HIGH-LEVEL NAVIGATIONAL FACILITIES FOR NETWORK AND RELATIONAL DATABASES

Kazimierz Subieta

Institute of Computer Science, Polish Academy of Sciences
P.O.Box 22, 00-901 Warsaw PKiN, Poland

The idea of navigation in a database, understood as a way of user thinking, is formulated in such a manner that no storage organization concepts are introduced to the user's awareness. The user may simultaneously control an unlimited number of navigational paths. Hence the proposed facilities are close to the features of high-level query languages. A concept of "navigational statement" is defined and studied. The navigational statements allow to a user to "walk through" a database. They may improve network database query languages. They may also be used to improve relational query languages, for example to improve the SEQUEL. Several aspects of the navigational statements, like incomplete statements, sets of statements, nested statements, etc. are presented.

INTRODUCTION

The idea of a navigation in a database was presented in [1] , where it was understood as a rather new way of user /programmer/ thinking. The "materialization" of this idea in the CODASYL DBTG proposal [5] meets with a great criticism /e.g. [7] /. The criticism concerns the following disadvantageous features:

- "a-record-at-a-time" technique, which is considered tiring for a user,
- many storage organization details are a burden for non-professionals and reduce the level of data independence,
- no mathematical definition.

Additionally, the currency mechanism was criticised as it may lead to situations a programmer finds hard to control.

In contrast, many high-level languages /sublanguages/ were introduced mainly in a context of the relational model, e.g. ALPHA [6] , SEQUEL and its successors [2] , [4] , and Query By Example [12] . These are free of disadvantages stated above and are based on well-established mathematical theories. Therefore they are claimed to be the future of database systems.

We return to the idea of navigation and we intend to show,that this idea, as a way of user thinking, is at least as good as the relational algebra, the relational calculus, the SEQUEL mappings, etc. In order to do this we try to remove three main sources of criticism: the "one-at-a-time" technique, dealing with storage details and lack of mathematical theory. Thus we introduce a formal model of access in a network database and formally define the semantics of the proposed facilities. They will be defined in such a way that no physical concepts are introduced to the user's awareness. Thus the level of data independence will not be reduced. Besides, a user will deal with navigation, where he can simultaneously control an unlimited number of navigational paths.

A basic concept introduced here is called the "navigational statement". From a syntactic viewpoint a navigational statement is a sequence of data names and data values. As far as semantics is concerned, a navigational statement is a function which maps a set of addresses onto an other set of addresses. /Addresses introduced here are "logical"; they should not be associated with physical addresses./ That is, a navigational statement determines the navigation from addresses which are "departure points" to addresses which are "arrival points".

The navigational statements may also be defined for the relational databases [10] ; what is more important, navigation in a relational database has some essential advantages in comparison to the well-known query language features,

In this paper we assume, that the navi-

gational statements are a tool which may be used for improvements in other database query languages, for example in SEQUEL, since the navigational statements defined here are not powerful enough to be a self-contained, complete query language. An alternative approach is possible, when well-known query facilities [11] are embedded into navigational statements making them relationally complete /if the property of relational completeness one considers essential/. Besides, the navigational statements may be embedded in a general purpose programming language /see e.g. [8] ,[9]/ which frees us of considerations about their completeness.

In this paper we consider some aspects connected with the idea in question, like some general properties which should be satisfied, elliptic /incomplete/ navigational statements, nested navigational statements, a "where" clause in navigational statements, sets of navigational statements and navigational joins. We intentionally do not introduce very sophisticated constructs since there are doubts if they will be really used in practical DBMS. We intend to show that navigational statements have positive impact on the user and machine efficiency, since they simplify most of the queries and they have a simple, effective implementation.

1. THE NETWORK DATABASE ACCESS MODEL

Let A be a set of addresses, N be a set of data names and V be a set of atomic values. Suppose that A, N and V are disjoint sets. We consider a database an ordered pair $\langle F,con\rangle$, where $F \subseteq A$ is called "field of vision", $con \subseteq A \times N \times (A \cup V)$ is a relation which will be called "database content". A database content contains triples $\langle a_1,n,a_2\rangle$ or $\langle a,n,v\rangle$. The first triple means that at address a_1 there is a datum with name n and with pointer leading to data at address a_2. The second triple means, that at address a there is a datum with name n and with atomic value v.

