CHAPTER 1

INTRODUCTION

1.1 GENERAL

Advancements in the database management area led to the extensive use of databases as the primary means for data storage. The importance of developing a supporting infrastructure for data sharing has been understood during the last years. The increasing need for collaboration among organizations in a wide variety of domains, from manufacturing and service industry to scientific virtual laboratory and disaster management has necessitated this need. In order to facilitate and enable collaboration among distributed, heterogeneous, and autonomous organizations, one of the first requirements that need to be met is enabling access to certain data that is to be shared among the stakeholder organizations. However, before sharing of any data could possibly occur, many existing structural heterogeneities among the stakeholder database schemas need to be resolved.

Manual resolution of schema heterogeneities is time consuming. This becomes more challenging in large networks. Without automated ways of removing such heterogeneities between separate database schemas of participants, data interoperability and effective collaboration goals cannot be met. Consequently, provision of automated schema matching (Erhard Rahm & Philip Bernstein 2000) and integration tools is an active area of research with numerous technical challenges.
Schema matching is a crucial step in this data integration process. A schema matching operation takes two or more schemas as input and produces a mapping between the elements of the input schemas that correspond semantically to each other. This research work proposes to improve automated schema matching by developing a schema matching technique using an Efficient Centroid Based Clustering Algorithm and Hybrid Hill Climbing approach with Genetic Algorithm.

The data in information integration system, schema matching architecture and its applications, motivation and objectives of the proposed research work are discussed in the following sections of this chapter.

1.2 DATA AND INFORMATION INTEGRATION SYSTEM

Data integration is an earlier research fields in the database area and has emerged shortly after introducing database systems into the business world (Arnon Rosenthal & Len Seligman 2001). In today’s business world, the enterprises own a multitude of data sources for progress in large-scale scientific projects and other business opportunities. Employing these systems is a pervasive challenge for enterprises to realize business opportunities in highly competitive markets. In this setting, the integration of existing information systems becomes more challenge in application that need to query across multiple autonomous and heterogeneous data sources. In general, data integration aims at combining technical and business process used to combine data from selected systems to form new unified whole meaningful and valuable information. The reason for integration is twofold: an integrated view can be created to make possible information access and reuse through a single information access point. Second, data from complementing information systems is to gain more and complete information to satisfy the need.
There is a manifold of applications that get pleasure from integrated data systems (AnHai Doan et al 2000). For instance, within the space of Business Intelligence (BI), integrated information is used for querying and coverage on business activities, OnLine Analytical Processing (OLAP), and data processing so as to enable forecasting, higher cognitive process, enterprise-wide planning, and to realize competitive advantages (Surajit Chaudhuri & Umeshwar Dayal 1997). For Customer Relationship Management (CRM), integrated information on individual customers, business surroundings trends, and current sales is used to improve customer services. Enterprise Information Portals (EIP) present integrated company information as personalized internet sites and represent single data access points primarily not only for employees, however conjointly for purchasers, business partners and also the public.

Similar to information and IT services is integrated, either to supply a single service access point to meet business necessities via additional comprehensive services. May be, integrated advancement and document management systems is used among enterprises to leverage intra structure collaboration. Based on the ideas of Business Process Reengineering (BPR), integrated IT services and applications that support business processes can help to reduce time-to-market. Thereby, interconnecting building blocks from designated IT services and applications allows supply chain management among individual enterprises. Moreover as cooperation on the far side the boundaries of traditional enterprises, as in inter-organizational cooperation, Business Process Networks (BPN), and virtual organizations. For instance, in e-procurement, supply and demand for producer goods are supplied with integrated information and services to contour the purchasing process for institutional patrons. Thus, it is potential to bypass intermediaries and to change direct interaction between supply and demand, as in Business-to-Business (B2B), Business-to-Consumer (B2C), and Business-to-Employee
(B2E) transactions. These trends are fueled by XML that is turning into the trade normal for data exchange. Web services give ability on different software applications running with different platforms. Within the enterprise context, the integration problem is commonly referred to as Enterprise Integration (EI). Enterprise integration denotes the aptitude to integrate information and associate functionalities from a wide spread of knowledge systems in an enterprise. This encompasses Enterprise Information Integration (EII) that issues integration on the information and data level and Enterprise Application Integration (EAI) that considers integration on the level of application logic.

Many companies gather a large amount of data about customers, products, sales, suppliers, etc. Much of those data is stored in relational databases, which provide a uniform interface (or schema) that is used by multiple software application to access the information. Unless standard products are used, the database schemata can be designed to suit the specific requirements of the company. The schemata can closely reflect the view of the company on the business domain and be optimized with respect to their applications. While the liberty that the database designers have may benefit each individual company, it creates additional problems when databases need to be integrated:

Assume a merger of two companies with the goal of generating positive synergy effects. Although some departments might be unaffected, the company can benefit from the merging of many operative areas, e.g., customer relationship or procurement. This process also involves the integration of existing databases, which is known to be an expensive process due to various forms of heterogeneity, which are the result of autonomous development of the data sources.
One way to bring together the information from the databases is to write new software that uses the existing data sources as shown in Figure 1.1. That application would query each data source individually and combine the retrieved results. While this solution is technically feasible, it is not desirable because in most IT infrastructures several software applications use the databases. If another application that requires data from all databases has to be developed, a lot of integration work (i.e., the resolution of heterogeneity) has to be redone. In addition, inconsistencies may arise if the applications are run concurrently. To avoid such problems, an Integrated Information System (IIS) is placed between the applications and existing data sources which are shown in Figure 1.2. An integrated information system presents a uniform interface, which can be used by several applications to access the sources. The creation of such applications is not as complex as in the previous case, because only the IIS and not all sources needs to be understood by the developers. In addition, the IIS can provide regular DBMS services, e.g., concurrency control.
The following subsection discusses the various existing problems in Data integration approaches.

1.2.1 The Problem of Data Integration

Integration of multiple information systems typically aims at combining selected systems so they sort a unified new whole and provides users the illusion of interacting with one single information system. Users are provided with a standardized logical view of data that is physically distributed over diverse data sources. For this, all data have to be represented using the same thought of principles (unified global data model and unified semantics). This task includes detection and resolution of schema and knowledge conflicts relating to structure and linguistics.

In general, information systems are not designed for integration (Alexe et al 2009). Thus, whenever integrated accesses to totally different source systems are desired, the sources and their information that do not work
along ought to be united by further adaptation and reconciliation practicality. This is not an integration setback whereas the goal is usually to provide a homogeneous, unified view on information from totally different sources; the actual integration task could rely on:

- the architectural view of an information system (see Figure 1.1),
- the content and functionality of the element systems,
- the kind of information that is managed by component systems (alphanumeric, multimedia, structured, semi-structured, unstructured data),
- requirements regarding autonomy of component systems,
- intended use of the integrated information system
- performance necessities
- The on the markert resources (time, money, human resources, know-how, etc.)

Several heterogenous problems are also to be considered. They are

- hardware and operating systems,
- data management software and package,
- data models, schemas, and data semantics,
- middleware,
- user interfaces, and
- Business rules and integrity constraints.

1.2.2 Approaches to Integration

In this section, an architectural perspective is applied to present an outline of the various ways to address the integration problem. The presented
classification distinguishes integration approaches in step with the extent to abstraction wherever integration is performed (Saha et al 2010).

