International Journal of Grid and Distributed Computing
Vol. 10, No. 9 (2017), pp.13-32
http://dx.doi.org/10.14257/ijgdc.2017.10.9.02

Data Storing in Intelligent and Distributed Data Warehouse using Unique Identification Number

Abhay Kumar Agarwal[1] and Neelendra Badal[2]

[1,2]*Kamla Nehru Institute of Technology, Sultanpur, India*
[1]*abhay.knit08@gmail.com,* [2]*n_badal@hotmail.com*

Abstract

The paper presents a novel approach that locates a most suitable data warehouse in IDDW to store user data in it. Unique Identification Number (UIN) is used for the purpose, which is generated by an algorithm presented in the paper. The address of the data warehouse in IDDW where user data is to be stored is found using UIN by the Data Warehouse Locator Bridge (DLB) in which an algorithm is executed. Both DLB and algorithm are also designed, developed and presented in the paper. The data used in the work is real world data. The approach is found feasible from the results obtained by performing experimental work on the implementation of the case presented. The result shows that the time consumed for the data to get stored in data warehouse build at different levels of IDDW decreases with level of IDDW. From the results, it is seen that any real world scenario that may be implemented like IDDW must have more number of levels and data warehouse should be available at lower most possible level of IDDW. The different real world scenarios that may be broken into hierarchies of different levels may be implemented like IDDW. It may also be applied in the scenarios where a unique identity is assigned to a user. Some of the scenarios where the IDDW may be useful to be implemented are: Banking; Vehicle registration system; UIDAI, Chain Business system, eNPS - National Pension System; etc.

Keywords*: Data Storing, Distributed, Intelligent, UIN, Data Warehouse*

1. Introduction

Data Warehouses are a crucial technology for current competitive organizations in the globalized world. It becomes more crucial in the case where the organization is distributed geographically. Such organization requires distributed data warehouse. Out of the so many concepts for designing distributed data warehouse one such concept is a hierarchal distributed data warehouse. Such distributed data warehouse if designed properly, proofs, to be a great boon to the organization implementing it. Efficient, data storage method in such hierarchal distributed data warehouse further facilitate the organizations.

IDDW is a *N*-Level hierarchal distributed data warehouse proposed in [1]. In this paper an efficient approach of data storage in IDDW is proposed. It provides a new functionality to IDDW by locating the data warehouse in IDDW using Unique Identification Number. The data is entered by the user and stored in the Common table of located data warehouse in IDDW. The proposed approach requires the restructuring of a Common table previously used in IDDW [1]. The various other modules are also needed in the proposed approach each of which are designed and implemented. Each such module are presented in the paper.

The advantages of the proposed approach are: data automatically get stored in most desired data warehouse; every user registered in the system gets a Unique Identification

Received (January 17, 2017), Review Result (May 17, 2017), Accepted (September 17, 2017)

ISSN: 2005-4262 IJGDC

Number; user data is stored in the data warehouse nearest to it; data extraction from IDDW becomes simple with the help of UIN; *etc*.

The various sections in the paper are as follows: In the next section Related Works are presented. In the next-to-next section, Restructuring of Common Table is presented. Subsequently, the Proposed Approach of locating data warehouse in IDDW, Case study, Experimental work and Result is presented. Finally the Conclusion is presented.

2. Related Works

Inmon W. H. in [6] proposed an approach to build the distributed data warehouse. The approach assumes the presence of both local and global data warehouses with data stored in each is mutually exclusive. White C. in [15] proposed an approach which is a combination of centralized data warehouse containing normalized and detailed data and a decentralized data marts which contains de-normalized and summarized data that is of importance to a specific user.

Zhou S. *et al*. in [16] proposed hierarchically distributed data warehouse (HDDW). This architecture combines the virtue of data marts and distributed data warehouse. HDDW was built with a bottom-up and level by level method, thus summarizing the data, level over level and simplifying management and maintenance processes. Su H. *et al* in [12] proposed a new model Concept Hierarchy Distributed Data Warehouse (CHDDW) suggests the constructional approach for making it is hierarchical, bottom-up based on the concept hierarchy. It combines the qualities of both data marts and distributed data warehouse.

Wehrle P. *et al*. in [13] proposed an architectural model for distributing a data warehouse on a computing grid. They uses the chunks for getting suitable fragments of data warehouse for each grid node and are indexed in a uniform way. Another work based on grid was proposed by Costa R. *et al*. in [5] in which authors investigated an efficient architecture Grid-Dwpa, to deploy large data warehouses in grids. Another work on grid-aware data warehouse was proposed by Lawrence M. in [8], the proposed work is for a two-tiered, grid-based data warehouse. The Grid consider the scenario where the data of a single organization was distributed across a number of operational databases at remote locations such that first tier have local data and remote database server in second tier. Each operational database has capabilities for answering OLAP queries and access to storage resources located near-by. Wehrle P. *et al*. in [14] proposed a work which also dealt with distributed, grid-aware environment. In proposed work fact table data is partitioned and distributed across participant nodes while dimension table data is replicated. For providing the local information about data stored at each node a local data index service is used, while for accessing remote data communication service is used, to do so communication service uses local data index service from participant grid's nodes. Noaman A. Y. *et al*. in [9],[10] proposed an architecture for distributed data warehouse that is based on top-Down approach and presents two fundamental issues: fragmentation and allocation of the fragment to various sites. Bernardino J. *et al*. in [2] designed a new technique called data warehouse stripping(DWS), which is a round robin data partitioning approach for relational data warehouse. Using this technique huge data warehouse can be distributed over a cluster of computers having same star schema at all computers and dimension table replicated in each machine. In [17] Zhao J. *et al*. describes that how Abstract State Machines (ASM) can be used to design distributed data warehouses. Chaki N. *et al*. in [3] proposed a new analytical model using Petri Net for distributed data management in a data warehouse. A generic scalable model design of a Virtual Data Warehouse has also been made by the authors. Petri Net is as an analytical tool used for modeling, concurrent, distributed, asynchronous, parallel system. Karima T. *et al*. in [7] presented and proposed a data warehouse decentralization strategy based on cost-based fragment allocation and replication algorithm. Using the algorithm proposed and Data

Warehouse Fragment Evaluator (DWFE), fragments were distributed through several sites.

3. Re-Structured Common Table in IDDW

The Common table used before in [1] has the number of columns equal to the number of levels in IDDW. The Common table after modification has one more column, as compared to the Common table used before. Thus, Common table now has one more column, as the number of levels in IDDW *i.e.*, N-Level IDDW has N+1 columns. The column added is named as UIN. The Common table after modifications is shown in Figure 1.

UIN	Level 1	Level 2	….	Level L	…..	Level N

Figure 1. The Modified Common Table

The first column in the Common table shown in the Figure 1 stores the Unique Identification Number abbreviated as UIN. The purpose of other columns in modified Common table i.e., second column to the last column is same, as discussed in [1]. In a nutshell, from a row of Common table shown in Figure 1 the Location Name in each level of a hierarchy of IDDW and the generated UIN corresponding Location Name is obtained.

The UIN is generated from the data entered by the user and a 3- digit unique ID that is serially generated at last level of IDDW. The entered data are the Location name in the first N-1 level (from top) of same hierarchy of IDDW.

4. Proposed Approach of locating Data Warehouse in IDDW

In this section, the proposed approach is presented. To provide the outlook of the approach, first, its flow chart is presented. Next, the design and role of various components needed in the approach are presented. Finally, the illustration of interconnections of various components and overall working is presented.

4.1. Flow Chart of the Proposed Approach to Locate Data Warehouse in IDDW

The user enters the data which is Location Names via active form designed for it. The Mapping tables provide the Location ID of each Location Names entered by the user. UIN Generation Module generates the UIN using the Location ID's provided by the Mapping tables generated from Central Look-Up Data Warehouse as shown in Figure 2.