Example 1

The schema of a network database is presented in Fig.1. It has two types of records, EMPloyee and DEPartment and three relationships: WORKS_IN, EMPLOYS /reverse to WORKS_IN/ and MGR. Arrows indicate the access direction.

Three example records of such database may, in our formalism, be written as a database $\langle F_1,con_1\rangle$ where $F_1 = \{a_1,a_2,a_3,\dots\}$, $con_1 = \{\langle a_1,EMP,a_{11}\rangle, \langle a_1,EMP,a_{12}\rangle, \langle a_1,EMP,a_{13}\rangle, \langle a_1,EMP,a_{14}\rangle, \langle a_{11},E\#,E1\rangle, \langle a_{12},ENAME,Smith\rangle, \langle a_{13},SALARY,1000\rangle, \langle a_{14},WORKS\_IN,a_3\rangle, \langle a_2,EMP,a_{21}\rangle, \langle a_2,EMP,a_{22}\rangle, \langle a_2,EMP,a_{23}\rangle, \langle a_2,EMP,a_{24}\rangle, \langle a_{21},E\#,E2\rangle, \langle a_{22},ENAME,Brown\rangle, \langle a_{23},SALARY,2000\rangle, \langle a_{24},WORKS\_IN,a_3\rangle, \langle a_3,DEP,a_{31}\rangle, \langle a_3,DEP,a_{32}\rangle, \langle a_3,DEP,a_{33}\rangle, \langle a_3,DEP,a_{34}\rangle, \langle a_{31},D\#,D1\rangle, \langle a_{32},DNAME,Toy\rangle, \langle a_{33},MGR,a_2\rangle, \langle a_{34},EMPLOYS,a_1\rangle, \langle a_{34},EMPLOYS,a_2\rangle, \dots\}$.

We introduce some basic definitions and notation. Let $B \subseteq A$. We put

$$values(B,con) = \{v \in V: (\exists b \in B, n \in N)\ \langle b,n,v\rangle \in con\} \quad /1/$$

$$pointers(B,con) = \{a \in A: (\exists b \in B, n \in N)\ \langle b,n,a\rangle \in con\} \quad /2/$$

When con is fixed beforehand, we shall use values(B) and pointers(B). We define

$$pointers^0(B) = B$$

$$pointers^i(B) = pointers(pointers^{i-1}(B)) \quad \text{for } i = 1,2,\dots \quad /3/$$

and

$$pointers^*(B) = \bigcup_{i=0}^{\infty} pointers^i(B) \quad /4/$$

$pointers^i(B)$ denotes all addresses which are accessible from B in exactly i steps. $pointers^*(B)$ denotes all addresses accessible from B. We denote a projection of the relation con on the first domain by $con|_A$,

$$con|_A = \{a: (\exists n \in N, x \in A \cup V)\ \langle a,n,x\rangle \in con\} \quad /5/$$

We also denote $con^*(B) =$

$$= \{\langle a,n,x\rangle \in con: a \in pointers^*(B)\} \quad /6/$$

$con^*(B)$ denotes the set of data accessible from B in the database content con.

The following general constraints should be obeyed by any database $\langle F,con\rangle$:

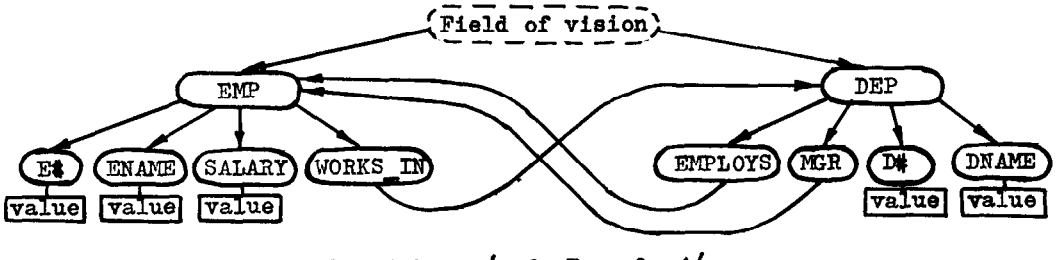

Fig.1. A schema of network database /c.f. Example 1/

/i/ No pointer leads to "garbage":

$$\text{pointers}(\text{con}|_A, \text{con}) \subseteq \text{con}|_A \qquad /7/$$

/ii/ The field of vision does not contain garbage: $F \subseteq \text{con}|_A$ /8/

/iii/ The whole database is accessible from the field of vision: $\text{con} = \text{con}^*(F)$ /9/

2. NAVIGATIONAL STATEMENTS - THE GENERAL ASSUMPTIONS

Let S be the set of navigational statements. Before considering the definition of syntax and semantics of S we have a look at some more general assumptions. Let $s \in S$. The semantics of the statement s we denote by $\|s\|$. By definition, $\|s\|$ is a function which maps a set of addresses into other set of addresses. The set of arguments of the navigational statements will be called the departure points and the result - arrival points. The function $\|s\|$ depends on particular database content con, thus $\|s\|$ is regarded as a mapping $\|s\|: \mathcal{P}(A) \times \text{CON} \longrightarrow \mathcal{P}(A)$, where CON is a set of all possible database contents. If there are no doubts which $\text{con} \in \text{CON}$ is investigated, instead of $\|s\|(B,\text{con})$ we shall write $\|s\|(B)$. The general syntactic assumption about the set S is as follows. We assume that given is a set σ of atomic statements such that any navigational statement may be represented as a sequence of atomic navigational statements. In the sequel the operation of concatenation will be represented by a dot or by nothing at all.

Let us consider a navigational statement $s \in S$, where $s = s_1 \cdot s_2 \cdot \ldots \cdot s_k$, $s_i \in \sigma$. If the statement s is applied to the set of addresses $A_0 \subseteq A$, then we can define a trajectory of navigation, which is a sequence of address sets $\langle A_0, A_1, A_2, \ldots, A_k \rangle$, where

$$\begin{aligned} A_1 &= \|s_1\|(A_0) \\ A_2 &= \|s_1 \cdot s_2\|(A_0) \\ &\ldots\ldots\ldots \\ A_k &= \|s_1 \cdot s_2 \cdot \ldots \cdot s_k\|(A_0) = \|s\|(A_0) \end{aligned} \qquad /10/$$

The trajectory of navigation /trajectory/ forms the path of "walking through" a database. The user may control the trajectory according to his needs by means of atomic statements s_i. These atomic statements may be, for example, data names or data values. For statement s and departure points A_0 the trajectory will be denoted $\text{traj}(s,A_0)$. The following general constraints concerning the navigational statements are assumed:

/iv/ The function s does not depend on the unaccessible part of the database:

$$\|s\|(B,\text{con}) = \|s\|(B,\text{con}^*(B)) \qquad /11/$$

/v/ For any one-to-one mapping $I: A \longrightarrow A$ holds:

$$I(\|s\|(B,\text{con})) = \|s\|(I(B),I(\text{con})) \qquad /12/$$

where $I(\alpha)$ denotes the structure formed from α by replacing each $a \in A$ occuring in α by $I(a)$. This constraint states that addresses are only auxiliary means for determining the connections among data, but their particular values are insignificant.

/vi/ If $\text{traj}(s,A_0) = \langle A_0, A_1, A_2, \ldots, A_k \rangle$ then for $i = 1,2,\ldots,k$ holds:

$$A_i \subseteq \text{pointers}^*\left(\bigcup_{j=0}^{i-1} A_j\right) \qquad /13/$$

This constraint says that navigation is "continous": a next element of a trajectory may be obtained from the former elements exclusively by means of pointers.

