

# Natural Language Processing for Conceptual Modeling

Lilac A. E. Al-Safadi

Department of Information Technology,  
College of Computer and Information Sciences,  
King Saud University, Riyadh, Saudi Arabia  
lalsafadi@ksu.edu.sa  
doi: 10.4156/jdcta.vol3.issue3.6

## Abstract

A semi-automated approach for the design of databases in enhanced-ERD notation is presented. It focuses on the very early stage of the database development which is the stage of user requirement analysis. It is supposed to be used between the requirements determination stage and analysis. The approach provides the opportunity of using natural language text documents as a source of knowledge for semi-automated generation of a conceptual data model. The system performs information extraction by parsing the syntax of the sentences and semantically analyzing their content.

## Keywords

Enhanced-ERD, entity-relationship diagram, conceptual modeling, NLP, natural language, database design, user requirement analysis.

## 1. Introduction

Database design is the process of creating a design for a database that will support a problem domain. The term “problem domain” represents the aggregation of knowledge about objects and active subjects, tied together with specific relations and pertaining to some common tasks [17].

Database design translates information and needs of the organization/user to be supported (ideas) into a physical realization of the database design to support those needs (code). Translating a conceptual model into a physical database is a systematic approach. However, translating user requirements and problem domain described in natural language, into the formal modeling used in database design is a great challenge. According to [10], “We are not really having a problem coding a solution—we are having a problem understanding what solution to code. . . . If you focus on requirements and verification and validation, the coding will take care of itself”.

Figure 1 shows the database development approach using four abstraction levels from reality to stored data:

- The description of the reality using natural language.
- The description of the conceptual model using enhanced-ER model.
- The mapping of the conceptual data model to logical data model.
- The translation of the logical data model into target DBMS (physical data model).

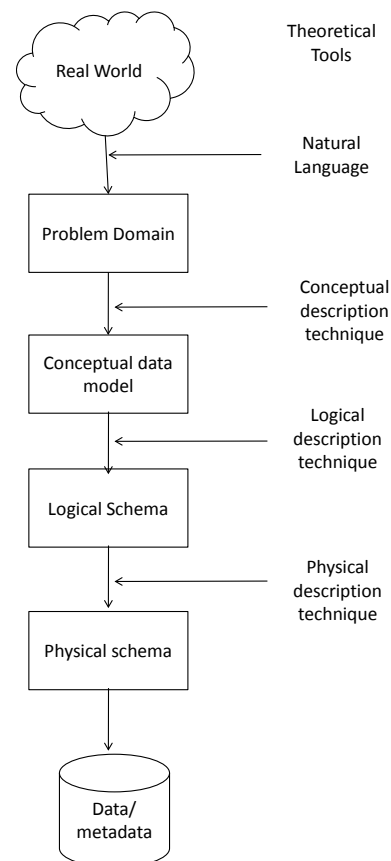


Figure 1. Data modeling levels of abstraction

A semi-automatic approach for designing databases has been proposed in this paper. The main focus of the work would be the construction of a tool to transform a natural language description into a conceptual data model (enhanced-ER model). This model would then form a basis for auto establishment of the database.

Section 2 describes the nature of the problem and the importance of bridging the gap between informal natural language and formal conceptual data models. An overview of enhanced-ER model and steps required in developing an enhanced-ER model are presented in section 3 and section 4, consecutively. A framework for applying natural language processing technique to user requirements analysis to develop conceptual models is proposed in section 5. Section 6 presents the existing case tools in database designing and surveys the past works related to the use of natural language in conceptual modeling. Case study and experimental results are proposed in section 7 and 8. Some final conclusion and future works are proposed in section 9.

## 2. The Description of the Problem

User requirement analysis is one of the most important stages in the database design process. It aims at understanding the problem domain and the nature of the user's problem in order to find a suitable database solution. The more precise the user requirement analysis is, the accurate the function of the database application.

Accurate database design is a reflection of well data modeling. The conceptual data model can serve as the foundation for establishing a database. User requirement analysis helps database designer to explore and identify entities, attributes and relationships that basically build the conceptual model.

Natural language is used to describe user requirement, but it is often complex, vague and ambiguous. Sentences are complex when they contain clauses and phrases that describe and relate several objects, conditions, events and/or actions. Sentences are vague when they contain generalizations, or they are missing important information, especially the subject or objects needed by a verb for completeness. Sentences are ambiguous when they are open to multiple interpretations [5]. In addition, Natural language is robust. Robustness is represented by incomplete knowledge, which is resulted from incomplete lexicon and incomplete grammar. All these troubles arise (naturally) when we discuss our needs and problems using natural language.

On the other hand, database requires more precision, formality and simplicity than that commonly found in natural language. Although SQL is a free-

format and it is more natural than programming languages, it typically limits expressions to a few simple reserved words, user-defined words, their combinations and structure.

Given these factors, the translation of natural language descriptions into database is quite a challenge. Meanwhile, we need ways to systematically map our rich and meaningful ideas into the limited forms of expression supported by SQL.

By investigating a number of user's requirements and conducting thorough experiments, we found that with suitable constraints on the syntax of written user requirements, Natural Language Processing (NLP) can serve as a tool for auto conceptual models generation. It helps to fill the gap between the informal natural language used to describe problems and the formal modeling languages used to specify database solutions.

DBDT is a database-designing tool that is mainly concerned with transforming of user requirement to SQL database establishment. Nowadays, most of the commercial available tools ignore a very important step in designing a database. They do not provide any kind of assistance for better user requirement understanding. In DBDT, we enhance the DB designing tools by incorporating a user requirements analysis technique that aids in extracting the elements necessary in designing and building Databases from the description of the problem domain.

DBDT has developed two approaches: One-click approach where the information extracted from the user requirement is automatically depicted by the conceptual model. Custom approach which involves the user in the information extraction process. User involvement refers to the user's intervention in handling ambiguities and clarification of the input. The resulted conceptual model will be strongly tailored to user's expectations.

