NAME

choose – Auswahl von Werten in **sh**-Scripten

SYNOPSIS

choose `-t`*Titeltext* `-a` `-b`*Breite* `-d` `-h` `-l`*Hoehe* `-m` `-w`*Text* *Auswahlitems* ...

BESCHREIBUNG

choose läßt einen oder mehrere der Commandlineparameter auswählen. Die Parameter werden tabellarisch angezeigt und können durch Eingabe der Platznummer ausgewählt werden. Die angewählten Werte werden als Wortliste nach **stdout** ausgegeben. Die Interaktion mit dem Benutzer findet über `/dev/tty` statt.

- Mehrere Items können so ausgewählt werden (Kommaliste): „`1,5,7`“
- Es können auch Ziffernbereiche ausgewählt werden: „`1-5,9`“.
- Es können auch Items wieder ausgeschlossen werden (Minuszeichen): „`1-9, -3`“ wirkt wie „`1,2,4,5,6,7,8,9`“

Exitstatus: normalerweise 0. 1 bedeutet: es gab gar keine Auswahl.

OPTIONEN

`-t`*Titeltext* Angabe eines Textes, der über der Auswahltabelle angezeigt wird.

`-a` 'alle', wenn ein „a“ eingegeben wird, werden alle Einträge ausgewählt.

`-b`*Breite* die Bildschirmbreite, die benutzt werden soll.

`-d` Display: nur Anzeige, nicht auf Auswahl warten.

`-h` numeriere die Items horizontal statt vertikal.

`-l`*Hoehe* Höhe der angezeigten Tabelle mindestens.

`-m` multi, erlaube Mehrfachauswahl.

`-w`*Text* auch leere Auswahl möglich; diese wird mit Exitstatus 1 angezeigt. Ist *Text* angegeben, wird die Prompt um „`[leer=`*Text*`]`“ bereichert.

Auswahlitems ... die restlichen Commandlineparameter werden in der Tabelle angezeigt und können ausgewählt werden.

FEHLER

`-h` funktioniert (noch) nicht.

NAME

jaantwort – Stellt Fragen, die mit Ja oder Nein beantwortet werden.

SYNOPSIS

jaantwort -d*Defaultantwort* [*Fragetext*]

BESCHREIBUNG

Stellt die Frage *Fragetext*, und holt die Ja/Nein Antwort von `/dev/tty`. Wenn *Defaultantwort* angegeben wurde, kann diese mit Enter gewählt werden. ExitCodes:

0 = „Ja“, wenn das 1. Zeichen der Antwort ‘j’, ‘J’, ‘y’ oder ‘Y’ war.

1 = „Nein“, wenn das 1. Zeichen der Antwort ‘n’ oder ‘N’ war.

Für die Benutzung in **sh**-Skripten gedacht.

OPTIONEN

`-d`*Defaultfrage* : ein Text, der mit Return bestätigt werden kann. Sonst eigene Eingabe nötig.

Fragetext : die Frage, die gestellt wird. Diese muß das abschließende Fragezeichen beinhalten und kann auch weg bleiben.

BEISPIEL

```
if jaantwort -dNein 'Alles loeschen ?'
then rm *.o *.a
fi
```

FEHLER

keiner

NAME

stadef – Beschreibung der stadef-Struktur

BESCHREIBUNG

stadef ist eine Datenbankstruktur, die im Laborbetrieb prozeßbezogene Daten speichert. stadef benutzt das textrel-Format zum Datenaustausch und die relationale Datenbank „db++“ zur Datenspeicherung.

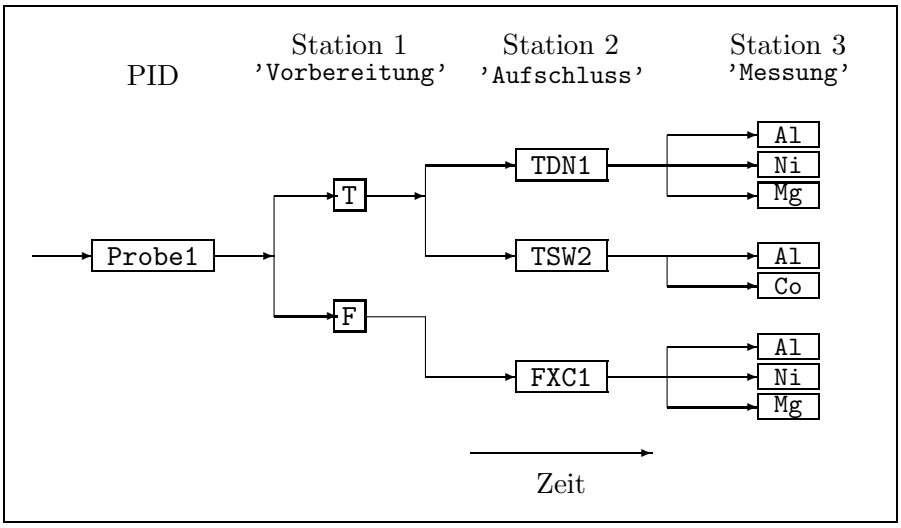
Grundkonzepte des stadef-Systems

Das System kann die Analyse von Proben verwalten, die in einem mehrstufigen Prozeß analysiert werden. Die Datenstruktur spiegelt den Fortlauf des Analyseprozesses im Labor wieder, und sorgt so für eine vollständige, redundanzfreie und sichere Datenerfassung.

Gegeben ist eine Probe, die durch eine „Probenkennung“ (PID) gekennzeichnet ist. Die PID ist in eine „Probenart“ und eine „Probennummer“ gegliedert. Je nach Probenart werden die Proben verschieden behandelt. Eine „Analyse“ besteht aus Probenmaterial, das nacheinander an verschiedenen „Stationen“ je einer von verschiedenen „Behandlungen“ unterworfen wird. Gespeichert werden die pro Behandlung anfallenden „Meßdaten“ (die i.A. für jede Behandlung verschieden sind). Nach einer Behandlung kann das resultierende Probenmaterial geteilt werden, um an der nächsten Station auf verschiedene Weisen weiterverwendet zu werden.

Dadurch ergibt sich für eine Probe eine baumartige Struktur, die alle Analysen beschreibt, die mit der Probe durchgeführt wurden.

Beispiel eines Analysenbaums:



Eine bestimmte Analyse im Analysenbaum wird durch den „Analysenpfad“ beschrieben. Dies ist eine Liste der durchgeführten Behandlungen in der Reihenfolge der Stationen (genauso wird in hierarchischen Dateisystemen auf einzelne Dateien zugegriffen).

Die textrel-Darstellung eines Analysenbaums ist einfach die Liste aller Analysenpfad. (Die Domänen haben folgende Funktionen: `ProbenKenn` ist die Probenkennung, `VorberMethode`, `AufschlMethode` und `MessMethode` sind die Behandlungen an den Stationen `Vorbereitung`, `Aufschluss` und `Messung`.)

```
> ProbenKenn  VorberMethode  AufschlMethode  MessMethode
  Probe1      T              TDN1            Al
  Probe1      T              TDN1            Ni
  Probe1      T              TDN1            Mg
  Probe1      T              TSW2            Al
  Probe1      T              TSW2            Co
  Probe1      F              FXC1            Al
  Probe1      F              FXC1            Ni
  Probe1      F              FXC1            Mg
```

Die verwendeten Relationen:

Die Datenstruktur wird in den verschiedenen Datenrelationen (db++) wie folgt gespeichert:

Es gibt eine „Hauptrelation“, die das Rückrat der Datenhierarchie bildet. In ihr ist jede Analyse durch ein Tupel dargestellt, das Probenkennung, den Analysepfad und Verweise auf die „Datenrelationen“ (für die an jeder Station anfallenden Meßdaten) enthält. In diesen werden die anfallenden Daten gespeichert, pro durchgeführter Behandlung in einem Tupel. Für eine Station müssen nur soviele verschiedene Datenrelationen angelegt werden, wie es insgesamt verschiedenartige Datensätze gibt (i.A. haben viele Behandlungen gleichartige Datensätze).

Wie findet man von einer bestimmten Analyse an einer bestimmten Station das Datentupel? Zuerst muß das die Analyse beschreibende Tupel in der Hauptrelation gesucht werden. Dann wird aus diesem Tupel der Wert der Behandlung genommen, die der Behandlung zugeordnete Datenrelation enthält dann die Daten. Die Tupel in den Datenrelationen sind mit eindeutigen „Keys“ (Schlüsselzahlen) versehen, die auch in der Hauptrelation bei jeder Station verzeichnet sind. Also wird aus dem Tupel der Hauptrelation der Datenkey für die Station abgelesen und in der Datenrelation das Tupel mit demselben Key gesucht.

Die Definitionsdatei „StationDef“

Für eine konkrete Anwendung werden in der „Definitionsdatei“ festgehalten:

- Die Reihenfolge und die Namen der Stationen.
- Die an den Stationen zugelassenen Behandlungen, und für jede Behandlung der Namen der Datenrelation, in der die Daten dieser Behandlung stehen. Außerdem alle Domänen, die zur Station gehören.
- Der Aufbau aller verwendeten Datenrelationen, d.h., die Namen der Relationen (auch mit Environmentvariablen) und Reihenfolge und Namen der Domänen. Dies definiert erst die genaue Form der bei den Behandlungen anfallenden Datensätze.
- Ein Verzeichnis aller Domänen, mit den Angaben des Datentyps (String, integer, real, etc.), und Angaben der „Domänenart“, die über die Funktion der Domäne in stadef Auskunft gibt. Es gibt folgende Domänenarten:

  KEY : Domäne, die den Key in Haupt- und Datenrelation hält. der Datentyp muß ‘long’ sein.

  PID : Domäne ist Teil der Probenkennung.

  PATH : Domäne enthält die Behandlung an einer Station.

  WDHKEY : Wiederholungskey. Zusatz zu PATH, eine ganzzahlige Zahl, die bei mehreren gleichen Analysepfaden einzelnen Wiederholungen identifiziert (s.u.).

  Folgende Domänenarten unterscheiden die Anwenderdaten nach Funktionen:

  STATUS : Domäne, in der Bearbeitungsstatus für Station steht.

DATUM, ZEIT : Zeitpunkt, an dem Meßwert anfiel.
PERSON : Name der Person, die Messung durchführte.
LABOR : Name des Labors, das die Messung durchführte.
RDATA : angefallene (reelle) Meß-Daten.
PROZESS : Variante der Meßmethode, Messgerät.
VORB : Domäne mit Information zur Probenvorbehandlung an der Station.
VDATA : Allg. Daten für die Laborverwaltung.

Die Domänenarten sind deshalb so ausgefeilt, da Domänen häufig nicht über ihren Namen, sondern über ihre Funktion und ihre Station bestimmt werden (siehe die Utility **defs**).

Für die Datentypen gibt es folgende Einschränkungen:

Art:	KEY	DATUM	ZEIT	PERSON	PATH
Typ:	long	long	long	String/enum	String/enum

Wiederholungskeys

Es kann nötig sein, bestimmte Analysen mehrfach durchzuführen. D.h., ab einer bestimmten Station wird der Probendurchlauf wiederholt. Die einzelnen Wiederholungen werden über die „Wiederholungskeys“ auseinandergehalten. Bsp.: Viermal wird Aluminium mit TDN1-Aufschluß gemessen. Bei der ersten Wiederholung wird nur die Messung wiederholt, bei der zweiten und dritten auch der Aufschluß. `AufschWDH` u. `MessWDH` seien die Namen der Wiederholungskeys an den Stationen `Aufschluss` u. `Messung`. Dann werden die vier Pfade so dargestellt:

```
> PID     AufschMethode  AufschWDH  MessMethode  MessWDH
  Probe1  TDN1           1          Al           1
  Probe1  TDN1           1          Al           2
  Probe1  TDN1           2          Al           1
  Probe1  TDN1           3          Al           1
```

Zugriff auf die stadef-Struktur

Zum Zugriff auf die stadef-Struktur gibt es folgenden Operatoren (siehe auch in den entsprechenden Handbuchseiten):

stationchk : Prüft die Definitionsdatei auf logische Fehler.

makerels : Legt die benötigten Relationen an.

new : Trägt leere Analysepfade in die Datenbank ein, und baut damit die Baumstruktur für die Meßdaten auf.

insert : Trägt Meßdaten, die mit Pfadangaben versehen sind, in die Datenstruktur ein.

extract : Liest Daten aus der Datenbank aus.

relink : Ändert die Analysepfade in der Datenbank nachträglich; löscht Daten, trägt neue Behandlungen ein und fügt neue Daten ein.

forget : Löscht Analysen aus der Datenbank.

