Testing the direct manipulative interface concept of an "active join" with end-user oriented benchmark tasks

Matthias Rauterberg

Work and Organizational Psychology Unit
Swiss Federal Institute of Technology (ETH)
Nelkenstr. 11, CH-8092 Zürich, Switzerland

Abstract

The results of a survey among a group of users of a Data-Base-Management-System (DBMS) that runs on personal computers and the results of various experimental studies indicate that specific support is necessary for this group of users in the form of the interface concept of active joins. An active join, presented in a list on the interface, enables end-users to manipulate data records by direct manipulation. The definition of an active join, the generation of active joins by direct manipulation, and the usage of active joins are introduced herein. An experimental comparison between the desktop interface with and withoutout active joins demonstrates the superiority of the concept of active joins.

Introduction

Since the development of the desktop interface for the relational data base management system ADIMENS [GEISS87] 35,000 installations of this product have been sold. The customer of a data base system on a PC has several complex problems to solve: (a) s/he must define the data structure of the data base, (b) s/he must implement this definition, and (c) s/he must operate the data base system. These problems must be solved without specific knowledge about data base theory and without specific support through a human data base administrator. The end-user[1] is responsible for both the entity integrity and the referential integrity her/himself.

A survey was carried out by publishing a questionnaire in the newsletter of the ADIMENS user group. The results of 218 questionnaires completed by end-users show that the average data base structure of 153 reported data bases consists of six files (basis relations), with one or two link keys among these files [HUNZ90]. These empirical facts indicate, that most of the end-users, who reported on own data base structures, are able to solve the (a) definition problem as well as the (b) implementation problem. The complexity of the average data base structure suggests the need of joins by the end-users. Because there was no item in the questionnaire about the frequency of the use of joins, it is unclear how many joins were defined and in use. However, in an experimental investigation six qualified experts were required to define a join using a simple retrieval language (the syntax of the retrieval language is given in the Appendix) [RAUT89a] [RAUT89b]. None of these six experts was able to do this without helpful hints from the investigator. This implies that in the context of a desktop interface, end-users refuse to switch over to command language mode. Nevertheless, 10% of the above mentioned data bases have no link keys among data base files. This indicates that this group of users needs special support.

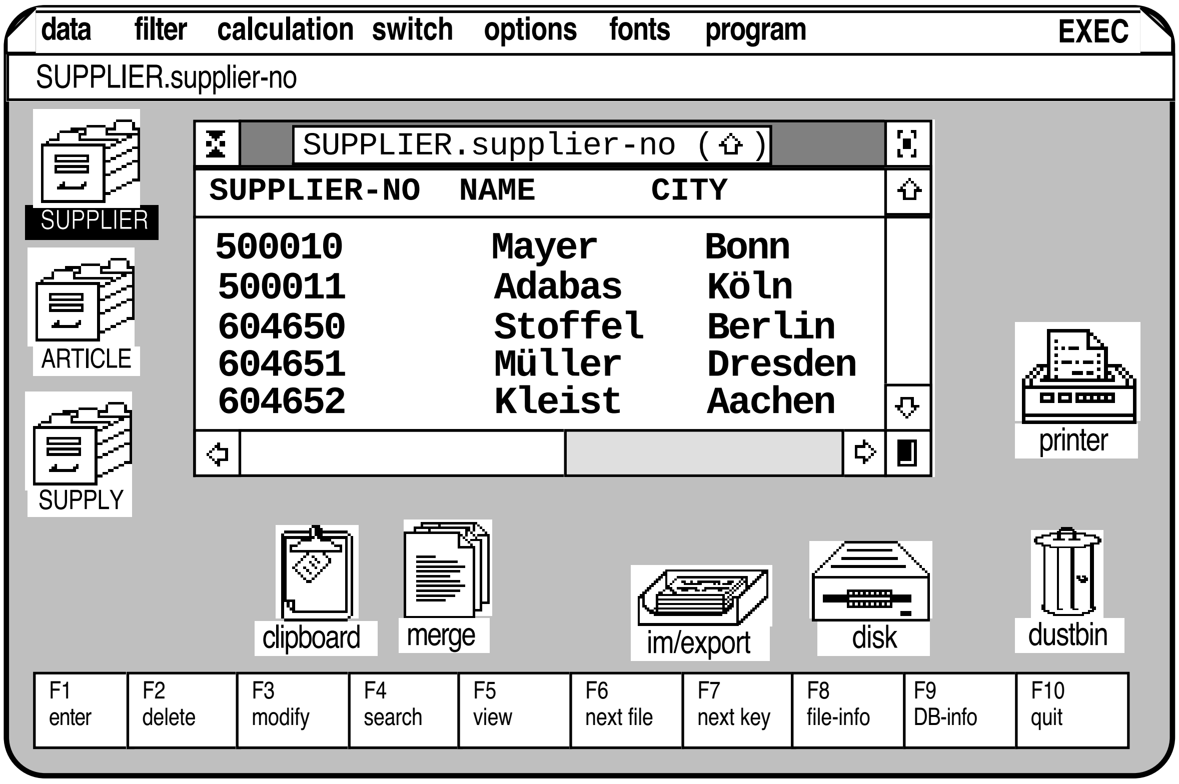
In the desktop interface of the ADIMENS system the data definition language (DDL) and the data manipulation language (DML) are partially realized with a direct manipulative dialogue structure. The data records of a file (basis relation) can be listed in a window (list mode; see Figure 1) or in a mask for each record (mask mode). Each row of the list in the list mode represents a single data record and can be handled with direct manipulative operations (e.g. copy, delete, modify, export, print, etc.).

