Journal of Software Engineering and Applications, 2023, 16, 561-585
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Use of Component Integration Services in Multidatabase Systems: A Feasible Solution for Integrating Academic Institutions or Commercial Industries

Mohammad Ghulam Ali[1*], **Mohammad Ghulam Murtuza**[2]

[1]Vinod Gupta School of Management, Indian Institute of Technology Kharagpur, Kharagpur, India
[2]Research Service Center, Post Graduate Department of Physics, Tilka Manjhi Bhagalpur University, Bhagalpur, India
Email: *ali@hijli.iitkgp.ac.in, *ali_iit@yahoo.com

How to cite this paper: Ali, M.G. and Murtuza, M.G. (2023) Use of Component Integration Services in Multidatabase Systems: A Feasible Solution for Integrating Academic Institutions or Commercial Industries. *Journal of Software Engineering and Applications*, **16**, 561-585.
https://doi.org/10.4236/jsea.2023.1611029

Received: October 16, 2023
Accepted: November 20, 2023
Published: November 23, 2023

Abstract

The book chapter is an extended version of the research paper entitled "Use of Component Integration Services in Multidatabase Systems", which is presented and published by the 13th ISITA, the National Conference of Recent Trends in Mathematical and Computer Sciences, T.M.B. University, Bhagalpur, India, January 3-4, 2015. Information is widely distributed across many remote, distributed, and autonomous databases (local component databases) in heterogeneous formats. The integration of heterogeneous remote databases is a difficult task, and it has already been addressed by several projects to certain extents. In this chapter, we have discussed how to integrate heterogeneous distributed local relational databases because of their simplicity, excellent security, performance, power, flexibility, data independence, support for new hardware technologies, and spread across the globe. We have also discussed how to constitute a global conceptual schema in the multidatabase system using Sybase Adaptive Server Enterprise's Component Integration Services (CIS) and OmniConnect. This is feasible for higher education institutions and commercial industries as well. Considering the higher educational institutions, the CIS will improve IT integration for educational institutions with their subsidiaries or with other institutions within the country and abroad in terms of educational management, teaching, learning, and research, including promoting international students' academic integration, collaboration, and governance. This will prove an innovative strategy to support the modernization and large expansion of academic institutions. This will be considered IT-institutional alignment within a higher education context. This will also support achieving one of the sustainable development goals set by the United Nations: "*Goal* 4: *ensure inclusive and quality education for all and promote* *lifelong learning*". However, the process of IT integration into higher educational institutions must be thoroughly evaluated, identifying the vital data access points. In this chapter, Section 1 provides an introduction, including the evolution of various database systems, data models, and the emergence of multidatabase systems and their importance. Section 2 discusses component integration services (CIS), OmniConnect and considering heterogeneous relational distributed local databases from the perspective of academics, Section 3 discusses the Sybase Adaptive Server Enterprise (ASE), Section 4 discusses the role of component integration services and OmniConnect of Sybase ASE under the Multidatabase System, Section 5 shows the database architectural framework, Section 6 provides an implementation overview of the global conceptual schema in the multidatabase system, Section 7 discusses query processing in the CIS, and finally, Section 8 concludes the chapter. The chapter will help our students a lot, as we have discussed well the evolution of databases and data models and the emergence of multidatabases. Since some additional useful information is cited, the source of information for each citation is properly mentioned in the references column.

Keywords

Relational Database, Component Integration Services, OmniConnect, Multidatabase, Global Conceptual Schema, Distributed Database, Local Conceptual Schema, Database Integration, IT Integration, Higher Education, Commercial Industries, HEIs

1. Introduction

In this section, we have discussed in detail the evolution of various database models, beginning with the flat file system, based on some published documents in a sequence manner. This will also help students in their academic learnings. We have also discussed well about the background and current situation of multi-database systems in Section 1.1.5 below.

Database and Database Management Systems in Brief

Database

We explain the data hierarchy. A bit (Character)—a bit is the smallest unit of data representation (value of a bit may be a 0 or 1). Eight bits make a byte which can represent a character or a special symbol in a character code. Field—a field consists of a grouping of characters. A data field represents an attribute (a characteristic or quality) of some entity (object, person, place, or event). Record—a record represents a collection of attributes that describe a real-world entity. A record consists of fields, with each field describing an attribute of the entity. File—a group of related records. Files are frequently classified by the application for which they are primarily used (employee file). A primary key in a file is the field (or fields) whose value identifies a record among others in a data file Database—is an integrated collection of logically related records or files. A database consolidates records previously stored in separate files into a common pool of data records that provides data for many applications. The data is managed by systems software called database management systems (DBMS). The data stored in a database is independent of the application programs using it and of the types of secondary storage devices on which it is stored [1]. Please also see the details of Database Concepts in [2].

Alternatively, we can say, a database is a self-describing collection of integrated records. A record is a representation of some physical or conceptual object. A database is self-describing in that it contains a description of its own structure. This description is called metadata—data about the data. The database is integrated in that it includes the relationships among data items, as well as including the data items themselves.

Database Management Systems

A database management system (DBMS) is a collection of software programs that gives a user the interaction ability to store, modify and extract data from a certain database. It enables the definition, creation, query, update and administration of databases [3].

1.1. Evolution of Database

Database management systems did not come into industries until 1960's. The first commercial database system information management systems (IMS) appeared right before 1970. These systems evolved from file systems [4].

Originally, databases were flat. This means that the information was stored in one long text file, called a tab-delimited file. Each entry in the tab-delimited file is separated by a special character, such as a vertical bar (|). Each entry contains multiple pieces of data in the form of information about a particular object or person, grouped together as a record [5].

The text file makes it difficult to search for specific information or to create reports that include only certain fields from each record. File management systems, also called FMSs in short, are ones in which all data is stored on a single large file.

Researchers in the database field, however, found the data has its value and models based only on data should be introduced to improve reliability, security, and efficiency of access and to overcome the drawbacks of file-based systems; this led to the introduction of the hierarchical data model and hierarchical database management systems. Data models provide a way in which the stored data is organized according to a specified structure or relationship for quick access and efficient management. Data models formally define data elements and relationships among data elements for a particular domain of interest.

A data model instance may be one of three kinds, according to ANSI/SPARC (the American National Standards Institute, Standards Planning and Requirements Committee) in 1975: 1) logical schema or conceptual schema (conceptual level); 2) physical schema (internal level); and 3) external schema (external level)

[6] [7] [8]. We have also discussed below in detail different generational data models, database management systems, and ANSI/SPARC schema layers.

1.1.1. 1st Generation Data Model and Database Management Systems (DBMSs)

Hierarchical and network data models are considered 1st generation data models. Hierarchical data models and hierarchical database management systems existed from mid-1960 to early 1980. The previous system drawback of accessing records and sorting records, which took a long time, was removed by the introduction of parent-child relationships between records in the database. Each parent item can have multiple children, but each child item can have one and only one parent. Thus, relationships in the hierarchical database are either one-to-one or one-to-many.

Many-to-many relationships are not allowed.

The origin of the data is called the root, from which several branches have data at different levels, and the last level is called the leaf. Hierarchical databases are generally large databases with large amounts of data [9].

The hierarchical data model organizes data in a tree structure, or, as some call it, an "inverted" tree. There is a hierarchy of parent and child data segments. A prominent hierarchical database model was IBM's first DBMS, called IMS. In the mid-1960s, Rockwell partnered with IBM to create the Information Management System (IMS). IMS DB/DC led to the mainframe database market in the 1970s and early 1980s.

In order to avoid all the drawbacks of the hierarchical data model, a new data model and database management system took root, which is called the network model and network database management system. The Network Data Model and Network Database Management System were introduced in 1969, almost the exact opposite of the Hierarchical Data Model. In this model, the main concept of many-many relationships was introduced. The network data model eliminates redundancy at the expense of more complicated relationships. This model can be better than the hierarchical model for some kinds of data storage tasks, but worse for others. Neither one is consistently superior to the other. A network model is called CODASYL (Conference on Data System Language).

