Chapter 2. Literature Review

Benchmark is an important method to evaluate the performance of different computer systems. It is wildly used in the database system performance measurement and evaluation for a long time. While XML emerging as a leading data format, several benchmarks used to evaluate the XML management system have been proposed. However, there is still no guideline for evaluation of ontology.

To develop a benchmark, it is necessary to define a workload first. The workload model consists of three parts: the data model, the operation model, and the control model. In XML benchmark, the operation model is a comprehensive set of queries, and it is the most important part in the benchmark. In this chapter, we introduce the XML query functionality first. It would help us to review the operation model of each existing XML benchmark. Consequently, three XML benchmark projects are introduced by their data model, operation model, and control model. Then we compare them with this research. Finally, the ontology related benchmark works are reviewed and discussed.

2.1. XML Query Capability

Benchmarking the XML data management systems should consider many factors. Designing a set of comprehensive queries to test the XML databases' performance is an important point. XML query languages should capture the whole characteristics of a XML document, and the functionalities they provide would influence the query performance. The W3C XML Query Language working group (Chamberlin, Fankhauser, Marchiori, & Robie, 2003) list 20 XML query language "must have"

functionalities, as Table 2.1 shows. Some of the expected functionalities may affect the efficiency of the system significantly.

Table 2.1: Desired Functionalities of XML Query Languages (Chamberlin et al., 2003)

Id	Functionality	Description
F1	Supported Operations	The XML Query Language MUST support operations on all data types represented by the XML Query Data Model.
F2	Text and Element Boundaries	Queries MUST be able to express simple conditions on text, including conditions on text that spans element boundaries.
F3	Universal and Existential Quantifiers	Operations on collections MUST include support for universal and existential quantifiers.
F4	Hierarchy and Sequence	Queries MUST support operations on hierarchy and sequence of document structures.
F5	Combination	The XML Query Language MUST be able to combine related information from different parts of a given document or from multiple documents.
F6	Aggregation	The XML Query Language MUST be able to compute summary information from a group of related document elements.
F7	Sorting	The XML Query Language MUST be able to sort query results.
F8	Composition of Operations	The XML Query Language MUST support expressions in which operations can be composed, including the use of queries as operands.
F9	NULL Values	The XML Query Language MUST include support for NULL values. Therefore, all operators, including logical operators, MUST take NULL values into account.
F10	Structural Preservation	Queries MUST be able to preserve the relative hierarchy and sequence of input document structures in query results.

F11	Structural Transformation	Queries MUST be able to transform XML structures and MUST be able to create new structures.
F12	References	Queries MUST be able to traverse intra- and inter-document references.
F13	Identity Preservation	Queries MUST be able to preserve the identity of items in the XML Query Data Model.
F14	Operations on Literal Data	Queries SHOULD be able operate on XML Query Data Model instances specified with the query ("literal" data).
F15	Operations on Names	Queries MUST be able to perform simple operations on names, such as tests for equality in element names, attribute names, and processing instruction targets, and to perform simple operations on combinations of names and data.
F16	Operations on Schemas	Queries SHOULD provide access to the XML schema or DTD for a document, if there is one.
F17	Operations on Schema PSV Infoset	Queries MUST be able to operate on information items provided by the post-schema-validation information set defined by XML Schema.
F18	Extensibility	The XML Query Language SHOULD support the use of externally defined functions on all data types of the XML Query Data Model. The interface to such functions SHOULD be defined by the Query Language, and SHOULD distinguish these functions from functions defined in the Query Language. The implementation of externally defined functions is not part of the Query Language.
F19	Environment Information	The XML Query Language MUST provide access to information derived from the environment in which the query is executed, such as the current date, time, locale, time zone, or user.
F20	Closure	Queries MUST be closed with respect to the XML Query Data Model.