## 3. Entity Relationship Model

Data modeling is the first step in the information systems design, serving as a bridge between real-world objects and the information systems model that resides in the computer [20].

The overall approach to database modeling involves four broad steps: (1) identify useful semantic concepts in the problem domain, (2) identify the properties of these concepts, (3) the meaningful association among the concepts, and (4) distinct subclasses and superclasses.

Some useful semantic concepts are entity, attribute, relationship, and subtype [13]. The enhanced-ER (EER model) is a well-known conceptual data models. These diagrams depict the E-R model's three main

components: entities (or objects), attributes, and relationships.

### Entities

An entity refers to the set of entities or real-world objects of the same type that share the same properties. An entity is represented by a rectangle containing the entity's name.

### Attributes

Attributes are descriptive properties for an entity type or a relationship type. Attributes are classified as simple or composite, single-valued or multivalued. They are represented by ovals and are connected to the entity with a line. Each oval contains the name of the attribute it represents. Attributes also have a domain. A domain is the attributes' set of possible values.

### Relationships

A relationship is an association between entities. Each relationship is identified so that its name is descriptive of the relationship. Relationships are represented by diamond-shaped symbols. A relationship's degree indicates the number of associated entities or participants. Three types of relationships are commonly used: unary relationships, binary relationships, and ternary relationship. The term Degree of Relationship is used to describe the relationship classification. It may be classified as one-to-one, one-to-many, and many-to-many. Cardinality expresses the specific number of entity occurrences associated with one occurrence of the related entity. Existence dependency means an entity's existence depends on the existence of one or more other entities. Relationship participation refers to either optional or mandatory. The generalization hierarchy depicts the parent-child relationship in the hierarchical database model, while in a relational database context; it depicts a relationship between a higher-level supertype entity and a lower-level subtype entity.

## 4. Developing an EER Model

Developing an EER Model for a problem domain requires several steps to be carried out [9, 13, 20]:

### Step 1: Identify entities

Define the main objects that the users are interested in. These objects are the entities for the model. The method of identifying entities is to examine the users' requirements specification. A common technique for extracting entities is by identifying nouns or noun phrases that are mentioned.

### Step 2: Identify relationships

Identify all the relationships that exist between these entities. Typically, relationships are indicated by verbs or verbal expressions.

### Step 3: Determine the multiplicity constraints of relationships

A model that includes multiplicity constraints more explicitly represents the semantics of the relationship and consequently results in a better representation of what is being modeled. Multiplicities are indicated by adjectives, determiners, model verbs and quantifies.

### Step 4: Identify and associate attributes with entities or relationships

Identify the types of facts about the entities and relationships chosen to be represented in the database. The attributes can be identified where the noun(s) or noun phrase(s) is a property, quality, identifier, or characteristic of one of the entities or relationships that has been previously found. A simple attribute is an attribute composed of a single component but a composite attribute is an attribute composed of multiple single components. In addition, an attribute can also be single-valued or multi-valued. A single-valued attribute is an attribute that holds a single value for an entity occurrence. A multi-valued attribute is an attribute that holds multiple values for an entity occurrence. Multi-values attribute may be indicated by quantifies.

### Step 5: Determine primary key attributes

This step is concerned with identifying the primary key for an entity. Primary keys are identified by adverbs.

### Step 6: Specialize/Generalize entities

The concept of specialization/generalization is associated with special types of entities known as superclasses and subclasses, and the process of attribute inheritance. The superclass is an entity that holds common attributes and relationships for all occurrences in the entity. Subclass is an entity that has a distinct role and holds specific attributes and relationships for some occurrences in the (superclass) entity. The concept of attribute inheritance is the process by which a member of a subclass may possess subclass- specific attributes, and inherit those attributes associated with the superclass.

Specialization is a top-down approach to defining a set of superclasses and their related subclasses. The set of subclasses is defined on the basis of some distinguishing characteristics of the entities in the superclass. When we identify a subclass of an entity, we then associate attributes specific to the subclass,

and also identify any relationships between the subclass and other entities or subclasses.

The process of generalization is a bottom-up approach, which results in the identification of a generalized superclass from the original subclasses. The process of generalization can be viewed as the reverse of the specialization process. If we apply the process of generalization on entities, we attempt to identify any similarities between them such as common attributes and relationships.

There are two important reasons for introducing the concepts of superclasses and subclasses into an ER model. The first is that it avoids describing similar concepts more than once. The second reason is that it adds more semantic information to the design in a form that is familiar to many people. In this step, there are two constraints that may apply to a superclass/subclass relationship called participation constraints and disjoint constraints. A participation constraint determines whether every occurrence in the superclass must participate as a member of a subclass. A participation constraint may be mandatory or optional. A superclass/subclass relationship with a mandatory participation specifies that every entity occurrence in the superclass must also be a member of a subclass. A superclass/subclass relationship with optional participation specifies that a member of a superclass need not belong to any of its subclass. The disjoint constraint describe the relationship between members of the subclasses and indicates whether it is possible for a member of a superclass to be a member of one, or more than one, subclass. If subclasses of a specialization/generalization are not disjoint (called nondisjoint), then an entity occurrence may be a member of more than one subclass. Generalization is indicated by entities with shared attributes and relationships.

## 5. User Requirements and Natural Language

NLP has several application tasks among these: Information Retrieval/Detection, Information Extraction, Question Answering Tasks, and Text Understanding (Artificial Intelligence).

User requirement analysis is an Information Extraction (IE) application of NLP. It is the identification of specific semantic elements within the user's requirements entered in textual form (i.e. entities, attributes, relationships, cardinalities and multiplicities). Almost all database design methodologies agrees on that in traditional conceptual modeling (such as Entity Relationship Diagrams or Object Role Modeling) nouns would end up as entities

(or attributes) and verbs would end up as relationships. Therefore, the role of NLP would be in identifying and extracting nouns, verbs, and other parts of speech (POS) needed in the determination of entities, relationships, attributes, cardinalities and multiplicities.

