

# English Sentence Structures and EER Modeling

Sven Hartmann

Sebastian Link\*

Department of Information Systems, Information Science Research Centre  
Massey University, Palmerston North, New Zealand  
E-mail: [S.Hartmann,S.Link]@massey.ac.nz

## Abstract

An input to the conceptual database design phase contains a description of the target database. This description is usually given in some natural language, for instance in English. Conceptual design aims at transforming English sentences into a conceptual database schema. A conceptual data model should therefore possess modeling features that can represent English sentence structures.

Since Chen himself has argued that the basic ER model has such constructs many extensions of this basic ER model have been proposed. Based on these new features we revise the several correspondences between English sentence structures and concepts of ER modeling. It turns out that EER modeling can provide both a well-defined semantics and improved modeling elements that naturally reflect English language sentence structures.

## 1 Introduction

The Entity-Relationship (ER) model has evolved into one of the most popular conceptual data models since its introduction in the late 1970s (Chen 1976). It provides an excellent communication tool between systems analysts, database designers, managers and potential database users during the crucial process of identifying user information requirements. Since these requirements are usually documented in a natural language, such as English, the database designer faces the difficult task to decipher the documentation and specification into a conceptual database schema, e.g. an ER schema.

Some research has been devoted towards finding guidelines that support the conversion of natural language descriptions into ER schemata and diagrams (Chen 1983). However, this research was mainly focused on the basic ER model. Since its proposal the research community has put a lot of effort in finding extensions of the ER model (EER)(Batini, Ceri & Navathe 1992, Thalheim 1997, Thalheim 2000).

**Contributions.** The purpose of this paper is to review the basic rules that state correspondences between English sentence structures and EER modeling features (Chen 1983). In particular, we study the impact of new ER features, such as specialisation, generalisation, higher-order relationship types and collection types, on the heuristic guidelines. Our findings suggest that these new features refine and very

much improve the existing correspondences. Moreover, they help to clarify the reason for introducing these additional modeling features and thus the reasoning process of ER modeling in general. The perhaps most striking fact is that higher-order relationship types allow the database designer to formally represent the naturally complex structure of English text, and in particular the interactions between different sentences. Therefore, the ER model is a conceptual database model with a well-defined semantics that does not only provide the starting point for high-quality and safe database design but also provides modeling features which very much resemble the structure of natural languages. This demonstrates that EER modeling is very suitable for transforming user requirements into a conceptual database schema. We have summarised our guidelines in Table 1.

**Related Work.** Our article is based on Chen's original guidelines (Chen 1983). Since then a reasonable number of proposals have been made to use natural language processing in database design (Buchholz, Cyriaks, Düsterhöft, Mehlan & Thalheim 1995, Eick & Lockemann 1985, Gomez, Segami & Delaune 1999, Omar, Hanna & Mc Kevitt 2004, Tjoa & Berger 1993, Tsen, Chen & Yang 1992). Mostly, these proposals deal with semi-automatic tools that support natural language processing, a detailed comparison of these proposals can be found in (Omar et al. 2004). Unlike previous research we specifically study the impact of EER features on the original guidelines.

**Organisation.** The article is organised as follows. In Section 2 we re-define the basic concepts of Entity-Relationship modeling, and summarise important extensions including specialisation, cluster types, collection types, nested attributes and higher-order relationship types. We also give brief examples to illustrate these concepts. The basic guidelines for converting English sentence structures into ER schemata and diagrams are reviewed in Section 3 strongly emphasising the impact of EER features. In Section 4 we apply our heuristics to some English text and demonstrate how a conceptual database schema can be derived gradually from a user requirements specification. We conclude in Section 5 and give a brief outline of future work.

## 2 Entity-Relationship Modeling

We will summarise the basic concepts of the ER data model, and then provide an overview of many of its extensions based on (Thalheim 2000).

### 2.1 Entity Types

An *entity type*  $E = (attr(E), id(E))$  consists of a name  $E$ , a finite and non-empty set of attributes  $attr(E)$  such that each attribute  $A \in attr(E)$  has

\*This research was supported by Marsden Funding, Royal Society of New Zealand

Copyright ©2007, Australian Computer Society, Inc. This paper appeared at The Fourth Asia-Pacific Conference on Conceptual Modelling, Ballarat, Australia. Conferences in Research and Practice in Information Technology, Vol. 67. John F. Roddick and Annika Hinze, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

Correspondences	
English sentence concept	EER feature
transitive verb	relationship type
common noun	component of relationship type
adjective	attribute of component
adverb	attribute of relationship type
numerical expression	attribute of object type
preposition	role name of component
gerund	relationship type that is component of another relationship type
clause	relationship type with components
complex sentence	relationship type of order higher than 1
alternative phrase	cluster type
plural	collection type/nested attribute
“is a” sentence	specialisation

Table 1: The correspondence between English language concepts and EER features

a domain  $dom(A)$ , and a key  $id(E) \subseteq attr(E)$  whose elements are called key attributes. An *entity* over  $E$  is a mapping  $e : attr(E) \rightarrow \bigcup_{A \in attr(E)} dom(A)$  such

that  $e(A) \in dom(A)$  holds for all  $A \in attr(E)$ . An *entity set*  $E^t$  over  $E$  is a finite set of entities over  $E$  that satisfy the unique key value property, i.e., for all  $e_1, e_2 \in E^t$  with  $e_1(A) = e_2(A)$  for all  $A \in id(E)$  we must have  $e_1 = e_2$ . We use  $ent(E)$  to denote the set of all entities of an entity type  $E$ .