The user has two different possibilities to find a single data record or a set of data records: the temporary *selection* operating on only one key attribute, or, the definition of a *filter* with logical conditions operating on any attributes. The selection and the filter is only definable for one data base file

1 The term "end-user" means a person, who works directly with the computer [ROCK83].

at once. To define more complex queries, the end-user has to use the retrieval language AdiTalk, which is similar to Dbase.

In the first version of the ADIMENS GT with the desktop interface, the generation of both tables, or lists, only with attributes of different basis relations (data base files) is done with a simple retrieval language (see Appendix). These tables, or lists, are only presented as output text in view, or browse, mode. The user has no possibility to operate on table entries in a direct manipulative fashion.



```
SUPPLIER-NO  NAME      CITY
500010       Mayer     Bonn
500011       Adabas    Köln
604650       Stoffel   Berlin
604651       Müller    Dresden
604652       Kleist    Aachen
```

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
enter	delete	modify	search	view	next file	next key	file-info	DB-info	quit

Figure 1: The desktop of the first version of the DBMS ADIMENS GT. The window in the middle of the desktop contains all five data records of the file named 'supplier'. The functions of all function keys of the keyboard (F1 .. F10) are shown at the bottom row of the desktop.

Different empirical studies were conducted to analyse the useability and efficiency of several aspects of the desktop interface for the first desktop version ADIMENS GT [RAUT89a] [RAUT89b] [RAUT90] [RAUT91]. One important interactive problem was observed: none of the investigated experienced users was able to define a join between two or more files without help. Furthermore, all of the users prefered the *list mode* (see Figure 1). Since there was no possibility for a temporary selection of data records in the list mode, the user had to define a global filter.

Also, in the first desktop version of ADIMENS GT (see Figure 1) all direct manipulative operations on data are possible only for data records of a single file, so that the basic idea of a relational DBMS could not be conceptualized adequately to support the end-user's mental model. To solve these interactive problems, a second version of the desktop interface has been developed and implemented. The new desktop interface ADIMENS GT+ can be characterized by the following features: the option of temporary selections in combination with the list mode and the option of definitions of virtual join files. The concept of virtual join files is based on equi-joins.

Different views of the data

There are three different views of the data structure of a data base: the *conceptual view* of all entities, the *internal view* of the physical organisation of the data records and the end-user's *external view* of parts of the data base [SCHLA83, p.26]. A PC end-user must be supported with specific interface features in all of these three views to provide a user oriented [SPINA90] and a task oriented [ULICH91] software design.

In most task contexts, end-users need a task specific combination of several attributes stored in different files. These combinations are commonly called joins. The possibility of direct manipulative operations on attributes combined in joins is an example of task oriented interface design [ULICH87] [ULICH89] [ULICH91] [DIN88].

1 The DBMS-administrator's view of the data

The administrator's view of the data encloses the conceptual view and the internal view, and is dominated by technical constraints: a minimum of redundancy and a maximum of consistency [DATE83]. These conditions can be realized in a relational DBMS by defining the appropriate relations. A simple example is a wholesale dealer who manages his orders with a relational DBMS. The exemplary data base is composed of three files: the article file, the supplier file and the supply file. These three files contain all relevant attributes to make an order, so the dealer needs a specific join.

2 The end-user's view of the data

The attributes that make up such a join can be handled in two different presentation modes, the list mode or the mask mode. In the mask mode, the user has full access to all attributes at once. For an overview of all data records in the virtual join file, users prefer the list mode (see Figure 2). In an experimental investigation six qualified experts were required to solve ten tasks [RAU89a] [RAU89b]. The total number of keystrokes to solve these tasks of all experts was 7311. The mask mode was used by the experts for only 21 keystrokes, indicating that they prefer the list mode.

If an end-user wants in a table, or list, defined by a (passive) join, s/he has to generate a merge document with an external text editor. In the desktop version of ADIMENS GT the merge document with the retrieval commands to define the above mentioned join 'order' looks as follows:

```
#QUANTITY#  #SUPPLIER-NO->SUPPLIER# #SUPPLIER-NAME# #CITY# #<-#
            #ARTICLE-NO->ARTICLE# #ARTICLE-NAME# #TYPE# #<-#
```

