3. ALGORITHM FOR FINDING DUPLICATES IN DISTRIBUTED DATABASE USING ONTOLOGY AND SHORTCUTS DICTIONARY

Distributed nature of database makes the task of comparison at different system parallely. The dictionary available is maintained as a local resource at the system which will reduce the network traffic. An application created uses the web services. Hence it will be possible to store the identified tuple ids at the location specified by the user or at the server side where web server resides.

Distributed Databases:

A distributed database [1] system allows applications to access data from local and remote databases. In a homogenous distributed database system, all sites use same DBMS product, for example, all databases are Oracle databases. It is much easier to design and manage. The approach provides incremental growth and allows increased performance. In a heterogeneous distributed database system, at least one of the databases is different than others Database, for example, one can be SQL Server and all others can be Oracle databases. Sites may run different DBMS products, with possibly different underlying data models. This occurs when sites have implemented their own databases first, and integration is considered later. Translations are required to allow for different hardware and/or different DBMS products. Typical solution is to use gateways. Distributed databases use client/server architecture to process information requests.

3.1 HOMOGENOUS DISTRIBUTED DATABASE SYSTEMS

A homogenous distributed database system is a network of two or more Databases that reside on one or more machines. Fig 3.1 illustrates a distributed system that connects three databases: hq, mfg, and sales. An application can simultaneously access or modify the data in several databases in a single distributed environment. For example, a single query from a Manufacturing client on local database mfg can retrieve joined data from the products table on the local database and the dept table on the remote hq database. For a client application, the location and platform of the databases are transparent. You can also create synonyms for remote objects in the distributed system so that users can access them with the same syntax as local objects.
3.2 HETEROGENEOUS DISTRIBUTED DATABASE SYSTEMS

In a heterogeneous distributed database system, at least one of the databases is different Database system from others. To the application, the heterogeneous distributed database system appears as a single, local, Database. The local Database server hides the distribution and heterogeneity of the data. The local Database server accesses the other Database system using local database Heterogeneous Services in conjunction with an agent. If you access the Database data store of the other database using a local database Transparent Gateway, then the agent is a system-specific application. For example, if you include a Sybase database in an Oracle Database distributed system, and then you need to obtain a Sybase-specific transparent gateway so that the Oracle Database in the system can communicate with it. Alternatively, you can use generic connectivity to access non-Oracle Database data stores so long as the non-Oracle Database system supports the ODBC or OLE DB protocols.

3.3 CLIENT/SERVER DATABASE ARCHITECTURE

A database server is the DBMS software managing a database, and a client is an application that requests information from a server. Each computer in a network is a node that can host one or more databases. Each node in a distributed database system can act as a
client, a server, or both, depending on the situation. Let us consider an example of Oracle client/server system. In Fig 3.2, the host for the hq database is acting as a database server when a statement is issued against its local data (for example, the second statement in each transaction issues a statement against the local dept table), but is acting as a client when it issues a statement against remote data (for example, the first statement in each transaction is issued against the remote table emp in the sales database). A client can connect directly or indirectly to a database server. A direct connection occurs when a client connects to a server and accesses information from a database contained on that server.

![Oracle Distributed Database System](image)

**Fig 3.2: Oracle Distributed Database System**

### 3.4 SQMD: SINGLE QUERY MULTIPLE DATABASE ARCHITECTURE

SQDM [2] is a distributed database system architecture composed of three tiers. Figure 3.4 shows a broad 3-tier architecture view for our scalable distributed database system. The scalable, distributed database system architecture is composed of three tiers – the web service client (front-end), a web service and message service system (middleware), agents and a collection of databases (back-end). The virtualization environments based on OpenVZ [3] and software (with web service, SQMD using publish/subscribe mechanism, and data clustering program) architecture concentrate on increasing scalability with increased size of distributed data, providing high performance service with the enhancement of query/response interaction time, and improving data locality. This message and service system, which represents a middleware component, provides a mechanism for simultaneously disseminating (publishing) queries to and retrieving (subscribing) the results.
of the queries from distributed databases. The message and service system interacts with a web service which is another service component of the middleware, and database agents. The web service acts as query service manager and result aggregating service manager for heterogeneous web service clients. The database agent acts as a proxy for database server.

![SQMD Architecture Diagram](image)

**Fig 3.3: SQMD Architecture**

### 3.5 WEB SERVICES

1. Web Service: A web service is a software platform to build applications running on a variety of platforms as services [4]. The communication interface for web service is described by XML that follows SOAP (Simple Object Access Protocol) standard [5]. Other heterogeneous systems can interact with the web service through a set of descriptive operations using SOAP. For web services SQMD used the open-source Apache Axis [6] library which is an implementation of the SOAP specification. Also we used WSDL (Web Service Description Language) [7] to describe our web service operations. A web service client reads WSDL to determine what functions are available for database service: one service is query request service and the other is reliable service which detects whether database servers fail.
2. Web Service Client (front-end): Web service clients can simultaneously access (or query) the data in several databases in a distributed environment. Query requests from clients are transmitted to the web service, disseminated through the message and service system to database servers via database agents. The well-known benefits of the three-tier architecture results in the web service clients do not need to know about the individual distributed database servers, but rather, send query requests to a single web service that handles query transfer and response.