Corollary /by induction/:

$$A_i \subseteq \text{pointers}^*(A_0) \qquad /14/$$

The constraints appearing below are not mandatory for all the defined navigational statements, but if they hold, it may be nice for the user.

/vii/ Additivity:

$$\|s\|(B_1 \cup B_2) = \|s\|(B_1) \cup \|s\|(B_2) \qquad /15/$$

The navigational statements defined in this paper usually possess the above property. However, this constraint may be contrary to the postulated high selective power of navigational statements, therefore for some extensions of navigational statements it may be dropped.

/viii/ Simple Markov's property: Let $s = s_1 \cdot s_2 \cdot \ldots \cdot s_k$ be a navigational statement and let $\text{traj}(s,A_0) = \langle A_0, A_1, \ldots, A_k \rangle$. The simple Markov's property holds for statement s, if there exists a fixed function g such that for any such trajectory, for $i = 1,2,\ldots$ holds

$$A_i = g(s_i, A_{i-1}) \qquad /16/$$

/This property is analogous to the property of controlled Markov's processes./ As in the case of additivity, this constraint may be nice for the user since only the last result of navigation is essential for further navigation. However, for similar reasons as above, this constraint may be dropped.

/ix/ Generalized Markov's property: Given the assumption as in previous rule, the generalized Markov's property holds, if there exists a fixed function g and fixed natural number t such that for $i \geqslant t$, $i \leqslant k$

$$A_i = g(s_i, \langle A_{i-t}, A_{i-t+1}, \ldots, A_{i-1} \rangle) \qquad /17/$$

In other words, the result of i-th step of navigation depends on signal s_i and the results of t previous steps. If $t = 1$ then the generalized Markov's property is equivalent to the simple Markov's property. In this paper we assume, that the defined statements satisfy the simple Markov's property or the generalized Markov's property for $t = 2$.

In the sequel we define some classes of navigational statements for network databases

assuming that each next class is an extension of the previous classes. When there will be a danger of ambiguities between the metalanguage and the defined language, the syntactic objects of the defined language will be given in brackets ⟦ ⟧. These brackets do not have any other function.

3. SIMPLE NAVIGATIONAL STATEMENTS

Syntax: We assume $\mathcal{G} = N \cup V$ and the following rules:

/i/$_s$ $n \in N \Rightarrow n \in S$

/ii/$_s$ $s \in S \wedge n \in N \Rightarrow ⟦s.n⟧ \in S$

/iii/$_s$ $s \in S \wedge n \in N \wedge v \in V \Rightarrow ⟦s.n.v⟧ \in S$

Semantics: The semantics of simple navigational statements will be denoted by $\|...\|_s$.

/i/$_s$ $\|n\|_s(A_0) = \{a \in A_0 : (\exists x) \langle a,n,x\rangle \in con\}$ /18/

/ii/$_s$ $\|s.n\|_s(A_0) = \|n\|_s(pointers(\|s\|_s(A_0)))$ /19/

/iii/$_s$ $\|s.n.v\|_s(A_0) = \{a \in \|s\|_s(A_0) : (\exists b \in pointers(a))\langle b,n,v\rangle \in con\}$ /20/

Examples We refer fo Example 1 assuming th that the set of departure points is the field of vision F_1.

Q1. Give all employees: EMP

Q2. Give salaries of all employees: EMP.SALARY

Q3. Give all employees who earn 1500: EMP.SALARY.1500

Q4. Give the name of Smith's manager: EMP.ENAME.Smith.WORKS_IN.DEP.MGR.EMP.ENAME

Q5. Give the names of Toy employees, who earn 1500: DEP.DNAME.Toy.EMPLOYS.EMP.SALARY.1500.ENAME

4. ELLIPTIC NAVIGATIONAL STATEMENTS

For the simple navigational statements data names are used as "sign posts", and all names on the path of navigation must be used. Usually such precise determination is not necessary. Hence in navigational statements some names may be dropped. /The same question for the case of relational model is considered in [3]./ When we drop some names, the system must "spontaneously" search in the proper direction. In the sequel we assume that the number of spontaneous steps is limited to one. Similarly, elliptic navigational statements may be defined, where the number of spontaneous steps is greater, or where this number is unlimited. The elliptic navigational statements defined below contain simple navigational statements as a special case, therefore the numbering of rules starts from this section.

