Modeling a Data Storage System (DSS) for Seamless Real-Time Information Support from Information Manufacturing System (IMS)

Mohammad Shamsul Islam
Faculty of Engineering & Computing
Dublin City University (DCU)
Dublin, Ireland
Email: mohammad.islam6@dcu.ie

Paul Young
Faculty of Engineering & Computing
Dublin City University (DCU)
Dublin, Ireland
Email: paul.young@dcu.ie

Abstract—**Nowadays, a large number of enterprises operate in a business time schedule of 24×7. These enterprises need to deliver information as fast as possible for information support. Therefore, the information manufacturing system of the enterprises should have the ability for seamless real-time information support. Data storage system in the information manufacturing system plays the role of providing non interrupted real-time information support. Therefore, 2-Data Storage System oriented information manufacturing system is developed for providing real-time information support. This 2-Data Storage System oriented information manufacturing system can provide real-time information support for a short period of time. However, it is not possible to provide seamless real-time information support by this information manufacturing system. Hence, modeling a data storage system for seamless real-time information support from the information manufacturing system is the purpose of this paper.**

Keywords-data loading; indexing; query processing.

I. INTRODUCTION

An IMS (Information Manufacturing System) is an information system that manufactures information from the raw data [23]. The most important component of the IMS is a DSS (Data Storage System). The DSS integrates multiple sources of the system and so contains raw data from multiple sources. Data come from multiple sources are processed by the refreshment function of the availability of data in the DSS. Available data in the DSS are then delivered as information by the execution of query function.

Traditionally, IMS works in the non real-time environment. Single or cluster (replication) DSS oriented IMS is used for providing information support for this non real-time environment. The DSS is updated periodically, typically in a daily, weekly or even monthly basis in the non real-time environment [24]. The DSS needs to update continuously for providing real-time information support with most recent data. Update is done with the refreshment function in the DSS. Continuous execution of the refreshment function (single DSS) and non simultaneous update (cluster DSS) can cause of the poor quality information support from the IMS [6]. More specifically, the poor quality information support occurs for not executing the refreshment and query function simultaneously (single DSS) or the propagation delay for updating the DSS (cluster DSS) of IMS. Therefore, these DSS oriented IMS are not suitable for real-time information support.

Enterprises such as stock brokering, e-business, online telecommunication, health system and traffic systems need to deliver information as fast as possible to knowledge workers or decision-makers who make a decision in a real-time or near real-time environment, according to the new and most recent data captured by an organization's IMS [12]. Therefore, Santos and Berardino [18] as well as Hanson and Willshire [10] developed a 2-DSS oriented IMS for providing real-time information support. However, this 2-DSS oriented IMS cannot provide real-time information support seamlessly. Nowadays, some enterprises need to operate in a business time schedule of 24 × 7 for providing information support in real-time environment. Therefore, the purpose of this research is for modeling a data storage system in the IMS that can provide non-interrupted real-time information support for the business time schedule of 24 × 7. The modeled data storage system is 3-DSS for serving the purpose.

The remaining part of this paper is organized as follows: Section 2 presents the related research of the data storage system. Section 3 describes the 3-DSS. Section 4 shows the regulating procedure of the tasks of the refreshment and query function in the 3-DSS. Section 5 presents the management of the system at the down period of principal 3-DSS and the execution of tasks for restarting the principal 3-DSS again in the system. Experimental evaluation as well as conclusion and future work are shown in Section 6 and Section 7 respectively.

II. RELATED RESEARCH

So far, some researches have been done over DSS (DW, distributed DW, etc.). Bouzeghoub et al. [1] and Vavouras et al. [21] presents the modeling of the data warehouse refreshment process. They explain the difference between loading and refreshment process to these papers. Analyzing the information manufacturing system of many organizations, Mannino and Walter [13] identify that timeliness and availability of data in the DSS are responsible for bad quality information in the IMS. They also find that the refresh period of a system influences the timeliness and availability of data. Theodoratus and Bouzeghoub [20] discuss the data currency quality factors in data warehouses and propose a DW design that considers these factors. An important issue for near real-time data integration is the accommodation of delays, which has been investigated for (business) transactions in temporal active databases in [17]. Vrbsky [22] developed a model to get approximate information from the IMS within a certain time in real-time environment. McCarthy and Risch [16] provide the data structure for execution of real-time queries. Capiello et al. [6] shows that multiple scattered DSS has the lowest degree of integration. It is evident that there is a data quality problem as a result of both long refresh period and propagation delay. On the other hand, single DSS in the IMS has the highest degree of integration, therefore, it does not make the data quality problem. Santos and Berardino [18] present a table structure replication technique to ensure fresh decision support for the real-time or frequently changing data. The table structure replications have two tables, the permanent table and the temporary table. The data stored in a temporary table are transferred to a permanent table for the deterioration of the query response . At the time of transfer, no access is possible in the table of data storage for the information support. Hanson and Willshire [10] developed a faster data warehouse model providing an auxiliary structure for quick query response. This is also a 2-DSS oriented IMS. It has a temporary table as well where only data will be loaded and no administrative overhead such as indexing will be done. In this model storage capacity of temporary tables is limited as it is installed in the non-volatile NVRAM. After fulfilling the 95% of the temporary tables, data is transferred to the permanent table. Therefore, no data access will be possible in this period. As a result, 24 × 7 services will be not possible with these 2-DSS oriented IMS seamlessly.

