Data Pictures on the Desktop*

Ágnes Hernádi, Aladár Heppes and Előd Knuth

Computer and Automation Institute
Hungarian Academy of Sciences
Kende u. 13—17., H-1052 Budapest, Hungary

Abstract

An overview of the coDB database management system is presented, focusing on the system substratum and the facilities provided to build and manage data contents called Pictures. The experimentally implemented system provides an unusual database interface which makes multi-contextual data dialogues intersession-resident and responsive to appropriate changes of the database. Contextual access to the data provides an important degree of functional separation. The information base splits into two parts: the database and the so-termed Gallery, the repository of Pictures.

1. Introduction

Single user graphical workstations which provide a multiwindow environment are becoming more and more common. Whereas a plenty of application programs are offered which deal skilfully with data like Excel [1], Cardfile (SAPANA) or even HyperCard [2], none of these pursue real database functions.

The idea is appealing: a kind of visual "data object editor" having the power for performing all database functions formerly associated with entry forms, query specifications and pieces of the schema separately. The appeal is to the inexperienced end-user who manipulates heterogenous, ill-structured data and not to the professional database person.

The challenge facing us, then, is to provide a highly flexible user friendly man-machine interface facility which not only allows for concealing the traditional concepts of schemas, data definition languages and data manipulation languages but also provides a single unifying tool serving simultaneously various purposes for entering and updating data, query interpretation, report generation and schema manipulation.

* Lecture presented at the 1st Finnish—Hungarian Workshop on Programming Languages and Software Tools, Szeged, Hungary, August 8—11, 1989.

A database interface inspired by office-, management- and personal information systems recently coming into view (such as Hyper-systems, the remarkable phenomenon of Macintosh etc. [2], [3], [4]) has been experimentally developed at our institute for AT & T UNIX PCs. This interface facility [5], [6], [7], [8], [9,] [10], [11] supports simultaneous usage of multiple views or data contexts, moreover makes these contexts intersession resident, sensitive to the current alteration of the database and is the only tool for accessing the database. Last but not least this facility is easy to understand, to learn and its putting to use is quick, requiring no programming knowledge.

2. On the Data Model of the Experimental System

We have developed an experimental system called cooperative Databases (co DB) [12]. This relies on an ultimately simple data scheme. In order to allow us to keep our attention intently fixed on the problems of that interface facility we have chosen a completely unsophisticated and practicable data model. Namely

- We apply a binary relationship model [13] with no subtyping, however, relationships are non-directed many-to-many.
- Types and relationships are maintained automatically by entering their instantiation, and destroyed utterly on deleting their last instantiation. Instances of types are called atoms, and that of relationships are called connections.
- Two constituents are put together to form an atom that is to say a ⟨type⟩ class-description appointing the type to which the atom belongs and a ⟨value⟩:

  ⟨type⟩⟨value⟩.

  Both constituents are character strings although implementations may have restrictions laid on them. There are no arithmetical operations interpreted on values and for the time being we don't even plan to introduce them.
- A connection is an unordered pair of atoms belonging to a particular relation, so three constituents are put together to form a connection like a **⟨relname⟩** and an unordered pair (**⟨type 1⟩⟨value 1⟩**, **⟨type 2⟩⟨value 2⟩**). So it seems a relation is made up of a **⟨relname⟩** taking the place of role and an unordered pair of existing types (**⟨type 1⟩**, **⟨type 2⟩**).
  Relation names are — they may even be empty ones — character string objects which are unique for any given pair of types. Two binary relationships are considered identical if and only if all three of their corresponding constituents are identical (disregarding the order of types).
  A type might as well be related to itself, and an atom might be a constituent of any number of connections within a given relation. Two connections are considered to be identical if and only if all three of their corresponding constituents are identical (neglecting the order of atoms). Accordingly the same pair of atoms might be connected in as many relations as one could desire and still appearing once at most in a given relationship.

3. The Way of Displaying Data and Context

We believe it helps to view a binary relationship as a paragraph of two lines marked by the indentation of the second one.