Alle Operatoren empfangen bzw. liefern ihre Daten im textrel-Format.

Um in **sh**-Ebene Zugriff auf die Datendefinitionen zu haben, gibt es den Abfrage-Interpreter **defs**.

ENVIRONMENT

DBMODUS – Filemaske für Relationen bei Neuerzeugung.
DEFDIR – Pfad für die Definitionsdatei.
LAPISUSER – Name des Benutzers, für Lockdatei.
LOCKDIR – Directory, in dem die Lockdatei angelegt wird.

DATEIEN

./StationDef, $DEFDIR/StationDef – Definitionsdatei
$LOCKDIR/LAPIS_LOCK – Lockdatei. Dient dem Ausschluß vom Mehrfachzugriffen.

NAME

defs – Abfragesprache für stadef-Struktur

SYNOPSIS

defs *Statements* `-n -k -s`

BESCHREIBUNG

defs gibt über eine Abfragesprache Informationen über die in der Datei `StationDef` definierte stadef-Struktur zurück. Es ist für den Einsatz in **sh**-Skripts entworfen, und kann auch zur Mengenarithmetik verwendet werden.

defs gibt Namen von Domänen etc. zurück, die durch Formeln berechnet werden.

Syntax

Es gilt folgende Syntax:

('{...}' : 1 bis n-malige Wiederholung. '[...]' : optionaler Teil.)

```
<statement>  :=  <exprlist> { [ ';' <exprlist> ] }
<exprlist>   :=  <expr> { <expr> }
<expr>       :=  <term> { [ '+' | '\' | '|' | '&' ] <term> }
<term>       :=  [ '!' ] <factor>
<factor>     :=  <Ident> | '(' <exprlist> ')' |
                 <function> ['(' <exprlist> ',' ')' ]
```

Semantik

Die Ausgabe enthält nie Werte doppelt (es werden immer echte Mengen ausgegeben).

- Bei Angabe mehrere, durch ';' getrennter `<exprlist>` erfolgt die Ausgabe in mehreren Gruppen, in folgender Weise: "result1 " "result2 " ...
  Die einzelnen Unterergebnisse können also mit

  `set -- 'defs` *expr*`;`*expr*`;`*expr*`'`

  auf die **sh**Variablen $1 ... $n verteilt werden.
- '|' : Mengenvereinigung, wie '+'.
- '&' : Mengenschnitt.
- '!' : ohne Funktion.

- <factor> kontrolliert das Anhängen von Typen.
- <Ident> : einfach nur ein Wort, ein Atom, der Name einer Domänen, einer Station, ...
- <function>:
  Folgende Funktionen sind eingebaut:

  doms(<expr>-*Liste*) :
  versucht, zu jedem <expr> alle zugehörigen Domänen zurückzugeben. Das klappt, wenn <expr> der Name einer Station, einer Relation oder ein Bezeichner einer Domänenart ist.
  '*' = alle Domänen geordnet.

  domtyp(<expr>-*Liste*) :
  gibt zu jedem Domänennamen in der <list> die Domänen-Art (= Funktion) zurück. '*'= alle Werte.

  fieldtyp(<expr>-*Liste*) :
  gibt zu jedem Domänennamen in <list> den Datentyp zurück. '*'= alle Werte.

  stats(<expr>-*Liste*) :
  gibt zu jedem <expr> die Station zurück.
  '*' = alle Stationen. Ok, wenn <expr> eine Domäne, eine Relation, oder eine dezimale Nr. ist.

  statsr(<expr>-*Liste*) :
  Stationen-Range: wie 'stats', aber Rückgabe aller Stationen zw. min. und max. Argument (z.B.: `statr(0,2)` gibt den Namen der 1., 2. und 3. Station zurück).

  rels(<expr>-*Liste*) :
  gibt zu jedem <expr> die Relationen zurück. Ok, <expr> wenn Domäne, oder Station ist. '*' = alle Relationen. (wie stats)

  behs(<expr>-*Liste*) :
  gibt zu jedem <expr> die möglichen Behandlungen zurück. <expr> wird als Station ausgewertet.

  reverse(<expr>-*Liste*) :
  dreht die Liste um.

nicht erkannte Worte werden einfach übernommen.

Optimierung: Der Definitionsfile wird erst gelesen, wenn er gebraucht wird.

OPTIONEN

Statement : Eine Abfrageausdruck, der der oben definierten Syntax entspricht.

`-n` : In der Ausgabe Domänen durch Zeilenvorschub trennen.

`-k` : In der Ausgabe Domänen durch Komma trennen.

`-s` : In der Ausgabe die Typenangaben von den Domänen entfernen.

BEISPIELE

Mengenarithmetik:

```
defs “(gelb gruen rot braun) \ (rot gelb blau)”
```

ergibt die Ausgabe ”`gruen braun`”.

Ausgabe aller Domänen der Station ’$Station’:

```
defs ”doms($Station)”
```

Ausgabe aller Domänen, die den kompletten Pfad bilden, und abspeichern in der **sh**-Variablen `doms`:

```
doms=‘defs ”doms(PID PATH WDHKEY)”‘
```

Dto., aber geordnet (doms(*) ist einen feste Domänenreihenfolge).

```
defs ”doms(*) & doms(PID PATH WDHKEY)”
```

Alle Stationen bis inkl. ’$Station’:

```
defs ”statsr(0 $Station)”
```

Die Domänen des Pfades bis einschl. $Station:

```
defs ”doms(*) & doms(statsr(0 $Station)) & doms(PID,PATH,WDHKEY)”
```

DATEIEN

StationDef – Definitionsdatei.

DIAGNOSE

Syntaxfehler werden erkannt.

NAME

extract – Auslesen der stadef-Datenbank

SYNOPSIS

extract *Suchkriterien*

oder

extract *Ausgabedomänen* -r*Filename*

BESCHREIBUNG

extract liest Daten aus der stadef-Datenbank aus. Es sucht nach bestimmten Analysen („Suchkriterien“), und zeigt von diesen bestimmte Informationen an („Anzeigedomänen“). Jeder gefundene Analysepfad wird mit den verlangten Datendomänen in textrel-Darstellung in die Ausgabe (nach **stdout**) geschrieben.

Die Suchbedingungen werden entweder über die Commandline angegeben, oder kommen aus einer Relation.

Commandline-Aufruf:

Ein *Kriterium* in der Commandline kann sein:

- '*Domäne relop Wert*‘: mit den *relops* `<, <=, ==, >=, >, ~, !~`. Datensätze (Tupel, Zeilen) ausgeben, bei denen *Domäne* einer bestimmten Bedingung genügt. (bes.: nur '~‘ = Suche nach Leertext)
- '*‘: Alle Domänen ausgeben.
- *Domäne*: die angegebene Domäne ausgeben.
- *RelationName* Alle Domänen der genannten Relation ausgeben.
- *StationName* Alle Domänen der genannten Station ausgeben.

Es werden die Analysen gesucht, die alle Kriterien erfüllen; auch die Domänen, auf die Suchkriterien gesetzt wurden, werden mit ausgegeben.

Suchen aus Textrelation

Wird nicht über die Commandline, sondern aus einer Textrelation gesucht (`-r`), so wird der oben beschrieben Extraktionsvorgang für jedes Tupel der Eingaberelation separat durchgeführt. Ausgabe-Domänen sind die angegebenen und alle Domänen des Eingabe-Tupels. Die Suchkriterien werden folgendermaßen aus den Eingabe-Tupeln gebildet:

- Feldinhalte werden, wenn möglich, nach '*low-high*‘ zerlegt; dies wird ausgewertet als '*low* `<=` *DomName*‘ und '*DomName* `<=` *high*‘
- der Feldinhalt '`~`*Ausdruck*‘ wird als 'regular expression‘ erkannt.
- der Feldinhalt '`!` *Ausdruck*‘ wird als 'ungleich expression‘ erkannt.
- andere Feldinghalte müssen genau erfüllt werden: '*DomName* `==` *Feldinhalt*‘
- ein leeres Feld wird ignoriert. (Nach einem Leerstring kann mit einem einzelnen '`~`‘ gesucht werden).
- ein Tupel mit auschließlich leeren Feldern wird ignoriert.

Das Suchen nach Textrelationen 'vervollständigt‘ die Daten in der Textrelation mit den Daten aus der Datenbank.

Angaben über die Ausgabe-Domänen sollten trotz nachgeschalteter Filter gemacht werden, da der Extraktionsvorgang so effizienter wird (Datenrelationen müssen evtl. nicht gelesen werden).

Da die Ausgabe-Reihenfolge von **extract** vom Aufbau der Datenbank und den gewählten Kriterien anhängt (und deshalb in manchen Fällen nicht vorhersagbar ist), empfiehlt sich folgender Aufruf zur Datenverdichtung:

```
extract ...  | trbundle | trsort >...
```

Exit-Status:
0 = Daten gefunden.
1 = keine Daten gefunden.

OPTIONEN

Kriterien : Suchbedingungen und Anzeigedomänen, jede Domäne in einer Suchbedingung wird auch angezeigt.

Ausgabedomänen : Domänen, die von jeder gefundenen Analyse angezeigt werden sollen.

`-r`*Dateiname* : Name der Datei, aus der Textrelationen mit Suchkriterien gelesen werden (1 Tupel = 1 Suchvorgang).

DATEIEN

StationDef – Definitionsdatei.

NAME

forget – Einträge aus der stadef-Datenbank löschen.

SYNOPSIS

forget `-i`*infile* `-o`*outfile* `-e`*errfile*

BESCHREIBUNG

forget löscht bestimmte Analysenpfade aus der stadef-Datenbank. Die Eingaberelation muß nur den Key der Hauptrelation enthalten, die Analysen mit den angegebenen Keys werden gelöscht.

Datensätze aus den Datenrelationen werden gelöscht, wenn der letzte auf einen Datensatz verweisende Eintrag in der Hauptrelation gelöscht wird.

In die Ausgabe werden die Tupel der Eingabe geschrieben, die einen erfolgreichen Löschvorgang bewirkten. Kann eine Analyse nicht gefunden werden, so wird das entsprechende Tupel der Eingabe mit einer Fehlermeldung in den Fehlerausgang geschrieben.

OPTIONEN

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

DATEIEN

StationDef – Definitionsdatei.

NAME

insert – schreibt Daten in die stadef-Datenbank zurück.

SYNOPSIS

insert `-d`*Datendomänen* `-raw` `-i`*infile* `-e`*errfile*

BESCHREIBUNG

insert schreibt Daten in die Datenbank zurück, die zuvor ausgelesen und dann vervollständigt wurden. Die Daten für die Analysen werden tupelweise aus einer Textrelationenliste gelesen. Ein Tupel der Eingabe enthält Daten für eine Analyse, die über den KEY der Hauptrelation (er muß in der Eingabe vorkommen und darf nur durch vorheriges Auslesen der Datenbank gewonnen werden) ausgewählt wird.

Geschrieben werden Daten für alle Stationen, deren KEY und PATH in der Eingabe vorhanden ist (PATH wählt die Relation aus, KEY das Tupel). Für eine Station werden alle Domänen der Eingabe geschrieben, die in der durch PATH angegebenen Behandlung möglich sind; die schreibbaren Domänen können aber auch in der Commandline beschränkt werden.

OPTIONEN

`-d`*Datendomänen* Nur die genannten Domänen der Eingabe werden geschrieben.

`-raw` Daten werden direkt in eine einzige db++-Relation geschrieben, nicht relationenübergreifend.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

DATEIEN

StationDef – Definitionsdatei.

DIAGNOSE

Wenn ein Tupel der Eingabe unbrauchbar ist, wird es zusammen mit einer Fehlerbeschreibung in die Fehlerausgabe geschrieben.