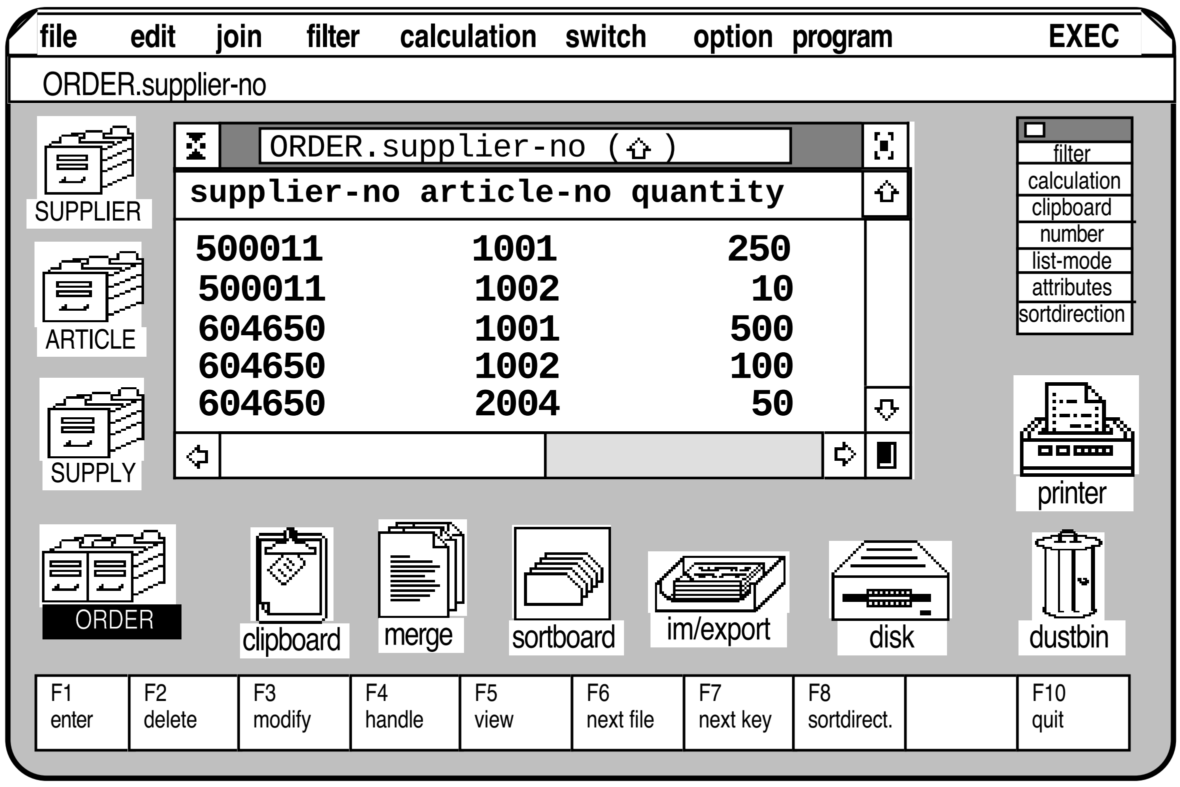
The user has to merge this document with the file 'supply' which contains the both linking keys "supplier-no" and "article-no" (the complete syntax is given in the Appendix).

In the specific task context of 'making an order' the wholesale dealer is not interested in knowing in which files the relevant attributes are stored. S/he needs all relevant attributes at once in one dialogue context. If the wholesale dealer wants to update one of the attributes, e.g. the attribute 'quantity', s/he then needs active access to this attribute in the context of all associated attributes. In the desktop version of ADIMENS GT users have only the possibilty to generate lists in a browse mode. This implies the necessity of *active joins*.

The concept of an "active" join

The idea of the concept of "active" joins is based on the psychological theory of task orientation and control [ULICH91]. Two prerequisites for the development of task orientation are: "(a) the individual should have control over the materials and processes of the task; and (b) the structural characteristics of the task (should) be such as to induce forces on the individual toward aiding its completion or continuation" [EMER78; p. 78]. Further on the two aspects of control are important: (1) "all those elements in which choice of how to do a job was left to the person doing it" and (2) "the extent to which an individual is free from intervention in the form of inspection and supervisory check-up" [EMER78; p. 79]. The concept of the complete task is discussed in detail somewhere else [see ULICH91].

Let us have a look towards our example. The simple task 'making an order' is characterised by the following feature: the dealer needs all information stored in three different files at once in one dialogue context to have task oriented control of his actions. He is really not interested in the informations stored in attributes like 'article-no' and 'supplier-no'. This kind of attributes is primary caused solely by technical reasons (the administrator's view!). An optimal task orientation would be given, if the DBMS could automatically manage attributes like 'article-no', etc. The end-user only needs to specify the different types of relations, which are relevant in the specific task context. Further on, a lot of task oriented research must be done to come to the right semantic of active joins, which can be implemented in DBMSs. A first step in this direction is described below.



```
supplier-no article-no quantity
500011         1001        250
500011         1002         10
604650         1001        500
604650         1002        100
604650         2004         50
```

F1	F2	F3	F4	F5	F6	F7	F8		F10
enter	delete	modify	handle	view	next file	next key	sortdirect.		quit

Figure 2: The desktop of the new version of the DBMS ADIMENS GT+. All virtual data records defined by the example in section 2.1 are listed in the window. The corresponding link file is named 'order'; this file is represented by a special icon with the combined card-index boxes. The functions of all function keys of the keyboard (F1 .. F10) are shown at the bottom of the desktop.

1 The definition of an "active" join

In the first version of the DBMS ADIMENS GT, the user could only take a look at the contents of attributes defined by joins using a simple retrieval language. The user had no possibility to change the attribute values in this passive browse mode. For this reason, this type of join is called a *passive join*.

In contrast an *active join* is defined as follows: the user has full access to all attribute values displayed on the output screen. This condition can be fulfilled in the context of a desktop interface in two different ways: 1. the attribute values can be directly manipulated in the output tables, or lists, or 2. the attribute values can be manipulated after transfer to a dialogue box. The second way is implemented in the new version of the desktop interface of ADIMENS GT+.

In the 'list mode' of the new ADIMENS GT+ version, the user has full access to all virtual data records of a join file. Each virtual data record is represented as a row in a table, or list. The complete list is mapped onto a desktop window (see Figure 2). The list is sorted by a selected key attribute, shown in the window header.