The UIN that is generated in UIN Generation Module is passed to Identifier Finder in Data Warehouse Locator Bridge (DLB) to calculate the Identifier. Data Warehouse Locator uses this Identifier to locate the most suitable data warehouse in IDDW to store the data entered by the user. The information about the availability of most suitable data warehouse is obtained from Central Look-Up Data Warehouse. Identifier Finder recursively calculates Identifier until the desired information is obtained from Central Look-Up Data Warehouse. The desired information obtained from Central Look-Up Data Warehouse is then validated and verified for the error if any, occurred in it.

After the desired information obtained from Central Look-Up Data Warehouse is validated and verified and has no error then the same is moved to ETL. It then loads the data in the most suitable data warehouse of IDDW. If the error is found in the desired information the entire process in DLB is re-executed, starting, from UIN moving to Identifier Finder as shown in Figure 2.

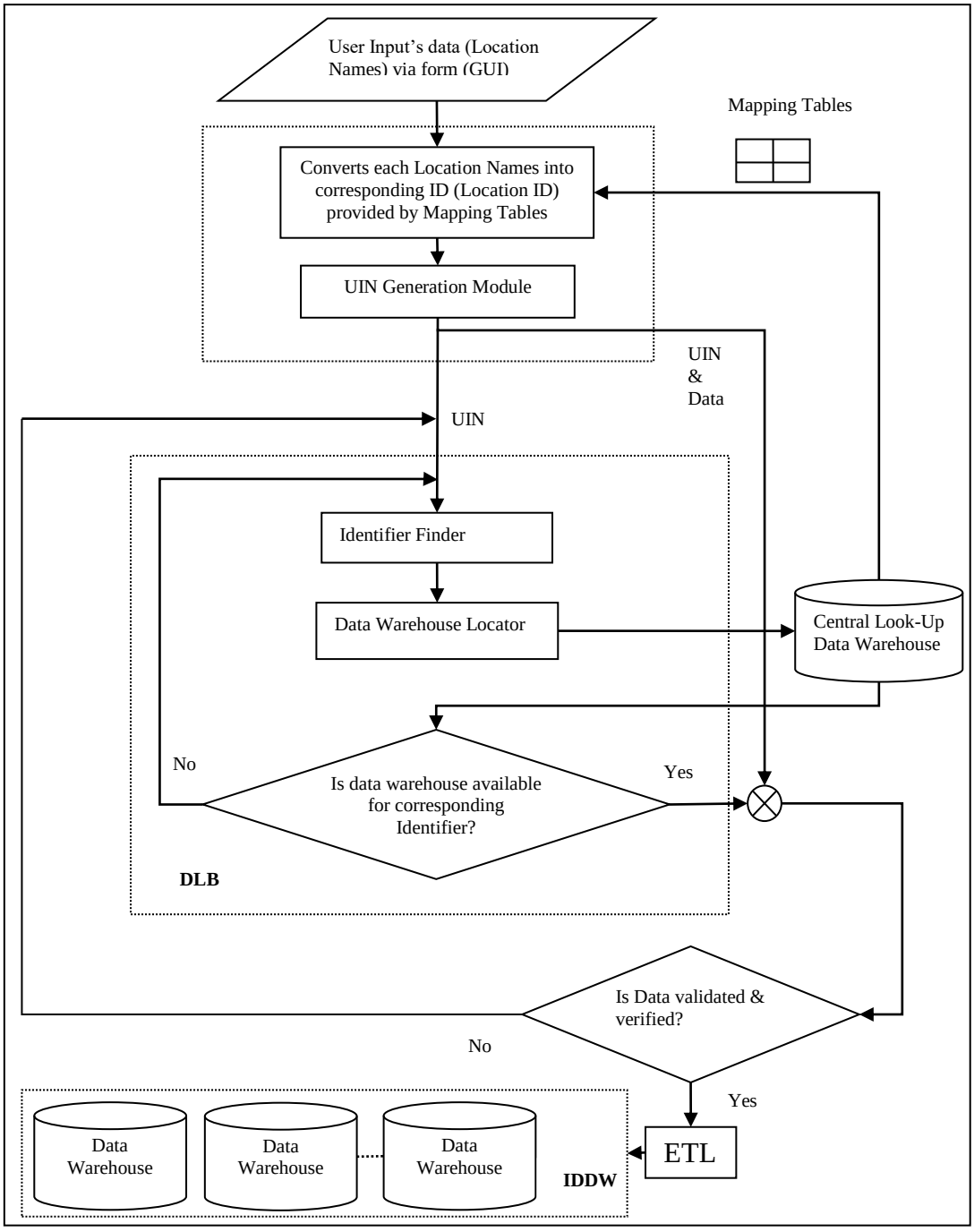

Figure 2. Flow Chart of Approach to Locate Data Warehouse in IDDW

The flow chart illustrates the proposed approach from the starting of the user entering the data to the storing of data in IDDW. The implementation of the proposed approach requires various components each with a specific role to perform, thus the design and role of each component required is presented in the next section.

4.2. Design and Role of Different Components used in the Proposed Approach

The various components required in the proposed approach include Central Look-Up Data Warehouse, Mapping Tables, User Interface Module, Global RAM, Replica Manager, Data Warehouse Locator Bridge (DLB), Validation & Verification Module and ETL tool. The role of each component is presented one-by-one in various sub-sections:

4.2.1. Central Look-Up Data Warehouse

Central Look-Up Data Warehouse is a centralized repository and plays an important role in the proposed approach. One of the important roles of it is that the Mapping Tables are generated from it. The contents in Central Look-Up Data Warehouse are N tables as shown in Figure 3. The N tables refer to the N levels of IDDW, respectively. Each table has four columns. The attribute of the first column in each table acts as a primary key. The name of other three columns is same in all the tables. All these N tables in Central Look-Up Data Warehouse are kept in the form of Star schema. In Star schema there is one Dimension table and N Fact tables. The Dimension table has one column and N rows. The N rows in Dimension table stores the value of attributes of each table that act as a primary key for that table.

Level 1

Identifier 1	Location Name	Availability of DW	Machine Address
$a_1 \dots a_{I_1}$			

Level 2

Identifier 2	Location Name	Availability of DW	Machine Address
$a_1 \dots a_{I_1} a_{I_1+1} \dots . a_{I_1+I_2}$			

.
.

Level L

Identifier L	Location Name	Availability of DW	Machine Address
$a_1 \dots a_{I_1} a_{I_1+1} \dots a_{I_1+I_2} \dots \dots a_{I_1+I_2 \dots\dots +I_{L-1}+1} \dots a_{I_1+I_2 \dots\dots +I_L}$			

.
.

Level N

Identifier N	Location Name	Availability of DW	Machine Address
$a_1 \dots \dots a_{I_1+I_2 \dots\dots +I_{L-1}+1} \dots a_{I_1+I_2 \dots\dots +I_L} \dots \dots a_{I_1+I_2 \dots\dots +I_L \dots\dots +I_N}$			

Figure 3. Contents in Central Look-Up Data Warehouse.

The names of four columns in the table of Level L are Identifier L, Location Name, Availability of DW and Machine Address. The name of first column of each table depends on the level number i.e. for Level 1 it becomes Identifier 1, for Level 2 it becomes Identifier 2 and so on up to Level N. The column with name, Identifier L in Level L stores the Location ID's clubbed together of all the locations from Level 1 to Level L. Thus it stores various Identifiers formed by the clubbed digits ($a_1 \dots a_{I_1} a_{I_1+1} \dots a_{I_1+I_2} \dots \dots a_{I_1+I_2 \dots\dots +I_{L-1}+1} \dots a_{I_1+I_2 \dots\dots +I_L}$). Here I_L is the number of digits in the Location ID of Level L. It means that column with name Identifier 1 in Level 1($L = 1$) stores various Identifiers formed by the clubbed digits ($a_1 \dots a_{I_1}$). As the clubbed digits ($a_1 \dots a_{I_1}$) is same as Location ID of Level 1 the name of first column in the table of this level can be named as Location ID. However to maintained uniformity it is named as Identifier 1.

