GOING --- A DATA SUBLANGUAGE USING A GRAPHICS DISPLAY

（グラフィックス・ディスプレーを用いたデータ準言語 GOING）

Yoshihisa Udagawa
and Setsuo Ohsuga

Institute of interdiscriplinary research
faculity of engineering, Tokyo University
4-6-1 Komaba, Meguro-ku, Tokyo 153, Japan

1. Introduction

This paper concerns design and implementation of a friendly user interface language for the relational databases. A less procedural or non-procedural language is considered to be preferable to end user languages, bacause it permits a user to focus on the functional specifications of a given problem rather than on its solution methodology.

So far, a number of non-procedural interface languages have been proposed for expressing queries against relational databases. GOING (a Graphics Oriented INteractive data sublanGuage) , the language discussed in this paper, belongs to the graphics oriented data languages. By taking advantage of two-dimensional representation of a query by using a graphics display , queries are expressed within a simple and easy-to-understand conceptual framework. GOING is designed to enable the user to express queries in terms of nodes, arcs, comparison predicates and functions. Other well known language with a similar basic orientation includes Query-by-Example [8] and CUPID [2].

The main features of GOING are as follows :

(1) GOING uses little Englishlike text, thus avoiding difficult language processing by the system and spelling mistakes by the users ;
(2) GOING does not require knowledge on the predicate logic ;
(3) GOING does not require to invent the variable names ;
(4) GOING provides a concise, easy-to-understand representation of queries ;
(5) GOING enables the user to state a query in a non-procedural way ;
(6) GOING provides high expressive power and has strong theoretical basis.

In section 2, we illustrate GOING and compare it with two other graphics oriented languages, i.e. CUPID, Query-by-Example. Section 3 discusses how to define a virtual relation through a GOING expression. Section 4 contains the proof of relational completeness of the data sublanguage GOING. Section 5 deals briefly with implementation of the overall database system.

2. Describing queries in terms of GOING expressions, comparing GOING with CUPID and Query-by-Example

In this section , the query language GOING is illustrated by means of example queries and is compared with other graprhics oriented languages, i.e. CUPID, Query-by-Example. The database consists of the following relations.

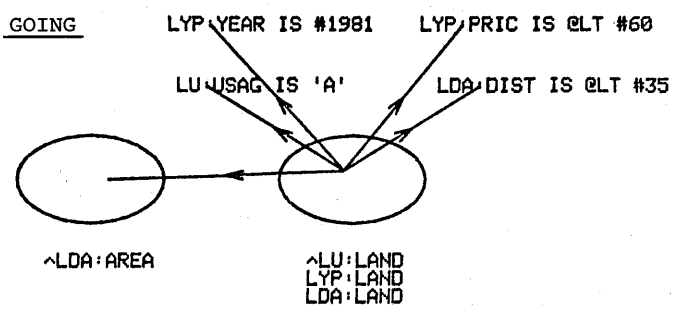
LYP(LAND, YEAR, PRIC) ;
LU (LAND, USAG) ;
LDA(LAND, DIST, AREA).

The relation LYP has a row giving price for each land's identifier and year. The relation LU gives the usage of each land. The relation LDA gives, for each land, its area and the distance from the center of a city in which it is located.

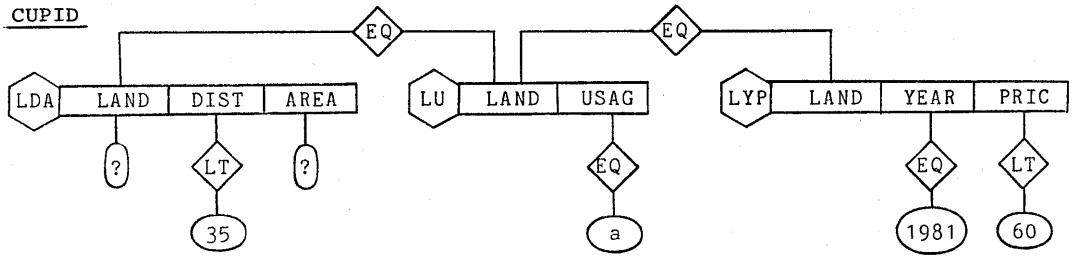
A query against more than one relation with Boolean conditions

Query 1. List the lands of usage a and their areas, which are less than 35 kilometer aprat from the center of a city and whose prices are less than 600,000 YEN/m^2 in the year 1981.

