

On the Development of Table Oriented Programming with o++o

KLAUS BENECKE
beneckeSysteme,
Schröders Garten 5,
39175 Gerwisch,
GERMANY

Abstract: - The paper describes the essential steps in the so-far development of o++o. It started with some papers on the Relational data model, SQL, and CONVERT. Further, it was based on my PhD on a powerful algebraic specification language. The objects of our data model o++o will be described in detail. The operations of the data model were designed step by step, often redesigned, when other operations were introduced or changed. This holds also for the objects. The essential redefinition of tabments was influenced by XML and XQuery. It is demonstrated that a lot of problems can be formulated very compactly. The *next*-recursion is not as powerful as general recursive functions, but this kind of recursion seems to be easier to understand because all intermediated steps are gathered in a table. The bill of the material problem is solved in a new way with o++o-numbers and a slightly changed recursive operation *nextonr*. o++o allows querying and visualizing not only fact- but also (structured) text data in combination. Further, an example in a few lines, which requires in EXCEL more than 6 working sheets is presented. Then a very simple to use operation *cross* is introduced. It allows to creation of structured pivot tables. Small companies often don't like to use database systems, because of their high complexity in usage. Finally, an example is presented that a query on many files can be formulated also in a compact way.

Key-Words: - data model, tabment definition, structured table, structured document, mass data operations, BOM-problem, cross-operation, queries to Wikipedia, queries to many files.

Received: March 22, 2024. Revised: October 21, 2024. Accepted: November 20, 2024. Published: December 31, 2024.

1 Introduction

The paper describes the essential steps in the so-far development of o++o. It started in 1980 with some papers about the Relational data model, [1], [2], [3] SQL [4] and CONVERT [5]. Further, it was based on my PhD on the powerful algebraic specification language of [6] and [7]. [8] is the first motivational publication for a new query language. The content of this paper is contained in section 3. The objects of our data model are described in section 2. The operations of the data model were designed step by step, often redesigned, when other operations were introduced or changed. This holds also for the objects. The essential redefinition of tabments was influenced by XML [9] and XQuery [10] (section 2).

In section 4 it is demonstrated that a lot of problems can be formulated very compactly in one line simply as a term. The *next recursion* of section 5 is not as powerful as general recursive functions, but this kind seems to be easier to understand because all intermediated steps are gathered in a table, such that errors can be fixed more easily. The bill of the material problem is solved in a new way

with o++o-numbers – these are essentially section numbers – and a slightly changed recursive operation *nextonr*.

The next section 7 shows that o++o allows querying and visualizing not only fact- but also (structured) text data in combination. Section 8 presents an example in a few lines, which requires in EXCEL more than 6 working sheets.

In section 9 a very simple-to-use operation *cross* is introduced. It allows to creation of structured pivot tables. Small companies often don't like to use database systems, because of the high complexity in usage. In section 10 an example is presented that a query on many files can be formulated also in a compact way.

2 What is a Tabment?

2.1 What is a Relation?

Definition of a relation 1970 ([1], [2])

- A relation R is a subset of a Cartesian Product $\text{dom}_1 \times \text{dom}_2 \times \dots \times \text{dom}_n$
- Operations: permutation, selection, (nat-) join, projection

Weaknesses of Codd's Definition:

- no column names
- 2 is not an object of the relational model, but {2}
- Bags, lists and arrays, +, ... are outside the Relational model

Definition of a relation 1983 ([11])

$R = \{A_1, A_2, \dots, A_n\}$ (set of column names)
 $r(R) = \{f: R \rightarrow \text{Dom}: \text{Dom} = \text{dom}_1 \cup \text{dom}_2 \cup \dots \cup \text{dom}_n: f(A_i) \text{ in } \text{dom}_i\}$
 $r \times | s = \{t: R \cup S \rightarrow \text{dom}: t \downarrow R \in r \ \& \ t \downarrow S \in s\}$

This definition of a join is graceful but does not help end-users. Algorithms seem to be better for understanding operations.

2.2 What is a Structured Table?



Fig. 1: Zidsch-i Ulugh Beg, Samarkand

Is there a structured table in Figure 1?

Ulugh Beg wrote his Zidsch-i-Sultani from 1420 until 1437. It contains coordinates of 1018 stars. That seems to mean, it contains 1018 small tables. It is visible already on the open page that the mini-tables contain red and black headlines. One of the headline types is probably a constellation and the other is the name of the star. Therefore, in o++o notation the scheme of the whole list of stars could be $\text{CONSTELLATION}(\text{STAR}(\text{COORD}, \dots l) l)$ Here, l is a symbol for list.

o++o offers several representations for structured tables, but the one presented in this old book is not directly implemented. You need an additional formatting.

In the following, we present algebraic specifications in short, because they contain only the essential parts of a definition. If implementation-related "details" are omitted, then one can more easily design new operations. For example, the below scheme specification contains 4 generating operations, where the last two are recursive and the properties that the error-scheme *empty_s* is a neutral value for the *pair_s* operation and the *pair_s* operation is associative.

sorts Field, Coll_sym ...

sorts Scheme

opers empty_s -> Scheme

inj (Field) -> Scheme

coll (Coll_sym, Scheme) -> Scheme

pair_s (Scheme, Scheme) -> Scheme

axioms pair_s(s, empty_s) = pair_s(empty_s, s) = s
 pair_s(pair_s(s, s'), s'')
 = pair_s(s, pair_s(s', s''))

The sort Value is the disjoint union of words, texts, integers, floats, bools, and rational numbers. For the school, we introduced also a bar. The sort contains only one element (/). It is useful only in combination with lists. Then 4 can be represented for lower classes or preschool children for example by the list ///.