XQuery has met all of the requirements except F12 and F16, and it becomes a standard query language to test the performance of XML data management systems. Generally speaking, queries to benchmark XML databases would fall into several categories: Match, Join, Navigation, Casting, Reconstruction, and Update. Queries for Match are mainly used to test the database ability to handle simple string lookups with a fully specified path. Join queries can be divided into two parts: Join on References, and Join on Values. References are an important part of XML, because they allow richer relationships than just hierarchical structure. Queries Join on References would test if query optimizer can take advantage of references to be joined. Queries Join on Values, on the other hand, would test the database's ability to handle large intermediate results. Differing from the former, their joins are on the basis of values. Navigation Queries investigate how well the query processor can optimize path expressions, and avoid traversing irrelevant parts of the tree. Strings are the basic data type in XML documents. Casting strings to another data type that carries more semantics is necessary. Queries for Casting challenge the ability of the database to cast different data types. Reconstruction Queries attempt to reconstruct the original document from its fragmentations stored in the databases. Update Queries try to add, delete, and modify elements in the XML document. These queries test the databases' ability to manage XML document. Furthermore, other XML query functionalities such as sort, ordered access, text search, and aggregation also should be captured in the benchmark query set.

2.2. XML Benchmarks

As XML becomes a dominant technology on the Web, it is beginning to be extensively used in various application domains. In order to manage large amounts of

XML documents, XML storage and management systems are being offered by most data management vendors. A benchmark to identify the important performance parameters for these various systems under varying levels of load and differing environments has thus become a necessity. XMark, XMach-1 and XOO7 are three benchmarks available today that can be used to evaluate certain aspects of XML database systems. We will first briefly describe these benchmarks and their queries before comparing them with this research.

2.2.1. XMark

XMark (Schmidt, Waas, Kersten, Carey, Manolescu, & Busse, 2002) is a single-user benchmark.

- Data Model

The data model of XMark is an Internet auction site. Therefore, its database contains one big XML document with text and non-text data. XMark enriches the references in the data, like the item IDREF in an auction element and the item's ID in an item element. The text data used are the 17000 most frequently occurring words of Shakespeare's plays. The standard data size is 100MB with a scaling factor 1.0 and users can change the data size by 10 times from the standard data (the initial data) each time. However, it has no support for XML Schema.

- Operation Model

In operation model of XMark, 20 XQuery challenges are designed to cover the essentials of XML query processing, as Table 2.2 shows. No update operations are specified in XMark.

Table 2.2: Queries Specified in XMark Benchmark

ID	**Description** (Schmidt et al., 2002)	**Functionality Coverage**
Q1	Return the name of the person with ID 'person0'.	Exact Match
Q2	Return the initial increases of all open auctions.	Ordered Access
Q3	Return the fist and current increases of all open auctions whose current increase is at least twice as high as the initial increase.	Ordered Access
Q4	List the reserves of those open auctions where a certain person issued a bid before another person.	Ordered Access
Q5	How many sold items cost more than 40?	Data-type Cast
Q6	How many items are listed on all continents?	Regular Path Expressions
Q7	How many pieces of prose are in our database?	Regular Path Expressions
Q8	List the names of persons and the number of items they bought. (joins person, closed auction)	Joins
Q9	List the names of persons and the names of the items they bought in Europe. (joins person, closed auction, item)	Joins
Q10	List all persons according to their interest; use French markup in the result.	Document Construction
Q11	For each person, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.	Joins
Q12	For each richer-than-average person, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.	Joins
Q13	List the names of items registered in Australia along with their descriptions.	Document Construction
Q14	Return the names of all items whose description contains the word 'gold'.	Text Search
Q15	Print the keywords in emphasis in annotations of closed auctions.	Regular Path Expressions
Q16	Confer Q15. Return the IDs of the sellers of those auctions that have one or more keywords in emphasis.	Regular Path Expressions

Q17	Which persons don't have a homepage?	Missing Elements
Q18	Convert the currency of the reserve of all open auctions to another currency.	Function Application
Q19	Give an alphabetically ordered list of all items along with their location.	Sorting
Q20	Group customers by their income and output the cardinality of each group.	Function Application

Table 2.2 shows the description of queries in XMark, and the query functionalities they covered. We find that XMark includes almost complete query functionality. Each query only tests a single aspect of XML. This would help people to explain the performance result easily. But some queries are functionally similar in testing certain features of the query optimizer. Another notable feature is that XMark does not specify any update operation.

- Control Model

In XMark, the control model contains a repetition factor, which indicates how often a query was executed in the same environment.

2.2.2. XMach-1

XMach-1 (Böhme & Rahm, 2001) is a scalable multi-user benchmark. The main objective of the benchmark is to stress-test XML systems under a multi-user workload.