Accordingly the first line displays an atom or a type and the second one the related atom or type preceded of course by the relation name if it is not actually an empty string. So a relation ⟨**relname**⟩(⟨**type1**⟩, ⟨**type2**⟩) or a connection ⟨**relname**⟩(⟨**type1**⟩⟨**value1**⟩, ⟨**type2**⟩⟨**value2**⟩) may appear in either form shown by Figure 1.

```
(i)     type 1:
          [relname] type 2:
    or
        type 2:
          [relname] type 1:

(ii)    type 1: value 1
          [relname] type 2: value 2
    or
        type 2: value 2
          [relname] type 1: value 1
```

Figure 1. Equivalent ways of displaying (i) a relation and (ii) a connection

As an example let us explore a database that records the main features of the twelve animal categories in the ancient Chinese lunar calendar. To represent this we select three types such as "*category*", "*in_the_cycle*" and "*year*", and two sorts of relations *R*1 for the relationship between "*category*", and "*in_the_cycle*", and *R*2 for the relationship between "*category*", and "*year*". Information about the Goat should appear in one of those forms in Figure 2, depending on our taste or purpose. We will see later, that any number of indentations are allowed.

Notice that no matter how we name *R*1 and *R*2 they remain redundant and even disturb us in understanding the represented connections. If both relation names were empty strings it would be quite similar to the traditional way of jotting. That's the reason why we allow relation names to be empty strings.

```
(i)     category: the Goat
          [R1] in_the_cycle: 8th
          [R2] year: 1907

(ii)    category: the Goat
          [R2] year: 1907
          [R1] in_the_cycle: 8th

(iii)   in_the_cycle: 8th
          [R1] category: the Goat
            [R2] year: 1907

(iv)    year: 1907
          [R2] category: the Goat
            [R1] in_the_cycle: 8 th
```

Figure 2. Alternative reflections of the same information

4. Pictures as the unified tools of interaction

A *picture* consists of an arbitrary number of hierarchically indented lines displaying atoms, types and relation names occasionally. More formally a picture is a forest of *picture lines* with the definition of

⟨picture line⟩ : = [⟨relname⟩:] ⟨type⟩: ⟨domain⟩
⟨domain⟩ : = BLANK | ⟨value⟩ | ⟨expression⟩

where **⟨expression⟩** is a selection criteria (e.g. a regular expression in terms of UNIX), and **BLANK** stands for *any* or *all* (no selection criteria).

In root lines no **⟨relname⟩** may appear. In non-root lines however theoretically a **⟨relname⟩** always appears at the very most it is empty (theoretically present but invisible). In a manner consistent with the above definition each picture line has a unique parent line unless it is a root line.

4.1. Validity and other characteristic properties of pictures

Informally speaking a picture is called *valid* if all the types, atoms, relations and connections referred to by any of its lines exist.

A valid picture is *filled* if each indented line in it possesses the following property: if its parent line contains a value, then it enumerates all the values connected to the atom displayed in its parent line by the named relationship.

A valid picture is *saturated* if each line in it which has at least one indented line, also has all possible indented lines in this very picture.

An empty picture trivially possesses all of these characteristic states.

5. Operations on Pictures

Any kind of user action can be carried out by using the appropriate operations.

5.1. Property Enforcing Transformations

To enforce picture properties and to carry out report generation we provide four transformations each of which acts on the whole picture.

VALIDATE

All the types, atoms, relations and connections appearing in the picture spring into existence if they have not existed in the database (see definition of valid picture).

FILL

The picture is to be converted into a filled one (see definition of filled picture). All the values fitting into a given place will be listed. In case of domain expression only values satisfying the expression will be included.

SATURATE

The picture is to be completed to become saturated (see definition of saturated picture).

EVALUATE

All feasible valid sequences satisfying every single domain specifications are to be determined.

Each value displayed on the picture is regarded as a restriction. Feasible pathes between two lines displaying atoms must fit on both ends. Reports are typically generated by this operation.

