acm
ASSOCIATION FOR COMPUTING MACHINERY

1976 SIGMOD

INTERNATIONAL CONFERENCE ON MANAGEMENT of DATA

PROCEEDINGS

WASHINGTON, D.C. JUNE 2 – 4

EDITED by JAMES B. ROTHNIE

NUL : A NAVIGATIONAL USER'S LANGUAGE FOR A NETWORK STRUCTURED DATA BASE★

Claude Deheneffe and Henri Hennebert
Institut d'Informatique
5000 Namur, BELGIUM

ABSTRACT

This paper presents an end-user's language which tries to solve the problem of an easy navigation through a DBTG-like data base structure. A request is expressed in a nonprocedural and hierarchically structured fashion. The dialogue is split into two main parts : first a data context definition, then the manipulations of this context.

A context is a part of the data base that the user is concerned with. A context definition is formed by a set of labelled lines; each line is a condition declaration on one entity-set. By means of labels and link names declared in the data structure, a line may be connected to another one; this expresses a 'join', by the named link, between the two entity-sets involved in the two lines.

The originality of the language lies in the fact that it permits the user to navigate easily and fairly naturally from one entity-set to another through a link; in fact, this navigation is mapped into a hierarchical structure which appears more comprehensible to the user.

On the other hand, a manipulation is a command such as print, update, insert or other standard actions the user may want to execute on the context.

KEY WORDS

Data Base Management Systems, Entity-Relationship Model, Network Model, Data Manipulation Language, Query Language, Navigation, Casual User.

I. INTRODUCTION

"A network environment is one of the most general structures used to represent relationship among data". It has been proposed as a basic tool for the architecture of Data Management Systems [1]. The DBGT proposals are concerned with this approach, but other high-level data models have been proposed [2,3,4,5] based on the concepts of entities and relationships.

An entity is an object that exists in our mind composed of a list of attribute values, entities being classified into different entity-sets. A relationship-set is a mathematical relation among several entites; we restrict ourselves to binary relationships; a set of such relationships is called a link. In all generality, a link may be a relation that is one-to-many or many-to-many. Moreover, a link is "strong" if the "target" entity exists if and only if, it is linked to the "origin" entity, otherwise the link is "weak" [5] .

It is supposed that each entity-set has a primary key composed of a group of one or more attributes, possibly inherited from another entity-set through a one-to-many strong link. This method of identifying entities using attributes from other entities can be applied recursively. A relationship can be identified by the entities involved; consequently the primary key of a relationship may be represented by the primary keys of the other entities concerned.

The conceptual structure described using this model requires an appropriate language which allows the user an easy navigation through the data; the user is not at all concerned with a one-entity-at-a-time selection but only interested in giving properties of entities required or using retrieval paths by means of links.

NUL is intended to provide the user with a restrictive set of building blocks for constructing requests in order to traverse the data base. The designation of entities has been separated from the specification of actions on the entities and links. The syntax has been conceived in such a way to allow a top-down structured approach. So it is claimed that NUL is an appropriate tool for a non-programmer user.

The basic facilities of NUL will be illustrated in the following sections by examples based on the data base described in fig. 1.

II. THE DATA-CONTEXT DEFINITION

In the first part of a user's request, NUL allows the user to specify a structured subset of the data base, providing tools similar to the thought processes that he would use to locate information if the data structure was placed in front of him. From a human point of view, it often appears useful to allow the user to decompose a request into simple statements; this avoids intricate

★ Within the framework of the national program for the advancement of research in computer science, sponsored by the Commission Interministérielle de la Politique Scientifique.

nested expressions which are undesirable for the non-specialist user. NUL permits this decomposition assigning each statement a label denoting a temporary set of occurrences for the entity-set concerned.

This context definition is entered interactively under the control of a conversational monitor which asks, step by step, which entities are to be included into the context and tags them.

An entity-set may be tagged by means of two main mechanisms : first a boolean expression of criteria on its attributes and on the existence of connected entities meeting other criteria; secondly its relation with an already restricted entity-set through a declared link.

The user is thus allowed to express his navigation through the data, writing down his request in a way that appears rather natural and easy.

Qualifications on properties of entities

A user may want to focus on a subset of entities satisfying certain criteria on its attribute values. This may be illustrated for example by Q1

Q1 : All the persons who earn less than 12,000 Dollars and live in Namur or Liege.

