MODELLING AND MANIPULATING PRODUCTION DATA BASES IN TERMS OF SEMANTIC NETS

R. Meersman, F. Van Assche

International Center for Information Analysis Services
CONTROL DATA BELGIUM INC.
Raketstraat 50, 1130 Brussels Belgium.

Abstract

In this paper we show, how semantic nets when interpreted as so-called conceptual schemata can be successfully made to serve as common denominator to two as yet disparate areas of artificial intelligence and real-life data management: Knowledge Representation and Data Base design.

First, we describe the requirements to Semantic Nets in the context of production data bases. Next we describe the automated mapping between the Semantic Net and (possibly several) DBMSs.

Further we show how this formal and machine accessible mapping allows to execute formally structured English Queries, and formulate expert rules in a data base context.
Implementation status and research directions for these ideas are given resp. suggested.

I. INTRODUCTION

In this paper we intend to discuss the practicability of interpreting certain types of semantic nets as conceptual schemata for a class of real-life production (possibly very large) databases. This has the double advantage of introducing more semantics and integrity into the database design as well as providing support for, say, an expert system by making the "contents" of the net (i.e. the database "under" the net) available through a very high level user interface.

The tools for making this happen mainly involve an intelligent(i.e. semantics preserving) mapping between the semantic net and the underlying database structure, and a user-friendly interpreter/compiler for this mapping.

II. SEMANTIC NETS AS A MODELLING METHOD FOR PRODUCTION DATA BASES

As other semantic nets (1) (3) (6) a Database Semantic Net (DB-SN) comprises two kinds of associations:

1. The taxonomic "is-a" association, e.g. "a BOMBER is a MILITARY-AIRCRAFT" (see figure 1).
2. Different fact or property associations, e.g. "An AIRLINE-COMPANY is flying a COMMERCIAL-AIRCRAFT".

This association can also be read in the other direction:

"a COMMERCIAL-AIRCRAFT is flown-by an AIRLINE-COMPANY".

Furthermore, the same expression "flown-by" can be used to construct a reference to a (set of) object(s):

"the COMMERCIAL-AIRCRAFT flown-by AIRLINE-COMPANY XYZ".

As far as individual occurrences are concerned, it becomes inappropriate to represent them in the semantic net when dealing with Very Large Data Bases (VLDB) in a production environment, and this for two reasons:

1. In a production-DB there are in general too many individual occurrences;
2. Production DBS are dynamic: individual occurrences come and go with a time scale considerably superior to that of changes in the net structure.

Therefore a DB-SN contains only object types of which the individual occurrences are instances. As a consequence, the DB-SN contains only fact types, of which facts are instances.

In figure 1 these object types are depicted as circles. Dashed circles for the lexical object types (having a listable representation) e.g. CONSTRUCTOR-NAME, and solid circles for the non-lexical object types, e.g. CONSTRUCTOR. Non-lexical occurrences do not have themselves an external representation but can be referred to by other lexical occurrences. (In general a reference tree is spanning one or more lexical occurrences).

For some object types the distinction is not made in natural language, e.g. MAXIMUM-SPEED, and thus we depict them as two concentric circles. The object types connected with a is-a link to a "larger" type are called subtypes.