Syntax:

/i/ $n \in N \Rightarrow n \in S$

/ii/ $s \in S \wedge n \in N \Rightarrow ⟦s.n⟧ \in S$

/iii/ $v \in V \Rightarrow v \in S$

/iv/ $s \in S \wedge v \in V \Rightarrow ⟦s.v⟧ \in S$

Semantics: The meaning of the above navigational statements we shall denote $\|...\|$. The symbol $\|...\|_s$ will be used in the sense of the previous section.

/i/ $\|n\|(A_0) =$

$$= \begin{cases} \|n\|_s(A_0) & \text{if this set is not empty} \\ \|n\|_s(pointers(\|s\|(A_0))) & \text{otherwise} \end{cases} \quad /21/$$

/ii/ $\|s.n\|(A_0) = \|n\|(pointers(\|s\|(A_0)))$ /22/

/iii/ $\|v\|(A_0) = \{a \in A_0 : v \in values(pointers(a))\}$ /23/

/iv/ Let $traj(s,A_0) = \langle A_0, A_1, A_2, \ldots, A_{k-1}, A_k\rangle$. Thus

$$\|s.v\|(A_0) = \begin{cases} \{a \in A_{k-1} : v \in values(A_k)\} \\ \text{if this set is not empty} \\ \{a \in A_k : v \in values(pointers(a))\} \\ \text{otherwise} \end{cases} \quad /24/$$

Examples

Q6. The address of Smith's record may be obtained by one of the following statements: EMP.ENAME.Smith , ENAME.Smith , EMP.Smith , Smith , Smith.EMP .

Q7. The salary of Smith: Smith.SALARY

Q8. Give the record of Smith who earns 1500: Smith.1500

/If one considers this form "too elliptic", a less elliptic form may be used, e.g. EMP.Smith.SALARY.1500 ./

Q9. The names of Toy employees earning 1500: Toy.EMPLOYS.SALARY.1500.ENAME

5. SETS OF NAVIGATIONAL STATEMENTS

Now we assume that set S contains expressions which are equivalent to sets of expressions from the previous section.

Syntax: We introduce auxiliary syntactic domains called "factor" and "subfactor".

/v/ $x \in N \cup V \Rightarrow$ x is a factor, x is a subfactor

/vi/ $\theta \in \{=, \neq, <, \leqslant, >, \geqslant, \ldots\} \wedge v \in V \Rightarrow$ $⟦\theta v⟧$ is a factor, $[\theta v]$ is a subfactor

/vii/ $f_1, f_2, \ldots, f_l$ are subfactors $\Rightarrow$ $⟦(f_1, f_2, \ldots, f_l)⟧$ is a factor, $l = 1,2,\ldots$

/viii/ $f_1, f_2, \ldots, f_k$ are factors $\Rightarrow$ $⟦f_1.f_2. \ldots .f_k⟧ \in S$, $k = 1,2,\ldots$

Semantics: The semantics of factors will be denoted by $\|...\|_f$.

/v/ $\|x\|_f = \{x\}$ /25/

/vi/ $\|\theta v\|_f = \{u : u\theta v\}$ /26/

/vii/ $\|(f_1, f_2, \ldots, f_l)\|_f = \|f_1\|_f \cup \|f_2\|_f \cup \ldots \cup \|f_l\|_f$ /27/

/viii/ $$\|f_1.f_2. \ldots .f_k\|(A_0) = \bigcup_{s \in S_1} \|s\|(A_0)$$ /28/

where $S_1 = \|f_1\|_f \times \|f_2\|_f \times \ldots \times \|f_k\|_f$ /29/

Examples

Q10. Give employees who earn more than 2500:
EMP.SALARY.>2500

Q11. Give names of employees working in the Toy or Stationery departments earning more than 1000 and less than 1500:
DEP.(Toy,Stationery).EMPLOYS.EMP.SALARY.
>1000.SALARY.<1500.ENAME