A domain or literal expression preceded by "^" symbol denotes to list the value of it. The arc which connects inside of the right ellipse and the first argument of the Boolean predicate LU:USAG IS 'A' indicates that there are some lands in the column LAND in the relations LU, LYP, LDA whose usages are 'A'.



The literal expressions LYP:PRIC IS @LT #60 and LDA:DIST IS @LT #35 denote that the values of the price in the relation LYP and the values of the distance in the relation LDA are less than 60 and 35, respectively.



Query-by-Example

LYP	LAND	YEAR
	1	'1981'

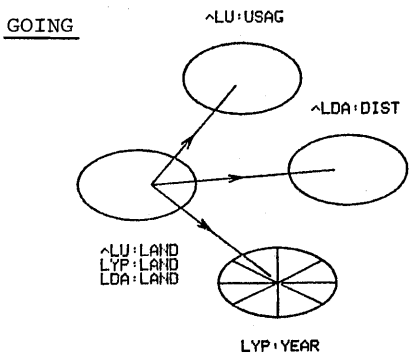
LU	LAND	USAG
	P.	'c'

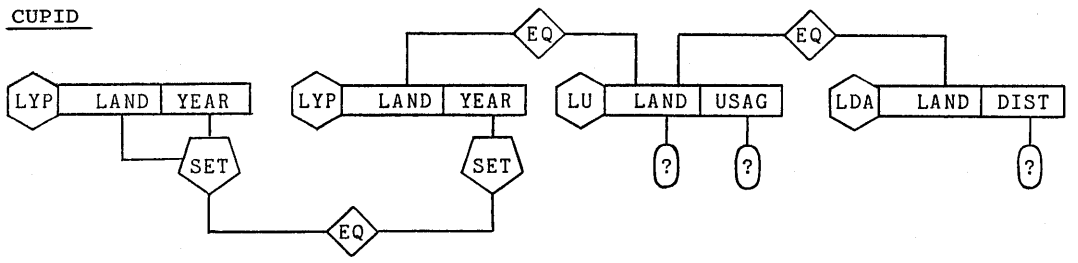
LDA	LAND	DIST	AREA
	1	<'35'	P.

A query against more than one relation with a universal quantification

Query 2. List the lands which are inquired in all the years, their usage and distance from the center of a city.

Note that only those domains referred need be represented in the GOING expression. In this example, the column PRIC in the relation LYP and column AREA in the relation LDA are not used in the above expression.





Query-by-Example

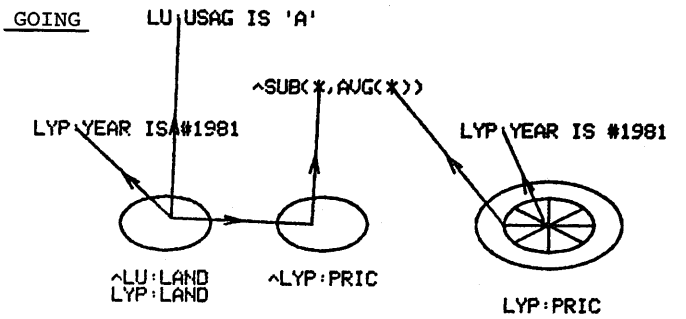
LYP	LAND	YEAR
	P. 1	all

LU	LAND	USAG
	1	P.