The relation between conceptual model and natural language has been analyzed in [16]. The authors noticed that efficiency and correctness of communication in the process of systems development could be improved when domain specialists can be confronted with natural language phrases expressing the change of the conceptual model.

The importance to generate natural language from conceptual model have been analyzed in [12]. The author noticed that most people do not understand formal languages, but they understand natural language, therefore it is desirable to have a tool, which automatically generates natural language from a formal specification.

### 5.1 DBDT Framework

[24] proposes several steps for processing natural language. These steps have been carried out by DBDT in order to process the user requirement entered in textual natural language.

#### 5.1.1 Morphological Analysis

Individual words are analyzed into their components, and non-word tokens (such as punctuation) are separated from words. Depending on the entities naming standards, entity's name must be in a singular form. We apply this standard using Morphological Analysis. We analyze individual nouns to make sure they are appropriate for entity name, by removing plural suffixes (s or es or ies) and converting plural entity names into singular.

#### 5.1.2 Syntactic Analysis

Linear sequences of words are transformed into structures that show how the words are related to one another. This parsing step converts the flat list of words of the sentence into a structure that defines the units represented by that list.

POS extraction is implemented in one of three ways:

– Performs POS extraction by matching with existent database that contain lists of all POS lexicon. This approach is prone to errors, since a given word can be found in two or more POS lists. For example: the lexicon (*advanced*) is found on both verb and

adjective lists. The lexicon (*Book*) is found on both noun and verb lists.

\_ Performs POS extraction by words special endings. The approach was tested on many of database cases, with an unacceptable percentage of correctness. For example, one of the famous noun's special ending is (- al). According to this, the lexicons (*several*) and (*rental*) are classified as nouns.

\_ Performs POS extraction by parsing sentences according to the English language grammar. This is the optimal approach adopted by DBDT. If the structure of the input sentence is grammatically correct, then each word is assigned to its appropriate POS category, depending on its position in the sentence. In DBDT, we parse each sentence according to English language grammar. As a result of this analysis: nouns, verbs, some determiners, adjectives and adverbs, that are playing the roles of keywords, are extracted.

Follows are basics of sentence structure presented in [3], and used by DBDT syntactic analysis approach:

- \_ A sentence is made up of one or more clauses.
- \_ A clause is made up of phrases.
- \_ Phrases are made up of words and other phrases.
- \_ Words are made up of morphemes.
- \_ Morphemes are basic building blocks of word including word's root and prefix / suffix.
- \_ Morphemes come together to form lexeme (word).
- \_ Words can be of many grammatical categories among these POS: Noun, Verb, Preposition, Adjective, Adverb, Conjunction, Article, Pronoun, Quantifies
- \_ Words come together to form phrases.
- \_ There are three main types of phrases: Noun phrase (NP), Verb phrase (VP), Prepositional phrase (PP)
- \_ In a multiword NP, main noun is called the head and other words are called modifiers
- \_ Modifiers can be of two types:
  - Specifier that indicates how many objects and their relationship. Such as Determiner (a, the, many, each...etc) and Numeric - ordinals (first) or cardinals (one)
  - Qualifier that occurs after specifier but before head, such as adjectives and other nouns.
- \_ Verb phrases contains head verb plus optional auxiliary verbs
- \_ Auxiliary verbs combine in different way to form sentences.
- \_ Prepositional phrases qualify other parts of sentence, and can be attached to verb or noun.
- \_ Phrases combine to form sentences, which express a complete thought.
- \_ Simple sentence can be subject-object, or NP-VP.
- \_ Next simplest has subject-verb-object, or NP-VP-NP
- \_ On the extreme, sentences can be very complex, with one embedded in another.

### 5.1.3 Semantic Analysis

The structures created by the syntactic analyzer are assigned meanings. This step must map individual words into appropriate objects in the knowledge base, and must create the correct structures to correspond to the way the meanings of the individual words combine with each other.

By semantic analysis we will be able to extract nouns that are playing the role of entities or attributes, and extract verbs that act as a relationship between entities. Also, cardinalities and multiplicities information may be extracted from determiners, adjectives, model verbs and quantifies.

### Recognition Techniques

Every word in a statement can contribute value to a conceptual model. We not only consider nouns and verbs to be relevant during analysis, but some of the other POS lexicon, can also add significant value to the collection and analysis of problems and requirements. The consideration of some adjectives, adverbs, prepositions and determiners during analysis enrich the process of relationships discovery and retain more of the semantic content and metaphoric richness of the source material. The following has been observed through reviewing a number of database cases:

- \_ Nouns contribute to entities or attributes.
- \_ Verbs contribute relationships between entities.
- \_ Determiners indicate cardinality of attributes and relationships.
- \_ Quantifies and some of modal verb (may/must) indicate multiplicity.
- \_ Adverb (uniquely) indicates PK of an entity.

### Entities

Most of relationships between entities are expressed in the form of simple sentence structure (Subject - Verb - Object). Focusing on the order meaning of sentence structure shows that subject and object are more likely to express entities that relate together using a verb. For example, in '*The student works on the project*', *Student* (subject) and *Project* (object) are detected as entities. In '*The project is assigned to many students*', *Project* (subject) and *Student* (object) are detected as entities.

### Attributes

Attributes are nouns mentioned along with their entity and are more likely preceded by the verbs has, have, or includes. These verbs indicate that an entity is attributed with a property. For example, in '*Student has id, name, and GPA*', *Student* is detected as an entity, and *Name*, *id* and *GPA* are detected as attributes.

### Relationships

The main verb that occurs between two entities (nouns) is more likely to be a relationship. Two entities can be separated by main verb only, by main verb and an auxiliary verb, or main verb and modal verb. For example, in ‘*The company is branched into many branches*’, *Company* and *Branch* are entities, and *Branched* is detected as their relationship.