6. NESTED NAVIGATIONAL STATEMENTS

In a navigational statement the value v can be replaced by an other navigational statement. The meaning of the inner statement s_1 may be considered a set of values placed at addresses which constitute the arrival points of navigation according to s_1. /This idea is employeed e.g. in SEQUEL [4]' ./ Thus, after this meaning is known, the outer statement may be evaluated as before. For the sake of syntactic distinction the inner statement will be given in square brackets.

Syntax:

/ix/ $s \in S \Rightarrow [\![[s]]\!] \in S$

/x/ $s_1, s_2 \in S \Rightarrow [\![s_1.[s_2]]\!] \in S$

Semantics:

/ix/ $\|[s]\|(A_0) = \bigcup_{v \in V_1} \|v\|(A_0)$, /30/

where $V_1 = values(\|s\|(A_0))$

/x/ $\|s_1.[s_2]\|(A_0) = \bigcup_{v \in V_2} \|s_1.v\|(A_0)$ /31/

where $V_2 = values(\|s_2\|(A_0))$

Example

Q12. Give employees whose salaries are the same as the salary of Smith:
EMP.SALARY.[Smith.SALARY]

7. A "WHERE" CLAUSE FOR NAVIGATIONAL STATEMENTS

As in many other database languages, after the key word "where" there appears a formula, which tests addresses belonging to the recently obtained result. For each of these addresses the formula produces a truth value and addresses for which the formula is false are removed from the result.

Syntax:

/xi/ $s_1, s_2 \in S \wedge \theta \in \{=, \neq, <, \leq, \ldots\} \Rightarrow$
$[\![s_1 \theta s_2]\!]$ is a formula

/xii/ $s \in S \Rightarrow$ s is a formula

/xiii/ s $\in$ S and g is a formula $\Rightarrow$
$[\![s \text{ where } (g)]\!] \in S$

Semantics: The semantics of formulas will be denoted by $\|\ldots\|_b$.

/xi/ $\|s_1 \theta s_2\|_b(A_0) = true \iff$
$\iff (\exists v_1 \in values(\|s_1\|(A_0)))$ /32/
$(\exists v_2 \in values(\|s_2\|(A_0))) \; v_1 \theta v_2$
/Otherwise it is false./

/xii/ $\|s\|_b(A_0) = true \iff \|s\|(A_0) \neq \emptyset$ /33/
/Otherwise it is false./

/xiii/ $\|s \text{ where } (g)\|(A_0) =$ /34/
$= \{a \in \|s\|(A_0) : \|g\|_b(pointers(a)) = true\}$

Examples

Q13. Give employees who earn more than their manager:
EMP where (SALARY>WORKS_IN.MGR.SALARY)

Q14. Give employees who work for Brown and earn more than 2500:
EMP where (DEP.MGR.Brown).SALARY.>2500.ENAME

8. NAVIGATIONAL JOINS

Now we assume that the meaning of navigational statement is a function which to some set of addresses subordinates a some k-ary relation over addresses.

Syntax:

/xiv/ $s_1, s_2, \ldots, s_k \in S \Rightarrow$
$[\![s_1 \times s_2 \times \ldots \times s_k]\!] \in S$

/xv/ $s, s_1, s_2, \ldots, s_k \in S \Rightarrow$
$[\![s.(s_1 \times s_2 \times \ldots \times s_k)]\!] \in S$

Semantics:

/xiv/ $\|s_1 \times \ldots \times s_k\|(A_0) =$ /35/
$= \{\langle a_1, \ldots, a_k \rangle : (\exists a \in A_0) \; a_i \in \|s_i\|(a)\}$

/xv/ $\|s.(s_1 \times \ldots \times s_k)\|(A_0) =$ /36/
$= \|s_1 \times \ldots \times s_k\|(\|s\|(A_0))$

Examples

Q15. Associate employees with their departments: EMP X EMP.WORKS_IN.DEP

Q16. Associate names of employees with their salaries and with names of their managers:
ENAME X SALARY X WORKS_IN.MGR.ENAME

Many other rules for navigational joins may be introduced.