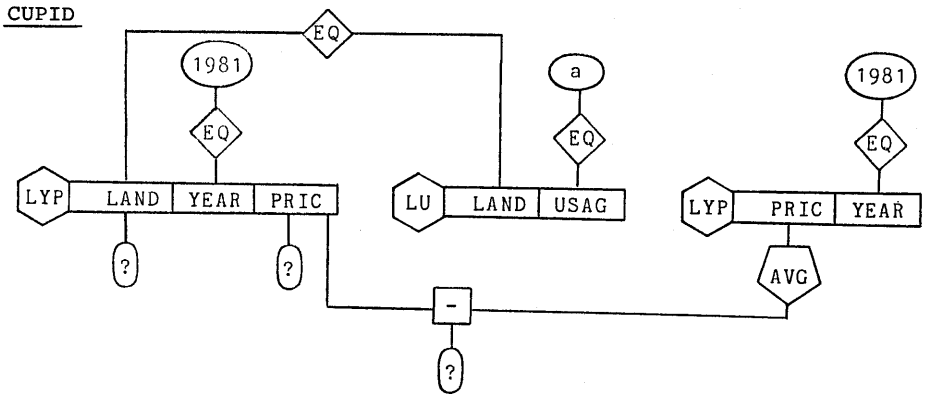
LDA	LAND	DIST
	1	P.

A query using built-in arithmetic and aggregation fuctions

Query 3. List, for each land of usage a and inquired in the year 1981, its identifire, its price and the difference of the price with the average price computed on all lands inquired in the year 1981.



The literal expression SUB(*,AVG(*)) demonstrates that a built-in function may be nested to any levels so far as the value of arguments are properly defined. Note that first argument of SUB(*, AVG(*)) is some values of column PRIC in the relation LYP, while the argument of AVG(*) is some subsets of column PRIC. This difference is explicitly expressed in the above GOING expression.

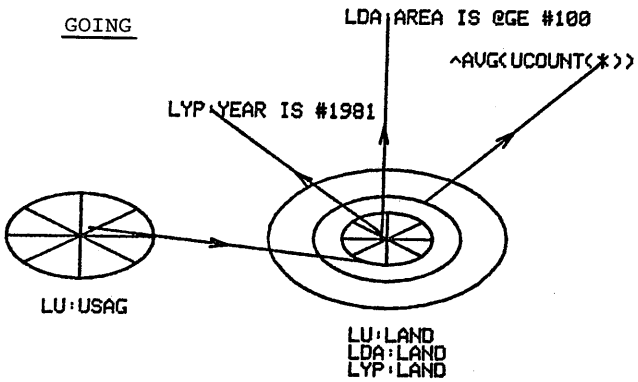


Query-by-Example : Not expressible

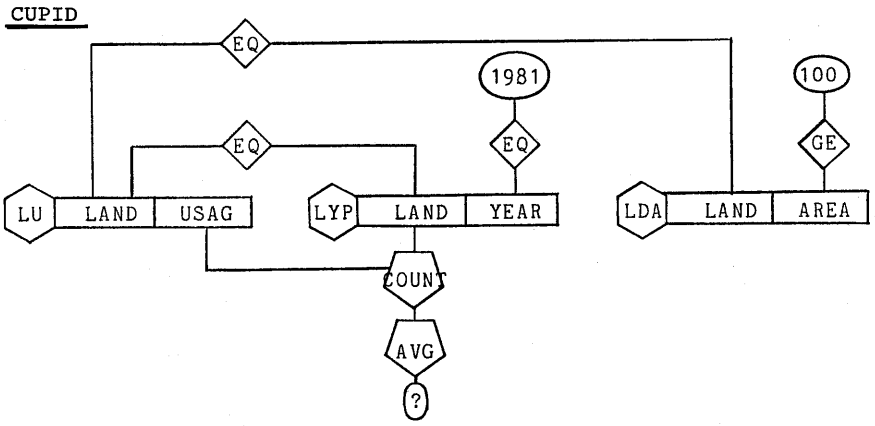
A query with a nested aggregation function and a Boolean condition

Query 4. What is the average number of lands per usage, which are inquired in the year 1981 and whose areas are not less than 100 square meter.