### Cardinality

The number of nouns (singular or plural), modal verbs (e.g. must, can) and adjectives (e.g. not more than) determines the cardinality of relationship types or attributes [22]. For example, in ‘*each doctor treats many patients*’, the cardinality is 1:M. The noun's singularity or plurality affects cardinality as well. For example, in ‘*each doctor treats patients*’, the cardinality is 1: M. Determiners that preceded an attribute can be used to indicate multi-value attribute. For example, in ‘*Company has many telephones*’, *telephone* is detected as a multi-value attribute.

A noun or prepositional phrase whose noun is singular gets a minimal and maximum cardinality of 1.

### Multiplicity

Quantifiers that precede an entity can be used to indicate its Multiplicity. Moreover, there are some keywords that affect multiplicity such as (at most, at least, may, must). For example, in ‘*The doctor may treat at most 5 patients*’, the patient's multiplicity is (0,5). In ‘*The doctor must treat patients*’, the patient's multiplicity is (1,\*).

#### 5.1.4 Discourse Integration

The meaning of an individual sentence may depend on the sentences that precedes it and may influence the sentences yet to come. The entities involved in the sentence must either have been introduced explicitly or they must be related to entities that were. The overall discourse must be coherent. In DBDT, some sentences may begin with a pronoun that refers to a subject in the previous sentence; we have to detect this reference in order to be more precise in relation extraction. For example, in ‘*The company has many branches. It employs many employees*’, *It* in the previous sentence refers to *the company*.

## 5.2 More Improvement Techniques

### 5.2.1 Entities Filtering Techniques

Entities filtering techniques is applicable to entities naming standards, which emphasize on the entity name

to be unique within data model. This uniqueness will be accomplished by auto removal of duplicated entities and optional removal of synonym entities.

Duplication is removed automatically in both one-click and custom approach. Synonyms removal requires user involvement in custom approach. The approach detects all synonyms founded within a textual description. The user can choose one of the synonyms to replace all others. The new synonym will take all the attributes and relationships for the entity. For example, if *teacher* and *instructor* were detected within a textual description, such as in ‘*Teacher has name and address*’ and ‘*Instructor is uniquely identified by Id*’. When the user chooses *instructor* to be replaced by *teacher*, the *instructor* attributes and relationship are assigned directly to *teacher*. As a result, *teacher* will have *name*, *address* and *Id* attributes. *Id* will be the primary key of *teacher* entity.

### 5.2.2 Relating Techniques

#### Entity- relationship relating technique

Entities are related in order to represent meaningful relationships. Entities that occur in the same sentence are more likely to be related to each other to form a relationship. Relationship that occurs between the related entities is more likely to be a relationship between these two entities in the conceptual model.

For example, in ‘*Company employs many employee*’ and ‘*Many departments belong to the company*’, *Company* is related to both *employee* and *department*.

#### Entity- attributes relating technique

Attributes that occur in the same sentence where entity is found, are more likely to be attributes for this entity. For example, in ‘*Student has id, name, and GPA*’, *Student* is detected as an entity. *Name*, *id* and *GPA* are detected as attributes for *Student*.

The verbs *has*, *have*, and *include* can be used to relate entities and their attributes. DBDT removes entity-attributes duplication. When a noun is classified as both attribute and entity, most probably it represents an entity.

#### Entity- cardinality relating technique

Determiners that precede an entity in a sentence, most likely represent its cardinality. For example, in ‘*Doctor treats many patients*’ and ‘*Patients are treated by many doctors*’, the determiners are used to find the cardinality of the relationship (*treat*), that relates *doctor* with *patient*. The detected cardinality is M:N.

#### Entity- multiplicity relating technique

Quantifies that precedes an entity in a sentence most likely represents its multiplicity. For example, in ‘*Doctor may treat at most 6 patients*’ and ‘*Patients must be treated by at least 2 doctors*’, the multiplicity of both *doctor* and *patient* in the *treat* relationship are *doctor (0,6)* and *patient (2,\*)*.

### 5.3 Writing Requirements in DBDT

The following syntax constraints on writing user requirements would increase analysis and extraction accuracy rate.

\_ The sentences in the requirements should be separated by (.).

\_ Each sentence within the user requirement must be a declarative sentence.

\_ Each sentence must be in a simple form. That is, subject, verb and object respectively.

\_ The sentences syntax should follow a specified grammatical syntax based on the Context Free Grammar (CFG).

#### 5.3.1 Context Free Grammar (CFG)

Syntactic analyzer is considered to be as top-down parser. It parses each sentence of user requirement text until terminals are encountered. This is accomplished according to English Language Grammar specified in the form of CFG as follows:

- Sentence → Noun phrase + Verb phrase + Complement
- Noun phrase → Noun  
→ Pronoun  
→ Determiner + Noun
- Verb phrase → main verb  
→ Auxiliary verb + main verb  
→ Auxiliary verb + Adv. + main verb  
→ Modal verb + main verb
- Complement → Noun  
→ Determiner + Noun  
→ List of Nouns  
→ Prepositional phrase
- Prepositional phrase → Preposition + Noun  
→ Preposition + Determiner + Noun  
→ Preposition + Preposition + Adj. + Determiner + Noun
- Preposition → by, for, at, in, to....
- Pronoun → it
- Adverb → Uniquely
- Adjective → Most, least
- Determiners → the, a, an, each, many, several...

- Auxiliary verb → verb to have, verb to be
- Modal verb → can, could, may, must...

#### 5.3.2 Proposed Syntax

Abbreviations:

- N means Noun.
- V means Verb.
- Det means Determiner.
- Aux means Auxiliary.
- Adv means Adverb.
- Adj means Adjective.
- Prep means Preposition.

