Die wichtigsten Schlüsselworte von o++o

(Stand 15.05.2018)

Zunächst sollen zwei Beispieltabellen in verschiedenen Formen dargestellt werden:

1. Beispiel: Eine flache (einfache) Tabelle mit 3 Zeilen und 2 Spalten:

```
TAB! Tabellarische Darstellung:
<TAB!
FACH,   NOTE l
Mathe   1
Chemie  2
Deutsch 2
!TAB>

HSQ! Hierarchisch SeQuentielle Repräsentation (2 Leerzeichen
zwischen den Feldern)
<HSQ!
FACH,NOTE l
Mathe  1
Chemie  2
Deutsch  2
!HSQ>

Ment! Dokumentdarstellung:
<META!
TABMENT! TABM
TABM! FACH,NOTE l
NOTE! ZAHL
FACH! WORT
!META>
<TABM>
  <FACH>Mathe</FACH>
  <NOTE>1</NOTE>
  <FACH>Chemie</FACH>
  <NOTE>2</NOTE>
  <FACH>Deutsch</FACH>
  <NOTE>2</NOTE>
</TABM>

Einzeilige Darstellung (in o++o Programmen sinnvoll.)
[FACH,NOTE! Mathe 1  Chemie 2 Deutsch 2]
```

2. Beispiel: Datei ottos.tab mit 3 Tupeln (strukturierte Sätze) mit je einer Wiederholgruppe

```
NAME,                 GEBORENIN,    (TAT,                                  JAHR l)l
Nicolaus Otto         Taunus (De)    Miterfinder des Ottomotors            1876
Otto Normalverbraucher De            erlernt Autofahren                    1960
                                     erlernt eine Programmiersprache       2020
Otto der Grosse       Altsachsen(De) Zum Koenig von Deutschland gewaehlt     936
                                     Die Ungarn auf dem Lechfeld geschlagen  955
                                     Erster Kaiser des Heil. Roem. Reichs    962
```

```
<HSQ!
NAME,GEBORENIN,(TAT,JAHR l)l
Nicolaus Otto  Taunus (De)
 Miterfinder des Ottomotors  1876
Otto Normalverbraucher  De
 erlernt Autofahren  1960
 erlernt eine Programmiersprache  2020
Otto der Grosse  Altsachsen(De)
 Zum Koenig von Deutschland gewaehlt  936
 Die Ungarn auf dem Lechfeld geschlagen  955
 Erster Kaiser des Heil. Roem. Reichs  962
!HSQ>

<META!
TABMENT! OTTOS
OTTOS! NAME,GEBORENIN,(TAT,JAHR l)l
JAHR! ZAHL
GEBORENIN,NAME,TAT! TEXT
!META>
<OTTOS>
  <NAME>Nicolaus Otto</NAME>
  <GEBORENIN>Taunus (De)</GEBORENIN>
  <TAT>Miterfinder des Ottomotors</TAT>
  <JAHR>1876</JAHR>
  <NAME>Otto Normalverbraucher</NAME>
  <GEBORENIN>De</GEBORENIN>
  <TAT>erlernt Autofahren</TAT>
  <JAHR>1960</JAHR>
  <TAT>erlernt eine Programmiersprache</TAT>
  <JAHR>2020</JAHR>
  <NAME>Otto der Grosse</NAME>
  <GEBORENIN>Altsachsen(De)</GEBORENIN>
  <TAT>Zum Koenig von Deutschland gewaehlt</TAT>
  <JAHR>936</JAHR>
  <TAT>Die Ungarn auf dem Lechfeld geschlagen</TAT>
  <JAHR>955</JAHR>
  <TAT>Erster Kaiser des Heil. Roem. Reichs</TAT>
  <JAHR>962</JAHR>
</OTTOS>
```