The column with name "Location Name" in each table store the Location Name of predefined locations on the basis of which IDDW is formed. The column with name "Availability of DW" stores either 1 or 0. The value '1' represents that data warehouse is available and the value '0' represents that data warehouse is not available, at the location

in IDDW corresponding to the Location Name and Identifier. For the value '1' in column "Availability of DW", the column Machine Address contains the address of machine having a data warehouse. While for value '0' in column "Availability of DW" a null value is contained in column Machine Address.

The values in column Identifier L provides the hierarchal information from the top most level up to the level L. Whenever, a new data warehouse is built at any location of any level in IDDW its information is entered in Central Look-Up Data Warehouse. The information consists of the Machine address in which data warehouse has been built. The status is also changed from 0 to 1 in Availability of DW column. Thus, Central Look-Up Data Warehouse is always updated with the information of last data warehouse built in IDDW.

4.2.2. Mapping Tables

Mapping tables is a feature, allows you to define virtual tables, displaying data from one or more linked physical tables. The mapping tables can include all, or a subset of, the fields in the physical tables and can also include calculated columns [18]. As the user enters the values in the textboxes of the form shown in Figure 6, the Central Look-Up Data Warehouse connected to the form generates the Mapping Tables like shown in Figure 4.

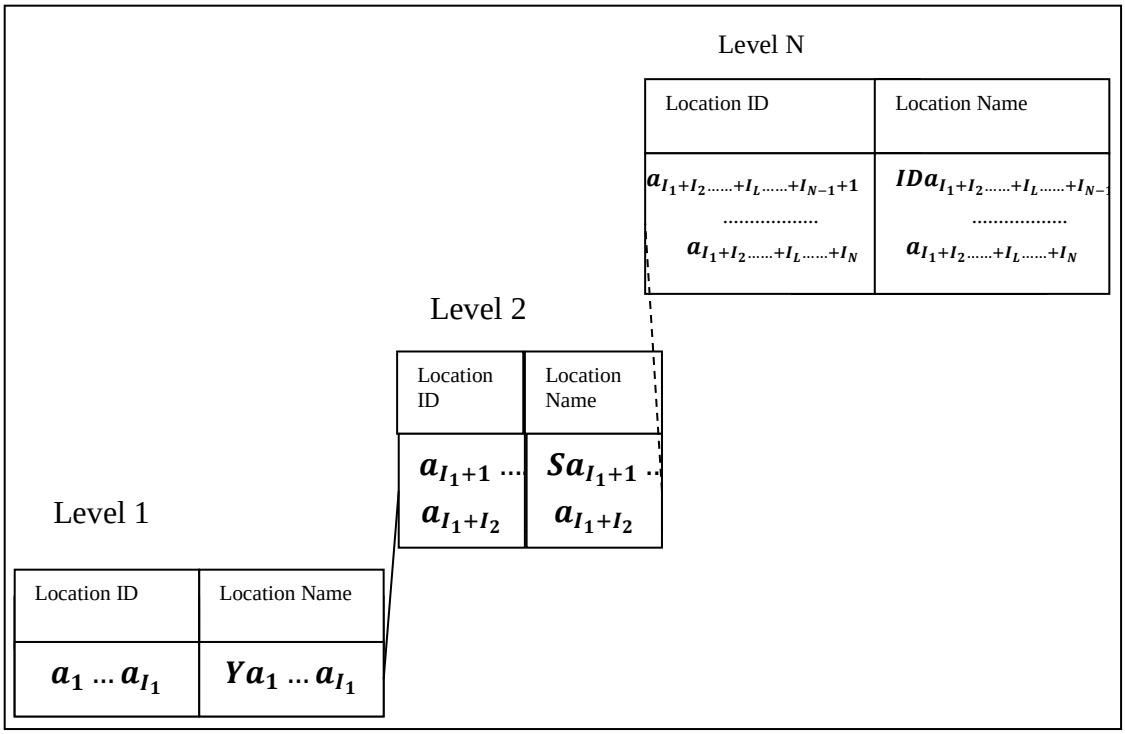

Figure 4. The Structure of Generated Mapping Tables

The generated Mapping Tables contains one table each for one level of N-Level IDDW. Each mapping table contains two columns: Location ID and Location Name. The Location ID, of all locations at any particular level, has the same number of digits. The Location ID is a number between minimum and maximum (both inclusive) value formed by the digits, pre-defined for a particular level. The mapping table of each level is linked to its upper-level table as shown in Figure 4. Mapping Tables performs the task of providing the Location ID of the Location Name entered by the user via the active form in User Interface Module.

4.2.3. User Interface Module

User Interface Module acts as an interface between a user and the system. It is designed to perform the following tasks: get user inputs; generates UIN; returns a copy of the generated UIN back to the user, and passes generated UIN and data to the Global RAM. User Interface Module consists of a Form, UIN Generation Module and a Summation block. Each is arranged as shown in Figure 5.

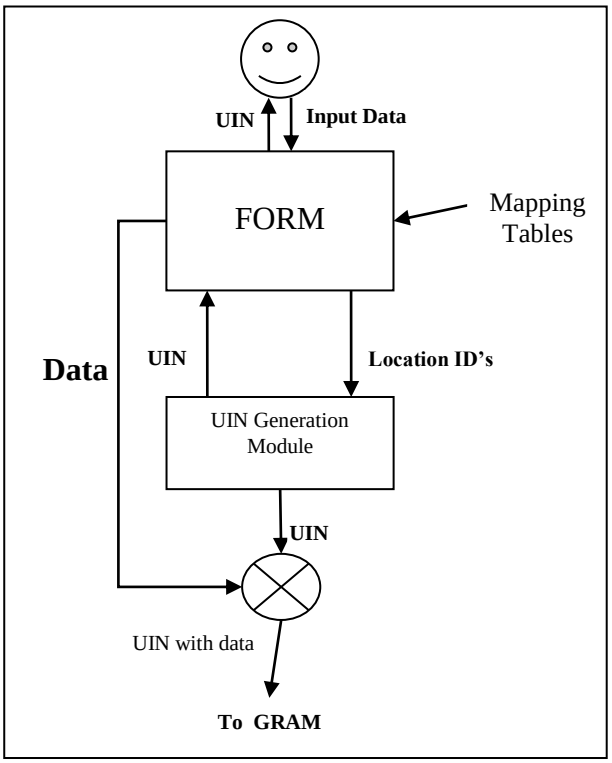

Figure 5. User Interface Module

A Form in user interface module is an active form, designed, to get user inputs and to display the processed information *i.e.*, UIN back to the user. The user enters the data *i.e.*, Location Names via the form. The data is processed to generate UIN which is displayed to the user via a form. The UIN Generation Module generates a UIN, returns it to the user via form and forwards the UIN to Global RAM via summation block where it summed up with the data entered by the user.

4.2.3.1. Form to Get User Inputs

There are N text boxes in the form designed like shown in the Figure 6. The N text boxes, represents N levels of IDDW. Each text box is labeled accordingly from the top with the name same as of each level of N-Level IDDW respectively. User enters the Location Names in the text boxes contained in the form.

