ADSG TOOL: DEVELOPING AN AUTOMATIC DATABASE SCHEME GENERATION BY INTEGRATING MULTIPLE DATABASES' INTO A SINGLE ROOF

SANNAPUREDDY MANI BHASKAR REDDY [#1], **D.BHANU PRAKASH** [#2], **SANTOSH KUMAR SHARMA** [#3]

[#1] MCA Scholar,Master of Computer Applications,
Vignan Institute of Information Technology, Duvvada, Visakhapatnam, AP, India.

[#2]Assistant Professor ,Department of CSE,
Vignan Institute of Information Technology, Duvvada, Visakhapatnam, AP, India.

[#3] Assistant Professor , Master of Computer Applications,
Vignan Institute of Information Technology, Duvvada, Visakhapatnam, AP, India.

ABSTRACT

Now a day's almost each and every small or large scale companies need database for storing their valuable data inside their storage locations. As there were many database's that are available in the market with individual advantages and a separate functionalities. But all the databases are almost used in command line interfaces rather than in graphical user interfaces. If all the databases are run on a GUI, then it will be very easy for accessing, inserting, retrieving the data to and from the database. Along with this advantage one more limitation that was currently faced by almost all software companies is accessing various database systems need knowledge on each and every database separately i.e. in terms of syntax and command block structure. In this proposed thesis, we developed an automatic database scheme generation tool that allows developers to access multiple databases under a single roof in a GUI model. Here in our proposed tool we try to integrate the functionality of three different databases like MYSQL, ORACLE and MS-ACCESS database. If this tool was developed and deployed successfully in all the software companies the user can focus much on application development rather than on the database designing aspects. As we are using this ADSG tool for database designing all the table creation, modification, insertion, deletion, rename and a lot more tasks can be easily done without the need of learning each and every query separately for individual database. This is a common tool for multiple databases and hence our proposed tool is best suited for application developers to easily create a table structure and map the fields easily according to the front end page which was designed in JSP.By conducting various experiments on our proposed system we finally came to an conclusion that this ADSG tool provides a rich platform for the developers who wish to migrate from one database to other ,as this single tool can give support for multiple databases under one roof.

Key Words: Automatic Database Scheme Generation, Database Design Aspects, Application Developers, Altering Schema, Creating Database, Multiple Databases, Java Server Pages.

I. INTRODUCTION

A database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views, and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

A database management system (DBMS) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include

MySQL,

PostgreSQL,

MongoDB,

MariaDB,

Microsoft SQL Server,

Oracle,

MS-Access

Sybase,

SAP HANA,

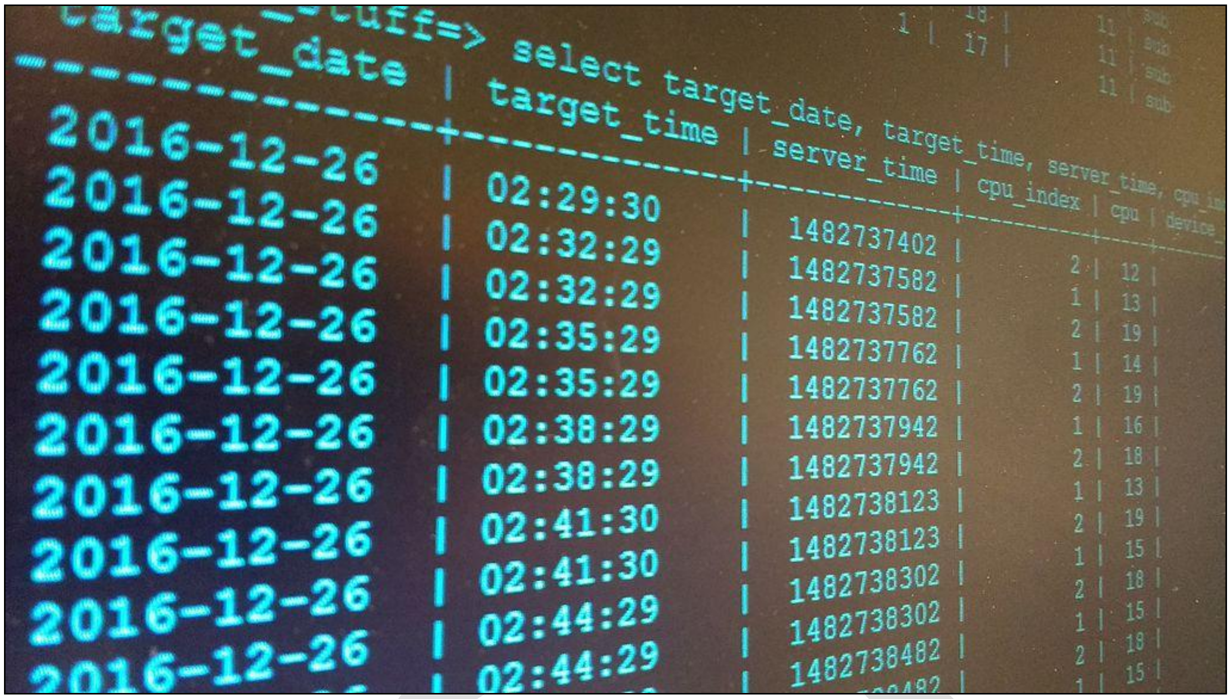
MemSQL

and IBM DB2.