Each row of the desktop window shown in Figure 2 represents a virtual data record. The following direct manipulative operations can be applied to all virtual data records represented in a window: delete, print, export, multi-level sort, etc. In order to change the attribute values, the user must first of all transfer the row, or virtual data record, to a dialogue box. This operation can be done by activating all rows of interest and then starting the function 'modify'.

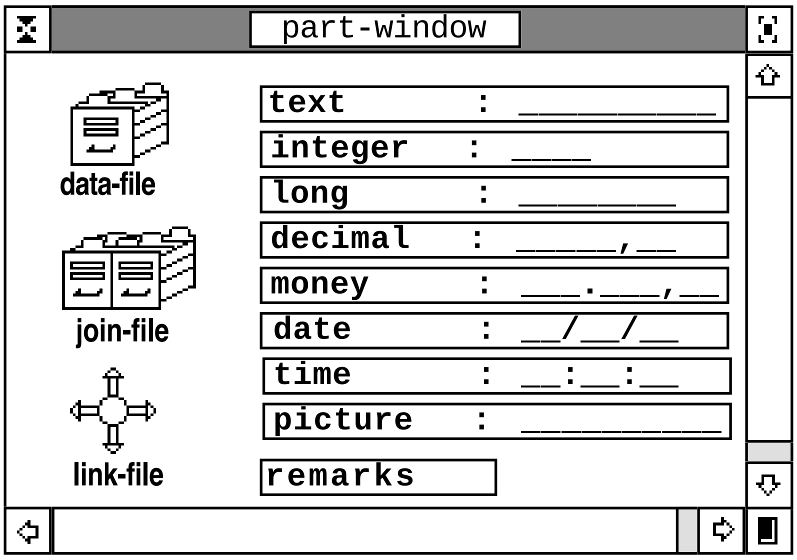


Figure 3: The "part window" (pw) contains the icons of the different file types (left column) and the icons of the different data types (right column). The user points to and activates the desired type with the mouse; then s/he moves a copy of the selected type into the working window area.

Now we will consider the direct manipulative method to define and create a data base with active joins. In the desktop version of ADIMENS GT+ a data base can consist of three different file types and eight different data types (as shown in Figure 3). One of these three file types is the join-file type. To create an active join the user has to change to the definition mode of ADIMENS GT+.

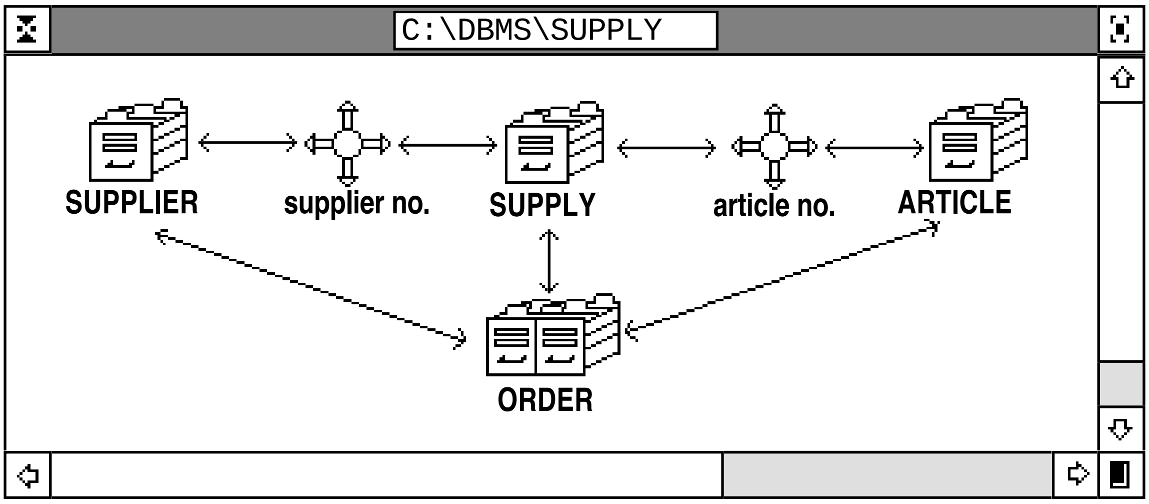


Figure 4: The "working window" of the file types (wwf) contains the icons of the three data files 'supplier', 'supply' and 'article', the join file "order" and the two link files "supplier no" and "article no" of the data base example "supply". The join file "order" contains attributes of the "supplier" file, the "supply" file and the "article" file. The link files "supplier no" and "article no" contain the corresponding linking key attributes. The link files are only relevant for end-users in the data base definition phase.

In the definition mode s/he copies all required objects (data file, join file, link file, and different types of attributes) from the part window (pw) (see Figure 3) into the working window and arranges them appropriately. There are two different types of working windows: the working window for the data, join and link files (wwf) (see Figure 4), and the working window for the different data types (wwd). An empty working window for the files (wwf) is automatically given.

After creating a file object in the wwf, the working window of the data types (wwd) can be activated by double click on the file icon in the wwf. Now, the user can create and arrange the different attributes of the active file by copying from the collection window into the wwd. The length of each data type is defined by a default value, which can be changed by the user immediately after creation.

The user can create and arrange all objects with direct manipulation. After defining all necessary attributes of all files the user needs only to activate the pull down menu item "DB generation", and the DBMS sets up the constructed data base automatically.

Once a join file is created including all attributes of only one or of several dependent data files, specific joins can be produced by defining projections. A projection is a selection of those attributes which are of special interest. The definition of a projection can be done interactively by pointing with the mouse to the required attributes. This dialogue mode can be activated by clicking on the menu item 'attribute selection' in the pull down menu 'switch' (see Figure 2).