An information system (Seligman et al 2010) is represented employing a layered architecture, as shown in Figure 1.3. On the top layer, users access information and services through varied interfaces that run on the top of various applications. Applications might use middleware - Transaction Processing (TP) monitors, Message Oriented Middleware (MOM), SQL-middleware, etc. - to access data and information via a data access layer. The data itself is managed by a data storage system. Usually, DataBase Management Systems (DBMS) are used to merge the data access and storage layer.

In general, the integration problem is addressed on every of the presented system layers. For this, the subsequent principal approaches as illustrated in Figure 1.3 are available:

- **Manual Integration**

  Here, users directly interact with all relevant information systems and manually integrate selected data. That is, users need to deal with totally different user interfaces and query languages. In addition, users have to be compelled to have detailed knowledge on location, logical data illustration, and data semantics.

- **Common User Interface**

  In this case, the user is furnished with a typical common user interface (e.g., a web browser) that gives a uniform look and feel. Data from relevant information systems on an individual basis conferred in order that homogenization and integration of data has to be done by the users (for instance, as in search engines).
• **Integration by Applications**

This approach uses integration applications that access numerous data sources and return integrated results to the user. This solution is sensible for a tiny low range of component systems. However, applications become more and more fat because the range of system interfaces and data formats to homogenize and integrate grows.

Figure 1.3  General Integration Approaches on Different Architectural Levels

• **Integration by Middleware**

Middleware provides reusable practicality that is generally used to solve dedicated aspects of the integration problem, e.g., as done by SQL-middleware. Whereas applications are reassured from implementing common integration functionality, integration efforts are still required in applications. Additionally, totally different middleware tools sometimes got to be combined to build integrated systems.

• **Uniform Data Access**

In this case, a logical integration of data is accomplished at the data access level. Global applications are supplied with a unified global view of physically distributed knowledge, though only virtual data is accessible on
this level. For alternative applications, local information systems keep their autonomy and may support additional data access layers. However, global provision of physically integrated data may be time-consuming, long since data access, homogenization, and integration got to be done at runtime.

- **Common Data Storage**

Replacement data storage is the area where the physical data integration is done by the transformation of data; the local sources will be retired or will stay operational. Usually, data access is provided by physical data integration. However, applications that access the local data sources will have to be migrated to the new data storage if the local data sources are retired. Periodical refreshing of the common data storage must be considered in case if the local data sources stay operational.

Based on the presented general integration approaches, concrete integration (Gross et al 2010) solutions are realized. Few important examples include:

- By providing a single point for read-only querying access to various data sources, e.g., as in TSIMMIS, mediated query systems represent a homogenous data access solution. A mediator that contains a global query processor is used to send sub queries to local data sources; then the results of returned local query are combined.

- Uniform data access which is being another form of portals are personalized doorways to the internet or intranet where information is provided to each user according to his detected information needs. Normally, by using click-stream analysis web mining is applied to determine user-profiles; therefore,
the interested information of each user can be retrieved and presented.

- A common data storage approach to integration is used by data warehouses. Data warehouse is a place where data from several operational sources (On-Line Transaction Processing Systems, OLTP) are extracted, transformed, and loaded (ETL). On-Line Analytical Processing (OLAP), are analysis which can be performed on cubes of integrated and aggregated data.

- Second example of common data storage is operational data stores where, a “warehouse with fresh data” is built by updating the immediately propagating data, in local data sources to the data store. Thus, for a proper decision support, there is availability of up-to-date integrated data. Like in data warehouses, data is not cleansed or aggregated or the data histories supported.

- By logically integrating data from the underlying local DBMS, the Federated Data Base Management Systems (FDBMS) achieve a uniform data access solution. These are full-fledged DBMS; which means, the data model, supporting global queries, global transactions, and global access control are implemented on their own. Normally, for building FDBMS the five-level reference architecture is employed.

- Work Flow Management Systems (WFMS) is one in which each single step is executed by a different application or user, that allows implementing business processes. Generally, interactions between applications and human users are supported by WFMS modeling, execution and maintenance of
processes. WFMS also represents an approach called integration-by-application.

- Integration is performed by integrating web services through software components (i.e., web services) which supports the interaction between machine-to-machine over a network that are conveyed by internet protocols by XML-based messages. Web services either represent a uniform data access approach or a common data access interface, depending on their offered integration functionality for manual or application-based integration.

- A high-level operation is introduced by model management between models (such as database schemas, UML models, and software configurations) and model mappings; these operations include matching, merging, selection, and composition of data. It is intended to reduce the amount of hand-crafted code required for transformations of models and mappings using a schema algebra that encompasses all the above operations as needed for schema integration. Model management comes under the category of manual integration.

- Data can be mutually shared and integrated through mappings in a Peer-to-Peer(P2P) integration between local schemas of peers, which is a decentralized approach to integration between distributed, autonomous peers. P2P integration constitutes either a data access approach or a data access interface for subsequent manual or application-based integration, depending on the integration functionality that is provided.

- In a grid environment, i.e., interconnected computing resources being used for high-throughput computing, the grid
data integration provides the basis for hypothesis testing and pattern detection in large amounts of data. To provide an integrated view over large (scientific) data sets often unpredictable and highly dynamic amounts of data have to be dealt with. By using middleware approach, grid data integration represents integration.

- Personal data integration systems are manual integration which are in a special form. By dedicated integration engineers or by users themselves a tailored integrated views are defined (e.g., by a declarative integration language). As intended by the particular user, each integrated view precisely matches the information needs of a user by encompassing all relevant entities with real-world semantics; thereby, the integrated view reflects the user’s personal way to perceive interest in his application domain.

- Another special form of manual integration, called collaborative integration is based on the idea to contribute to a data integration system to users for using it. Here, users answer questions concerning the mappings by a initial partial schema mapping that is presented; To expand the system capabilities and to refine the mappings these answers are used. The task of schema mapping is distributed over participating users which is similar to folksonomies, where data is collaboratively labeled for later retrieval.

- Co-existence of all data (i.e., both structured and unstructured) is propagated rather than full integration in a data space systems. It is used to provide the same basic functionality, e.g., search facilities, independently of their degree of integration over all data sources. Additional efforts are made
to integrate the required data sources only when more sophisticated services are needed. In general, the data space systems may simultaneously use each of the general integration approaches that is presented.

Different forms of heterogeneity that can be observed in such scenarios are discussed in the following sections. Also, discussion on different ways of integrating information sources is done, and showed that in the development of such systems schema matching is an important step.

1.2.3 Heterogeneity and Conflicts in Data Integration

For several decades data integration has been a research area in computer science under various names: multi database systems (Witold Litwin et al 1990) federated database systems (Amit Sheth & James Larson 1990) mediator-based systems (Gio Wiederhold 1992), data warehouses (Surajit Chaudhuri & Umeshwar Dayal 1997) and peer database management systems (Alon Halevy et al 2003). All of them have to deal with heterogeneity, even though there are different types of integrated information systems that differ with respect to their level of coupling, materialization, etc. When data sources are autonomously developed, heterogeneity arises. Widely accepted standards must be followed, in order to avoid heterogeneity. However, in most of the domains such standards do not exist or not sufficient. Thus, to develop the standard to a specific need or to create their own domain model the developers have to customize them. As a result, in the creation of an integrated information system one inevitably has to be dealt with heterogeneity.