**Example 2.1.** We can specify an entity type CLIENT as follows: its attribute set is  $attr(CLIENT) = \{Name, Birthday, Address, Phone\}$  with domain assignment  $dom(Name)=STRING$ ,  $dom(Birthday)=DATE$ ,  $dom(Address)=STRING$ , and  $dom(Phone)=NUMBER$ . The key attributes of CLIENT are  $id(CLIENT) = \{Name, Birthday\}$ . An entity set may consist of the following three clients:

(John Fox, 08/08/1980, 88 Main Street, 3508888),  
(John Fox, 02/12/1967, 23 Te Awe Awe, 3539465),  
and  
(Lisa Hunter, 02/12/1967, 7 Park Ave, 356 1154).

Note that none of these clients has the same values on all key attributes.  $\square$

We visualise entity types by rectangles together with its name inside. If desired, the attributes can be attached to the rectangle, and the key attributes can be underlined. We implicitly assume that we know the domains of the attributes. The entity type CLIENT from Example 2.1 is visualised in Figure 1.



Figure 1: Visualisation of the entity type CLIENT

## 2.2 Relationship Types

A *relationship type*  $R = (comp(R), attr(R), id(R))$  consists of a name  $R$ , a finite, non-empty set of components  $comp(R)$ , a finite set of attributes  $attr(R)$  such that each attribute  $A \in attr(R)$  has a domain  $dom(A)$ , and a key  $id(R) \subseteq comp(R) \cup attr(R)$  whose elements are called key components and key attributes, respectively. A *relationship* over  $R$  is a mapping  $r : comp(R) \cup attr(R) \rightarrow \bigcup_{E \in comp(R)} ent(E) \cup$

$\bigcup_{A \in attr(E)} dom(A)$  such that  $r(E) \in ent(E)$  for all

$E \in comp(R)$  and  $r(A) \in dom(A)$  for all  $A \in attr(E)$ . A *relationship set*  $R^t$  over  $R$  is a finite set of relationships over  $R$  that satisfy the unique key value property, i.e., for all  $r_1, r_2 \in R^t$  with  $r_1(X) = r_2(X)$  for all  $X \in id(R)$  we must have  $r_1 = r_2$ . We refer to entity and relationship types jointly as *object types*. Note that we use *set semantics* to describe relationships (Thalheim 2000). At the moment,  $comp(R)$  consists of entity types only. However, we will discuss other options for components shortly.

**Example 2.2.** Consider the entity type DVD-COPY= $(\{Title, Director, Year, CopyNr\}, \{Title, CopyNr\})$  in which we leave the domains implicit. We specify the relationship type RENTAL= $(\{CLIENT, DVD\}, \{RentalDay, DueDay\}, \{DVD, RentalDay\})$ . The two relationships

((John Fox, 08/08/1980, 88 Main Street, 3508888), (The Godfather, F.F. Coppola, 1972, 1), 04/01/2006, 05/01/2006),  
((John Fox, 02/12/1967, 23 Te Awe Awe, 3539465), (The Godfather, F.F. Coppola, 1972, 1), 05/01/2006, 07/01/2006).

form a relationship set over RENTAL.  $\square$

Relationship types are commonly visualised by a diamond. It is linked by edges to the rectangles representing its components. For key components, the corresponding edge is marked by a dot. If desired, the attributes are attached to the diamond. Key attributes are again underlined. The object types from Examples 2.1 and 2.2 are visualised in Figure 2.

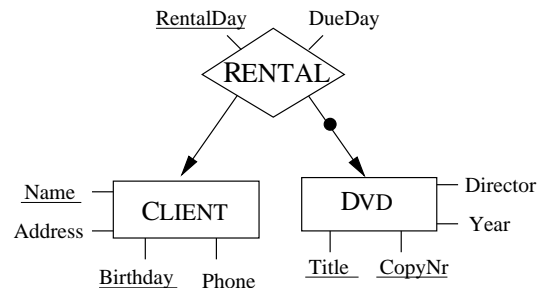


Figure 2: Visualisation of Relationship type RENTAL and its components

## 2.3 Relationship Types with Role Names

It may well occur that a relationship type must be used to model relationships between objects of the same component. In order to avoid confusion we associate distinct *role names* with the components. Role names can also be utilised to improve the readability of the ER diagram.

As an example, we may specify the relationship type DESCENDANT with components  $comp(DESCENDANT) = \{Child : CLIENT, Parent : CLIENT\}$ , attributes  $attr(DESCENDANT) = \emptyset$  and key  $id(DESCENDANT) = comp(DESCENDANT)$ . When we visualise the relationship type DESCENDANT we simply draw two edges to the entity type Client and label them by the corresponding role names Child and Parent. The example is visualised in Figure 3.

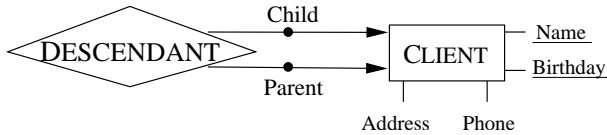


Figure 3: Visualisation of Role Names

## 2.4 Specialisation and Generalisation

Sometimes, objects in the target of the database can be represented by more than just a single abstract concept. For instance, animals are living creatures, mammals are animals, and therefore also living creatures. Consequently, it would be good to derive abstract concepts from other abstract concepts, such as the more specific concept MAMMAL from the more general concept ANIMAL. This idea is known as *specialisation*. The derived object type is a *subtype* of the more general *supertype*. A subtype inherits all features of its supertype, but often adds some new properties.

The subtype  $U$  may be modelled as a unary relationship type whose single component is just its supertype  $C$ . Clearly,  $U$  may have some additional attributes, and we may use  $C$  as the key for  $U$ , i.e.  $U = (\{C\}, attr(U), \{C\})$ . Note that every object of type  $C$  gives rise to at most one object of type  $U$ .