FEHLER

Wenn zwischen dem Auslesen der Datendatei und dem Wiederrückschreiben Operationen wie **forget**/**new** oder **relink** vorkommen, zeigen die KEYs der Datendatei u.U. nicht mehr auf die urpsrünglichen Daten. Dann werden die Werte in andere Tupel geschrieben, als gemeint.

NAME

makerels – erzeugt / überprüft Relationen gemäß dem Definitionsfile.

SYNOPSIS

makerels *Filename*

BESCHREIBUNG

makerel überprüft die Relationen gemäß den Angaben im „StationDef"-File, und erzeugt neue Relationen, wenn sie noch nicht existieren.

Die Relationen werden bzgl. des aktuellen Directories angelegt, wenn in `StationDef` nicht absolute Pfade stehen (die auch durch Verwendung von Environment-Variablen gebildet werden können).

Die Hauptrelation wird mit folgenden „Sekundärkeys" (db++: zusätzlichen Sortierreihenfolgen) angelegt:

- PID-Domänen, dann Liste der PATH/WDHKEY Angaben in der Reihenfolge der Stationen.
- Für jede Station : KEY, PATH, WDHKEY, Key der Hauptrelation.

Dies optimiert häufige Suchvorgänge. (Begrenzung: nur insgesamt 7 Keys erlaubt).

OPTIONEN

Filename: Optionaler Name der Definitionsdatei. Default: `StationDef`.

DATEIEN

StationDef – Definitionsdatei.

FEHLER

Filemask der Relationen ist immer 0666, nicht $DBMASK.

NAME

new – trägt neue Analysenpfade in die stadef-Datenbank ein

SYNOPSIS

new `-i`*infile* `-o`*outfile* `-e`*errfile*

BESCHREIBUNG

new trägt neue Analysenpfade in die stadef-Datenbank ein, d.h., es erzeugt und verkettet Tupel in Haupt- und Datenrelationen, die später mit Daten gefüllt werden.

Eingabe ist eine Textrelationenliste. Jedes Tupel enthält einen einzufügende Analysepfad und wird einzeln bearbeitet.

Das Eingangs-Tupel muß enthalten: alle Domänen der Probenkennung, und für jede Station die Behandlung, und evtl. einen WDHKEY, falls der gleiche Analysenpfad mehrfach eingetragen werden soll. Default ist: alle WDHKEYs auf 1.

Die Analysepfade werden als Zweige an den schon vorhandenen Analysebaum der entsprechenden Probe angefügt. Enthält das Eingangtupel noch andere Domänen, so werden Haupt- und Datentupel des neuen Astes des Analysepfades mit diesen Daten gleich initialisiert. Datentupel von Stationen, die im Baum vor dem Abzweig des neuen Analysepfades liegen, werden aber nicht überschrieben.

Erfolgreich verarbeitete Analysepfade aus der Eingabe werden zusammen mit den erzeugten bzw. schon vorhandenen KEYS in die Ausgabe geschrieben, sonst werden sie mit einer Fehlermeldung in die Fehlerausgabe kopiert.

OPTIONEN

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

SIEHE AUCH

Zur Erläuterung des Analysebaums und seiner textrel-Darstellung siehe auch STADEF(1).

DATEIEN

StationDef – Definitionsdatei.

DIAGNOSE

Im Fehlerfall werden die entsprechenden Eingabetupel mit folgenden Fehlermeldungen in die Fehlerausgabe geschrieben:

Tupel schon vorhanden: Die Analyse war schon in der Datenbank. Soll die Analyse trotzdem rein, müssen andere WDHKEYs gewählt werden.

Behandlung existiert nicht: Im Analysepfad kommt eine nicht-existente Behandlung vor.

FEHLER

Es gibt keine Angabe, ab welcher Station ein Analysepfad an den vorhanden Analysebaum angehängt wurde.

Der Algorithmus zum Finden neuer, unbelegter KEY-Werte in einer Relation liest dabei alle Tupel durch.

NAME

relink – ändert Zweige im Analysenbaum

SYNOPSIS

relink `-i`*infile* `-o`*outfile* `-e`*errfile*

BESCHREIBUNG

relink ändert den Aufbau des Analysenbaums, in dem es an bestimmten Stationen die alte Behandlung löscht und eine neue einfügt. Es ändert aber nicht die Anzahl der Analysen.

Eingabe ist eine textrel-Relation. Jedes Tupel enthält Daten zum Ändern eines Analysepfades an mehrere Stationen. Die Änderung der einzelnen Stationen erfolgt von einander unabhängig in der Reihenfolge der Stationen.

Es muß in der Eingabe angegeben sein:

- den HRel-KEY für jeden zu ändernden Analysenpfad.
- für jede ändernde Station:
  - PATH, evtl. WDHKEY (Default 1).
  - evtl. Daten-Domänen der Station.

Für jede zu ändernde Station wird das entsprechende PATH (und WDHKEY)-Feld geändert, das Datentupel wird gelöscht und ein anderes Datentupel angehängt. Das neue Datentupel wird mit den zusätzlich angegebenen initialisiert. Dabei werden folgende Situationen behandelt:

- Doublette: ist der dadurch entstehende, volle Analysepfad schon vorhanden, gibt's nur eine Fehlermeldung.
- Relink: gibt es schon den Teil-Pfad bis zur Station, so wird das Datentupel der Station auch an den neuen Analysepfad angebunden.
- New: gibt es den neuen Teil-Pfad noch nicht, so wird ein neues Datentupel erzeugt.

Es werden die Eingabetupel mit den neuen KEYS an den Stationen in die Ausgabe geschrieben, im Fehlerfall mit einer Meldung in die Fehlerausgabe

OPTIONEN

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

BEISPIEL

Die Beispiele nehmen an, daß die Domänen folgende Bedeutung haben:

Domäne	Funktion
`HKey`	KEY der Hauptrelation
`PID`	Probenkennung
`Stat1`	Behandlung an Station 1
`Stat1Key`	KEY von Station 1
`Stat2`	Behandlung an Station 2

Die stadef-Datenbank habe folgenden Inhalt:

```
# Inhalt der Datenbank vor 'relink'
 >  HKey  PID   Stat1  Stat1Key  Stat2
    10    P100  PIWY   99        AKE
    11    P100  PIWY   99        AAS
    12    P100  PIZZ   37        WILLI
```

Beispiel 1: Andere Daten anschliessen

```
# Eingabe
 >  HKey  PID   Stat1  Stat1Key
    10    P100  PIZZ   egal
```

⟶

```
# Datenbank nach 'relink'
 >  HKey  PID   Stat1  Stat1Key  Stat2
    10    P100  PIZZ   37        AKE # !
    11    P100  PIWY   99        AAS
    12    P100  PIZZ   37        WILLI
```

In den Datenrelationen hat sich nicht geändert. Der Verweis in Analyse 10 wurde von PIWY auf PIZZ geandert. Gäbe es aber nicht die Analyse 11, so wäre in der Datenrelation für Station 1 der Eintrag 99 für PIWY gelöscht wurden (da er dann nicht mehr verwendet wird).

Beispiel 2: Neue Daten einschreiben

```
# Eingabe
 > HKey  PID   Stat1  Stat1Key
   10    P100  PAZZ   egal
```

⟶

```
# Datenbank nach 'relink'
 > HKey  PID   Stat1  Stat1Key  Stat2
   10    P100  PAZZ   <neu>     AKE # !
   11    P100  PIWY   99        AAS
   12    P100  PIZZ   37        WILLI
```

In der Datenrelation für Behandlung PAZZ steht jetzt ein neues Tupel.

DATEIEN

StationDef – Definitionsdatei.

DIAGNOSE

Bei Fehlern werden die entsprechenden Eingabetupel in die Fehlerausgabe geschrieben, mit folgenden Fehlertexten:

„HRel-KEY-Domaene fehlt.“ : Der KEY der Hauptrelation fehlt.

„HRel-Tupel nicht gefunden!“ : Der KEY der Hauprelation gibt keine existierende Analyse an.

„neuer Pfad nicht eindeutig.“ : Die Änderung kann nicht durchgeführt werden, da es den neuen Pfad schon gibt.

FEHLER

Die Änderung an mehreren Stationen gleichzeitig wurde noch nicht getestet.

NAME

stationchk – Testen der Definitionsdatei.

SYNOPSIS

stationchk -k*Relationen* `-l` *Filename*

BESCHREIBUNG

stationchk checkt die Definitionsdatei `StationDef`, ob sie konsistent ist. Ist die Option `-k` angegeben, wird auch noch die Datenbank auf Konsistenz der KEYs geprüft.

Der Reihe nach werden folgende Tests durchgeführt:

- Öffnen und Einlesen der Datei.
  Dies geschieht bei jedem stadef-Operator.
- Kontrolle der verwendeten Namen.
  Checkt, ob für jedes Objekt (Station, Behandlung, Domäne) ein eindeutiger Name gewählt wurde; und ob jede vereinbarte Domäne auch benutzt wird.
- Vergleich der Relationen mit ihren Definitionen.
  Es wird der Aufbau der vorhandenen Relationen mit den angegebenen Definitionen verglichen.
- Typenkontrolle der Domänen.
  Kontrolliert, ob die Domänen auch einen ihrer Art gemäßen Datentyp haben. Es muß gelten:

Art:	KEY	DATUM	ZEIT	PERSON	PATH
Typ:	long	long	long	String	String

- Check der KEY-Hierarchie.
  Check, ob jede Datenrelation genau eine Key-Domäne hat, ob jede Station mit Daten 1 KEY-Domäne in der Hauptrelation hat, ob alle erwähnten PID, und WDHKEYS in der Hauptrelation stehen, ob alle Relationen außer der Hauptrelation zu einer Station gehören.

  Optional (da zeitaufwendig):
- Daten-KEY-Test
  Check, ob alle Keys in den Datenrelationen vom Datenbank- Mechanismus in aufsteigender Reihenfolge gelesen werden.

  Check, ob jeder KEY nur einmal vorkommt.

- Datenverbund-Test
  Check, ob zu jedem HRel-Tupel alle Datenfelder in den Datenrelationen angesprochen werden können (sehr zeitaufwendig).

OPTIONEN

`-k`*Relationen* Führe auch den Daten-KEY-Test und den Datenverbund-Test durch.
Wenn *Relationen* angegeben ist: Der Daten-KEY-Test wird nur auf den angegebenen Relationen ausgeführt. Sonst: auf allen Relationen.

`-l` Wenn `-k`*:* Datentupel, die von keinem HRel-Tupel mehr gebraucht werden, löschen (Garbage-Collection).

Filename Optionaler Name der zu ladenden Definitionsdatei.

DATEIEN

StationDef – Definitionsdatei.

FEHLER

Bei Typenkontrolle: Fehler, wenn bei PATH statt String ein Aufzählungstyp vorkommt.

Bei KEY-Hierarchie: auch PATH-Domänen dürfen nur in der Hauptrelation vorkomen.

Die durchgeführten Tests sind alle ad hoc entstanden. Es ist noch nicht bewiesen, daß sie die Brauchbarkeit der Datei garantieren.

Dieser Test ist nur nötig, weil `StationDef` so kompliziert aufgebaut ist.

NAME

stationdef_t – Testprogramm für `StationDef`

SYNOPSIS

stationdef_t *Definitionsfile* `-c`*Filename* `-r`*Filename*

BESCHREIBUNG

stationdef_t liest die Definitionsdatei `StationDef` und gibt sie in anderer Form wieder aus.

Wird keine der Optionen verlangt, so erfolgt ein ausführliches Listing nach **stdout**.

OPTIONEN

Definitionsfile: Datei, die eingelesen wird. Default: `StationDef`.

`-c`*Filename* : erzeugt komprimierte, wieder lesbare Datei mit identischer Information.

`-r`*Filename* : erzeugt pro Station 1 Relation mit den gültigen Behandlungsnamen.

```
> Domname der PATH-Domäne
  Behandlung
  ...
```

DATEIEN

StationDef – Definitionsdatei.