S1 ← PERSON SUCH THAT SALARY < 12000
AND TOWN = 'NAMUR', 'LIEGE'.

The look-up of occurrences in the PERSON entity-set is made by marking the occurrences of PERSON where SALARY value is less than 12,000 dollars and TOWN is 'NAMUR' or 'LIEGE'. This subset is denoted S1. A qualification acts as a restriction on an entity-set.

A qualification is introduced by a 'SUCH THAT' clause and applies to the qualified entity-set a boolean expression of criteria. Each criterion is the comparison of an attribute with a value or list of values by means of one of the usual relational operators =, <, >, <=, >=, ... The priority of the boolean operators AND, OR can be changed by the use of parentheses.

Note that the label is given by NUL, and permits the user, subsequently to reference the corresponding set.

Existential qualification on connected entities

Suppose two entity-sets A and B, and L a link connecting A to B, it appears desirable to allow the user to tag occurrences of A provided that each is connected to an occurrence of B. Q2, Q3 and Q4 illustrate this use of existential qualification.

Q2 : All the persons who live in Namur and have a skill.

ST ← PERSON ST TOWN = 'NAMUR' AND HAVE SKILL.

An occurrence of the PERSON entity-set is marked if its TOWN is 'NAMUR' and if it is connected to at least one occurrence of the SKILL entity-set. This set definition may be expressed in the predicate calculus in the following way :

$$x \in S1 \Leftrightarrow x \in \mathrm{PERSON} \wedge x.\mathrm{TOWN} = \text{'NAMUR'} \wedge$$
$$(\exists\, y : y \in \mathrm{SKILL} \wedge [x,y] \in \text{SKILL-OF-PERSON})$$

Moreover, NUL permits the expression of criteria on the connected entity by means of a QUALIFICATION FLAG (QF). The introduction of this last concept avoids nested criteria in a statement. It notifies the system that the entity-set specified in an existential qualification has to meet some conditions not yet stated. Consequently, NUL asks the user to express qualifications on this last entity-set.

The use of a QUALIFICATION FLAG is shown in the following example :

Q3 : All the persons who live in Namur and have any skill in the French language.

S1 ← PERSON ST TOWN = 'NAMUR' AND HAVE SKILL = QF1.

QF1 ← *SKILL ST* CODE = 'FRENCH'.

By means of the Qualification Flag QF1, the user notifies the system that he wants to qualify the person's skill. By displaying '*QF1 ← SKILL ST*' NUL asks him to introduce the corresponding qualification. The predicate calculus expression of Q3 is :

$$x \in S1 \Leftrightarrow x \in \mathrm{PERSON} \wedge x.\mathrm{TOWN} = \text{'NAMUR'} \wedge$$
$$(\exists\, y : y \in \mathrm{SKILL} \wedge [x,y] \in \text{SKILL-OF-PERSON} \wedge y.\mathrm{CODE} = \text{'FRENCH'})$$

Several Qualification Flags (QF) may be used in one statement and in the expansion of another QF. In the last examples, Q2 and Q3, the name of the link is not used explicitly in the 'HAVE' clause. However, in case of ambiguity, ie. when several links have been declared between two entity-sets, the link name must be specified in the following manner :

⟨link-name⟩ HAVE ⟨ entity-set-name⟩ = QFi

Moreover, in order to provide the user with an easy and flexible syntax, alternative key words may be used in place of HAVE.

In the foregoing, we have discussed various points concerning qualification expressions, but we have omitted the explicit use of quantifiers in the 'HAVE' clause. In fact, the existential quantifier SOME was assumed. NUL also permits flexible use of the following quantifiers :

EXACTLY ⟨ integer⟩

AT LEAST⟨integer⟩

AT MOST ⟨ integer⟩

NO

SOME (= AT LEAST 1) may be omitted

ALL

The following is a more complete example :

Q4 : All the departments which have 3 employees with all skill levels greater than 3 and a manager who was born after 1932 and has a skill level of 5.

```
S1  ← DEPARTMENT ST EMPLOYEE ARE AT LEAST
      3 PERSON = QF1 AND MANAGER IS PERSON = QF2.
QF1 ← PERSON ST HAVE ALL SKILL = QF3.
QF3 ← SKILL ST LEVEL > 3.
QF2 ← PERSON ST BIRTHDATE > 1932
                AND HAVE SKILL = QF4.
QF4 ← SKILL ST LEVEL = 5.
```

Note that all '*QFi ... ST*' are displayed by NUL to request expansion of the Qualification Flags.