The below operation *add* is partial. It is defined only if the second argument is of the element type of the first. E.g., you can add a word to a set of words, but not a set of words to a set of words. The last axiom expresses: A second addition of an element to a set does not change the set, which results from adding the element once to the set. The last but one axiom expresses that the order of adding elements to sets and bags is not of importance. *iff* abbreviates *if and only if*. Empty is therefore a partial operation. It is defined only for collection schemes, e.g. Xl or $X, Y l$.

sorts Value (all elementary values)

sorts Table

opers abs_empty -> Table

empty (s:Scheme iff coll?(s)) -> Table

el_tab(Field, Value) -> Table

head (Table) -> Scheme

add (t1:Table, t2: Table

iff red(head(t1)) = head(t2)) -> Table

pair (Table, Table) -> Table

axioms head(abs_empty) = empty_s

head(ele_tab(f, v)) = inj(f)

```

if red(head(t1))=head(t2)
  then head(add(t1,t2))=head(t1)
head(pair_t(t1,t2))=pair_s(head(t1),head(t2))
pair(abs_empty,t)=pair(t,abs_empty)=t
pair(t1,pair(t2,t3))=pair(pair(t1,t2),t3)
if coll_type(head(t1))!=list &
  red(head(t1))=head(t2)=head(t3)
  then add(add(t1,t2),t3) = add(add(t1,t3),t2))
if coll_type(head(t1))= set
  then add(add(t1,t2),t2)) =add(t1,t2)
    
```

By this definition, 2 is yet not a structured table, but 2 is tagged by X, for example.

2.3 Algebraic Tabment Definition

Because of the rise of XML our table definition was modified. The introduction of column names is not restricted to values, now. By *tag0* a column name or better a tag can be made around an arbitrary table. *alternate* converts a tabment into a choice element. Therefore, especially documents can be handled by the definition, so we called the objects TABLEDOCUMENT.

We do not repeat the introduction of the operations empty_t, empty, add and pair. el_tab is now replaced by el_tab and tag0.

```

sorts Tabment
opers ...
  ele_tab Value    -> Tabment
  tag0(Field,Tabment) -> Tabment
  alternate(Scheme,Scheme,Tabment)
    -> Tabment
axioms ...
  head(tag0(f,t))=inj f
  head(ele_tab(v))=inj"TEXT" ,...,inj"BOOL"
  head(alternate(s1,s2,t))=alternate_s(s1,s2)
end
    
```

Now, 2 is a tabment, for example. An alternate is until now rarely used.

2.4 OCaml Tabment Definition

The below OCaml definition is more complicated than the above algebraic specification. It is a little more complicated than necessary from a purely logical point of view. Namely, it contains a subtype *tabtree*. This binary tree is introduced for sets and bags for quick access. Therefore, it is also possible to introduce indexes in RAM.

```

type tabment =
| Empty_t (* Tabment with empty head:error *)
| El_tab of value      (* elementary Value *)
| Tuple_t of tabment list (* n-Tuple *)
    
```

```

| Coll_t of coll_sym * scheme * tabcontainer
      (* collection *)
| Alternate_t of (scheme list) * tabment
| Tag0 of name * tabment
and tab_tree= (* binary tree for direct access *)
| Empty_set_btrees
| Node_set_btrees
      of tab_tree * tabment * tab_tree * int
and tabcontainer =
| Empty_c
| Single_c of tabment
| CList of clist
| CTree of ctree
and clist = { len: int; l: tabment list }
and ctree = { leng: int; typ: coll_sym; tr: tab_tree }
    
```

3 SQL Criticism 1982

In [8], SQL already 1982 was criticized from the point of view of structured tables. The below student file contains an ID (PKZ) and 2 repeating groups. For children data and foreign languages. This was some of the essential information at this time for GDR students. Here, *l* abbreviates list and *m* set (German: Menge).

```

STUDENT:PKZ,NAME, FIRSTNAME, LOC,
        GROUP, FAC, SEX,
        (CHILDDAT, CHILDCARD 1),
        (LANGUAGE, GRADE 1)m
    
```

For the first 2 queries, it is important to know that Alikendorf is a very small village and Magdeburg a big town, where the university is situated. Here is assumed that the user wants a structured output-table, if the output will be relatively large.

1. Find all students living in Alikendorf!

```

aus STUDENT
sel LOC=Alikendorf
gib NAME,PKZ,GROUP m
    
```

2. Find all students living in Magdeburg!

```

aus STUDENT
sel LOC=Magdeburg
gib FAC,(GROUP,(NAME,PKZ m)m)m
    
```

3. Find for *each* student of seminar group LM3/80 the set of languages with Grade 4!

```

aus STUDENT
sel GROUP="LM3/80"
sel LANGUAGE! GRADE=4
gib NAME,PKZ,LANGUAGEm m
    
```

The second condition does not select students, such that all students probably some with an empty set of languages will appear in the result. There is no need for the NULL of SQL. The next query is motivated by the fact, that you need two sub-queries with SQL, if you want to apply the 2 conditions *SEX=male* and *SEX=female*

4. Bring the data of girls and boys in different columns!

```
aus STUDENT
BOY :=NAME if SEX=male
GIRL:=NAME if SEX=female
gib GROUP,(BOY,PKZ m),(GIRL,PKZ m)m
```

Better?:

```
BOY?,GIRL?:=NAME, _ if SEX=male !
( _,NAME)
```

By the fifth example, it is demonstrated that even preschool children are able to understand and realize the basic algorithm behind *gib* on paper or on a blackboard.