There are various classifications of heterogeneity in the literature (Susanne Busse et al 1999) (Won Kim & Jungyun Seo 1991), where in four types of heterogeneity are distinguished:
1. Technical heterogeneity

2. Data model heterogeneity

3. Structural heterogeneity

4. Semantic heterogeneity

Technical heterogeneity is concerned with, hardware, operating system, or networking infrastructure which are issues of technical nature. With existing middleware technology conflicts resulting from technical heterogeneity can be solved. Furthermore, when data is stored in various formats, data model heterogeneity has to be resolved by the integrators. Structured information (e.g. relational or hierarchical data model), semi-structured data (e.g. XML) or unstructured documents should be able to be integrated by an integrated information system. By wrapping the data sources conflicts resulting from data model heterogeneity are usually resolved. Each source wrapper is translated between the data model of the IIS and the data model of the information source. Wrapping web sources is an active research topic (Valter Crescenzi et al 2001).

Structural heterogeneity derives from the fact that, information can be structured in various ways by giving a data model. Problems arising from structural heterogeneity are, data-metadata conflicts (a piece of information which is an attribute value in one data source is also an attribute or table name in another data source) or partitioning (different groupings of attributes into tables). Due to semantic heterogeneity various other sorts of conflicts occur. The schemata under consideration might be different even when the data is structured in the same way, e.g., attributes might have varying labels. Naming conflicts occurs as a result of using homonyms (same word having a different meaning) and synonyms (different words having the same meaning) when
defining the attribute names. Structural heterogeneity and semantic heterogeneity results in differences between the schemata under schematic heterogeneity. Data conflicts occur when the same information is represented in different ways, e.g. different formats, abbreviations, or acronyms, which is also a result of semantic heterogeneity. In addition, increase in complexity of data integration occurs in some tables which contain erroneous, contradictory, or missing data.

Figure 1.4 shows an extract of an integration scenario with a single source table R and an integrated schema S, which consists of two tables S₁ and S₂. Here, the integrated schema is called target schema. Source table R contains employee data, which includes the name, salary, age, department, and head of the department. In the target schema also similar information can be found, but structured as two tables S₁ and S₂. Data about the departments, including the name and head of each department, is shown in table S₂. Employees are represented in table S₁ by their name, salary, and a foreign key to the tuple representing the department in which they work. Foreign key is shown as dashed arrow, while a semantic relationship between attributes of the two schemata is indicated by solid arrows.

Several examples for structural heterogeneity can be found here: First, the employee information is structured as a single table in the source, but the target schema contains two tables. Second, the name of each employee is represented in two attributes in R, while the target table S₁ has a single attribute containing both the given name and surname. Third, the source table R has an attribute Age, which does not have a corresponding partner in the target database.
In both schema and instance levels, semantic heterogeneity occurs. The attribute that represents the head of the department is labeled Dept Head in R and DHead in S₂. DeptHead and DHead must be identified as synonyms, in the process of developing the integrated information system. Both R and S₁ contains an attribute Salary, that represents the salary of employees. However, both attributes use different currencies: The salary is shown in Euro in R, while S₁ uses US dollars. Finally, the result of semantic heterogeneity is the data conflicts in duplicate tuples. It is shown that the employee Sam Adams is represented both in the source and the target database. However, after translation of salary in the target database into Euro, it is seen that the salary of Sam Adams is much lower than in the source database. For this conflict, there are several possible reasons e.g., Sam Adams has received a major pay raise which is not shown in the target database, or the salary in either of the database has been printed wrongly, etc. The resolution of heterogeneity is a labor-intensive process. Few standard solutions exist for some types of heterogeneity. Middleware applications can be used to bridge different technologies, as stated above. By the existing wrapper technology data model heterogeneity can be resolved. But, for structural and semantic heterogeneity there is no automatic solution. Two research areas are concerned with these forms of heterogeneity they are: schema mapping and duplicate detection.
The goal of schema mapping which is the semi-automatic construction of mappings, (Shvaiko & Euzenat 2005) describes the structural and semantic relationships between two schemata. A mapping is constructed within two steps: schema matching (i.e., the detection of attribute correspondences) and query discovery (i.e., the generation of a mapping based on the detected correspondences). Attribute correspondences (shown as solid arrows in Figure 1.4) connects the semantically related attributes. The goal of schema matching solutions is to find such attribute correspondences that have to deal with structural and semantic heterogeneity. The query discovery process determines a schema mapping, based on the detected attribute correspondences. The process is more challenging, because there is possibility of several interpretations of a set of correspondence, and resolving ambiguities. The process of identifying different representations of the same real-world entity is called duplicate detection. An example for such a duplicate is the two representations of Sam Adams in Figure 1.4. By data conflicts duplicate detection is complicated.

1.2.4 Types of Integrated Information Systems

Different types of integrated information system exist as mentioned above. Various types of Integrated Information System such as mediator-based systems, data warehouses, peer data management systems, and multi-database systems are described, which is as follows. Schema mapping which is also shown, resolves structural and semantic heterogeneity, and is an important part in most of those systems. The following, explains what mappings is and how they are used by different systems.

1.2.4.1 The Mediator-Based System (MBS)

There is a need to decide whether the data is to be stored in the data integration layer, or whether the integrated information system uses the
source to answer a user query, when developing an integrated information system. The mediator architecture (Figure 1.5), is an example for virtual integration which was originally described by Wiederhold (Gio Wiederhold 1992): At the data integration layer, the mediator uses the data sources to answer queries, instead of storing the data in a database. The mediator translates the user query into many queries that are sent to the sources, when the user sends a query. Then the retrieved results are merged and presented to the user.

There can be different types of sources e.g., relational or XML database systems (Alergawy et al 2009), web sites, etc. Data model heterogeneity is resolved by wrappers, which receive a query from the mediator and translate it into a query that is understood by the source, as stated above. Later the wrapper retrieves the result and converts it into the canonical data model, i.e., the data model which is used by the mediator.

![Figure 1.5 The Mediator Architecture](image)

Mediator-based systems have the advantage that only the user needs to understand the schema and data model of the mediator like all other integrated information systems. The details of the sources from the user are
completely hidden by the mediator layer. In addition, the sources are not affected by the mediator; instead they can be used in the same way as they have been used before the mediator was in place. The autonomy of the data sources is retained as the mediator acts as another database application. But, this raises additional challenges. The mediator and its mappings need to be adapted whenever a data source is modified. This provides for modifications of the schema, data model, or technical infrastructure of the data source. Such change having automatic detection is an active research area (Kristina Lerman et al 2003), (Robert McCann et al 2005). In contrast to this, adding, removing, or modifying data does not affect the mediator rather, the user always receives an up-to-date result which ensures the use of virtual integration.

1.2.4.2 The Data Warehouse

To obtain a global view of the company, the management of a large enterprise requires decision support systems. Online Analytical Processing (OLAP) applications are used (Surajit Chaudhuri & Umeshwar Dayal 1997) to extract relevant information about the performance of the enterprise. These applications are able to process large amount of data and extract hidden knowledge, e.g., purchasing patterns of customers. The data used by OLAP tools comes from a data warehouse, which in turn receives data from many operative databases and applications within the company.