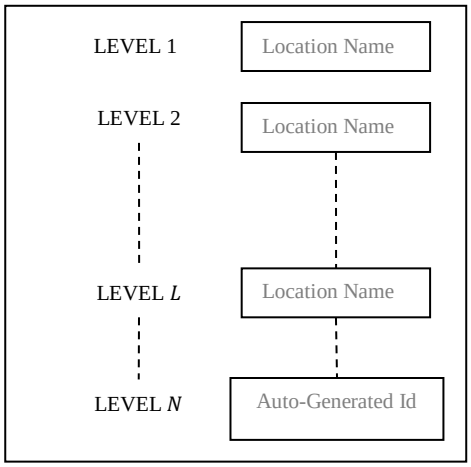

Figure 6. An Outlook of the Form

The Location Name to be entered by the user in the number of text boxes depends upon the level of IDDW to which user belongs. There are two types of users first type requires the Location Name to be entered for each level. Such user enters the Location Name in each text boxes from the top up to $N-1$ level. Second type requires the Location Name up to some intermediate level from top. Such users enter the Location Name in only those text boxes and rests are left blanked. For both type of users a unique ID consisting of I_N number of digits [1] is auto-generated serially at the level N of IDDW. The Location ID is generated for each Location Name entered by user in the text boxes, respectively. The Location ID is obtained via Mapping Tables that are generated from the Central Look-Up Data Warehouse. The text boxes that are left blanked by the user, number of zeros equal to number of digits in that level, Location ID, is assigned. Finally, all the Location ID's are clubbed together, in hierarchal order from the top, and is passed to the UIN Generation module for the generation of UIN.

4.2.3.2. UIN Generation Module

UIN Generation Module generates the UIN. The UIN is composed of Location ID's of all the levels clubbed together in hierarchal order and a check digit C_k. The Location IDs clubbed together is moved to UIN generation module *i.e.*, $(a_1 \dots a_{I_1} \dots a_{I_1+I_2 \dots\dots+I_{L-1}+1} \dots a_{I_1+I_2 \dots\dots+I_L} \dots a_{I_1+I_2 \dots\dots+I_L \dots\dots+I_N}$ (where; I_L represents the number of digits in the Location ID of level L) is moved to UIN generation module. The check digit C_k is calculated by using an algorithm shown in Figure 7.

To calculate the check digit C_k the algorithm shown in Figure 7, use digits of Location ID's that are clubbed together in hierarchal order. The purpose of the check digit C_k in UIN is the identification of single digit error generated, if any, in it. The single digit error means any one digit of UIN is changed to some other value. The error in UIN might occur during transposition, transmission, and transcription.

Thus the generated UIN is $a_1 \dots a_{I_1} \dots a_{I_1+I_2 \dots\dots+I_{L-1}+1} \dots a_{I_1+I_2 \dots\dots+I_L} \dots a_{I_1+I_2 \dots\dots+I_L \dots\dots+I_N} C_k$. Once, the UIN is generated it is returned to the user and is forwarded to GRAM via summation block where it is summed up with the data entered by the user and need to be stored in the Common table of data warehouse in IDDW.

Step 1: Double the value of alternate digits of the Location ID's of all the levels clubbed together in hierarchal order beginning with the second digit from the left i.e.
Do $2a_i \ \forall$ even i
where; $i = 1 \ to \ I_1 + I_2 \ldots\ldots + I_L \ldots\ldots + I_N$

Step 2: Add the individual digits comprising the products obtained in Step 1 to each of the unaffected digits of the original number.
$M = m_1 + m_2 + m_3$ Where;
$m_1 = \sum_{\substack{i=1 \\ i \ is \ odd}}^{16} a_i$
$m_2 = \sum_{\substack{i=1 \\ i \ is \ even}}^{16} 2a_i \ \forall \ a_i < 5$
$m_3 = \sum_{\substack{i=1 \\ i \ is \ even}}^{16} (Ten's \ of \ 2a_i + Unit \ of \ 2a_i) \ \forall \ a_i \geq 5$

Step 3: Subtract the total from the near highest multiple of 10 to determine the check C_k digit.
$C_k = M' - M$ Where; M' = Ceiling (M, 10)

Figure 7. An Algorithm to Calculate Check Digit C_k

4.2.4. Global Ram (GRAM)

Random-Access Memory (RAM) is a form of computer data storage. It allows stored data to be accessed in any order (*i.e.*, at random). "Random" refers to the idea that any piece of data can be returned in a constant time, regardless of its physical location and whether or not it is related to the previous piece of data [11]. A Global RAM of the order of Gigabytes is placed between User Interface Module and Replica Manager. It is available to all the users entering their data. It stores the UIN generated in User Interface Module along with the data entered by the user. UIN and data in GRAM remained stored until it is not stored in the most suitable data warehouse in IDDW, without any error. Also, the UIN and data stored in GRAM are utilized by Replica manager for further processing.

4.2.5. Replica Manager

Replication is the process of creating and maintaining replica versions of database objects (*e.g.*, tables, files *etc.*) in a distributed database system. Replication can improve performance and increase the availability of applications because alternate data access options become available. For example, users can access a local database rather than a remote server to minimize network traffic. Furthermore, the application can continue to function if parts of the distributed database are down as replicas of the data might still be accessible. The role of a Replica Manager is to create or delete copies of file instances, or replicas, within specified storage systems [4]. Often, a replica is created because the new storage location offers better performance or availability for accesses to or from a particular location.

Replica Manager generates the two replicas of both UIN and data stored in GRAM. The two replicas are needed because each is passed to two different places in DLB. One replica is moved to Identifier Finder and another replica is moved to summation block. As UIN is only required by Identifier Finder so from one of the replica UIN is extracted and moved to Identifier Finder, while the second replica is moved as it is to summation block in DLB. The role of Replica Manager also comes into play when more than one user needs to access the same information simultaneously.

4.2.6. Data Warehouse Locator Bridge (DLB)

Data Warehouse Locator Bridge (DLB) is one of the most vital components designed in the proposed approach. It includes four sub-components: a summation block, Local RAM (LRAM), Identifier Finder and a Data Warehouse Locator. First two sub-

components are used as a standard. The other two sub-components are software components and implemented by writing a program for the algorithm designed and presented in Figure 8.

The Identifier Finder obtains UIN from the Replica Manager. It uses the obtained UIN to find the Identifier, which it passes to the Data Warehouse Locator. Data Warehouse Locator uses this Identifier to finds the Machine Address from Central Look-Up Data Warehouse. The request made by Data Warehouse Locator to Central Look-Up Data Warehouse and the response it gets in vice-versa is done like RFTS. If a Machine Address is obtained then it is moved to summation block, else Data Warehouse Locator makes a request to Identifier Finder for a new Identifier. Summation block clubs together the Machine Address with the second replica consisting of UIN and data obtained from Replica Manager. The entire information comprising of the Machine Address, UIN and data are moved to LRAM in DLB. LRAM stores the obtained information which is further used by Validation and Verification module.

Algorithm to perform the tasks of DLB

The algorithm running in DLB does the following:

- Calculates Identifier recursively until Machine Address is obtained;
- Using Identifier, finds the Machine Address of data warehouse from Central Look-Up Data Warehouse; and
- Transfers the Machine Address, UIN and data from LRAM to Validation and Verification module.

The detail regarding the variables used in the algorithm and the various inputs are given to the algorithm for its execution is as follows:

- Three integer variables y, i and n
- Two arrays m and p, of type integer and each of size $n-1$
- Level-Number, 'L' varying from 1 to N
- A UIN of n digits $(a_1 a_2 \dots \dots \dots \dots a_{n-1} c_k)$ //c_k is $a_n{}^{\text{th}}$ digit
- Number-of-Levels in IDDW be N // maximum value of level number
- Set of Location ID with predefined digits $(I_1, I_2, \dots I_L, \dots I_N)$ respectively for the levels of IDDW

The algorithm starts with the declaration of three variables y, i and n in step 1 and two arrays m and p in step 2. In step 3 and step 4, algorithm reads the generated UIN and N the Number-of-Levels in IDDW, respectively. Step 5, computes the number of digits n in UIN. In step 6 the value of $n-2$ is assigned to variable y. The value of N is assigned to variable L in step 7. In step 8, algorithm read the value of I_L *i.e.*, digits in predefined Location ID of level L of IDDW. In step 9 the value of Identifier is calculated and stored in an array. It fulfills the role of Identifier Finder. The calculated Identifier is moved to Central Look-Up Data Warehouse in step 10. In step 11, as the calculated Identifier matches with the Identifier in the table of level L in Central Look-Up Data Warehouse, then corresponding value in Availability of DW field of same table is looked for if it is founds to be 1 then Machine Address of the data warehouse for the Identifier is moved to Data Warehouse Locator in DLB which is further transferred to LRAM via summation block and then next to Validation and Verification module and algorithm stops. Else data warehouse not found message is sent to Data warehouse Locator, a new value of variable y is calculated, a new value of L is calculated and the algorithm repeats from step 8 to 11.