A database is not generally portable across different DBMSs, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one DBMS. Database management systems are often classified according to the database model that they support; the most popular database systems since the 1980s have all supported the relational model as represented by the SQL language. Sometimes a DBMS is loosely referred to as a 'database'[1],[2].

APPLICATIONS OF DATABASE

Databases are used to support internal operations of organizations and to underpin online interactions with customers and suppliers (see Enterprise software).Databases are used to hold administrative information and more specialized data, such as engineering data or economic models. Examples of database applications include computerized library systems, flight reservation systems, computerized parts inventory systems, and many content management systems that store websites as collections of webpage's in a database.



```
                                   1 |  18 |          11 | sub
                                     |  17 |          11 | sub
                                                      11 | sub
                                                      11 | sub
tuff=> select target_date, target_time, server_time, cpu_in
target_date | target_time | server_time | cpu_index | cpu | device
------------+-------------+-------------+-----------+-----+-------
2016-12-26  | 02:29:30    | 1482737402  |         2 |  12 |
2016-12-26  | 02:32:29    | 1482737582  |         1 |  13 |
2016-12-26  | 02:32:29    | 1482737582  |         2 |  19 |
2016-12-26  | 02:35:29    | 1482737762  |         1 |  14 |
2016-12-26  | 02:35:29    | 1482737762  |         2 |  19 |
2016-12-26  | 02:38:29    | 1482737942  |         1 |  16 |
2016-12-26  | 02:38:29    | 1482737942  |         2 |  18 |
2016-12-26  | 02:41:30    | 1482738123  |         1 |  13 |
2016-12-26  | 02:41:30    | 1482738123  |         2 |  19 |
2016-12-26  | 02:44:29    | 1482738302  |         1 |  15 |
2016-12-26  | 02:44:29    | 1482738302  |         2 |  18 |
2016-12-26  | 02:44:29    | 1482738482  |         1 |  15 |
                                        |         2 |  18 |
                                        |         1 |  15 |
```

FIGURE. 1 REPRESENTS AN EXAMPLE OF DATA BEING RETURNED AFTER A SUCCESSFUL QUERY IN POSTGRES SQL.

From the above figure 1, we can clearly get an idea that SQL query is written on a Command Prompt for getting the Target_Date, Target_Time, Sever _Time and a lot more fields from a user created table. This query is written on a SQL command prompt by suing PostGres SQL.Once if the query was correctly executed then we can able to find out the resultant data as a result.

Existing DBMSs provide various functions that allow management of a database and its data which can be classified into four main functional groups:

1. **Data Definition –** Creation, modification and removal of definitions that define the organization of the data.
2. **Update –** Insertion, modification, and deletion of the actual data [3].
3. **Retrieval –** Providing information in a form directly usable or for further processing by other applications. The retrieved data may be made available in a form basically the same as it is stored in the database or in a new form obtained by altering or combining existing data from the database [4].
4. **Administration –** Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information that has been corrupted by some event such as an unexpected system failure [5].

Both a database and its DBMS conform to the principles of a particular database model [6]. "Database system" refers collectively to the database model, database management system, and database [7].

Physically, database servers are dedicated computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with generous memory and RAID disk arrays used for stable storage. RAID is used for recovery of data if any of the disks fail. Hardware database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. DBMSs are found at the heart of most database applications. DBMSs may be built around a custom multitasking kernel with built-in networking support, but modern DBMSs typically rely on a standard operating system to provide these functions. Since DBMSs comprise a significant market, computer and storage vendors often take into account DBMS requirements in their own development plans [8]. Databases and DBMSs can be categorized according to the database model(s) that they support (such as relational or XML), the type(s) of computer they run on (from a server cluster to a mobile phone), the query language(s) used to access the database (such as SQL or XQuery), and their internal engineering, which affects performance, scalability, resilience, and security.

