A Natural Language Interpreter for the Construction of Conceptual Schemas

Leone Dunn and Maria Orlowska

Key Centre for Software Technology
Department of Computer Science
The University of Queensland
St. Lucia, Q.4067 Australia

ABSTRACT

The problem of designing a relational database for an information system is a complex one; it involves the analysis of an informal, and often incomplete, statement about some enterprise and then formal application of the established theories.

This paper describes the first phase of the design of a natural language interpreter for constructing NIAM (Nijssen's Information Analysis Method) conceptual schemas. The entire system is intended as a tool for modelling relational databases. The analyst interacts with the system about a particular Universe of Discourse (UoD) using example sentences in natural language. From the example input, the system constructs the conceptual schema of a particular UoD. During the process, the analyst may subsequently retrieve and alter the information until the conceptual schema design phase is complete. The model is used to construct the relational database. The first phase of the system described in this paper constructs a 'first pass' at elementary fact types, the basic units of knowledge representation in NIAM. This involves the following:

- allocating entity type names and role names on the basis of sentence instances supplied by the analyst;
- automatic support for splittability decision, the key factor in determining elementary fact types.

1. Introduction

The problem of designing a relational database is a complex one. Not for nothing is the Universe of Discourse (UoD) represented as a cloud. Not for nothing have information systems researchers spent the past fifteen years or so developing conceptual theories to formalize this cloud. For example, the E-R model[1] and NIAM[2].

If we take NIAM for instance, there are also a number of precisely defined steps that guide the analyst through the conceptual schema design phase of the paper model. Even then, the analyst has problems in going from the unstructured definition of a UoD to the identification of the basic units of information in NIAM namely, the identification of elementary fact types and their component parts - entity types and roles. (It is assumed the reader is familiar with NIAM. Otherwise they are referred to [2]).

For example, the following sentences describe a very limited UoD:

(1) Wirth wrote Pascal in 1975.

(2) Smith migrated to Australia in 1925.

Allocating the surface objects *Pascal, Australia, 1975/1925* to entity types (semantic classes) e.g. **language, country, date,** is problematic. The decision as to whether *Wirth* and *Smith* are common entity types is more difficult. Whether these objects should be allocated to a common semantic class, such as *person*, or whether they should be allocated to separate classes, such as *creator* and *migrator*, depends on the context of a particular UoD.

Assuming that type allocation has been successful, another problem is determining whether the fact types, *person wrote language in date* and *person migrated to country in date*, are elementary, or whether they are splittable into smaller semantic units. Neither syntactic nor semantic criteria alone can determine splittability. The same fact types could be splittable in one UoD but not in another. The decision as to whether a fact type is splittable depends on other factors, namely whether a referent is unique and specific in a particular UoD.

The major issue in designing the interpreter was to provide a simple, easy-to-use facility that assists the expert through these initial difficult stages of conceptual schema design as automatically as possible, though some user interaction is necessary. Other major factors influencing the design were:

- The system is intended as a tool for designing real life (and often very large) databases. It is not intended as an experimental toy;
- The system is domain-independent. It is intended that the system can be used for designing conceptual schemas for any UoD;
- Natural language is the means of interaction with the system. Natural language was chosen because it frees the applications expert from having to learn the complexities of

every new CASE tool that becomes available.

The research area focussing on database design tools is currently very active[3]. One tool designed specifically for constructing NIAM conceptual schemas is the Semantic Data Dictionary (SDD). The SDD assumes as a starting point the existence of the elementary fact type. The major part of this research is aimed at assisting the analyst in the identification of elementary fact types and therefore does not overlap.

2. System Architecture

The main function of the interpreter is to assist the expert in designing a relational database for an information system by means of the NIAM conceptual schema. It allows the user to define the UoD using (a subset of) natural language. It provides automatic support for the determination of uniqueness constraints and for the splittability decision, the key factors in determining elementary fact types. It assists the user in solving the problem of naming conventions (common entity types and roles), another key factor in determining elementary fact types. It also provides automatic support for selecting the derived fact types from the set of elementary fact types. Once the elementary and derived fact types have been established, the relational tables can be automatically created. Therefore, the system is conceptually divided into the following modules. The Type Allocation Module, the Elementary Fact Type Construction Module, the Naming Convention Module, The Derived Fact Type Construction Module, and the Relational Database Construction Module. Figure 2-1 shows the overall structure of the system.