1.1.2. 2nd Generation Data Model (Relational Model) and Relational Database Management Systems (RDBMs)—1970

The Relational Data Model is classified as a 2nd Generation Data Model. In order to overcome all the drawbacks of the previous data models and database management systems, the relational data model and the relational database management system were introduced in 1970, in which data gets organized as tables and each record forms a row with many fields or attributes in it. Relationships between the tables are also formed. A tuple or row contains all the data of a single instance of the table. In the relational data model, every tuple must have a unique identification or key based on the data that uniquely identifies each tuple in the relation. Often, keys are used to join data from two or more relations based on matching identification. The relational model also includes concepts such as foreign keys, which are primary keys in one relation that are kept in another relation to allow for the joining of data. In 1970, Dr. Edgar F. Codd at IBM published the Relational Data Model. A relational database is a collection of relations, or tables. By definition, a relation is a set of tuples having the same attributes. Operations that can be performed on relationships are select, project, and join. The join operation combines relations; the select queries are used for data retrieval; and the project operation identifies attributes. Similar to other database models, even relational databases support insert, delete, and update operations.

Basically, relational databases are based on relational set theory. Normalization is a vital component of the relational database model. Relational operations, which are supported by relational databases, work best with normalized tables. A relational database supports relational algebra and relational calculus, consequently supporting the relational operations of set theory. Apart from mathematical set operations, namely, union, intersection, difference, and Cartesian product, relational databases also support select, project, relational join, and division operations. These operations are unique to relational databases. Relational databases support the important concept of dynamic views. In a relational database, a view is not part of the physical schema; it is dynamic. Hence, changing the data in a table alters the data depicted by the view. Views can subset data, join and simplify multiple relations, dynamically hide the complexity of the data, and reduce the data storage requirements [10].

Two major relational database system prototypes were created, 1) INGRES was developed at the University of California-Berkeley and became commercial and followed up POSTGRES which was incorporated into Informix. This ultimately led to Ingres Corp., Sybase, MS SQL Server, Britton-Lee, and Wang's PACE. This system used QUEL as query language and 2) System R was developed at IBM in San Jose and led IBM's SQL/DS & DB2, ORACLE, HP's Allbase, and Tandem's Non-Stop SQL. DB2 became one of the first DBMS (Database Management System) product based on the relational model [11].

Relational databases use Structured Query Language (SQL), which was invented by IBM in 1970. SQL is a declarative language, which is an easy and human-readable language. SQL instructions are in the form of plain instructions, which can be put into the database for implementation. Most of the database vendors support the SQL standard. Relational databases have excellent security. A relational database supports access permissions, which allow the database administrator to implement need-based permissions for the access of the data in database tables. Relational databases support the concept of users and user rights, thus meeting the security needs of databases. Relations are associated with privileges like create privilege, grant privilege, select privilege, insert privilege, and delete privilege, which authorize different users for corresponding operations on the database. The other important advantages of relational databases include their performance, power, and support for new hardware technologies, as well as their flexibility and capacity to meet all types of data needs. Relational databases are scalable and provide support for the implementation of distributed systems. Owing to their advantages and applications in the operations of data storage and retrieval in modern times, relational databases have revolutionized database management systems [12].

SYBASE is an example of a relational database as it is also mentioned above. However, SYBASE ASE also has object-oriented features. In the relational data model, values are atomic (a value is one that cannot be decomposed into smaller pieces by the DBMS, such as a bank account number, an employee code, etc.); columns in a relational table are not repeating groups or arrays; each row is unique; column values are of the same kind; the sequence of columns is insignificant as the ordering of the columns in the relational table has no meaning. Columns can be retrieved in any order and in various sequences. The benefit of this property is that it enables many users to share the same table without concern for how the table is organized. It also permits the physical structure of the database to change without affecting the relational tables; the sequence of rows is insignificant. The main benefit is that the rows of a relational table can be retrieved in different orders and sequences. Adding information to a relational table is simplified and does not affect existing queries. Each column has a unique name, and certain fields may be designed as keys, which mean that searches for specific values in those fields will use indexing to speed them up. The RDBMS allows for data independence, which helps to provide a sharp and clear boundary between the logical and physical aspects of database management. The RDBMS provides simplicity; this provides a more simple structure than those that were being used before it. A simple structure that is easy to communicate with users and programmers and that can be used by a wide variety of users in an enterprise can interact with a simple model. The RDBMS has a good theoretical background, which means that it provides a theoretical background for the database management field.

1.1.3. 3rd Generation Data Model and Database Management System (DBMS)

The ER-Model and Semantic Data Model are considered part of the 3rd Generation Data Model. In 1976, six years after Dr. Codd published the relational model, Dr. Peter Chen published a paper in the ACM Transaction on Database Systems introducing the entity relationship model (ER Model). An ER model is intended as a description of real-world entities. The ER data model views the real world as a set of basic objects (entities) and relationships among these objects (entities, relationships, and attributes).

It is intended primarily for the DB design process by allowing the specification of an enterprise scheme. This represents the overall logical structure of the database. Although it is constructed in such a way as to allow easy translation to the relational model, the ER diagram represents the conceptual level of database design. A relational schema is at the logical level of database design. The ER model is not supported directly by any DBMS. It needs to be translated into a model that is supported by DBMSs. The entity-relationship model (or ER model) is a way of graphically representing the logical relationships of entities (or objects) in order to create a database. Any ER diagram has an equivalent relational table, and any relational table has an equivalent ER diagram. The semantic data model was developed by M. Hammer and D. McLeod in 1981.

The Semantic Data Model (SDM), like other data models, is a way of structuring data to represent it in a logical way. SDM differs from other data models, however, in that it focuses on providing more meaning to the data itself rather than solely or primarily on the relationships and attributes of the data [13].

SDM provides a high-level understanding of the data by abstracting it further from the physical aspects of data storage. The semantic data model (SDM) has been designed as a natural application modeling mechanism that can capture and express the structure of an application environment. It can serve as a formal specification and documentation mechanism for a database, support a variety of powerful user interface facilities, and be used as a tool in the database design process.

Object-Oriented Data Model (OODM) and Object Relational Data Model (ORDM) are considered under the semantic data model. Object-Oriented Data Model and Object-Oriented Database Management System (OODBMS) (introduced in the early 1980s and 1990s): Research in the field of databases has resulted in the emergence of new approaches such as the Object-Oriented Data Model and Object-Oriented Database Management Systems, which overcome the limitations of earlier models. The object-oriented model has adopted many features that were developed for object-oriented programming languages. These include objects and inheritance (the inheritance feature allows new classes to be derived from existing ones). The derived classes inherit the attributes and methods of the parent class. They may also refine the methods of the parent class and add new methods. The parent class is a generalization of the child classes, and the child classes are specializations of the parent class. There is also polymorphism (a mechanism associating one interface with multiple code implementations) and encapsulation (the association of data with the code that manipulates that data by storing both components together in the database). Object-oriented features are mainly complex objects, object identity, encapsulation, classes, inheritance, overriding, overloading, late binding, computational completeness, and extensibility. Object database features are mainly persistence, performance, concurrency, reliability, and declarative queries. We do not discuss OODM in detail here.