Data warehouses perform materialized integration in contrast to mediator-based systems. The largest data warehouses store several tera bytes of data, and accessing the sources every time a user sends a query, clearly does not scale, because of huge amount of data. New data is copied in a regular interval from the sources to the warehouse, and the user queries are processed on that local database. Data is extracted from the operative databases and transformed into the warehouse schema in the Extract-Transform-Load (ETL) process, before it is loaded into the data warehouse.
The ETL process is usually performed when the load on the operative databases is very low, because it is time consuming and also requires a lot of resources. Later, the user can send queries even when the operative databases are heavily used, because the data warehouse uses its own copy of the data to produce a query result. It is noted that the warehouse data is not always up-to-date but, because of the nature of OLAP queries, the quality of the query answers is not affected. The user is usually interested in aggregates, e.g., sales per month, and the difference between the last warehouse update and the current state of the operative databases has only a little effect on the query result.

In ETL process, the schema design of operative databases and warehouse database follows different design principles, since it usually involves complex schema transformations. The schemata of the operative databases are designed and optimized with respect to their local applications. To avoid redundancy it closely reflects the application domain and follows normalization rules. A typical design pattern is followed by the warehouse schema, e.g., star schema or snowflake schema (Surajit Chaudhuri & Umeshwar Dayal 1997). The ETL process involves schema mappings that describe the transformation of source data such that the data can be used in later stages of the process.

1.2.4.3 Other Types of Integrated Information Systems

In the last decade peer-to-peer (P2P) systems have become a very active research area (Neil Daswani et al 2003), (Stephans Androutsellis-Theotokis & Diomidis Spinellis 2004). The main purpose of P2P networks is file sharing, so several efforts have been taken to apply the peer-to-peer concept in data management (Alon Halevy et al 2003), (Ryan Huebsch et al 2003), (Wee Siong Ng et al 2003) (Armin Roth et al 2006). There is no
distinction between client and server in a P2P network; instead, all nodes in the network act as both consumer and provider of information.

P2P file sharing considers very large networks (i.e., hundreds or thousands of nodes) on research; the peer data management (PDMS) is also conceivable on a smaller scale. Figure 1.6 describes a schematic representation of a PDMS network of a global company. Five databases are connected in a PDMS network, to facilitate information exchange between different locations. Each node acts both as a client and a server, i.e., a user in Berlin can send a query based on its local schema, and the result also contains data from other four locations. In the Figure the arrows indicate schema matching, which are used for query rewriting. It is noted that even though the mappings are directed, queries can be forwarded in any direction using either GaV or LaV query rewriting (Igor Tatarinov & Alon Halevy 2004).

Figure 1.6 A Peer Data Management System

Techniques developed in the context of mediator-based systems, e.g., GaV or LaV query rewriting, can also be applied in PDMS networks.
Due to more general architecture additional problems arise. The mappings on the path need to be combined (Jayant Madhavan & Alon Halevy 2003), (Ronald Fagin et al 2004), if a query has to take several hops from the query node to the network node that contains the relevant data. Some relevant attributes in the composed mapping might be lost, because the schema of an intermediate node does not contain those attributes which affects query routing. The path, the query takes from the user to the source is predefined in mediator-based systems. Several distinct paths are possible in a PDMS network. Finding a good path that avoids information loss is a challenging problem (Karl Aberer et al 2003). Problems can arise when the network is assumed to be unstable, i.e., if node can enter and leave the network dynamically, besides mapping composition and query routing. Such a system where every time a node enters a network, would require (semi) automatic creation of semantic mappings.

Multi-database systems do not require predefined mappings since they differ from the previously discussed architectures. But, they provide a query language which allows the user to send a single query that involves multiple sources. The user interacts with each source and does not use a global schema. In contrast the multi-database system provides a query language that can be used to describe, querying Internet sources individually and merging the results manually, in a single expression. This process requires the user to know about the semantic relationships between the sources, also in order to specify a reasonable query the user needs a mental model of the mappings. There may be a contradiction that since multi-database systems do not provide an integration layer, they are no variant of integrated information systems.

An integral part of various types of integrated information system is schema matching that describes the semantic relationship between
heterogeneous schemata. The database integration system is built so that the creation of mappings is closely bound to the principle approach. Therefore the Schema matching principles, issues and applications are described one by one in the following sections.

1.3 SCHEMA MATCHING

The use of databases as the primary means for data storage has increased due to the advancements in the database management area. This enriched the development of independently designed heterogeneous databases, to store their own organization specific data, by institutions and organizations. Thus there is an increasing growth in importance and complexity in reconciling these heterogeneous data sources. In this data integration process, schema matching is a crucial step. A schema matching operation produces a mapping between the elements of the input schemas, by taking two or more schemas as input that correspond semantically to each other.

In application domains like data warehousing, E-commerce and querying data over multiple data sources schema matching (Bellahsene et al 2011) is a fundamental operation. Schema matching is used to postulate a semantic correspondence between the heterogeneous data sources, in Data Space Management Systems (DSMSs). Domain experts perform manual schema matching operations, this can be a costly operation due to high labor costs for large data sources. Thus it is a labor intensive process especially for large data sources because the domain expert manually searches for all possible element matches. The data sources involved are also prone to frequent changes, inculcating these new changes to the schema mapping that is already generated, is very important to avoid incorrect or even failed integration. Therefore manual schema matching is very brittle in the underlying data sources with respect to changes. Hence considering these
reasons, schema matching should be fast and reasonably accurate with less or no human intervention. Therefore, using available information about schema elements, their data types, schema structure and the data instance available, it should be possible to perform an efficient schema matching.

The schema matching (Gal 2006) complexity is because of inherent differences between the participating data sources, such as the fact that various organizations have various preferred database management systems (DBMS) software for their organizational needs. For example, one organization may use MySQL DBMS while another Oracle and also within the same organization the choice of DBMS can be different even for different applications. Even though there is diversity in the choice of DBMSs, as dictated by application requirements the data sources can opt to store data in different formats. Few may use structured representation for storing data, like relations and tables, while others may opt for semi-structured representations like XML, or else they use graphs, or trees. For example, one data source uses XML documents to store data, while another may do this in the form of a relation table. Though there is a lot of diversity in the data storage and representation, schema matching must handle this very precisely.

Creation of heterogeneous data sources is done by different people. These people can choose to assign names to schema elements in different ways and can have different design for the same concept. Because of the diversity of representations of the same concept, the complexity of schema matching is further increased. For example, the creator of one data source can have table name ‘STUDENT’ which he chooses to store student information in the form of relation, while the creator of another may name it 'STUDENT INFO' or 'ENROLLMENTS'. From the use of synonyms schema matching algorithms must identify this diversity in element names such as stems and
similarity scores between schema elements is generated by hyponyms by performing name-based schema matching.

The same information can be stored by data sources in more than one entity. Storing student information in relation name 'STUDENT' is done in one data store, while another data store for the same information can split in two relations as 'MSC STUDENT' and 'PHD STUDENT' as dictated by the application requirements. Horizontal partitioning of the data over multiple relations is an example for this. Similar diversity can also be found in the attributes of information stored in different data sources, in representation, naming and splitting of data over a number of entities. For example, the creator of one data source uses 'HOUSE NO', 'STREET', 'CITY', 'COUNTRY', 'POST CODE' can choose to store student addresses in separate attributes, while another creator can choose to have only two attributes 'ADDRESS' and 'POSTAL CODE', which is an example of one-to-many partitioning of data where an attribute in one data source is partitioned over multiple attributes in another data source.