The Type Allocation Module and the Elementary Fact Type Construction Module are described in this paper.

3. The Natural Language Interpreter

The core of the Type Allocation Module is the natural language interpreter. Like most natural language processing systems, the interpreter was designed for a specific task, namely, the construction of NIAM conceptual schemas. Because of the nature of the task, ie the fact that the the system is domain-independent, requiring the interpreter to engage in knowledge acquisition where the input is almost always unknown, existing theories and methodologies commonly used in natural language processing systems could not be used in the present system.

3.1. Background Issues

Natural language processing (NLP) is one of the oldest areas of artificial intelligence (AI) research. In recent years, a lot of research has been undertaken on context-based disambiguation. That is, the use of context to disambiguate sentence and text meaning for language interpretation[4][5] and utterance-planning [6][7].

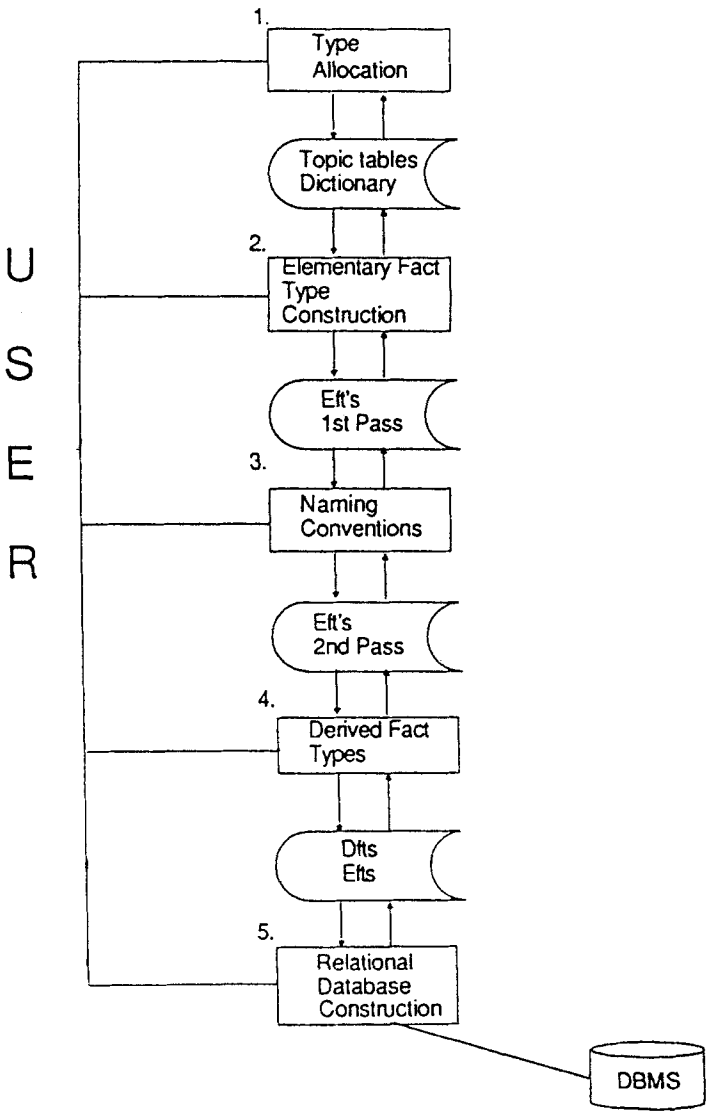

Figure 2.1 System Architecture

Most systems developed so far assume large bodies of prestored knowledge,making use of the now widely-accepted knowledge representation formalisms, such as [8][9][10][11][12] and the like. Learning systems, though concentrating on much smaller UoDs[13] also make use of these 'predictive' discourse structures. When unknown words or phrases are encountered in the input, the stereotyped encoded situations provide a predictive context for disambiguating meaning. The present NLP system, which is to be used as a database design tool, has to be more flexible. It would be inefficient to prestore knowledge in a system of this type. The knowledge required would have to be at such a high level of abstraction requiring complex inferencing mechanisms to disambiguate the meaning of the input. Also, due to the potential size of the system - a UoD description may contain thousands of fact types, it was not feasible to engage heavily in knowledge acquisition dialogue, as is the case with some of the more experimental systems designed for knowledge acquisition [14]. Rather than prestore knowledge, an interactive parser has been developed[16], which has some commonalities with one developed for a Japanese machine translation project[15].