Object-Relational Data Model and Object Relational Database Management Systems (ORDBMS): The object-relational model is designed to provide relational database management that allows developers to integrate databases with their data types and methods. It is essentially a relational model that allows users to integrate object-oriented features into it. The primary function of this new object-relational model is to provide more power, greater flexibility, better performance, and greater data integrity than those that came before it. The ORDBMS provides an addition of new and extensive object storage capabilities to the relational models at the center of the more modern information systems of today. These services assimilate the management of conventional fielded data, more complex objects such as a time series or more detailed geospatial data (such as imagery, maps, and vector data), and varied dualistic media (such as audio, video, images, and applets). This can be done due to the model's ability to summarize methods with data structures. The ORDBMS server can implement complex analytical data and data management operations to explore and change multimedia and other more complex objects. It can be said that the object relational model is an evolutionary technology; this approach has taken on the robust transactional and performance management aspects of its predecessors and the flexibility of the object-oriented model. Database developers can now work with somewhat familiar tabular structures and data definition but with more power and capabilities. This also allows them to perform such task all the while assimilating new object management possibilities. Also the query and procedural languages and the call interfaces in the object relational database management systems are familiar. Object-relational models allow users to define data types, functions, and operators. As a direct result of this, the functionality and performance of this data model are optimized. The massive scalability of the object-relational data model is its most notable advantage, and it can be seen at work in many of today's vendor programs [14].

In ORDBMS, SQL-3 is supported. SQL3 is a superset of SQL/92. SQL3 supports all of the constructs supported by that standard, as well as adding new ones of its own, such as Extended Base Types, Row Types, User-Defined Types, User-Defined Routines, Sub-Types and Super-Types, Sub-Tables and Super-Tables, Reference Types and Object Identity, and Collection Types.

1.1.4. 4th Generation DBMS

This is also based on an object-oriented data model, has come into existence. The Versant database is an example of a 4th generation database. Each of these models modeled the data and the relationship between the data in different ways. Each of the models encountered some limitations in being able to represent the data, which resulted in other models compensating for the limitations.

1.1.5. The Emergence of Multidatabase Systems and Why Academic Institutions and Industries Demand Multidatabase Systems

Keeping in mind the progress in communication and database technologies (concurrency, consistency, and reliability), this has increased the data processing potential. Various protocols are proposed and implemented for network reliability, concurrency, atomicity, consistency, recovery, and replication [15] [16] [17] [18] [19].

The current demand is now how to access data from various existing distributed local databases, either heterogeneous or homogeneous, that are spread across the network over the globe and to maintain all those distributed databases locally as and when required. Heterogeneous distributed local data sources mean there is no homogeneity among the databases at various sites, or at least the distributed local component databases differ in some important respect (e.g., the DBMS they are running or perhaps the data model implemented by it—relational, object-oriented, etc.). Efficiently retrieving information from heterogeneous and distributed local data sources has become one of the top priorities for academic institutions and commercial industries across the business and academic world. Information from these distributed local data sources needs to be integrated into one single system so that the user can retrieve the desired information through an integrated system with a single query. Therefore, the concept of a multidatabase (an integration of distributed databases) system has emerged.

A multidatabase system (MDBS) is a database system that resides on top of existing distributed local autonomous database systems and presents a single database to its users. MDBS usually maintains a single global database schema and a single global conceptual schema, which is an integration of all distributed local database schemas. The schema contains the format, structure, and organisation of the data in a system. MDBS maintains only the global conceptual schema, and each distributed local database schema usually maintains all user data locally. Creating and maintaining the global schema, which requires the use of database integration techniques, is a critical issue in multidatabase systems. A variety of approaches to schema integration have been proposed. The main schema integration problem is associated with combining the diverse distributed local schemas of the different databases into a coherent global view by reconciling any structure or semantic conflicts between the local component databases [20].

Now, a crucial demand in the multidatabase system is how to maintain the schema structure of the multidatabase and all the remote distributed local databases, as all remote distributed local databases are autonomous and evolve over time, so that the schema of the multidatabase and all the remote distributed databases remains consistent and coherent. The issue has been addressed to a certain extent under the project entitled "Method and System for Local and Distributed Remote Schemas Structure Modifications, Propagation, and Maintenance of Metadata in Multidatabases and All Subsequent Locally Distributed Remote Databases" [21].

There is still a challenge in the current situation of multi-database systems: how to maintain metadata seamlessly and without manual interventions of each remote participating database and the global database, as all remote participating databases are autonomous, evolve over time, and also of heterogeneous data models and the databases so that they form a coherent global view by reconciling any structure or semantic conflicts between the local component databases, also discussed above in a different way. We have initiated to work out practically this approach.

It is also true, and we can again say that multidatabase systems have gained the attention of many researchers. They attempt to logically integrate several different independent distributed DBMSs while allowing the local DBMSes to maintain complete control of their operations. It means all existing remote or distributed databases are autonomous and evolve over time. The multidatabase system will address data that resides in more than one distributed database with a single query. On the other hand, there is a possibility for different users to have different interpretations of the same data. Thus, the demand for a multidatabase system is to provide an interpretation of data with the same or similar meaning that has different representations [16].

Again, the multidatabase system is a database system that provides integrated, global access to autonomous, heterogeneous local distributed databases through a single request. It integrates the data from pre-existing, heterogeneous local databases in a distributed environment by presenting to global users a single global conceptual schema, a single data model, and a single query language [22].

Creating and maintaining the global schema, which requires the use of database integration techniques (including schema mapping, correspondence investigation, and transformation of attributes in a similar format), is a critical issue in the multidatabase system. A variety of approaches to schema integration have been proposed, e.g., [16] [17] [19] [20] [23]-[29].

Application scenarios and challenges in educational institutions and commercial industries from the perspective of distributed local relational databases are well justified through our practical approach by presenting a multidatabase system in this article in Section 2 and 4 below. A detailed term of reference about academic or industry challenges for collating information resources at one point of access from distributed locations should be prepared by academic and business leaders in their respective domains of interest. Then, accordingly, policy and procedure and their implementation for the construction of the global conceptual schema will take effect. Connecting and accessing distributed local relational databases will not have any major technical issues for the same.

2. Component Integration Services (CIS) and OmniConnect Consider the Integration of Heterogeneous Relational Distributed Local Databases from the Perspective of Academics (Digital and Literacy in Terms of Teaching and Learning, Research, Collaboration, and Information Sharing in Higher Education)

Component Integration Services (CIS) and OmniConnect, considering heterogeneous relational distributed local databases, will provide an integration and technical level of interoperability of data from multiple sources in a central virtual data store for higher education institutions across domestic and international boundaries (a 360˚ view of data and real-time data access).

If we consider the above-mentioned approach, especially for academic integration, this will give greater insights for students, staff, and faculty: up-to-date and readily available information. This will provide students with an authentic and interactive learning opportunity platform (a cost-effective, digitally delivered learning platform able to transfer knowledge and facilitate effective learning).

This will consider negotiation for data ownership, data access in terms of consolidation of enrollment in various academic and research programs and human resources, with greater emphasis on foreign origin by birth, examining data quality, and data security.

If any academic institution and its subsidiaries have their head office in any country and have many branch units across the globe, they want efficient and quick retrieval of information from all branch units for any kind of decision support for the academic leaders and students as well as their learning and outcomes. Also, in the case of academic institutions, they want to integrate with other academic institutions within the country and abroad according to their policies, terms, and conditions; in the long term, they want collaboration and access to some other data points. The Component Integration Services (CIS) and OmniConnect of the Sybase Adaptive Server Enterprise (ASE) shall meet this very requirement by constituting a global conceptual schema in the multidatabase system by integrating all heterogeneous relational remote or distributed local schemas.