Notations:

- UPPER-CASE letters represents keywords.
- | indicates a choice among alternatives; (e.g. a|b|c).
- { } indicates a **required** element.
- ... Indicates **optional** repetition of a noun one or more times.

Rules:

- {N} | {Pronoun} | {Det + N}.
- +
- { V } | {Aux + V } | { Aux + Adv + V } | { Modal V + V }.
- +
- {N} | {Det + N} | {N,...N AND N}
- | {Prep + N}
- | {Prep + Det +N}.
- | {Prep + Prep + Adj + Det + N}.

According to the previous rules:

- The entity name should be one word in order to be detected correctly as an entity.
- The Attribute name should be one word in order to be detected correctly as an attribute.
- The Relationship name can be at most two words in order to be detected correctly as a relationship (Verb + Prep).
- Entered attributes must be in the form of Attribute1, Attribute2, Attribute3 AND Attribute4.

#### 5.3.3 Some Ways for Writing Sentences

##### Relationships

The	Teacher	Teaches	Student
Det	N	V	N

##### Primary Keys

Teacher	Is	Uniquely	Identified	by	id
N	Aux	Adv	V	Prep	N
Teacher	is	identified	by	id	

	<i>N</i>	<i>Aux</i>	<i>V</i>	<i>Prep</i>	<i>N</i>	
<b>Attributes</b>	Company	has	id	name	And	Address
	<i>N</i>	<i>Aux</i>	<i>N</i>	<i>N</i>	<i>AND</i>	<i>N</i>
<b>Cardinality</b>	Doctor	Treats	Many	patients		
	<i>N</i>	<i>V</i>	<i>Det</i>	<i>N</i>		
<b>Multiplicity</b>	Instructor	must	teach	at most	one	course
	<i>N</i>	<i>Model V</i>	<i>V</i>	<i>Adj</i>	<i>Det</i>	<i>N</i>
	Student	takes	at most	six	courses	
	<i>N</i>	<i>V</i>	<i>Adj</i>	<i>Det</i>	<i>N</i>	

## 6. Comparison with previous work

### 6.1 Comparison with other Case Tools in Databases Design

Four existing database design tools are reviewed; DeZign for Databases ([www.datanamic.com](http://www.datanamic.com)), CASE Studio 2 ([www.casestudio.com](http://www.casestudio.com)), Visible Advantage™ ([www.visible.com](http://www.visible.com)) and DBDesigner4 ([fabforce.net/dbdesigner4](http://fabforce.net/dbdesigner4)). Below are some distinct features of DBDTs in comparison with the above commercial tools:

- DBDT shares one common objective with the professional database designing tool, which allows user to visually create enhanced-ER model.
- DBDT provides two modes of database design; one-click or customized. In customized approach, the user will be guided to enhanced-ER modeling through a wizard.
- The analysis of the user’s requirements is absent in all the above tools but DBDT.
- Only DBDT and DBDesigner support weak entities.
- Only DBDT supports auto normalization of relations.

### 6.2 Comparison with other Application of NLP in Conceptual Modeling

An analysis of a number of works that apply NLP in conceptual design has been conducted. All studied works have user involvement during the process. Because of the incomplete presentation of knowledge, ambiguities and redundancies, full automation of the design process is fundamentally impossible [14]. As a consequence, the tools must be able to interact with the designer. A semi-automatic design process is far more economical than an entirely manual process [14].

[2] investigated the application of linguistic metaphors to object-oriented design. Cockburn's investigation leads us toward mechanisms for reducing adverbs and adjectives to verbs (that apply to nouns). While his investigation does not supply one, a linguistic model for such transformations between the primary word classes is rather straightforward. This paper offers such a model and suggests that adverbs and adjectives should not merely be reduced to their corresponding verbs. They need to be considered part of a larger framework - adverbs, adjectives and even verbs may also be reduced to nouns.

[11] introduced one of the first attempts to apply automated tools to requirements analysis and the automatic generation of object models from requirements documents. While the translation of the initial requirements into a suitable knowledge base requires human interaction to resolve ambiguities, the subsequent translation of the domain knowledge into object models is automated. Their process and tools can help a requirements analyst begin to bridge the gap between informal requirements and formal software models.

[18] presents a method to automatically generate a conceptual model from the system's textual descriptions of the use case scenarios in Spanish language. The requirements model is analyzed in order to establish the static structure (conceptual model) and dynamic structure (sequence diagrams, state diagrams) of the future system. Techniques of natural language processing (NLP) and conceptual graphs are used as the basis of the method.

[6] deals with a natural language dialogue tool for supporting the database design process. It illustrates how natural language (German) can be used for obtaining a skeleton design and for supporting the acquisition of semantics of the prospective database. The approach is based on the assumption that all nouns are transferred into entities and sentences with the main verb ‘have’ are transferred into an attribute.

[23] present a translation scheme for transforming natural language queries into relational algebra through the class diagram representations. Based on a logical form developed by extending the UML class diagram notations, a transformation model is presented to support the automatic transformation of natural language queries into relational algebra by employing appropriate NLP techniques and object-oriented analysis methods. The proposed logical form has the advantage that it can be mapped from natural language constructs by referring to the conceptual schema modeled by class diagrams, and can be efficiently transformed into relational algebra for query execution.

All previous researches address developing conceptual models using natural languages. However,

there is still much that needs to be accomplished to bridge the gap between natural language constructs and database schemas. Our project makes exhaustive use of NLP techniques to analyze the textual description of the problem domain by parsing sentences according to English language grammar and use of lexicons and knowledge bases. This allows us to consider text with a certain level of ambiguity and to cover relevant linguistic aspects, like complement and prepositional phrases.

Some distinct features of the transformation tool in DBDT compared with the above mentioned works are:

- The target users are database designers
- DBDT uses different improvement techniques to obtain the desired EER model.
- The transformation tool is also generic, in which it is not application domain-dependent.
- DBDT explores the use of natural language as a metaphorical basis for the structure (syntax) of sentences, and for the naming (semantics) of database components – entities, attributes, relationships, cardinalities and multiplicities.