The ambiguities dealt with at this stage by the interpreter are schema ambiguities, ie. ambiguities of (fact) types and (fact) instances[16]. The 'common entity type' problem , which had to be taken into account in the initial design stages, is a problem for all conceptual models and not considered to be a weakness of the present system. For large databases, a dictionary of synonym classes has to be set up prior to beginning the UoD description. The system then helps the analyst by collecting types and querying synonym classes[17].

4. The Type Allocation Module

Type refers to the entity types, or semantic classes of objects in the UoD. The major function of type allocation is to translate the natural language input sentences into their underlying semantic forms. Each instance of a natural language sentence represents one role-type pattern in the UoD. The semantic pattern of each input sentence is identical to the topic table, described below in 4.3.

4.1. Input to Type Allocation

The input to the type allocation module is a subset of natural language summarized by the following grammar.

1. Sentence -> NP Vgp Obj* (Reduced_S)*
2. NP -> (det) instance|(det) type|(det) type "," instance

3. Obj -> (Prep) NP
4. Reduced_S -> AND Vgp Obj*
5. Vgp -> (Aux) Verb
6. det -> the,this,that,those,a,an,all,each,....
7. instance -> any nominal string
8. type -> any nominal string
9. verb -> any verb string

4.1.1. Examples

(a) NP (Noun Phrase) strings.CS112(instance),the student,Adams(det type "," instance), the 3rd of January,1989(det instance), person(type), all students(det type).

(b) Vgp (Verb Group) strings. studies(verb), is enrolled(aux verb).

(c) Obj (Object) strings. on the 3rd January,1989(prep NP),CS112(NP).

(d) Reduced_S strings. and studies CS112(AND Vgp Obj), and is held in CS 718 and is taught by Halpin(Reduced_s Reduced_S).

(e) Sentence strings. The following sentences are interpreted by the grammar.
CS112 is held on Mondays at 3pm during semester 1 and is taught by Halpin.
The 3rd of January is a date.
Jones is enrolled in a BSc and studied CS112 and scored 7.
All students have id-codes.

4.2. Grammatical Restrictions

The subset of natural language was chosen for the initial version of the interpreter because of its suitability for database description. While it is not expected that the user enter only functional dependencies, at the same time the system is not in tended as an experiment in natural language processing. Future versions of the system will extend the natural language processing capability without change to the existing theory or methodology. At present, the following restrictions apply.

(a) The only form of the sentence that is recognized is the declarative. Questions and commands are not processed at this stage.

(b) Complex sentences joined by and are parsed only if they have a common subject NP. No other ellipsis is dealt with.

(c) Sentence modification is adverbial by means of prepositional phrases only. Adjectival

modifiers, such as relative clauses and other adjectival phrases are not dealt with in the present system.

(d) Only complete information about types may be entered. Other sentence connectives, such as *or* will not be parsed.

(e) Quantification is syntactic only. If a noun phrase begins with a quantifier, such as *all,each,every* etc., the input sentence is treated in the same way as any other input sentence. The semantics of quantification is also intended for future versions.