Heterogeneous data sources also differ while representing the domain information. The data type assigned to a column, the length of the values and the constraints on them is included in the domain information. Data types and constraints are system specific and therefore they can vary from one data source to another data source. This diversity in domain information should be tackled by schema matching algorithms and use it to perform domain-based schema matching.

The schema catalogues stored by a DSMS is used to extract all the information about a schema. It contains information, such as table structure, number of columns, column names, data types and lengths of the columns and the constraints on the columns. The structure information of elements is used
by structure-level schema matching, to generate correspondence between
them.

Catalog information can be used to perform element-level and
structure-level schema matching. The outcome of statements (such as SQL
CREATE) used in the Data Definition Language (DDL) can also serve as a
source of information to perform element-level and structure-level schema
matching. In data sources where neither catalog information nor DDLs
statements are available, data instances can be used to extract information
about the schema and this instance information can be used to perform
instance-level schema matching.

The definition of schema, architecture of schema matching and its
applications are discussed in this section.

1.3.1 Schemas

There are different kinds of schemas, such as relational database
schema, XML schema, domain conceptual schema, etc. Moreover, there are
various ways to represent them. For instance, XML schema can be outlined
by such standards as DTD, XML Schema, Relax NG, etc. However, neither
form of schemas nor their representations should to be associate obstacle for a
general discussion. The various definitions of “Schemas” and its application
are given below.

**Definition 1.1:** Set of elements of schema is (element types, attributes,
simple types, etc.) connected by some structure.

Definition 1.1 implies that schemas into account have a graph
structure. Generally graphs ought to be transformed into trees. For instance, in
XML schema a shared element type definition is transformed into several
local element type definitions. Even though schemas are graphs, sometimes the structural properties are ignored. For instance, within the holistic paradigm the schemas are typically considered as unstructured sets of components.

Each node in the graph will contain some meta-data connected, for instance, variety restrictions, optionality, list of synonyms, etc. This information can be oppressed by matching algorithms. In Figure 1.7, two probable schemas for purchase orders are showed. The tree structure of schemas should not be mixed with the tree structure of grammatical XML documents, as a result of instance documents would have actual values pictured as nodes of the tree.

![Figure 1.7 Purchase order schemas $S_1$ (left) and $S_2$ (right)](image)

There are three varieties of schemas mentioned: input, source, and target. Any schema within the matching process is referred to as input schema. The term source schema is typically used as a synonym for input schema, however typically it means that there is a target schema to which the source schema has got to be matched. Once the matching is not directional the source and also the target is not necessary, because any schema can be either a source or a target.
In a match operation, the input schemas require the elements to be matched. Therefore, it is instructive to study specific typical schemas and their elements. Based on the application domain, structure of the relation may be available in many different formats and languages, such as SQL, UML, DTD, XSD, and OWL. Figure 1.8 shows sample schemas for purchase order in SQL, XSD, and OWL, respectively.

- **SQL** allows defining schemas for relational databases, to query and handle data stored in a schema. A relational schema comprises a set of tables. Object instances are warehoused as records of column values within a corresponding table.

- **XSD** is progressively used to define structure of XML documents for data exchange over the Web. The main modules of an XSD schema are elements.

- **OWL** is usually used for specifying ontology on the Semantic Web. Ontology objects at abstracting the knowledge of a domain and supports a semantic-richer representation of the real world than database or document schemas. OWL classes may have instances that are stored within the same XML document.

![Sample schemas for purchase order](image)

*Figure 1.8 Sample Schemas for Purchase Order*
A schema is well-defined as a group of schema elements connected by some structure. For example, from a relational database schema, the tables and columns can be extracted as schema elements, and control relationships between tables and attribute and referential constraints expressed by foreign keys between the tables. In an XSD schema, schema elements include XML instances and attributes, while schema structure entails of control relationships between element and sub-elements as specified by complex types. From an OWL ontology, classes and properties are obtained as schema elements, while relationships among classes and containment relationships between classes and their properties constitute structure.

1.3.1.1 Input Information

In order to solve a given match problem, employing all kinds of information available helps to characterize the semantics of schema elements and to detect their similarity, it differentiate as schema information, instance data, and auxiliary information:

- **Schema information**: Schemas offer different types of information, such as element names, description, data types, schema structure and other relationships are examined to depict and associate the semantics of schema elements.

- **Instance data**: In different applications, that is data integration and transformation; instance data will exist for the schemas to be coordinated and can also be exploited to characterize the content and semantics of schema elements.

- **Auxiliary information**: This grouping contains all further varieties of information which can be exploited to detect similarity between elements.
1.3.1.2 Output Information

Specified two input schemas S₁ and S₂, the match operation returns as output a mapping between them, also called the match result. A mapping is defined to be a set of mapping elements, each of which agrees that certain elements of S₁ are mapped, certain elements in S₂. Each correspondence may have a mapping expression, which specifies how the S₁ and S₂ elements are related with each other.

Semantics: The expression of mapping use simple relations over identity, the study of nomenclature relationships set-oriented relationships or functions. Functions are the form of semantic correspondences, as they specify how to convert instances of S₁ elements in those of S₂ elements. For the specification of mapping expressions different languages can be used.

- **Directionality:** Mapping expressions can be expressed directional or unidirectional. For indicating equality relations, between scalars, synonymy between terms, equivalence of sets, etc., are called unidirectional, while remaining are typically directional.

- **Invertibility:** Expressions of mapping may or may not be invertible. By perception, unidirectional expressions are the inverse of themselves, and thus invertible. However, directional ones may or may not be invertible. While expressions specifying 1:1 communications are often invertible, complex ones connecting collections of schema elements A mapping is invertible if all mapping expressions of its communications are invertible, otherwise uninvertible.
Automated schema matching based on heuristics that are not easily taken in a precise mathematical way (Chuang & Chang 2008). The key aim is to produce a mapping that attempts to understanding of what users reflect to be a best match. Result of matching mostly comprises of corresponding schema elements but does not accurately specify how the elements are related with each other, without mapping expressions. First step of schema matching is creating a semantic mapping between elements of two schemas. The second step is to improve the identified correspondences with real mapping expressions.

1.3.2 General Schema Matching Architecture

To understand the schema matching task, first the type of information is identified that the matching tasks taken as input and produces as output. According to Shvaiko & Euzenat (2005), the input in schema matching is a pair of schemas that are not homogenous. The output is given in the form of matching which isolate elements from these two schemas that have a semantic relationship. The general concept of schema matching is illustrated in Figure 1.9 below.

Figure 1.9 General Schema Matching Architecture
In general, the input information of a matching problem may include any type of knowledge about schemas to be matched, such as their instances and their domains. In the matching approaches, the primary input data consists of two schemas (i.e., source and target XML schemas) defined above as $S_1$ and $S_2$. The input schemas contain different information types such as names, descriptions, data types, and constraints. They are modeled as schema trees for the matching process. To analyze the linguistic contexts in the schema tree, English words are used (including their abbreviations and acronyms) and also the concatenated words (e.g., Address, Purchase Order, and Ship Location) in the labels of nodes.

To increase performance of semantic matching, several matching algorithms often require auxiliary information such as thesauri and dictionaries. The main auxiliary information used is WordNet, an electronic lexical database where relations such as homonymy are available to relate English word meanings. Other domain-specific dictionaries can be used in the case where used words do not belong to WordNet (Cruz et al 2009).