5.2. Operation modes

In order to reduce the number of *depicting* operations operation modes were introduced. These serve as distinctive marks of the particular classes of effects. There are three operation modes:

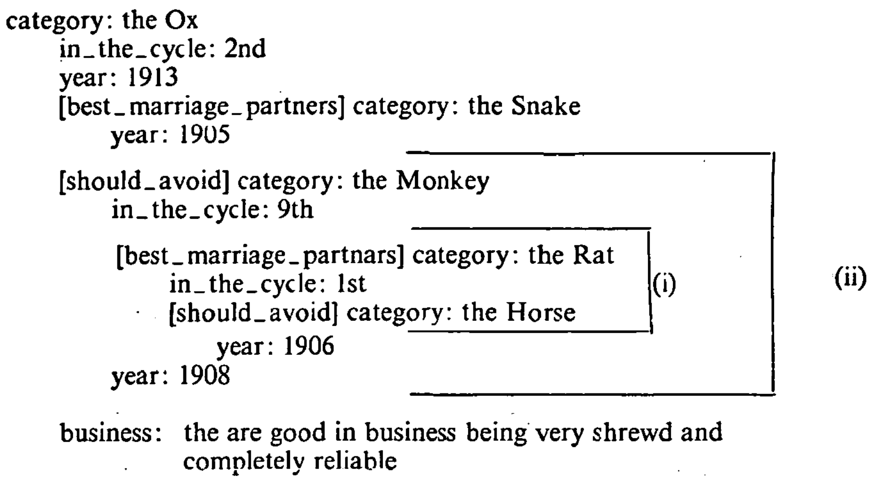
FREE	mode serves for temporal depicting with no effect on the database.
CHECK	mode is the default mode for all valid pictures. This mode serves the purpose to develop our view on data. Therefore in this mode no operation has update effect and operations violating the validity constraint are refused.
ENFORCE	mode provides the only way to alter the database. In this mode the VALIDATE transformation is called after each depicting operation the result of which violates the validity constraint.

The FREE→CHECK mode transition is refused whenever the picture is not valid. Other mode transitions are never refused. The FREE→ENFORCE mode transition is equivalent to a call of the picture state transformation VALIDATE, and as such must be confirmed.

5.3. Depicting Operations

These operations act on the selected part, that is on a subtree, a line or a token of a picture. They may or may not change the database itself depending on the current operation mode, however, according to the *What You See Is What You Get* paradigm no invisible change may occur. The whole interaction is supported with forms and icons requiring no syntactic knowledge of the user.

On selecting a subtree, we speak about a weak subtree if no restriction applies to the selection. But if the remainder must still constitute a picture, we speak about a strong subtree, see Figure 3.



```
category: the Ox
    in_the_cycle: 2nd
    year: 1913
    [best_marriage_partners] category: the Snake
        year: 1905
    [should_avoid] category: the Monkey
        in_the_cycle: 9th
        [best_marriage_partnars] category: the Rat
            in_the_cycle: 1st                          (i)          (ii)
            [should_avoid] category: the Horse
                year: 1906
        year: 1908
    business: the are good in business being very shrewd and
              completely reliable
```

Figure 3. Example of (i) a weak and (ii) a strong subtree

REMOVE (strong subtree) (option)

The selected subtree disappears recursively from the picture. In the ENFORCE mode data objects are to be deleted too.

Option	Actions taken
with root (default)	The whole selected subtree disappears. In the ENFORCE mode each atom included in the selected subtree is to be deleted with all their connections. Corresponding types and relations can be destroyed utterly. No other inductive effect is taken.
without root	If differs from the default option in that the root line of the subtree does not disappear. In the ENFORCE mode each atom displayed by the root lines of the disappearing part is to be disconnected from the atom displayed by the root line of the selected subtree. The corresponding relation can be destroyed utterly.
disconnect	The whole selected subtree disappears. In the ENFORCE mode the atom displayed by the root line of the selected subtree is to be disconnected from the atom in its visible parent line, if there is any. The corresponding relation can be destroyed utterly.

CLEAR (weak subtree)

All the domains in the selected subtree are to be made blank. Identical lines with no indented hierarchy are only to be displayed once. It never alters the database.