II. RELATED WORK

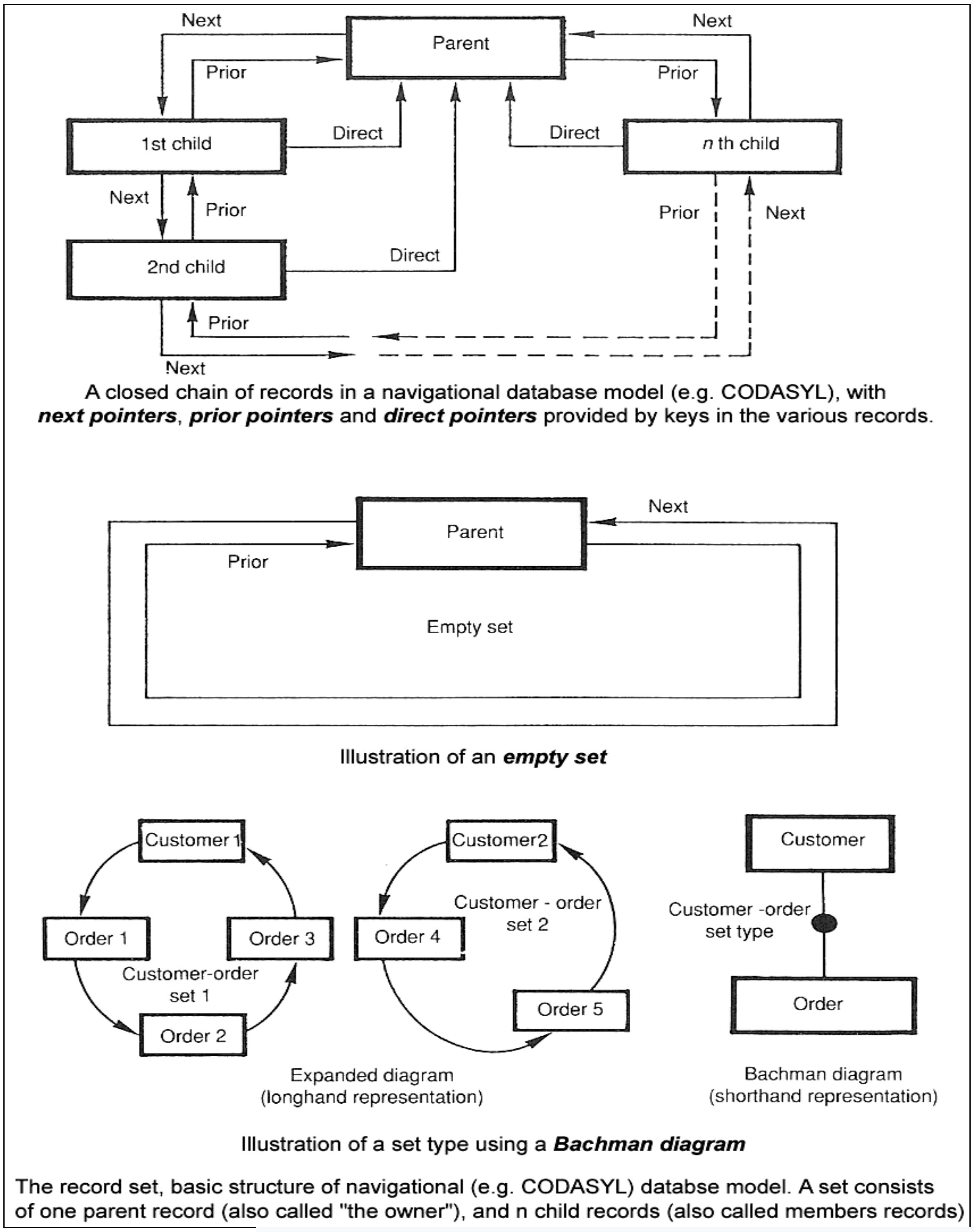
In this section we mainly discuss about the related work that was carried out in order to design this proposed ADSG tool for integrating multiple databases under a single roof for performing various database operations. Here in this section we mainly discuss about the history of the database.

MAIN MOTIVATION

Following the technology progress in the areas of processors, computer memory, computer storage, and computer networks, the sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. The development of database technology can be divided into three eras based on data model or structure: navigational,[9] SQL/relational, and post-relational.The two main early navigational data models were the hierarchical model, epitomized by IBM's IMS system, and the CODASYL model (network model), implemented in a number of products such as IDMS.

The relational model, first proposed in 1970 by Edgar F. Codd, departed from this tradition by insisting that applications should search for data by content, rather than by following links. The relational model employs sets of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware become powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2015 they remain dominant: IBM DB2, Oracle, MySQL, and Microsoft SQL Server are the top DBMS [10]. The dominant database language, standardised SQL for the relational model, has influenced database languages for other data models. Object databases were developed in the 1980s to overcome the inconvenience of object-relational impedance mismatch, which led to the coining of the term "post-relational" and also the development of hybrid object-relational databases.

The next generation of post-relational databases in the late 2000s became known as NoSQL databases, introducing fast key-value stores and document-oriented databases. A competing "next generation" known as NewSQL databases attempted new implementations that retained the relational/SQL model while aiming to match the high performance of NoSQL compared to commercially available relational DBMSs.