The data storage model of this research does not need to transfer data by pushing the system to offline or by stopping the system. Therefore, it will be possible to provide 24 × 7 services seamlessly with this DSS model. Further, it will update the data in the DSS in real-time manner for providing the real-time information support.

III. 3-DATA STORAGE SYSTEM (3-DSS)

According to [1][13][18], refreshment and query function execute in a data storage system of the IMS to make the data available and for the information support respectively.

Refreshment Function: This is a complex process comprising the tasks, such as data loading, indexing and propagation of data for synchronizing data in the information manufacturing system (IMS) [1][13][18].

Data loading: Key activities of data loading include extraction, transformation, integration, cleaning etc. Therefore, storage of manipulating [insert, update] data are to extract from the sources , then, transformed data if the source data are in the different format. After that, extracted and transformed data are to integrate and to clean for loading data in the data storage system [1][13][18].

Indexing: Update the index for newly loaded data or delete data to align the data in the data storage system [18]. Indexing determines the effective usability of data collected and aggregated from the sources and increases the performance of the data storage system for information support [1][13].

Propagation of Data: Data is propagated through the refreshment process for synchronizing the data of multiple DSSs of the system.

Query Function: This function of data storage system in IMS is done by the query processing task. A requested query of the user is processed in the data storage system for delivering the information to the user.

In the 3-DSS, three individual DSS are mutually interconnected with each other. The tasks of the refreshment and the query function of data storage system work simultaneously in three individual DSS . Therefore, data loading and indexing with updated data propagation task of the refreshment function work in two individual DSS of 3-DSS and another DSS of 3-DSS executes the task of the query function at the same time. After each successive period, the tasks of the refreshment and the query function of data storage system will interchange with cyclic order. As, propagation of manipulated data in the DSS is done simultaneously at the working period of the task of the functionalities, there may not have propagation delay. Therefore, 3-DSS will hold exact the same data. The 3-DSS is shown in Figure 1.

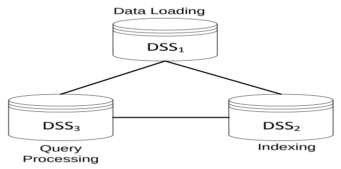


Figure 1. 3-DSS

In the following sub-sections, the execution process of 3-DSS, protocol of 3-DSS and the partitioning procedure of 3-DSS will be discussed.

A. *Execution Process of 3-DSS*

Suppose, DB1, DB2 and DB3 is three individual DSS for 3-DSS. These three DSS are mutually interconnected with each other. Now, if DB1 store some data from operational data sources, DB2 and DB3 must have the same data. The tasks of the query and refreshment function work simultaneously in these three data storage systems. There must have a synchronization of starting and finishing time of the tasks of the query and refreshment function of these three data storage systems. Manipulated data from operational data sources will load into one database. At the same time, another database will do the indexing and updated data propagation task for synchronizing data with other two DSS and the third one will be used for query processing. The indexed and query processing database lead the process. When the indexing with the propagation of updated data and the query processing are finished, the tasks of the refreshment and query function of data storage system will interchange with cyclic order. The rotation algorithm for the interchanging process is given in the Figure 2.

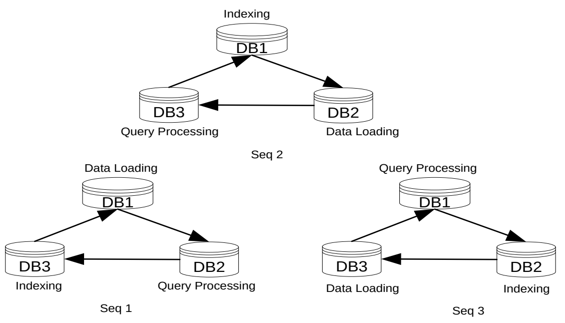


Figure 2. Rotation of the tasks of functionalities in 3-DSS

The algorithm for the rotation of function (Data Loading, Indexing and Query Processing) in data storage system is shown in Figure 3.

Rotation of Tasks of Functionalities in 3-DSS (Data Loading, Indexing with update propagation, Query Processing)

```
DB1 = Database 1, DB2 = Database 2, DB3 = Database 3

Step   1: DB1=>Data Loading, DB2=>Nothing, DB3=>Nothing

Step 2: DB1=>Indexing and send updated data to DB2, DB2=>Data Loading, DB3=>Nothing

Step 3: DB2=>Indexing and send updated data to DB3, DB3=>Data Loading, DB1=>Query Processing

Step 4: DB3=>Indexing and send updated data to DB1, DB1=>Data Loading, DB2=>Query Processing

Step 5: DB1=>Indexing and send updated data to DB2, DB2=>Data Loading, DB3=>Query Processing

Step 6: Repeat Step 3, 4 and 5.
```

Figure 3. Algorithm for Rotation of Tasks of Functionalities in 3-DSS

In the algorithm, Steps 1 and 2 indicate the initialization of the system. Data come from multiple sources are loaded into DB1 in Step 1. DB1 executes the indexing task and send the updated data to DB2 and DB2 loads the updated data and source data simultaneously in Step 2. Each task of the functionalities of data storage system works simultaneously in Steps 3, 4 and 5. As, data have been loaded into DB2 in Step 2, DB2 is indexed in Step 3 and send the updated data to DB3. At the same time, DB3 loads the manipulated data including the updated data of DB2. Further, DB1 provides the information by processing the query request of the system in Step 3. Steps 4 and 5 will follow the same process but interchange the roles of each DB of the system. Therefore, Steps 3, 4 and 5 will continue repeatedly in the system.