MOVE (strong subtree)

The selected subtree is to be moved. The subtree disappears from the source picture. This operation can be used in an inter-picture sense too. See PASTE for terminating a MOVE.

COPY (weak subtree)

The selected subtree is to be copied. The source picture remains unchanged. This operation can be used in an inter-picture sense too. See PASTE for terminating a COPY.

PASTE (line)(option)

This operation terminates a COPY or MOVE operation. The subtree to be copied or moved is inserted into the picture according to the option specified. In the ENFORCE mode database update requires confirmation.

Option	Actions taken
after (default)	The copied/moved subtree is to be inserted after the selected line (skipping of course all the lines marked by a longer indentation). The root line of this subtree is to be marked by the same indentation as the selected line.
before	The copied/moved subtree is to be inserted prior to the selected line, and its root line is to be marked by the same indentation as the selected line.
under	The copied/moved subtree is to be inserted immediately after the selected line. Its root line is to be marked by an indentation as compared to the selected line.

ADD LINE (line) (option)

A line is to be created and inserted into the picture according to the option specified. In the ENFORCE mode database update may occur.

Strings to be displayed in the inserted line should be entered through a form asking for them. Menus of tokens already entered into the database are available.

Option	Actions taken
after (default)	The created line marked by the same indentation as the selected one is to be inserted after the selected line (skipping of course all the lines marked by a longer indentation).
before	The created line marked by the same indentation as the selected one is to be inserted ahead of the selected line.
under	The created line marked by an indentation as compared to the selected one is to be inserted immediately after the selected line.

Suppose we have a picture including the portion of Figure 4. If we want to enter some other information about the Goat let us say the nature of those born in the Year of the Goat right after the line displaying it, then we have to select (mark) either the line displaying the name of this category and to ADD LINE under it, or the line displaying its serial number in the cycle of Twelve and to ADD LINE before it, see Figure 5.

```
...
category: the Goat
     in_the_cycle: 8th
     year: 1907
...
```

Figure 4. A portion of a picture

If we want to display another category let's say the Pig right after the information about the Goat, we can select the line displaying the name of this latter category and ADD LINE after it, see Figure 5.

```
...
category: the Goat
     nature: generous and shy, blessed with many
          virtues and just as many failings
     in_the_cycle: 8th
     year: 1807
category: the Pig
...
```

Figure 5. Result of the two ADD LINE operations on the picture in Figure 4

UNFOLD (line) (option 1, option 2)

Our view of information displayed in the selected line is to be opened out by revealing all the lines which could come up as indented ones according to the database's content. It never alters the database.

Option 1	Actions taken
natural only (default)	If the selected line displays an atom then all the atoms connected to it should be displayed. If however the selected line does not display an atom, all the relations interpreted on the type in the selected line should be displayed.
full	All the relations interpreted on the type in the selected line are to be involved neglecting whether they have any connections referring to the atom in the selected line.

Option 2	Actions taken
non-repeating (default)	The parent line of the selected one is not to be displayed between the indented lines.
repeat parent	Even the connection or relation between the selected line and its parent line is to be involved.

For the indented lines of the selected one which are already in the picture the following rules apply:

- Already existing lines will not be repeated;
- If the selected line displays an atom and the already existing indented one displays a blank domain then this blank domain will be filled with appropriate values instead of repeating this latter line;

- If the indented line contains a domain expression, it remains unchanged, and an other indented line will be inserted with the same relation and type names displaying values or not depending on the selected line.

```
...
category: the Goat
     nature: generous and shy, blessed with many
          virtues and just as many failings
     in_the_cycle: 8th
     year: 1907
category: the Pig
     in_the_cycle: 12th
     [should_avoid] category: the Snake
     [best_marriage_partners] category: the Rabbit
     year: 1911
            1923
            1935
...
```

Figure 6. After unfolding the line which displays the category "Pig" (with default options)

ADD VALUE (domain) (option)

A new atom of the type specified in the line of the selected domain is to be inserted. If the domain did not contain a value no option is offered. The required value will be inserted in that very domain, however, it must not contradict the selection criteria, if there is any specified.