The outcome of the matching process typically provides similarity score ranging in $[0, 1]$ interval, where 1 (or 0) means that two matching pairs are exactly the same (or totally different). In the case of multiple matching results, two approaches were proposed. The first is to produce a collection of k best matching candidates for each matching pair, called top-k matching, ranked according to their semantic similarity scores. The second is to produce all possible matching pairs based on the given matching cardinality. An element from one schema can participate in zero, one, or several matching correspondences with elements from the other schemas (hereafter called 1-to-1, 1-to-n or n-to-1, or n-to-m, respectively). Note that a top-k matching is a
special type of 1-to-n matching, because an element from one schema can be
matched to k elements from other schemas at most.

Some of the schema matching applications are discussed in the
following section.

1.3.3 Applications of Schema Matching

To emphasize the importance of schema matching operation, this
dissertation presents some of the important applications of schema matching
(Erhard Rahm and Philip Bernstein 2001) as follows,

- Data space Management systems

Heterogeneous data sources, though different, are assumed to be
modeling same concept at least partially. Pay-as-you-go data integration of
such heterogeneous data sources is one of the primary functions of DataSpace
Management Systems (DSMS). DSMS is in response to the high upfront cost
associated in the creation and maintenance of near perfect integrated sources
and their low adoption rate to changes in data sources. DSMS provides agile
data integration with low upfront and maintenance cost by extensive use of
automation in bootstrapping stage and use of user feedback for improving the
quality of data integration over time. Integration of independently developed,
heterogeneous data sources, require postulating schematic correspondence
between elements of data sources and derivation of schema mappings from
this schematic correspondence. Automatic schema mapping using model
management techniques is a crucial step for DSMS. This process of
postulating schematic relationship between data sources is called schema
matching for DSMS.
- **Schema Integration**

  Schema integration is the primary motivation to perform schema matching. Schema integration is the process of automatically creating a global view of multiple independently developed schemas. To create global view of independently developed schemas, integration process must identify relationships between schema elements of the data sources (Christine Parent & Stefano Spaccapietra 1998). Data sources as modeled independently by different people will inherently have diversity in their schema representation; schema integration process must resolve this diversity in schema representation to create schema element mappings. Once schema mappings are created mapped elements of the data sources are combined to create a unified global view. For example, schema integration is required to combine the search results fetched from multiple data sources to provide a final unified search result.

- **Data Warehouse**

  Data warehouses are used for analytical processing to facilitate better decision making. Data warehouses are building by an Extract Transform and Load process (ETL) which extracts data from multiple data sources, transforms data from these data sources into data warehouse format and then loads data in data warehouse. Transformation processes require matching elements from data source to elements in data warehouse. This matching operation is schema matching. Automatic schema matching can generate initial elements mapping without any human input. Data warehouse designers can then confirm this initial mapping and reconciles the semantics of the data sources with that of the data warehouse.
• **E-commerce**

Advancement in networking technologies and global connectivity facilitated by World Wide Web allowed trading partners to communicate with each other easily. Trading partners exchange messages that represent their business transactions. As each trading partner's system was developed independently, it is common for them to use different message formats. Example: different XML message schemas or custom data structures. To allow trading partners to exchange messages, application must be able to transform messages from one format to another. This transformation of messages requires to perform matching between schemas of messages. This transformation is largely done by application designers manually. Thus, automating schema matching will reduce this manual mapping task by proving application designers with initial mappings, which application designers can validate or modify.

• **Semantic Query Processing**

Semantic query processing is another valuable application of schema matching operation. In semantic query processing user describes the output of a SQL query using the SELECT clause; this query out will be in terms of concepts familiar to user which may not be same as the elements of the database against which query need to be processed. To process such a query request, system needs to first figure out correspondence between elements of the schema and user specified concepts. Similar semantic mapping needs to be done for WHERE clause of the SQL query. For semantic query processing system need to perform schema matching at run time.
• **Semantic Web**

Web makes are finding difficult to locate, organize, and integrate information of interest. Because of enormous quantity of data, human users have no other choice but to hang on on the backing of computers to accomplish these tasks. However, maximum of the Web’s content today is considered for humans to read, but not for machines to interpret and manipulate. The idea of the Semantic Web is to improve the current contents of Web pages with semantic explanations, which can be understood by computer programs. The improvement is done by annotating the Web contents with ontologies, which, in the Web terms, are documents formally defining the semantics of and relations between concepts.

Given wide application of schema matching across many domains, schema matching operation is thus very critical and should be generic. Some of the open issues related to schema matching are given in the forthcoming section.

### 1.3.4 State of the Art and Open Issues

Schema matching need in varied applications (Do & Rahm 2007) led to develop many techniques and prototypes. The research work indicates the high potential of techniques and algorithms for schema matching. However, it has number of issues, which are not addressed in earlier work and thus need further examination:

- **Expressiveness of Modern Schema Languages**

  Recent schema languages may support different advanced modeling capabilities leading to important problems of schema matching. Mostly, such design styles can be used instead of to model the identical or similar real-world concepts (Cruz et al 2009), leading to different, yet semantically
similar, and structures in the schemas. Schemas matching are taking advantage of powerful languages.

- **Dealing with Large Schemas**

  Schemas are constantly growing in terms of size and complexity. Furthermore, the schemas regularly use shared elements to avoid unnecessarily diverse specifications and retain a low schema complexity for easier care. At the end, the match operation wants to inspect a vast search space to discovery plausible correspondences; a major challenge, which requires very efficient approaches to deal with.

- **Combination of Match Algorithms**

  To achieving great matching accuracy for large different schemas, considering a single criterion is unlikely to achieve success. As a consequence, it is a need to associate and use multiple techniques at identical time. For this purpose, earlier techniques have followed either a questionable hybrid or composite combination approaches of matching.

  By distinction, a composite match approach combines the results of many independently executed match algorithms, which might successively be hybrid or composite. This enables for a high flexibility, as there is the potential for choosing the match algorithms to be executed based on the match task at hand.

- **Schema Matching Evaluation**

  For finding a solution for a specific match problem, it is important to know which of the best available techniques exist. The only way to approach this goal is to validate the quality in real-world scenarios. (Qian et al
Evaluation thus represents an important task in developing a match solution and has also been extremely considered in earlier techniques.

- **Reuse of Match Results and Data Integration**

  Aim of reuse at exploiting different kinds of auxiliary information to solve a match task. This can help when schema elements cannot be compared merely using metadata in the schemas. Recent techniques use a simple form of reuse at the level of elements of schema by looking up element correspondences by using user-specified correspondences to train machine-learning algorithms on instances of a schema. A more general approach is to reuse entire previously identified match results. In fact, it is observed that new schemas to be matched are often very similar to previously matched schemas. Reusing the existing match results can thus result in significant savings of manual effort. However, the potential of this approach has not yet been studied in current schema matching work (Sellami et al 2010).

- **Graphical User Interface for Match**

  For fully automatic solution needs a user-friendly interface to improve effectiveness of a match system. On the one side, the match process should be achieved interactively, so that the domain knowledge of the user can be actively integrated in identifying corresponding schema elements.

  Unfortunately, most techniques developed so far focus on some research aspects and offer no or only a basic user interface. Commercial tool developed at IBM providing comprehensive graphical user interface CLIO.