4.2.1. Type Allocation Input/Output

(a) The Instance - Type Dictionary. The form of the Instance - Type Dictionary table is:

Word Category Type

---

where Word refers to the lexical item itself, Category refers to the syntactic category of the lexical item, and Type refers to the entity type, or semantic class of the word (where relevant). The structure of the dictionary is uniform for both language specific and world knowledge. (b) Topic Tables. The topic table has the following form TT = [S,<R1,O1>,<R2,O2>,...<Rn,On>], as shown below:

Subject	Role	Objects
Syntactic Type: Semantic Type: Instance: Unique:	Role1	Syntactic Type: Semantic Type: Instance:
	.	.
	.	.
	Rolen	Syntactic Type: Semantic Type: Instance: Unique:

It conforms to the linguistic theory convention of representing (complex) sentences as a sequence of grammatical relations. Grammatical relations in this grammar consist of both syntactic and semantic features. That is, working from left to right, a sentence consists of a

subject and a number of role-object pairs, which make up the predicate. Objects are classified in this grammar as either direct or indirect. Direct objects are the noun phrases or prepositional phrases immediately following the verb(s) in the sentence. Prepositional objects are all other noun or prepositional phrases in the sentence. This is necessary for the construction of NIAM elementary fact types.

Subjects and objects have four features which are used by various modules in the system:
SyntacticType - refers to subject or object type,
SemanticType - refers to the entity type of a subject or object,
Instance - refers to the actual surface word or phrase,
Unique - refers to an integer value used for numbering uniqueness constraints.

Roles are expressed by the main verb and any prepositional case roles present. The case roles used by this system are shown in 4.5.

For every sentence entered by the user a topic table is constructed. The topic is the subject or focus of the sentence. It is the entity type about which attributes, or relationships to other entity types are classified. So that in a UoD which talks about students and their enrolments, the topic or subject of the sentence is the student. The subject must appear in first place. On the other hand,if the UoD were about degrees, for example, the topic would be degree and must be expressed with degree in the first position. It is possible to represent any sentence using this approach.

Example. The topic table constructed for the first example input sentence given in 4.1. is (including only SemanticType features for subject and object): TT=[subject,<is-held-on,day>,<at-loc,time>,<during-loc,semester>- ,<is-taught-by,lecturer>], as in:

Subject	Role(s)	Object Type(s)
SyntacticType:Subject	is_held_on	SyntacticType:Direct
SemanticType:Subject		SemanticType:Day
Instance:CS112		Instance:Monday
Unique:		Unique:
	at_location	SyntacticType:Prep
		SemanticType:Time
		Instance:3pm
		Unique:
	during_location	SyntacticType:Prep
		SemanticType:semester
		Instance:semester 1

	Unique:
is_taught_by	SyntacticType:Direct
	SemanticType:Lecturer
	Instance:Halpin
	Unique:

4.3. Output of the Type Allocation Module

(a) Type Tables. For each topic table constructed during type allocation , a corresponding type table is presented to the analyst on the completion of this stage. A type table T has the form T = [S,O1,O2,O3,...On] eg.

Object-Type	Object-Type1	Object-Type2	....	Object-Typen
Subject-instance	Object-instance1	Object-instance2	....	Object-instancen

That is, the columns are ordered corresponding to the underlying form of the declarative sentence, where S stands for the subject entity type and O stands for object types. The semantic types are used as column headings. The example instance from the topic table is presented as a row value. The type table corresponding to the above topic table is T = [subject,day,time,lecturer].

subject	day	time	lecturer
CS112	Mon	3pm	Halpin

The analyst is requested at the completion of type allocation to populate these tables with a small but significant set of examples relevant to a particular UoD. Rather than have the user key in a natural language sentence for each instance, this method reduces the amount of dialogue interaction and has less potential for conflicting or redundant information.

4.4. Prestored Knowledge

The only prestored knowledge in the system, along with the parser, is a table of entries for closed word classes used by the parser. These are:

Word	Category	Type
in,on,under,..	preposition	location
from,out of,off	preposition	source
to,into,onto	preposition	goal
with,by	preposition	instrument
via,per,over	preposition	path
for	preposition	purpose
the,this,that,..	determiner	nil
a,an	determiner	nil
each,all,every	determiner	nil
some	determiner	nil
is,are,was,were	aux,verb	nil
has,have,had	aux,verb	nil

4.5. Function of Type Allocation

The Type Allocation module is responsible for the following:

- interpretation of the natural language sentences ;
- maintenance of the Instance-Type Dictionary;
- allocation of types and roles;
- rejection of more than one instance of the same instance-type / role-type pattern;
- querying the possibility of the same input sentence having been represented in a different form;
- querying any type inconsistencies or incomplete information;
- interacting with the analyst concerning any additional or updated information relating to type specification;
- providing the user with empty type tables in preparation for elementary fact type construction.