9. NAVIGATIONAL STATEMENTS FOR RELATIONAL DATABASES

Theoretically, the navigational statements defined so far may be used for the relational database. For example, for the supplier-and-part model the query:

Q17. Give names of suppliers supplying Bolts.

may be expressed as: SUPPLIER.S#.
[SP.P#.[PART.PNAME.Bolt.P#].S#].SNAME

However, we are forced to an extensive use of the nested navigational statements, thus the proposed facility can not confirm its advantages. Therefore we formulate navigational statements for the relational database somewhat differently by introducing a new technique of navigation. The main idea relies on navigation according to identical

values stored in a database. That is, when during a navigation we reach an address in a database with value v, we simultaneously reach all other addresses with value v. Incidentally, this technique may also be used as an additional tool for navigation in a network database.

The relational database

We introduce the following sets: R - a set of names of relations, $\mathcal{A}$ - a set of names of attributes and V - a set of values /a universal domain/. Let $X \subseteq \mathcal{A}$. X-tuple /tuple/ is a mapping which associates a value with each name in X. We shall write "t maps a" if t is an X-tuple and $a \in X$. Row is an ordered pair $\langle r,t \rangle$, where $r \in R$, t is a tuple. Relational database /database/ is a set of rows. Place is an ordered triple $\langle r,t,a \rangle$, where $\langle r,t \rangle$ is a row, $a \in \mathcal{A}$. The place $\langle r,t,a \rangle$ has a value $t(a)$. A place determines the fragment of a table lying at the intersection of a row and a column. The remaining relational model folklore is inessential for this paper.

Let db be a database and let P denote the set of all places of database db, i.e.

$$P = \{\langle r,t,a \rangle : \langle r,t \rangle \in db \text{ and } t \text{ maps } a\} \quad /37/$$

We assume that given is an equivalence relation $E \subseteq P \times P$. This relation determines classes of places with identical "key" values, e.g. the definition of relation E may be the following:

$$\langle r_1,t_1,a_1 \rangle \, E \, \langle r_2,t_2,a_2 \rangle \text{ iff } \langle r_1,t_1,a_1 \rangle = \langle r_2,t_2,a_2 \rangle \text{ or } t_1(a_1) = t_2(a_2) \in V_k \quad /38/$$

where $V_k \subseteq V$ is a subset of "key" values.

We assume that relation E defines the additional, "predefined" access paths to data. It may be easily recognized that rows and places are addresses in the sense of the previous section.

Navigational statements

We define only simple navigational statements. As in the previous sections, all the other aspects may also be considered. More about them is presented in [10]. Syntactically, we assume that navigational statements are sequences of elements of $R \cup \mathcal{A} \cup V$. The meaning of a navigational statement is a function whose argument is a set of rows /a set of places/ and whose values is also a set of rows /a set of places/. The meaning of statement s will be denoted by $|s|$. We assume that

$$|s.n|(Y) = |n|(|s|(Y)) \quad /39/$$

where s is an arbitrary navigational statement, $n \in R \cup \mathcal{A} \cup V$. This property /which implies the simple Markov's property/ allows us to simplify the semantics: it can be given for one-element statements only.

A column name $a \in \mathcal{A}$ may be applied to a set of rows H, with the meaning:

$$|a|(H) = \{\langle q,t,a \rangle : \langle q,t \rangle \in H \wedge t \text{ maps } a\} \quad /40/$$

This operation may be recognized as the usual projection. A relation name $r \in R$ may be applied both to the set of rows H or to the set of places Q. When relation name r is applied to the set of places Q, it implies a navigation from places in Q to places connected with them by relation E. Formally

$$|r|(H) = \{\langle r,t \rangle : \langle r,t \rangle \in H\} \quad /41/$$

$$|r|(Q) = \{\langle r,t \rangle : (\exists b \in \mathcal{A})(\exists p \in Q)\ p \, E \, \langle r,t,b \rangle\} \quad /42/$$