B. *3-DSS Protocol*

A protocol is a set of rules for regulating a system [9]. 3-DSS is the data storage system where multiple tasks will execute simultaneously. Therefore, 3-DSS are to maintain a set of rule for avoiding the cumbersome operations of the 3-DSS in the IMS.

1. N, 2N, 3N,...............NN amount of manipulating new data will be loaded each time to be available in the DSS after completion of the task of the refreshment function.
2. Three individual DSS of the 3-DSS will be located contiguously in the same place.
3. Functionalities do not interchange if the indexing task is in progress. It means that functionalities do not interchange before the completion of indexing task.
4. Functionalities do not interchange if the propagation of update data from one DSS to another DSS is in progress. Otherwise, the propagated data receiver DSS can not load the all new data of the sender DSS.
5. Functionalities do not interchange if query processing task is in progress. Otherwise, the query can process an incomplete query result.
6. Throughput of the three interconnected network link of 3-DSS should be same.
7. Rotation of the tasks of functionalities will be clockwise cyclic order like the Figure 2.
8. Previous DSS of indexing DSS will always process the query and next DSS of indexing DSS always load the manipulated data. It is seen in Figure 2 that in every sequence (seq) previous DSS of indexing DSS process the query.

9. Previous DSS of query processing DSS will always load the manipulated data and next DSS of query processing will always indexed the loaded data of the DSS. It is seen in Figure 2 that in every sequence (seq) previous DSS of query processing DSS load the manipulated data.
10. Previous DSS of data loading will always indexed the loaded data of the DSS and next DSS of data loading DSS will always process the query. It is seen in Figure 2 that in every sequence (seq) previous DSS of data loading DSS indexed the loaded data.
11. Interchange of the tasks of functionalities will be done after the completion of the indexing and query processing tasks.

C. 3-DSS Partitioning

Partitioning is the technique of fragmenting large relations (tables) into smaller ones. In the large DSS, (tables), if manipulation of data (insertion, update, delete) is done, it needs more time to rebuild the indices of the large DSS (tables). As a result, a requested query may not execute in time or may provide a poor query result. The partitioning of a large table can resolve this problem. In a partitioning DSS (tables), the problem that created at the time of index rebuilding for the manipulation of data is limited only in a particular partition. Therefore, except the partitions where data is being manipulated, other partitions of the DSS (tables) can provide the query result for the requested query as the index rebuilding process is limited to the certain partition. Additionally, it needs less time to rebuild the index than the non-partitioned DSS (tables) as the volume of data of each partition will be certain. There are two ways to partition a DSS: vertically and horizontally. Vertical partitioning involves splitting the attributes (columns) of a DSS (tables), placing them into two or more DSS (tables) linked by the DSS (tables) primary key. Horizontal partitioning involves splitting the tuples of a DSS (table), placing them into single or more DSS (tables) with the same structure. For keeping the certain volume of data in the DSS (table), horizontal partitioning will be used in the 3-DSS. There are two types of horizontal partitioning: primary and derived. Primary horizontal partition (HP) of a DSS (table) is performed using attributes defined on that DSS (table). On the other hand, derived horizontal partition is the fragmentation of a DSS (table) using the attributes defined on another DSS (tables) [4]. Horizontal partitioning for 3-DSS is discussed in below,

Let, F is the primary or fact table, D is the derived or dimensional table. Example of a primary (fact) relational table and derived (dimensional) table are depicted in Figure 4.

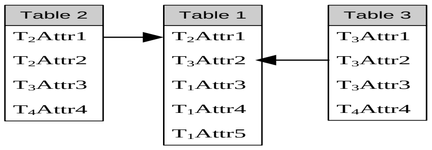
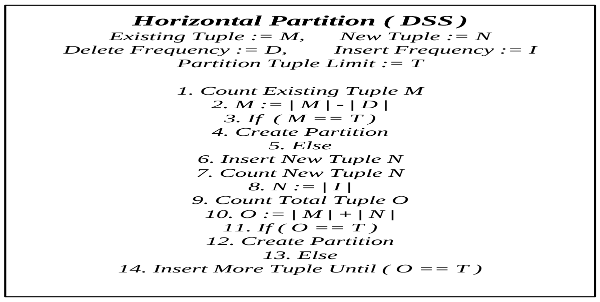


Figure 4. Primary (fact) relational table and derived (dimensional) table

In Figure 4, Table 2 and Table 3 is primary table. On the other hand, Table 1 is derived table. Fragmentation of Table 1 depends on Table 2 and Table 3. If primary Table 2 and Table 3 are manipulated, Table 1 must have to be manipulated. Therefore, tuple of Table 1, Table 2 and Table 3 will be horizontally partitioned simultaneously considering the instruction of the predicate.