5. Three examples of the stroke-list operation

Given: a "stream of cars":

```
Golf Han Polo Han Wartburg Atto
Golf Trabant Golf Han Polo
```

Count companies

```
BYD |||
IFA ||
VW ||||
```

Count models with company

```
BYD Atto |
BYD Han ||
IFA Trabant |
IFA Wartburg |
VW Golf ||
VW Polo ||
```

structured table

```
BYD Atto |
Han ||
IFA Trabant |
Wartburg |
VW Golf ||
Polo ||
```

Corresponding heads:

```
COMPANY,STROKE1 m
COMPANY,MODEL,STROKE1 m
COMPANY,(MODEL,STROKE1 m)m
```

The above algorithm counts and sorts simultaneously. Because the result of a query is in general relatively small, we have relatively small amounts of data to sort. If the result table is not very small, then the user can choose a structured output table (third example). If we assume that we have 10 companies and each company has 10 models, then we need on average not more than 5 company comparisons and 5 model comparisons. This seems to be an efficient sorting algorithm, not yet mentioned in [12].

4 One-line Programs

Average (+:) is an important aggregation. ++ (sum) stands for many plus signs. All basic aggregations are unary operations and written postfix. Binary operations like *rnd* (round) are written always infix. They are applied in the written order from left to right and top down.

Column names are assigned using the assignment symbol (:=). They can make comments superfluous. You cannot do without them for more complex problems.

avg.otto: Compute an average of some marks
AVG:=1 3 5 4 3 4 2 ++: rnd 2
Result (tab)
AVG
3.14

x .. y generates all numbers from *x* to *y*.

factorial.otto: Make a great number better readable
TWELF_FACTORIAL:=1 .. 12 ** '3
Result (tab)
TWELF_FACTORIAL
479'001'600

Grouping numbers by apostrophe is an idea from Switzerland. It has the advantage that there will be no confusion with existing concepts. There exist for example two versions of EXCEL. In one the comma is used as a separator for decimal numbers and the point (dot) to group the numbers and in the other, it is vice versa. This causes serious problems for international companies.

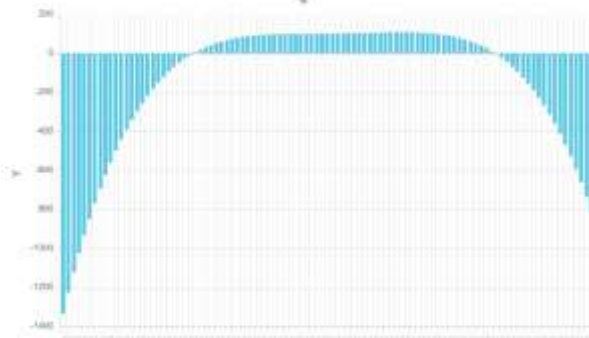
product.otto: Compute the product of 123 and 17 with matrix multiplication.
100 20 3 *mat (10,7) giball ZAHL1 ++
Result (tab)
ZAHL
2019

giball Xl corresponds to //X from XQuery resp. XPATH. We remark that

100 20 3 *mat (10,7) ++
 also computes the desired product.


percent.otto: Percent calculations			
NET	:=	200	
GROSS	:=	NET +% 19	
NET_WRONG	:=	GROSS -% 19	
NET_OK	:=	GROSS net 19	
Result (tab)			
NET	,	GROSS	,
NET_WRONG	,	NET_OK	
200	238.	192.78	200.

In a pocket calculator, the second line has to be typed in the following order: 200 + 19 %
 If we use brackets in both possible ways
 (200 + 19) %
 or
 200 +(19 %)
 than results 2.19, 200.19, respectively. Therefore, new operations have to be introduced.

local_max.otto: Compute a local maximum of the polynomial “-2*x^4 +2*x^3 +3*x^2+ 2* x +100” of degree 4	
MAX:= -5 ...5!0.0001 poly [-2 2 3 2 100] max	
Intermediate result: diagram (bars) of the following program: YI:=-5 ...5!0.1 poly [-2 2 3 2 100]	
	
Result (tab)	
MAX	
106.48568667	

The diagram shows that the interval borders are not the maximal points. That’s why the result has to be a local maximum.

The second line of the below program starts with more than 3 blanks. Because of this indentation, both lines compose one logical program line.

area.otto: Compute an approximation the blue area of the below diagram	
AREA:= -5 ...5!0.000'01 poly -2 2 3 2 100 abs sqrt *0.000'01 ++	
Diagram of the program: YI:= -5 ...5!0.1 poly [-2 2 3 2 100] abs sqrt	
	
Result (tab)	
AREA	
128.930466378	

The *-coll* operation is a "collection difference"-operation. It is applied to a list and a set, below.

sieve_erathostenes.otto: Compute all primes until 120	
PRIME1:=2 ..120 -coll (2 ..60 *mat (2 ..13 transpose) giball ZAHLm)	
Result (tabh)	
PRIME1	
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113	