4.6. Algorithm PROCESS_SENTENCE

The high-level algorithm for processing input sentences.

Input: a subset N of natural language sentences S.

Output: a set of Topic Tables TT representing the underlying semantic form of sentence(s). The updated Instance_Type dictionary ITD.

PROCESS_SENTENCE(N)

begin

```
while N do begin
 parse S and interactively determine fact types and instances; If instance then check dictionary
else if type then check topic information
 else reject sentence; If information correct then store
  else send message to user querying inconsistencies
     revise accordingly
  end;
return(TT,ITD)
end.
```

The worst case time complexity of this algorithm is O(n2).

Conceptually, the construction of the dictionary and the construction of the topic tables are two separate stages. When the parser encounters an input sentence of the form <instance> IS A <instance/type>, the dictionary construction module is invoked. The high-level algorithm for dictionary construction is as follows:

4.7. Algorithm DICTIONARY_CONSTRUCTION

Input: A subset N of natural language sentences of type IS_A.

Output: A set of updated Instance-Type Dictionary tables ITD.

DICTIONARY_CONSTRUCTION(N)

```
begin while N do begin
  parse IS_A;
  check dictionary for instance;
  If instance exists and corresponding type matches input type
     send message to user and reject pattern
 else if instance exists with different type
     send message to user to confirm type pattern(s);
 check type in dictionary;
 if type exists and corresponding instance matches input instance
     send message to user and reject pattern
 else if type exist with different instance
     send message to user to confirm patterns;
 update instance-type knowledge in dictionary
end;
```

return(ITD)
end.

Discussion

If the analyst had entered the sentence *Adams is a student*, and had repeated at some stage the example, the system would reject the pattern. If, on the other hand, the analyst had entered *Adams is a manager*, the system queries the seemingly conflicting information to determine whether the fact that Adams is a student has to be deleted, or the fact that Adams is a manager has to be added. This option is relevant when instances may be of more than one type in the UoD eg Adams is both a student and a manager. Similarly, if the analyst had entered Jones is a student, ie where more than one instance of the same type were entered. The system checks first with the user to confirm whether two in stances of the same type pattern are necessary, before rejecting the pattern.

Example Dialogue. (User is in lowercase,system is in upper case).

> What do you know?
> system lists types and instances.
> Adams is a student
> I ALREADY KNOW THAT
> Jones is a student.
> I HAVE ADAMS IS A STUDENT. DO YOU WANT MORE THAN ONE PATTERN?
> yes (pattern stored).
> no (pattern rejected).
> Adams is a manager
> I HAVE ADAMS IS A STUDENT
> DO YOU WISH TO CHANGE ANYTHING?
> delete Adams is a student. or,
> add Adams is a manager.

The instance-type dictionary and the topic tables are constantly being accessed and updated during type allocation. If the parser encounters input sentences other than the IS_A sentence type, it assumes that a fact and thus a topic table has to be checked. The high-level algorithm for type allocation is as follows.

4.8. Algorithm TYPE_ALLOCATION

Input: A subset N of natural language facts F. Output: A set of updated topic tables TT.

TYPE_ALLOCATION(N)

begin while N do begin

```
parse F;
allocate types and roles;
check types and roles with information in topic tables(s);
If collection of types and roles exists in same syntactic form
  reject pattern
else if collection of types and roles exists in different
  syntactic form then
   verify with user whether same or different topic; store types and roles in relevant topic
table
end;
return(TT)
end.
```