SCHLÜSSELWORT	KATEGORIE	BESCHREIBUNG
+	binäre algebraische Operation	**Addition**: 3+6=9 (2,3,Paul)+4=(6,7,Paul)
*	binäre algebraische Operation	**Multiplikation**: (2,3,Paul)*4=(8,12,Paul); tab*z1: jede Zahl von tab wird mit z1 multipliziert; der Rest bleibt unverändert tab*si=tab*1;tab*no=tab*0; tab1*tab2 sonst: beide Tabmente müssen den gleichen Typ haben
-	binäre algebraische Operation	**Subtraktion**: 7-3 = 4: Differenz
:	binäre algebraische Operation	**Division**: 6 : 4 = 1.5
++	unäre algebraische Operation	**Summe**: [1 2 3] ++ ergibt 6 1, Wladimir, 8 ++ ergibt 9
**	unäre algebraische Operation	**Produkt**: [2 3 7]** = 42
--	unäre algebraische Operation	**Mehrfachsubtraktion**: [55 3 4] -- = 48
::	unäre algebraische Operation Aggregation	**Mehrfachdivisionen**: [64 2 2 2] :: = 8

++1	unäre algebraische Operation Aggregation	Anzahl (Statistik): `[5 6] ++1 =2`; Anzahlen der (komplexen) Zeilen jeder Komponente;
		`[2 3 5 7 8 9], {rr tt zz uu},99`
		`++1`
		`= (6,4,1)`

+m, +b, +l	binäre Kollektionsoperation	mengentheoretische Vereinigung, es können aber auch Elemente zur Kollektion hinzugefügt werden, …., das Kollektionssymbol gibt den Zieltyp an

-m, -b, -l	binäre Kollektionsoperation	**Mengendifferenz**: Differenz von flachen Kollektionen
		`<TAB!`
		`X,Y 1`
		`1 2`
		`3 4`
		`!TAB>`

		`-l ##`
		`<TAB!`
		`X,Y l`
		`3 4`
		`1 5`
		`!TAB>`
		`=`
		`X, Y l`
		`1  2`

:m, :b, :l	binäre Kollektionsoperation	mengentheoretischer Durchschnitt,….
*m, *b, *l	binäre Kollektionsoperation	kartesisches Produkt
,	binäre Tabmentoperation	**Paarbildung:** 1,6,(otto,9) = 1,6,otto,9 (4-Tupel) 1,otto =(1,otto) ist vom (Typ ZAHL,WORT) 1,2 ungleich 1.2
;	Basissymbol	**Semikolon:** kann vor einem binären Operator stehen; das zweite Argument reicht dann bis zum nächsten Semikolon bzw. bis zum Zeilenende: 1+2 ;* 3+4 ;*2+3 = 105
„ “ bzw. „ “	Basiszeichen	[1 2 4] = Liste von 3 Zahlen trennt auch Werte und Operationen; Eine Einrückung (vier Leerzeichen) verbindet neue Zeile mit der vorangehenden zur logischen Einheit
=	Relationssymbol	ORT = Hadmersleben selektiert beispielsweise alle Personen, die in Hadmersleben wohnen

:=	Basissymbol	**Zuweisung**:
		`BRUTTO:=NETTO + TARA`
		`neuname := ausdruck`
		(neuname wird mit dem Resultat der Berechnung von ausdruck belegt); Aggregationszuweisungen kommen auch in der gib-Klausel vor

::=	Basissymbol	Einer vorhandenen Spalte werden neue Werte zugewiesen. `n ::= e` **ersetze** den Inhalt der „Spalte“ n durch die Werte des Ausdrucks e

=:	Basissymbol	**Zuweisung**:
		`1,2`
		`=: $XX`

&	binäre Boolesche Operation	**Konjunktion (und):** si & no = no
|	binäre Boolesche Operation	**Disjunktion** (oder): si | no = si

->	binäre Boolesche Operation	**logische Implikation**: si `-> no = no`
<->	binäre Boolesche Operation	**logische Äquivalenz: no** `<-> no = si`
>, <, <=, >=	Relationen	`3>4` ist falsch (no)
“	Anführungszeichen	wird benutzt um Texte mit Leerzeichen zu einer Einheit zu verschmelzen Otto =“Otto“, “Es regnet heute“ ist ein Text; (Es regnet heute) ist eine Liste von Wörtern.