NAME

textrel – Beschreibung des textrel-Formates

BESCHREIBUNG

textrel ist ein ASCII-Format, das zum Austausch von Daten in Form von Datentabellen (Relationen) dient. Im Gegensatz zum rein relationalen Ansatz wird eine Datenbasis als Liste mehrere verschiedener Relationen behandelt. Dadurch können auch kompliziertere Datenbestände einheitlich gehandhabt werden. (Dahinter steht der Ansatz, beliebige Formen durch verschiedene Rechtecke zu approximieren.)

Einheit ist eine Datei, die mehrere Textrelationen enthält.

Format der ASCII-Datei:

Die Dateien bestehen als einzelnen Worten, d.h., die Datenfelder sind durch 'white-space' Zeichen getrennt, und stehen nicht an festen Spaltenpositionen.

Kommentare sind überall erlaubt, sie beginnen mit dem '#'-Zeichen und gehen bis zum Zeilenende.

Relationen werden mit dem '>'-Zeichen eingeleitet, danach folgen die Domänen-Bezeichnungen mit ihren Typen, diese sind abgetrennt durch den Slash '/'. Erlaubte Typen siehe unten.

Jedes Datentupel muß in einer eigenen Zeile stehen. Die Domänenwerte (auch: Feldinhalte, Felder) sind Textstrings und werden in der Reihenfolge ihres Auftretens den im Header vereinbarten Domänen zugeordnet. Sind weniger Werte als Domänen da, sind die restlichen Werte Leerstring, mehr Werte als Domänen werden weggeworfen. Die Werte sind zwar Textstrings, werden aber als Werte des mit der Domäne angegebenen Feldtypes interpretiert.

Da die Feldwerte auch 'white-space'Zeichen enthalten können, können Werte auch mit Quotes ('"') umgeben werden. Das Escape '\' escaped Spaces, Quotes, und sich selber.

Siehe folgendes Beispiel:

```
# Kommentare sind überall erlaubt
# Sie gehen vom '#'-Zeichen bis zum Zeilenende
#
# Der Anfang einer Relation wird mit dem '>' markiert...
 >  Marke    Typ     Besitzer
    Daimler  190     Willi
    Opel     Kadett  "Lieschen Mueller"  # Feldinhalt mit Space
    Ferrari  ""      Onassis             # Leerstrings mit ""
    VW       Passat  Otto\ Maier         # Space kann auch escaped werden.
# Die naechste Relation in dieser Datei.
# Hier wird mit Datentypen gearbeitet, s.u.
 >  Datum/D  Messwert/r  Messgeraet/eGeraet
    1.1.89   12.3        TCS
    2.2.90   12(3)       TCB
    6.4.63   <<<         XSV
```

Datentypen:

Es gibt folgende Datentypen:

Name	Codezeichen	Bedeutung
`char`	c	1 Zeichen
`short`	s	16bit Integer
`integer`	i	short oder long
`unsigned`	u	ganze Zahl
`long`	l	32bit Integer
`float`	f	32bit-reelle Zahl
`double`	d	64bit-reelle Zahl
`string`	S	Zeichenkette, Schreibung egal. (Defaulttyp)
`text`	t	Zeichenkette, Schreibung egal. (wie string)
`Text`	T	Zeichenkette, Groß/klein NICHT egal
`lfloat`	r	32bit reelle Zahl mit Genauigkeit u. Sonderzuständen
`ldouble`	R	64bit reelle Zahl, dto.
`datum`	D	Tagesangabe
`zeit`	Z	Zeitangabe
`enum`	e*Name*	selbstdefinierter Aufzählungstyp, s.u.

Selbstdefinierte Aufzählungstypen:

Eigene Konstantenmengen können als Aufzählungstypen dem System zur Verfügung gestellt werden. Die eigenen Typen werden in einer textrel-Datei vereinbart, mit den Domänen

`code`, Typenname und evtl. `sort`, welches eine Sortierreihenfolge unabh. von den Codes definiert.

Beispiel:

```
# Hier wird der Aufzählungstyp 'eMarke' vereinbart.
# er wird nach 'code' sortiert
 >  code  Marke
    10    Lada
    20    Fiat
    40    VW
    50    Daimler
    45    BMW
# 'eTyp', alphabetisch.
 >  code  Typ     sort
    10    Kadett  10
    20    Corsa   6
    30    Panda   20
    40    Uno     30
```

Die Aufzählungsreihenfolge ist egal. Der Name der Typendatei wird aus der Environmentvariable 'ENUMTYPES' gelesen, wenn so nicht verfügbar, wird 'enums.dat' aus dem aktuellen Directory genommen.

Utilities:

Es wird eine Vielzahl von Utilities zur Behandlung von textrel-Daten zur Vefügung. Diese nennen wir 'Operatoren', und ihre Namen beginnen alle mit 'tr'. Die Operatoren empfangen Daten i.A. über die Standardeingabe und leiten ihre Ausgabe in die Standardausgabe und in die Fehlerausgabe.

OPTIONEN

Folgende Optionen werden von allen Operatoren benutzt::

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

`-t` Selbsttest.

DATEIEN

textrel_t.dat – Testdatei für viele Operatoren.
trenums_t.dat – Typen dafür.

ENVIRONMENT

ENUMTYPES – Name der Datei für selbstdefinierte Aufzählungstypen.

NAME

tr2text – konvertiert Textrelation in Strom von Worten

SYNOPSIS

tr2text `-l -sh -i`*infile* `-o`*outfile*

BESCHREIBUNG

tr2text konvertiert Textrelation in einen Strom von Worten, die als normaler Text ausgegeben werden, indem u.a. die Überschriftenzeile weggelassen wird.
Dies ist nützlich, wenn die Daten von anderen Textfiltern verarbeitet werden sollen.

OPTIONEN

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-l` erzeuge lange Ausgabewörter: 1 Ausgabewort = 1 Zeile(=Tupel). Default ist ein Wort pro Feld.

`-sh` escape Sonderzeichen so, daß die **sh**-Shell die Wortgrenzen mit

```
eval set -- `tr2text ...  `
```

akzeptiert.

DATEIEN

textrel_t.dat

FEHLER

keiner

NAME

tr2wk3 – konvertiert Textrelation ins Lotus 1-2-3-Format

SYNOPSIS

tr2wk3 `-n -i`*infile* `-o`*outfile*

BESCHREIBUNG

tr2wk3 konvertiert eine Textrelation in eine `.WK3`-Datei, wie sie von Lotus 1-2-3 verwendet wird. Dabei bleibt die tabellarische Struktur der Textrelation erhalten.

Die Domänennamen erscheinen als Spaltenüberschriften, die textrel-Datentypen der Domänen werden so gut wie möglich in die Datentypen von Lotus 1-2-3 umgewandelt.

Datenkonversion

Wenn die Option `-n` nicht gesetzt ist, werden die textrel-Typen wie folgt umgewandelt:

textrel	Lotus 1-2-3
`typerr` (Fehlerdarstellung)	Label
`notyp` (kein Typ)	Label
`char`	Label
`short`	Number
`int`	Number
`unsigned`	Number
`long`	Number
`float`	Number
`double`	Number
`string`	Label
`text`	Label
`Text`	Label
`lfloat`	Label
`ldouble`	Label
`datum`	Label
`zeit`	Label
`enum`	Label

Die Darstellung von `lfloat` und `ldouble` wird den Lotus-Konventionen für Zahlen (Numbers) angepaßt.

- Ohne die Option `-n` wird aus der Zahl ein 'Label', aber ohne explizite Präzisionsangabe, und statt Dezimalpunkt mit Komma.
- ist `-n` gesetzt, werden die Zahlen in 'Number' konvertiert. Die Präzision geht verloren, bestimmte Ausnahmezustände werden in best. Zahlen gewandelt:

Bedingung	Zahlendarstellung
`<<<` (unter der Nachweisgrenze)	1E-18
`***` (nicht gemessen)	1E100

  Es soll gewährleistet sein, daß ungültige Messungen sich als nicht ignorierbar hohe Werte bemerkbar machen.

tr2wk3 wurde zusammen mit Klaus Bartels (Prager Schule) entwickelt.

OPTIONEN

`-n` steuert die Umwandlung von 'labor-reals' ins Lotus-Format, s.o.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

GRENZEN

- nur die erste Relation in der Datei wird konvertiert.
- PC-basierte Lotus-Versionen können zu große `.WK3`-Dateien wg. Speichermangel nicht einlesen.

NAME

traggr – Aggregationen über Textrelationen

SYNOPSIS

traggr *Domänen ...* `-f -l -p -v -a`*FormelA* `-b`*FormelB* `-c`*FormelC* `-i`*infile* `-o`*outfile* `-e`*errfile*

BESCHREIBUNG

traggr verfügt über eine Domänenmenge; es zieht sich von der Textrelation immer solche Blöcke von Tupeln rein, bei denen alle angegebenen Domänen den gleichen Wert haben. Für jeden Block wird nur 1 Tupel ausgegeben, dabei werden pro Domäne mehrere Werte zu einem Wert verrechnet. Dies geschieht über 3 Formeln:

a : wird bei Blockbeginn ausgewertet.

b : wird bei jedem neuen Tupel ausgewertet.

c : wird bei Blockende ausgewertet.

Da die Formeln auch Reihungen zulassen, können mehrere Domänen geändert werden. Das Ausgabetupel wird mit dem ersten Tupel eines Blockes initialisiert.
Die Namen der Ausgabedomänen sind normal, die des momentanen Tupels beginnen mit '_'. Besondere Variable:

`_i` : Nr des momentanen Tupels im Block, am Schluß (in Formel C) Gesamtanzahl.

Die Namen der Domänen des Eingabetupels haben vorne ein '_'. Alle anderen Namen beziehen sich auf Domänen des Ausgabetupels. Variable mit '_' am Anfang sind 'readonly'. Zuweisungen an neue Variablen erzeugen neue Domänen im Ausgabetupel.

Formelsyntax

Die Formeln müssen den Regeln des C-Formel-Interpreters 'cinter' genügen. Besonderheiten:

- Der Ausdruck „ *Domänenname*[*Ausdruck*]“ liefert den Wert der Domänen aus dem ersten Tupel des jeweiligen Eingabeblocks, für das *Ausdruck* ungleich 0 ist. Damit lassen sich index-ähnlich alle Tupel des Eingabeblocks ansprechen, nicht nur das aktuelle.
- : Alle Namen in so einem Ausdruck müssen mit '_' beginnen, s.o.

- Tupel, bei deren Berechnung 'cinter' Fehler moniert, werden nach *errfile* geschrieben.

OPTIONEN

Domänen ... eine Liste von Domänennamen, die zur Blockeinteilung alle gleiche Werte haben müssen. Default: alle Domänen der Eingabe.

`-f` keine Berechnung, von jedem Block das erste Tupel zurückgeben.

`-l` keine Berechnung, von jedem Block das letzte Tupel zurückgeben.

`-p` gibt jedes Tupel aus, nicht nur die Aggregation.

`-v` 'verbose': zeige während der Rechnung alle Zugriffe auf Variable an.

`-a`*FormelA* Formel, die am Anfang jedes Blockes ausgewertet wird.

`-b`*FormelB* Formel, die für jedes Tupel des Blockes ausgewertet wird.

`-c`*FormelC* Formel, die am Ende jedes Blockes ausgewertet wird.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

BEISPIEL

Wir haben folgende Relation:

```
> Nr  PNr      MMethode  MWdhKey  Result
  1   B100/88  Al        1        1.6
  2   B100/88  Al        2        1.5
  3   B100/88  Al        3        1.1
  4   B100/88  Mg        1        0.8
  5   B100/88  Mg        2        1
  6   B109/88  Al        2        0
  7   B109/88  Al        1        0.5
  8   B109/88  Al        1        99
  9   B109/88  Al        2        82
  10  B109/88  Al        3        33
```

Der Aufruf

```
traggr PNr MMethode '-aSumme=0,Count=0' '-bCount++,Summe+=_Result'
'-cResult=Summe/_i'
```

bildet in `Result` die Mittelwerte, und gibt in `Count` die Anzahl der verrechneten Werte an.

```
> Nr  PNr      MMethode  MWdhKey  Result  Summe  Count
  1   B100/88  Al        1        1.4     4.2    3
  4   B100/88  Mg        1        0.9     1.8    2
  6   B109/88  Al        2        42.9    214.5  5
```

DATEIEN

traggr_t.dat

NAME

trbundle – komprimiert einen Strom von Textrelationen

SYNOPSIS

trbundle `-i`*infile* `-o`*outfile*

BESCHREIBUNG

trbundle komprimiert einen Strom von Textrelationen. Es hat folgende Funktionen:

- Faßt alle gleichen Relationen zu einer zusammen, d.h., kombiniert alle Tupel gleicher Struktur zu einer Relation.
- Löscht aufeinanderfolgende identische Tupel.