Occasionally, it is desirable to model alternatives, e.g., having a relationship to various kinds of objects. For example, a department may employ lecturers or tutors or general staff. For this it would be good to have an abstract concept that models all of them, that is, we would like to comprise several object types to a single new type. This idea is known as *generalisation* since the new abstract concept is more general than each of the individual ones.

A *cluster type*  $U$  consists of a finite, non-empty set  $comp(U)$  of components  $C_1, \dots, C_n$ , normally  $n \geq 2$ . We denote this cluster type by  $C_1 \oplus \dots \oplus C_n$ . In order to use role names, we also allow components of the form  $p_i : C_i$  rather than simply  $C_i$ . Clusters model disjoint unions, i.e., an object set  $\mathcal{I}(U)$  associated with a cluster type  $U$  is the disjoint union of its component's object sets  $\mathcal{I}(U) = \bigcup_{i=1}^n \{(i, o) \mid o \in \mathcal{I}(C_i)\}$ .

We visualise cluster types by the symbol  $\oplus$  attaching the name of the cluster type to it, and drawing (labeled) edges from  $\oplus$  to the components of the cluster type. An example of specialisation and generalisation can be found in Figure 4.

## 2.5 Collection Types

It becomes sometimes necessary to model collections of objects. For example, a course might be taught by more than a single lecturer and the readings of this course may consist of a ranking of articles or book chapters. In such cases it is advantageous to have an abstract concept modeling a finite collection of objects of the same type. That is, we would like to derive a new object type from a given one by apply-

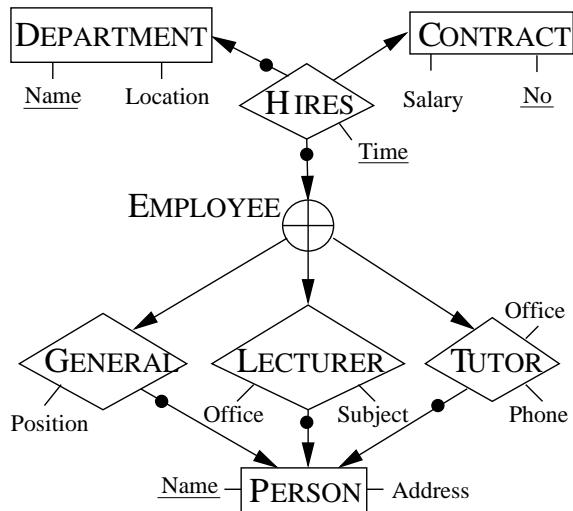


Figure 4: Specialisation and Generalisation

ing some kind of collection constructor. Usually, we can distinguish between four kinds of collections:

1. lists in which duplicates are allowed and order matters,
2. sets in which duplicates are not allowed and order does not matter,
3. bags in which duplicates are allowed and order does not matter, and
4. rankings in which duplicates are not allowed and order does matter.

A *list-, set-, bag- or ranking-type*  $U$  has a single component  $C$ , i.e.  $comp(U) = \{C\}$ . We denote a list type by  $U[C]$ , a set type by  $U\{C\}$ , a bag type by  $U\langle C \rangle$  and a ranking type by  $U[C]$ . The object set  $\mathcal{I}(U)$  associated with a collection type  $U$  is just a set of finite lists (finite sets, finite bags, finite rankings, respectively) of object in  $\mathcal{I}(C)$ . Collection types are visualised using a circle together with the corresponding symbol of the collection type inside, and drawing a (labeled) edge to its component. As an example we may consider entity types ARTICLE, BOOK and LECTURER as illustrated in Figure 5. The object READING is a cluster type with components ARTICLE and BOOK. We then specify a set-type LECTURERS with component LECTURER that models finite sets of lecturers. Intuitively, every such set models a group of lecturers who deliver a course together. Moreover, we specify a ranking-type READINGS with component READING. This ranking-type models a list of recommended readings for each course. The relationship type COURSE models information about course offerings where each course is described by its course id, course name and the semester the course is held together with a ranking of recommended readings and a set of lecturers that deliver this course. The ER diagram can be found in Figure 5.

## 2.6 Higher-Order Relationship Types.

So far, we have only allowed entity types to occur as components of relationship types. However, the examples above and best practice suggest to allow also relationship types, cluster types and collection types to occur as components of another relationship type. For convenience, we call these four kinds of types jointly *object types*. Entity types are object types without components while all other object types have at least one component. We need to ensure that the

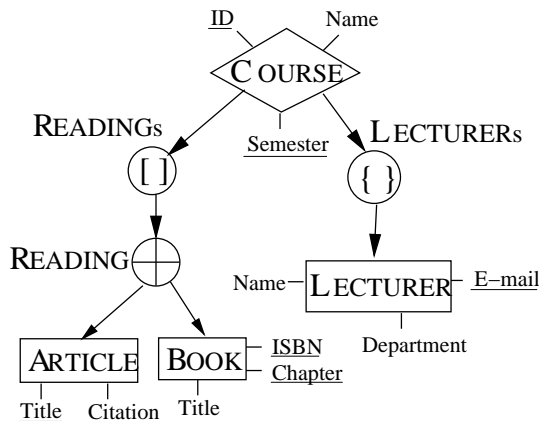


Figure 5: Collection Types

components of an object type are well-defined. For that we assign an order to each object type. Let  $U$  be an object type with component set  $comp(U)$ . The order of  $U$  is

- 0, if  $U$  is an entity type,
- $k$ , if all components of  $U$  have order less than  $k$  and at least one of its components has order  $k - 1$ .

It is a simple exercise to extend the definitions of relationship and relationship set, correspondingly (Thalheim 2000).