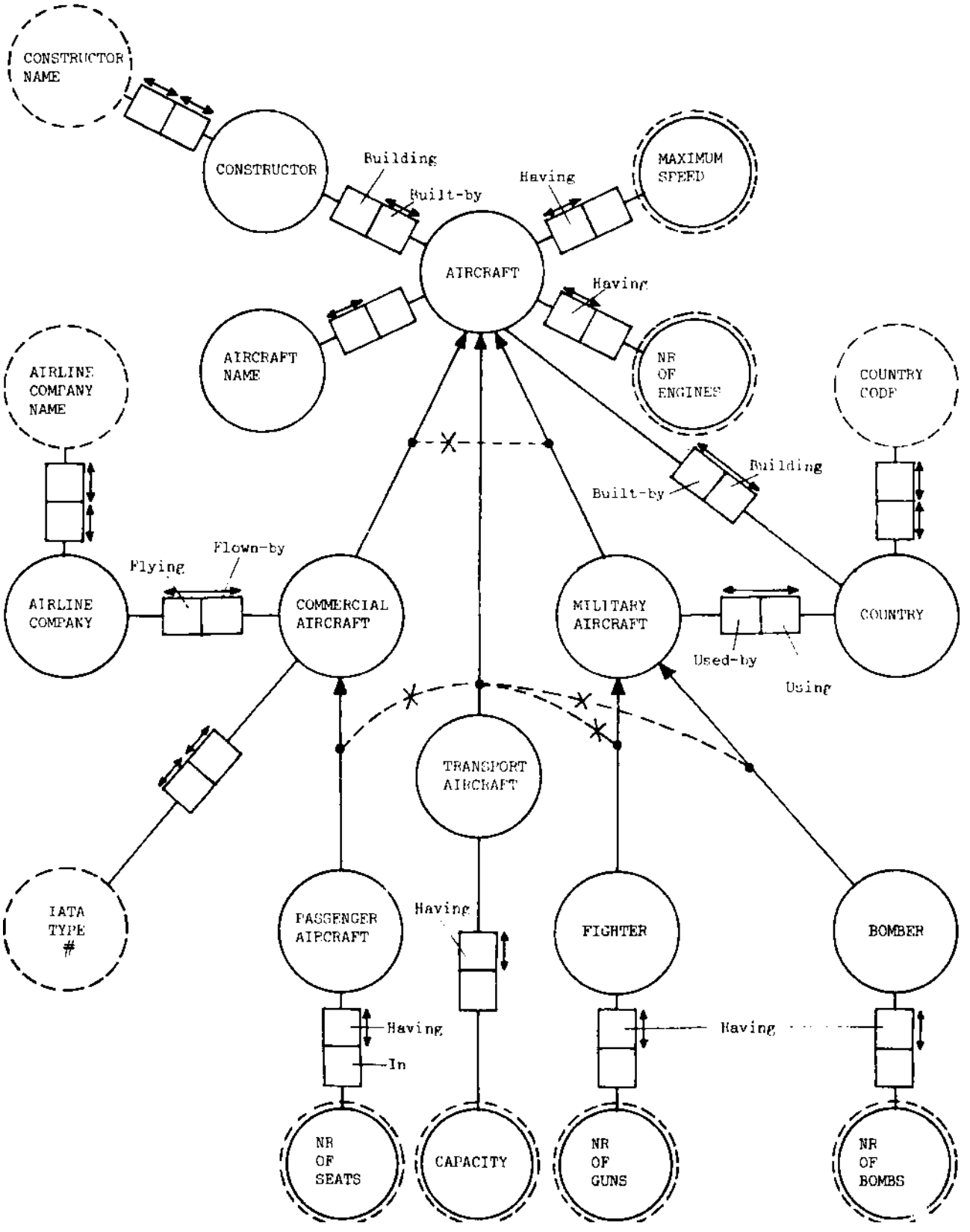

Figure 1: Example of a Semantic Net

To populate a VLDB means: to qualify object instances as belonging to an object (sub)type and to add fact instances.

Obviously, given such freedom to alter the population of the database on the spot, it becomes essential to provide integrity rules, in order to avoid pollution. An example of such an integrity rule might be that the aircraft is built-by only one constructor (it is depicted in figure 1 by a double headed arrow). Note this is a functional dependency.

Another integrity rule might be that commercial aircraft is disjoint from military aircraft. The DB-SN together with its integrity rules constitute a so-called conceptual schema of an Information System.

It is the closeness of these Semantic Nets to natural language concepts that will allow the specification of (integrity) rules in, e.g. Structured English. Examples of simple integrity rule declarations in the (formal) RIDL language (4) (see also below) are given in figure 2. Words underlined are RIDL keywords.

```
enforce AIRCRAFT is built-by
                    only one CONSTRUCTOR
enforce COMMERCIAL-AIRCRAFT
                    is disjoint from
        MILITARY-AIRCRAFT
enforce AIRCRAFT having MAXIMUM-SPEED
                    is included in
        AIRCRAFT having AIRCRAFT-NAME
enforce COUNTRY is using
                    only MILITARY-AIRCRAFT
                    built-by same COUNTRY
```

Figure 2 : Example Integrity Rules

Such integrity rules are getting increasing attention in the modelling of data bases. The recent ISO report on Conceptual Schemata (9) states that all of the integrity rules should appear in the Conceptual Schema. Hence this schema must be a sufficiently powerful tool for specifying them, and the database manager (the schema interpreter) sufficiently powerful to enforce them. It must be emphasised here that the ISO report uses the Conceptual Schema in a much "richer" sense than is done in classical database theory.

III. THE AUTOMATED MAPPING TO A DBMS ENVIRONMENT

For reasons of storage and manipulation efficiency, the DB-SN and its rules are translated to a data base schema. This data base schema is expressed in 5th normal form relations and its integrity rules.