5 Simple Recursive Assignments

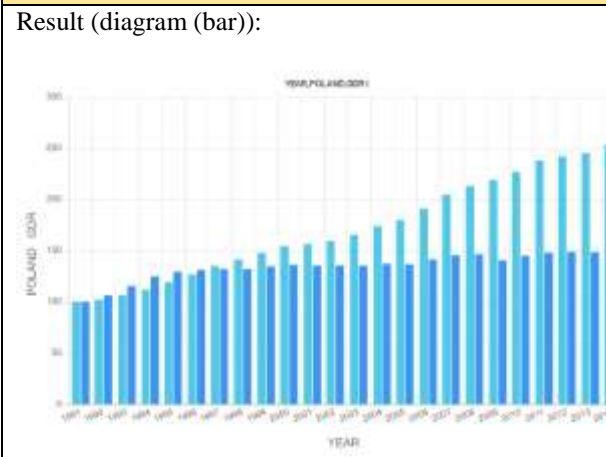
pred denotes the predecessor. By the first line, each of the numbers 2025 to 2055 is tagged by YEAR. *Next* can be considered as a binary operation. As already mentioned, binary operations are always written infix. Below, *preds* abbreviates *AMOUNT1 pred, AMOUNT11 pred*.

interests1and11.otto: Compare the development of an account within 30 years, with 1 and 11 percent interests (compounded yearly). Print only each fifth line.	
YEAR1:=2025 .. 2055 AMOUNT1,AMOUNT11:=100.,100. next preds +% (1,11) at	
YEAR rnd 2 sel YEAR rest 5 =0	
Result (tab)	
YEAR ,AMOUNT1 ,AMOUNT11 1	
2025 100.00 100.00	

2030	105.10	168.51
2035	110.46	283.94
2040	116.10	478.46
2045	122.02	806.23
2050	128.24	1358.55
2055	134.78	2289.23

```

gdp.poland_gdr.otto: Compare the GDP
development of Poland and East Germany for
the years, where both data exist.
INCREASE_POLAND1:= 0. 3.3 3.8 -7.2
-7.0 2.0 4.3 5.2 6.7 6.2 6.5 4.6
4.6 4.6 1.2 2.0 3.6 5.1
3.5 6.2 7.0 4.2 2.8 3.6
5.0 1.6 1.4 3.3 3.8 3.1
4.8 5.4 4.7 2.0 6.9 5.6
0.2
YEAR:=INCREASE_POLAND pos +1986
leftat INCREASE_POLAND
INCREASE_GDR1:=0. 1.85 -47.8 0. 6.2
8.7 8.1 3.5 1.6 0.5 0.2 1.8 1.2
-0.6 0.2 -0.3 1.3 -0.2 3.4 2.9
0.6 -3.9 3.2 1.9 0.6 -0.1 1.4
YEAR:=INCREASE_GDR pos +1987
join2
sel YEAR>1990
POLAND,GDR := 100.,100. next preds +%
(INCREASE_POLAND,INCREASE_GDR)
at YEAR
gib YEAR,POLAND,GDR 1
YEAR::= YEAR text
    
```

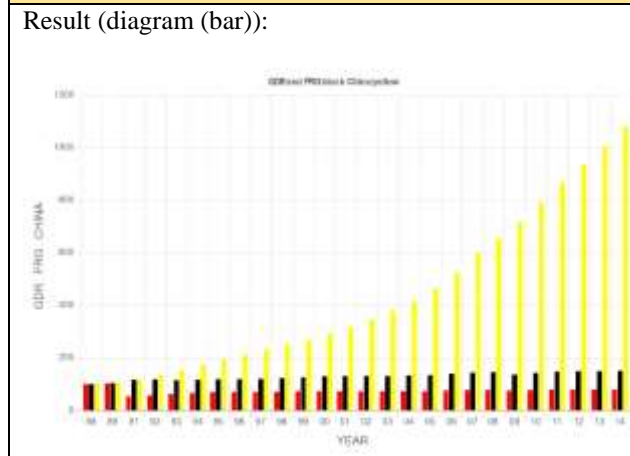


```

gdp_gdr_frg_china.otto: Compare the GDP of
East Germany, West Germany, and China from
1988 until 2014
<TAB!
YEAR, GDRINC,FRGINC,CHINAINC 1
1988 0. 0. 0.
1989 1.85 3.9 4.2
1991 -47.8 11.09 13.56
1992 6.2 1.7 14.3
    
```

```

1993 8.7 -2.6 13.9
1994 8.1 1.4 13.1
1995 3.5 1.4 11.
1996 1.6 0.6 9.9
1997 0.5 1.5 9.2
1998 0.2 2.3 7.8
1999 1.8 2.1 7.6
2000 1.2 3.1 8.4
2001 -0.6 1.1 8.3
2002 0.2 0.1 9.1
2003 -0.3 -0.1 10.
2004 1.3 1.6 10.1
2005 -0.2 0.8 11.3
2006 3.4 3.8 12.7
2007 2.9 3.3 14.2
2008 0.6 1. 9.6
2009 -3.9 -6.1 9.2
2010 3.2 4.3 10.6
2011 1.9 3.8 9.5
2012 0.6 0.4 7.7
2013 -0.1 0.1 7.7
2014 1.4 1.6 7.4
!TAB>
#sel YEAR>1991 # # line comment
GDR,FRG,CHINA := 100.,100.,100.
next preds +%
(GDRINC,FRGINC,CHINAINC)
at CHINAINC
TITEL:="GDR:red FRG:black
China:yellow"
gib TITEL,(YEAR,GDR,FRG,CHINA 1)
YEAR::= (YEAR text subtext 3!2)
rnd 1
RGB:=red leftat GDR
RGB:=black leftat FRG
RGB:=yellow leftat CHINA
    
```