Projection of a qualified entity through a link

Up to now, the 'SUCH THAT' and 'HAVE' clauses allow the specification of homogeneous occurrences-set. Such a set may be used as starting point for navigating along the links defined in the data base structure. This may be illustrated by Q5.

Q5 : All the departments located in Namur and all the employees of these who were born after 1935.

```
S1 ← DEPARTMENT ST LOCATION = 'NAMUR'.
S2 ← FOR S1 BY EMPLOYEE PERSON ST BIRTHDATE > 1935.
```

The set S1 of all the departments located in Namur is projected through the link EMPLOYEE into the occurrences-set of the PERSON entity-set for which BIRTHDATE is greater than 1935; the result of this operation is the set S2. This may be expressed in the predicate calculus as follows :

$$x \in S1 \Leftrightarrow x \in \text{DEPARTMENT} \wedge x.\text{LOCATION} = \text{'NAMUR'}$$

$$y \in S2 \Leftrightarrow \exists\, x : x \in S1 \wedge [x,y] \in \text{EMPLOYEE} \wedge y \in \text{PERSON} \wedge y.\text{BIRTHDATE} > 1935$$

The data-context definition Q5 differs from the following one :

Q'5 : All the employees who work in Namur and were born after 1935.

```
S1  ← PERSON ST BIRTHDATE > 1935
              AND EMPLOYEE IN DEPARTMENT = QF1.
QF1 ← DEPARTMENT ST LOCATION = 'NAMUR'.
```

In fact, Q5 defines two sets S1 and S2, whereas Q'5 defines only one set S'1 identical to S2.

NUL keeps track of the occurrences of the named link between the two sets S1 and S2. These may be used subsequently in the manipulations on this context. Just as in the qualification on connected entities, the link-name may be omitted in the 'FOR' clause. For instance, if the skills of the persons S2 are to be included in Q5, the following statement is added :

```
S3 ← FOR S2 SKILL.
```

General structure of a data-context definition

A context definition is composed of a set of labelled statements where each label denotes a temporary occurrences-set. As seen in the 'FOR' clause,each statement may be associated with a preceding one by using its label. In this way, statements form a set of hierarchies The roots are the statements that are not introduced by a 'FOR' clause; the first statement is necessarily a root. The leaves are the statements whose labels are not used in a 'FOR' clause. The hierarchies apply in the same way to the corresponding occurrences-sets. Thus, the navigation is mapped into a set of hierarchies that are more natural to the user than the original data base network.

The syntax of the language permits a top-down structured definition, treating one entity-set at a time even when existential qualification on connected entities are used.

The context definition is initialized by NUL printing out the key word '*CONTEXT ?*' and closed when the 'END' statement is given by the user. Q6 gives a more complete example.

Q6 : All the departments which are located in Namur, the employees working in these departments who earn more than 12,000 dollars and have at least one skill level of 3, the skills of these employees and the managers of the former departments; the department number 9012 and its employees.

```
CONTEXT ?
S1  ← DEPARTMENT ST LOCATION = 'NAMUR'.
S2  ← FOR S1 BY EMPLOYEE PERSON ST SALARY > 12000
              AND HAVE SOME SKILL = QF1.
QF1 ← SKILL ST LEVEL = 3.
S3  ← FOR S2 SKILL.
S4  ← FOR S1 BY MANAGER PERSON.
S5  ← DEPARTMENT ST NUMBER = 9012.
S6  ← FOR S5 BY EMPLOYEE PERSON.
S7  ← END.
```

This data-context corresponds to the occurrences-sets of fig. 2

III. MANIPULATIONS ON A CONTEXT

NUL provides the user with a list of commands to use on a previously defined data-context; these facilities include report generation, insertion, deletion and update of entities or links.