Step 1: Declare variables y, i, n
Step 2: Create two arrays 'm' and 'p' each of size $n-1$
Step 3: Read UIN
Step 4: Read N
Step 5: Compute n from UIN
Step 6: $y = n - 2$
Step 7: $L = N$
Step 8: Read I_L
Step 9: i = 0 to y
{
$m[i] = a_{i+1}$
$p[]$ = Read $m[i]$
}
End i
Step 10: Move $p[]$ through RFTS via Data Warehouse Locator to Central Look-Up Data Warehouse.
Step 11: At $p[]$= =Identifier in the table of level L in Central Look-Up Data Warehouse
If the value in the field "Availability of DW"=1
{
Return through RFTS the Machine Address to Data Warehouse Locator in DLB,
Transfer Machine Address from Data Warehouse Locator to LRAM via summation block,
Move UIN and data along with Machine Address from LRAM to Verification and Validation module,
END
}
Else
{
Return through RFTS a message, data warehouse not found, to Data Warehouse Locator,
$y = y - I_L$
$L = L - 1$
Repeat steps 8 to 11
}

Figure 8. An Algorithm to Perform the Tasks of DLB

4.2.7. Validation and Verification Module

This module identifies the error that might occur in any single digit of UIN. The importance of this module lies in the fact that if any single digit of UIN is changed and not looked for then data is stored in the incorrect data warehouse in IDDW. The error in UIN might occur during transposition, transmission and transcription of information in the entire process. The module identifies the error with the help of Check digit (the last digit of UIN).

The validation and verification module obtains Machine Address, UIN, and data from LRAM. It recalculates the check digit from the obtained UIN by using the algorithm in shown in Figure 7. Once the check digit is recalculated it is compared with the check digit already present in the same UIN. On comparing, if both the check digits turned out to be same it is assumed that no single digit error has occurred in UIN. However, if it comes out to be different than it is assumed that error has occurred in UIN and the entire process repeats by again accessing UIN and data from GRAM.

Thus, it can be seen that check digit present in UIN plays an important role in this module. If any single digit is changed to some other value and if this check digit has not been included in UIN then it would have been difficult to identify the error generated in

UIN thus leading to storage of data in an incorrect data warehouse of IDDW.

4.2.7. ETL Tool

ETL (Extraction, Transformation, and Load) tool performs the tasks of extracting Machine Address, UIN, and data from Validation & Verification module. It transforms the extracted UIN and data into the format, suitable for storing it, in a Common table of the data warehouse in IDDW. The Machine Address extracted helps to locate the data warehouse in IDDW to load UIN and data in the Common table of the located data warehouse.

4.3. Components Interconnections and Data Flow in the Proposed Approach

The flow chart of the proposed approach, design of components used in the approach and their roles has been discussed in previous sections. This section explains the proposed approach, the interconnection of various components used in the proposed approach and the flow of data and information in the proposed approach.

The flow of data in the proposed approach starts with the entering of the data (Location Names) by the user. The user enters its data in the form available in User Interface Module shown in Figure 9. User Interface Module performs its defined tasks of generating UIN, returning UIN to the user and moving of UIN and data to GRAM. The UIN is a n digits number with last digit as a check digit. The first $n-1$ digits of UIN are generated with the help of Mapping Tables.

GRAM stores the information (UIN and data) obtained from User Interface Modules until the most suitable data warehouse in IDDW, to store data, is not located. Replica Manager obtains the same information from GRAM for further processing. Replica Manager generates two replicas of this information. One of the replicas of information is moved to summation block in DLB, while from another replica of information UIN is extracted and moved to Identifier Finder in DLB.

UIN is used as an input to calculate Identifier by an algorithm executed in DLB. The identifier is calculated, recursively, until Data Warehouse Locator obtains the Machine Address of a most suitable data warehouse in IDDW from Central Look-Up Data Warehouse. The Machine Address is obtained by Data Warehouse Locator, through RFTS from Central Look-Up Data Warehouse. Once Machine Address is found Data Warehouse Locator passes this Machine Address to summation block in DLB. Here, the two information's, first the Machine Address and second the UIN and data obtained from Replica Manager combines, which is then stored in LRAM in DLB. The stored information in LRAM is used by Validation & Verification module.

Validation & Verification module identifies the transposition, transmission and transcription errors that might be generated in UIN. If an error is generated, it sent a request to GRAM for resending the information containing UIN and data entered by the user. If, no error is generated the information containing Machine Address, UIN and with data is moved to ETL tool. Finally, ETL performs its tasks and the UIN and data are stored in the Common table of the most suitable data warehouse in IDDW.

After presenting in the previous sections, the flow chart of proposed approach, designing and role of components used in proposed approach and component interconnections and data flow in proposed approach, the Case Study is presented is presented in the next section.