6 The Bill of Material (BOM) Problem

BOM-problems occur often in industry. Surely, not only very large data sets have to be handled. Below, it can be seen that the whole BOM is stored in one structured table. Both collections of the input-table are sets. That means we have direct access to each tuple and sub-tuple, if the part or part number is given. In the first step, otto-numbers are generated. We shall see that these numbers are also important for structured texts like books or Wikipedia. The operation *nextonr* is similar to *next*, but it ends already if an *ottonr* of the same or smaller length follows. The rest is realized by *gib*.

bom.otto: Print the BOM of the car Wartburg.				
<TAB!				
PART,	PROPERTY,	(SUBPART,	COUNT m)	m
Bushing	cylindrical			
Engine	heavy	Piston	6	
		Screw	8	
Piston	light	Bushing	1	
		PistonRing	2	
Rim	smooth			
Trabant	modern	Body	1	
		Engine	1	
		Wheel	4	
Wartburg fast		Body	1	
		Climate	1	
		Engine	1	
		Wheel	4	
Wheel	round	Rim	1	
		Screw	5	
		Tire	1	
!TAB>				
onrs Wartburg				
COUNTOTTO:= COUNT nextonr				
COUNTOTTO pred *COUNT at COUNT				
gib SUBPART,TOTAL m TOTAL:= COUNTOTTO!++				
Result (tab)				
SUBPART,		TOTAL m		
Body		1		
Bushing		6		
Climate		1		
Engine		1		
Piston		6		
PistonRing		12		
Rim		4		
Screw		28		
Tire		4		
Wheel		4		
Intermediate result without last line (tab)				
PART,	PROPERTY,	(OTTONR,	SUBPART,	COUNT,
				COUNTOTTO
m)1				
Wartburg fast	1	Body	1	1
	2	Climate	1	1
	3	Engine	1	1
	3.1	Piston	6	6

3.1.1	Bushing	1	6
3.1.2	PistonRing	2	12
3.2	Screw	8	8
4	Wheel	4	4
4.1	Rim	1	4
4.2	Screw	5	20
4.3	Tire	1	4

7 Queries to German Wikipedia

The Wikipedia is now stored with the Relational DBMS MariaDB. Each entry is a CLOB (Character Long Object). Therefore, Wikipedia is from the point of view of the Relational DBMS simply of type

TITLE,CONTENT m

CONTENT has no structure, such that only very few queries can be formulated with SQL. In our approach, the text-data have the structure

TITLE, (ANR, ATITLE, CONTENT 1)m

Here ANR is the section number (an otto number) and ATITLE the corresponding title. Therefore, it is possible, for example, to use conditions of the following types

ANR=3.5.4

TITLE in [Bern Sofia]

Bern in ATITLE

[Bern Sofia] in CONTENT

...

To a Wikipedia. *X in Y* means each word out of *X* is also a word in *Y*. The FACT-data of Wikipedia do not have a simple scheme. It exists an INFOBOX for each object-type like river or town. In the below query *wiki_river_town.otto* it becomes evident that the Wikipedia was not designed for queries of the below kinds. The “Repeating groups” for towns on a river are divided in GROSSSTAEDTE (big towns) und MITTELSTAEDTE (middle towns) as comma-separated lists. If we introduce for example a repeating group STADTI instead of GROSSSTAEDTE and MITTELSTAEDTE, then the corresponding query can be simplified considerably.

The first queries below, give a first impression of the size of a mini-Wikipedia of Germany used here.

wiki_cnt.otto: How many entries are in RAM?	
wiki	
++1	
Result (tab)	
ZAHL	

60

```
wiki_titles.otto: Sort all titles.
wiki
gib TITELm
Result (tabh)
TITELm
Abraham_Lincoln      Acre      Al-Biruni
Alan_Turing           Albigenser
Alexander_der_Gro|fe Alicia_Silverstone
Alphabet      Altweibersommer      Amazonas
American_Standard_Code_for_Information_I
nterchange      Ampere      Angela_Merkel
Angelina_Jolie      Anglizismus      Ankara
Anna_Seghers      Anthony_Hope      Anthropologie
Antike      Apostilb      Ar_(Einheit)
Arbeit_(Sozialwissenschaften) Archimedes
Arch|ñologie      Ariel_Sharon      Aristoteles
Arthur_Harris
Arthur_Wellesley,_1._Duke_of_Wellington
Arzt Astronom Astronomie Atheismus Atom
Atomare_Masseneinheit
Auf|fenbandruptur_des_oberen_Sprunggelenk
es Bautzen Bydgoszcz Cottbus Donau Havel
Heidelberg Isar Krakau Mekong Neckar Nil
Nur-Sultan Oranienburg Prag Rathenow
Reutlingen Rhein Rom Sokrates Spree
Stuttgart      Warschau      Weichsel
|ångstr|Âm_(Einheit)
```

```
wiki_archimedes_Turing_summary.otto: Give
the first 300 letters of the summaries of
Archimedes and Turing.
wiki
sel TITEL in [Archimedes Alan_Turing]
sel ANR=0
gib TITEL,INHALT m
INHALT::=INHALT subtext 1!300
Result (xml)
<?xml version="1.0" encoding="ISO-8859-
1" ?>
<!DOCTYPE TABM [
<!ELEMENT TABM (TITEL,INHALT)*>
<!ELEMENT TITEL (#PCDATA)>
<!ELEMENT INHALT (#PCDATA)>]>
<TABM>
  <TITEL>Alan_Turing</TITEL>
  <INHALT>&apos;&apos;&apos;Alan
Mathison      Turing&apos;&apos;&apos;
OBE[[Ref]], [[Ref]]FRS [] (*&#160;23.
Juni 1912 in London; † 7. Juni 1954 in
Wilmslow, Cheshire) war ein britischer
Logiker, Mathematiker, Kryptoanalytiker
und Informatiker. Er gilt heute als
einer der einflussreichsten Theoretiker
der frühen Computerentwicklung und
Inform</INHALT>
  <TITEL>Archimedes</TITEL>
```

```
<INHALT>&apos;&apos;&apos;Archimedes
von      Syrakus&apos;&apos;&apos;
(griechisch Αρχιμήδης ὁ Συρακούσιος
&apos;&apos;&apos;Archimédēs ho
Syrakoúsios&apos;&apos;&apos;; * um 287 v.
Chr. vermutlich in Syrakus; † 212 v.
Chr. ebenda) war ein griechischer
Mathematiker, Physiker und Ingenieur. Er
gilt als einer der bedeutendsten
Mathematiker der Antike. Seine Werke
waren auch</INHALT>
</TABM>
```