Display of information

A complete report may be described by means of the 'PRINT' command which specifies the attributes whose values are to be displayed. Each attribute name is prefixed by a label associated to its entity-set. The report involves one hierarchy of the context. Any label may be selected as the root of the report. The structure of the report will be defined from this label downwards. Any label of the context-hierarchy may be introduced into the report. They are structured hierarchically and must be compatible with the original context-structure. Moreover, labels that are higher in the hierarchy than the root of the report may be concatenated to it, and their other dependent labels are then considered as subordinated to this concatenated root. The general structure of the report is reflected using parentheses in the PRINT command.

```
Q7 :
CONTEXT ?
S1 ← DEPARTMENT ST ...
S2 ← FOR S1 BY MANAGER PERSON ST ...
S3 ← FOR S1 BY EMPLOYEE PERSON ST ...
S4 ← FOR S3 SKILL ST ...
S5 ← FOR S3 CHILDREN ST ...
S6 ← END.
COMMAND ?
PRINT S1.NUMBER, (S3.NAME, S3.SALARY,
                 (S4.CODE, S5.CHRISTIAN-NAME)).
```

This 'PRINT' structure is merely a subset of the original structure; the concatenation mechanism is illustrated by Q'7.

```
Q'7 : PRINT  S3.NAME, S3.SALARY,
             (S5.CHRISTIAN-NAME, S5.BIRTHDATE),
             S1.NUMBER, (S2.NAME).
```

These two examples are illustrated in fig. 3. In Q'7, note the concatenation of S3 (the root of the report) and S1 (the root of the context-hierarchy).

Q'7 will give the following results :

```
S3.PERSON
   NAME   =  JOHNSON
   SALARY =  10000
     S5.CHILDREN
        CHRISTIAN-NAME = BILL
        BIRTHDATE      = 1953

        CHRISTIAN-NAME = KATE
        BIRTHDATE      = 1956
S1.DEPARMENT
   NUMBER = 9012
     S2.PERSON
        NAME = SMITH
S3.PERSON
              ⋮
```

If all the attributes of an entity-set are to be printed out, only the corresponding label is required in the argument of the PRINT. For example, if all the attributes of the departments and of the employees are to be printed out but only the skill codes, the following can be used :

```
PRINT S1, (S3, (S4.CODE)).
```

Insertion of an entity

After an'INSERT' command has been issued, NUL asks for the attribute values of the new entity and the primary key of the entities to which it is connected by strong links. During these operations, the data-context is not necessary, it merely provides a means of knowing the values of the primary keys.

Q8 : Give a new skill to Johnson.

```
CONTEXT ?
S1 ← PERSON ST NAME = 'JOHNSON'.
S2 ← END.
COMMAND ?
PRINT S1.NUMBER, S1.BIRTHDATE.
S1.PERSON
   NUMBER    = 50324
   BIRTHDATE = 1936

   NUMBER    = 73432
   BIRTHDATE = 1926
COMMAND ?
INSERT SKILL.
SKILL.CODE  =       complete the form and
SKILL.LEVEL =       send it back to NUL.
PERSON.NUMBER =     give the primary key
                    of the person.
COMMAND ?
```

Update of entities

The 'UPDATE' command works on a list of attributes prefixed by labels declared in the data-context. This list of attributes must belong to one hierarchy only of the data-context.

The update process is guided by NUL which displays the values of the attributes, following a dynastic order through the occurrences-hierarchies and skipping all nodes not included in the update-list. At each step, the user can modify the displayed values and return them to NUL for update. During this navigation, he can jump over a sequence of entities giving the label of the next occurrences-set from which the navigation must continue. This will be made clearer in example Q9.

Q9 : Change some skill levels of employees working in the department 9012.

```
CONTEXT ?
S1 ←  DEPARTMENT ST NUMBER = 9012.
S2 ←  FOR S1 BY EMPLOYEE PERSON ST HAVE SKILL.
S3 ←  FOR S2 SKILL.
S4 ←  END.
COMMAND ?
UPDATE S2.NAME, S3.CODE, S3.LEVEL.
S2.NAME = CLARK
!          no update of this template, jump to the
           first skill; this attribute was in the
           list only so the user can check the name.
S3.CODE  = FRENCH
S3.LEVEL = 2
           level value is changed and the template
           returned to NUL.
S3.CODE  = GERMAN
S3.LEVEL = 1
! S2       Jump to the next employee, there is no-
           thing more to update for Clark.
S2.NAME  = CHERTON
! S2       Jump to the next employee.
```

```
S2.NAME  = JOHNSON
!
S3.CODE  = GERMAN
S3.LEVEL = 4
           level value is changed and the template
           returned to NUL.
S3.CODE  = FRENCH
S3.LEVEL = 3
! END      stop update command.
COMMAND ?
```

In this example, we see that '!' asks for the next entity and '!Si' restarts the navigation at the next occurrence of Si in the dynastic order.