2 The operations on "active" joins

At any time during operating a data base the end-user can define or modify active joins. This is no problem, because each definition of a join is based on attributes of data files. Only the data files (basis relations) define the structure of the data base. This feature of join files is the reason why we call join files 'virtual data files'.

Unsolved problems are the semantic of the functions 'enter' and 'delete'. To protect end-users from unnecessary loss of data, a sophisticated protection mechanism was implemented. Each data file can have the following four protection modes: enter, modify, delete, view. Each attribute of a data file can be protected from unallowed reading, writing, and modifying, too. These different protection modes of attributes are neccessary, to increase the safety of join operations. For example, it is dangerous to modify the value of the attribute 'article-no' or 'supplier-no' in the task 'making an order'.

The current implemented version of an active join is not powerful enough to allow end-users a semantic correct delete or enter operation. So, in all of these cases the end-user has to operate directly on the data files. Another restriction is the fact, that each active join is an equi-join. If an end-user is interested in other conditions of comparison, s/he must use the retrieval language AdiTalk.

To test the usability of our concept of active joins a user oriented benchmark test was done.

The user benchmark test of the "active" join concept[1]

In an experimental investigation, the two different desktop interfaces (ADIMENS GT and ADIMENS GT+) were compared. Two samples of students in computer sciences ('group-1': n = 16, and 'group-2': n = 15) tested *each* interface version of the DBMS ADIMENS. The first group started with the old version (GT) and then tested the new one (GT+), the second group tested in reversed sequence (GT+ -> GT).

The samples of users can be described as follows:

Group-1 (GT->GT+):

average age 24 (± 2)[2] years; 14 men, 2 women; 6. (± 2) semester computer sciences; 4138 (± 4022; range: 450-15.745) hrs. general EDP[3] previous experience, including 338 (± 515; range: 10-1.500) hrs. previous experience with DBM-systems [other than ADIMENS], and 3 (± 9; range: 0-35) hrs. with ADIMENS.

1 We thank Christian Rolfsen for his support conducting the experimental investigation.
2 (±xx) stands for "standard deviation" of the distribution.
3 EDP = Electronic Data Processing.

Group-2 (GT+->GT):

average age 24 (± 3) years; 14 men, 1 woman; 5. (± 2) semester computer sciences; 3706 (± 4562; range: 705-15.535) hrs. general EDP previous experience, including 67 (± 119; range: 0-350) hrs. previous experience with DBM-systems [other than ADIMENS], and 7 (± 21; range: 0-75) hrs. with ADIMENS.

There are no statistically relevant differences between group-1 and group-2, except the different amount of DBMS experiences (t-test, two tailed, df=30, $p \leq 0.05$). All users had to complete the following user benchmark task given in two parallel forms:

Task description for the interface without 'active joins' (ADIMENS GT):

"Create a list using the given retrieval language. This list should contain all values of the attribute 'quantity' of the file 'supply' and of the attribute 'article-name' of the file 'article'. Produce a adequate merge document with the text editor, merge the document with the file 'supply', and print the result."

Task description for the interface with 'active joins' (ADIMENS GT+):

"Create a list using the join file 'order'. This list should contain all values of the attribute 'quantity' and of the attribute 'article-name'. Define a adequate projection, and print the result."

The dependent variables were the time consumed ("test processing time" = TPT) and the number of keystrokes used ("number of keystrokes" = NOK) to complete the benchmark task. Each pressing of a key (or a mouse click) and a timestamp belonging to this keystroke was automatically recorded in a logfile. The investigator made sure that each test was completed.

Table 1: The results of the dependent variable "test processing time" = TPT.

TPT	User Group-1	User Group-2
1. task	[GT] interface 709 ±321 seconds	[GT+] interface 282 ±186 seconds
2. task	[GT+] interface 209 ±169 seconds	[GT] interface 542 ±288 seconds

Table 2: The results of the dependent variable "number of keystrokes" = NOK.

NOK	User Group-1	User Group-2
1. task	[GT] interface 154 ±38 keystrokes	[GT+] interface 25 ±20 keystrokes
2. task	[GT+] interface 32 ±19 keystrokes	[GT] interface 145 ±54 keystrokes

A variance analysis with the two factors "interface [GT vs. GT+]" ($F_{1,30} = 49.2$, $p \leq 0.001$), and "sequence [group-1 vs. group-2]" ($F_{1,30} = 0.43$, $p \leq 0.516$) was computed. The main effect of the "interface" is significant (see Table 3, second column). A variance analysis including the number of keystrokes as a covariate with the same two factors "interface" ($F_{1,30} = .001$, $p \leq 0.961$), and "sequence" ($F_{1,30} = 0.43$, $p \leq 0.516$) was also computed (see Table 3, third column).

The significant interaction between the factor "interface" and the factor "sequence" indicates, that the decrease of the test processing time can neither be explained exclusively by learning ($F_{1,30} = 4.87$, $p \leq 0.035$), nor by the amount of interactivity in combination with learning ($F_{1,30} = 3.88$, $p \leq 0.059$).