- Data Model

The data model of XMach-1 is designed for B2B applications and considers text documents and catalog data. It assumes that size of the data files exchanged will be small. It provides support for DTD only and does not consider XML Schema for optimization.

- Operation Model

The operation model of XMach-1 consists of eight queries and three update operations, shown in Table 2.3.

Table 2.3: Queries Specified in XMach-1 Benchmark

ID	**Description** (Böhme & Rahm, 2001)	**Functionality Coverage**
Q1	Get document with URL X.	Exact Match, Joins
Q2	Get doc_id from documents containing phrase X in a paragraph element.	Regular Path Expressions, Text Search, Joins
Q3	Start with first chapter element and recursively follow first section element. Return last section elements.	Function Application, Ordered Access
Q4	For a document with doc_id X return flat list of head elements which are children of section elements.	Regular Path Expressions
Q5	Get document name (last path element in directory structure) from all documents which are below a given URL fragment.	Regular Path Expressions
Q6	Get doc_id and id of parent element of author element with content X.	Joins
Q7	Get doc_id from documents which are referenced at least X times.	Function Application
Q8	Get doc_id from the last X updated documents having an author attribute.	Sorting, Join
M1	Insert new document.	Update Operation
M2	Delete document with doc_id X.	Update Operation
M3	Update name and update_time attributes for document with doc_id X.	Update Operation

Queries specified in XMach-1 cover typical database functionality (join, aggregation, sort) as well as information retrieval and XML-specific features (document assembly, navigation, element access). Update operations cover inserting

and deleting of documents as well as changing attribute values. We find that some queries contain several query functionalities. For example, Q8 needs count, sort, join and existential operations and accesses metadata. It is hard to analyze the experiment result and ascertain which feature leads to the given performance result. Specially, XMach-1 has defined three update operations that are unique across other XML benchmarks.

- Control Model

To achieve a true multi-user environment with a realistic number of concurrent clients, XMach-1 requires that each browser and loader runs at most one operation at a time. Furthermore, after completing an operation there is a think time between 1 and 10 seconds before the next operation is started.

2.2.3. XOO7

XOO7 (Li, Bressan, Dobbie, Lacroix, Lee, Nambiar, & Wadhwa, 2001) is an XML version of the OO7 benchmark, which was designed to test the efficiency of object-oriented DBMS. XOO7 is a single-user based benchmark for XMLMS that focuses on the query processing aspect of XML.

- Data Model

The data model of XOO7 comes from the OO7 benchmark by mapping the OO7 schema and data set to XML. No specific application domain is modeled by the data of XOO7. It is based on a generic description of complex objects using component-of relationships. XOO7 also proposes three different databases of varying size: small, medium, and large. It supports DTD only.

- Operation Model

In operation model, XOO7 provides relational, document and navigational

queries that are specific and critical for XML database applications. These queries test the primitive features and each query covers only a few features. Table 2.4 displays the queries adopted in XOO7.

Table 2.4: Queries Specified in XOO7 Benchmark

ID	**Description** (Li et al., 2001)	**Functionality Coverage**
Q1	Randomly generate 5 numbers in the range of AtomicPart's MyID, then return the AtomicPart according to the 5 numbers.	Exact Match
Q2	Randomly generate 5 titles for Documents, then return the first paragraph of the Document by lookup on these titles.	Ordered Access
Q3	Select 5% of AtomicParts via buildDate (in a certain period).	Function Application
Q4	Find the CompositePart if it is later than BaseAssembly it is using (comparing the buildDate attribute).	Ordered Access
Q5	Within the same BaseAssembly, return the AtomicParts once finding a Document that has MyID equals to its docId.	Text Search
Q6	Select all BaseAssemblies with earlier buildDate from one XML database where it has the same "type" attributes as the BaseAssemblies in another database.	Document Construction
Q7	Randomly generate two phrases among all phrases in Documents. Select those documents containing the 2 phrases.	Function Application
Q8	Repeat query1 but replace duplicate elements using their IDREF.	Sorting
Q9	Select all AtomicParts with corresponding CompositeParts as their sub-elements.	Document Construction
Q10	Select all ComplexAssembly with type "type008" without the knowledge of the path.	Joins
Q11	Among the first 5 Connections of each CompositePart, select those with length greater than	Joins