This query contains the nested aggregation function AVG(UCOUNT(*)). The argument of this function is a set of subsets of lands inquired in the year



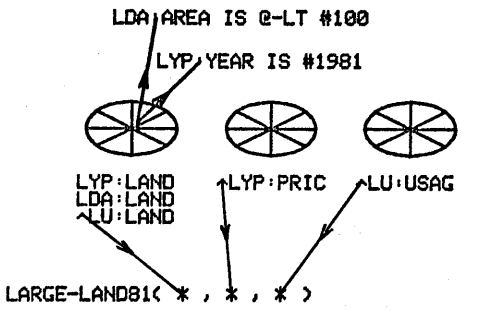
1981 and corresponding area is greater than or equal to (not less than) 100 square meter. The subsets of lands are determined for each usage. This fact is represented by the arc connecting inside of the ellipse LU:USAG and the innermost ellipse of the right figure.



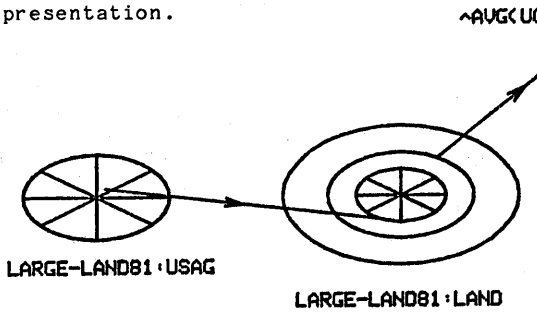
Query-by-Example : Not expressible

3. Defining a virtual relation through a GOING expression

A virtual relation supports high level relational database schema. In other words, it provides the language level interface which is suited to user's goals. Virtual relations are an important component of a database system because (1) users often focus on a subset of the database, (2) from the human-machine point of view, the re-execution of similar queries causing extensive retrievals is expensive and increase the human labor. GOING allows a user to define a virtual relation that can be used for formulating queries. An example is given here. A virtual relation Large-land81(l,p,u) is defined by the following GOING expression by using the base relations LYP, LDA and LU. It gives, for each land inquired in the year 1981 and whose area is not less than 100 square meter, its price and usage. Once Large-land81(l,p,u) is defined , a user can describe the query 4 in section 2 by the following GOING expression.



Comparing this expression with one for the query query 4 illustrated in section 2, we find that the former GOING expression is shorter and clearer than the latter in its representation.



4. Relational completeness of the data sublanguage GOING

The concept of "relational complete" is defined by [1] and is an accepted standard for the theoretical power of relational database query languages. A query language is said to be relational complete if any query expressible by a relational algebra (a relational calculus) can be described by a statement in the language.

In this section, we provide the operations in the relational algebra and the semantically equivalent GOING expressions. Thus, GOING is shown to be at least relational complete, in fact it is strictly more powerful than the relational algebra , which will be also discussed.

In the following, let us assume that R and S are relations over the attributes A1...Am and B1...Bn respectively. A and B represent subsets of {Ai}, {Bj}, where $1 \leq i \leq m$, $1 \leq j \leq n$, respectively. $\vec{x}$, $\vec{y}$ indicates the vectors of variables $\{x_i\}$, $\{y_j\}$ defined over the attributes A, B. $\overline{A}$ denotes the subset complementary to A, i.e. $\overline{A} = \{Ai\} - A$. $\vec{b}$ is a tuple of appropriate degree with appropriate attributes.

In section 4.1 , we list each operator of relational algebra, the formula in the relational calculus, the formula in the multi-layer logic [3,4,6] and the corresponding expression in GOING. Section 4.2 deals briefly with the gaps between GOING and relational algebra.