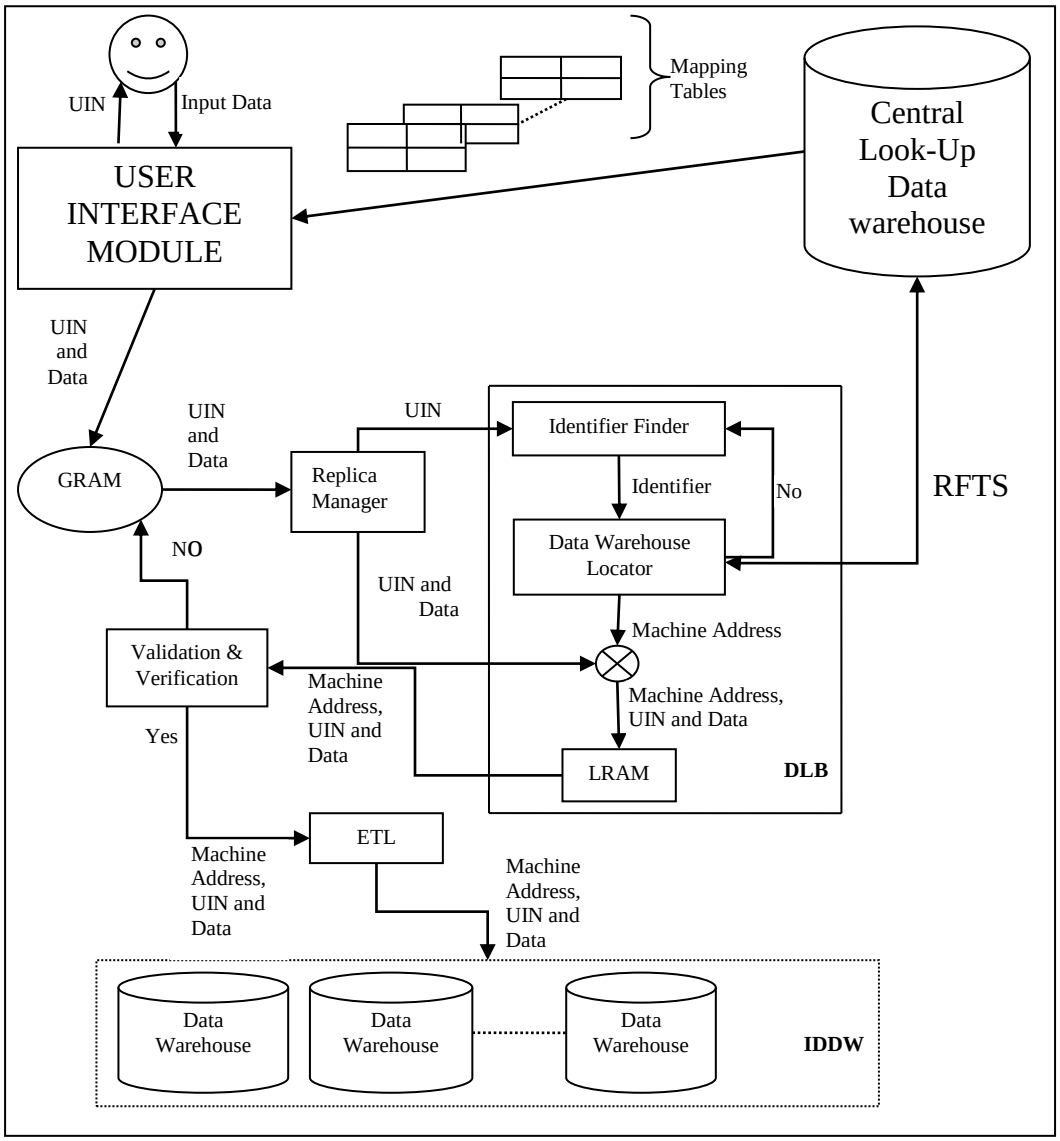

Figure 9. Components Interconnections and Data Flow in the Proposed Approach

5. Case Study

The 8-Level hierarchal structure presented in the case study of the previous chapter is utilized to evaluate the proposed approach presented in this chapter. The 8-Level hierarchal structure is shown again in Figure 10. The number of levels and name of each level is same in both the 8-Level hierarchal structure shown in the [1] and Figure 10. The number of levels N is 8 and their names from top are as Y for Year, S for State, U for University, I for Institute, D for Department, C for Category, P for Position and ID for Unique id, respectively. The other information shown in Figure 10 is the Location ID in the bracket placed after the level name. From it number of digits in each Location ID can be known. Like, for Level II which is Level State the Location ID is written like S (a_3a_4) which means the Location ID of all locations at Level State has two digits. Likewise, the number of digits in the Location ID of all the locations at each level is known. Thus the number of digits in the Location ID of various levels of 8-Level hierarchal structure are $I_1 = 2, I_2 = 2, I_3 = 2,\ I_4 = 3, I_5 = 2, I_6 = 1, I_7 = 1$ and $I_8 = 3$ respectively.

According to the number of digits in Location ID's of each level, a year is represented by two digits *i.e.*, last two digits of a year, for *e.g.*, 2013 is allotted Location ID=13. For any year a state is represented by two digits 01 to 99, for *e.g.*, Uttar Pradesh by Location ID, say 30. For any state a university is represented by two digits 01 to 99, for *e.g.*, UPTU by Location ID, say 20. For any university an institute is represented by three digits 001 to 999, for *e.g.*, KNIT by Location ID, say 104. For any institute a department is represented by two digits 01 to 99, for *e.g.*, Computer Science & Engineering by Location ID, say 10. For any department, a category is represented by one digit 1 to 9, for *e.g.*, Faculty by Location ID, say 5. For any category a position is represented by one digit 1 to 9, for *e.g.*, Professor by Location ID, say 2. And finally, a unique ID of three digits (between 001 and 999) is assigned serially to each user. Thus a hierarchy 'H1' is formed from top to bottom with Location Name (Location ID) provided for each level, {H1: 2013(13) | Uttar Pradesh (30) | UPTU (20) | KNIT (104) | CSE (10) | Faculty (5) | Professor (2) | Unique ID}.

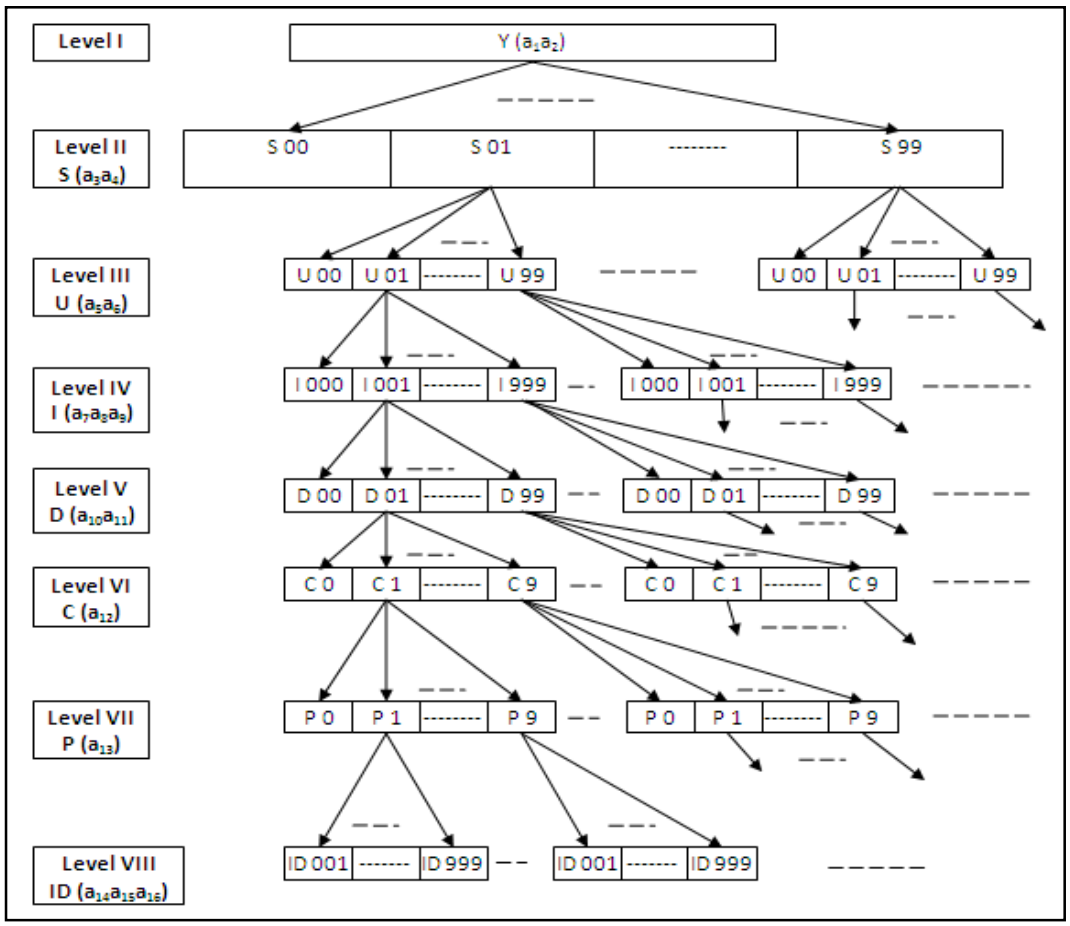

Figure 10. Case Study Fitted to form 8-Level Hierarchal Structure

The Common table in each data warehouse in 8-Level hierarchal structure previously had the same number of columns as a number of levels in 8-Level hierarchal structure. As illustrated, the number of columns in the Common table present in the proposed approach has one more column than the number of columns in the Common table used previously. The column added here in Common table is named as UIN. Thus, accordingly, each Common table is modified in each data warehouse built in the 8-Level hierarchal structure presented in the previous chapter. The column named UIN added in each Common table stores the UIN.

Next section presents the Experimental Work that contains two subsections Experimental Set-Up and Experiment.

6. Experiment Work

The hierarchy H1 of 8-level hierarchal structure and the other components used in the proposed approach is implemented using different hardware and software. The requirement of such hardware and software is presented in the Experimental Set-Up presented in the next subsection. How the different components are connected is also presented in the Experimental Set-Up.

6.1. Experiment Set-Up

In the implementation of the proposed approach all the programs are written in JAVA with Net Beans: 6.9 as an IDE, using the Web server: Apache Tomcat 6.0.26. All the tables required in the implementation of the proposed approach are built in the database MySql 5.0.45. The hardware requirements include one 16-port Switch, one database server and ten machines each with the following specifications: Core 2 Duo processor, 3.0 GHz, and 1GB RAM.