3. Message and service middleware system: For communication service between the web service and middleware, and the middleware and database agents, we have used Narada Brokering [8, 9] for message and service middleware system as overlay built over heterogeneous networks to support group communications among heterogeneous communities and collaborative applications. The Narada Brokering from Community Grids Lab (CGL) [10] is adapted as a general event brokering middleware, which supports publish/subscribe messaging model with a dynamic collection of brokers and provides services for TCP, UDP, Multicast, SSL, and raw RTP clients. The Narada Brokering also provides the capability of the communication through firewalls and proxies. It is an open source and can operate either in a client-server mode like JMS [11] or in a completely distributed peer-to-peer mode [12].

4. Database Agent: In Figure 3.4, the database agent (DBA) is used as a proxy for database server (PostgreSQL). The DBA accepts query requests from front-end users via middleware, translates the requests to be understood by database server and retrieves the results from the database server. The retrieved results are presented (published) to the requesting front-end user via message /service system (broker) and web service. Web service clients interact with the DBA via middleware, and then the agent communicates with PostgreSQL database server. The agent has responsibility for getting responses from the database server and performs any necessary concatenations of responses occurred from database for the aggregating operation of the web service. As an intermediary between middleware and back-end (database server), the agent retains communication interfaces (publish/subscribe) and thus can offload some computational needs.

5. Database Server: A number of data partitions split by data clustering or horizontal partitioning method are distributed into PostgreSQL database servers. Another benefit of database servers based on the three-tier architecture is that they do not concern about a large number of heterogeneous web service clients but need to only process queries and return results of the queries. We used the open source database management system PostgreSQL.
With such an approach using open management system, we can build a sustainable high functionality system taking advantage of the latest system technologies with appropriate interface between layers (agents and database host servers) in a modular fashion.

### 3.6 Shortcut Dictionary

Semantics is a grand challenge for the current generation of computer technology. It is the key for unlocking the door to personal agents that can roam the semantic search. Semantics of the term used depends on the context in which it is used.

In record linkage problem, context sensitive data plays an important role. Accuracy of deduplication of records depends on the knowledge of vocabulary used at local level and also the context in which it is used. For example, the abbreviation ‘St.’ stands for Saint as well as for Street. Apart from this, many dictionary words have different representations in the local context. Street is also referred to as Road and hence the abbreviation ‘St’ used for Street and the abbreviation ‘Rd’ used for Road should be referred as the same entity. Apart from this, there may be many abbreviations used at the local level. The term ‘M. G. Road’ is very common in cities in India. This term also is referred as ‘Mahatma Gandhi Road’ when full form is needed. Any of the string matching method does not match it perfectly being they differ syntactically. Rather in real sense the string difference between these strings should be treated as zero for perfect identification of the duplication.

In our implementation we created a database of shortcuts referred in Indian context. The database contains the words, their shortcuts. Synonyms of the word are generated by interfacing the wordnet dictionary to our software. Many implementations are available for Wordnet interface. One of the API available is HUNSPELL library. Our implementation uses HUNSPELL library API for wordnet data thesaurus, which is available free of cost for used and is discussed in chapter 2. For every synonym in the list, corresponding shortcut is selected and the string is compared using these shortcuts comprising of the shortcuts of the synonyms.

In our proprietary software, use of this facility is made available in indexing of records using different encoding techniques and as well as in string matching algorithms. Using the synonyms at indexing level gives the surprising results but a careful use is suggested. The field that has more chances of having thesauruses is required to be selected as otherwise there in unnecessary increase of time complexity of the algorithm for indexing.
3.7 THE ARCHITECTURE THAT IS DEVELOPED

We have developed architecture suitable for deduplication of records in distributed environment. A problem of deduplication applied for distributed databases raises a time complexity issue due to large volume of data available in the system.

The number of comparisons involved in the check increases the duration required for getting the duplicates. If every comparison is carried out at the process invoking terminal, the system with very high configuration is needed. Many organizations have data distributed at different campus and need the processing of data for deduplication. Hence instead of running the task at same machine, if it is divided at number of systems available, then this distributed processing will save the time. Use of web services helps you to access the library from the web server and needs no special installations of the software. Fig 3.5 describes the way of working of the system that we have developed. Distributed processing makes it convenient to send only the results to the invoking system which in tern does the consolidation task for the user.

Consider the network of n systems with one server. Let there be m database servers in the network. A web server is equipped with the software that we have developed. For deduplication task that is to be carried out here, the steps involved in this process are listed below:

- Consider a data entry at node n1. A record entered at node n1 is to be checked for duplication at all the databases available in the system.

- Node n1 will generate a message for all the nodes through a socket to all n identical nodes having databases, for the check of duplication. The message will have all the data needed for check of duplication.

- The target nodes n2 will start the process of duplicate detection after receiving the message from node n1. Details of the duplicate records detected so, will be transferred to node n1.

After receiving the data related to duplicate records from all target systems n, n1 in tern will consolidate the results and display it accordingly.
Fig 3.4: A Flow of System to Identify Duplicates using Ontology and Shortcuts in Distributed Environment

The algorithm so designed has many advantages in the view of resources required. The list of advantages is given below

1. System requirement is not a bottleneck
2. Processing is distribute on different systems
3. A large database can be stored on different system and the algorithm can be used. This will start distributed processing on available systems.
4. Type of database on the network is not a problem.
5. Location independent
6. No need of installation of libraries.
7. Single point control

REFERENCES:
5. Simple Object Access Protocol (SOAP), http://www.w3.org/TR/soap12-part1/
7. WSDL (Web Service Description Language), http://www.w3.org/TR/wsdl
10. Community Grids Lab (CGL), http://communitygrids.iu.edu