## 7. Case Study

DBDT was tested on several case studies. This section illustrates the step carried out in producing an enhanced-ERD for the following case study.

‘A store has many branches. Each branch must be managed by at most 1 manager. A manager may manage at most 2 branches. The branch sells many products. Product is sold by many branches. Branch employs many workers. The labor may process at most 10 sales. It can involve many products. Each Product includes product\_code, product\_name, size, unit\_cost and shelf\_no. A branch is uniquely identified by branch\_number. Branch has name, address and phone\_number. Sale includes sale\_number, date, time and total\_amount. Each labor has name, address and telephone. Worker is identified by id’.

This work requires several steps to be carried out in order to achieve the desired EER model from the natural language input of the case study.

### Step 1: Read natural language text into system

To start with, a natural language input text would be read into DBDT. There are two ways to allow the natural language text into the system: reading from a text file or enabling the user to type in the problem in the provided workspace area. There are two modes of database design; one-click or customized. In case of customized method selection, the user will be guided to ERD through the wizard.

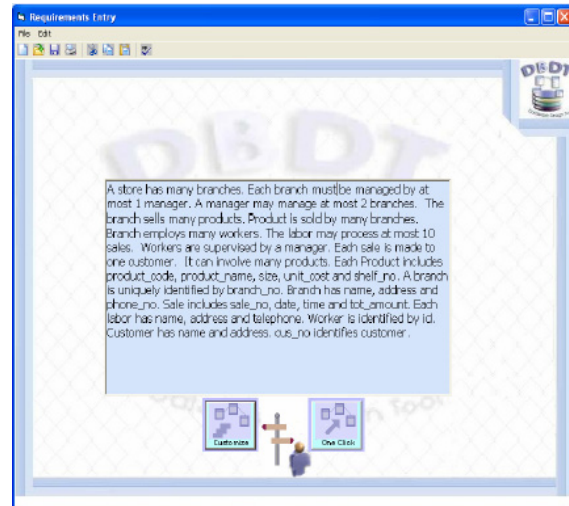


Figure 2. User Requirement Entry Screen

### Step 2: Morphological Analysis

This process analyses individual words and removes plurals. In this example, *workers* is replaced by *worker*. Entities are suggested in their singular form.

### Step 3: Syntactic Analysis

In order to obtain the corresponding syntactic category for each word, the sentences have to be parsed, and words need to be classified according to their syntactic position. Each word of the sentences is tagged according to their position in the sentence. An example of the result using the proposed Syntactic Analysis applied to scenario is shown in Table 1.

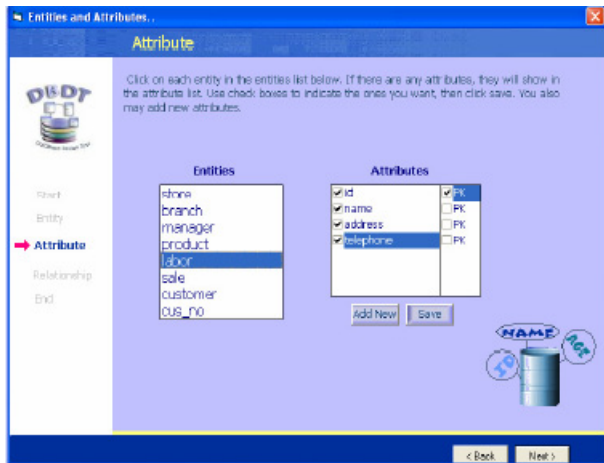
After reaching this stage, further steps need to be done to refine the result. The next step to be followed is classifying these tagged sentences into their corresponding category in relevance to ERD schema.

**Table 1.** Classification of words according to their syntactic analysis

Sentence No.	N	Aux	Det.	Model V	V	Prep	Adj.
1	Store Branch	Has	many				
2	Branch manager		must 1		Managed	By	
3	Manager Branches		2	may	Manage		At most
4	Branch Product		many		Sell		
5	Product branch		many		Sold	By	
6	Branch Worker		many		Employ		
7	Labor Sale		10	may	Process		At most
8	It Product		many		Involve		

**Step 4: Semantic Analysis**

This step will classify all the relevant words into their classes. Referring to the semantic analysis and recognition techniques presented earlier, table 2 classifies entities, attributes and relationship, in addition to cardinalities and multiplicities.



**Figure 3.** Snapshot of Semantic Analyzer

**Table 2.** A semantic classification of nouns

Sentence No.	Entity	Attribute
1	Store	Branch (multi-value)
9	Product	Product_code Product_name Size unit_cost Shelf_no
10	Branch	Branch_number (PK)
11	Branch	Name Address phone_number
12	Sale	Sale_number Date Time Total_amount
13	Worker	Name Address Telephone
14	Worker	Id (PK)

Figure 3 shows a snapshot of the results of the semantic analyzer. By clicking on any entity that is displayed in the entities list, its attributes will appear along with its primary key. The user can add or delete attributes.

**Table 3.** A semantic classification of sentences

Sentence No.	Entity	Relationship	Entity	Cardinality	Multiplicity
2	Branch	managed by	Manager	1:1	1..1
3	Manager	manage	Branch	1:M	0..2
4	Branch	sell	Product	1:M	1..*
5	Product	Sold by	Branch	1:M	1..*
6	Branch	employ	Worker	1:M	1..*
7	Labor	process	Sale	1:M	0..10
8	It	involve	Product	1:M	0..*

Once the classification has been done, some filtration techniques are applied to further determine which ERD category does a given word belongs to.

**Step 5: Discourage Integration**

This step will identify entities that were not introduced explicitly but through the selected relating previous sentences. Sentences that begin with a pronoun refer to a subject in the previous sentence, usually first identified entity in previous sentence. In our scenario, *It* is replaced by *labor*.