For the above, CIS will provide location transparency and functional compensation. It will provide a uniform view of enterprise data to academic users. Functional compensation allows Component Integration Services to emulate all features of Transact-SQL and interact with a data source only when actual data is needed. With this capability, the full range and power of Transact-SQL can be applied to any data source, whether or not the data source provides support for a particular feature of Transact-SQL. Component Integration Services, together with SQL Anywhere, SAP IQ, and various DirectConnect interfaces, extend the reach of SAP ASE by enabling transparent access to database management systems anywhere in the enterprise. This transparent, extended reach of SAP ASE makes it easy for Enterprise Portal components to: 1) access data from anywhere and present it as dynamic content on Web pages; 2) execute transactions that span heterogeneous boundaries; and 3) view an entire enterprise through a single view provided by the global metadata stored in the SAP ASE/Component Integration Services system catalogs. Component Integration Services allows users to access both SAP® and non-SAP databases on different servers. These external data sources include host data files, tables, views, and RPCs (remote procedure calls) in database systems such as SAP ASE and Oracle. Using Component Integration Services, we can: 1) access tables on remote servers as if the tables were local; 2) perform joins between tables on multiple remote, heterogeneous servers. For example, it is possible to join tables between an Oracle database management system (DBMS) and an SAP ASE and between tables on multiple servers; 3) transfer the contents of one table into a new table on any supported remote server by means of a select into statement; 4) maintain referential integrity across heterogeneous data sources; and 5) access native remote server capabilities using the Component Integration Services passthrough mode. Component Integration Services can be used by anyone who needs to access multiple data sources or legacy data. It can also be used by anyone who needs to migrate data from one server to another. A single server is often used to access data on multiple external servers. Component Integration Services manages the data regardless of the location of the external servers. Data management is transparent to the client application. Component Integration Services, in combination with EnterpriseConnect™ and MainframeConnect™, provide transparent access to a wide variety of data sources, including: 1) Oracle; 2) Informix; 3) Microsoft SQL Server; 4) SAP Adaptive Server Enterprise; 5) SQL Anywhere; and 6) SAP IQ. Mainframe data, including: 1) ADABAS; 2) IDMS; 3) IMS; and 4) VSAM [30].

3. About Sybase Adaptive Server Enterprise (ASE)

Sybase Adaptive Server Enterprise (ASE) is high-performance relational database management system software manufactured and sold by Sybase, Inc. ASE is a versatile, enterprise-class RDBMS that is especially good at handling OLTP workloads. ASE is used intensively in the financial world (banks, stock exchanges, and insurance companies), in e-commerce, as well as in virtually every other area, such as academic institutions. It has now come to be known as SAP Adaptive Server Enterprise (SAP ASE). Please also see the details about the SAP Adaptive Server Enterprise (SAP ASE) in [31].

The most recent ASE release is ASE version 15.7 (released in September 2011); the previous version is 15.5. ASE 15.7 is also known as “the SAP release” since this is the ASE version that SAP is using to support the Business Suite ERP package on top of Sybase ASE.

The most recent version of Sybase is SAP Sybase Adaptive Server Enterprise (ASE), which is a high-performance relational database management system for mission-critical, data-intensive environments. It ensures the highest operational efficiency and throughput on a broad range of platforms [32].

ASE started its life in the mid-eighties as “Sybase SQL Server”.

4. Role of Component Integration Services and OmniConnect of Sybase ASE under Multidatabase System

Component Integration Services (CIS) is a feature of Adaptive Server Enterprise. CIS allows Adaptive Server to present a uniform view of enterprise data to client applications and provides location transparency to enterprise-wide data sources. The CIS allows users to access both Sybase and non-Sybase data servers. These external data sources include host data files and tables, views, and remote procedure calls (RPCs) in data servers such as Adaptive Server Enterprise, DB2, and Oracle [33].

The Component Integration Services (CIS) is the software layer that extends the reach of the Adaptive Server to external data. The CIS layer is used by OmniConnect and ASE to provide access methods for proxy tables. The services of CIS are also used to provide a degree of SQL syntax transformation, so that interaction with a remote data source can be done in its native language [34].

In this chapter, in the Mulidatabse System, the CIS extends the reach of ASE to external data.

OmniConnect [35] is one of several Adaptive Server Enterprises bundles, which uses the Component Integration Services (CIS) features to provide access to external data sources. OmniConnect allows users to access both Sybase and Non-Sybase database on different servers. These external data source include host data files and tables, views and RPCs (remote procedure calls) in database systems such as Adaptive Server, Oracle, and DB2 as shown in **Figure 1** below and as it has already mentioned earlier too.

As ASE and OmniConnect support access to a large number of external data sources. For access to other Sybase Adaptive Server databases (Enterprise, Anywhere, IQ), no gateway is required. For access to non-Sybase data sources, access is generally provided through a DirectConnect™ gateway. Gateways are provided for the following data sources; Oracle, DB2, AS/400, Informix, Microsoft SQL Server, Ingres, Rdb, SQL/DS, Datacom, Teradata, Model 204, RMS files (OpenVMS only), Non-relational data, such as ADABAS, IMS, IDMS, VSAM, and sequential files through InfoHub. In addition, new features of Adaptive Server Anywhere will provide access to a wide variety of desktop data sources, such as; Microsoft Access, Microsoft Excel spreadsheets, Microsoft SQL Server, Lotus Notes databases, Foxpro, Text files, Other ODBC and JDBC data sources [34].

As Adaptive Server presents a uniform view of enterprise data to client application and provides location transparency to enterprise-wide data sources as I

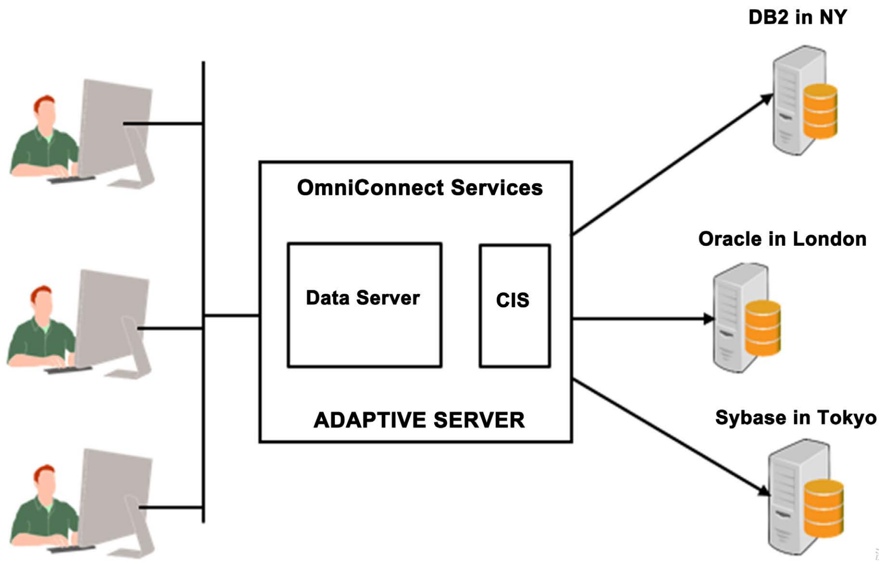

Figure 1. OmniConnect connects to multiple vendors database-resketch [35].

have discussed earlier. The Location transparency insulates a client application from knowledge of the physical storage location of data. For objects whose storage location is actually external to the local database, access methods provide the means to issue queries over a network, resolving pathnames, network, and syntax issues in a manner transparent to the database application. All references to these externally located objects are made through proxy tables, which appear in system catalog as ordinary table objects. Access methods are associated with each proxy table when the proxy table is created, providing the ability to reference these objects through normal SQL statements such as select, insert, delete, and update. Also, these objects can be referenced from within stored procedures, views, and triggers [34].

We have discussed about the proxy table in detail in our recently published research paper [36].

In this chapter, Multidatabase System is termed a local database system; client application is an end-user application of the Multidatabase System; physical storage location of data refers to all remote/distributed and heterogeneous relational databases; and the System Catalog, which is created in the Multidatabase System during the integration of all remote and heterogeneous relational databases and is also termed the Global System Catalog, is also described in this chapter.

OmniConnect uses the CIS features of ASE to provide customers with enhanced interoperability. Unlike the ASE, it does not provide local data storage [35].