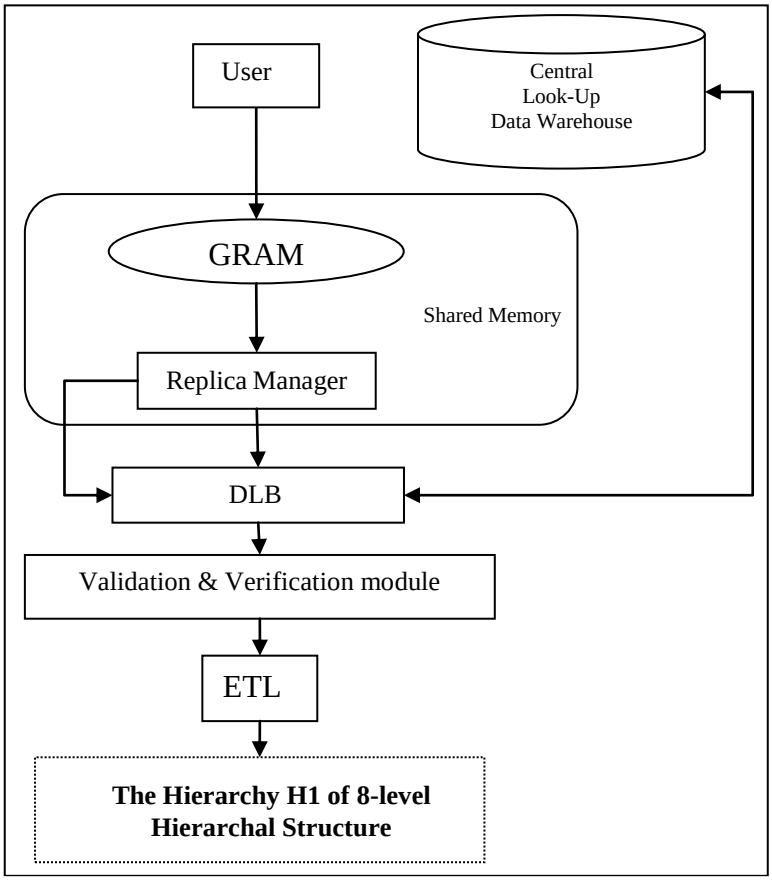

Figure 11. Set-Up to Perform Experiments

The hierarchy H1 is designed using the seven ports of the 16-port Switch and implemented using the program written for it. The seven ports of a switch represent top seven levels of hierarchy H1 and a machine is connected to each port. On another machine, the Central Look-Up data warehouse is built. One machine is used to perform the role of DLB by executing the algorithm shown in Figure 8. The same machine also performs the role of Validation and Verification module by execution of algorithm shown

in Figure 7. From one machine a user enters it's data. The same machine also performs all the roles of User Interface Module. The components GRAM, Replica Manager and ETL are used as a standard. GRAM and Replica Manager are placed in shared memory and are accessible to all the users. The entire set-up is shown in Figure 11.

With the completion of experimental set-up work, next, the experiment is performed. The next subsection presents the Experiment that is performed on the set-up.

6.2. Experiment

The experiment starts by placing a data warehouse that is built using a database server in the location of Level Year. It is the first location of hierarchy H1. In the set-up, this location is represented by the port of a switch where the machine (with physical address 00-1A-4D-1C-AB-49) is connected. As the data warehouse is placed in the location of Level Year, the details about it are added to the Central Look-Up Data Warehouse. The content of Central Look-Up Data Warehouse at this stage is shown in Figure 12.

The user enters the Location Name of the first seven locations from the top of hierarchy H1 via active form designed for it. Through scripting, Location ID for each entered seven Locations is obtained from Mapping Tables. The Mapping Tables are generated from the Central Look-Up Data Warehouse the content of which is shown in Figure 12. A three digit unique ID is generated for the last level in hierarchy H1. The Location ID obtained for the Location name entered by the user are shown in the bracket as 2013(13) | Uttar Pradesh (30) | UPTU (20) | KNIT (104) | CSE (10) | Faculty (5) | Professor (2) and a unique ID (001).

All the Location ID's obtained through scripting are clubbed together in hierarchal order to obtain 1330201041052001. UIN generation module uses algorithm shown in Figure 7 and calculates the check digit ($C_k = 7$). Once a check digit is calculated it is appended in the last of Location ID's clubbed together in hierarchal order, thus, a 17 digits UIN is generated (UIN = 13302010410520017). Once a UIN is generated it is returned to the user and information containing UIN and data (13302010410520017| 2013| UP| UPTU| KNIT| CSE| Faculty| Professor| ID|) is moved to GRAM.

GRAM stores the information until the most suitable data warehouse in 8-Level hierarchal structure *i.e.*, hierarchy H1, to store data, is not located. Replica Manager obtains the same information from GRAM for further processing. Replica Manager generates two replicas of this information. One replica of the information is moved to summation block in DLB, while from another replica of information UIN 13302010410520017 is extracted and moved to Identifier Finder in DLB.

Identifier Finder in DLB finds the first Identifier (1330201041052001) from the UIN using the algorithm shown in Figure 8, which is executed in DLB. The calculated Identifier is moved to Data Warehouse Locator in DLB. Corresponding to this Identifier Data Warehouse Locator looks for the Machine Address in the table of Central Look-Up Data Warehouse. As a data warehouse is placed in the location of Level Year, "No Data Warehouse is found" message is returned to Data Warehouse Locator.

The same process of generating Identifier by Identifier Finder and looking for Machine Address by Data Warehouse Locator repeats until Identifier (13) is generated. For the Identifier (13) machine address 00-1A-4D-1C-AB-49 is returned to Data Warehouse Locator. The obtained machine address is moved to LRAM along with (13302010410520017| 2013| UP| UPTU| KNIT| CSE| Faculty| Professor| ID|) via summation block.

The Validation and Verification block re-calculates check digits and compare it with the original. Both the digits come out to be same in this case which means no error has occurred in UIN. The entire information containing Machine Address 00-1A-4D-1C-AB-49, UIN 13302010410520017 and data 2013| UP| UPTU| KNIT| CSE| Faculty| Professor| ID|) is extracted by ETL.

Finally, ETL performs its defined tasks to store UIN 13302010410520017 and data

2013| UP| UPTU| KNIT| CSE| Faculty| Professor| ID in the Common table of the data warehouse present in the location of Level Year in hierarchy H1.

Year

Identifier	Location Name	Availability of DW	Machine Address
13	2013	1	00-1A-4D-1C-AB-49

State

Identifier	Location Name	Availability of DW	Machine Address
1330	Uttar Pradesh	0	NULL

University

Identifier	Location Name	Availability of DW	Machine Address
133020	UPTU	0	NULL

Institute

Identifier	Location Name	Availability of DW	Machine Address
133020104	KNIT	0	NULL

Department

Identifier	Location Name	Availability of DW	Machine Address
13302010410	CSE	0	NULL

Category

Identifier	Location Name	Availability of DW	Machine Address
133020104105	Faculty	0	NULL

Position

Identifier	Location Name	Availability of DW	Machine Address
1330201041052	Professor	0	NULL

Unique ID

Identifier	Location Name	Availability of DW	Machine Address
1330201041052001	NULL	0	NULL
1330201041052002	NULL	0	NULL
…	…	.	…
1330201041052999	NULL	0	NULL

Figure 12. The Contents in Central Look-Up Data Warehouse

7. Results

The time elapsed in the execution of the entire experiment, starting, after the data entered by the user till storing of it in the Common table of most suitable data warehouse placed at Level Year of hierarchy H1 comes out to be 7.135 milliseconds. Real data of 998 other users were entered with the same condition in same hierarchy H1 with unique ID varying from 002 to 999. Average time taken by each comes out to be 7.839 milliseconds. The average time of 7.839 milliseconds is greater than 7.135 because of the following reasons: communication delays; resending of information from GRAM due to error identified in UIN; etc. A further experiment is done by placing a data warehouse at a different level of hierarchy H1 one-by-one from the top. The same number of records *i.e.*, 999 and the same type of records were entered by the user for each case, and the following results were obtained shown in table 1.