Discussion

Assuming that the following sentence is the first sentence entered for a particular UoD, and ignoring the type/instance queries which were discussed above. (11) Adams is enrolled in a BSc and studied CS112 and scored 7. The parser works from left to right checking each word in the input string and matching these words with the entries in the dictionary and user - supplied in formation. Assuming that all word strings are matched and the parse is successful, the following role-entity type pattern is created: (a) person is enrolled in degree and studied subject and scored rating. If another sentence is entered, such as (12) Jones is enrolled in a BA and studied PD102 and scored 6. This sentence would be rejected by the system , as the role-entity type pattern created (b) person is enrolled in degree and studied subject and scored rating, is identical to the pattern created previously for sentence (11). Similarly, if the analyst had entered a sentence such as (13) Wirth wrote Pascal in 1973, giving the role- entity type pattern (a) person wrote language in location date. The following sentence translating to the same role- entity type pattern as (13)(a), though presented in a different syntactic form, (14) Pascal was written by Wirth in 1973, would be queried by the sytem,allowing the user the option of introducing a new topic, leading to the construction of additional tables, or being simply reminded that the pattern already exists.

If the interpreter encounters an unknown string during parsing, a message is sent to the user requesting further information. For example, if sentence (12) above were entered by the user and Jones had not been entered during stage 1 (instance-type specification),the system would not find a match for Jones in the dictionary. The user then has the option of specifying the sentence in the form described in 4.1 above, namely(18) The student, Jones is enrolled .....etc. This method of knowledge acquisition is not feasible for large UoD's for obvious reasons.

On completion of type allocation, the system presents the role entity type patterns to the user in the forms shown (14)(a)(b) above. If additional information were to be added , for example the fact that student got position 3, the sentence may be entered here. eg (19) Adams got position,3. At this stage the information would be appended to the table under discussion, rather than a new table being created.

Similarly, if a role - entity type pair were to be deleted, the user may delete at this stage. On completion of type allocation, a type table is presented to the user, corresponding to the topic tables.

Example Dialogue

> Adams is enrolled in a BSc and studied CS112 and scored 7.
> OK, or, if string Adams does not exist
> WHO OR WHAT IS ADAMS?
> The student , Adams is enrolled in a BSc....etc
> Jones is enrolled in a BA and studied PD102 and scored 6.
> I ALREADY HAVE THAT PATTERN.
> Adams got position,3.
> OK
> Wirth wrote Pascal in 1973.
> OK
> Pascal was written in 1973 by Wirth.
> IS THIS THE SAME AS PERSON WROTE LANGUAGE IN DATE?
> yes.
> OK, or
> no,
> OK, I'M CONSTRUCTING A NEW TOPIC TABLE.
> YOUR TYPE PATTERNS ARE
STUDENT IS-ENROLLED-IN DEGREE STUDIED SUBJECT SCORED RATING GOT POSITION.
> DO YOU WISH TO CHANGE ANYTHING?
> no, or
> delete got position.
> OK

5. Elementary Fact Type Construction

Elementary fact types are important for the construction of the relational database tables. Incorrect elementary fact type structures can lead to redundancies or information loss in the database. Elementary Fact Type Construction involves two interdependent stages:

(a) determination of uniqueness constraints;

(b) construction of the elementary fact types. Uniqueness constraints in this system differ from the NIAM method, in that they are determined on the basis of the output report, rather than on the conceptual schema. Stages (a) and (b) are almost fully automatic.

5.1. Input to the Elementary Fact Type Construction Module

Type tables corresponding to the topic tables (described above in 4.1) and illustrated again below.

5.2. Output from the Elementary Fact Type Construction Module

Candidates for elementary fact types in the following form:

Bft = Type1,Type2

.

.

Nft = Type1,Type2,..Typen

This system differs from NIAM in that it takes as a starting point binary fact types. That is, potential unary fact types are converted to binary fact types. Most of the fact types expected will be nary. The analyst may change the elementary fact types constructed by the system on completion of this stage. The naming conventions for NIAM elementary fact types are at this stage incomplete. That is, they are not in the form of entity - type role pairs (described in [2]). The system has not yet checked for common entity types (also described in [2]). Derived fact types also have to be determined before the efts are complete for ralational tables to be constructed.

5.3. Function of Elementary Fact Type Construction

The Elementary Fact Type Construction module performs the following tasks:

- automatically determines uniqueness constraints;
- validates these uniqueness constraints (unique columns) by offering the user a counter example;
- constructs a 'first pass' at elementary fact types, based on previously acquired linguistic knowledge (4.2) and the uniqueness constraints;
- validates the elementary fact types with the user and allows any alterations to be made.