Using an OmniConnect, we can avail following features:

Some of them have been discussed above.

1) Access tables in the remote database servers as if they were local; 2) Perform joins between tables in multiple remote, heterogeneous database servers. For example, it is possible to join tables between an Oracle Database Management System (DBMS) and Sybase ASE and between tables in multiple Sybase ASEs; 3) Transfer the contents of one table into a new table on any supported remote database server using a select into statement; 4) The Multidatabase System will allow transparent access to heterogeneous data sources through an application that is developed using any front-end development tool such as PowerBuilder, Microsoft Access, or DataEase; 5) It will maintain referential integrity across heterogeneous data sources; and 6) It will access native remote database server capabilities using the CIS passthrough mode.

We have used the Component Integration Services Technique (CIS) and OmniConnect of Sybase ASE to integrate multiple distributed local component heterogeneous relational databases to create a global conceptual schema in the multidatabase system, and a query that is submitted on the global conceptual schema will then be translated into a number of sub-queries on physically remote distributed heterogeneous relational databases.

The global conceptual schema in the multidatabase system contains a group of proxy tables that are used to access remote tables and views. Proxy tables are key to location transparency. A proxy table is a local table in the multidatabase system containing metadata that points to a remote table or view of the remote and local heterogeneous relational databases. The global conceptual schema in the multidatabase system contains metadata and some statistics about all remote database tables and views.

When proxy tables are created within the local server (Multidatabase System), metadata from all the remotely located distributed databases is stored within the Local System Tables of the Multidatabase System. This metadata can be queried locally to quickly obtain information about proxy tables. This information includes column attributes and index definitions and is presented to local applications in Sybase terms. Obviously, the maintenance of this metadata can become an issue, as the enterprise or local remote database are generally a fairly dynamic entity and subject to change. For this reason, metadata management becomes an extremely important issue. To help with this task, the administration utility, Sybase Central™, has been enhanced to provide features that will import and synchronise metadata for proxy tables [34].

Here, metadata is called the global system catalog, and the local server is the Multidatabase System. Worked out the very critical issue "maintenance and synchronization of metadata of the multidatabase system and all subsequent distributed local database system up-to the certain extents". Also obtained a copyright to this novel invention. Please see the details in [21].

Performance is the leading concern expressed by most users of distributed environments. Component Integration Services (CIS) addresses many of these concerns by focusing on two separate aspects of distributed query processing; 1) Query Optimization: Analyzing query syntax to establish optimal join strategy and join order and, 2) Query Decomposition: Analyzing query syntax and determining the amount of work to be pushed to remote sites for processing. These aspects of query processing are addressed by the global optimization features of CIS, which extend the standard ASE query optimizer's knowledge of tables to include consideration of external data sources. In a multidatabase system, query optimization plays an important role in query performance. The query optimization process attempts to minimize query response time and reduce query cost. Global queries are decomposed into multiple sub-queries that will be executed in different remote local database systems. When the results from each of the remote local database systems are returned-back, the data must be manipulated and merged in such a way that is conforms according to the global schema and the canonical data format. The query optimizer within OmniConnect and ASE has been upgraded to evaluate this cost for proxy tables. This is done in two important ways; 1) Distribution Statistics: When the update statistics command is issued, distribution statistics for each index on the proxy table are obtained from the external object. These distribution statistics are then stored in new system catalogs, systabstats and sysstatistics, and are then used by the optimizer to establish relative access costs based on knowledge of index usage and 2) Network

Access Cost: The optimizer has also been enhanced to estimate the cost of network access, when running a query that references a proxy table [22] [34].

However, effort in constructing the global schema using CIS may be expensive with respect to the frequency of usage.

5. Database Architectural Framework

The ANSI/X3/SPARC architecture [6] [7] [8] as shown in **Figure 2** below is claimed to be based on the data organization in DBMS standardization. It recognizes three views of data; 1) Local internal schema/view: Local (internal) schema/view shows how the data is stored on each site. The format of the internal schema is dependent on the LDBMS of each site; 2) A global conceptual schema/view: A global (conceptual) schema describes the data throughout a network and shows what data is at what site. The global schema usually stored in a global directory; 3) A user external schema/view: A user (external) schema/view shows how user will view and manipulate the data.

6. Technical Implementation Overview for the Integration of Distributed Databases and Constitution of Global Conceptual Schema in the Multidatabase System in General, Either from an Academic or Commercial Perspective

We illustrate above the concept of a proxy table and data mapping in **Figures 3-6** above. In the above **Figure 3**, table T1 is located at remote database server 1, and table T2 is located at remote database server 2. A proxy table P1 related to

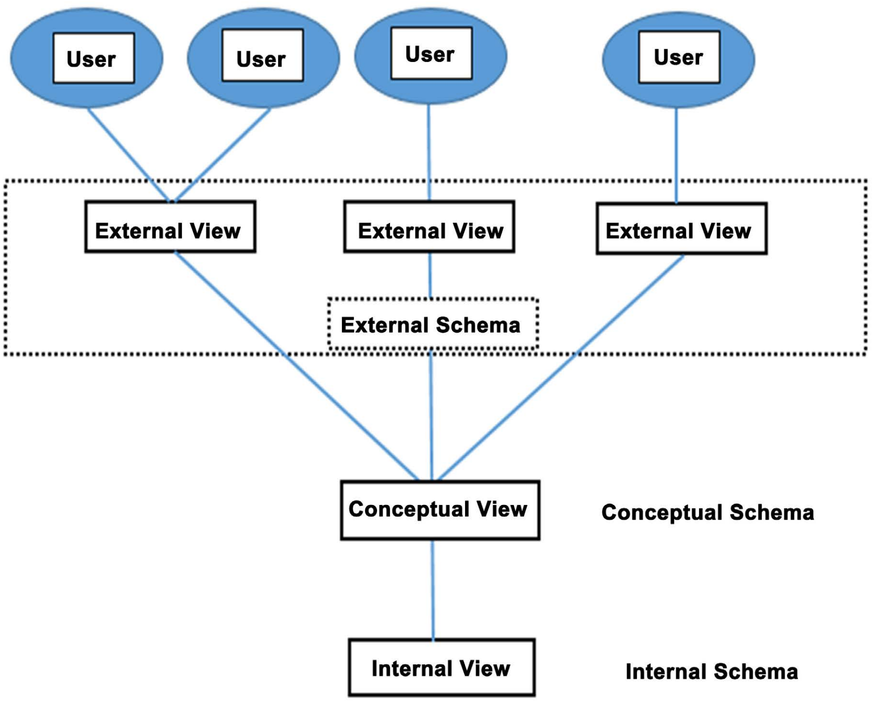

Figure 2. ANSI/SPARC architecture [18] [19].

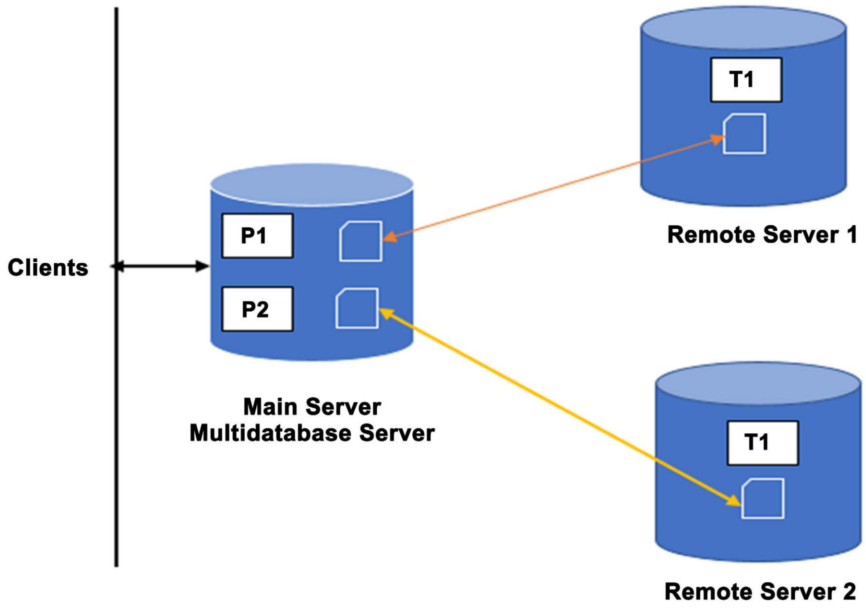

Figure 3. One-to-one mapping between a proxy table of the multidatabase and a table of the local remote database [19].