A predicate is the Boolean expression over the attributes of a relational table and constants of the attribute's domains. Horizontal partitioning can be defined as a pair (T, Φ), where T is a relation and Φ is a predicate. This predicate partitions T into at most 2 fragments with the same set of attributes. The first fragment includes all tuples of t of T which satisfy Φ, i.e., t = Φ. The second fragment includes all tuples t of T which does not satisfy Φ, i.e., t ≠ Φ. It is possibly one of the fragments to be empty if all tuples of T either satisfy or do not satisfy Φ.

Let Φ = (counted tuple = N), which results into fragment horizontally where tuple of a relational table will be counted. If the condition of the predicate Φ is true, relational table will be fragmented into 2 partitions. The first partition will hold N numbers of the tuple. If manipulation of data in the information manufacturing system (IMS) is stopped after being partitioned, the second partition will remain empty. When manipulation of data in the IMS is continued and the total insertion of tuple reaches to N number, this partition will again fragment into two pieces. This partitioning process will continue as long as the information manufacturing system is not stopped. This single table partitioning process can be applied to the multiple relation tables of the DSS in the IMS. The horizontal Partitioning algorithm and the partitioning algorithm of the 3-DSS are given in Figure 5 and Figure 6, respectively.


```
Horizontal Partition ( DSS )
Existing Tuple := M,      New Tuple := N
Delete Frequency := D,      Insert Frequency := I
Partition Tuple Limit := T

1. Count Existing Tuple M
2. M := | M | - | D |
3. If ( M == T )
4. Create Partition
5. Else
6. Insert New Tuple N
7. Count New Tuple N
8. N := | I |
9. Count Total Tuple O
10. O := | M | + | N |
11. If ( O == T )
12. Create Partition
13. Else
14. Insert More Tuple Until ( O == T )
```

Figure 5. Horizontal Partitioning algorithm for 3-DSS

```
Partitioning ($DSS_1$, $DSS_2$, $DSS_3$)
1. Locate the DSS that execute L ( )
2. If ( DSS == $DSS_1$ )
3. Execute Horizontal Partition ( DSS) for $DSS_1$ in Loading Period L(t)
4. Else If ( DSS == $DSS_2$ )
5. Execute Horizontal Partition ( DSS) for $DSS_2$ in Loading Period L(t)
6. Else
7. Execute Horizontal Partition ( DSS) for $DSS_3$ in Loading Period L(t)
8. Go On Step1 and Repeat
```

Figure 6. Partitioning algorithm for 3-DSS

Partitioning of DSS of the 3-DSS will be done by following the partitioning algorithm given in Figure 6. According to this algorithm, DSS will be located for the partitioning in the executing period of data loading of a particular DSS. Then, the horizontal partitioning algorithm will be applied for partitioning the DSS of the 3-DSS. The horizontal partitioning algorithm is shown in Figure 5. In this algorithm, Existing and new tuples have to calculate for the horizontal partitioning. Deleting of a tuple from DSS will detect the tuple from the existing tuple. On the other hand, insertion of the tuple will count as a new tuple. A DSS will be partitioned if total tuples of the DSS is reached in the partition limit N. Therefore, the existing tuple will be counted by deducting the deleted tuple from the DSS. Hence, the number of existing tuples will be counted for checking whether the existing tuple is in partition limit or not. If, it is in partition limit, the partition will be created. Otherwise, new tuple will be inserted in the DSS. Therefore, new tuple will be counted by the number of new insertions. After that, total tuple will be counted by adding existing tuple with new tuple. Henceforth, it will be checked for whether a counted number of total tuples are in partition limit or not. If total tuple equals to the partition limit, the partition will be created. Otherwise, more tuple has to be inserted until the total tuple reaches to the range of the partition limit of the tuple.

IV. Regulating Procedure of the Tasks of Refreshment & Query Function of 3-DSS

3-DSS executes three individual DSS simultaneously in the information manufacturing system (IMS). The tasks of the functionalities of data storage system work in these three individual DSS of the 3-DSS. These tasks also need to interchange in the 3-DSS. Further, this 3-DSS is to give an assurance of quick update of data for the real-time information support. As a result, DSS of the 3-DSS is fragmented with a partitioning procedure. Therefore, there must have a coordination and communication among the tasks of the functionalities of the 3-DSS for doing the simultaneous operation, interchanging of the tasks of the functionalities and the partitioning of the DSS. The regulator algorithm will play the role for making the coordination and communication among the tasks of the functionalities with the help of some other algorithms. The regulator algorithm is given in Figure 7.

```
Regulator ($DSS_1$, $DSS_2$, $DSS_3$)
(Loading), (Indexing and propagation) and (query processing) is represented as L, Ix and IS respectively
3-DSS is controlled by a controller thread represented as CT

Step 1: Create Thread 1, Thread 2, Thread 3 and Controller Thread as $T_1$, $T_2$, $T_3$ and CT
Step 2: $T_1 \rightarrow$ Ix ( ),  $T_2 \rightarrow$ IS ( ),  $T_3 \rightarrow$ L ( )
Step 3: $T_1 \rightarrow$ L ( ),  $T_2 \rightarrow$ Ix ( ),  $T_3 \rightarrow$ IS ( )
Step 4: $T_1 \rightarrow$ IS ( ), $T_2 \rightarrow$ L ( ),  $T_3 \rightarrow$ Ix ( )
Step 5: CT $\rightarrow$
        I.  Execute Synchronizing Agent Algorithm
        II. Execute Partitioning Algorithm
Step 6: Go on step 2 and repeat from step 2
```