	"len".	
Q12	For each CompositePart, select the first 5 Connections with length greater than "len".	Document Construction
Q13	For each BaseAssembly count the number of documents.	Function Application
Q14	Sort CompositePart in descending order where buildDate is within a year from current year.	Sorting
Q15	Find BaseAssembly of not type "type008".	Function Application, Joins
Q16	Return all BaseAssembly of type "type008" without any child nodes.	Document Construction
Q17	Return all CompositePart having Connection elements with length greater than Avg(length) within the same CompositePart without child elements.	Ordered Access, Joins
Q18	For CompositePart of type "type008", give 'Result' containing ID of CompositePart and Document.	Ordered Access, Joins
Q19	Select all of CompositePart, Document and AtomicPart.	Document Construction
Q20	Select the last connection of each CompositePart.	Ordered Access
Q21	Select the third connection's AtomicParts of each CompositePart.	Ordered Access
Q22	Select the AtomicPart whose MyID is smaller than its sibling's and it occurs before that sibling.	Ordered Access, Data-type Cast
Q23	Select all Document after the Document with MyID = 25.	Ordered Access

XOO7 contains large amount of queries, each query covers only a few features. Comparing to the other two benchmarks, XOO7 has certainly the highest ratio which stresses its data-centric focus. However, we can find that some queries are focus on the same functionality. Similar to XMark, no update operation is specified in XOO7.

- Control Model

  The control model in XOO7 is the number of repetitions.

2.3. XML Benchmarks Comparison

A brief comparison of key features on these three XML benchmarks against this research is described in Table 2.5. The key features include application focus, evaluation scope, database and workload characteristics, etc.

Table 2.5: Comparison of Benchmarks over Workload Characteristics

Feature	**XMark**	**XMach-1**	**XOO7**	**This Research**
Evaluation Scope	Query Processor	DBMS	Query Processor	Heterogeneous Information Integration
Application Domain	E-Commerce	E-Commerce	Generic	Generic
Data Model				
Documents	Single-document	Multi-documents	Multi-documents	Multi-documents
Scalability of Document Number	1	10^4~10^7	Unlimited	Various
Scalability of Document Size	10MB~10GB	16KB	Unknown	Various
Data Heterogeneity	XML document only	XML document only	XML document only	Heterogeneous data sources
Nodes/KB	18	10	67	Various
Operation Model				
Queries	20	8	23	14
Update Operation	0	3	0	0

Table 2.6 groups queries of each benchmark by query functionality. Compared to other XML benchmarks, XMark provides a concise and comprehensive set of queries. However, it does not provide update operations to manipulate XML documents. XMach-1 only defines a small number of XML queries that cover multiple functions

and update operations for which system performance is determined. XOO7 maps the original queries of OO7 into XML, and adds some XML specific queries. In general, XMach-1, XMark and XOO7 cover only a subset of the XML query requirements. In this research, we attempt to propose a generic workload model. In order to cover the whole functionalities of XML query processing, we combine queries of these three XML benchmarks and integrate them into 10 types of queries. In particular, the information integration system is generally used for query data, not provide data manipulation functions. Therefore, the query model in this research does not support update operations.

Table 2.6: Comparison of Benchmarks over Query Functionalities

Query Functionality		**XMark**	**XMach-1**	**XOO7**	**This Research**
Exact Match		√	√	√	√
Joins	Join on Reference	√	√	√	√
	Join on Value	√			√
Regular Path Expressions	Full Sub-path	√	√		√
	Unknown Sub-path	√	√		√
Document Construction	Structure Preserving	√		√	√
	Structure Transforming	√		√	√
Ordered Access		√	√	√	√
Sorting	By String	√	√		√
	By Non-string			√	√
Missing Elements		√			√
Text Search		√	√	√	√
Data-type Cast		√			√
Function Application		√	√	√	√
Update Operation			√		

2.4. Ontology

Currently the information integration issue attracts researchers from all around the world. Numerous information integration systems are already available and the number is growing fast. Ontologies play an important role for integration as a way of formally defined terms for communication. They aim at capturing domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused, shared, and operationalized across applications and groups.