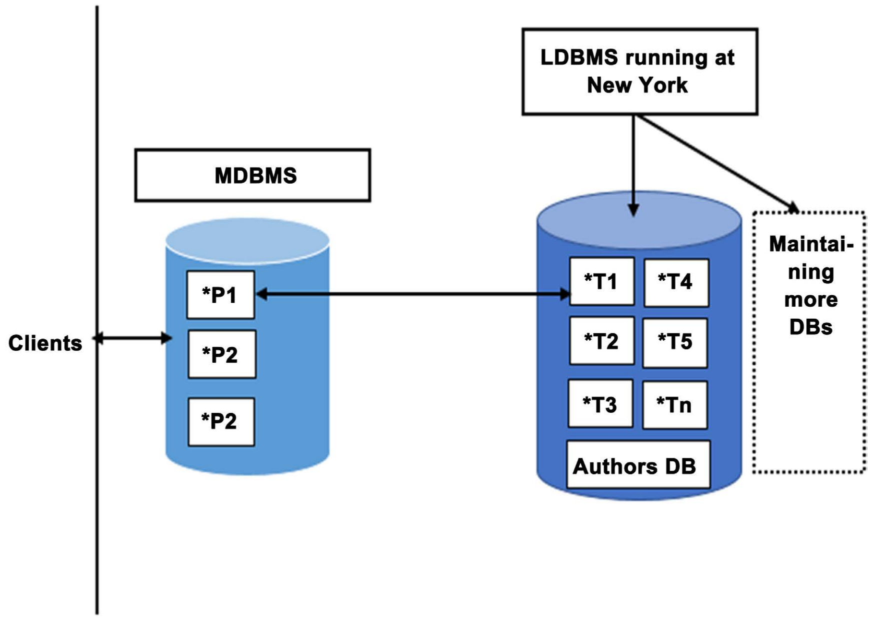

Figure 4. One-to-one mapping between a proxy table of the multidatabase and a table of the local remote database [17].

table T1 and a proxy table P2 related to table T2 are created on the main server or local server, *i.e.*, in the multidatabase system. Then the client of the multidatabase system will access remote tables T1 and T2 using proxy tables P1 and P2. Similarly a case of Figure 4. If attributes in both tables T1 and T2 are the same, the client can join both proxy tables and get the results accordingly. There is a possibility that the proxy table is created using local views, and local views are created using the tables of the local or remote database server as it is shown in

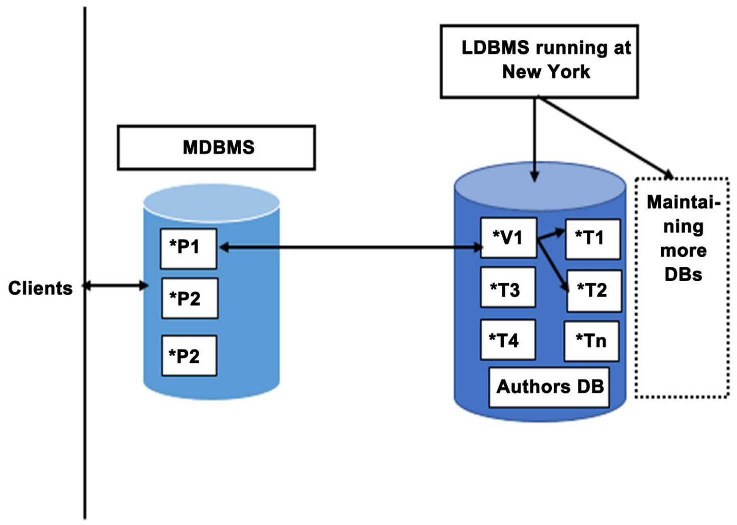

Figure 5. One-to-one mapping between a proxy table of the multidatabase and a view of the local remote database, where the local view is created by two local tables [17].

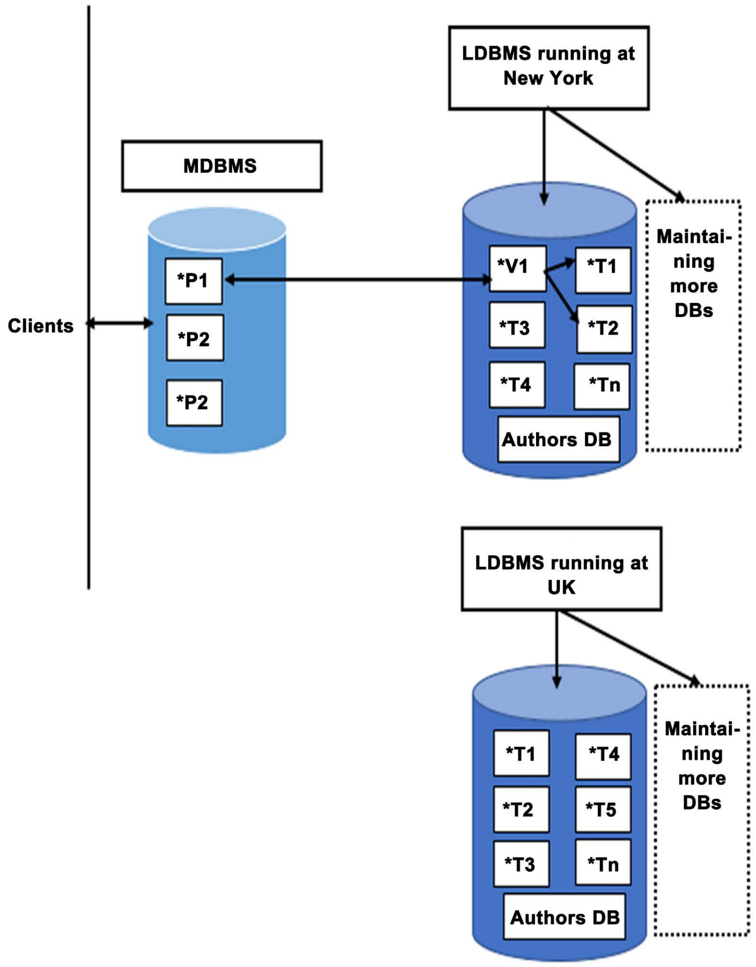

Figure 6. Query execution using two proxy tables, where the first proxy table is directly mapped to a local remote view of the database server located in New York, where the remote view is created by two tables. The second proxy table is directly mapped to the local remote table of a database server located in the UK [17].

Figure 5 and Figure 6. The access mechanisms for all remote tables and views are based on the programming logic of the client software of the Multidatabase System and the Remote Server Definition.

CIS has extended the definition of a remote database server within ASE to include the notion of a server class. Server classes establish the access paths to external or remote databases, and are included in the pathnames of proxy tables. The syntax used to define a server to ASE and OmniConnect is [34]:

sp_addserver server_name, server_class, net_name.

The parameters for this stored procedure are:

server_name—the name of the external server, used within proxy table pathnames.

server_class—the class name associated with this server. For e.g. ASEnterprise, ASAnywhere, ASIQ, sql_server, db2.

net_name—the name of the server found in the interfaces file.