If the selected domain already contains a value, the new value is to be inserted according to the option specified.

In the ENFORCE mode database update may occur.

The value to be displayed in the selected domain should be entered through a form asking for it. The menu of values already entered into the database are available.

Option	Actions taken
after (default)	The line containing the new atom will be placed right after the indented hierarchy of the line displaying the selected domain. The skeleton of the indented hierarchy of the selected line if there is such a hierarchy at all, will be inserted after the line containing the new atom. This skeleton contains all relation and type names but domains remain empty.
before	The line containing the new atom will be inserted above the line displaying the selected domain.

DELETE_VALUE (domain)

The selected domain must contain a value which is to be abandoned. In the ENFORCE mode the atom is to be deleted with all its connections. No type or relation can be destroyed, however.

EDIT EXPRESSION (domain)

A selection criteria is to be defined or modified. Editing the selected domain is refused if it contains a value (see EDIT TOKEN). No value may be specified (see ADD VALUE).

```
...
category: the Goat
     nature: generous and shy, blessed with many
          virtues and just as many failings
     in_the_cycle: 8th
     year: 1907
category: the Rooster
     nature:
     in_the_cycle:
     year:
category: the Pig
     in_the_cycle: 12th
     [should_avoid] category: the Snake
     [best_marriage_partners] category: the Rabbit
     year: 1911
           1923
           1935
...
```

Figure 7. Result of the ADD VALUE operation with an argument displaying the category "Goat" and the option after

EDIT TOKEN (token)

One of the strings displayed in the selected line is to be altered either in the picture containing the selected token (FREE mode), or in the database, and in all pictures displaying this very string (ENFORCE mode). This operation is unavailable in the CHECK mode.

Respectively either the relation or the type is to be renamed or the value is to be changed keeping all the connections. Editing the selected domain is refused if it is either empty (see ADD VALUE) or contains an expression (see EDIT EXPRESSION).

6. Galleries

Our pictures are stored in a special directory called Gallery which supports transactions dealing with pictures.

6.1. Picture Qualification

Pictures stored in a Gallery are qualified but their quality can be altered any time.

- A *Sketch* is a picture which need not be valid. Pictures to be depicted in FREE mode, or having become invalid are always requalified to this quality. This is the quality of created pictures too.

- A *Composition* is a valid picture which however can become invalid and requalified to Sketch as the database changes.
- A *Protected* picture may never turn invalid. Depicting operations in ENFORCE mode on any picture violating this restriction are refused.
- A *Master Piece* moreover may not be changed at all. Depicting operations in ENFORCE mode on any picture violating this restriction are refused.

Beside these there are two *standard*, read-only pictures in each Gallery. The picture *Types* contains all the existing types. The picture *Scheme* contains all the existing relationships. These pictures or any of their parts can be copied freely however.

6.2. Gallery Organization

A Gallery consists of two main parts

- the *Exhibition* in which all pictures are updated according to the EDIT TOKEN operations and checked up on being valid or not; and
- the *Archive* in which pictures are not maintained at all.

6.3. Transactions dealing with pictures

Each Gallery belongs to a single co DB. On opening the Gallery its Exhibition-menu is displayed. At request the Archive-menu is displayed too but in a separate window. From these menus the user can access the picture transactions namely:

- Create Picture
- Open Picture
- Delete Picture
- Copy/Move Picture
- Rename Picture
- Requalify Picture

All pictures have to be created except the standard ones. Opening a picture the operations on pictures are available. From an opened Gallery any number of pictures can be opened simultaneously. The other transactions work roughly in a way as can be expected.

7. Implementation issues

As we have already mentioned, co DB is experimentally developed for AT & T UNIX PCs. The implementation exploits

- the hierarchic file system of UNIX;
- the multiple window management capability supported by TAM routines; and
- the manipulation of abstract objects at the operation system's level provided by UA.

We manipulate two abstract objects at the operation system's level: the *co DB* database and the *Gallery*.

Commands assumed to be applicable to all ordinary abstract objects of this level such as create, open, close, delete, move, copy, rename are also defined on both of these objects.