Methode:
Jede Relation wird in einem eigenen Diskfile gesammelt, am Schluß werden sie wieder ausgegeben. Es gibt keine Begrenzung für die Zahl der gleichzeitig offenen Dateien.

OPTIONEN

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

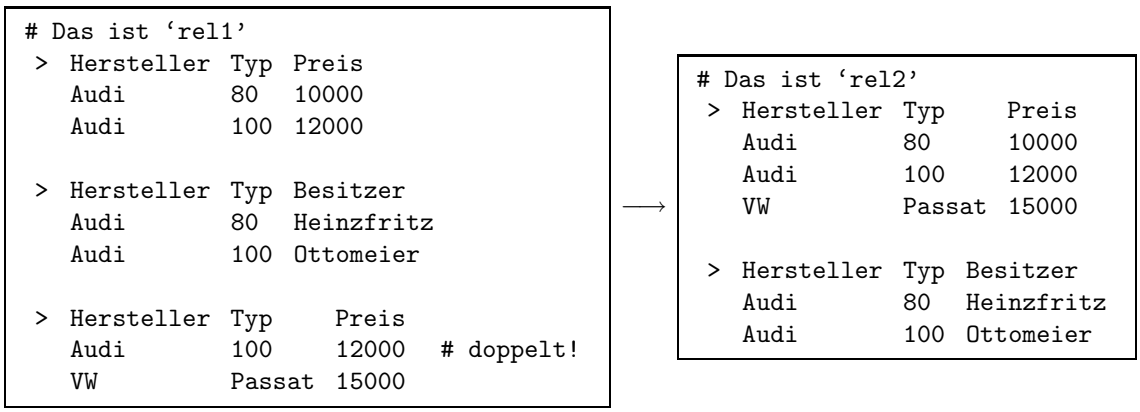
`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

Das Kommando

```
trbundle <rel1 >rel2
```

hat folgenden Effekt:



DATEIEN

trbundle.dat – Zum Testen
$TMPDIR/bundlexxxx – temporäre Dateien.

FEHLER

Keiner. Funktioniert es wirklich mit beliebig vielen Dateien?

NAME

trcalc – führt Berechnungen in Textrelationen durch.

SYNOPSIS

trcalc [*expr*] `-b -x -f`*Formelfile* `-i`*infile* `-o`*outfile* `-e`*errfile*

oder

trcalc `-f` *Textrel* `-d`*Doms* `-b -x -i`*infile* `-o`*outfile* `-e`*errfile*

BESCHREIBUNG

trcalc liest Textrelationen durch und wertet pro Tupel den Wert einer Domäne mit dem C-Formel-Interpreter **cinter** aus und ersetzt die Formel durch das Ergebnis.

Syntax und Möglichkeiten der Formelsyntax siehe 'cinter.c'/'cinter_t.c'/'cinter.h'

Die Zuweisung *DomName*=*Wert* wird ausgewertet, in dem die genannte Domäne auf den Wert gesetzt wird. Evtl. wird sie auch erst neu erzeugt!

Ist eine Formel in der Kommandozeile angegeben (*expr*), so wird sie anstelle des Domänenwertes für alle Tupel ausgewertet.

OPTIONEN

expr: werte für alle Eingabetupel *expr* aus. (Vorrang vor `-d` , `-f`).

`-d`*Doms* die genannten Domänen der Eingabe werden ausgewertet, und dadurch geändert. Ist eine der genannten Domänen nicht da, passiert auch nicht.

`-f`*Formelfile* werte pro Eingabetupel alle Ausdrücke der genannten Textrelation aus (nur die unter `-d`*Doms* angegebenen). (funktioniert das ?)

`-b` Boolean. Unterscheide Eingabetupel nach Wert der letzten Berechnung: Wenn Ergebnis != 0: Ausgabe nach *errfile*, sonst *outfile*. Treten irgendwelche Fehler auf, ist das Ergebnis auch FALSE.

`-x` eXpression: Ersetze Formel nach der Auswertung nicht durch das Ergebnis, sondern lasse sie stehen. Änderungen der Werte im Tupel gibt es dann nur durch Variablenzuweisungen in der Formel an andere Domänen.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

BEISPIEL

Das Kommando

```
trcalc -dDom2 <rel1 >rel2
```

hat folgenden Effekt:

```
# Das ist 'rel1'
 >  Dom1  Dom2          Dom3
    2     'Dom1 + 1'    5
    4.01  'Dom3 * 0.1'  1.1
```



```
# Das ist 'rel2'
 >  Dom1  Dom2  Dom3
    2     3     5
    4.01  0.11  1.1
```

DATEIEN

trcalc_t.dat, (trcalc_t.out, trcalc_t.err)

FEHLER

Irgendeine der Optionen ging doch nicht richtig. War es `-f` ?

Es werden nur die standard-**cinter** Datentypen unterstützt, nicht die **tr**-Datentypen Datum, Zeit und enums.

NAME

trcat – Textrelationen verketten

SYNOPSIS

trcat `-i`*infile* `-o`*outfile*

BESCHREIBUNG

trcat gibt die Daten der Eingabe als formatierte Textrelationenliste aus.

OPTIONEN

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

```
cat datei1 datei2 | trcat >datei3
```

ist die korrekte Sequenz zum Verbinden 2er Textrelationen.

FEHLER

keiner.

NAME

trcheck – Filter für Textrelationen

SYNOPSIS

trcheck *Regelfile* `-i`*infile* `-o`*outfile* `-e`*errfile*

oder

trcheck `-y` `-i`*infile* `-o`*outfile* `-e`*errfile*

BESCHREIBUNG

trcheck gibt nur die Tupel der Eingabe zurück, die den in mehreren 'Regelrelationen' definierten Regeln genügen; alle anderen Tupel landen im Fehlerausgang.

Eine zweite Option checkt auf Domänenwerte, die nicht ihren Domänentypen entsprechen.

Ein Tupel kommt durch, wenn es jeder Regelrelation in der Regeldatei genügt. Es genügt *einer* Regelrelation, wenn

- Es keine gemeinsamen Domänen zwischen Eingabe und Regelrel. gibt.
- Es min. ein Tupel in der Regelrel. gibt, so daß
  - alle gemeinsamen Domänen in Eingabetupel und Regeltupel denselben Wert haben.
  - (neu) für alle gemeinsamen Domänen sich die Domäne des Eingabetupels dem regulären Ausdruck n der Regel entspricht. ('.' = 1 bel. Zeichen, '.*' = 0..n Zeichen, '[...]' = Zeichenmenge. usf. (siehe 'regex.c').
    Reguläre Ausdrücke werden durch ein '~' eingeleitet. Achtung: '\(' muß als '\\(' vorkommen (textrel-Ebene).

OPTIONEN

Regelfile die Datei mit den Regeldaten.

`-y` kein Match gegen Regeldatei, sondern Typenüberpprüfung.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

BEISPIEL

Es soll geprüft werden, ob für jede 'PArt' auch die richtige 'AKarte' zugeordnet ist. Die Regelrelationen sehen dann so aus:

traditionell

```
# gueltige Werte fuer 'PArt'
 > PArt
   B
   P
   W
# gueltige Werte fuer 'AKarte'
 > PArt
   AKarteB
   AKarteP
   AKarteW
# gueltige Kombinationen
 > PArt  AKarte
   B     AKarteB
   P     AKarteP
   W     AKarteW
```

mit regulären Ausdrücken

```
# gueltige Werte fuer 'PArt'
 > PArt
   ~[BPW]
# gueltige Werte fuer 'AKarte'
 > AKarte
   ~AKarte[BPW]
# gueltige Kombinationen
 > PArt  AKarte
   B     AKarteB
   P     AKarteP
   W     AKarteW
```

DATEIEN

trcheck_t.dat – Testdaten.

trcheck_t.rdt – Testregeln.

trenum_t.dat – Enum-Typen zum Test.

trcheck_t.ok – Erfolgs-Daten des Selbsttests.

trcheck_t.err – Fehlerergebnis des Selbsttests.

DIAGNOSE:

Für die Tupel, die beim Vergleich mit Regeldaten durchgefallen sind, wird bei der ersten konfliktuöse Regel jeweils ein Fehlertext in der (neuen) Domäne `Error` erzeugt:

```
trcheck/Regel <Nr der Regel in Datei>:  <Domänen der Regel>
```

Bei Typenfehler gibt es die Meldung

```
trcheck:  Typenfehler <Text> in Domaene <Name>
```

oder

```
trcheck:  Typenfehler in Domaene <Name> ab <Resttext>
```

SIEHE AUCH

trjoin, wg. Optimierungen.

FEHLER

Veraltet.

Im Gegensatz zu **trjoin** kein optimierender Vergleich über sortierte Zugriffe, daher steigt die Rechenzeit quadratisch mit der Relationenlänge.

trjoin hat daher eine Option, um die Grundfunktion von **trcheck** ebenfalls durchzuführen!

NAME

trcolsin – ASCII-Tabellen nach Textrelationen wandeln

SYNOPSIS

trcolsin `-s` *Domäne von-bis ...* `-i`*infile* `-o`*outfile*

BESCHREIBUNG

trcolsin wandelt einen ASCII-Text, die in festen Spalten bestimmte Werte enthält, in eine Textrelation.

Eine Zeile des Textes erzeugt ein Tupel.

Die zu erzeugenden Domänen werden zusammen mit der Information, in welchen Spalten die Daten dafür stehen, in der Kommandozeile angegeben.

Die Option `-s` setzt den „Synchron"-Modus: es werden dann die Werte nach Wortgrenzen eingelesen, die sich rechts von den angegeben Spalten befinden.

OPTIONEN

`-s` Synchronisiere Eingabefelder mit Wortgrenzen.

Domäne von-bis Pro Domäne muß der Name, gefolgt vom Spaltenbereich, angegeben werden. Domänennamen und Spaltenbereich können voneinander durch Kommas oder Spaces getrennt sein.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

```
trcolsin Name,1-10,Einkommen,11-99 <eingabe
```

ist ein Beispiel für die Aufruf-Syntax.

FEHLER

keiner.

NAME

trcompress – komprimiert eine bestimmte Domäne in Wertebereiche

SYNOPSIS

trcompress *DomName* `-l -i`*infile* `-o`*outfile*

BESCHREIBUNG

trcompress ist die Umkehroperation zu **trexpand**. Wenn möglich, werden aufeinanderfolgende Tupel, die bis auf die Domäne *DomName* gleich sind, zu *einem* Tupel reduziert, wobei alle aufgetretenen Werte der Domäne entweder als *von-bis*-Bereich oder als Komma-Aufzählung dargestellt werden.

OPTIONEN

DomName die Domäne, über die komprimiert werden soll.

`-l` erzwinge Darstellung als Komma-Liste, auch wenn *von-bis*-Bereich möglich wäre.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

Das Kommando

```
trcompress PlfdNr <rel1 >rel2
```

bewirkt

```
# Das ist Relation 1
 >  SerienNr  PArt  PlfdNr  PJahr
     W22/91     W     1201     91
     W22/91     W     1203     91
     W22/91     W     1205     91
     W22/91     W     1206     91
     W22/91     W     1207     91
```

⟶

```
# Das ist Relation 2
 >  SerienNr  PArt   PlfdNr    PJahr
     W22/91     W      1201       91
     W22/91     W      1203       91
     W22/91     W    1205-1207    91
```

Die Daten sollten so sortiert sein, daß ähnliche Tupel auch zusammen liegen. Beispiel:

```
trsort -lPlfdNr <rel1 | trcompress PlfdNr >rel2
```

DATEIEN

trcompress.dat – Testdaten

trenum_t.dat – Aufzählungstypen für Test

SIEHE AUCH

trexpand

FEHLER

keiner.