When we issue a query on the global conceptual schema of the multidatabase system using proxy tables with join either with union or intersect, the multidatabase system will open connections to the remote database servers and will pass the part of the query (sub-queries) involved to the remote tables or views on the remote database servers, where sub-queries will be executed locally and results will be sent back to the multidatabase system. The result will be stored in the multidatabase system data cache or in the working storage area.

If any time a proxy table is updated by the client, the update command will be sent to the remote database server, and the table will be updated there locally.

We can define a practically unlimited number of remote database servers and proxy tables for the multidatabase system. Sybase ASE supports a very simple SQL command to create a proxy table. We can create proxy tables for the entire remote database server by using a single command-line proxy database. However, it is not necessary in the case of building a global conceptual schema in the multidatabase system.

Location transparency of the remote data is enabled by creating a local proxy table in the multidatabase system; once remote servers are defined, we can easily map to the remote table or view.

If the table already exists on the remote database server, we use the create existing table statement. This statement defines the proxy table for the multidatabase system for an existing table on the remote database server. Metadata from the remote locations identified by pathname will then be imported into the CIS and also compared with the column list. The data type of the columns must be compatible (convertible); otherwise, the command will be rejected. Pathname represents a four-part identifier value, consisting of server.dbname.owner.objectname. The server-name component must represent a server added via sp_addserver.

If a table does not exist on the remote database server, we use the create table statement. This statement creates a new table on the remote database server and also defines the proxy table for that table in the multidatabase system. We illustrate some examples here for creating a proxy table:

Example 1

To create a proxy table named PEmployee on the multidatabase system from a remote table named Employee of the database server named USAASEServer and database name USAHR, we use the following syntax:

Create exixting table PEmployee at "USAASEServer.USAHR.dbo.Employee".

Example 2

To create a new table called Employee on the remote database server USAASEServer and the database USAHR and define the proxy table PEmployee for that table on the Multidatabase System, we use the following syntax:

Create table Employee (Id integer not null, Name char (30) not null, Address char (45) null) at "USAASEServer.USAHR.dbo.PEmployee".

Example 3

One-to-one mapping between a proxy table of the multidatabase and a view of the local remote database, where the local view is again created by two local tables:

Create proxy_table P1 external tablenat "USAASEServer.USAHR.dbo.V1".

Where create poxy_table is a shorthand version of the create existing table.

Example 4

Query execution using two proxy tables: the first proxy table is directly mapped with a local remote database view that is created by two local tables, and the second proxy table is directly mapped with a local remote table of a database server located in the UK.

Select Id, Name from P1 union select Id, Name from P2.

Please also see the detailed work from [15].

7. Query Processing in CIS

The query processing steps taken when component integration services are enabled are similar to the steps taken by SAP ASE. The exceptions are: 1) If a client connection is made in passthrough mode, the SAP ASE query processing is bypassed and the SQL text is forwarded to the remote database server for execution; and 2) When *select*, *insert*, *delete*, or *update* statements are submitted on the database server for execution, additional steps may be taken by the component integration services to improve the query's performance if local proxy tables are referenced. The steps are: 1) Query Parsing; 2) Query Normalization; 3) Query Preprocessing; 4) Decision Point; 5) SAP ASE Optimization and Plan Generation; 6) Component Integration Services Plan Generation; and 7) Component Integration Services Remote Location Optimizer. Please see the details in [30].

We also discuss here about some common query optimization techniques and strategies.

The performance of queries involving proxy tables that reference two or more remote servers is critical to the success of the CIS features incorporated into

Adaptive Server Enterprise. Several optimization strategies are provided to make distributed query processing as optimal as possible within the constraints of the current Adaptive Server Enterprise query processor. Using update statistics, Component Integration Services creates extremely accurate distribution statistics for remote tables. This information is used to determine the optimal join order, giving Component Integration Services the ability to generate optimal queries against remote databases which may not support cost-based query optimization. If quickpass mode can be used, Component Integration Services produces a simplified query plan. When statements contain proxy tables, they are executed more quickly when processed by the remote server than when processed through the Adaptive Server plan generation phase. Adaptive Server optimization and plan generation evaluates the optimal path for executing a query and produces a query plan that tells the Adaptive Server how to execute the query. Adaptive Server generates a query plan containing the optimal join order for a multitable query without regard to the storage location of each table. If remote tables are represented in the query, Component Integration Services, which takes the storage location into account, performs additional optimization for the following conditions: 1) Join processing. 2) Aggregate processing. In order to make intelligent evaluations of a query to improve performance in the above areas, statistics are required. These are obtained by executing the command update statistics for a specific table [36].

We may also implement the distribution of data using relation fragmentation to different sites, replication and allocation, and local optimization at each site. This may reduce the amount of irrelevant data accessed by the applications of the database, thus reducing the number of disk accesses. Fragmentation is a technique to split a single relationship in a database into two or more partitions. Also, the combination of the partitions supports the original database without any loss of information. Fragmentation is mainly considered horizontal fragmentation, vertical fragmentation, and mixed or hybrid fragmentation. The main advantage of fragmentation is that it improves the performance of distributed database design by increasing efficiency since data is stored only where it is needed. Fragments can be allocated at different network sites in a process called data allocation, but fragmentation also has some disadvantages, such as that managing fragmented data can lead to increased complexity, data integrity, and ensuring consistent data integrity across fragments and recovery. Additionally, the complexity of retrieving and reassembling fragmented data during recovery can be time-consuming. In our approach, we do not consider data fragmentation and replication. However, we may consider it if needed. There are other aspects in the multidatabase: the mediator performs the two first steps, query decomposition and data localization, by rewriting queries using views and query optimization (in our approach, CIS OmniConnect may be considered as mediator), whereas the wrapper performs the last layer, query translation and execution, by returning to the mediator the results provided by the execution on each DBMS of translated queries according to its language syntax (we do not involve the mediator in our approach since we are considering homogenous relation databases of different database vendors). Therefore, we have briefly discussed the query optimization above.

8. Conclusion

At present, in the fields of computer science, engineering, and information technology, the multidatabase system is a hot topic of research in parallel and distributed computing. In this chapter, we have discussed how to integrate heterogeneous relational distributed local databases that are spread across the globe and how to create a global conceptual schema in the multidatabase system with a set of proxy tables using Component Integration Services (CIS) and OmniConnect. Any academic institution can implement this technique for integrating with its' subsidiaries and with other academic institutions of similar standards within the country and abroad. This is feasible for academic institutions and commercial industries as well. Considering the academic institutions, the CIS will improve IT integration for higher education institutional performance in terms of educational management, teaching, learning, research, and governance and will prove an innovative strategy to support the modernization and large expansion of higher education institutions, including to promote international students' academic integration and institutional collaboration. The same technique can easily be applied in the commercial industry. Section 1 discussed well about the evolution databases and data models and emergence of multidatabase systems based on the published documents and our research outcomes. The chapter will help our students a lot, as we have discussed well the evolution of databases and data models and the emergence of multidatabases. The main significance, contributions, and innovations of the article's study are to implement existing software technologies pertaining to proxy tables and the proxy database in creating a global conceptual schema, a global database, and a global catalog/global data dictionary/global metadata and presenting them in the form of a multidatabase system in terms of academic and industrial demand in a consistent and coherent view of information resources from all respective local component distributed databases. Future research would be in the direction of local schema modification, propagation, and maintenance of metadata on the global conceptual schema, considering distributed heterogeneous database platforms practically. This is a very challenging issue. However, we have resolved the same issue in the homogenous distributed database platform of different vendor products practically and also obtained copyright of patent nature.

Acknowledgements

We have tested this approach on our one viable project. This is an effective technique that we have proposed. The industry will benefit. Any academic institution can implement this technique for integrating with its subsidiaries and with

other academic institutions of similar standards within the country and abroad. Whatever information is cited is properly included in the references. There is no funding for this research. Since some additional useful information is cited, the source of information for each citation is properly mentioned in the references column.