Table 1. Average Time to Store User Data in the Common Table of the most Suitable Data Warehouse in the 8-Level Hierarchal Structure Formed for Discussed Case

Level in hierarchy	Location of Data Warehouses in hierarchy	Location Identifier	Average Time in (milliseconds)
1	Year	13	7.839
2	Uttar Pradesh	1330	7.290
3	UPTU	133020	6.819
4	KNIT	133020104	6.349
5	CSE	13302010410	5.879
6	Faculty	133020104105	5.308
7	Professor	1330201041052	4.781

The graph is plotted shown in Figure 13 for the results shown in table 1. In the graph, time is taken on y-axis while on the x-axis names of various levels are given. Time taken to store data entered by the user in data warehouse built at each level one-by-one is compared with the time it took for data to store in data warehouse placed at level Year. The graph also shows that the average time needed to store data, decreases, as a data warehouse is placed at lower levels in 8-Level hierarchal structure.

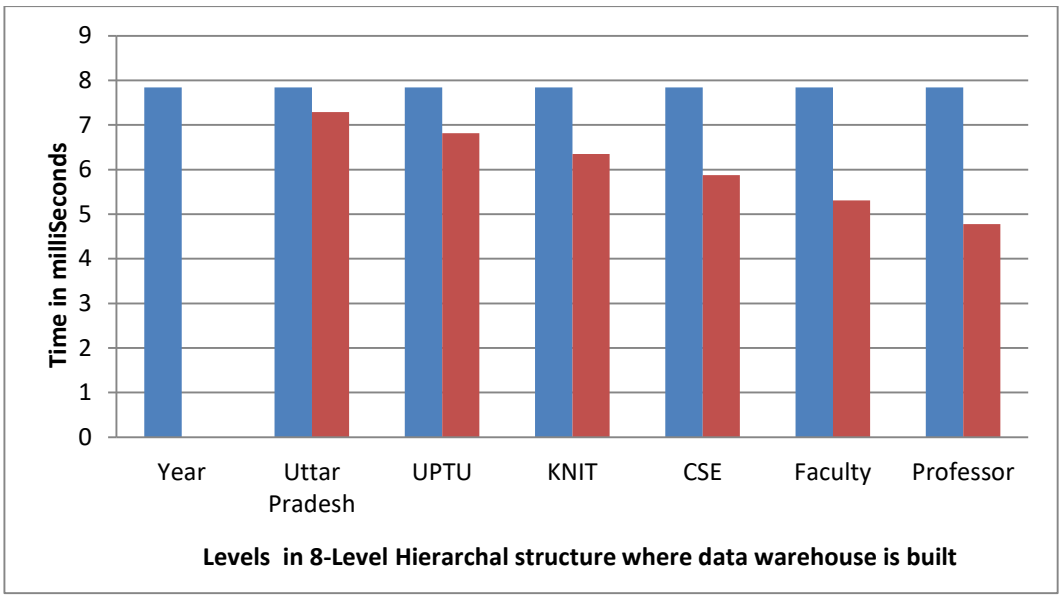

Figure 13. Average Time to Storeuser Entered Data in the Common Table of the Most Suitable Data Warehouse in the 8-Level Hierarchal Structure Formed for Discussed Case

8. Conclusion

It has seen that proposed novel approach, for locating a most suitable data warehouse in IDDW to store data, works well. The same is logically as well as experimentally verified. It is experimentally verified by the design, development and implementation of the case, presented. It is also seen that entered data by user gets stored correctly in the most suitable Data Warehouse in IDDW. It gets stored by locating the Data Warehouse intelligently with the help of UIN. It is also noted from the experiment that the average time taken for data to get stored is reduced from 7.839 milliseconds to 4.781 milliseconds when a Data Warehouse is made available at Level 7 as compared to the time taken when

a Data Warehouse is available at Level 1 of the IDDW.

References

[1] A. K. Agarwal and N. Badal, "A Novel Approach for Intelligent Distribution of Data Warehouses", Egyptian Informatics Journal, Cairo, Egypt, vol. 17, **(2016)**, pp. 147-159.
[2] J. Bernardino and H. Madeira, "Experimental Evaluation of a New Distributed Partitioning Technique for Data Warehouses", International Database Engineering and Applications Symposium, **(2001)**, pp. 312-321.
[3] N. Chaki and B. B. Sarkar, "Virtual Data Warehouse Modelling Using Petri Nets for Distributed Decision Making", Journal of Convergence Information Technology, vol. 5, no. 5, **(2010)** July, pp. 8-21.
[4] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury and S. Tuecke, "The Data Grid: Towards Architecture for the Distributed Management and Analysis of Large Scientific Datasets", Journal of Network and Computer Application, vol. 23, no. 3, **(2000)**, pp. 187-200.
[5] R. Costa and P. Furtado, "Data Warehouses in Grids with High QoS", DaWaK, Vol. 4081 of Lecture Notes in Computer Science, Springer, **(2006)**, pp. 207-217.
[6] W. H. Inmon, "Building the Data Warehouse", John Wiley & Sons, **(1993)**.
[7] T. Karima, A. Abdelatif and O. Habib, "Data warehouse decentralization strategy", IEEE International Conference on E- business Engineering, **(2010)**, pp. 72-76.
[8] M. Lawrence and A. Rau-Chaplin, "The OLAP-Enabled Grid: Model and Query Processing Algorithms", Proceedings of the 20th International Symposium on High Performance Computing Systems and Applications (HPCS'06), Canada, **(2006)** May.
[9] A. Y. Noaman and K. Barker, "A Horizontal Fragmentation Algorithm for Fact Relation in a Distributed Data Warehouse", Proceedings of the eight International Conference on information and knowledge management (CIKM'99), **(1999)** November, pp. 154-161.
[10] A. Y. Noaman and K. Barker, "Distributed data warehouse architectures", Journal of data warehousing, vol. 2, no. 2, **(1997)** April, pp. 37-50.
[11] "Sequential access memory and Random access memory", http://www.ustudy.in/node/7611, **(2008)**.
[12] H. Su, C. Tang, S. Qiao, C. Li, T. Zhang and S. Dai, "Construct Distributed RFID Data Warehouse Based on Concept Hierarchy", IEEE Internnational Workshop on Anti-Counterfeiting, Security, Identification, Xiamen, Fujian, **(2007)** April, pp. 461-463.
[13] P. Wehrle, M. Miquel and A. Tchounikine, "A Model for Distributing and Querying a Data Warehouse on a Computing Grid", 11th International Conference on Parallel and Distributed Systems(ICPADS'05), vol. 1, **(2005)** July, pp. 203- 209.
[14] P. Wehrle, M. Miquel and A. Tchounikine, "A Grid Services-Oriented Architecture for Efficient Operation of Distributed Data Warehouses on Globus", 21st International Conference on Advanced Networking and Applications (AINA '07), Canada, **(2007)** May, pp. 994-999.
[15] C. White, "A Technical Architecture for Data Warehousing", Info DB Journal, vol. 9, no. 1, **(1995)** February, pp. 5-11.
[16] S. Zhou, A. Zhou, X. Tao and Y. Hu, "Hierarchically Distributed Data Warehouse", IEEE Journal, **(2000)** May, pp. 848-853.
[17] J. Zhao and K.-D. Schewe, "Using Abstract State Machines for Distributed Data Warehouse Design", In Conceptual Modelling 2004 - First Asia-Pacific Conference on Conceptual Modelling (Eds. S. Hartmann and J. Roddick), Dunedin, New Zealand, Vol. 31 of CRPIT, **(2004)**, pp. 49-58.
[18] http://support.alphasoftware.com/WhatsNewInV9/generated_NewInA5V9_Videos.htm dated: 15/02/14.