NAME

trdoms – Aufbau von Textrelationen anzeigen

SYNOPSIS

trdoms *Filename*

BESCHREIBUNG

trdoms gibt die Domänen der Textrelationen zurück. Es wird 1 Zeile pro Relation ausgegeben, mit den Domänennamen, wie sie im Relationenheader vorkommen.

Die Anzahl der Zeilen ist gleichzeitig die Anzahl der einzelnen Relationen.

OPTIONEN

Filename : lies Eingabe aus *Filename*. Default: **stdin**.

FEHLER

keiner.

NAME

tred – Editor für Textrelationen

SYNOPSIS

tred *Filename* -b*Breiten* -c*Filename* -d*Doms* -fd*Directory* -fm -h*Doms* -pf *Wort,...* -r*Doms* -sc*StatusDom,Code,Code,...* -ss*StatusDom,Code* -sw*Zahl* -t *Titel* -v*Filename* -w*Doms* -wd*Directory*

BESCHREIBUNG

tred ist ein Editor, mit dem textrel-Dateien editiert werden können. Er umfaßt viele Sonderfunktionen und Modi.

Schirm-Layout:

Ganz oben zeigt **tred** einen optionalen Titel an. Die Schirmmitte nimmt eine Lotus-ähnliche Tabelle ein, in der ein Auschnitt aus der aktuellen Relation angezeigt wird. Über der Datentabelle sind die Domänen-Namen dargestellt. Unter dem Arbeitsblatt befindet sich abgesetzt ein Tupelpuffer, der für viele Bearbeitungsvorgänge Werte speichert. Im Arbeitsblatt werden immer alle Domänen dargestellt, es findet kein „horizontales Scrollen“ statt. Wenn der Platz nicht ausreicht, wird erst in enge Bildschirmdarstellung geschaltet, dann werden die Domänen nur noch verkürzt dargestellt.

Ganz unten schließlich steht eine Infozeile. Sie gibt u.a. an: Die Nr. der aktuellen Relation und die Gesamtanzahl aller Relationen („Tabelle“), dasselbe für Tupel („Zeile“) und Domänen („Spalte“). Dann folgt eine unverkürzte Anzeige der aktuellen Domänen und des aktuellen Feldinhalts.

In der Infozeile werden auch Warnungen angezeigt und Abfragen und Eingaben durchgeführt.

Arbeitsmodi:

tred verfügt über folgende Arbeitsweisen („Modi“):
Im *Move-Modus* kann man sich frei im Arbeitsblatt bewegen und die meisten Operationen durchführen, wie in und aus dem Puffer kopieren, Tupel suchen, die Marke setzen, Wechseln zwischen den Relationen, Dateien schreiben und lesen, etc. Man kann keine Eingaben machen (damit man nicht beim Rumfuhrwerken aus Versehen Werte ändert).

Im *Edit-Modus* kann man mit einem Zeileneditor den Wert des aktuellen Feldes ändern. Nach Eingabe findet eine Kontrolle statt, ob der Wert mit dem entsprechenden Domänentyp verträglich ist und zusätzlich kann der Wert gegen eine Muster-Liste verglichen werden (siehe Option `-v`). Nach Eingabe wird der Wert auch gleich in den Puffer kopiert.

Im *Select-Modus* schließlich kann man aus einer Auswahlliste einen oder mehrere Werte auswählen, die dann ins aktuelle Feld kopiert werden. Die angezeigten Werte können auf solche Werte begrnezt werden, die mit den Werten der Felder links im Tupel verträglich sind (s. Option `-c`).

Andere Funktionen:

tred bietet die Möglichkeit bestimmte Domänen in der Anzeige zu unterdrücken, bzw. gegen Eingaben zu schützen. Auch Tupel, die in einer bestimmten Domäne bestimmte Werte enthalten, sind gegen Eingaben geschützt (Option `-sc`).

tred wurde für VT220-Terminals geschrieben und benötigt deren Fähigkeit zur engen Bildschirmdarstellung und das erweiterte Tastenfeld.

OPTIONEN

Filename : Name der Datei, editiert wird.

`-b`*Breiten* : eine Liste von Ausdrücken „*Domäne=Zahl*“, die minimale Feldbreiten für die Anzeige vorgibt.

`-c`*Filename* : (Check) Datei mit einer Liste von Relationen, die gültige Wertkombinationen für bestimmte Domänen spezifizieren. Wird im Select-Modus verwendet.

`-d`*Doms* : Definition der Domänen, sie werden in der angebenen Reihenfolge angezeigt.

`-fd`*Directory* : Arbeitsverzeichnis für die Befehle Datei-Schreiben und Datei-Laden.

`-fm` : Formularmodus: Löschen und Einfügen von Tupel nicht erlaubt.

`-h`*Doms* : (Hidden) Domänen, die nicht mit angezeigt werden.

`-pf` *Wort,...* : Max. 4 Worte, die mit den Tasten PF1 bis PF4 eingegeben werden können.

`-r`*Doms* : (readonly) Domänen, die angezeigt, aber nicht verändert werden dürfen.

`-sc`*StatusDom,Code,Code,...* : (Status Check) Eingabe nur in Tupeln zulassen, in denen *StatusDom* keinen der angegebenen Werte hat.

`-ss`*StatusDom,Code* : (Status Set) Angabe über Statuskontrolle: In der Domäne *StatusDom* werden Veränderungen des Tupels registriert, indem *Code* dort hineingeschrieben wird, wenn irgendwo im Tupel eine Domäne geändert wird.

`-sw`*Zahl* : (screen width) Angabe der gewünschten Schirmbreite (für Terminaleinstellung).

`-t`*Titel* : Angabe des Titels, der über dem Arbeitsblatt agezeigt wird.

`-v`*Filename* (verify/format) Datei mit Textrelationen, die die gültigen Formate für die einzelnen Domänen enthalten. Die Datei muß die Domänen `DomName`, `Formats` und `Help` enthalten. Für jede zu prüfende Domäne gibt es ein Tupel, das in `DomName` den Namen der Domäne und unter `Formats` eine Kommareihung von mehreren Kriterien der Form "*Operator Wert*" enthält, die alle erfüllt sein müssen. Operatoren sind `<, <=, != >, >=, ~, !~`. *Operator* muß durch ein " " vom *Wert* getrennt sein. Unter `Help` kann ein Text stehen, der bei Regelverletzung angezeigt wird.

`-w`*Doms* : (write Doms) Domänen, die als einzige nicht „read-only" sind.

`-wd`*Directory* : (working directory) stellt das Arbeitsverzeichnis ein, für das Schreiben und Lesen von Dateien.

TASTENBELEGUNG

Es folgt eine Übersicht über die in den verschiedenen Modi erlaubten Kommandos:

Move-Modus:

Esc: Unterbricht unvollständige Eingaben.

↑: gehe auf dasselbe Feld des vorherigen Tupels.

↓: gehe auf dasselbe Feld des nächsten Tupels.

←: gehe auf das vorherige Feld im aktuellen Tupel, oder Sprung auf letztes Feld des vorherigen Tupels.

→: gehe auf das nächste Feld im aktuellen Tupel, oder Sprung auf ersten Feld des nächsten Tupels.

Next Page: Blättere eine Seite nach unten, zum Ende der Relation.

Prev Page: Blättere eine Seite nach oben, zum Anfang der Relation.

Enter: Gehe auf 1. Feld des nächsten Tupels.

Ctrl-L: Baue Schirmbild neu auf und berechne Anzeigebreiten der Domänen neu.

Ctrl-Z *Pfeiltaste*: Setze den 'Automove' auf die Richtung, die der *Pfeiltaste* entspricht. (Automove ist die Richtung, in die sich das aktuelle Feld bewegt, wenn im Edit-Modus erfolgreich eine Eingabe gemacht wurde).

[ ]: Gehe in den Edit-Modus.

Tab: Dto.

Ctrl-R: Gehe in den Select-Modus, und zeige nur Werte an, die sich mit den anderen Werten in der aktuellen Zeile vertragen.

Ctrl-F: gehe in den Select-Modus, und zeige alle Werte an.

Select Enter: Kopiere aktuelles Tupel in den Puffer.

Select Select: Kopiere aktuelles Feld in den Puffer.

Insert Enter: Aktuelles Tupel aus dem Puffer übernehmen.

Insert Insert: Aktuelles Feld aus dem Puffer übernehmen.

Select Insert: Mehrere Felder der aktuellen Domäne mit einem Wert füllen, und zwar insgesamt *Anzahl* Felder, beginnend mit dem aktuellen, und die entsprechenden Felder der nächsten Tupel, die im Abstand *Intervall* von einander liegen. Intervall, Anzahl und Wert werden abgefragt.

Insert ↓: Felder der aktuellen Domäne ab dem aktuellen Feld abwärts schieben. Am aktuellen Feld entsteht ein leeres Feld, das Feld des letzten Tupels geht verloren.

Insert ↑: Das aktuelle Feld löschen und die restlichen Felder hoch schieben.

Kommandos mit „großer“ Wirkung:

Ctrl-X Insert: Neues Tupel in die Relation einfügen, aktuelles Tupel runterschieben.

Ctrl-X Remove: Aktuelles Tupel aus der Relation löschen.

Ctrl-X ↑: An den Anfang der Relation gehen.

Ctrl-X ↓: Ans Ende der Relation gehen.

Ctrl-X Ctrl-W: Editor verlassen, Datei vorher speichern.

Ctrl-X Ctrl-C: Editor verlassen, Datei nicht speichern.

Ctrl-X S: Datei sichern. Dateiname wird abgefragt, es gilt das mit `-wd...` eingestellte Verzeichnis.

Ctrl-X L: Neue Datei laden, die alte vergessen. Dateiname wird abgefragt, Verzeichnis s.o.

Ctrl-X Prev Page: In die vorherigen Relation der Datei gehen.

Ctrl-X Next Page: In die nächste Relation der Datei gehen.

Select M: Das aktuelle Feld markieren.

Ctrl-G: Auf das markierte Feld gehen.

Select Find: Suche beginnen. Gesucht werden alle Tupel, die in einer Domäne einen bestimmten Wert haben oder einem bestimmten Ausdruck genügen. Suchausdrücke sind:
"$[<, <=, !=, >, >=]$ *Wert*", oder "$[\tilde{}, !\tilde{}]$ *Suchmuster*".
Domäne und Ausdruck werden abgefragt; ein '.' bei der Domäne bedeutet die aktuelle Domäne.
Gesucht werden kann: vom aktuellen Tupel nach unten ('u') oder nach oben ('o'), oder vom Anfang der Relation nach unten ('U') oder vom Ende nach oben ('O').

Find: Finde nächste Entsprechung der angegebenen Suche.

Select .: Setze Autokomma bei einer Domäne. 'Autokomma' heißt, daß nach Eingabe eines Wertes automatisch vor der $n$ten Stelle von rechts ein Dezimalpunkt eingefügt wird. Der Domänenname und n für die Domäne werden abgefragt. Mit '0' wird das Autokomma wieder abgestellt.

Edit-Modus:
Diese Tasten sind wirksam, wenn eine Texteingabe erfolgt:

←: in der Zeile ein Zeichen nach links.

→: in der Zeile ein Zeichen nach rechts.

Ctrl-A: an den Anfang der Zeile.

Ctrl-E: ans Ende der Zeile.

Ctrl-D: Position im Feld markieren, auf die der Cursor zu Beginn der Texteingabe springt.

← : nach links, und Zeichen dort löschen.

Remove: aktuelles Zeichen löschen.

Ctrl-C: die ganze Zeile löschen.

Ctrl-Z: Rest der Zeile löschen.

Insert: Wert aus Puffer in die Zeile kopieren.

Ctrl-R, Ctrl-F: s.o.

Tab: Eingabe bestätigen, Edit-Modus verlassen.

Esc: Eingabe verwerfen, Edit-Modus verlassen.

Enter: Eingabe bestätigen, und per 'Automove' auf das nächste Eingabefeld gehen.

↑: Eingabe bestätigen, und zur nächsten Eingabe hoch gehen.

[↓]: dto., runter gehen.