NUL also permits a global modification of an attribute over one occurrences-set, e.g. after the skill modifications in Q9, give an annual gratuity of 500 dollars to all employees of department number 9012.

```
COMMAND ?
UPDATE ALL S2.GRATUITY.
S2.GRATUITY =        return template with value 500.
COMMAND ?
```

Remark : the part of the key inherited from another entity-set may not be used in an update-list.

Delete of entities

The 'DELETE' command can be applied to a unique entity or to an occurrences-set of the data-context. All dependent entities with strong links are automatically deleted :

Q10 : Delete the French skill of person number 54396 and all skills of level 1.

```
CONTEXT ?
S1 ← SKILL ST LEVEL = 1.
S2 ← END.
COMMAND ?
DELETE SKILL.
PERSON.NUMBER =
SKILL.CODE    =   enter values of the primary key.
COMMAND ?
DELETE ALL S1.
COMMAND ?
```

Manipulation of link occurrences

With the 'DETACH' command, the user can delete a link occurrence. He must know the primary keys identifying the entities involved in the link occurrence.

```
COMMAND ?
DETACH PERSON FROM DEPARTMENT BY EMPLOYEE.
DEPARTMENT.NUMBER =  enter values of
PERSON.NUMBER     =  the primary keys.
COMMAND ?
```

The insertion of a link occurrence is somewhat similar to the 'DETACH' command. The primary key of the entities to be linked are also needed :

```
COMMAND ?
ATTACH PERSON TO DEPARTMENT BY EMPLOYEE.
DEPARTMENT.NUMBER = enter values of
PERSON.NUMBER     = the primary keys.
COMMAND ?
```

Another command is also needed to change a link occurrence. As far as weak links are considered the two preceding commands can be used to achieve this. However, when a strong link occurrence is to be changed, a new mechanism has to be introduced to avoid inconsistencies in the data base. This new command may be used for weak links as well.

```
COMMAND ?
TRANSFER PERSON TO DEPARTMENT BY EMPLOYEE.
FROM DEPARTMENT.NUMBER =   enter values of
 TO  DEPARTMENT.NUMBER =   the primary keys.
     PERSON.NUMBER     =
COMMAND ?
```

NUL only allows deletion of a strong link occurrence when this operation does not imply deletion of an entity. So, one-to-many strong links cannot be deleted.

IV. SUMMARY AND CONCLUDING REMARKS

The data manipulation facilities of NUL have been outlined in this paper, providing the user with a restricted set of mechanisms with which he is able to navigate fairly easily through his data.

An implementation scheme is currently being studied in order to prove the feasibility of such a language.Moreover , a theoretical study leading to an extension to a data model with n-ary relationships could be fruitful. It seems to us that this approach should be investigated further in the future.

ACKNOWLEDGMENTS

The authors wish to thank Miss M. Hewitt, Messrs F. Bodart, J-L Hainaut, B. Lecharlier, H. Leroy and W. Paulus.

REFERENCES

[1] Tsichritzis, D., A network framework for relation implementation,IFIP-TC-2 Special Working Conference "A technical in-depth evaluation of the DDL", Namur, January, 1975.

[2] Codasyl, Data base task group report, ACM, New York, 1971.

[3] Deheneffe C., Hennebert H., Paulus W, "A relational model for a data base, Proc. IFIP Congress 74, North Holland Publishing Co., Amsterdam.

[4] Hainaut J.L.and Lecharlier B.,"An extensible semantic model of data base and its data language",Proc. IFIP Congress 74, North Holland Publishing Co., Amsterdam.

[5] Pin-Shan Chen P., "The entity-relationship model - Toward a unified view of data", ACM Transactions on Database Systems.

[6] Bachman C.W., "The programmer as navigator", CACM 16,11 November, 1973.

[7] Miisfiit, "Internal papers of CETE", Z.I. des milles Aix-en-Provence, France.

[8] Codd E.F., "A relational model for large shared data banks, CACM 13,6 June, 1970.

[9] Chamberlin D.D., Boyce R.F.,"SEQUEL : A structured English query language, Proc.1974 ACM-SIGMOD Workshop, May, 1974, Michigan.