Limitations of Work

We have not considered the distributed, heterogeneous local databases. However, we have considered distributed heterogeneous relational local databases.

Results and Discussion

The approach has been completely tested in our one live project. It is to be considered more scalable and resilient. The information from the distributed environment is being accessed in a secured manner.

Disclaimer

This chapter is an extended version of the article published by the author(s) in the Conference Proceedings, 13th ISITA; National Conference of Recent Trends in Mathematical and Computer Sciences, T.M.B. University, Bhagalpur, India, January 3-4, 2015.

Conflicts of Interest

The authors have declared that no competing interests exist.

References

[1] Joshi, K. (2023) Chapter 6, Database Management. https://www.umsl.edu/~joshik/msis480/chapt06.htm

[2] Kroenke, D. and Auer, D. (2007) Database Concepts. 3rd Edition, Prentice, New York.

[3] Mansour, M. and Yang, W. (2017) History of Database Applications. Term Paper, Spring, The University of Rochester's Computer Science Department, Rochester.

[4] Lin, C. (2003) Object-Oriented Database Systems: A Survey.

[5] HND Database & Design. Compiled: By Engr. Shahzadah Ashraf Bande'Shah. https://www.studocu.com/row/document/beaconhouse-national-university/computer-science/database-and-design-by-bandeshah/56389764

[6] Umar, A. (1988) Distributed Database Management Systems Issues and Approach. Technical Report, The University of Michigan, Ann Arbor.

[7] Tsichritzis, A.K. (1978) The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems. *Information Systems*, **1**, 173-191. https://doi.org/10.1016/0306-4379(78)90001-7

[8] Ozsu, T. and Valduriez, P. (2020) Principles of Distributed Database Systems. Springer Nature, Cham.

[9] How Does the Evolution of Database Systems Happened? https://www.geekinterview.com/question_details/35087

[10] What Are the Advantages of Relational Data Model? https://science.blurtit.com/634750/what-are-the-advantages-of-relational-data-model?expand_article=1

[11] Berg, K.I., Seymour, T. and Goel, R. (2013) History of Databases. *International Journal of Management & Information Systems*, **17**, 29-36. https://doi.org/10.19030/ijmis.v17i1.7587

[12] Advantages of Relational Databases. https://techspirited.com/advantages-of-relational-databases

[13] Database Management. https://databasemanagement.fandom.com/wiki/Semantic_Data_Model

[14] Object-Relational Model. https://www.learn.geekinterview.com/it/data-modeling/object-relational-model.html

[15] Ali, M.G. (2022) Adopting a Proxy Database to Prevent Direct Access to Distributed Transaction Databases Ensuring Information Security. *Journal of Advances in Mathematics and Computer Science*, **37**, 43-55. https://doi.org/10.9734/jamcs/2022/v37i330441

[16] Ali, M.G. (2010) A Framework for Creating Global Schema Using Global Views from Distributed Heterogeneous Relational Databases in Multidatabase System. *Global Journal of Computer Science and Technology*, **10**, 31-35.

[17] Ali, M.G. (2010) A Framework for Creating Global Schema Using Proxy Tables from Distributed Heterogeneous Relational Databases in Multidatabase System. *International Journal of Computer and Electrical Engineering*, **2**, 846-851. https://doi.org/10.7763/IJCEE.2010.V2.238

[18] Ali, M.G. (2010) Local Schemas Modifications Propagation and Maintenance of Metadata in Multidatabase System. *International Journal of Computer Applications*, **6**, 6-9. https://doi.org/10.5120/1112-1456

[19] Ali, M.G. and Murtuza, M.G. (2015) Use of Component Integration Services in Multidatabase Systems. 13*th ISITA*; *National Conference of Recent Trends in Mathematical and Computer Sciences*, Bhagalpur, 3-4 January 2015, 172-176.

[20] Ali, M.G. (2009) Object-Oriented Approach for Integration of Heterogeneous Databases in a Multidatabase System and Local Schemas Modifications Propagation. *International Journal of Computer Science and Information Security*, **6**, 55-60.

[21] Ali, M.G. (2019) Method and System for Local and Distributed Remote Schemas Structure Modifications Propagation and Maintenance of Metadata of Multidatabase and All Subsequent Local Distributed Remote Databases. Copyright Retained with IIT Kharagpur, Reg. No. L-84975/2019, Copyright Office, Government of India.

[22] Ko, C.Y. (2000) Three Approaches to a Multidatabase System. *Proceedings of the Philippine Computing Science Congress* (*PCSC*), Manila, 29 November-1 December 2000.

[23] Czejdo and Taylor, M. (1991) Integration of Database System Using an Object-Oriented Approach. *Proceedings of* 1*st International Workshop on Interoperability in Multidatabase Systems*, Kyoto, 7-9 April 1991, 30-37.

[24] Gotthard, W., Lockemann, P.C. and Neufeld, A. (1992) System-Guided View-Integration for Object-Oriented Databases. *IEEE Transactions on Knowledge and Data Engineering*, **4**, 1-22. https://doi.org/10.1109/69.124894

[25] Kaul, M., Drosten, K. and Neuhold, E.J. (1990) ViewSystem: Integrating Heteroge-

neous Information Bases by Object-Oriented Views. *Proceedings of* 6*th International Conference on Data Engineering*, Los Angeles, 5-9 February 1990, 2-10.

[26] Koh, L. and Chen, A.L.P. (1994) Integration of Heterogeneous Object Schemas. In: Elmasri, R.A., Kouramajian, V. and Thalheim, B., Eds., *Entity-Relationship Approach*, Springer-Verlag, Berlin, 297-314. https://doi.org/10.1007/BFb0024375

[27] Ching-Ming, C. (2001) Schema Integration between Object-Oriented Databases. *Tomkang Journal of Science and Engineering*, **4**, 37-44.

[28] Duwairi, R.M. (2003) A Framework for Generating and Maintaining Global Schemes in Hetrogeneous Multidatabases Systems. Proceedings of the 2003 *IEEE International Conference on Information Reuse and Integration* (*IN*-2003), Las Vegas, 27-29 October 2003, 200-207.

[29] Chung, S.M. and Mah, P.S. (1995) Schema Integration for Multidatabase Using the Unified Relational and Object-Oriented Model. *CSC*'95: *Proceedings of the* 1995 *ACM* 23*rd Annual Conference on Computer Science*, Nashville, 28 February-2 March 1995, 208-215. https://doi.org/10.1145/259526.259556

[30] SAP Adaptive Server Enterprise, Component Integration Services Users Guide, Component Integration Services Overview. https://help.sap.com/docs/SAP_ASE/af9a1f6aa41e49cb8759274040665d0f/a9fc7fb1bc2b10149fe08325c19c990e.html?version=16.0.3.4

[31] SAP Adaptive Server Enterprise (SAP ASE). https://www.sap.com/india/products/sybase-ase/features.html

[32] SAP Sybase Adaptive Server Enterprise. https://www.sap.com/india/index.html?url_id=auto_hp_redirect_indial http://www.sybase.com/products/databasemanagement/adaptiveserverenterprise

[33] Real Time Data Services 3.5, Replication Server Heterogeneous Replication Guide, Data Server Issues. https://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.rs_15.0.rshetrep/html/rshetrep/CHDJJBJG.htm

[34] Olson, S. (1998) Distributed Query Processing Using Sybase Adaptive Server Enterprise and OmniConnect 11.9.2.

[35] OmniConnectTM, Introducing OmniConnect for Sybase Adaptive Server Enterprise. https://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.rs_15.0.rshetrep/html/rshetrep/CHDJJBJG.htm

[36] SYBASE Component Integration Services User's Guide. https://infocenter-archive.sybase.com/help/topic/com.sybase.dc32702_1250/pdf/omni_ug.pdf