[Prev Page]: dto., nach links gehen.

[Next Page]: dto., nach rechts gehen.

[PF1] ... [PF4]: einen von 4 festen Werten eingeben. Die Werte werden in der Commandline bestimmt.

Select-Modus:
Diese Tasten sind wirksam, wenn die Liste mit auswählbaren Kürzeln angezeigt wird:

[←], [→], [↑], [↓]: Bewegung durch Werteliste.

[Prev Page]: gehe auf 1. Wert.

[Next Page]: gehe auf letzten Wert.

[Enter]: Bestätige Auswahl, und Schluß.

[Esc]: Abbruch.

Buchstaben: gehe auf 1. Wert, der mit den eingetippten Buchstaben beginnt (Suchsequenz).

[←]: lösche ein Zeichen der Suchsequenz.

[+]: markiere Wert für Auswahl.

[–]: nimm Markierung zurück.

[ ]: wechsle Markierung.

Die obengenannten Funktionstasten sind u.U. durch folgende Tasten ersetzt (beim PC wird der ZSTEM-Terminalemulator beschrieben):

VT220		PC		F-Tasten
[Select]	=	[Ende]	=	[F6]
[Find]	=	[Pos1]	=	[F7]
[Insert]	=	[Einfg]	=	[F8]
[Remove]	=	[Entf]	=	[F9]
[Prev Page]	=	[Bild↑]		
[Next Page]	=	[Bild↓]		
[PF1]	=	[Num Lock]		
...		...		
[PF4]	=	[–]		

ENVIRONMENT

TRED_RULES : Default-Regeldatei (siehe Option `-c`)
TRED_FORMATS : Default-Formatdatei (siehe Option `-v`)

FEHLER

keiner.

NAME

trexpand – expandiert Wertebereiche in mehrere Tupel

SYNOPSIS

trexpand *Domänen ...* `-r` `-a`*Domänen ...* `-n`*Domänen ...* `-c`*Domname* `-i`*infile* `-o`*outfile* `-e`*errfile*

BESCHREIBUNG

trexpand nimmt eine Textrelation und expandiert für jedes Tupel alle Domänenenwerte, die es als numerische *von-bis*-Bereiche oder als aufgezählte Listen vom Werten erkennt, in je einzelne Werte. So entstehen aus einem Eingangstupel mehrere Ausgangstupel.

Es gibt 2 Arten von Expansionen:

numerisch : wenn ein Feld die Form *num1-num2* hat UND einen Integer-Typ, so wird es in alle Zahlen *num1* <= n <= *num2* aufgelöst.

Aufzählung expandiert wird eine Reihe von Werte, die durch ',' getrennt sind.

trexpand erkennt selbst, welche Expansion es anwenden, muß, diese kann aber auch vorgegeben werden.

OPTIONEN

Domänen ... : Domänen, die in der angegebenen Reihenfolge expandiert werden sollen (die letzte läuft am schnellsten durch).

`-r` Expandiere auch regular expression, wenn endliche Grundmenge da (bei Aufzählungs-Typen).

`-a`*Domänen ...* Erzwinge bei diesen Domaenen Aufzählungsexpansion oder expandiere sie gar nicht. (leer = bei allen oben angegebenen Domänen).

`-n`*Domänen ...* Erzwinge bei diesen Domaenen numerische Expansion, oder expandiere sie gar nicht. (leer = bei allen oben angegebenen Domänen).

Sonst wird pro Feld automatisch entschieden, ob es nach `-a` oder `-n` behandelt wird.

`-c`*Domname* : (Count) Expandiere nicht, sondern schreibe in die (auch neue) Domäne *Domname* die Zahl der Tupel, die sich bei Expansion ergeben würde. Ist die Domäne schon vorhanden, wird die Zahl der Tupel mit dem alten Wert der Domäne multipliziert.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

BEISPIEL

Beispiel für numerische Expansion:

```
trexpand Dom3 <rel1 >rel2
```

bewirkt:

```
# Das ist 'rel1'
 >  Dom1  Dom2   Dom3
    1     a,b,c  10-11
    2     d      99-101
```

```
# Das ist 'rel2'
 >  Dom1  Dom2   Dom3
    1     a,b,c  10
    1     a,b,c  11
    2     d      99
    2     d      100
    2     d      101
```

Beispiel für Aufzählungsexpansion:

```
trexpand Dom2 <rel1 >rel2
```

bewirkt:

```
# Das ist 'rel1'
 >  Dom1  Dom2   Dom3
    1     a,b,c  10-11
    2     d      99-101
```

```
# Das ist 'rel2'
 >  Dom1  Dom2  Dom3
    1     a     10-11
    1     b     10-11
    2     c     10-11
    2     d     99-101
```

DATEIEN

trexpand.dat – Testdaten

trexpand.err – Fehlerausgabe für Test.

trenum_t.dat – Aufzählungstypen für Test.

DIAGNOSE

Folgende Fehler werden mit Text in die Fehlerausgabe geleietet:

`Domaene` *Name*`:` `Formatfehler` *Beschreibung* `ab` *Text*

Dies weist auf Werte hin, die mit ihrem Domänentyp nicht verträglich sind.

SIEHE AUCH

trcompress

FEHLER

keiner.

NAME

trformat – formatierte Ausgabe von Textrelationen

SYNOPSIS

trformat -gd*Doms* -ng*Doms* -zg*Doms* -zk*Doms* -zd*Doms* -jr*Doms* -jm*Doms* -jl*Doms* -jc*char,Doms* -pg*Zahl* -rs*Zahl* -ds*Zahl* -ez*Text* -wd*Zahl* -td*Filename* (-tst) -i*infile* -o*outfile*

BESCHREIBUNG

trformat bereitet Textrelationen für eine Präsentation auf. Es bietet folgende Möglichkeiten:

- die Ausgabe erfolgt mit einem Domänenkopf auf jeder Seite.
- die Daten werden in Gruppen von jeweils fester Anzahl ausgegeben.
- Die Daten können nach Sprüngen in bestimmten Domänen in Gruppen eingeteilt werden.
- bestimmte Domänenen können mittig, links- oder rechtsbündig oder mit einem Zeichen übereinander zentriert werden.
- aufeinanderfolgende Werte können durch Gänsefüßchen (") abgekürzt werden.
- es können Titelzeilen und ein Textvorspann mit ausgegeben werden.

trdoms wurde von Bernd Fink erstellt.

OPTIONEN

-gd*Doms* („GruppenDoms“) für die angegebenen Domänen werden gleiche Werte zu einer Gruppe zusammenfaßt. Wenn eine neue Gruppe beginnt, wird diese durch eine Zeile aus Gruppenzeichen (Option -zg*char*) getrennt. Ein um 1 grösserer Wert führt nicht zu einem Gruppenwechsel.

-ng*Doms* („NoGaense“) in den angegebenen Domänen werden auch bei aufeinanderfolgenden identischen Werten kein Folgezeichen („Gänsefüßchen“) ausgegeben; der Aufruf ohne Domänen schaltet die Option für alle Domänen ein.

-jr*Doms* angegebene Domänen werden rechts justiert.

-jl*Doms* angegebene Domänen werden links justiert.

-jm*Doms* angegebene Domänen werden mittig justiert.

`-jc`*Char,Doms* angegebene Domänen werden mit dem übergegebenen Zeichen übereinander ausgerichtet justiert. Insbesondere können mit '`-jc.`*Doms*' numerische Werte mit dem Dezimalpunkt aufeinander ausgerichtet werden.

`-pg`*Zahl* Zeilenanzahl pro Seite einstellen; default = 66.

`-ez`*Text* („ErsteZeile“) Überschrift, die auf jeder Seite erscheint. „###“ steht für die Seitennummer.

`-td`*Filename* („TitelDatei“) Datei am Anfang des Ausdrucks mit einfügen.

`-wd`*Zahl* Papierbreite einstellen.

`-rs`*Zahl* („RelationenSep“) Zeilenanzahl des Raums zwischen Relationen einstellen; default = 5.

`-ds`*Zahl* („DatenSep“) Anzahl der Datensätze bis Trennerausgabe einstellen; default = 10.

`-zg`*Char* („ZeichenGruppen“) Eine Zeile aus dem übergebenen Zeichen trennt die Gruppen. Default = Blanks. Ausschalten mit `-zg0`.

`-zk`*Char* („ZeichenKopfzeile“) Eine Zeile aus dem übergebenen Zeichen unterstreicht die Kopfzeile. Default = '='. Ausschalten mit `-zk0`.

`-zd`*Char* („ZeichenDaten“) Eine Zeile aus dem übergebenen Zeichen trennt bestimmte Anzahl von Datensätzen. Default = '-'. Ausschalten mit `-zd0`.

`-tst` Selbsttest.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

FEHLER

es geht beinahe alles.

NAME

triddoms – stellt identische Domänen fest

SYNOPSIS

triddoms -v*Val* -i*infile*

BESCHREIBUNG

triddoms gibt die Liste der Domänen nach **stdout** aus, deren Werte innerhalb aller Textrelationen der Eingabe konstant bleiben.

OPTIONEN

-v*Val* nur Ausgabe der Domänen, die alle gleich und deren Wert *Val* ist. Normale Anwendung: '-v' = alle Domänen, die durchgängig leer sind.

-i*infile* lies Eingabe aus *infile*. Default: **stdin**.

FEHLER

keiner.

NAME

trjoin – Textrelationen verknüpfen und erweitern

SYNOPSIS

trjoin *RegelRelation* `-1 -f -c -u -k`*KeyDomänen* `-w`*WertDomänen* `-i`*infile* `-o`*outfile* `-e`*errfile*

BESCHREIBUNG

trjoin verknüpft zwei Textrelationen über einen „Join“. Es wird eine Domänenmenge („KeyDomänen“) angegeben, und für jedes Tupel der Eingabe-Textrelation werden alle Tupel der anderen Relation („RegelRelation“) gesucht, die in den KeyDomänen dieselben Werte haben. Pro Match gibt es ein Ausgabetupel, daß aus den Domänen der Eingaberelation plus ausgewählten Domänen („WertDomänen“) der Regelrelation besteht. Eine Regelrelation wird nicht benutzt, wenn sie eine geforderte Keydomäne oder Wertdomäne nicht enthält. Gibt es zu einem Tupel keinen Match, wird es in die Fehlerausgabe geschrieben.

In der Regelrelation werden beim Match auch “regular expressions” ausgewertet. Die erlaubten Muster sind bei der Beschreibung von **tred** bei der Erklärung der Option `-v` beschrieben.

trjoin fügt also in Textrelationen neue Domänen hinzu, deren Inhalt von bestimmten anderen Domänen abhängt.

Welcher Wert in Abhängigkeit von welchen anderen Domänen eingesetzt wird, hängt von der Regelrelation ab, sowie von den ’KeyDomänen’, die im Aufruf übergeben werden. (diese wählen den/die Einträge aus, aus denen Information übernommen wird).

Funktion: Die Regelrelation wird komplett eingelesen, und pro neuer Relation der Eingabe nach den sich neu ergebenden KeyDomänen neu sortiert (Optimierung).

Die Option `-c` führt die Funktion von **trcheck** durch.

OPTIONEN

`-k`*KeyDomänen* Angabe der KeyDomänen. Möglichkeiten:

`-k`*Namen* : Keys sind für alle Tupel die genannten.

`-k` : keine Keys, Match mit jedem Regeltupel.

`-k*r` : Keys sind alle Domänen des aktuellen Regeltupels.

`-k*r` *Doms* : dto., aber beschränkt auf *Doms*. (ein Space vor *Doms* lassen!)

`-k*i`*:* Keys sind alle Domänen des aktuellen Eingabetupels.

`-k*i` *Doms* : dto., aber beschränkt auf *Doms* (ein Space vor *Doms* lassen!)

Default: Keys ist die jeweilige Schnittmenge der in der Eingabe und den Regeln vorkommenden Domänen.

`-w`*WertDomänen* : Die WertDomänen werden aus dem ausgewähltem Regeltupel an die Ausgabe angehängt. Besondere Ausdrücke:

`-w*` alle des ausgewählten Regeltupels.

`-w*`*Doms* : dto., beschränkt auf *Doms*.