Table 3 Results of the two factorial variance analysis with the factor "interface" and the factor "sequence" for the dependent variable "test processing time" (variable TPT). The variable "number of keystrokes" (variable NOK) is used as a covariate in a second variance analysis.

dependent variable: Test Processing Time (TPT)	covariate: without			covariate: "number of keystrokes"		
source of variation	d f[1]	F	p	d f	F	p
"interface" [GT vs GT+]	1	49.24	.001	1	.001	.961
"sequence" [group-1 vs group-2]	1	.43	.516	1	.43	.516
"interface"⊗"sequence" interaction	1	4.87	.035	1	3.88	.059

Table 4: The mean of the dependent variable "test processing time" = TPT for the interface GT (without active joins) and GT+ (with active joins).

TPT	[GT] interface	[GT+] interface
global mean	628 ± 312 seconds	244 ± 178 seconds

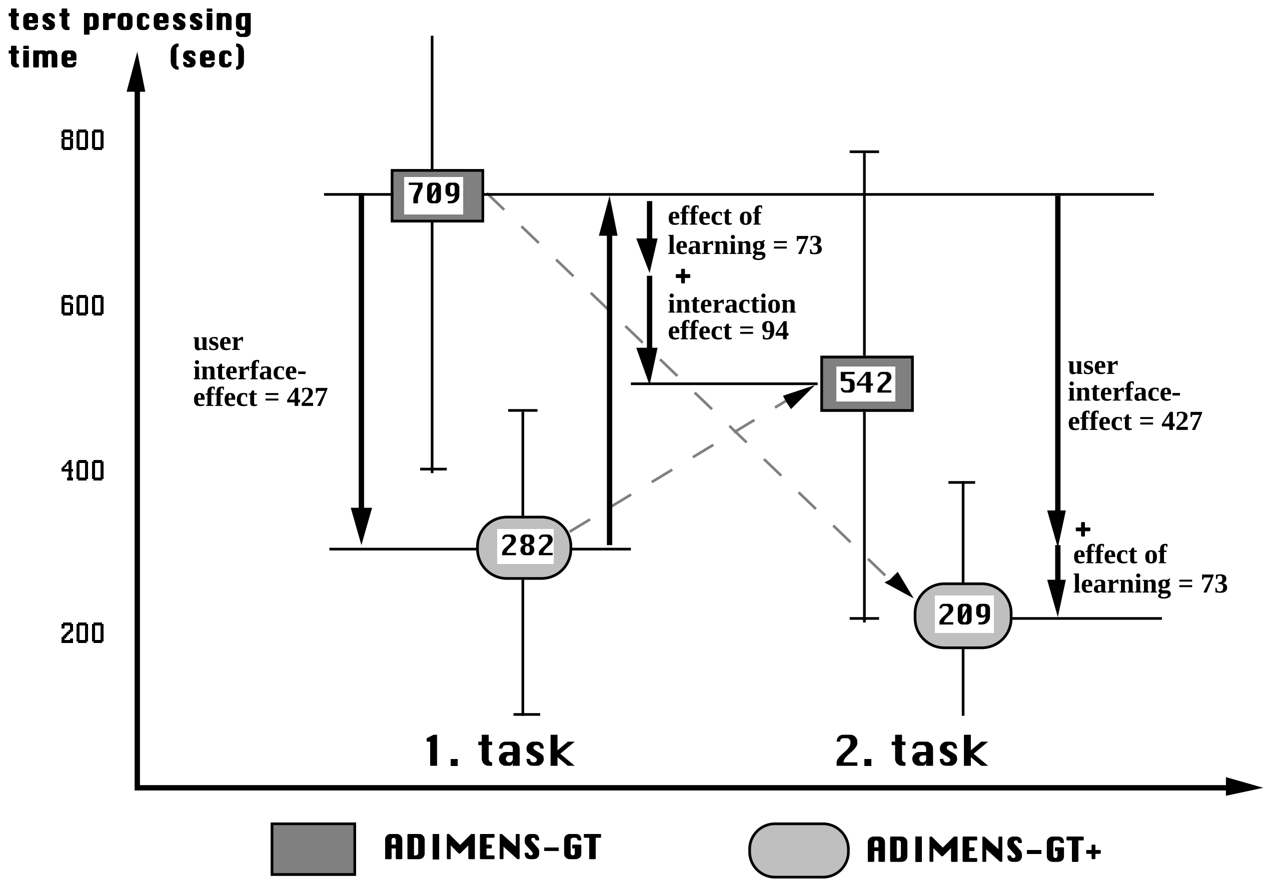


Figure 5: The diagram shows the results of the variable "test processing time" for the two different user interfaces and the two parallel benchmark tasks. The effects of the user interface, of learning, and of the interaction are illustrated.

1 df means "degree of freedom"; F is the F-test value; p stands for the significance level (the alpha-error term).

The test processing time working with the new interface (GT+; 244 sec. ± 178 sec.) was 39% of the test processing time working with the old interface (GT; 628 sec. ± 312 sec.). This decrease of test processing time can be completely explained by the different amount of "interactivity" measured by the variable "number of keystrokes". The users completed the benchmark task with the new GT+ interface much quicker than with the old GT interface (see Table 4).

The significant interaction effect (Table 3) can be illustrated by calculating the following three effects: the user interface effect ("UE") [UE = $TPT_{GT,1.task} - TPT_{GT+,1.task}$]; the effect of learning ("LE") [LE = $TPT_{GT,1.task} - TPT_{GT+,2.task}$.– UE]; the interaction effect ("IE") [IE = $TPT_{GT+,1.task}$ + UE – $TPT_{GT,2.task}$ – LE] (see Figure 5).