4.1. Relational operators and corresponding GOING expressions

Codd uses the four operators union, intersection, difference, and Cartecian product to manipulate relations as sets of tuples. The four special relational operators, i.e., projection, join, restriction and division, are used to give the relational algebra its selective power. Note that the division operator is not strictly necessary for relational completeness.

Union

Relational algebra : $R \vee S$

Relational calculus : $\{ r \mid r \in R \vee r \in S \}$

Multi-layer logic : $(\exists \vec{x}/A) [R(\vec{x}) \vee S(\vec{x})]$

GOING :

Intersection

Relational algebra : $R \cap S$

Relational calculus : $\{ r \mid r \in R \text{ and } r \in S \}$

Multi-layer logic : $(\exists \vec{x}/A) [R(\vec{x}) \& S(\vec{x})]$

GOING :

Difference

Relational algebra : $R - S$

Relational calculus : $\{ r \mid r \in R \text{ and } r \notin S \}$

Multi-layer logic : $(\exists \vec{x}/A) [R(\vec{x}) \& \neg S(\vec{x})]$

GOING :

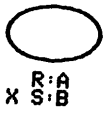
Cartesian Product

Relational algebra : $R \times S$

Relational calculus : $\{ (r,s) \mid r \in R \text{ and } r \in S \}$, where (r, s) denotes the ordered pair of r and s in that order.

Multi-layer logic : $(\exists \vec{x}/A)(\exists \vec{y}/B) [R(\vec{x}) \& S(\vec{y})]$, where sets of domains A and B are disjoint, i.e., $\{A\} \cap \{B\} = \emptyset$.

GOING :



Projection

Relational algebra : R [A]

Relational calculus : { r[A] ¦ r ∈ R}

Multi-layer logic : $(\exists \vec{x}\hat{}/A)$ [R(...,$\vec{x}$,...)] , where ... indicates the relation contain other columns.

GOING :

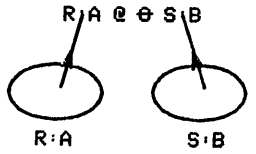
Theta-join

Relational algebra : R [A θ B] S , where θ is one of the Boolean conditions { = , ≠ , > , < , ≥ , ≤ }.

Relational calculus : { (r,s) ¦ r ∈ R and r ∈ S and (r[A] θ s[B]) } .

Multi-layer logic : $(\exists \vec{x}/A)(\exists \vec{y}/B)$ [R(...,$\vec{x}$,...) & S(...,$\vec{y}$,...) & BC($\vec{x}$,$\vec{y}$)], where BC is an element of Boolean conditions { EQ, NE, GT, LT, GE, LE }.

GOING :



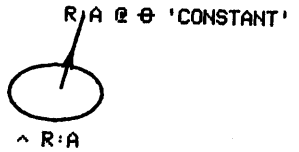
Restriction

Relational algebra : R [A θ 'b'] , where θ is an element of { = , ≠ , > , < , ≥ , ≤ }.

Relational calculus : { r ¦ r ∈ R and (r[A] θ '$\vec{b}$') }.

Multi-layer logic : $(\exists \vec{x}/A)$ [R(...,$\vec{x}$,...) & BC($\vec{x}$,'$\vec{b}$')] , where BC is an element of { EQ, NE, GT, LT, GE, LE }.

GOING :



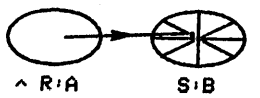
Division

Relational algebra : R [Ai ÷ Bj] S

Relational calculus : { r[$\overline{Ai}$] ¦ r ∈ R and { $\vec{y}$ ¦ (r[$\overline{Ai}$], $\vec{y}$) ∈ R } ⊇ S[Bj] }, where (r[$\overline{Ai}$], $\vec{y}$) denotes the ordered pair of r[$\overline{Ai}$] and y in that order.