1.3.5 **Problem Statement**

- In general, majority of the existing algorithms are concentrating to improve the performance metrics such as precision, recall,
Computational time, Sensitivity, Specificity and F measure by proposing the modification in the existing methodology. Those existing schema matching algorithms may be either working with the principles of clustering or entropy and mutual information based properties.

- One of the popular existing schemas matching algorithm called K-Means clustering (Tajunisha & Saravanan 2010) is a partitioning clustering technique in which clusters are formed with the help of centroids. On the basis of these centroids, clusters can vary from each other in different iterations. It is observed that the algorithm can be very sensitive to the initial centroids. This caused to reduce the performance metrics such as precision, recall etc. Hence to overcome such a problem, the existing K means algorithm is modified by means of centroid section which is called an Efficient Centroid Based Clustering Algorithm for schema matching (ECBCA) to improve performance metrics.

- Another popular schema matching algorithm proposed by Jaewoo Kang & Jeffrey Naughton (2008) to improve the performance metrics is Entropy and Mutual Information based. Those existing algorithms did not use any optimization techniques such as Hill climbing or Genetic algorithm. Hence, in the proposed research work, to improve the performance metrics, the existing Kang – Naughton schema matching technique is modified by integrating with the Hill climbing and Genetic algorithm optimization techniques.
1.3.6 Classification of Search Optimization Techniques

Various approaches are involved in achieving the optimal schema matching; taxonomy of schema matching will be discussed in the next chapter of this thesis work. The following sections discuss the Genetic Algorithm (GA), and other optimization techniques used for obtaining optimal schema matching schemes. The taxonomy of the various search optimizations is shown in the Figure 1.10.

![Figure 1.10 Taxonomy of Search Optimization Techniques]

1.4 GENETIC ALGORITHM

Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. GAs are a branch of Evolutionary computing, a rapidly growing area of artificial intelligence. They are encouraged by Darwin's theory regarding evolution - "survival of the fittest".

GAs might out-perform each specialized and random search strategies on complicated search spaces (Holland 1975). They with efficiency
exploit illustration information to speculate on new search points with expected improved performance, and take issue from other optimization methods by functioning on parameter coding not parameters themselves, looking out from a population of search points, using objective function information not domain specific knowledge, and using probabilistic rules for moving from one state to another state rather than deterministic. GAs is the ways of solving problems by biological processes nature uses; i.e. Selection, Crosses over, Mutation and Termination, to evolve a solution to a problem.

1.4.1 Outline of the Basic Genetic Algorithm

1. Start: Create random population of \( n \) chromosomes

2. Fitness: Calculate the fitness function \( f(x) \) of each chromosome \( x \) in the population.

3. New population: Generate a new population by repeating following steps until the new population is complete.

- Selection: Select two parent chromosomes from a population according to their fitness (better the fitness, bigger the chance to be selected)
- Crossover: With a crossover probability, cross over the parents to form a new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
- Mutation: With a mutation probability, mutate new offspring at each locus (position in chromosome).
- Accepting: Place new offspring in the new population
4. Replace: Use new generated population for a further run of the algorithm

5. Test: If the end condition is satisfied, stop, and return the best solution in current population

6. Loop: Go to step 2

Note: The genetic algorithms performance is basically influenced by two operators referred to as crossover and mutation. These two operators are the most vital parts of GA (Barbosa 2008).

Figure 1.11 Flow Chart for Genetic Programming
Flow Chart for Genetic Programming is shown in Figure 1.11.

- **Operators of Genetic Algorithm**

  Genetic operators employed in genetic algorithms maintain genetic diversity. Genetic diversity may be important for the method of evolution.

  - Reproduction (or Selection)
  - Crossover (or Recombination)
  - Mutation.

- **Selection**

  Selection is the first operator worked on population. Chromosomes from population are selected to be parents to crossover and create offspring.

- **Crossover**

  After the selection of parent chromosomes, the parent chromosomes get cross over among them with a cross over probability and the child chromosomes are obtained as the output from the crossover phase.

- **Mutation**

  Mutation is used to maintain genetic diversity from one generation of chromosomes to the next. Mutation happens during evolution according to a user-definable mutation probability, usually set to low value. Mutation modifies one or further gene values in a chromosome from its initial state. Here new gene values being added to the gene pool. With the new gene values, the genetic algorithm may be able to arrive at best solution.

  The following section discusses the effects of schema theorem.
1.4.2 Effects of the Schema Theorem

The study of genetic algorithms starts with the Schema Theorem. Implicit parallelism, building blocks, and combination greatly influence the design of crossover operators.

1.4.2.1 Implicit Parallelism

“The Schema Theorem provides the first evidence of the implicit parallelism of genetic plans. Each schema represented in the population \( B(t) \) increases or decreases independently of what is happening to other schemata in the population.” Implicit parallelism offers a computational leverage that is supposed to be distinctive to genetic algorithms.

1.4.2.2 The Building Block Hypothesis

The Schema Theorem indicates to completely schemata independently. However, low-order schemata are less likely to be disrupted by crossover, so they are more likely to follow the predicted expectations. Thus, an interpretation of the Schema Theorem recommends that low-order schemata are the important “building blocks” manipulated by crossover. “Just as a child creates magnificent fortresses through the arrangement of simple blocks of wood, so does a genetic algorithm seek near optimal performance through the juxtaposition of short, low-order, high-performance schemata, or building blocks.”

1.4.2.3 Combination

Concept of combination need implicit parallelism and low-order building blocks. Therefore, the interpretation of the Schema Theorem also impacts crossover design models.
1.5 OTHER GENERAL OPTIMIZATION METHODS

Different iterative improvement techniques such as hill climbing simulated annealing, Tabu search, A-Teams, and ant colonies are mentioned in this section.

1.5.1 Hill Climbing

Hill climbing is the simple algorithm for neighborhood search (Milenic Mitchell et al 1994). Apply a series of operators to create a “neighborhood” of new solutions. Best solution that increases on the start solution, and carry on until the existing solution is better than all of its neighbors. This concluding result will be a local optimum. Since hill climbing routines have no global perception, they are also known as “local optimizers”- each solution in an examine space is mapped to its adjoining locally optimal solution. Hill climbing results can be unsatisfactory because of no attempt is made to find the global optimum. The Hill climbing can be enhanced by restarting the process from random start points.

1.5.2 Tabu Search

Tabu search is used to promote wandering. From the procedure of random hill climbing, the first refining move is taken. If no move causes an improvement, the next best another is taken. At the subsequent step, it is relatively possible that the only improving move will be back into the local optimum. To overcome the problem of stuck in a two-step cycle; inverses of recent moves are made “Tabu”. This search can quickly come out of the local optimum and towards alternative search space region. Tabu search is not concentrating only the quality of the “final” solution and also interested in finding the best solution.
1.5.3 Ant Colonies

Ant colonies are an optimization technique based on the communication processes employed by foraging ants. Assuming that a solution can be represented as a path through a graph, the probability that an edge is traversed by an “ant” is influenced by the amount of “pheromone” deposited on that edge. Typically, only the best solution of a “generation” is allowed to leave pheromone. Then, with the updated pheromone trails, a new generation of ants creates a new generation of solutions. The pheromone communication process allows diversity to be actively controlled.