Discussion

The analyst is presented with type table(s) corresponding to topic tables constructed in stage 2 of type allocation,for example:

Student Degree Subject Rating

Adams BSc CS112 7

and requested to populate these tables with a small but significant poulation, an example of which would be:

Example 1.

Student	Degree	Subject	Rating
Adams	BSc	CS112	7
Adams	BSc	CS100	6
Adams	BSc	PD102	5
Brown	BA	CS112	7
Brown	BA	PD102	6
Collins	BSc	CS100	5

Based on the data supplied, the system then automatically determines the unique (composite) columns by searching each individual column, pairs of columns, triples of columns in all possible combinations till the number of columns in the table has been covered. In the above example, only student-subject are unique. When a unique (composite) column is encountered, a counter example is offered to enable the user to validate the correctness of this. Counter examples are constructed by taking the value from the first row of the suspected unique column(s) and joining with a non equal value from a neighbouring column. For example, in the above table, the student-subject column is unique. The system constructs the string <Adams BA CS112>, and asks the user to verify whether such a row is possible. In other words, can the student-subject values be repeated. If the response is no, then the column type is marked with an integer value (1..number of columns in the table), in the unique field of the topic table. The integer values help to determine the elementary fact types. Although the size of the type table(s) to be searched is not large (they are only a subset of a universal relation), corresponding to a complex English sentence (we can expect possibly ten type columns) , the complexity of the search is reduced by the following assumptions.

1. The columns of the table are ordered with subject as first argument,direct object(s) as second and third arguments, and modifying prepositional objects in this case as subsequent arguments working from left to right.

2. The table is such that the subject is the topic of the sentence. All subsequent types are attributes of the subject, or stand in some semantic relationship to the subject. Otherwise a different sentence pattern would have been chosen in 2.1.. Based on these assumptions we state the following.

Observation 1. If a table T = [S,O1,O2,O3,....On] and the uniqueness constraint covers the

subject column only, then the table consists of binary fact types of the form Eft = [<S,O1>,<S,O2>,<S,O3>,....<S,On>].

Proof. Project on the unique column in the above combinations. As the column projected on is unique, the cardinality of the tables created is the same as the original table. Because the object columns are functionally dependent on the unique subject, no new rows (ie no new information) are created after joining.

Observation 2. If a table T = [S,O1,O2,O3,...On] and a direct object is covered by a uniqueness constraint (in the absence of a unique grammatical subject),then elementary fact types are binary and are constructed by combining the first unique direct object column with each other column giving Eft = [<UniqueColumn,S>,<UniqueColumn,O2>,..<UniqueColumn,On>].

Note. Types marked as prepositional objects in the topic tables are ignored as potential uniqueness constraints, as these are semantically only adjunct arguments, not belonging to the semantic core of the sentence.

Proof. Project on the unique column in the above combinations. As the column projected on is unique, the cardinality of the tables created is the same as the original table. Because the subject and/or object columns are functionally dependent on the unique object , no new rows are created after joining.

In the case where no single subject or direct object column is found to be unique the second stage of the search is to pick up composite unique columns (where uniqueness constraint covers 2 to n-1 columns) and determine columns that are functionally dependent on the unique columns.

Observation 3. If a table T = [S,O1,O2,O3,..On] and the uniqueness constraint covers the subject (or the first direct object) column, then the elementary fact types are either the original unique columns + each other column that overlaps in uniqueness with the first column, or a reduced column + each remaining unmarked column.

Proof. Project on the unique columns, combined with other columns as stated above. By projecting on the unique columns,the cardinality of the split tables remains the same as the original. Each other column is a function of the (reduced) unique column. Therefore on joining no new rows are created.

Example 2.