+text	binäre Textoperation	**Konkatenation (Verkettung)**:
		`Hallo +text ” “ +text Otto =`
		`“Hallo Otto”`
		Der Typ der ersten Tabelle bleibt unverändert.

{}	Klammern für Mengen von Werten	{1 1}={1}
{{ }}	Klammern für Multimengen von Werten	{{1 1}} ungleich {{1}}
[ ]	Klammern für Listen von Werten	[1 3 4]
#	Basissymbol	**Kommentar**: der folgende Text der Zeile ist Kommentar
$	Basissymbol	bezeichnet eine (Tabment) **variable**: $X:=(8,9)
(#, #)	Begrenzer	**Kommentar: Beginn/Ende** von mehrzeiligem Kommentar
|	Basissymbol	**Alternative**: X|Y l; **Strich** (bar): [| | |]

+tup	Mengenoperation	`<TAB!`
		`X,Y l`
		`1 2`
		`3 4`
		`!TAB>`
		`+tup (4,5)`
		`=`
		`X,Y l`
		`5 7`
		`7 9`
		zu jedem **Tupel** des Tabments wird ein zweites Tupel addiert

*tup	Mengenoperation	**Tupelmultiplikation:** (2,3) *tup (4,5)= (8,15)
-tup	Mengenoperation	**Tupeldifferenz**:
:tup	Mengenoperation	**Tupeldivision**:

*mat	Matrizenoperation	**Matrixmultiplikation**: (1,2) *mat [2 3]=8
		`<TAB!`
		`X1,X2,X3 l`

		1 0 2 0 2 0 0 0 8 !TAB> *mat ## <TAB! X1, X2, X3 l 1. -0. -0.25 -0. 0.5 -0. 0. -0. 0.125 !TAB> = X1, X2, X3 l 1. 0. 0. 0. 1. 0. 0. 0. 1.
-1mat	Matrizenoperation	<TAB! X1,X2,X3 l 1 0 2 0 2 0 0 0 8 !TAB> -1mat = X1, X2, X3 l 1. -0. -0.25 -0. 0.5 -0. 0. -0. 0.125 **inverse Matrix**
<, >	Begrenzer	**Beginn/Ende von Tags:** <HSQ: ; :TAB>
/	Basissymbol	AUTOR/NAME: in der XML-Repräsentation ist der **Tag** NAME is **direkt in** AUTOR **enthalten**; das wird z.B. benötigt, wenn die folgenden zwei Typen gegeben sind: AUTOR! NAME, VORNAME EDITOR! NAME, VORNAME, FIRMA
//	Basissymbol	AUTOR//VOR: zwischen AUTOR und VOR sind mehrere Tags erlaubt; AUTOR! NAME, FIRMA NAME! VOR, MITTEL?, NACH aber AUTOR/NAME/VOR ist effizienter
!	Basissymbol	**Begrenzer**: [X! 77 88] wird bei einzeiligen Tabmenten benutzt
&&	unäre Aggregation	Statistik: **für alle** Aggregation: si,66,si && = si

||	Aggregation	Statistik: **Existenzaggregation:**
		`WEIBLICHl ||`
		`1=2,7=8`
		`||`
		`= no`

..	binäre Operation mit Listenoutput	`1 .. 4 **`
		`= 24`
		(entspricht hier der Fakultätsfunktion)

...	Operation mit Liste als Ergebnis	`1 ... 4,2` = 1 3 x ... y,z: alle Zahlen x, x+z, x+2z,..., x+n*z: x+n*z<=y
abs	unäre Zahlenoperation	der absolute Betrag einer (P)Zahl: 7~ abs = 7 6 abs = 6

add	Mengenoperation	t1 add t2: **füge** t2 in t1 bzgl. gleicher Spaltennamen **ein**; der Typ von t1 ist der Ergebnistyp; ähnlich zur gib-Klausel
		`<TAB!`
		`X,Yl m`
		`1 2 3`
		`4 5`
		`!TAB>`
		`add &&`
		`<TAB!`
		`Y,X`
		`6 1`
		`!TAB>`
		`=`
		`X, Yl m`
		`1  2 3 6`
		`4  5`