A good ontology should represent the domain specific knowledge explicitly. The question is how do we know an ontology is good? The answer is the ontology benchmark. There are plenty of benchmark studies in other fields like database or compilers. However, there are no specific benchmarks studies or tools for evaluating ontology-based applications. In fact, there is still no guideline to evaluate ontologies and related technologies.

In this section, we introduce the role of ontologies in information integration first. And then we discuss a major inference task, which is the main operation of an ontology benchmark. Finally, the ontology related benchmark works are reviewed and discussed.

2.4.1. Ontology and Information Integration

Traditional integration approaches use inexpressive models of database schemas or XML trees to integrate heterogeneous data sources. This would cause many semantic heterogeneity problems. Ontologies provide much richer modeling means with classes and properties organized into is-a hierarchy and enriched with axioms and relations processable with inference. Main benefits for an ontology-based

approach are illustrated as follows (Maier, Aguado, Bernaras, Laresgoiti, Pedinaci, Pena, & Smithers, 2003):

- The ability to picture all occurring data structures, for ontologies can be seen as nowadays most advanced knowledge representation model.
- The combination of deduction and relational database systems, which extends the mapping and business logic capabilities.
- A higher degree of abstraction, as the model is separated from the data storage.
- Its extendibility and reusability.

Almost all ontology-based integration approaches ontologies are used for the explicit description of the information source semantics. With respect to the integration of data sources, they can be used for the identification and association of semantically corresponding information concepts. Some approaches use ontologies not only for content explication, but also either as a global query model or for the verification of the (user-defined or system-generated) integration description (Wache, Vögele, Visser, Stuckenschmidt, Schuster, Neumann, & Hübner, 2001). Ontologies are usually expressed in a logic-based language, so that fine, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties, and relations. Therefore, ontologies not only have the expressiveness needed in order to model the data in the sources, but their reasoning ability can help in the selection of the sources that are relevant for a query of interest, as well as to specify the extraction process. Ontologies let domain experts, system developers, and applications perform reasoning about information content in an application domain.

2.4.2. Ontology and Reasoning

Ontologies intend to provide a machine-understanding syntax for information

integration. Understanding is closely related to reasoning. Reasoning is important to ensure the quality of an ontology. During ontology design, it can be used to test whether concepts are non-contradictory and to derive implied relations. It may also be used when the ontology is deployed, one can determine the consistency of facts stated in the annotation with the ontology or infer instance relationships (Baader, Horrocks, & Sattler, 2003). Therefore, reasoning is the major operation in the ontology-based application. The workload model of the ontology benchmark should identify key reasoning tasks in the operation model.

Tempich and Volz (2003) mention that a reasoner supporting ontology languages usually offers several different query services with respect to an ontology. These query services primarily target queries about classes. They fall into four categories, class-instance membership queries, class subsumption queries, class hierarchy queries, and class satisfiability queries. The description of these query services is described in Table 2.7. There are similar queries about properties, ie. property-instance membership, property subsumption, property hierarchy, and property satisfiability, and also the possibility to check the consistency of the whole ontology.

Table 2.7: Query Services (Tempich & Volz, 2003)

Query Service		**Description**
Class-instance Membership Queries	**Ground**	Determine whether a given individual is an instance of a given class.
	Open	Determine all the individuals in an ontology that are instances of a given class.
	"All-classes"	Determine all the classes in an ontology that a given individual is an instance of.
Class Subsumption Queries		Determine if a given class is a subclass of another class.
Class Hierarchy Queries		Return all/most-specific superclasses of a

	given class and/or all/most-general subclasses of a given class.
Class Satisfiability Queries	Determine if the definition of a given class is generally satisfiable (consistent).

Simov and Jordanov (2002) cite that ontologies within their ontology-based project have two types of reasoning tasks, terminological reasoning and instance reasoning. Terminological reasoning checks the classes are defined and the relations between them are explicitly represented. Instance reasoning involves first an already developed ontology (after some terminological reasoning) and next large amounts of instances. The basic tasks of these two types of reasoning are shown as Table 2.8.