An *Entity-Relationship schema* (ER schema) is a finite set  $\mathcal{S}$  of object types such that for each object type  $U$  in  $\mathcal{S}$  and each of its components  $C$  or  $p : C$  in  $comp(U)$  we have that the object type  $C$  belongs to  $\mathcal{S}$  as well. An *instance*  $\mathcal{I}$  of an ER schema  $\mathcal{S}$  assigns each object type  $U$  in  $\mathcal{S}$  an object set  $\mathcal{I}(U)$  such that

- for each relationship type or cluster type  $U$  in  $\mathcal{S}$ , for each of its components  $C$  or  $p : C$ , and for each object  $o \in \mathcal{I}(U)$  we have that  $o(C)$  or  $o(p : C)$  belongs to  $\mathcal{I}(C)$ , and
- for each collection type  $U$  in  $\mathcal{S}$ , and for its single component  $C$  or  $p : C$ , for each collection  $O \in \mathcal{I}(U)$  we have that every  $o \in O$  belongs to  $\mathcal{I}(C)$ .

An *Entity-Relationship diagram* (ER diagram) of an ER schema  $\mathcal{S}$  is a directed graph with the elements of  $\mathcal{S}$  as nodes, and with edges from a node  $U$  to a node  $C$  for all components  $C \in comp(U)$ , and edges from node  $U$  to node  $C$  labelled with  $p$  for all components  $p : C \in comp(U)$ .

## 2.7 Nested attributes

So far, we have only used flat attributes to describe the properties of entity or relationship types. However, such properties might also be rather complex. For this purpose, nested attributes have been introduced that arise from flat attributes by applications of some type constructors (Thalheim 2000). Examples of such nested attributes are  $Address(City, Zip, Street(Name, No))$  resembling the format of an address on letters using the record constructor, or  $Readings[Article(Title, Citation) \oplus Book(Title, ISBN, Chapter)]$  modeling a list of readings consisting of articles or book chapters. We will not repeat the formal definition of nested attributes (Thalheim 2000) and their associated domains since their use and value should be intuitive from this simple examples. We assume from now on that attributes of entity and relationship types may also be nested.



Figure 6: Illustration of Heuristic 1

## 3 Heuristics for English Sentence Structures

We will now revise the basic guidelines for converting English language structures into ER schemata and diagrams (Chen 1983).

**Heuristic 1 (Common Nouns and Transitive Verbs).** *Transitive verbs relate common nouns to one another. In this case the transitive verb corresponds to a relationship type and the common nouns correspond to components of this relationship type.*  $\square$

**Comparison to Chen’s proposal.** Heuristic 1 summarises Rule 1 and 2 (Chen 1983). The heuristic provides more flexibility than the original heuristics in which common nouns correspond to entity types. Descriptions in natural language usually consist of more than one sentence, and the meaning of the sentences are coupled rather than independent. Common nouns may therefore refer to abstract concepts which do not necessarily refer to entity types but to relationship types of higher order. The feature of allowing higher order relationship types soundly reflects the complex nesting of natural language text. It is also very natural in supporting the relationships between different sentences. Note that the order of the relationship type in Heuristic 1 is not constrained, and the number of its components can be any arbitrary positive integer to allow maximum flexibility.  $\square$

**Example 3.1.** *Consider the English statement A person drives a vehicle to a destination. Here, the transitive verb drive relates the three nouns person, vehicle and destination to one another. According to Heuristic 1 we obtain a ternary relationship type DRIVE together with three components PERSON, VEHICLE and DESTINATION. The statement by itself suggests that PERSON, VEHICLE and DESTINATION correspond to entity types. However, the context of the statement may suggest otherwise. For instance, PERSON may already be some subtype. In that case, PERSON corresponds to a relationship type of higher order.*  $\square$

The next two heuristics are new and do not have an analogue in Chen’s proposal.

**Heuristic 2 (Specialisation).** *The phrase “is (a)” can relate two common nouns  $X$  and  $Y$  to one another, say “ $X$  is a  $Y$ ”. In this case  $X$  corresponds to a subtype with a supertype that corresponds to  $Y$ .*  $\square$

**Example 3.2.** *Let us consider the sentence A customer is a person who buys products at a store. This statement can be divided into the two statements A customer is a person and A customer buys products at a store. The first of these sentences suggests to have a unary relationship type CUSTOMER with component PERSON. Applying Heuristic 1 to the second sentence suggests to have a relationship type BUY with components CUSTOMER, PRODUCT and STORE. As before, the context of the sentence will decide the order of these object types. Figure 7 shows the simplest case in which PERSON, PRODUCT and STORE represent entity types, CUSTOMER a unary relationship type of order 1 and BUY a ternary relationship type of order 2.*  $\square$

The last example illustrates the candidacy of prepositions as role names of components.

**Heuristic 3 (Prepositions).** *Prepositions within prepositional verbs refer to an object of the sentence. In this case the preposition corresponds to the role name of the component corresponding to the object.* □

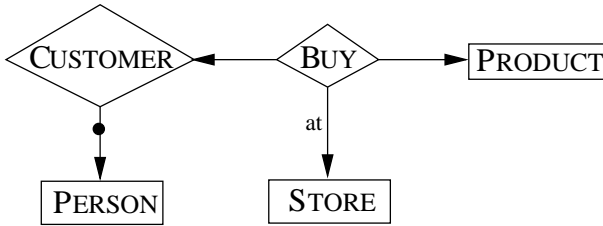


Figure 7: Illustration of Heuristics 1, 2 and 3

Note that the ER diagram in Figure 7 also illustrates Heuristic 1. In particular, the component CUSTOMER of BUY is a relationship type itself.

So far, there are no attributes in the ER diagrams of our examples. This is natural since we have looked at statements without considering their context. Usually, the context contains further information that is important to characterise object types, and such information can occur in form of adjectives and adverbs.