The performance of the query can be significantly improved if the first selection is replaced by

```
keys [Archimedes Alan_Turing]
```

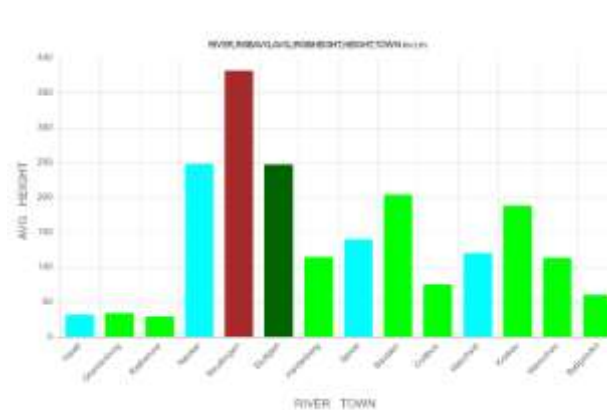
The keys-operation keys operation uses the binary tree access to sets, but it is an additional operation, which should be realized in the future invisible in the background.

```
wiki_archimedes_contents.otto: Print the
contents of the Archimedes entry.
wiki
sel TITEL=Archimedes
gib ANR,ATITEL 1
Result (tab)
ANR, ATITEL 1
0 Einleitung
1 Leben
2 Schriften
3 Werk
3.1 Physik
3.1.1 Hebelgesetz
3.1.2 Archimedisches Prinzip
3.2 Mathematik
3.2.1 Flächenberechnungen
3.2.2 Stellenwertbasiertes Zahlensystem
3.2.3 Archimedisches Axiom
3.2.4 Archimedische Körper
3.3 Technik
3.3.1 Archimedische Schraube
3.3.2 Kriegsmaschinen bei der Belagerung
von Syrakus
3.3.3 Brennspiegel
3.3.4 Weitere Erfindungen
4 Überlieferung
5 Sonstiges
6 Textausgaben
7 Übersetzungen
8 Literatur
8.1 list_element
8.2 list_element
...
8.22 list_element
9 Einzelnachweise
```


wiki_river_town.otto: Generate a structured diagram for given rivers with corresponding towns with sea level heights. Sort the rivers with average height by river and the towns of each river by height downwards. Choose yourself suitable colors.

```
wiki
keys [Neckar Havel Spree Weichsel]
TOWN1:=GROSSSTAEDTE split "," trim
      at GROSSSTAEDTE
TOWN1:=MITTELSTAEDTE split "," trim
      at MITTELSTAEDTE
rename TITEL! RIVER
gib RIVER,TOWNm m
=: $RIVER_TOWN
aus $RIVER_TOWN
gib TOWNm
=: $TOWNM
aus wiki
keys $TOWNM
gib TITEL,HOEHE m
rename TITEL!TOWN
HEIGHT:=HOEHE nthzahl 1
=: $TOWN_HEIGHT
aus $RIVER_TOWN
join $TOWN_HEIGHT
gib RIVER,AVG,(HEIGHT,TOWN m-) m
AVG:=HEIGHT!++:
rnd 0
RGBAVG :=cyan leftat AVG
RGBHEIGHT:=brown if HEIGHT>300 !
           darkgreen if HEIGHT>220 !
           green leftat HEIGHT
```

Result (diagram bar)



RIVER	,AVG	,(HEIGHT	,TOWN	m-) m
Havel	32.	34	Oranienburg	
		29	Rathenow	
Neckar	248.	382	Reutlingen	
		247	Stuttgart	
		114	Heidelberg	
Spree	140.	204	Bautzen	
		75	Cottbus	
Weichsel	120.	188	Krakau	
		113	Warschau	
		60	Bydgoszcz	

tt

=: \$XX

Assigns *tt* to the tabment variable *\$XX*. As a result of *aus* an otto-program starts again.

8 A BMI-Example - A Problem for EXCEL

bmi.otto: Visualize all the average BMIs of all persons "older than" 20.

```
<TAB!
NAME, LENGTH, (AGE, WEIGHT m)m
Klaus 1.68 18 61
      30 65
      61 80
Rolf 1.78 40 72
Kathi 1.70 18 55
      40 70
Walleri 1.00 3 16
Viktoria 1.61 13 51
Bert 1.72 18 66
      30 70
Peter 1.70 18 70
      60 100

!TAB>
sel NAME! AGE>20
BMI:=WEIGHT:LENGTH:LENGTH
gib AGE,(BMI,NAME m)m
AGE:= AGE text
totalhierar ++:
rnd 2
```

Result (diagram bar)



Result (tab)

BMAVG2	,(AGE	,BMAVG2	,(BMI	,NAME	m)m
24.38	18	21.79	19.03	Kathi	
			21.61	Klaus	
			22.31	Bert	
			24.22	Peter	
	30	23.35	23.03	Klaus	
			23.66	Bert	
	40	23.47	22.72	Rolf	
			24.22	Kathi	
	60	34.60	34.60	Peter	
	61	28.34	28.34	Klaus	

With EXCEL it is not possible to sort the given structured table immediately. You have to generate a table of type
 NAME,LENGTH,AGE,WEIGHT m

Then a selection “Give me all names for which an age greater than 20 exists” is not a property of one flat tuple. Further, you have to group by AGE to get the second BMI and then you have to merge the intermediate results and you have to hide the duplicates of the (AGE,BMIAVG2)- pairs and the BMIAVG.