Figure 7. Regulator algorithm for 3-DSS

In the regulator algorithm of the 3-DSS, four threads will be created for the execution of the operation of 3-DSS. Thread 1, thread 2 and thread 3 are created for the simultaneous operation of the tasks of the functionalities of the 3-DSS in three individual DSS. On the other side, controller thread CT is constructed for the execution of the synchronizing agent and the partitioning algorithms. Synchronizing agent and partitioning algorithms are shown in Figure 8 and Figure 6 respectively. As thread 1, thread 2 and thread 3 work simultaneously, so, when thread 1 executes indexing function, thread 2 and thread 3 will execute the query and loading function, respectively. This will continue until thread 1, thread 2 and thread 3 get a message from the synchronizing agent of controller thread to change their tasks of the functionalities. If thread 1, thread 2 and thread 3 get a message from the synchronizing agent of controller thread to change their activities, then, thread 1 executes the function for loading task, thread 2 and thread 3 execute function for indexing and query processing tasks respectively. These activities of threading will continue until these threads do not get any message to interchange their activities. As soon as, each thread gets the message to interchange their activities, thread 1 will start query processing, thread 2 will start loading of data and thread 3 will start the indexing task. Hence, the sequence of activities of among threads will continue until the system is stopped by the user or any other reasons. For the shortening of indexing period, DSS will be partitioned by partitioning algorithm after a certain number of storage data. This partitioning process will provide the service from the controller thread together with synchronizing agent. Details of the synchronizing agent algorithm are given in Figure 8.

```
Synchronizing Agent ($DSS_1$, $DSS_2$, $DSS_3$)
1. Ix ( ), L ( ) and IS ( ) is executing simultaneously in $T_1$, $T_2$ and $T_3$ by rotation.
2. Ix ( ) And IS ( ) inform CT of its completion status
3. L ( ) sends a message to CT to get the status information of Ix ( ) And IS ( )
4. Waiting for the reply of CT about the status of Ix ( ) And IS ( )
5. IS ( ) sends a message to CT after its completion to get the status information of Ix ( )
6. Waiting for the reply of CT about the status of Ix ( )
7. Ix ( ) sends a message to CT after its completion to get the status information of IS ( )
8. Waiting for the reply of CT about the status of IS ( )
9. Completion of the status of Ix ( ) And IS ( ) = Boolean Value
10. If Boolean Value of Ix ( ) = False AND Boolean Value of IS ( ) = False
    10.1    Continue Current Functions in $T_1$, $T_2$ and $T_3$
11. Else If Boolean Value of Ix ( ) = True AND Boolean Value of IS ( ) = False
    11.1    Continue Current Functions in $T_1$, $T_2$ and $T_3$
12. Else If Boolean Value of Ix ( ) = False AND Boolean Value of IS ( ) = True
    12.1    Continue Current Functions in $T_1$, $T_2$ and $T_3$
13. Else
    13.1    $T_1$ : Move to Next Function and Execute
    13.2    $T_2$ : Move to Next Function and Execute
    13.3    $T_3$ : Move to Next Function and Execute
14. Set
    14.1    Next Function $\longrightarrow$ Current Function in $T_1$
    14.2    Next Function $\longrightarrow$ Current Function in $T_2$
    14.3    Next Function $\longrightarrow$ Current Function in $T_3$
```