**Heuristic 4 (Adjectives and Adverbs).** *An adjective corresponds to an attribute of a component, and an adverb corresponds to an attribute of a relationship type.*

**Comparison to Chen’s proposal.** Heuristic 4 generalises Rule 3 and 4 of Chen’s original proposal. Again, it provides more flexibility since the order of the relationship type and components are not constrained. □

**Example 3.3.** *Let us consider the statement*

A 25-year old customer buys a 200 dollar watch paying with her credit card.

together with the sentence of the previous example. The adjective phrase 25-year-old is modifying the noun customer suggesting that age becomes an attribute of CUSTOMER. Similarly, the phrase 200 dollar watch indicates to use price and name as an attribute of PRODUCT. Finally, the adverbial phrase paying with her credit card modifies the verb buy, i.e., we may use the attribute method of payment as attribute of the relationship type BUY. □

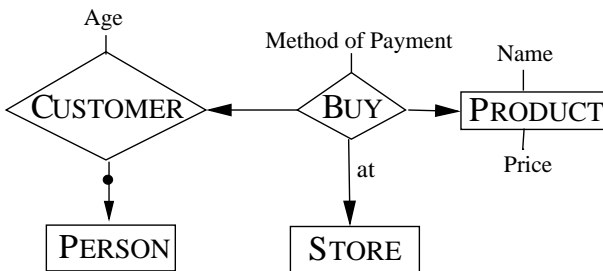


Figure 8: Illustration of Heuristic 4

Since we are mainly interested in the semantics of sentences we may use equivalent forms of sentences in order to apply our guidelines.

**Heuristic 5 (Sentence Conversion).** *If a sentence has the form: “There are ... X in Y”, we can convert it into the equivalent form “Y has ... X”. Heuristic 1 can be applied to the latter sentence in case X is considered to be an abstract concept or X may correspond to a (nested) attribute of the object type that corresponds to Y. Alternatively, X can also correspond to a collection type that is component of the object type corresponding to Y.* □

**Comparison to Chen’s proposal.** Heuristic 5 corresponds to Chen’s Rule 5. The heuristic adds the possibility of using nested attributes or also collection types as components. The preference for these alternatives will be dependent on the context of the sentence and the modeling objectives. □

**Example 3.4.** *Consider the statement There are over 500 players participating in this event. An equivalent statement may say that The event has over 500 players. According to Heuristic 1 this may result in a relationship type HAS with components EVENT and PLAYER. In such a database schema collection types are completely avoided. Alternatively, we may have a relationship type EVENT with a set-valued type PLAYERS={PLAYER} as its component. This provides the database user with three different object types about which information is stored in the database. Finally, we may have an object type EVENT with a nested attribute Participating{Player}. In this case there is only one object type. While this scenario represents a very compact ER diagram there will not be any separate tables in which information about players is stored. The three alternatives are illustrated in Figure 9. The database designer will base the choice on the context in which the English statement appears.* □

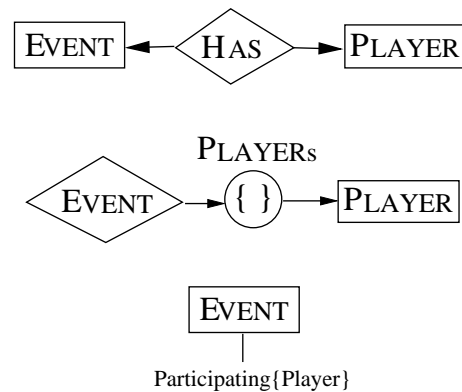


Figure 9: Illustration of three Alternatives for Heuristic 5

Chen’s rules 6,7 and 8 deal with sentence restructuring and algebraic operations, and mainly indicate those concepts in English language that help to identify attributes of object types in an ER schema. Since the additional ER features do not have a major impact on these rules, and also due to lack of space, we will not go into further details of these heuristics.

Recall that transitive verbs correspond to relationship types. A gerund is a noun that has been converted from a verb. It can therefore represent an object on which further relationships may be defined on within another clause.

**Heuristic 9 (Gerunds).** *A gerund corresponds to a relationship type which is the component of a further relationship type.* □

**Comparison to Chen’s proposal.** Heuristic 9 generalises Rule 9 of Chen’s proposal. There are no longer restrictions on the order of the object types. Moreover, the somehow unnatural concept of *relationship-converted entity type* that is used in Rule 9 becomes unnecessary in the framework of higher-order object types. Heuristic 9 is therefore also more natural than the original Rule 9. □

**Example 3.5.** Consider the sentence Agassi plays Sampras, and playing is supervised by Lars Graf. Here, Agassi and Sampras represent object of the object type PLAYER, and Lars Graf represents an object of the object type REFEREE. The transitive verb play corresponds to a binary relationship type MATCH with two components first:PLAYER and second:PLAYER. Moreover, SUPERVISE is a relationship type with components MATCH and BY:REFEREE. □

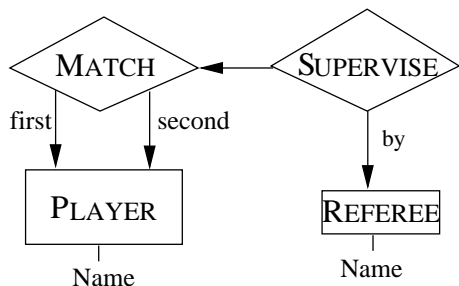


Figure 10: Illustration of Heuristic 9

Clauses are major building blocks of English sentences. They are used to build another clause or are decomposed further into subclauses. The application of the next correspondence will usually result in higher order relationship types.