After finishing the benchmark test with both interfaces each user was asked for her/his opinion. The (GT->GT+) group-1 rated the question "The possibilities which are offered by the join files, are..." with a mean of 4.75 (± 0.45 standard deviation); (5 = very useful, 4 = useful, 3 = neither/nor, 2 = necessary, and 1 = not necessary). The (GT+->GT) group-2 rated the same question with 4.60 (± 0.51). The difference between both groups is not significant. This result means, that all end-users of this investigation prefer the concept of "active joins".

Conclusions

The dominance of technical aspects in conventional interface design led to insufficient usability of DBMS´s by end-users. Taking the user's task oriented view seriously leads to a useful interface concept: the *active* join. This interface concept was implemented in a relational DBMS and tested and compared with the old interface without the active join function. The empirical results show a significant superiority of the interface with the active join concept. Active joins can be implemented with other systems like SQL/Forms, too.

The superiority of the interface concept of active joins can be explained by the decrease of necessary number of keystrokes to solve the benchmark task. The more keystrokes a user needs to solve a task, the more possibilities the user has to make a mistake. This result is in accordance with the results of Margono and Shneiderman [MARG87].

The direct manipulative handling of active joins makes it possible for end-users to define their individual task oriented DBMS interface objects and their task oriented view of the data. There are still some unsolved problems: the task oriented semantic of the join operations 'enter' and 'delete'. Taking the user's view seriously will help solving these problems in the future.

Bibliography

DATE83 Date C.J.,
An introduction to database systems, vol. 1&2, Addison-Wesley 1983.

DIN88 DIN 66 234,
Bildschirmarbeitsplätze - Grundsätze ergonomischer Dialoggestaltung, DIN 66 234, part 8, Beuth 1988.
translated: "German Industrial Norm No. 66 234: Work places with display units - principles of dialogue design."

EMER78 Emery F.,
Characteristics of Socio-Technical Systems, Australian National University - Centre for Continuing Education 1978, (An abridged version of TIHR Doc. 527: The emergence of a new paradigm of work; Emery F., ed.; Canberra).

GEISS87 Geiss D.,
Ein endbenutzerorientiertes Datenbanksystem auf der grafischen Benutzerschnittstelle GEM, University of Karlsruhe Department of Computer Science, 1987.
translated: "An end-user oriented database system based on the Graphical Environment Manager GEM." unpublished Dissertation.

HUNZ90 HUNZIKER C. & HÄSSIG S.,
ADIMENS-Umfrage: Untersuchung von Datenbankstrukturen. unpublished technical

report WS'89/90, Swiss Federal Institute of Technology - Work and Organizational Psychology Unit 1990.
translated: "ADIMENS-survey: analysis of database structures."

MARG87 Margono S. & Shneiderman B.,
A study of file manipulation by novices using commands vs. direct manipulation, (26th Annual Technical Symposium Washington D.C. Chapter of the ACM Gaithersburg, MD, June 11, 1987; pp. 154-159).

RAUT89a Rauterberg M.,
MAUS versus FUNKTIONSTASTE: ein empirischer Vergleich einer desktop- mit einer ascii-orientierten Benutzungsoberfläche, Teubner 1989, (Proceedings of "Software-Ergonomie ´89"; Maass S. & Oberquelle H., eds; pp. 313-323).
translated: "Mouse versus Function key: an empirical comparison between a desktop- and an ascii-oriented user interface".

RAUT89b Rauterberg M.,
Ein empirischer Vergleich einer desktop- mit einer menü-orientierten Benutzungsoberfläche für ein relationales DBMS, Springer 1989, (Proceedings of the "GI-Jahrestagung 1989, vol I; Informatik Fachberichte Nr. 222"; Paul M., ed.; pp. 243-258).
translated: "An empirical comparison between a desktop- and an ascii-interface of a relational data base system".

RAUT90 Rauterberg M.,
Experimentelle Untersuchungen zur Gestaltung der Benutzungsoberfläche eines relationalen Datenbanksystems, Swiss Federal Institute of Technology - Work and Organizational Psychology Unit 1990, (Reports of the research project "User oriented software development and interface design", no. 3; Spinas P., Rauterberg M., Strohm O., Waeber D. & Ulich E. , eds.).
translated: "Experimental investigations for the interface design of a relational database system".

RAUT91 Rauterberg M.,
Benutzungsorientierte Benchmark-Tests: eine Methode zur Benutzerbeteiligung bei der Entwicklung von Standardsoftware, Swiss Federal Institute of Technology - Work and Organizational Psychology Unit 1991, (Reports of the research project "User oriented software development and interface design", no. 6; Spinas P., Rauterberg M., Strohm O., Waeber D. & Ulich E. , eds.).
translated: "User-oriented Benchmark-Tests: a method to participate end-users in the development of standard software".

SCHLA83 Schlageter G. & Stucky W.,
Datenbanksysteme - Konzepte und Modelle, Teubner 1983.
translated: "Database systems - concepts and models".

SPINA90 Spinas P., Waeber D. & Strohm O.,
Kriterien benutzerorientierter Dialoggestaltung und partizipative Softwareentwicklung - eine Literaturaufarbeitung, Swiss Federal Institute of Technology - Work and Organizational Psychology Unit 1990, (Reports of the research project "User oriented software development and interface design", no. 1, Spinas P., Rauterberg M., Strohm O., Waeber D. & Ulich E. , eds.).
translated: "Criteria of user oriented dialogue design and participative software development - an overview".