Figure 8. Synchronizing Agent algorithm for 3-DSS

Figure 8 presents the algorithm for the interchange process among of the tasks of the functionalities of the 3-DSS. It shows, how the tasks of the functionalities of the 3-DSS communicate with each other for providing their service rotationally in three individual DSS of the 3-DSS. According to the regulator algorithm of the 3-DSS, each task of the functionalities of the 3-DSS (indexing, loading and query processing) executes simultaneously in three separate threads by rotation. Further, each individual DSS of the 3-DSS changes role after a certain period of time. Query processing and indexing tasks are not possible to stop in the middle of the execution or before the completion of these tasks. Therefore, the indexing and the query processing tasks can be called dependent tasks. On the other hand, it is possible to stop the loading of data at any moment of time. Therefore, this task could be called independent task. Hence, the interchanging process of the tasks of the functionalities in the 3-DSS depends on both the indexing and the query processing tasks. In line 1 of algorithm indicates that step 1, step 2 and step 3 of regulator algorithm will be executed simultaneously by rotation. In line 2, function of the indexing and the query processing tasks will inform their current status to the controller thread. Then, the function of the loading task will send the message to the controller thread to know the current status of the indexing and the query processing function in line 3. The controller thread will deliver a reply about the status of the indexing and the query processing in line 4. In line 5 and 6, the query processing function will send a message to the controller thread to know the status of the indexing function after the completion its task and wait for the reply. Similarly, in line 7 and 8, indexing function will do the same and wait for the reply about the status of the query function. Now, from line 10 to line 13 shows that whether the tasks of the functionalities of the 3-DSS will be interchanged or not. If the completion status of the query processing or the indexing function is false, the tasks of the functionalities of the 3-DSS will not be interchanged. So, from line 10 to line 12, current function is continued in thread 1, thread 2 and thread 3. In line 13, the completion status of both the query processing and the indexing function is true, so, the tasks of the functionalities of the 3-DSS is interchanged. For this reason, current function of each thread is stopped and move to the next function. Current function move to the next function in the regulator algorithm mean that indexing, loading and query processing function execute in step 1, step 2 and step 3 respectively in thread 1; query processing, indexing and loading function execute in step 1, step 2 and step 3 respectively in thread 2 and loading, query processing and indexing function execute in step 1, step 2 and step 3 respectively in thread 3. Now, if the current function of step 1 of thread 1, thread 2 and thread 3 are indexing, query processing and loading function respectively, next function will be the function of thread 1, thread 2 and thread 3 of step 2 and so on for step 2 and step 3. Therefore, when the next function will be prepared for execution, it will be executed as current function. Finally, in line 14, next function is set and executed as the current function in thread 1, thread 2 and thread 3. The whole process executes repeatedly to continue the interchanging of the tasks of the functionalities in three individual DSS simultaneously in the 3-DSS.

V. ADDITIONAL TASKS FOR HANDLING THE SYSTEM FOR THE FAILURE OF PRINCIPAL 3-DSS

Two 3-DSS can be installed in IMS for the real-time information support seamlessly. One 3-DSS can be said principal 3-DSS. Another can be told alternative 3-DSS. The principal 3-DSS can be down at any moment of time in the system for the crashing or other difficulties for any of the DSS of the principal 3-DSS. An alternative 3-DSS can facilitate the seamless real-time time information support at the down period of the principal 3-DSS. Therefore, some of the additional tasks have to include in the regulator algorithm of Figure 7 for handling the system for the failure of the principal 3-DSS. These tasks will be executed in the controller thread. The additional tasks that will be included in the controller thread are,

Sending the Source Data to Alternative 3-DSS: The source data will be stored in the alternative 3-DSS at the same time of loading data in the principal 3-DSS. It is done by sending source data to one DSS of the alternative 3-DSS. This DSS then replicates the data to other two DSS of the alternative 3-DSS.

Recovery System: The log based or the shadow paging recovery system described in [19] can be used for recovering the data for crashing of 3-DSS.

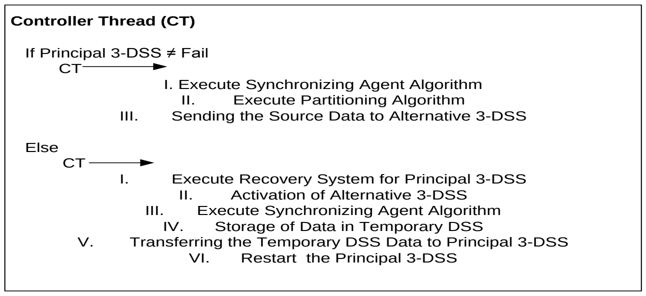
Activation of Alternative 3-DSS: Alternative 3-DSS will be activated for information support just after the failure of principal 3-DSS. This alternative 3-DSS will then work just like the principal 3-DSS.

Storage of Data in Temporary DSS: The source data will also be stored in the temporary DSS for supporting the principal 3-DSS. It will store data as long as principal 3-DSS will be down.

Transferring the Temporary DSS Data to Principal 3-DSS: After fixing the problem of the principal 3-DSS, the stored data in the temporary DSS will now be transferred to the principal 3-DSS.

Restart the Principal 3-DSS: Now, the principal 3-DSS will be restarted and the activities of principal 3-DSS will be released from the alternative 3-DSS. This alternative 3-DSS will then perform its general task.

Now, the controller thread of the regulator algorithm in Figure 7 can be written as:



```
Controller Thread (CT)

If Principal 3-DSS ≠ Fail
    CT ──────►
            I. Execute Synchronizing Agent Algorithm
            II. Execute Partitioning Algorithm
            III. Sending the Source Data to Alternative 3-DSS

Else
    CT ──────►
            I. Execute Recovery System for Principal 3-DSS
            II. Activation of Alternative 3-DSS
            III. Execute Synchronizing Agent Algorithm
            IV. Storage of Data in Temporary DSS
            V. Transferring the Temporary DSS Data to Principal 3-DSS
            VI. Restart the Principal 3-DSS
```

Figure 9. Additional tasks in controller thread for handling both principal and alternative 3-DSS

In Figure 9, if principal 3-DSS is not in failing state, then, the controller thread will execute the general 3-DSS tasks and Sending the source data to alternative 3-DSS task. On the other hand, if the principal DSS is in failing state, then, the controller thread will execute recovery system for principal 3-DSS. Then, it will activate the alternative 3-DSS. At the same time, synchronizing agent algorithm will start to work on the alternative 3-DSS. Storage of data in temporary DSS, transferring the temporary DSS data to principal 3-DSS and restarting the principal 3-DSS tasks will also execute in the controller thread. Once the principal 3-DSS restarts, the controller thread will again execute the tasks of not failing state.