A value v may be applied to a set of places Q. As a result we obtain a set of rows possessing a place from Q having value v, i.e.

$$|v|(Q) = \{\langle x,t \rangle : (\exists b)\langle x,t,b \rangle \in Q \wedge t(b) = v\} \quad /43/$$

Examples We assume the usual "supplier-and-part" model with a description:
S (S#,SNAME,CITY) SP (S#,P#,QTY)
P (P#,PNAME) . We also assume that the argument of a navigational statement is the whole database /all rows/ and that V_k contains values from domains S# and P#.

Q18. Names of parts supplied by Smith:
S.SNAME.Smith.S#.SP.P#.P.PNAME

Q19. Names of parts supplied by suppliers supplying Bolts:
P.PNAME.Bolt.P#.SP.S#.SP.P#.P.PNAME

Q20. The quantity of part P2 supplied by Smith from London:
S.SNAME.Smith.CITY.London.S#.SP.P#.P2.QTY

10. REMARKS ABOUT IMPLEMENTATION

The navigational statements have a very simple and effective implementation. In the network case the usual means of connecting data, e.g. pointers, may immediately be used. These may be associated with inverted tables in order to avoid sequential searching at first steps of navigation.

The implementation of the technique which we apply for the relational database is also easy and effective. Below we briefly describe one of the possible methods. We assume, that our pointers may lead to places. Each place with a value belonging to V_k is associated with such a pointer. The pointers create rings, where each ring connects places with the same value. Hence a basic navigational operation - a transition from some place p to all places q such that pEq - may be performed very quickly. This method may also be associated with inverted tables.

CONCLUSION

We propose new language facilities for the casual user based on a navigation concept. We show that these facilities may be used as an improvement of network database query languages, or, after extensions, they may constitute a self-contained, high-level query language. They may also be used for the relational database, where a somewhat different technique of navigation is assu-

med. The navigational statements, as a query language, have many advantages, such as simple syntax, simple and formal semantics, brevity, possibility of handling incomplete statements and simple, effective implementation

REFERENCES

[1] Bachman,C.W.: The programmer as navigator. CACM 16/11 /1973/ 653-658

[2] Blasgen,M.W.,et al.: System R: An architectural overview. IBM System Journal 12/1 /1981/ 41-62

[3] Carlson,C.,and Kaplan,R.: A generalized access path model and its application to a relational data base systems. Proc.ACM SIGMOD Intl.Conf. on the Management of Data /1976/

[4] Chamberlin,D.D.,et al.: SEQUEL-2: a unified approach to data definition, manipulation and control. IBM J.Res.Dev. 20/6 /1976/ 560-575

[5] CODASYL Data Base Task Group Report. ACM New York /1971/

[6] Codd,E.F.: A data base sublanguage founded on the relational calculus. Proc. ACM SIGFIDET Workshop on Data Description, Access and Control /1971/ 35-68

[7] Date,C.J., and Codd,E.F.: The relational and network approaches: comparison of the application programming interfaces. Proc.ACM SIGMOD Workshop on Data Description, Access and Control /1974/

[8] Date,C.J.: An introduction to the unified data language UDL. Proc. of 6-th Intl.Conf. on VLDB, Montreal /1980/ 15-32

[9] Stonebraker,M., and Rowe,L.A.: Observations on data manipulation languages and their embedding in general purpose programming language. Proc. of 3-rd Intl.Conf. on VLDB, Tokyo /1977/ 128-143

[10] Subieta,K.: Navigational facilities for relational data base. Information Systems 8/1 /1983/ 29-36

[11] Subieta,K.: Semantics of non-procedural language for network database. Institute of Computer Science PAS Report 449 /1981/ /submitted to ACM Transactions on Database Systems/

[12] Zloof,M.: Query by Example. AFIPS Conf. Proc. vol.44 /1975/ 431-438