	Nürnberg	35.0	26.8	26.8	68.8	90.0
	Stuttgart	35.0	26.8	26.8	68.8	90.0
	Würzburg	35.0	26.8	26.8	68.8	90.0
	Freiburg	40.9	38.7	38.7	72.0	96.0
	Konstanz	40.9	38.7	38.7	72.0	96.0
	München	43.2	34.2	34.2	72.7	92.3
	Passau	43.2	34.2	34.2	72.7	92.3
	min	30.5	19.3	19.3		
	avg	38.2	28.2		68.1	
	max	53.6	38.7			96.0
Kazakhstan	Almaty	89.0	66.0	65.0	87.0	102.0
	min	89.0	66.0	65.0		
	avg	89.0	66.0		87.0	
	max	89.0	66.0			102.0
min	min	30.5	19.3	19.3		
avg	avg	47.4	36.9		72.0	
max	max	89.0	71.0			106.0

9 Structured Pivot Tables

Now, a climate radiation table with towns of 3 countries are given. The radiation values are given for each of the 12 months. The ID contains in front of each town 8 additional letters. These 8 letters are omitted by the second program-line. An ID is a text consisting of letters, such that ID ++1 gives the number of characters of the ID. The last but one program line is added only to get a smaller output-table to fit into the column of this page.

radiation.otto: Generate a structured pivot table for all months and all countries for 3 aggregation types simultaneously.						
climate_radiation.tab						
ID:=ID subtext 9 ! (ID ++1 - 8)						
gib LAND,(ID,JAN,..,DEC l)m						
cross min,++:,max						
proj- FEB,..,NOV						
rnd 1						
Result (tab)						
LAND	,(ID	,JAN	,DEC	,MIN?	,AVG?	,MAX?
						1) 1
Bulgaria	Varna	63.0	59.0	59.0	80.2	100.0
	Shumen	59.0	57.0	57.0	80.1	98.0
	Ruse	72.0	57.0	57.0	88.1	106.0
	Burgas	64.0	55.0	55.0	79.9	99.0
	Plovdiv	88.0	71.0	67.0	83.9	98.0
	Sofia	52.0	40.0	40.0	71.6	89.0
	Haskovo	78.0	65.0	65.0	83.7	96.0
	min	52.0	40.0	40.0		
	avg	68.0	57.7		81.1	
	max	88.0	71.0			106.0
Germany	Potsdam	30.5	22.3	22.3	66.4	94.5
	Schwerin	30.5	22.3	22.3	66.4	94.5
	Teterow	30.5	22.3	22.3	66.4	94.5
	Dresden	34.2	22.3	22.3	66.4	93.0
	Essen	31.2	19.3	19.3	60.9	82.6
	Köln	31.2	19.3	19.3	60.9	82.6
	Münster	31.2	19.3	19.3	60.9	82.6
	Kassel	30.5	23.1	23.1	65.1	90.0
	Trier	30.5	23.1	23.1	65.1	90.0
	Chemnitz	51.3	36.5	36.5	73.3	96.0
	Leipzig	51.3	36.5	36.5	73.3	96.0
	Cham	53.6	35.7	35.7	70.5	86.3
	Hof	53.6	35.7	35.7	70.5	86.3

The below given table seems to represent grades of a gradebook in a compact way. If all subjects and names already exist in this table, no new subject and no new name need to be inserted when inserting a grade, even if the corresponding grade list was still empty. A hierarchical table of the type SUBJECT,(NAME,MARKl m) m contains each subject only once, but the names must be inserted several times. If this amount of data is stored in a flat table SUBJECT,NAME,MARK l, even more redundancy is created and even more memory space is wasted. In a flat set (relation of the relational data model), a time or position column would also have to be introduced, which makes the whole thing even more unwieldy and inefficient.

The columns and rows in the table above could also be swapped.

classbook.otto: Compute a pivot table for the averages of each pupil and each subject.						
<TABH!						
SUBJECT,ERNST1,CLARA1,SOPHIA1,ULRIKE1 1						
Math	2 3 2	1 1 2	1 3 2	1 2 3 3		
Phy	2 3	4 2	1 2 i	2 5		
			1 3			
German	1 1		3 5	4 4 1		
!TABH>						
cross ++:						
rnd 1						
Result (tab)						
SUBJECT	,ERNST1	,CLARA1	,SOPHIA1	,ULRIKE1	,AVG?	1
Math	2.3	1.3	2.0	2.3	2.0	
Phy	2.5	3.0	1.8	3.5	2.5	
German	1.0		4.0	3.0	2.7	
avg	2.0	2.0	2.3	2.8	2.3	

We remark that the operation ++: can be applied also to the list [1 2 i 1 3] of physics of Sophia, although it contains a non-numerical value i, which abbreviates ill.