Multi-layer logic : $(\forall \vec{x}_i/Ai)$ [S[$\vec{x}_i$] → R(...,$\vec{x}_i$,...)], where Ai is the domain of a variable $\vec{x}_i$ and S[x_i] is an abbreviation of $(\exists \vec{x}_1/A_1)\ldots(\exists \vec{x}_{i-1}/A_{i-1})\ (\exists \vec{x}_{i+1}/A_{i+1})\ldots(\exists \vec{x}_n/A_n)$ $S(\vec{x}_1,\ldots,\vec{x}_i,\ldots\vec{x}_n)$.

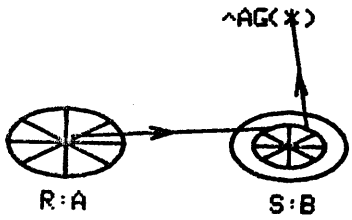
GOING :



4.2. The gaps between the GOING expression and the relational algebra

The relational algebra has two deficiencies for practical use. First, it does not contain arithmetic functions. In GOING, as shown in section 2 , the arithmetic functions are provided as built-in functions. They can be composed so far as domains and ranges are properly defined. The other problem with the

relational algebra is that the relational algebra is essentially a first order language. Thus, queries containing grouping operations and aggregation functions can not be expressed. Since, in GOING, the concept of a set and a set of sets etc. are dealt with explicitly, grouping operations and aggregation functions are described without expedient. (Note that any GOING expressions correspond to a formula in the multi-layer logic. This fact is discussed in [3,4,7].) For example, the expression below applies the given aggregation function AG to a subset of S:B , which is determined for each elememt of R:A.



5. Implementation of a database system

Figure 1 illustrates an overall database system designed and implemented. This system supports a user interface language GOING. The query described by GOING expression is then translated into the corresponding formula in the multi-layer logic by the GOING processor. GOING has been implemented on top of the MRDOS operating system for Data General Corporation ECLIPSE S/130 computer system and programmed in FORTRAN IV. The results of experiments show that GOING processor is efficient enough comparing with database access. Typcal GOING expressions are translated into corresponding formulas in the multi-layer logic in a second.

A database system based upon the multi-layer logic, named SBDS-F3 (Structure Based Deduction System - Fortran version 3) has also been developed by modifying SBDS-F2 that is based on the many-sorted logic. SBDS-F3 parses a query expressed in the multi-layer logic, developes a virtual relation (if any) by using its definition and reduces the resultant query into a sequence of operations in the relational algebra. Thus, in the overall database system , SBDS-F3 plays the role of transforming non-procedural expressions into corresponding procedural expressions. On SBDS-F3 , [3,4,6] can be referred.

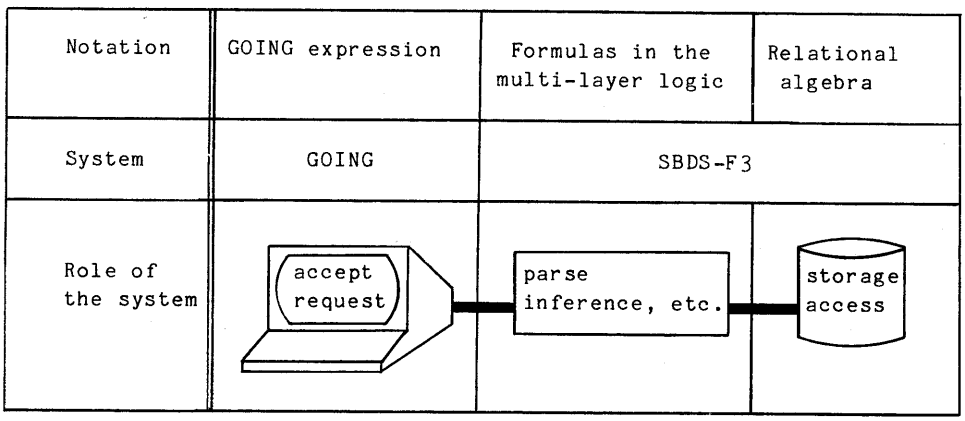


Figure 1. Overall database system designed and implemented.