add1	binäre Mengenoperation	`[1 7]`
		`add1 6`
		`=[1 7 6]`
		$x add1 $y: füge ein Element $y zur Kollektion $x hinzu

at	Schlüsselwort	**Erweitere an einer Position** `ext n:=e at n2`: erweitere rechts neben n2
atom	gib-Klausel	atomares Subtabment `atom! HOBBYl` HOBBYl wird während der Umstrukturierung als Gesamtheit transferiert
aus	Schlüsselwort	`aus studenten.tab`
avec	Schlüsselwort	**Selektion** `avec GEHALT > 5000` nur die spezifizierten (komplexen) Zeilen verbleiben im Ergebnis

		`avec ABTEILUNG!  GEHALT >5000` alle Abteilungen, in denen ein Angestellter mehr als 5000 verdient.
BAR	Datentyp	enthält nur ein Element (!); daher erst BARl sinnvoll
begin, end	Begrenzer	**Beginn/Ende** eines Unterprogramms
BOOL	Datentyp	enthält 2 Wahrheitswerte (si,no); zu Ehren des englischen Mathematikers George Boole

comp	binäre Operation	`NAME,   VORNAME,ORT`
		`Miller  Paul    Magdeburg`
		`comp ORT`
		ergibt `MD;` siehe auch `nth`

cos	trigonometrische Operation	**Kosinus**: 0 `cos=1.`

crosstab	Mengenoperation	**cross table**:
		`<TAB!`
		`NAME, (FACH,   NOTE m)m`
		`Paul   Mathe   1`
		`       Deutsch 2`
		`Sophia Mathe   2`
		`       Bio     1`
		`       Deutsch 1`
		`!TAB>`
		`crosstab`
		`=`
		`NAME,  BIO?, DEUTSCH?, MATHE? m`
		`Paul         2         1`
		`Sophia 1     1         2`
		die Werte der vorletzten Spalte werden zu Spaltennamen und die Werte der letzten Spalte werden die zugehörige Werte

csv	Suffix	**csv-Datei** kann als Input oder Output von o++oPS Programmen dienen

dann	Schlüsselwort	`1=1 dann 2,3 = 2`
		`1<1 dann 2,3 = 3`
		`condition2 dann expr1, expr2`

dann1	Schlüsselwort	`XX := ORT=Halle dann1 gut` XX erhält nur einen Eintrag, wenn die Person aus Halle ist, d.h. es wird um XX? erweitert.

det	Matrizenoperation	`<TAB!`
		`X1,X2,X3 l`
		`1  0  2`
		`0  2  0`
		`0  0  8`
		`!TAB>`

		`det`
		`= 16.`
		Determinante

div	binäre algebraische Operation	**ganzzahlige Division**
		`7 div 3 = 2`
		`8.55 div 2.1 = 4`

divrest	binäre algebraische Operation	**ganzzahlige Division mit Rest**:
		`7 divrest 3 = 2,1`
		`7.1 divrest 3.5 = 7,0.1`

ext	Basisoperation	**extension (Erweiterung)**: `ext e at n2` erweitere das gegebene Tabment um (komplexe) e-Werte rechts neben jedem n2-Wert

gib	Basisoperation	**restrukturiere, sortiere, aggregiere, vereinige, eliminiere Duplikate,...**:
		`aus studenten.tab`
		`gib FAK,(ORT,NAMEm)m)m`
		transformiere ein Tabment in ein Tabment mit gegebenen Schema bzw. gegebener TTD

gib+	Basisoperation	restrukturiere mehrere Tabmente; vor der Restrukturierung wird der „join“ („Durchschnitt“) der gegebenen Tabellen realisiert.
		`aus  studenten1.tab,examen1.tab`
		`gib+ FAK,(NAME,(KURS,NOTE m)m)m`