The automated mapping to such a data base schema consists of three parts:

1. Analysis of the semantic net
2. Reference type evaluation and generation
3. Translation of the DB-SN to 5NF relations

A. Analysis of the semantic net

One may view the analyser as a practical implementation of a theorem prover. It detects redundancies, implications and inconsistencies by building inferences from the semantic net and a subclass of its integrity rules.

We can take the subtype structure as a very simple example. If we look at the semantic net as a graph, we see that the subtype links constitute a subgraph which in itself is a set of connected directed graphs. The analyser is programmed with the following rules about these directed graphs:

1. The graph must contain no circuits.
2. All arcs are distinct (have different vertices)
3. The graph must contain exactly one vertex (the root) with zero incoming arcs ("indegree" zero).

One task of the analyser is then of course, to refuse subtype structures violating these rules. Inconsistencies may also arise between a subtype structure and integrity rules: in figure 1, it is obvious that adding any new object type which would be a subtype of both PASSENGER-AIRCRAFT and FIGHTER would result in an inconsistent subgraph.

An example of a redundancy - that is detected by the analyser - is to state that a BOMBER is an AIRCRAFT. It is implied by the statement that a BOMBER is a MILITARY-AIRCRAFT and that a MILITARY-AIRCRAFT is an AIRCRAFT, since a subtype link is a transitive association.

Similar redundancies and inconsistencies are detected when analysing the fact types and the integrity rules upon them.

B. Referenceability analyser and reference type generator

Since it is the intention to process the information by a machine (a DBMS) it is a requirement that every object is referenceable by machine processable (i.e. lexical) objects.

This is accomplished by constructing a reference type for every non-lexical object type. The discussion of valid reference types is outside the scope of this paper and can be found in (8). In short, a reference type can be seen as a directed graph with no loops, exactly one node with indegree zero (the NOLOT to be referenced) and all lexical nodes must be with outdegree zero. The construction of the directed graph is subject to certain validity conditions, since not all facttypes and subtype links are eligible.

As an example IATA-TYPE-NR is a good reference type for a COMMERCIAL-AIRCRAFT, while the AIRLINE-COMPANY is not, since it is not unique for a COMMERCIAL-AIRCRAFT.
However, the IATA-TYPE-NR is not a good reference type for AIRCRAFT since only some of them (the commercial aircraft) have such a number. On the other hand all COMMERCIAL-AIRCRAFT have an AIRCRAFT-NAME inherited from "being-an AIRCRAFT".

The reference type generator tries to construct a reference type for every non-lexical object type using an AND/OR graph search. This is done according to some optimality criteria, following some tie breaking rules, and taking into account the above mentioned validity conditions. When it fails in doing so, the DB-SN is said to be non referenceable and cannot be mapped to a relational data base schema without additional assumptions.

The above might seem a trivial task on a semantic net as depicted in figure 1. However, it is easy to see that complexity increases exponentially with the size of the semantic net.

C. Translation of the DB-SN to 5NF relations

The translation to 5NF relations or record types involves four major phases :

1. The translation of the non-lexical level to the lexical level, making use of the identified reference types.
2. The translation of subtypes. Subtypes do not exist at the DB schema level. They are translated into integrity rules.
3. The grouping of the binary associations into N-ary relations.
4. The translation of the integrity rules. Integrity rules expressed on non-lexical facts must be translated to integrity rules expressed on pure lexical attributes or combinations thereof. Some integrity rules will take a completely different form when expressed in a DB schema. For instance, the third integrity rule expressed in figure 2 will be translated as a null value permission (see e.g. Date [2] on the attribute MAXIMUM-SPEED in the generated relation "AIRCRAFT".

IV. MANIPULATING PRODUCTION DATA BASES IN TERMS OF A SEMANTIC NET DESCRIPTION

As said before, our research is focused on real-life production environments where very large and/or complicated databases are the rule. Such databases are accessed typically through large sets of application programs of greatly varying type. Most often, these will be written in a suitable implementation language such as COBOL, PASCAL, ADA, etc. and manipulate the database directly in its record format: these languages provide natural representation for record-like structures.

Of course, if the database was designed and implemented using techniques and generators as discussed in section III, it then becomes very desirable to possess a very high level language in which one is able to describe and manipulate the information in terms of the defining semantic net. This language can then be interpreted on the target database machine resulting from the automated mapping, or be the source input for generators on this machine.

For the DB-SN described above, such a language has been defined. It is called RIDL for "Reference and Idea Language" and its compiler is able to interpret the information produced by the automated mapping in order to translate back and forth between semantic net concepts and the record types of the generated database.