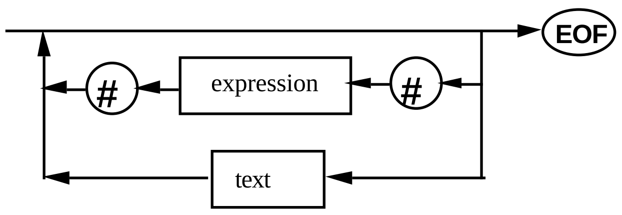
ULICH87 Ulich E.,
Some aspects of user-oriented dialogue design, North-Holland 1987, (System Design for Human Development and Productivity: Participation and Beyond; Fuchs-Kittowski K., Docherty P., Kolm P. & Mathiassen L., eds.).

ULICH89 Ulich E.,
Arbeitspsychologische Konzepte der Aufgabengestaltung, Teubner 1989, (Proceedings of "Software-Ergonomie ´89"; Maass S. & Oberquelle H., eds.; pp. 51-65).
translated: "Task design concepts of work psychology".

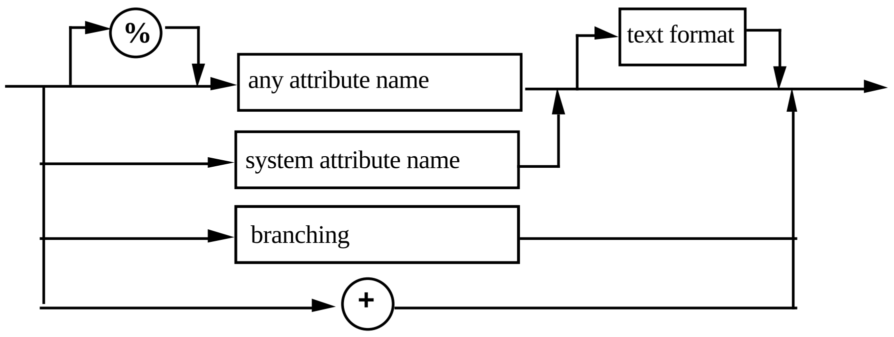
ULICH91 ULICH E, RAUTERBERG M, MOLL T, GREUTMANN T, STROHM O.,
Task Orientation and User-Oriented Dialog Design. International Journal of Human Computer Interaction, vol. 3, no. 2,.117-144.

Appendix: syntax diagram of the retrieval language

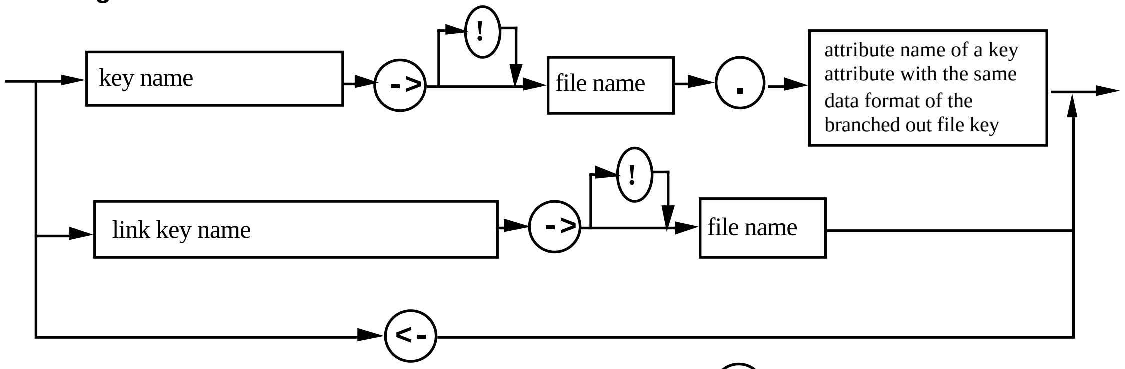
table:



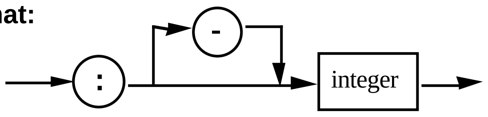
expression:



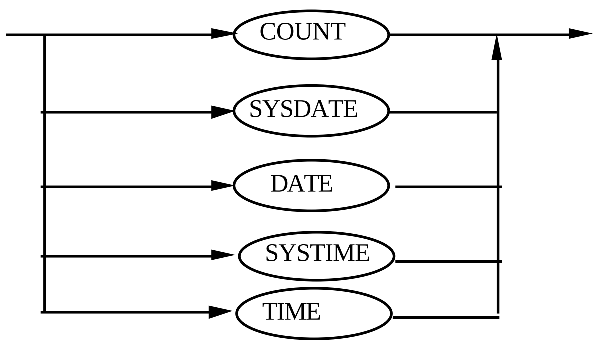
branching:


!
key name
->
file name
.
attribute name of a key attribute with the same data format of the branched out file key
!
link key name
->
file name
<-

text format:



system attribute name:



#	: left or right bracket
%	: output only if attribute value not empty
+	: summing up attribute values
->	: branching operator
!	: constrains the branching to exact one other record
-	: left justification
:	: filling up the output with blanks
.	: separation of file and attribute name

attribute name: { selection list of all attribute names of the active file}
or
{ selection list of all numbers of the attribute names of the active file, which are automatically generated by the DBMS}

LABORATOIRE
· DE · BASES ·
DE · DONNEES
EPF LAUSANNE

Proceedings of the

Database Research in Switzerland

Conference

September 30 – October 1, 1991

Lausanne, Switzerland

Edited by

Stefano Spaccapietra

1991