**Heuristic 10 (Clauses).** A clause refers to a relationship type with its components. □

**Comparison to Chen’s proposal.** This is similar to the previous comparison. Heuristic 10 appears to be more natural than Rule 10, simply because of the concept of higher-order object types. □

**Example 3.6.** We analyse the statement Managers decide which machine is assigned to which employee. The main clause which machine is assigned to which employee results in a binary relationship type ASSIGNMENT with components MACHINE and EMPLOYEE. The decision on the assignment results in a further binary relationship type DECIDE with components MANAGER and ASSIGNMENT applying Heuristic 1. □

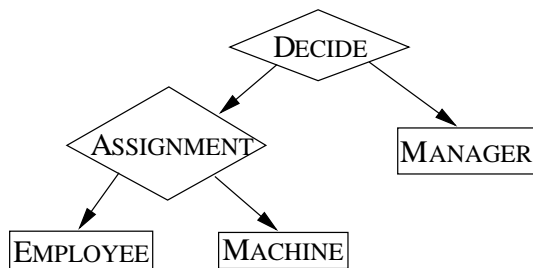


Figure 11: Illustration of Heuristic 10

Complex sentences contain at least one main and one dependent clause. The clauses themselves may be divided into further subclauses.

**Heuristic 11 (Sentences).** A complex sentence corresponds to a relationship type which has at least another relationship type as its component. □

**Comparison to Chen’s proposal.** As seen before, the framework of higher-order object types makes Heuristic 11 appear more natural than Rule 11 of the original proposal. □

**Example 3.7.** Consider the English statement A tour is organised by a travel agency into day trips on which tourists visit various sights and are led by a tour guide. The statement is divided into a main and a relative clause. First, we obtain a ternary relationship type DAYTRIP with components TOURIST, SIGHT and GUIDE. Moreover, we derive another ternary relationship type ORGANISE with components TOUR, AGENCY and DAYTRIP. □

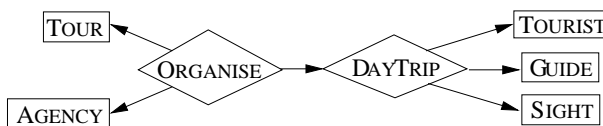


Figure 12: Illustration of Heuristic 11

The remaining two heuristics do not have counterparts in Chen’s original proposal since they refer to new ER features not present in the basic ER model.

**Heuristic 12 (Alternatives).** In a sentence of the form “...Z is (either) X or Y...” Z corresponds to a cluster type whose components are the object types that refer to X and Y. □

**Example 3.8.** The sentence Academics are either lecturer or tutors, and employees are either academics or general staff results in the cluster types  $ACADEMIC = LECTURER \oplus TUTOR$  and  $EMPLOYEE = GENERAL-STAFF \oplus ACADEMIC$ . □

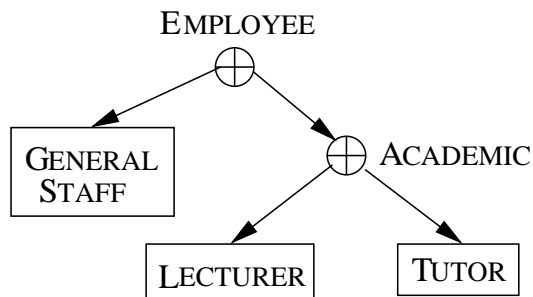


Figure 13: Illustration of Heuristic 12

**Heuristic 13 (Collections).** In a sentence of the form “...X is a collection of Y...” X corresponds to a collection type whose single component is the object type that Y refers to. It depends on the semantics of the sentence if list, bag, set or ranking type is used. Furthermore, the usage of plural in an English sentence may indicate the presence of a collection type or a nested attribute. □

**Example 3.9.** We analyse the sentence A customer purchases a bag of products. *Heuristic 1* tells us to define a relationship type PURCHASE with a component CUSTOMER and a further component. This further component should represent a bag of products, i.e., we obtain a bag type PRODUCTS(PRODUCT). Moreover, the sentence A lecturer has several phone numbers. may indicate to use a relationship type LECTURER with the collection type PHONENUMBERS{PHONE} as its component. Based on this choice the database users can access three different tables to find information about lecturers, collections of phone numbers, and individual phone numbers, respectively. Alternatively, we may have an object type LECTURER with the nested attribute Numbers{Phone}. In this case a database user can access only one table to find information about lecturers. The database designer will base his choice on the context in which the English statement appears. □

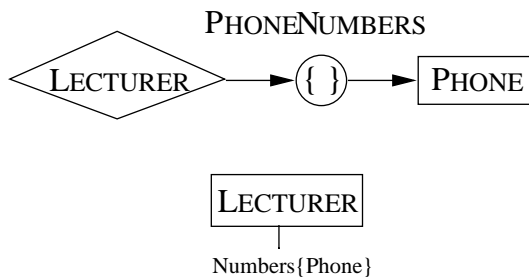


Figure 14: Illustration of Heuristic 13 and use of nested attributes

#### 4 A Step-by-Step Example

In this section, we will apply our heuristics not to single English sentences but a (small) English text. We believe that the additional features of ER modelling, in particular the higher-order types, naturally represent the interactions between different sentences.

Suppose the following English language specification is the output of the requirements analysis: *The database stores information about a university. Each person should have a name, and an address that person lives at. Students, general staff, lecturers and professors are all persons. Every student has a student ID and a majoring subject. For the general staff we keep track of their position. Lecturers have a department where they work and a certain teaching area. Professors are also associated with a department, and have several areas of research expertise. Graduate students are students with a degree and study a specific topic. They are supervised by either lecturers or professors within a semester. Courses have a course number and a title, and are taught by a number of lecturers and professors within a certain semester. The course co-ordinators teach courses on the basis of a list of recommended textbooks each of which comes with a title and an ISBN.*

We will now apply our heuristics to convert this specification into an ER diagram. The first sentence gives an indication about the application domain but does not identify any relevant object types that we need to model.