The following section discusses the clustering concepts and various types of clustering algorithms.

In this research work, the genetic algorithm is combined with the hill climbing approach to improve the performances of the automated schema matching algorithms.

1.6 CLUSTERING CONCEPTS

Clustering is an important problem with many applications, and a number of different algorithms and methods have emerged over the years (Pei et al 2006). The goal of clustering is to group data points into homogeneous groups, where the homogeneity is usually measured by distances or similarities among data points. Clustering is a data mining (Han et al 2011) technique that groups similar objects without any supervised information. However, in various applications there is access to additional information or domain knowledge about the types of groups that are sought in the data, such as complementary information about actual similarities between object pairs.
1.6.1 Types of Clustering Algorithms

Main categories of clustering algorithm, viz, Hierarchical, Partition, Spectral, Grid based and Density based clustering algorithms are discussed below.

1.6.1.1 Hierarchical Clustering Algorithm

The hierarchical clustering algorithm (Alofairi 2012) is based on tree shaped structure. It is divided into agglomerative hierarchical clustering and divisive hierarchical clustering. In the agglomerative approach is also called as the bottom up approach in which each data point is considered to be a separate cluster and iteration of the clusters is merged based on condition. In the divisive approach is also called as top down approach. Here all data points are considered as a single cluster and split into a number of clusters, based on certain condition. LEGCLUST, BIRCH (Balance Iterative Reducing and Clustering using Hierarchies), CURE (Cluster Using Representatives) and Chameleon are example for clustering algorithm.

1.6.1.2 Partition Clustering Algorithm

Partitioning clustering algorithm typically result in a set of M clusters, every entity belonging to one cluster. Centroid is used to group the objects. Description of the object is being clustered. In cases wherever real-valued data is available, the arithmetic mean of the attribute for all objects within a cluster provides an appropriate representative; alternative types of centroid may be required in other cases. If the measure of clusters is huge, the centroids are more clustered to supply a hierarchy inside a dataset. The partition clustering algorithm splits the data points into k partition, where every partition represents a cluster. The partition is done based on certain
objective functions. Square error criterion is calculated by using this Equation 1.1,

\[ E = \sum \sum \| p - m_i \|^2 \] (1.1)

where \( p \) is the point in a cluster and \( m_i \) is the mean of the cluster. The cluster should exhibit two properties; (1) each group must hold at least one object (2) every object must fit to accurately one group. The problem of this algorithm is whenever a point is close to the center of another cluster; it gives a poor result due to overlapping of the data points (Wu et al 2004).

\[ \text{K- Means Clustering Algorithm (Chen Zhang & Shixiong Xia 2009)} \] is one of the partition based clustering algorithms. The advantage of the K-means algorithm is (Fahim et al 2006) that it is easy to implement and can work with any norms. The disadvantages of the K-means algorithm is depend on the initial value of the centroids. The local optimum does not need to be a global optimum.

\[ 1.6.1.3 \text{ Spectral Clustering Algorithm} \]

Spectral clustering refers to a category of Eigen structure of a similarity matrix. Clusters are designed by partitioning data points using the similarity matrix. Spectral clustering algorithm has three main stages such as preprocessing, spectral mapping and post mapping. Preprocessing step constructs the similarity matrix. Spectral Mapping deals with the construction of Eigen vectors for the similarity matrix. Post Processing deals with the clustering of data points. Spectral clustering algorithm does not consider local optima, assumptions on the cluster shape are not made, simple to implement and works faster.
1.6.1.4 Grid based Clustering Algorithm

The grid structure of grid based algorithm based on finite number of cells. Actions are done on these grids. The main advantage of this method is lower processing time. Complexity of Clustering is based on the number of populated grid cells. It does not hang on the number of entities in the dataset. The characteristics of this algorithm are, no distance computations and clustering is achieved on précised data points. Example for this algorithm is STING.

1.6.1.5 Density based Clustering Algorithm

The density based algorithm based on threshold value. It handles noisy data. The features of this algorithm are follows: it handles clusters of arbitrary shape, handles noisy data, only one scan is needed of the input dataset and the density parameters to be initialized. The examples of this algorithm are DBSCAN, DENCLUE and OPTICS. The survey of these clustering algorithms will be discussed in Chapter 2.

In this thesis, Centroid based Clustering Algorithm is proposed to improve the performances of the automated schema matching.

The following sections discuss the objectives, contributions and organization of the thesis work.

1.7 OBJECTIVES

The main objectives of this thesis are as follows:

1. To propose the schema matching with the improved precision, recall rate and computation time by
i. developing a schema matching technique using Efficient Centroid Based Clustering Algorithm

ii. developing a schema matching technique using Optimal Hill Climbing Approach with Genetic Algorithm

2. To validate the proposed techniques, by comparing the performances and to find out the best schema matching algorithm

1.8 CONTRIBUTIONS

The major contributions of this research work are listed below.

1) The developed Efficient Centroid Based Clustering Algorithm and an Optimal Hybrid Hill Climbing Approach with Genetic Algorithms considerably improve the performances such as Precision, Recall, Computation time, sensitivity, specificity and f-measure over the other existing schema matching techniques.

2) The proposed Efficient Centroid Based Clustering Algorithm shows 54.02 % and 14.56 % improvement in precision over the existing Kang-Naughton Mutual Information (KNMI) and K-Means algorithm for the census dataset.

3) The developed Efficient Centroid Based Clustering Algorithm for schema matching technique shows improved recall performance by 46.41 %, 25.71 % and 16.48 % over the existing KNE, KNMI and K-Means algorithms, for the Census dataset.
4) The proposed Hybrid Hill Climbing Approach with Genetic Algorithm for schema matching improves sensitivity by 13.77%, 9.11% and 2.81% over the existing Kang-Naughton entropy only, Kang-Naughton Mutual Information and the modified Kang-Naughton Genetic Algorithm (KNGA) for Census dataset.

5) The proposed Hybrid Hill Climbing Approach with Genetic Algorithm for schema matching shows 57.86% and 11.93% improvement in precision rate over the existing Kang-Naughton Mutual Information and the modified Kang-Naughton Genetic algorithm schema matching technique respectively.

6) Among the proposed Efficient Centroid Based Clustering Algorithm and Hybrid Hill Climbing Approach with Genetic Algorithms for schema matching, Hybrid Hill Climbing Approach with Genetic Algorithms performs better in improving precision, recall, computation time, sensitivity and specificity over the other.

1.9 ORGANISATION OF THE THESIS

The organization of the thesis is as follows:

Chapter 1 presents the introduction to Integrated Information System, schema matching, the issues and challenges in schema matching, various evolutionary algorithms related to schema matching applications, and also the objective of the research work.

Chapter 2 discusses the various existing research works that are relevant to the work presented in this thesis.
Chapter 3 describes an Efficient Centroid Based Clustering Algorithms for schema matching to improve the precision, recall rates and computation time when automating the schema matching schemes.

Chapter 4 discusses Optimal Hybrid Hill Climbing Approach with Genetic Algorithms to improve the precision, recall rates and computation time when automating the schema matching schemes.

Chapter 5 compares the performances of the proposed Efficient Centroid based Clustering Algorithms and the Optimal Hybrid Hill Climbing Approach with Genetic Algorithms with the various other existing automated schema matching algorithms.

Chapter 6 concludes the contributions and findings in addition to suggestions for future enhancement.