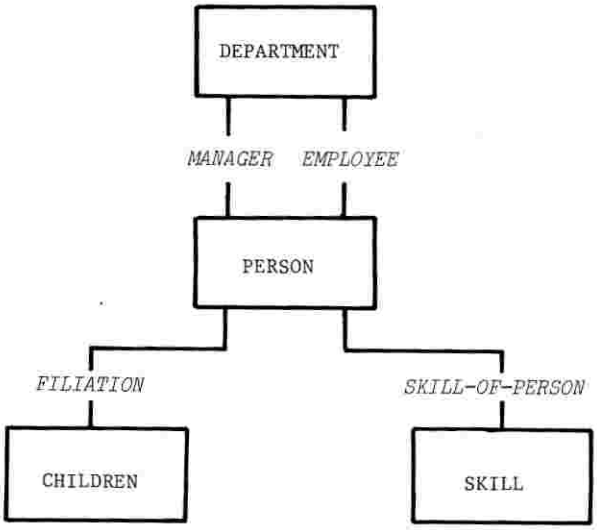
[10] Boyce R.F., Chamberlin D.D., King W.F., Hammer M.M., "Specifying queries as relational expressions : the SQUARE data sublanguage", CACM 18-11, November, 1975.

[11] Fehder P. L., "The representation-independent language", IBM Technical Report RJ 1121, IBM Research laboratory, San Jose, Calif., November, 1972.

APPENDIX

The syntax below has been stated only to give the reader a good idea of valid NUL expressions. It is not complete, for example, parentheses may be introduced in the expansion of ⟨ exp⟩.

```
⟨ request⟩ ::= ⟨ context⟩ ⟨ actions⟩

⟨ context⟩ ::= ⟨ statement⟩ | ⟨ statement⟩ ⟨ context⟩|
               END

⟨ statement⟩ ::= ⟨ qualification⟩ |
                 ⟨ for-clause⟩ ⟨ qualification⟩

⟨ for-clause⟩ ::= FOR ⟨ label ⟩ |
                  FOR ⟨ label⟩ BY ⟨ link-name⟩

⟨ qualification⟩ ::= ⟨ entity-set-name⟩ |
                  ⟨ entity-set-name⟩ SUCH THAT ⟨ exp⟩

⟨ exp⟩ ::= ⟨ exp-and ⟩ | ⟨ exp-and⟩ OR ⟨ exp⟩

⟨ exp-and⟩ ::=  ⟨ term ⟩ | ⟨ term ⟩ AND ⟨ exp-and⟩

⟨ term ⟩ ::= ⟨ attribute-name⟩ ⟨ rel-op⟩ ⟨ value-list⟩ |
    ⟨ have-clause⟩ ⟨ entity-set-name⟩ |
    ⟨ have-clause⟩ ⟨ entity-set-name⟩ = QF ⟨ integer⟩

⟨ have-clause⟩ ::= ⟨ have-arg⟩ |
                   ⟨ link-name⟩ ⟨ have-arg⟩

⟨ have-arg⟩ ::= ⟨ have-op⟩ | ⟨ have-op⟩ ⟨ quantifier⟩

⟨ have-op ⟩ ::= HAVE | IS | ARE | IN | BY | OF | WITH

⟨ quantifier  ::= SOME | ALL | NO | AT LEAST ⟨ integer⟩ |
                  AT MOST ⟨ integer⟩ | EXACTLY ⟨ integer⟩

⟨ rel-op ⟩ ::=   = | > | >= | ¬= | < | <=

⟨ value-list⟩ ::= ⟨ value⟩ | ⟨ value⟩ , ⟨ value-list⟩

⟨ value ⟩ ::= '⟨ string ⟩' | ⟨ number⟩

⟨ actions⟩  ::= ⟨ action-statement⟩ |
                ⟨ action-statement⟩ ⟨ actions⟩

⟨ action-statement⟩ ::= ⟨ print-clause⟩ |
                        ⟨ insert-clause⟩ |
                        ⟨ update-clause⟩ |
                        ⟨ delete-clause⟩ |
                        ⟨ detach-clause⟩ |
                        ⟨ attach-clause⟩ |
                        ⟨ transfer-clause⟩

                              ⋮
```