The next two sentences identify PERSON as an abstract concept. Having read the entire specification one may decide to specify PERSON as an entity type. The second sentence can be broken into the sentences: *Each person has a name* and *Each person has an address that person lives at.* Since *name* and *address*

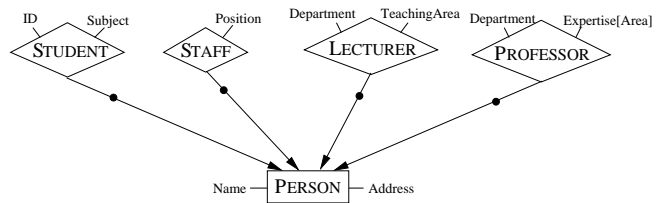


Figure 15: ER-diagram after analysing Sentences 1-7

are not abstract concepts, but properties of the entity type PERSON we apply Heuristic 5 to identify *name* and *address* as attributes of PERSON.

The next sentence is equivalent to the following sentences: *A student is a person; Each general staff is a person; A lecturer is a person; and A professor is a person.* Using Heuristic 2 for these sentences we can identify several subtypes of PERSON, namely the unary relationship types STUDENT, STAFF, LECTURER and PROFESSOR all having the single component PERSON.

Applying Heuristic 5 to the sentence *Every student has a student ID and a majoring subject* we identify *ID* and *Subject* as attributes of the subtype STUDENT. In the same way we identify *Position* as an attribute of the subtype STAFF, *Department* and *TeachingArea* as attributes of LECTURER, and *Department* as an attribute of PROFESSOR. Moreover, the sentence *Professors have several areas of research expertise* points us to applying Heuristic 13 and identifying *Expertise[Area]* as a ranking-valued attribute of PROFESSOR, too. Here, we choose a ranking of areas since each area of research will be mentioned precisely once and in the order of relevance for the professor. This results in the ER diagram in Figure 15.

The sentence *Graduate students are students with a degree and study a specific topic* can be broken into the two sentences: *A graduate student is a student* and *A graduate student has a degree and studies a specific topic.* We can apply Heuristic 2 to the first of these two sentences and identify GRADUATE as a subtype of STUDENT. Subsequently, we can apply Heuristic 5 to identify *Degree* and *Topic* as attributes of GRADUATE.

The sentence *Graduates are supervised by either lecturers or professors within a semester* is divided into the following two sentences: (i) A supervisor is either a lecturer or a professor, and (ii) Graduate students are supervised by a supervisor within a semester. Applying Heuristic 12 yields a cluster type SUPERVISOR with components LECTURER and PROFESSOR. Applying Heuristics 1, 3 and 4 to the second sentence yields a binary relationship type SUPERVISE with components GRADUATE and *by:SUPERVISOR*, and attribute *Semester*. The refined ER diagram is illustrated in Figure 16.

We will now analyse the sentence *Courses have a course number and a title, and are taught by a number of lecturers and professors within a certain semester.* This sentence may be broken into the following sentences: (i) Courses have a course number and a title, (ii) Courses are taught by a number of teachers within a certain semester, and (iii) A teacher is either a lecturer or a professor. The first of these sentences identifies COURSE as an entity type, and applying Heuristic 5 *CourseNo* and *Title* as attributes of COURSE. Applying Heuristic 12 to the third sentence yields a cluster type TEACHER with components LECTURER and PROFESSOR. Notice the importance of having two distinct cluster types SUPERVISOR and TEACHER with the same components. A supervisor does not necessarily teach any courses and neither

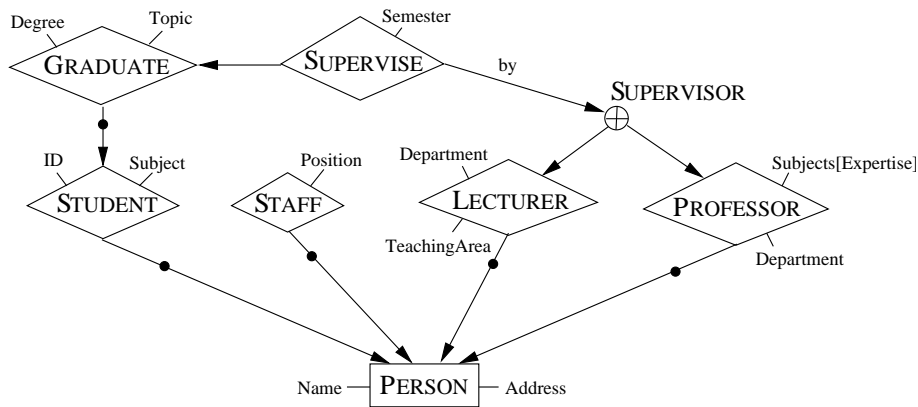


Figure 16: ER-diagram after analysing Sentences 1-9

does a teacher necessarily supervise any graduates. Finally, the second sentence points to a relationship type TEACH with components COURSE and a set type TEACHERS with single component TEACHER. These object types are motivated by Heuristics 1 and 13. Moreover, Heuristic 4 indicates to use *Semester* as an attribute of TEACH.

It remains to analyse the sentence *The course coordinators teach courses on the basis of a list of recommended textbooks each of which comes with a title and an ISBN*. We divide this sentence into the following parts: (i) A textbook has a title and an ISBN, and (ii) Teachers teach courses on the basis of a list of recommended textbooks. The first part identifies TEXTBOOK as an entity type with attributes *Title* and *ISBN* (Heuristic 5). The second part introduces a ranking type READINGS with component TEXTBOOK based on Heuristic 13, and suggests to use READINGS as a component of TEACH (Heuristics 1 and 13). The final ER diagram is illustrated in Figure 17.