**Step 6: Filtration and relating techniques**

This process analyses individual entities and removes redundancies. Synonyms are detected and

suggested entity names are displayed according to the standards of entity naming. In this example, *labor* is replaced by *worker*. Suggested entities are in a singular form and without duplication.

**Step 7: Produce preliminary model**

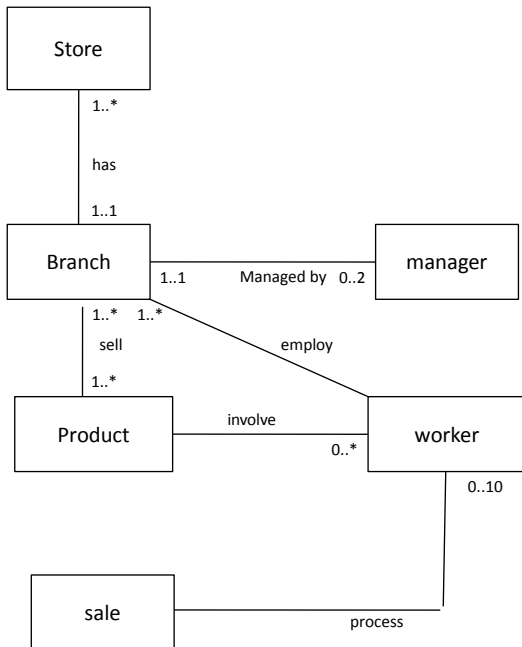
Once all the selected words have been assigned its ERD element type, a preliminary model will be produced by the system. Figure 4 represents the preliminary ER model of the case study. This model will be validated by the human designer in the next step to check if there are any discrepancies or missing elements. Missing cardinality and multiplicity indicators are not depicted. Entities that were identified by reference to previous sentences referred to *labor* (*worker*) while it was suppose to refer to *sale*.

**Step 8: Human intervention**

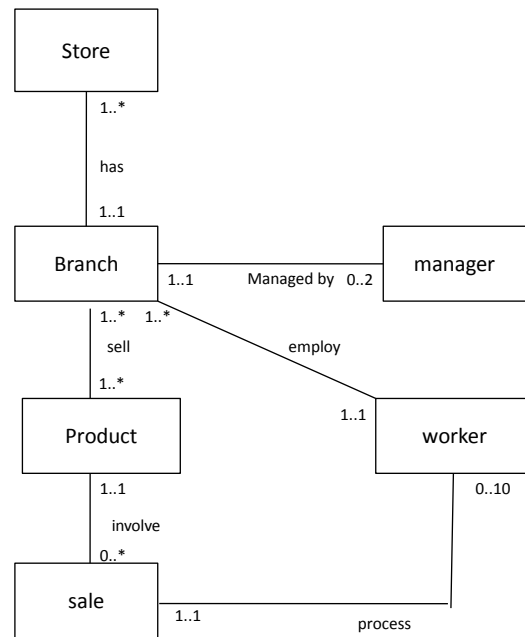
At this stage, designer would then instruct the system to make any necessary changes, if required. Three functions would be made available: Delete, Add and Update. The history file will be updated at this stage.

**Step 9: Produce final model**

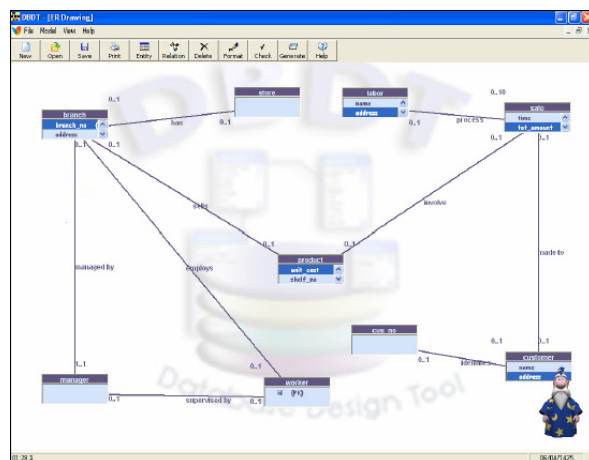
The final step is to actually produce the desired solution of the ER problem. Figure 5 below shows the amendment that has been done to the preliminary model.



**Figure 4.** A preliminary ER model of the case study



**Figure 5.** A final ER model of the case study



**Figure 6.** Snapshot of Resulted enhanced-ER Model

**8. Experimental Results**

DBDT has been tested on four cases

**Case study # 1**

A company has many departments. Each department may manage at most 3 projects. Each project is managed by at most 1 department. The company hires many employees. Each employee is assigned to at most 2 projects. Each project has

project\_id and budget. Each employee has name, employee\_id and address. Each employee is uniquely identified by Employee\_id.

**Case study # 2**

A library contains libraries, books, authors and patrons. Libraries are described by libname and location. Books are described by title and pages. Authors are described by authname. Patrons are described by patname and pat\_weight. A library can hold many books. The book can appear in many libraries.

**Case study # 3**

A hospital has patients. Each patient has name, address, sex and SSN. Patient is identified by patient\_id. Many doctors work in the hospital. Doctor treats many patients. Patient is treated by many doctors. Many departments are included in the hospital. Each department has at most 20 nurses.

**Case study # 4**

The company has at least 30 instructors. It must handle at most 100 trainees. The company offers at least 5 courses. Each course is taught by many instructors. Each instructor teaches at most 2 courses. Instructor may be assigned to research. Each trainee must undertake at least 1 course. Each company has id, name, address and net. The company is uniquely identified by id.