Given the following user-populated type table:

Subject	Credit Points	Semester	Enrolment	Lecturer
CS100	8	1	500	PP
CS102	8	2	500	GR
CS112	8	1	300	TH
CS113	8	2	270	TH
CS380	16	1	50	PB
CS380	16	2	45	AL

Eft1 = Subject,Semester,Enrolment (original unique columns + FDs).
Eft2 = Subject,CreditPoints (reduced unique columns + FDs)
Eft3 = Subject,Lecturer " "

Note. The method illustrated above, does not take into account overlapping uniqueness constraints which leads to the problem with B.C.N.F.

In cases where the uniqueness constraint covers a whole row of the table,the following observation applies.

Observation 4. If a table T = [S,O1,O2,O3,...On] and the uniqueness constraint covers the whole row of the table, then the elementary fact types are either (a) binary, as in Eft = [<S,O1>,<S,O2>,..<S,On>]. In cases where for each unique subject column value, the product of the unique types in the remaining columns = the number of rows for each entry in the table; or, (b) nary. That is, Eft = [S,O1,O2,O3,...On].

Example 3.

Person	Skill	Language
Smith	cook	English
Smith	cook	Greek
Smith	cook	French
Smith	type	English
Smith	type	Greek
Smith	type	French
Jones	type	English

Example 4.

Person	Vehicle	Company
Smith	car	GMH
Smith	truck	GMH
Smith	4WD	Toyota
Smith	car	Toyota
Jones	4WD	GMH
Jones	car	GMH

In Example 3 above,

Eft1 = Person,Skill
Eft2 = Person,Language.

In Example 4 above,

Eft = Person,Vehicle, Company.

6. Conclusion

This concludes a description of the first stage in a relational database design tool. The method described is simple, easy to follow, and requires reasonable interaction with the user. The remaining steps in the process ie naming conventions and the determination of derived fact types will be presented in another paper. The theoretical aspects of the remaining modules have been developed. This phase of the system is currently under implementation.

References

[1] Chen,P.(1976)."The Entity Relationship Model: Towards a Unified View of Data",ACM TODS 1,No.1.

[2] Nijssen,G.M. and Halpin,T.J."Conceptual Schema and Relational Database Design: A fact oriented approach". Prentice Hall of Australia,Pty,Ltd.

[3] Maciaszek,L.A.(1987)."Conceptual Database Design Methodology and Tools".Technical Report,No.87/10,Dept. of Computing Science,University of Wollongong,NSW,Australia.

[4] Alshawi,H.(1987)."Memory and Context for Language Interpretation".Cambridge University Press,London.

[5] Hirst,G.(1988)."Semantic Interpretation and the Resolution of Ambiguity".Cambridge University Press,London.

[6] Wilensky,R.(1981)."PAM(Plan Applier Module)". In Inside Computer Understanding, Schank and Riesbeck,eds.

[7] Appelt,D.E.(1985)."Planning English Sentences".Cambridge University Press,Cambridge,New York.

[8] Simmons,R.F.(1973)."Semantic Networks:Their Computation and Use for Understanding English Sentences". In Computer Models of Thought and Language, Schank and Colby,eds.

[9] Charniak,E.(1981)."The Case-Slot Identity Theory".In Cognitive Science,No.7.

[10] Charniak,E. and Minsky,M.(1975)."A Framework for Frame-based Systems".

[11] Schank,R. and Abelson,R.(1977)."Scripts,Plans,Goals and Understanding". Lawrence Erlbaum Assoc.,Hillsdale,New Jersey.

[12] Schank,R.(1982)."Reminding and Memory Organization". In Strategies for Natural Language Processing. Lehnert and Ringle,eds.

[13] Rumelhart,D.E. and McClelland,J.L.(1985)."On Learning the Past Tenses of English Verbs".In Parallel Distributed Processing.Vol. 2. Rummelhart,ed.

[14] Haas,N. and Hendrix,G.(1983)."Learning By Being Told". In Machine Learning, Michalski et al eds.

[15] Tomita,M.(1985)."Efficient Parsing for Natural Language".Kluwer Academic Publishers.

[16] Dunn,L.(unpublished manuscript)."An Interactive Parser for Conceptual Schema Disambiguation".U.Qld.

[17] Dunn,L.(unpublished manuscript)."Designing Relational Databases with Natural Language".U.Qld.