giball	Basisoperation	`giball X | Y l` **Liste aller** X- und Y-**Elemente** (beliebige Tiefe); entspricht dem Doppelslash ...//X|Y von XPath
gibtop	Basisoperation	`gibtop Xl` entspricht dem Slash: t/X: Liste aller X-Subtabmente von t, die in der obersten Ebene von t vorkommen.
hoch	binäre algebraische Operation	2 hoch 3 = 8 e hoch 2 `= 7.38905609893`
hsq	Suffix	In- und Outputdatei; jeder Zeile entspricht ein Segment; die Felder eines Segments werden mit 2 oder mehr Leerzeichen voneinander getrennt
in	Relation	sind linke und rechte Seite Kollektionen gleichen Typs (bis auf Tags, so stellt in die „mengentheoretische“ Inklusion dar, [1 3] in [1 4 3] =si ist die linke Seite vom Elementtyp der rechten, so ist in die Elementrelation 2 in { 6 7 2} =si ansonsten wird gefragt, ob jedes Wort der linken Seite in der rechten Seite vorkommt: `“Student ein“ in “Ein Student lebt`

		gut.“ = si jedes Wort der linken Seite ist in der Menge der Worte der rechten Seite enthalten
leftat	Schlüsselwort	`ext n:=e leftat n2`: siehe oben **at**
like	Boolesche Operation	`Hadmersleben like "?admers*"` = si `'?'`: repräsentiert ein Zeichen `'*'`: null oder mehrere Zeichen

linreg	Aggregation	`<TAB!`
		`FLASCHENPREIS, VERKAUFTEMENGE l`
		`20 0`
		`16 3`
		`15 7`
		`16 4`
		`13 6`
		`10 10`
		`!TAB>`
		`linreg`
		=
		`Y0, ANSTIEG`
		`19.7321428571 -0.982142857143`

lists	unäre Basisoperation	**Liste von Listen der Elemente:**
		`[X! 1 2] lists 2`
		=
		`Xl l`
		`1 1`
		`1 2`
		`2 1`
		`2 2`

ln	unäre algebraische Operation	**natürlicher Logarithmus**; e ln =1.
m, m-, b, b-, l, l-, a, s, ?	Kollektionssymbole	m: Menge, m-: Menge umgekehrt sortieren, b (bag) Multimenge, l: Liste, a: Kollektion vom ANY-Typ, s: Strom (Stream noch nicht angefangen); ?: optionaler Wert; die Kollektionssymbole werden postfix notiert und können ohne Leerzeichen an ein Tag angehängt werden.
mal	Tabmentoperation	5 mal "Ich liebe Dich!" = Ich liebe Dich! Ich liebe Dich! Ich liebe Dich! Ich liebe Dich! Ich liebe Dich!

max	Aggregation	Statistik: **Maximum**
		`1.1,2,Hallo`
		`max = 2.`

median	Aggregation	`(2 6 3 2),7,8`

		`median`
		ergibt
		`4.5`

ment	Suffix	Dokumentdarstellung eine Tabments; unterscheidet sich von XML durch vereinfachte Angabe der Metadaten

min	Aggregation	Statistik: **Minimum**
		`1.1,2,Hallo`
		`min`
		`ergibt 1.1`

no	Booloperation	**Boolesche Konstante:** Wahrheitswert falsch; entspricht Antwort no (spanisch)

not	unäre Boolesche Operation	`si not = no`
		Negation
		`X not not = X`

nth	binäre Operation	n-te Komponente bzw. n-tes Element `emperors.tab nth 2 comp NAME`

nurtext	unäre Textoperation	`1,a1,Butter`
		`nurtext = a1 Butter`
		Verkettung aller TEXT-und WORT-Werte eines Tabments (Operation kann auch durch andere Operationsfolgen ausgedrückt werden)