Apart from the fact that each Gallery refers to a single co DB, any number of Galleries can be associated with a co DB. On opening a co DB the menu of Galleries associated with it is displayed.

8. Summary

We have expounded a new, non language oriented approach of interactive database interface in contrast to [15], [16], [17], [18] etc. This approach by matching modern requirements gives naive users demanding database supply altering dynamically an easy-to-use tool to interact directly with database.

We introduced the concept of picture which are user friendly abstract objects, having no fixed structure. Their content is transient, and they serve as a unified tool for accessing the database.

Our approach also provides a consolidated mechanism to draw computer aided comparison between independent databases containing diverse data, and to settle database communication protocols aiding interaction between them.

References

[1] Jones, E., Using Excel TM for the PC, (Osborne McGraw-Hill, Berkeley, California, 1988).

[2] Macintosh, HyperCard User's Giude (Apple Computer, Inc., 1988).

[3] Christie, B. (ed.), Human factors of the user-system interface, (North-Holland, Amsterdam, 1985).

[4] Shu, N. C., Visual Programming, (Van Nostrand Reinhold Company, New York, 1988).

[5] Hernádi, Á., Bodó, Z., Knuth, E., A different interactive interface for database management systems, Proceedings of the 11th Int'l Seminar on Database Management Systems, Seregelyes, Hungary (Oct. 3—7, 1988), pp. 85—94.

[6] Knuth, E., Vaina, A. M., Bodó, Z., Hernádi, Á., Beyond data crunching: A new approach to database interaction, Proceedings of the 3rd Austrian—Hungarian Informatics Conference on "Beyond number crunching", Retzhof, Austria (Sept. 14—16, 1988), pp. 91—104.

[7] Bodó, Z., Hernádi, Á., Knuth, E., Adatok mint "kepek", Proceedings of the 4th National Congress of the John von Neumann Society for Computing Sciences on "Application '89", Pecs, Hungary (March 28—Apr. 1, 1989), Vol. II, pp. 308—318 (in Hungarian).

[8] Hernádi, Á., Bodó, Z., Knuth, E., Context-reflecting pictures of a database, Proceedings of the EUUG Spring '89 Conference "UNIX: European Challenges", Brussels, Belgium (Apr. 3—7, 1989), pp. 273—282.

[9] Knuth, E., Hernádi, Á., Bodó, Z., Pictures at a Data Exhibition, in: Chang, S. K., (ed.), Visual Languages and Visual Programming (Plenum Publishing Company, in print).

[10] Knuth, E., Hernádi, Á., Bodó, Z., Gyenese, J., Data Gallery, Proceedings of the World Conference on Information Processing and Communication, Seoul, Korea (WOCON—INFOR 89, June 13—16, 1989).

[11] Knuth, E., Hernádi, Á., Heppes, A., Beech, D., Visual database management, Proceedings of the 4th IFIP Working Conference on User Interfaces, Napa Valley Lodge in Yountville, CA (Aug. 21—25, 1989, NORTH HOLLAND, in print.

[12] Bodo, Z. et al., Data Gallery — Common Base Definition, Reference manual Version 9. (Computer & Automation Institute, Hungarian Academy of Sciences, Budapest, December 1988.)

[13] Bracchi, G., Paolini, P. and Pelagatti, G., Binary Logical Associations in Data Modelling, in: Nijssen, G. M., (ed.), Modelling in Database Management Systems (North-Holland, Amsterdam, 1976).

[14] Vermeir, D. and Nijssen, G. M., A Procedure to Define the Object Type Structure of Conceptual Schema, Information Systems, Vol. 7 No. 4 (1982) pp. 329—336.

[15] Lans, R. F. van der, Introduction to SQL (Addison—Wesley, 1988).

[16] Hartman, P. A., R: BASE System V and 5000, (TAB Books Inc., USA, 1988).

[17] dBASE III User Manual (Copyright Ashton—Tate, 1984).

[18] INFORMIX (Registered trademark of Relational Database Systems Inc., USA).