. The IT companies are looking for better performance for the data warehousing processing as the DWH environment holds bulk amount of the data, we concentrate on FLAT file source in ETL processing because it is the second highest source used by the DWH environment. In this application we will include technical aspects of relevant task management operations in real time data handling applications for progressive environment specifications. These results are accessed modularity with event management operations.

V. REFERENCES

[1].Satkaur , Research scholar, S.K.I.E.T. ,Kurukshetra, Haryana, International Journal of Advanced Research in computer Science and Software Engineering , Volume 3, Issue 5, May 2013 ISSN:
2277 128X .

[2].JensDittrichJorgeArnulfo,Quian´eRuizInformation Systems Group Saarland University, Efficient Big Data Processing in Hadoop Map Reduce, Proceedings of the VLDB Endowment, Vol. 5, No. 12Copyright 2012 VLDB.

[3].Lizhe Wang, School of Computer, China University of Geosciences, G-Hadoop: Map Reduce across distributed data centers for data-intensive computing, Future Generation Computer Systems, The international journal of grid computing and esciences 2012 Elsevier.

[4]Bo Dong, Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China, A Novel Approach to Improving the Efficiency of Storing and Accessing Small Files on Hadoop: a Case Study by PowerPoint Files, 2010 IEEE International Conference on Services Computing.

[5].By Muhammad Inayat Ullah, Gomal University, Transformation of Flat File into Data Warehouse, Global Journal of Computer Science and Technology Volume 11 Issue 13 Version 1.0 August 2011.

[6].Improved Extraction mechanism in ETL process for building of a Data Warehouse, MPSTME, SVKM's NMIMS, Mumbai .

[7].Ranjit Singh, Research Scholar, University College of Engineering (UCoE), Punjabi University, A Descriptive Classification of Causes of Data Quality Problems in Data Warehousing, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 2, May 2010 41ISSN (Online): 1694- 0784.

[8].Bitmap Index as Effective Indexing for Low Cardinality Column in Data Warehouse -International Journal of Computer Applications (0975 – 8887) Volume 68– No.24, April 2013.