onein	Relation	`[1 2] onein [1 3 4] = si` wie in, nur dass nur ein Element (ein Wort) der linken Seite ist in der rechten enthalten sein muss.
polygon	unäre algebraische Operation	`[X,Y! 0 0  1 1  0 1]` zeichnet die 2 Strecken (0,0) bis (1,1) und (1,1) bis (0,1)
polynom	binäre algebraische Operation	`[3 1 4] polynom 2` berechnet den Funktionswert von X*X*3+X+4 an der Stelle 2
pos	unäre Positionsoperation	`NOTE pos < 5` die ersten vier Noten
pos-	unäre Positionsoperation	`NOTE pos- < 5` die letzten 4 Noten
pred	unäre Positionsoperation	**Vorgänger** 1. innerhalb von ext: NAME pred: NAME-Wert des Vorgängers 2. innerhalb von rec: AMOUNT pred AMOUT-Wert des Vorgängers
pred_n	binäre Positionsoperation	`NOTE pred_n 3` dritter **Vorgänger** innerhalb einer Kollektion
PZAHL	Datentyp	**Zahl mit Punkt** (früher Kommazahl=Float): 2.34
pzahl	Konvertierungs-operation	**konvertiere** eine Zahl oder einen Text in eine **PZAHL**

pzahlen	unäre Tabmentoperationen	alle Pzahlen eines Tabments werden ausgegeben (keine Typkonvertierungen)
rename to	Basisoperation	`rename X to Y`: **ersetze** jeden Spaltennamen X durch den Namen Y
rest	binäre algebraische Operation	Rest der ganzzahligen Division: `7 rest 3 = 1`

rnd	binäre algebraische Operation	`[2.1436 5.88] rnd 1`
		=
		`2.1 5.9`
		`z rnd n`: **runde** z auf n Ziffern nach dem Punkt

sans	Schlüsselwort	**Selektion** `sans ORT=Magdeburg` sans: ohne die spezifizierten (komplexen) Tupel
saetze	Textoperation	`LEBENSLAUF saetze` **Liste aller Sätze**; Das Resultat ist vom Typ: SATZl
si	Boolesche Konstante	Wahrheitswert **wahr (entspricht der Antwort ja)**
sin	unäre trigonometrische Operation	`3.14159 sin` =2.65358979335e-06 **Sinusfunktion**
sqrt	unäre algebraische Operation	**Quadratwurzel** `4 sqrt` =2.
streuung	unäre Aggregation (mad)	[1 2 5 3 5 1] streuung = 1.5

strip	unäre Basisoperation	`<TAB!`
		`X,Y?,Zl,Wm m`
		`1  2  3`
		`  4`
		`!TAB>`
		`strip`
		=
		`(X, Y?, Z?,  Wm)?`
		`1  2  3`
		`  4`
		Alle Kollektionssymbole, zu denen jede Kollektion höchstens 1 Element enthält, werden durch ? ersetzt.

subtext	dreistellige Textoperation	`aBCdE subtext 2,3` = BCd text subtext beg, len: beg beginnt bei 1 an zu zählen, len ist die Länge des Ergebnistextes; der zweite Inputwert muss ein Paar von ganzen Zahle sein
subtext2	dreistellige	`aBCdEfgh subtext2 "B", fg`

	Textoperation	=CdE text3 subtext2 text1, text2: Text von text3 zwischen text1 und text2

subtextend	binäre Textoperation	`asdfgh subtextend 4`
		`=fgh`
		Rest des Textes ab der spezifizierten Position

subtextend2	binäre Textoperation	`asdfgh subtextend2 4`
		`=dfgh`
		Rest des Textes ab der spezifizierten Position von hinten gezählt

succ	unäre Positionsoperation	`NOTE succ`: **Nachfolger** innerhalb einer Kollektion; das gesamte (Sub)Tupel ist der Nachfolger
succ_n	binäre Positionsoperation	`NOTE succ_n 3`: dritter **Nachfolger** innerhalb einer Kollektion
tab	Suffix	ein **Tabment in tabellarischer Sicht**
TABMENT	Tag	**virtueller Tag** um das gegebene Tabment
tag	unäre Tabmentoperation mit Parameter	`1 tag X` t1 tag ROOT1: um t1 wird ein ROOT1-Tag gesetzt

tags	unäre Tabmentoperation mit Parameter	`1 .. 3 tags X`
		`=`
		`X1`
		`1 2 3`

TEXT	Datentyp	**text Datentyp** (string)