VI. EXPERIMENTAL EVALUATION

The experiments have been done for the 2-DSS and 3-DSS oriented IMS. 2-DSS is used as the benchmark for showing the real-time information support of 3-DSS. Temporary DSS is considered for the experiment of 2-DSS. Only loading of data task work in the temporary DSS of the 2-DSS for providing the real-time information support . The tasks of the DSS functionalities work in three individual DSS simultaneously in the 3-DSS. Four machines were used for doing the experiments. Among the four machines, three machines were used for implementing the 3-DSS, another is used as a server for controlling the 3-DSS, inserting data from the sources to the 3-DSS and sending the query request to the 3-DSS. Server machine and one more machine were used for the experiment of the 2-DSS. Multi core 2.2 Ghz processors, 4GB RAM and 5400 r.p.m hard drive were used for the server machine. The rest of the machine used single core 1.69 Ghz processor, 1GB RAM and 5400 r.p.m hard drive. SQL server was the database software for creating the data storage system.

For doing three experiments, 1GB data was stored in both the 2-DSS and the 3-DSS oriented IMS. Fifty thousand new rows (tuple) were extracted from the sources and stored in the 2-DSS and the 3-DSS with the refreshment function at the time of each individual experiment for delivering the data for the query request. One experiment of 3-DSS was done without storing 1GB data in the 3-DSS. In the real world, user may send the query request in the refreshment period. For this reason, query and refreshment functions executed simultaneously in these experiments. Query request for retrieving all newly inserted data was sent repeatedly after each single minute. Therefore, the query result was delivered for each respective query request. These query results were measured to get the result for both the 2-DSS and the 3-DSS oriented IMS. Start of data insertion time means the first data insertion time from the source to the DSS and query delivery time is the time the query result is delivered. Therefore, the distance between start of data insertion time and query delivery time is measured by subtracting query delivery time from the start of data insertion time. Volume of query result data is measured by dividing the number of inserted data in the DSS with the total data for insertion. The results are given in Table I, Table II, Table III, and Table IV.

TABLE I. EXPERIMENTAL RESULT OF 2-DSS (BENCHMARK DSS)

Query Request No.	Distance between Start of Data Insertion Time and Query Delivery Time (Minute)	Volume of Query Result Data (%)
1	1	11.00
2	2	20.80
3	3	31.20
4	4	41.27
5	5	50.30
6	6	61.16

TABLE II. EXPERIMENTAL RESULT OF 3-DSS (NO EXISTING DATA)

Query Request No.	Distance between Start of Data Insertion Time and Query Delivery Time (Minute)	Volume of Query Result Data (%)
1	1	10.70
2	2	20.30
3	3	30.60
4	4	41.15
5	5	50.00
6	6	60.80

TABLE III. EXPERIMENTAL RESULT OF 3-DSS (1GB EXISTING DATA)

Query Request No.	Distance between Start of Data Insertion Time and Query Delivery Time (Minute)	Volume of Query Result Data (%)
1	1	10.30
2	2	18.20
3	3	26.37
4	4	33.32
5	5	41.60
6	6	49.78

TABLE IV. EXPERIMENTAL RESULT OF 3-DSS (PARTITIONING)

Query Request No.	Distance between Start of Data Insertion Time and Query Delivery Time (Minute)	Volume of Query Result Data (%)
1	1	10.90
2	2	20.40
3	3	30.38
4	4	41.00
5	5	50.10
6	6	61.10

Table I is representing the experimental result for the 2-DSS. Table II, Table III and Table IV are showing the experimental result for the non-partitioned 3-DSS, the non-partitioned 3-DSS with 1 GB existing data and the partitioned 3-DSS with 1 GB existing data respectively. Query result of Table I and Table II is almost the same. There is a big difference between the query result of Table I and Table III. Indexing of data was the cause for this difference. It was not visible in the Table II for the low volume of data (only newly data was inserted and no existing data were there). As, partition was done after 1GB of data , Table IV presents almost the same volume of query result for each query request like Table I. Therefore, it can be said that 3-DSS can provide real-time information support. Further, as 3-DSS does not need the data transfer from non-indexed DSS to indexed DSS, it can provide information support seamlessly.