A closed chain of records in a navigational database model (e.g. CODASYL), with ***next pointers***, ***prior pointers*** and ***direct pointers*** provided by keys in the various records.

Illustration of an ***empty set***

Illustration of a set type using a ***Bachman diagram***

The record set, basic structure of navigational (e.g. CODASYL) databse model. A set consists of one parent record (also called "the owner"), and n child records (also called members records)

FIGURE. 2 REPRESENT THE ILLUSTRATION OF A SET TYPE USING A BACHMAN DIAGRAM

The introduction of the term *database* coincided with the availability of direct-access storage (disks and drums) from the mid-1960s onwards. The term represented a contrast with the tape-based systems of the past, allowing shared interactive use rather than daily batch processing. The Oxford English Dictionary cites [11] a 1962 report by the System Development Corporation of California as the first to use the term "data-base" in a specific technical sense.

As computers grew in speed and capability, a number of general-purpose database systems emerged; by the mid-1960s a number of such systems had come into commercial use. Interest in a standard began to grow, and Charles Bachman, author of one such product, the Integrated Data Store (IDS), founded the "Database Task Group" within CODASYL, the group responsible for the creation and standardization of COBOL. In 1971, the Database Task Group delivered their standard, which generally became known as the "CODASYL approach", and soon a number of commercial products based on this approach entered the market.

The CODASYL approach relied on the "manual" navigation of a linked data set which was formed into a large network. Applications could find records by one of three methods:

1. Use of a primary key (known as a CALC key, typically implemented by hashing)
2. Navigating relationships (called sets) from one record to another
3. Scanning all the records in a sequential order

Later systems added B-trees to provide alternate access paths. Many CODASYL databases also added a very straightforward query language. However, in the final tally, CODASYL was very complex and required significant training and effort to produce useful applications.IBM also had their own DBMS in 1966, known as Information Management System (IMS). IMS was a development of software written for the Apollo program on the System/360. IMS was generally similar in concept to CODASYL, but used a strict hierarchy for its model of data navigation instead of CODASYL's network model. Both concepts later became known as navigational databases due to the way data was accessed, and Bachman's 1973 Award presentation was *The Programmer as Navigator*. IMS is classified as a hierarchical database. IDMS and Cincom Systems' TOTAL database are classified as network databases. IMS remains in use as of 2014 [12].

III. PROPOSED AUTOMATIC DATABASE SCHEME GENERATION TOOL BY INTEGRATING VARIOUS DATABASES UNDER SINGLE ROOF

In this section we will mainly discuss about the proposed ADSG Tool for integrating the various databases under a single roof. In this section we mainly discuss about the normalization algorithm first and then discuss about other techniques that were used in our proposed tool.

NORMALIZATION ALGORITHM

This algorithm takes input as a Head pointer of linked list, which stores a relation in 1NF, in computers memory in a linked list format as discussed above. Second input is a Flag3NF. Database designer will provide value of flag Flag3NF, if designer want to normalize this relation up to 3NF, one will set this flag.

For normalizing this relation in 2NF, designer will reset this flag. During the process of normalization, in step 2 it creates table structures , which are nothing but array of strings and then these table structures are used to create actual tables in Oracle. This algorithm internally uses another algorithm called AttributeInfo, that provides PrimeAttributes[ ], AllAttributes[ ] and PrimeKeyNodeIds[ ] , which are used by remaning part of the algorithm.

Algorithm_Normalization(Head, Flag3NF)

```
 {
Input: - Head pointer of linked list holding all the attributes and functional dependencies of a
         relation to be normalized.
         A flag named Flag3NF, which is set to 1 if user wants to normalize up to 3NF
         otherwise normalization will be done up to 2NF only.
Output: - If flag3NF=1 Tables created in 3NF in Oracle, else Tables created in 2NF in
          Oracle.
        Let A1, A2and A3 be the string arrays used to hold the set of related attributes taking
participation in full FD, partial FD and transitive dependencies (TD), respectively.
        A2 and A3 are divided into two components namely determiner and dependent, for
storing determiner and dependent attributes participating in a given type of dependency.
         A2 has two components as A2-dependent[] and A2-determiner[] used for storing
dependent and determiner attributes, respectively, participating in a partial FD.
        Similarly A3 will have two components A3-determiner[] and A3-dependent[], used
              for storing determiner and dependent attributes, respectively, participating in TD.
Let Listptr and Trav are pointer variables of type structure node. 1.
Calculate number of prime attributes and store attributes taking participation in different types of
functional dependencies in string arrays A1, A2 and A3.
Set listptr=Head; /*Here Head is a pointer variable pointing to first node of linked list.
Call
{
PrimeKeyNodeIds[ ],
PrimeAttributes[ ],
AllAttributs[ ]}=AttributeInfo (listptr)
/* it returns total no of prime attributes in KeyCount. /*
After execution of this algorithm we will get node_ids of all the prime /*attributes in array
primeKeyNodeId[] and their attribute names in array /*
PrimeAttributes[] and list of all attributes in array AllAttributes[]
 For (each non- key attribute)
do the following
{
1a. Initialization.
Set Flag1=0
Flag2=0,
Flag3=0;
 index1=1,
index2=1;
index3=1.
 /* index1, index2 and index3 are used for indexing of array A1, A2 and A3, /*
respectively. A2-determiner[] array is used to store determiners and
```