text	unäre Textoperation	`13.2,[ab cc],Bc`
		`text`
		`=13.2 ab cc Bc`
		alle Werte werden in **Text** umgewandelt und verbunden

textsep	binäre Textoperation der zweite Inputwert ist der Separator	`1 .. 10 textsep ","`
		`=`
		`1,2,3,4,5,6,7,8,9,10`

textindex	binäre Textoperation	`“Heute wird schoenes Wetter“`
		`textindex wir`
		`=7`

textcut	binäre Textoperation	

time	algebraische Operation	`ext X:= time`
		`=1449251939.91`
		Systemzeit; muss in der Regel zweimal angewandt werden, um die Differenz zu bilden

untag	unäre Basisoperation	`1 tag XX untag`
		`=1`
		untag(Tag(n,t'))=t'; streiche den äußersten Beginn (und

		End) Tag
upper	unäre Textoperation	`1.2,aW upper` = `1.2  AW` (hsq Ausgabe) jeder Kleinbuchstabe wird in einen Großbuchstaben umgewandelt; der Rest verbleibt unverändert.

variance	Aggregation	`[1 2 4 6] variance`
		=
		`4.91666666667`

vertical	Basisoperation	`<TAB!`
		`NAME,  BIO?,DEUTSCH?, MATHE? l`
		`Paul         2          1`
		`Sophia 1     1          2`
		`!TAB>`
		`vertical ##`
		`FACH,NOTE l:=BIO,GERMAN,MATHE`
		=
		`NAME, (FACH,    NOTE m)m`
		`Paul    DEUTSCH  2`
		`        MATHE    1`
		`Sophia  BIO      1`
		`        DEUTSCH  1`
		`        MATHE    2`
		vertical X,Y m :=C,D,E: die Spaltennamen C,D,E erscheinen in der Spalte X und die entsprechenden Werte in Spalte Y

vlists	unäre Basisoperation	variabel lange Listen; die Operation stimmt mit lists überein, nur dass alle kürzeren Listen noch im Ergebnis eingeschlossen sind.
weg	Basisoperation	`weg XX Y`: vergiss die Spalten (Tags) XX und Y

wege	Basisoperation	eine gegebene Tabelle tab:
		`SUP,XX,…,(SUB,YY,…l)m` wird als gerichteter, gewichteter, zyklenfreier Graph mit Kanten von SUP nach SUB interpretiert.
		`tab`
		`wege sup0`
		ist die Liste aller Wege von sup0 bis zum „Endknoten“.
		Sie ist vom `((SUB,YY,…)l)l.`
		`<TAB!`
		`SUP, (SUB,ANZ l)m`
		`t0`
		`t1`
		`t2`
		`t3    t1  5`
		`      t4  6`
		`t4    t2  3`

		`t0 2`
		`!TAB>`
		`wege t3`
		`=`
		`(SUB, ANZ 1)1`
		`t4 6`
		`t0 2`
		
		`t4 6`
		`t2 3`
		
		`t4 6`
		
		`t1 5`
		(Die Leerzeilen wurden eingefügt.)

worte	unäre Textoperation	`"We are 6."`
		`worte`
		={6 are we}
		alle **Worte** eines Tabments

xml	Suffix	studenten.xml: **XML-file**
ZAHL	Datentyp	beliebig große ganze Zahlen (bigInt)

zahl	unäre Konvertierungs-operation	**konvertiere** TEXT oder PZAHL **in ZAHL**
		`"12" zahl`
		`=12`

zahl1	unäre Konvertierungsoperation	`"24:5:33" zahl1`
		`=24`
		erste Zahl im Text; der Text muss mit einer Ziffer beginnen

zahl2	unäre Konvertierungsoperation	`"24.05" zahl2`
		`=5`
		zweite Zahl im Text

zahl3	unäre Konvertierungsoperation	`"24:AA:5::087" zahl3`
		`=87`
		dritte Zahl im Text

zahlen	unäre Tabmentoperationen	alle Zahlen eines Tabments werden ausgegeben (keine Typkonvertierungen)
zufall	Operation mit Listenoutput	`5 zufall 1,6` = 1 4 2 6 1