VII. CONCLUSION AND FUTURE WORK

This paper showed that the 3-DSS model can provide seamless real-time information support from the IMS. It is quite possible to down any of the DSSs of the 3-DSS at the period of execution in the real world. Therefore, there should have a redundant or replication system of 3-DSS to provide information support service in the down period. Additionally, a recovery system needs to develop for this 3-DSS for recovering the data for the failure of the system for crashing or some other reasons. Replication and recovery system are described briefly in this paper. Further, details work is needed on the dynamic partitioning system. Therefore, future work will be conducted on the replication system of the 3-DSS, the recovery system of the 3-DSS and the dynamic partitioning system of the 3-DSS in a broader aspect. Additionally, comparison of 3-DSS oriented IMS with the 2-DSS oriented IMS will also be the future research work.

REFERENCES

[1] M. Bouzeghoub, F. Fabret and M. Matulovic-Broqué, "Modeling Data Warehouse Refreshment Process as a Workflow Application", Proceedings of the International Workshop on Design and Management of Data Warehouses, 1999, pp. 6.1-6.12.

[2] D.P. Ballou and H.L. Pazer, "Modeling data and process quality in multi-input, multi-output information systems", Management Science, vol. 31, 1985, pp. 150–162.

[3] C. Batini and M. Scannapieco, "Data Quality: Concepts, Methodologies and Techniques", Publisher: Springer, Berlin, Germany, 2006.

[4] L. Bellatreche, K. Karlapalem, M. Mohania and M. Schneider, "What can partitioning do for your data warehouses and data marts?", IEEE, 2000, pp. 437-445.

[5] C. Cappiello, C. Francalanci and B. Pernici, "Data Quality and Multichannel Services", PhD Thesis. Politecnico di Milano, 2005.

[6] C. Cappiello, C. Francalanci and B. Pernici, "Time-Related Factors of Data Quality in Multichannel Information Systems", Journal of Management Information Systems, vol. 20, 2003, pp. 71-92.

[7] C. Cappiello, C. Francalanci and B. Pernici, "A Self-monitoring System to Satisfy Data Quality Requirements", Springer Verlag, vol. 3761, 2005, pp. 1535-1552.

[8] C. Cappiello and M. Helfert, "Analyzing Data Quality Trade-Offs in Data-Redundant Systems", Interdisciplinary Aspects of Information Systems Studies , Physica-Verlag HD, 2008, pp. 199-205.

[9] B.A. Forouzan, C. Coombs and S.C. Fegan, "Data Communications and Networking", Publisher: Tata Mcgraw-Hill, 2003.

[10] J.H. Hanson and M.J. Willshire**, "**Modeling a Faster Data Warehouse", IEEE, 1997, pp. 260-265.

[11] Y. Hu, S. Sundara and J. Srinivasan, "Supporting Time-Constrained SQL Queries in Oracle", Proceedings of the 33rd international conference on Very large data bases, 2007, pp. 1207-1218.

[12] W.H. Inmon, R.H. Terdeman, J. Norris-Montanari and D. Meers, "Data Warehousing for E-Business", J. Wiley & Sons, 2001.

[13] M.V. Mannino and Z. Walter, "A framework for data warehouse refresh policies". Decision Support Systems, vol. 42, 2006, pp. 121-143 .

[14] C.L. Pape and S. Gancarski¸ "Replica Refresh Strategies in a Database Cluster", Springer-Verlag, LNCS vol. 4395, 2007, pp. 679-691.

[15] C.L. Pape, S. Gancarski and P. Valduriez, "Refresco: Improving Query Performance Through Freshness Control in a Database Cluster", Springer-Verlag, vol. 3290, 2004, pp. 174-193.

[16] T. Padron-McCarthy and T. Risch, "Performance-Polymorphic Execution of Real-Time Queries**.** Workshop on Real-Time Databases: Issues and Applications", Newport Beach, California, USA, 1996. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.2149. [Accessed 2nd January 2013].

[17] J.F. Roddick and M. Schrefl, "Towards an Accommodation of Delay in Temporal Active Databases", 11th Australasian Database Conference (ADC), 2000.

[18] R.J. Santos and J. Bernardino, "Real-Time Data Warehouse Loading Methodology", ACM, vol. 299, 2008, pp. 49-58.

[19] A. Silberschatz, H.F. Korth and S. Sudarshan, "Database System Concepts". Publisher: Mcgraw-Hill, 1997.

[20] D. Theodoratus and M. Bouzeghoub, "Data Currency Quality Factors in Data Warehouse Design", Int. Workshop on Design and Management of Data Warehouses (DMDW), 1999.

[21] A. Vavouras, S. Gatziu and K.R. Dittrich, "Modeling and Executing the Data Warehouse Refreshment Process",IEEE, 2000, pp. 66-73.

[22] S.V. Vrbsky, "A data model for approximate query processing of real-time databases", ACM Data & Knowledge Engineering, vol. 21, 1996, pp. 79-102.

[23] R.Y. Wang, M. Ziad and Y.W. Lee, "Data Quality", Publisher: Kluwer Academic, 2001.

[24] T. Zurek and K. Kreplin, "SAP Business Information Warehouse From Data Warehousing to an E-Business Platform", 17th Int. Conf. on Data Engineering (ICDE), 2001.

[25] K. Zdenek , M. Kamil, M. Petr and S. Olga, "On updating the data warehouse from multiple data sources", Springer, vol. 1460, 1998, pp. 767-775.