ACKNOWLEDGEMENTS

I wish acknowledge all the members of Meetings of Information Systems for their encouragements and constructive comments.

I would like to express my appreciation to Dr.K.Agusa, Ohno Lab. of Kyoto University , for providing the SAFE editor system. The SAFE system is very useful and exclusively used for preparing this paper.

REFERENCES

1] Codd,E.F. "Relational completeness of data base sublanguages," Data Base Systems, Courant Computer Science Symposia Series, Vol.6 (R. Rustin, Ed.), Prentice-Hall, (1971),pp65-98

2] McDonald,N. "CUPID -- A graphics oriented facility for support of non-programmer interactions with a data base," ERL, Univ. Calif. Berkeley, Mem #ERL-M563, (Novem. 1975).

3] Udagawa,Y " A Study on Design and Implementation of a Database System Based on Predicate Logic," Doctorial Thesis, Tokyo University, February,1982.

4] Udagawa, Y. and Ohsuga,S. "The multi-layer logic as a relational database query language and its reduction algorithm to relational procedures," (in Japanese), Joho-Shori (submitted).

5] Udagawa,Y. and Ohsuga,S. "Design and implementation of a database system based on the multi-layer logic," Proc. of Advanced Database Symposium (Dec. 1981) pp31-42.

6] Udagawa,Y. and Ohsuga,S. "Construction of SBDS-F3 : a relational database with inference mechanism," RIMS, Univ of Kyoto, Kokyu-Roku, 1982 (in printing).

7] Udagawa,Y. and Ohsuga,S. "Implementation of GOING : a data-language using graphics display," RIMS, Univ of Kyoto, Kokyu-Roku, 1982 (in printing).

8] Zloof,M.M. : Query-by-Example : a data base language, IBM Syst. J.4, pp324-343(1977).

APPENDIX

Commands for GOING expression

Command 1 : <c> S m <c/r> .
Command 2 : <c> P m <c/r> .
Command 3 : <c> T m <c/r> .
Command 4 : <c> N s <c/r> .
Command 5 : <c> A <c> E <c/r> .
Command 6 : <c> E <c> E <c/r> .
Command 7 : <c> A <c> A <c/r> .
Command 8 : <c> E <c> A <c/r> .
Command 9 : <c> D <c> <c/r> .
Command 10 : <c> X s <c/r> .

As an example, the query 4 illustrated in Section 2 is expressed by means of the following command sequence :

```
<c> S 4 <c/r>                          ----  (1)
<c> N LU:USAG <c/r>                    ----  (2)
<c> T 6 <c/r>                          ----  (3)
<c> N LU:LAND <c/r>                    ----  (4)
<c> N LDA:LAND <c/r>                   ----  (5)
<c> N LYP:LAND <c/r>                   ----  (6)
<c> A <c> E <c/r>                      ----  (7)
```

(First, the carsor is pointed inside of the ellipse defined by (1) , next it is on the side of the innermost ellipse specified by (3).)

```
<c> X LYP:YEAR IS #1981 <c/r>          ----  (8)
<c> A <c> E <c/r>                      ----  (9)
```

(First, the carsor is pointed inside of the innermost ellipse specified by (3), next it is pointed the first argument of the expression specified by (8), i.e.":".)

```
<c> X LDA:AREA IS @GE #100 <c/r>       ----  (10)
<c> A <c> E <c/r>                      ----  (11)
```

(Similar to the command 9.)

```
<c> X ^AVG(UCOUNT(*)) <c/r>            ----  (12)
<c> A <c> E <c/r>                      ----  (13)
```

(Similar to the command 9.)

However, the order of commands is immaterial.