To show how the RIDL syntax allows to manipulate knowledge at a higher level than record- or relation-oriented syntaxes, we refer to (4,5) and (10). Also, the examples below should give an indication of some of its alleged power.

On the one hand, the language has a purely functional component which centers around information referencing: one always specifies which information is needed at a given point without having at all to provide the procedure that tells how to obtain it from the database. On the other hand, a complete procedural component is also present in RIDL in order to allow people to handle more easily "natural" procedural aspects (e.g. computations, control structure), as well as supporting, e.g. system design.

As an example, consider the RIDL query (words underligned are keywords of the language):

```
list COUNTRY-NAME of COUNTRY using
AIRCRAFT not
built-by same COUNTRY
```

of which the meaning should be self-evident from the semantic net. However, to build an application program that executes this at the generated record-level is in general not a trivial task (even in the case above it is not difficult, but tedious) and certainly involves specific programmer effort. Even in RIDL, the above query can be equivalently formulated as:

```
for each C in COUNTRY:

  if MILITARY-AIRCRAFT used-by C
     is not included in
        AIRCRAFT built by C
  then
     list COUNTRY-NAME of C
  end if
end for
```

which illustrates some of its "procedural" aspects. Other such constructions are full support of procedures and functions (with object-oriented parameter passing), set operations and the capability to define macros.

Note that the subject of the list statement above is a type of lexical objects. This is indeed a requirement: only the occurrences in the (lexical) "leafs" of the semantic net have listable representations.

V. STATUS OF IMPLEMENTATION AND FUTURE DIRECTIONS

The tools described in section III are implemented and released under the name Information Analysis Support Tools (IAST) (7).

At the time of writing, it is being used in about ten production projects. One of them has a semantic net of + 380 object types + 700 fact types and subtype links and + 1460 Tntegrity rules: about 50 times larger than the semantic net of figure 1.

It is our experience that the system does a more thorough analysis and a better data modelling than experienced information analysts. This is further compounded by the near-impossibility of manual analysis, mapping and maintenance for very large semantic nets.

A RIDL compiler is currently under construction on a few machines. The most promising architecture seems to be a two-pass interpreter where the first pass generates pseudo code given to a pure "semantic-net interpreter", and this one generates record- or relational-oriented database manipulation code which is as much as possible DBMS-independent. This design delegates DBMS and machine-dependence to the final interpreter.

A full RIDL query compiler according to this architecture has been built and is undergoing testing.

Future fundamental developments include the specification and enforcement of expert "production" rules in RIDL and the extension of the theorem proving functions of IAST to include some of these expert rules and a still wider class of integrity rules.

An example of such rule on the DB-SN of figure 1 might be:

```
rule R01

when add "COUNTRY C is building
     MILITARY AIRCRAFT A"
do
delete (COUNTRY minus C) is using A
end rule
```

Note how this rule would implement a (possible) enforcement of the last integrity rule of figure 2.

REFERENCES

ABRIAL J.R., Data Semantics. In J.W. Klimbie and K.L. Koffeman (Eds), Data Base Management Systems, Elsevier North-Holland, New York 1974.

2 DATE C.J., An Introduction to Database Systems. Third edition. Addison Wesley Publishing Company, 1981.

3 DELIYANNI A., KOWALSKI R.A., Logic and Semantic Networks.
CACM, Volume 22, Number 3, March 1979.

4 MEERSMAN R., The RIDL Coneptual Language, ICIAS Report (to appear, 1983).

5 MEERSMAN R., The High Level End-User. In Data Base: The 2nd Generation. Infotech State of the Art Report, Pergamon Press 1982.

6 SCRAGG G, Semantic Nets as Memory Models. In Charniak E. and Wilks Y. Computational Semantics, (1976), p. 101-127.

7 VAN ASSCHE F., ACKERMANS N., GHYS C. IAST: A Data Management Development System (to appear).

8 VAN ASSCHE F., SIMONS D., VANHOEDENAGHE M. The Automated Mapping from a Binary Conceptual Schema to a 5th NF Data Base Schema (to appear).

9 VAN GRIETHUYSEN J.J. (ed). Concepts and Terminology for the Conceptual Schema and the Information Base.
ISO Report, publication number ISO/TC97/SC5-N695, 1982.

10 VERHEYEN G.M.A. and V. BEKKUM J. NIAM: An Information Analysis Method.
In: Information Systems Design Methodologies: A Comparative Review, Proceedings of the IFIP TC8 Working Conference, Noordwijkerhout, *The* Netherlands, May 1982.
North-Holland Publishing Company.