10 Queries to Many Similar Files

The following query demonstrates an application for e-bills. Here, it is assumed that the user – for example the owner of a small company - stores each bill in a separate file. From o++o point of view it does not matter, whether all the bills are stored as XML-files or tab, or hsq-files, .. In the below example we consider only four bill-files, all of the same type and the master data for products and clients:

```
bill1,..,bill4: CLIENT,BILLNR,DATE,
                PAID,(PRODUCT,QUANTITY 1)
products: PRODUCT,VAT,PRICE1 m
clients: CLIENT,NAME,STREET,TOWN m
```

bill2.tab in hsq-format
CLIENT,BILLNR,DATE,PAID,(PRODUCT, QUANTITY 1)
CLIENT BILLNR DATE PAID PRODUCT QUANTITY
Senioren 36-21 11.12.2021 no Baguette 3 Roll 200 "Hemp bread" 6

+coll2 is again an abbreviation for +coll+coll. That means it is a unary union operation. It can also be replaced by *transpose*.

bills.otto: Compute all total prices of all bills.
bill1.tab,..,bill4.tab +coll2 join products.tab join clients.tab PRICE:=QUANTITY*PRICE1 +% VAT gib BILLNR,NAME,TOWN,(PRODUCT,PRICE m)m total ++ rnd 2
Result (hsq)
BILLNR,NAME,TOWN,(PRODUCT,PRICE m) 1 BILLNR NAME TOWN PRODUCT PRICE
33-21 "Seniorendomi MD" "39175 Gerwisch" Baguette 11.77 Roll 618.46 sum 630.23 36-21 "Seniorendomi MD" "39175 Gerwisch" Baguette 7.06 "Hemp bread" 22.47 Roll 72.76 sum 102.29 44-21 Backschwein-Tanne "14822 Brück" Baguette 7.06 Roll 72.76 sum 79.82 45-21 Backschwein-Tanne "14822 Brück" Baguette 7.06 "Hemp bread" 22.47

Roll 327.42 "Rye loaf bread" 22.26 sum 379.21 sum sum sum sum 1191.55

11 Conclusions and Future Work

It remains a lot of work. o++o has to be tested in schools and applications, documentations have to be extended and improved, o++o has to be put on top of Relational and NoSQL database systems. Here, some optimization techniques are needed. ...

Acknowledgment:

I thank Heinz Kaphengst and Horst Reichel for the development of the algebraic specification language for partial operations and Rüdiger Achilles for valuable remarks to this paper. Further, thanks are directed to the following computer experts for their valuable contributions to the older implementations of the o++o system: Wolfgang Reichstein, Dmitri Schamschurko, Martin Schnabel, Andreas Hauptmann. Stephan Schenkl and Eicke Redweik worked for the project from 2019-2021 "Intelligent Analysis and Visualization of German Wikipedia", which was funded by EFRE EU and IB Sachsen-Anhalt.

References:

- [1] E. F. Codd, „A Relational Model of Data for Large Shared Data Banks“, *Communications of the ACM*, Vol. 13, No. 6 June 1970, S. 377-387.
- [2] E. F. Codd, „*Relational Completeness of Database Sublanguages*“, in R. Rustin (ed.), *Database Systems*, Prentice- Hall, Englewood Cliffs, N.J., 1972
- [3] E. F. Codd, „Extending the Database Relational Model to capture more Meaning“, *ACM Transactions on Database Systems*, Vol. 4, No 4, Dec. 1979, S. 379-434.
- [4] M. M. Astrahan, D. D. Chamberlain, "Implementation of a Structured English Query Language", *Communications of the ACM* 18 10, Oct. 1975 pp. 580-587.
- [5] N. C. Shu, B. C. Housel, V. Y. Lum, „CONVERT- A High Level Translation Definition Language for Data Conversion“, *Communications of the ACM*, Vol.18, Nr.10,Oct., 1975, pp. 557-567
- [6] H. Kaphengst, H. Reichel, *Algebraic Theory of Algorithms* (Algebraische

Algorithmentheorie), VEB Robotron, Wiss. Informationen und Berichte, Nr. 1/71 Reihe A, Sommer 1971.

- [7] H. Reichel "*Initial Computability, Algebraic Specifications, and Partial Algebras*", Claredon Press, Oxford, UK, 1987.
- [8] K. Benecke. "AFUL- A query language for databases". In Weiterbildungszentrum für mathematische Kybernetik und Rechentechnik/Informationsverarbeitung, Studientexte Datenbanken TU Dresden Heft 59, pp. 100 - 107, 1982.
- [9] D. Chamberlain et al. "XML Query Use Cases", [Online]. <http://www.w3.org/TR/xmlquery-use-cases>, 2007 (Accessed Date: December 10, 2024).
- [10] W3C, "XQuery 3.1: An XML Query Language", W3C Recommendation 21 March 2017, Status Update (6 April 2021), [Online]. <https://www.w3.org/TR/xquery-31/> (Accessed Date: December 10, 2024).
- [11] D. Maier, "*The Theory of Relational Databases*", Computer Science Press, Financial Times Prentice Hall, ISBN-10: 0914894420, ISBN-13: 978-0914894421, 1983.
- [12] Sorting Algorithm, [Online]. https://en.wikipedia.org/wiki/Sorting_algorithm (Accessed Date: December 10, 2024).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

There is only one author, therefore authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflicts of interest to declare.

Copyright Note

A commercial use of the paper and its ideas is not allowed.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0

https://creativecommons.org/licenses/by/4.0/deed.en_US