Table 2.8: Reasoning Tasks (Simov & Jordanov, 2002)

Reasoning Type	**Task Description**
Terminological Reasoning	Checking whether a class definition is consistent by itself or with respect to a set of other class descriptions.
	Checking whether a given class definition is more general than another class definition.
	Construction of explicit hierarchy of class names on the base of their class definitions.
Instance Reasoning	Find the most specific classes that describe a partially specified instance.
	Find all instances in the dataset which are instances of a given class definition.
	More complex queries involving instance data. Such could be getting all pairs of instances related in a way.
	Checking the consistence of the instance data with respect to the ontology.

We find that terminological reasoning is similar to class subsumption queries, class hierarchy queries, and class satisfiability queries. Instance reasoning is similar to

class-instance membership queries. This would provide this research with the basis of major reasoning tasks in the operation model of the ontology benchmark workload model.

2.4.3. Ontology and Benchmark

To the best of our knowledge, the benchmark presented here is the first one for ontology-based information integration. The ontology benchmark model in this research differs from database benchmarks, such as Wisconsin benchmark, OO7 benchmark, and BUCKY benchmark. They are all DBMS-oriented and storage benchmarks, and there is no inference ability included. In this research, the ontology workload model is applied to an information integration system, and we focus on the inference ability of the ontology.

Ontology and XML are often found together and are often confused. XML is a standard for marking up - adding additional information, called metadata - to documents. The purpose of XML is to tag textual information with additional structure that enables it to be “understood” and exchanged by programs. However, XML tags still require humans to interpret their meanings. Therefore, XML benchmarks only focus on structural and syntactic evaluation of systems, and they have no semantics. On the other hand, ontology benchmark is devoted to capture the semantic expressions in the system. Thus, ontology and XML are complementary technologies: ontology provides the meaning for XML standards; XML provides a valuable medium for information exchange between programs that share the same ontology (Andersen, 2001).

As mentioned above, there is still no guideline for evaluation of ontology-based application. Horrocks and Patel-Schneider (1998) benchmark description logic

systems, or so-called knowledge bases. Description logics (DLs) are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way. Description logic systems provide their users with various inference capabilities that deduce implicit knowledge from the explicitly represented knowledge. Horrocks and Patel-Schneider try to evaluate the reasoning algorithms in description logics. The knowledge base is composed of a Tbox and an Abox, as Figure 2.1 shows. Terminological part (Tbox) is a set of axioms describing the structure of domain. Assertional part (Abox) is a set of axioms describing concrete situation (Horrocks, 2002). They are related to this research. In an information integration system, the ontology can be viewed as the Tbox, and the heterogeneous data can be viewed as Abox. However, the logic described is only a subset of the ontology languages, such as DAML+OIL and OWL. DAML+OIL and OWL can be seen to be equivalent to a very expressive description logic. They provide more constructors and allow more axioms than description logic. Therefore, the inference services of ontology are more complex than traditional description logic systems.

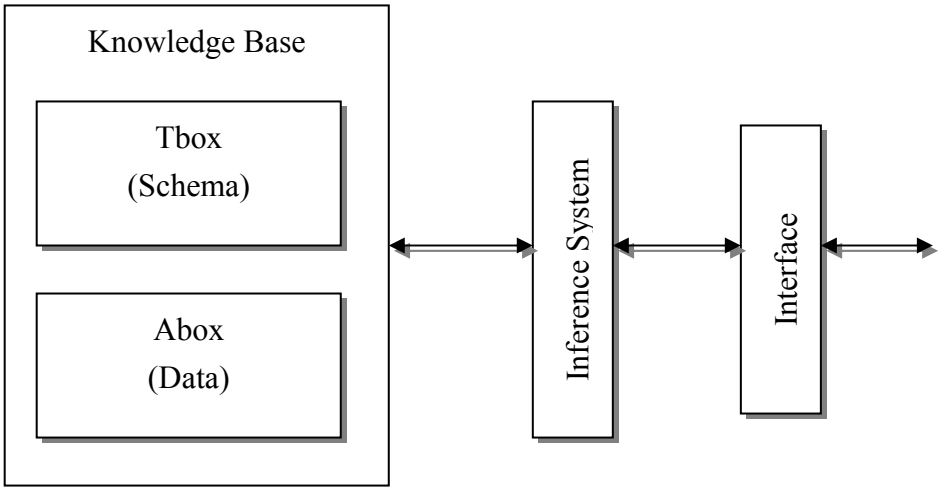

Figure 2.1: DL System Architecture (Horrocks, 2002)