/* A2-dependant[] stores dependant attributes participating in Partial FD. /*

Flag1, Flag2 and Flag3 are set for Full, Partial and transitive dependency, /* respectively.
1b. Finding non-key attributes and their determiners for finding each type of dependency holds on this relation by traversing its linked list.

```
Node * trav; Trav = Head;
while(Trav ptrTonext ≠ NULL)
If ( Trav  keyAttribute = 0)
Then Find the determiner_ id[] of Trav
```

/* where determiner_id[] is an array of node-ids of all International Journal of Database Management Systems (IJDMS), Vol.3, No.1, February 2011 143 /*

the determiner attributes

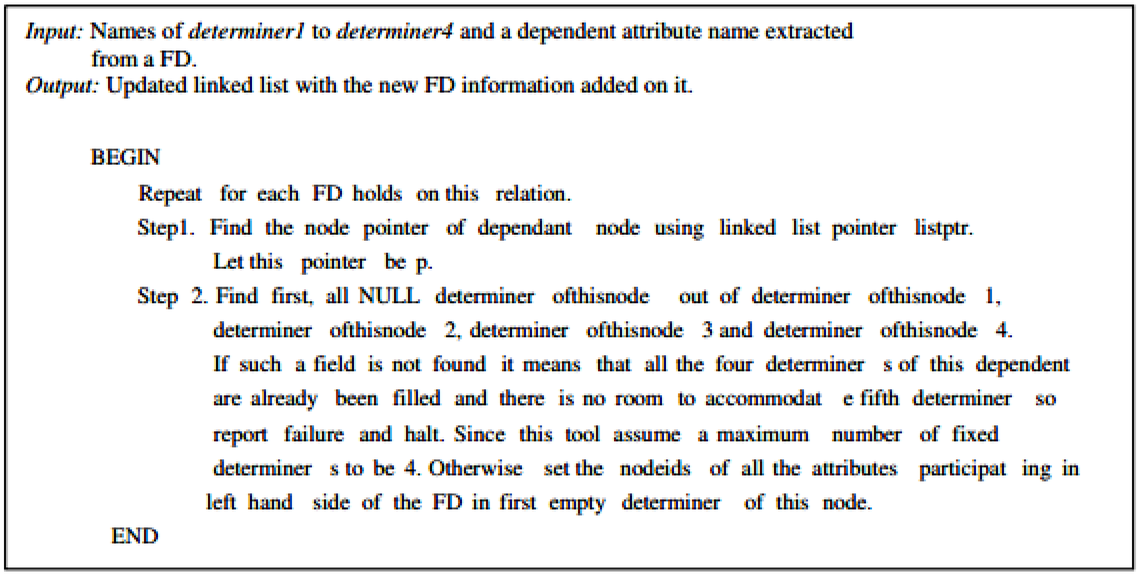
```
If (determiner_id[] of Trav == primeKeyNodeId[])
Then Set Flag1=1;
/* Full FD exists /* two arrays are equal if they have exactly same elements /*
may be ordered in different sequence.
End If (determiner_id [Trav] ⊂ primeKeyNodeId[])
/*means partial FD exists Then Set Flag2=1 End /*
 where ⊂ is proper subset operator
 If (determiner_id [Trav] ∉primeKeyNodeId[])
/* where ∉ is does not belong to operator /*
means Transitive FD exists

Then Set Flag3=1 End 1c.
```

Storing attributes participating in full functional dependencies in A1.

```
 If (Flag1==1) /* means full functional dependency exists */
Then /*save attribute pointed by Trav in array A1,
 as A (1 index )1 = A (1 index )1 ∪ (Trav → attribute _ name) ∪ PrimeAttributes[]
End
```