NUMBER	*TYPE*	*LOCATION*
5012	ADMIN	BRUSSELS
5624	RESEARCH	NAMUR
5000	RESEARCH	LIEGE

DEPARTMENT ENTITY-SET

NUMBER	*NAME*	*ADDRESS*	*SALARY*	*GRATUITY*	*BIRTHDATE*
50324	JOHNSON	NAMUR	10000	0000	1936
51396	SMITH	BRUSSELS	15000	250	1933
26204	JONES	BRUSSELS	17000	500	1935
12102	LEE	NAMUR	12000	300	1928
21180	LEE	BRUSSELS	15000	450	1941
41200	CLARK	LIEGE	16000	000	1947

PERSON ENTITY-SET

PERSON-NUMBER	*CHRISTIAN-NAME*	*BIRTHDATE*
50324	BILL	1953
50324	KATE	1956
51396	JOHN	1962
21180	BILL	1953

CHILDREN ENTITY-SET

PERSON-NUMBER	*LEVEL*	*CODE*
50324	1	FRENCH
50324	3	GERMAN
26204	3	FRENCH

SKILL ENTITY-SET

PERSON-NUMBER	*DEPARTMENT-NUMBER*
50324	5012
50324	5000
41200	5012
12102	5012

EMPLOYEE LINK (weak)

PERSON-NUMBER	*DEPARTMENT-NUMBER*
51396	5012
21180	5000
26204	5624

MANAGER LINK (weak)

FIG. 1.

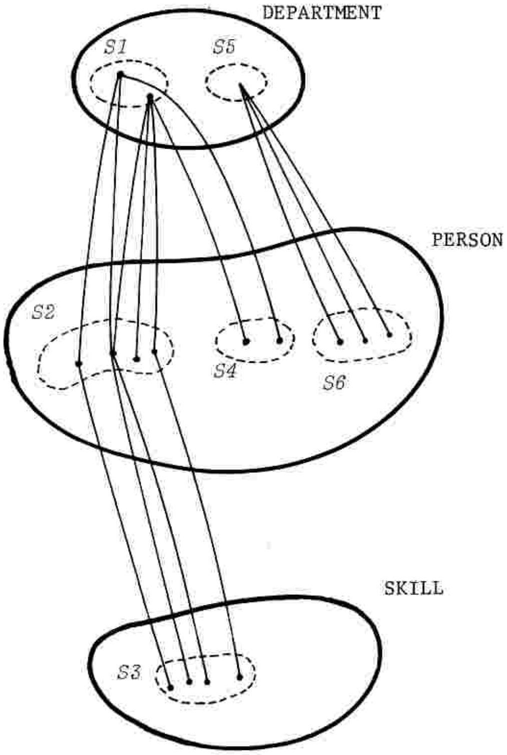
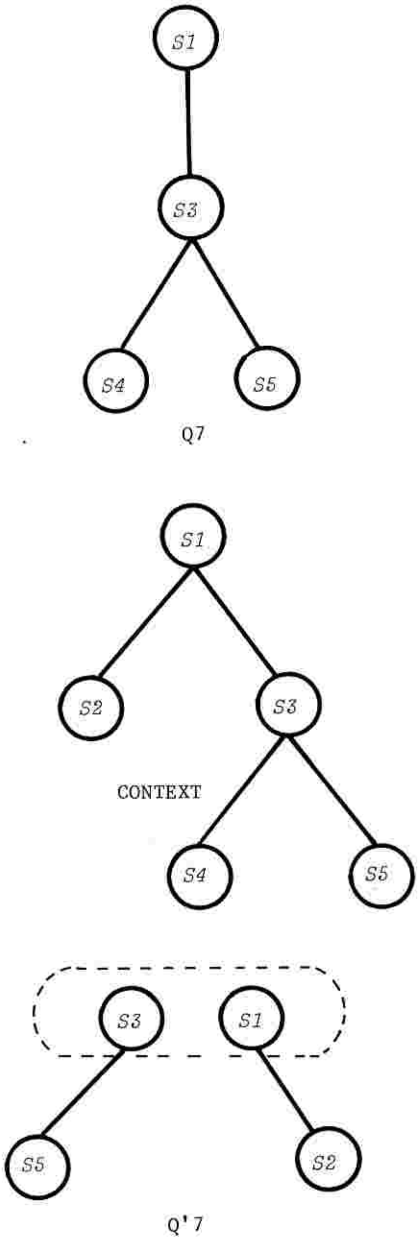


FIG. 2.



Q7

CONTEXT

Q'7

FIG.3.