The results are presented in the following table to indicate the correctness percentages:

Case study #	Recognized Entities	Attributes recognition	Relationship recognition	Resulted ERD
1	100%	100%	100%	100%
2	80%	45%	50%	60%
3	80%	100%	84%	88%
4	100%	100%	100%	100%

By investigating the above case studies and the resulted ER models, we found that the more compliance a use case with the constraints on the syntax of written user requirements presented earlier in section 5, the more accurate the NLP can serve as a tool for auto user requirement analysis. In addition, the lexicon used while testing these cases affected the recognition results.

**9. Conclusion and Future Work**

DBDT is a CASE Tool that aimed at aiding Database designer in the Database development process. The developed model produced very satisfying results and features not provided by existing commercial CASE tools. Among these features auto analysis of user requirement using NLP and custom model database generation. Further work is required for auto recognition of ternary relationships, Composite and derived attributes, Relationship attributes, Fan trap and Chasm trap for detecting errors of an ER diagram, and Specialization/generalization.

*Acknowledgements.* We are grateful for prototype development by Fatma Al Al-Turkestani, Hala Al Al-Shamlan, May Al Al-Habeeb, Reham Al Al-Abduljabbar, Ghoyom Al Al-Shammmary, Hetoon Al Al-Sagri, Rima Al Al-Sager, Tahani Al Al-Manie.

**10. References**

- [1] J. Allen, "Natural Language Understanding", Addison Wesley, 2nd Edition, 1994.
- [2] A. Cockburn, "Using Natural Language as a Metaphoric Basis for Object-Oriented Modeling and Programming", IBM Technical Report TR-36.0002, 1992.
- [3] B. Azar, "Fundamentals of English Grammar", 2nd Edition, Prentice Hall, 1992.
- [4] G. Booch, J. Rumbaugh, I. Jacoson, "Unified Modeling Language User Guide", Addison-Wesley Professional 1999.
- [5] N. Boyd, "Using Natural Language in Software Development", Journal of Object-Oriented Programming, Vol. 11, No. 9, Feb. 1999.
- [6] E. Buchholz, A. Dusterhoft, "Using Natural Language for Database Design", Proceedings Deutsche Jahrestagung für KI 1994 – Workshop. Reasoning about Structured Objects: Knowledge Representation meets Databases, 1994.
- [7] R. Carasik, S. Johnson, D. Patterson, G. Glahn, "Towards a Domain Description Grammar: An Application of Linguistic Semantics". ACM SIGSOFT Software Engineering Notes, Vol. 15, No. 5, Oct. 1990, pp. 28-43.
- [8] N. Chinchor, "MUC-7 Information Extraction Task Definition". [http://www-nlpir.nist.gov/related\\_projects/muc/proceedings/ie\\_task.html](http://www-nlpir.nist.gov/related_projects/muc/proceedings/ie_task.html). (1998).
- [9] B. Connolly, C. Thomas, "DataBase Systems: A Practical Approach to Design, Implementation and Management". 4th ed. Harlow: Addison Wesley, 2005.
- [10] D. Cook, "Evolution of Programming Languages and Why a Language Is Not Enough To Solve Our Problems", CrossTalk: J. Def. Software Eng., Vol. 12, No. 12, 1999, pp. 7– 12.

- [11] D. Cordes, D. Carver, "An Object-Based Requirement Modeling Method", *Journal of the American Society for Information Science* Vol. 43, No. 1, 1992, pp. 62-71.
- [12] H. Dalianis, "Concise Natural Language Generation from Formal Specifications", Ph.D. Thesis, (Teknologic Doktorsavhandling), Department of Computer and Systems Sciences, Royal Institute of Technology/Stockholm University, 1996.
- [13] C.J. Date, "Introduction to Database Systems, An (8th Edition)", Addison Wesley, 2003.
- [14] C. Eick, O. Lockemann, "Acquisition of terminological knowledge using database design techniques" In Proc. SIGMOD, 1985.
- [15] W. Hersh, "Information Retrieval". <http://distance.ohsu.edu/~BBuser/documents/MINF514/IR9.pdf> (2004).
- [16] J. Hoppenbrouwers, B. van der Vos, S. Hoppenbrouwers, "NL Structures and Conceptual Modelling: The KISS Case" In R. van de Riet, J. Burg, A. van der Vos, editors, *Application of Natural Language to Information Systems*, 1996, pp. 197—209.
- [17] M. Loginov, A. Mikov, "Automated Problem Domain Cognition Process in Information Systems Design", *International Journal Information Theories & Applications*, Vol.14, 2007.
- [18] A. Montes, H. Pacheco, H. Estrada, O. Pastor, "Conceptual Model Generation from Requirements Model: A Natural Language Processing Approach", *Proceedings of the 13th international conference on Natural Language and Information Systems: Applications of Natural Language to Information Systems*, London, UK ISBN: 978-3-540-69857-9, 2008, pp. 325 – 326.
- [19] Motoshi Saeki, Hisayuki Horai, Hajime Enomoto., "Software Development Process from Natural Language Specification". *Proceedings of the 11th International Conference on Software Engineering (ICSE-11)*, IEEE Computer Society Press, 1989.
- [20] P. Rob, C. Coronel, "Database Systems: Design, Implementation, and Management", *Course Technology*, 8 edition, (2007)
- [21] N. Stageberg, D. Oaks, "An Introductory English Grammar", Wadsworth Publishing, 5th edition, 1999.
- [22] A. Tjoa, L. Berger, "Transformation of Requirements Specifications Expressed in Natural Language into an EER Model", *Proceeding of the 12th International Conference on ER-Approach*, Airlington, Texas USA, 1993.
- [23] F. Tseng, C. Chen, "Enriching the class diagram concepts to capture natural language semantics for database access", *Data & Knowledge Engineering*, Vol. 67, No. 1, 2008, pp. 1-29.
- [24] Tsujii, J. "Information Extraction from Scientific Texts". [www.phil-fak.uniduesseldorf.de/~rumpf/talks/Tsujii.pdf](http://www.phil-fak.uniduesseldorf.de/~rumpf/talks/Tsujii.pdf) (2004).