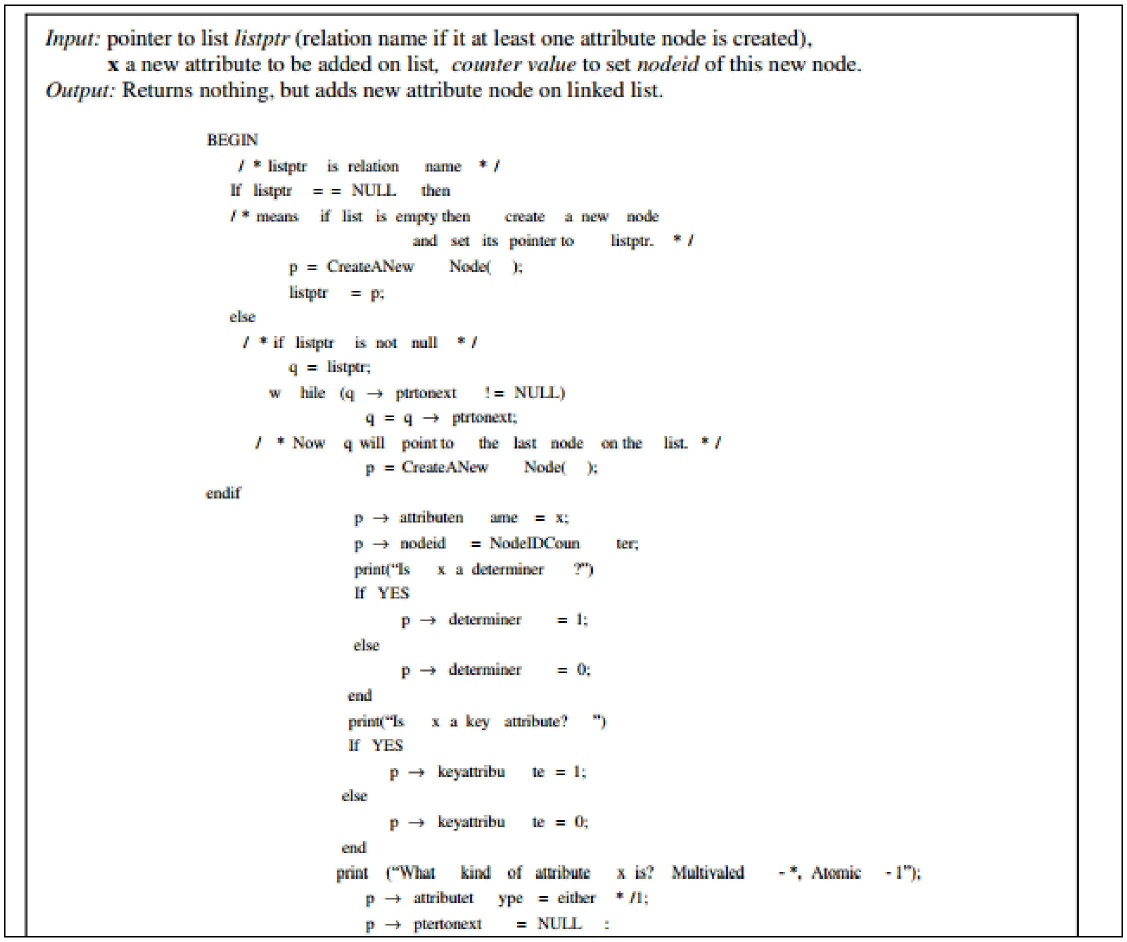
/* note that A1 will always have only one entry 1d. Storing attributes participating in partial functional dependencies in A2-dependant and A2-determiner. If (Flag2==1) /* means partial dependency exist */
then
/*save attributes pointed by Trav and all its determiner attributes in arrays /*

Algorithm AddNewAttribute (listptr, x, NodeIDCounter) This algorithm adds a new attribute node with attribute name x on linked list using a nodeid= NodeIDCounter value. Name of the relation is used as listptr, which points to the first node on that linked list. If listptr=NULL means list is empty we need to create first node for that relation. It uses function CreateANewNode( ), which creates a new node and returns its link. This algorithm uses two variable pointers p and q. This algorithm is described in above.



Input: Names of *determiner1* to *determiner4* and a dependent attribute name extracted from a FD.
Output: Updated linked list with the new FD information added on it.

```
BEGIN
    Repeat for each FD holds on this relation.
    Step1. Find the node pointer of dependant node using linked list pointer listptr.
           Let this pointer be p.
    Step 2. Find first, all NULL determiner ofthisnode out of determiner ofthisnode 1,
           determiner ofthisnode 2, determiner ofthisnode 3 and determiner ofthisnode 4.
           If such a field is not found it means that all the four determiner s of this dependent
           are already been filled and there is no room to accommodat e fifth determiner so
           report failure and halt. Since this tool assume a maximum number of fixed
           determiner s to be 4. Otherwise set the nodeids of all the attributes participat ing in
           left hand side of the FD in first empty determiner of this node.
END
```



Input: pointer to list *listptr* (relation name if it at least one attribute node is created),
x a new attribute to be added on list, *counter value* to set *nodeid* of this new node.
Output: Returns nothing, but adds new attribute node on linked list.