## 5 Conclusion and Future Work

We have reviewed the basic guidelines that link together English sentence constructs and ER modeling elements (Chen 1983). We feel that this review is beneficial since many new and useful ER features have been added since the introduction of the basic ER model (Chen 1976, Batini et al. 1992, Thalheim 2000). Indeed, the usefulness of these additional modeling features is confirmed by our analysis. The extended ER model can offer not only a well-defined modeling framework but also a highly practical modeling language in which all of its features correspond to natural language elements.

We would like to stress that the use of higher-order object types not only guarantees sound semantics but also reflects the naturally complex structure of English text. Sentences appear together in a context in such a way that they represent the intended meaning. Therefore, the sentences are coupled accordingly and structured hierarchically. Such a hierarchy can be represented naturally by higher-order object types. Another advantage of higher-order types is that additional information can be easily embedded in an existing schema. This appears to be particularly valuable in component-based development and design (Thalheim 2005).

A challenging problem is that of *integrity constraint acquisition* using natural language processing (Albrecht, Buchholz, Düsterhöft & Thalheim 1995). Candidates for *key constraints* may be indicated by “key” words such as *distinct* or *unique*, but there are plenty of other classes of integrity constraints (Thalheim 2000).

## References

- Albrecht, M., Buchholz, E., Düsterhöft, A. & Thalheim, B. (1995), An informal and efficient approach for obtaining semantic constraints using sample data and natural language processing., in ‘Semantics in Databases’, pp. 1–28.
- Batini, C., Ceri, S. & Navathe, S. (1992), *Conceptual Database Design: An Entity-Relationship Approach*, Benjamin/Cummings.
- Buchholz, E., Cyriaks, H., Düsterhöft, A., Mehlan, H. & Thalheim, B. (1995), Acquiring complex information from natural language for EER database design, in ‘NLDB’.
- Chen, P. P. (1976), ‘The Entity-Relationship model: Towards a unified view of data’, *Transactions on Database Systems (TODS)* **1**, 9–36.
- Chen, P. P. (1983), ‘English sentence structure and entity-relationship diagrams’, *Information Science* **29**, 127–149.
- Eick, C. & Lockemann, P. (1985), Acquisition of terminology knowledge using database design techniques, in ‘SIGMOD’, pp. 84–94.
- Gomez, F., Segami, C. & Delaune, C. (1999), ‘A system for the semiautomatic generation of ER models from natural language specifications’, *DKE* **29**(1), 57–81.
- Omar, N., Hanna, P. & Mc Kevitt, P. (2004), Heuristics-based entity-relationship modelling through natural language processing, in ‘Fifteenth Irish Conference on Artificial Intelligence and Cognitive Science (AICS-04)’, pp. 302–313.
- Thalheim, B. (1997), The strength of ER modeling, in ‘Conceptual Modeling’, pp. 227–242.
- Thalheim, B. (2000), *Entity-Relationship Modeling*, Springer-Verlag.
- Thalheim, B. (2005), ‘Component development and construction for database design’, *DKE* **54**(1), 77–95.
- Tjoa, A. M. & Berger, L. (1993), Transformation of requirement specifications expressed in natural language into an EER model, in ‘Entity-Relationship Approach’, number 823 in ‘Lecture Notes in Computer Science’, Springer, pp. 206–217.
- Tsen, F., Chen, A. & Yang, W.-P. (1992), ‘On mapping natural language constructs into relational algebra through E-R representation’, *DKE* **9**, 97–118.

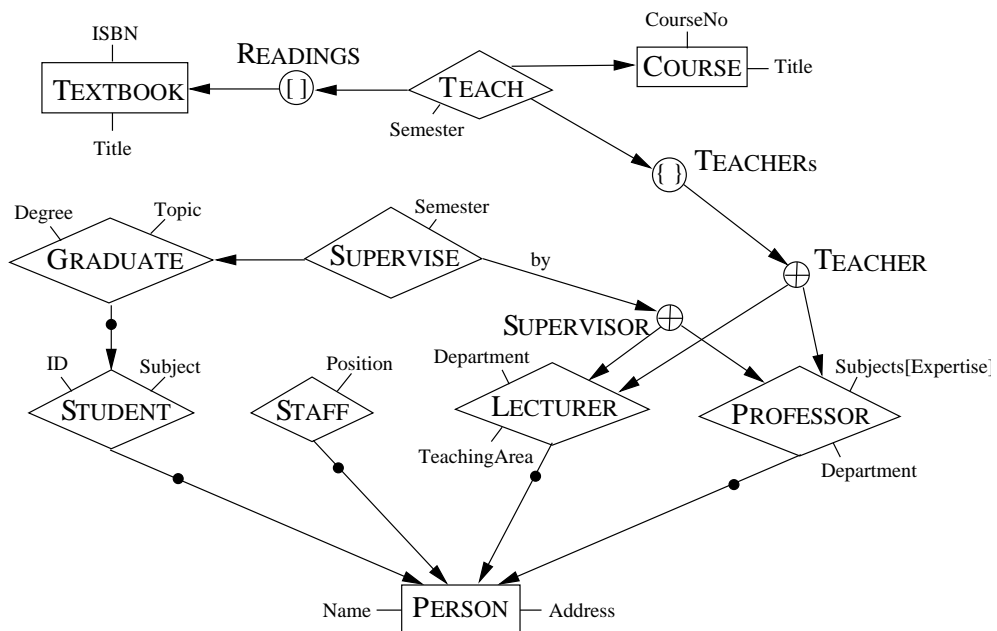


Figure 17: Final ER-diagram