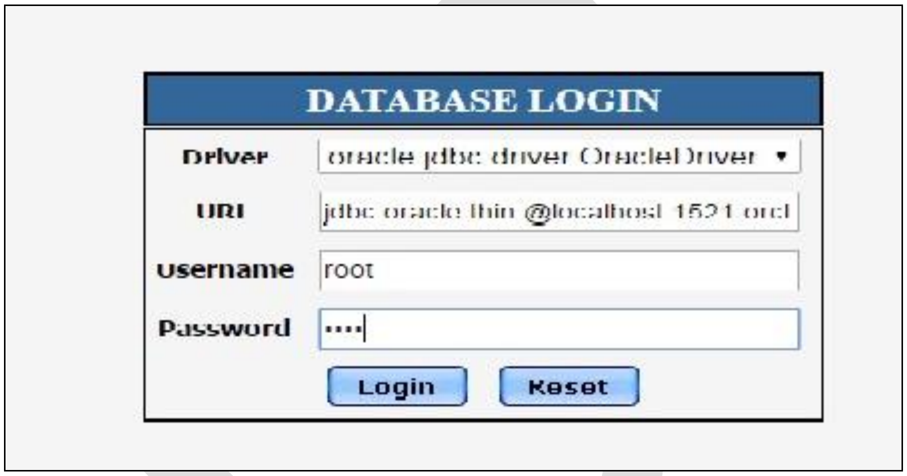
```
BEGIN
   / * listptr is relation name * /
  If listptr = = NULL then
  / * means if list is empty then create a new node
                     and set its pointer to listptr. * /
        p = CreateANew Node( );
        listptr = p;
  else
   / * if listptr is not null * /
        q = listptr;
      w hile (q → ptrtonext != NULL)
              q = q → ptrtonext;
    / * Now q will point to the last node on the list. * /
              p = CreateANew Node( );
endif
             p → attributen ame = x;
             p → nodeid = NodeIDCoun ter;
             print("Is x a determiner ?")
             If YES
                 p → determiner = 1;
             else
                 p → determiner = 0;
             end
             print("Is x a key attribute? ")
             If YES
                 p → keyattribu te = 1;
             else
                 p → keyattribu te = 0;
             end
            print ("What kind of attribute x is? Multivaled - *, Atomic - 1");
              p → attributet ype = either * /1;
              p → ptertonext = NULL :
```

FIGURE. 3 REPRESENT THE ALGORITHM FOR ADDING A NEW FD IN A RELATIONAL LINKED LIST

IV. RESULT ANALYSIS

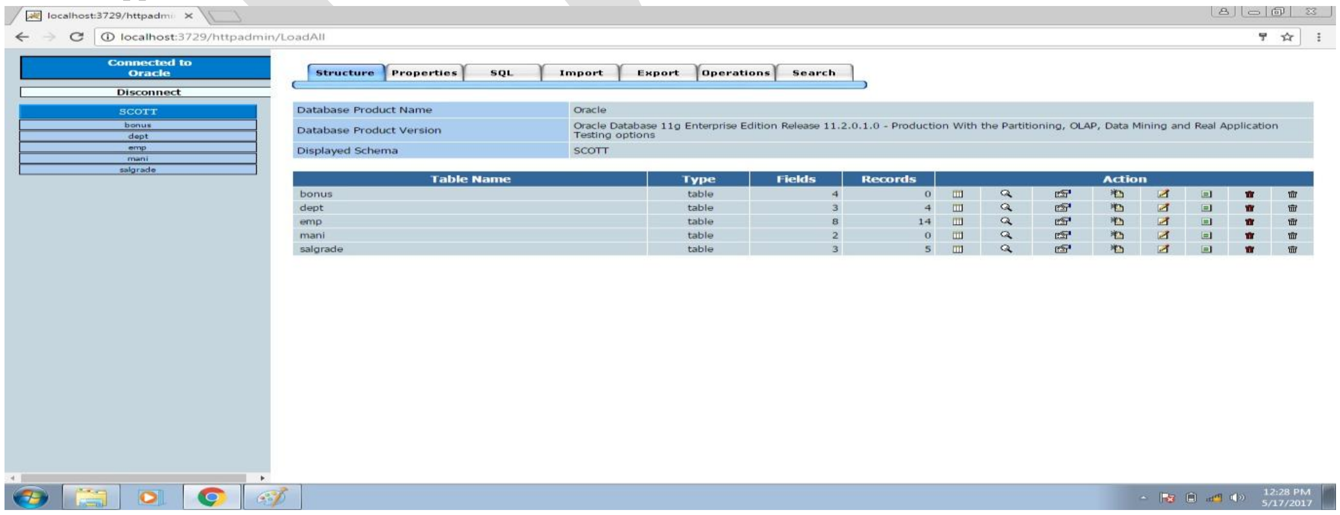
In this section we will mainly discuss about the proposed ADSG TOOL method which is mainly used for integrating multiple data bases on a single roof and provide a rich facility to schema generation in a GUI model. Here we integrated three types of databases like MY-SQL,ORACLE,MS-ACCESS as the main data bases for accessing the application. As we implemented the whole application using JEE as main platform, we took JSP and HTML as front end technologies for designing the user interfaces of our proposed application and we take MY-SQL,ORACLE,MS-ACCESS as the back end database for storing and retrieving the valuable information to and fro from the database.

SAMPLE WINDOW THAT REPRESENTS THE LOGIN INTO THE ADSG TOOL


DATABASE LOGIN

Driver	oracle.jdbc.driver.OracleDriver
URL	jdbc:oracle:thin:@localhost:1521:orcl
username	root
Password	••••

From the above window we can clearly find out the user try to enter into this tool with the help of the valid id and password that was assigned for the oracle database at the time of registration.Here if the user enters all the valid id and password then only he can enter into the system,if not he cant able to enter into the application.


localhost:3729/httpadmin/LoadAll

Database Product Name	Oracle
Database Product Version	Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production With the Partitioning, OLAP, Data Mining and Real Application Testing options
Displayed Schema	SCOTT

Table Name	Type	Fields	Records
bonus	table	4	0
dept	table	3	4
emp	table	8	14
mani	table	2	0
salgrade	table	3	5

From the above window we can clearly identify that user after substituting all the valid details,they can enter into the main window and they can able to see the details like login date and time along with number of tables that are available in the database and so on.

V. CONCLUSION

In this paper we for the first time have implemented a new tool like ADSG for integrating multiple databases under a single roof and perform the all the database operations in a GUI mode. The entire project has been developed and deployed as per the requirements stated by the user, it is found to be bug free as per the testing standards that are implemented. Any specification untraced errors will be concentrated in the coming versions, which are planned to be developed in near future. The system at present does not take care of the money payment methods, as the consolidated constructs need SSL standards and are critically to be initiated in the first face, the application of the credit card transactions is applied as a developmental phase in the coming days. The system needs more elaborative technicality for its inception and evolution.

VI. REFERENCES

[1] Bachman, Charles W. (1973). "The Programmer as Navigator". Communications of the ACM. 16 (11): 653–658. doi:10.1145/355611.362534. (Subscription required (help)).

[2] Beynon–Davies, Paul (2003). Database Systems (3rd ed.). Palgrave Macmillan. ISBN 978-1403916013.

[3] "Update – Definition of update by Merriam-Webster". merriam-webster.com.

[4] "Retrieval – Definition of retrieval by Merriam-Webster". merriam-webster.com.

[5] "Administration – Definition of administration by Merriam-Webster". merriam-webster.com.

[6] Tsitchizris & Lochovsky 1982.

[7] Beynon–Davies 2003.

[8] Graves, Steve. "COTS Databases For Embedded Systems", Embedded Computing Design magazine, January 2007. Retrieved on August 13, 2008.

[9] Argumentation in Artificial Intelligence by Iyad Rahwan, Guillermo R. Simari

[10] Graves, Steve. "COTS Databases For Embedded Systems", Embedded Computing Design magazine, January 2007. Retrieved on August 13, 2008.

[11] Argumentation in Artificial Intelligence by Iyad Rahwan, Guillermo R. Simari

[12] Proctor, Seth (12 July 2013). "Exploring the Architecture of the NuoDB Database, Part 1". Archived from the original on 15 July 2013. Retrieved 12 July 2013.

[13] Tsitchizris, Dionysios C.; Lochovsky, Fred H. (1982). Data Models. Prentice–Hall. ISBN 978-0131964280.

[14] Ullman, Jeffrey; Widom, Jennifer (1997). A First Course in Database Systems. Prentice–Hall. ISBN 0138613370.

VII. ABOUT THE AUTHORS

SANNAPUREDDY MANI BHASKAR REDDY is currently pursuing his 3 Years MCA in Department of Computer Applications at Vignan Institute of Information Technology, Duvvada, Visakhapatnam, AP, India. His area of interests includes Networks.

D.BHANU PRAKASH is currently working as an Assistant Professor, in Department of CSE at Vignan Institute of Information Technology, Duvvada, Visakhapatnam, AP, India. He received M.Tech from Acharya Nagarjuna University. He has more than 9 years of experience in teaching field. His research interest includes Computer Networks, Security and Operating System.

SANTOSH KUMAR SHARMA is currently working as an Assistant Professor, in Department of Computer Applications at Vignan Institute of Information Technology, Duvvada, Visakhapatnam, AP, India. He received M.Tech from Berhampur University-ODISHA and now he is pursuing (PhD) from Berhampur University. He has more than 9 years of experience in teaching field. His research interest includes Computer Networks, Security, Data mining, Mobile Computing and Wireless Computing, Distributed Operating System.