`-1` Gib nur einen, den ersten Match pro Eingabetupel aus.

`-f` (freier Match): Match auch, wenn eine angegebene Keydomäne nicht im RegelTupel vorkommt (kein Match, wenn vorhanden mit anderem Wert).

`-c` Funktion wie **trcheck**:

`trcheck` *Regelrelation*

ist funktional identisch mit

`trjoin` *Regelrelation* `-c ,`

nur viel schneller.

`-u` (unprotect) Normalerweise werden Werte nur aus der Regelrelation in die Ausgabe übernommen, wenn dadurch keine Werte aus der Eingabe überschrieben werden. Die Option `-u` nimmt diesen Schutz zurück, es wird dann immer geschrieben.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

BEISPIEL

Hier kommen einige Anwendungen für **trjoin**. Wir benutzen immer folgende Eingabedatei:

```
# Eingabe-Daten 'dat'
 > PlfdNr   Programm  Aufschluss  Messung
   B120/91  BV1       AAS1        X1
   B120/91  BV1       AAS1        X2
   B120/91  BV1       AAS1        X3
   B120/91  BV1       AAS2        X1
   B120/91  BV1       AAS2        X2
   B120/91  BV1       AAS2        X3
   B120/91  BV1       AAS3        X3
```

Der Aufruf

```
trjoin regel1 -1 -kAufschluss,Messung -w -e/dev/null
```

gibt die gültige Kombinationen in der Eingabe zurück.

```
# Regel-Datei 'regel1':
# Erlaubte Kombinationen.
 >  Aufschluss  Messung
    AAS1         X1
    AAS1         X2
    AAS2         X1
    AAS2         X3
```

$\longrightarrow$

```
 >  PlfdNr   Programm  Aufschluss  Messung
    B120/91  BV1       AAS1        X1
    B120/91  BV1       AAS1        X2
    B120/91  BV1       AAS2        X1
    B120/91  BV1       AAS2        X3
```

Der Aufruf

```
trjoin regel2 -kVProg -wVSieb,VLager
```

expandiert die Programmkürzel in mehrere Tupel

```
# Regel-Datei 'regel2':
# Werte einiger Programme.
 >  VProg  VSieb    VLager
    BV1    nix      nix
    BV1    sieben1  lagern1
    BV2    siebx    lagerx
# wird ignoriert, da
#'VSieb' und 'VLager'
# gefordert
 >  VProg  VMisch
    BV1    misch1
    BV1    misch2
```

$\longrightarrow$

```
 >  PlfdNr   Prog  Auf.  Mess  VSieb    VLager
    B120/91  BV1   AAS1  X1    nix      nix
    B120/91  BV1   AAS1  X1    sieben1  lagern1
    B120/91  BV1   AAS1  X2    nix      nix
    B120/91  BV1   AAS1  X2    sieben1  lagern1
    B120/91  BV1   AAS1  X3    nix      nix
    B120/91  BV1   AAS1  X3    sieben1  lagern1
    B120/91  BV1   AAS2  X1    nix      nix
    B120/91  BV1   AAS2  X1    sieben1  lagern1
    B120/91  BV1   AAS2  X2    nix      nix
    B120/91  BV1   AAS2  X2    sieben1  lagern1
    B120/91  BV1   AAS2  X3    nix      nix
    B120/91  BV1   AAS2  X3    sieben1  lagern1
    B120/91  BV1   AAS3  X3    nix      nix
    B120/91  BV1   AAS3  X3    sieben1  lagern1
```

Der Aufruf

```
trjoin regel3 -1 -f -wInfo
```

bindet Formeln an die Tupel, wobei alle Domänen der Eingabe Key sind.

```
# Regel-Datei 'regel3':
# Einheiten der Werte.
 > Auf.  Mess.   Info
   AAS1  ~X[12]  AAS1-X1/2 kg
   AAS2  X1      AAS2-X1 gr
   AAS3  X3      AAS3-X3 kg

 > Info
   sonst Pfund
```

⟶

```
 > PlfdNr   Prog  Auf.  Mess  Info
   B120/91  BV1   AAS1  X1    AAS1-X1/2 kg
   B120/91  BV1   AAS1  X2    AAS1-X1/2 kg
   B120/91  BV1   AAS1  X3    sonst Pfund
   B120/91  BV1   AAS2  X1    AAS2-X1 gr
   B120/91  BV1   AAS2  X2    sonst Pfund
   B120/91  BV1   AAS2  X3    sonst Pfund
   B120/91  BV1   AAS3  X3    AAS3-X3 kg
```

Der Aufruf

```
trjoin regel4 -k -w*
```

erzeugt schließlich (da ohne Keys immer Match) das kartesische Produkt der beiden Relationenlisten.

GRENZEN

- Wertdomänen, die schon in der Eingabe vorkommen, werden nicht geändert; außer es wird die Option `-u` abgegeben.
- bei Einsatz von `-1` ist zu beachten, daß die Tupelreihenfolge der Regelrelation durch interne Sortiervorgänge verändert werden kann.

NAME

trmkwdh – Wiederholungsanforderungen berechnen.

SYNOPSIS

trmkwdh `-i`*infile* `-o`*outfile*

BESCHREIBUNG

trmkwdh liest eine Liste von Eintragen der DB mit kompletter Pfad-Information (ProbenID, PATH- und WDHKEY-Domänen). Es erzeugt daraus neue Pfadinformationen für Wiederholungs-Anforderungen ab einer Station, indem es die WDHKEYs der Station 1 hochzählt.

Ausgegeben werden die wirklich neuen Pfade.

OPTIONEN

Station Name der Station, ab der die angegebenen Pfade wiederholt werden sollen.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

(Es gehören `AMethode` und `AWdh` zur Station Aufschluss; und `MMethode` und `MWdh` zur Station Messung.)

```
trmkwdh Aufschluss <rel1 >rel2
```

```
# 'rel1'
 >  AMethode  AWdh  MMethode  MWdh
    A         1     B         1
    A         1     B         2
    A         2     B         1
    X         1     Y         2
```

⟶

```
# 'rel2'
 >  AMethode  AWdh  MMethode  MWdh
    A         3     B         1
    X         2     Y         2
```

(Die Daten von `MMethode` und `MWdh` sind hier ohne Bedeutung.)

```
trmkwdh Messung <rel1 >rel2
```

```
# 'rel1'
 >  AMethode  AWdh  MMethode  MWdh
    A         1     B         1
    A         1     B         2
    A         2     B         1
    X         1     Y         2
```

⟶

```
# 'rel2'
 >  AMethode  AWdh  MMethode  MWdh
    A         1     B         3
    A         2     B         2
    X         1     Y         3
```

(nur diese 3 sind wirklich neue Pfade)

DATEIEN

StationDef

FEHLER

keiner.

NAME

trmvdoms – Domänen-Layout umgestalten.

SYNOPSIS

trmvdoms -a*Domänen-Umbenennungen* -d*Domänen* -j|k*Domänen* -m*Domänen* -n*Domänen/Zuweisungen* -r*Domänen* -v*Domänen* -y*Domänen* -i*infile* -o*outfile* -e*errfile*

BESCHREIBUNG

trmvdoms dient dazu, Domänen zu löschen, neu hinzuzufügen, in ihren Typen zu ändern oder nur die Domänenreihenfolge zu ändern.

OPTIONEN

Bedeutung der Optionen (Bearbeitung in absteigender Priorität) :

-r*Domänen* (Remove) : Domänen, die auf keinen Fall ausgegeben werden.

-d*Domänen* : die Domänen der Ausgabe sind genau die angegebenen, in der angegebenen Schreibweise. Bringt ALLE Relationen im Eingabestrom auf das gleiche Format. (ohne Typen!)

-m*Domänen* (Mit) : löscht alle Domänen außer den angegebenen.

-v*Domänen* (Vorne): die angegebenen Domänen werden in der angegebenen Reihenfolge an den Anfang der Domänenliste der Textrelation bewegt (Umordnen).

-a*Domäne=NeuerName,...* (renAme): Den angeführten Domänen neue Namen (evtl. mit neuen Typen) geben.

-n*Domäne[=Val],...* (New) : Liste von Einträgen „*Domäne*“ oder „*Domäne=Val*“. Die Domänen werden neu hinzugefügt (wenn nicht schon da), und mit *Val* besetzt (wenn angegeben). Setzt Typen.

-y*Domänen* (tYpen) : setzt die angegebenen Domänen auf die mit ihnen angegebenen Typen. Nötig nach 'new'.

-j*Domänen* (Join): faßt die angegebenen Domänen zu einer zusammen, indem Namen und Werte mit dem „_“ zu einem Wort verbunden werden.

-k*Domänen* : trennt Bündelungen mit „_“.

-i*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

DATEIEN

trenum_t.dat
textrel_t.dat

FEHLER

Die `-k` Option ist nicht implementiert.

Die `-a` Option geht laut einer früheren Handbuchversion nicht richtig, der Fehler konnte aber nicht reproduziert werden.

Es ist nicht recht klar, welche Optionen gemeinsam verwendet werden dürfen; und in welcher Reihenfolge sie ggf. wirken.

NAME

trselect – Daten über Suchkriterium auswählen.

SYNOPSIS

trselect '*Domname Operator Expression*' `-d`*Domänen* `-i`*infile* `-o`*outfile* `-e`*errfile*

BESCHREIBUNG

trselect filtert bestimmte Tupel aus der Eingabe, entsprechend den über die Commandline angegebenen Suchkriterien.

Ein Suchausdruck hat die Form:
Domäne Operator Expression
(keine Spaces!)

Domäne wird verglichen. Kommt sie nicht vor, wird der Vergleich bei jedem Tupel als 'wahr' angenommen!

Operator kann einer der folgenden sein:

`<=`,`<`, `>`, `>=`: Ein Tupel ist ausgewählt, wenn der Wert der Domäne *Domäne* größer oder kleiner als die Konstante in *Expression* ist.

`==`, `!=` Ein Tupel ist ausgewählt, wenn der Wert der Domäne *Domäne* gleich oder ungleich der Konstanten in *Expression* ist. Die Art des Vergleichs richtet sich in jedem Fall nach dem Datentyp von *Domäne*.

`~`, `!~` Ein Tupel ist ausgewählt, wenn sich die Textdarstellung des Wertes der Domäne *Domäne* (nicht) mit dem Textmuster *Expression* deckt. (regular expression).

Tupel, die allen Kriterien entsprechen, kommen in die Ausgabe, alle anderen in die Fehlerausgabe.

Besonderheit: Der Ausdruck „*Domäne* `!~ .*`“ bewirkt, daß alle Tupel, die die Domäne enthalten, nach *errfile* geschrieben werden, der Rest nach *outfile*.

OPTIONEN

`-d`*Domänen* : Zusätzliches Kriterium: Nur Tupel mit genau den Domänen sind ok.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

`-e`*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

DATEIEN

trenum_t.dat – Test-Typen
textrel_t.dat – Test-Daten
trselect.err – Test-Fehlerausgabe

FEHLER

Der Auswahlausdruck '*Domname Operator Expression*' wird in einer anderen Syntax angegeben als die Formeln in **trcalc**.

NAME

trseper – löse Tabellen in einzelne Datenzeilen auf.

SYNOPSIS

trseper -n*Domäne* -m*Domäne* -d*Domänen* -i*infile* -o*outfile* -e*errfile*

BESCHREIBUNG

trseper macht aus einem Tupel mit mehreren Domänen mehrere Tupel mit einer Domänen. Es ordnet die Werte mancher Domänen unter einer neuen Domäne gesammelt neu an, und schreibt zur Unterscheidung der Werte die alten Domänennamen in eine andere Domäne.

Existiert keine der Domänen aus *Domänen*, so wird das Tupel nach *errfile* ausgegeben.

trseper kann auch zum Umnennen einer Domäne verwendet werden:

```
trseper -n<neuer Name> -d<alter Name>
```

Die hier durchgeführte Operation könnte auch 'untabelize' ('detabellisiere') heißen: Daten in Tabellenform werden in ausführliche Zeileform überführt.

OPTIONEN

-d*Domänen* : Domänen, die verschwinden sollen.

-n*Domäne* (neu) : Name der neugebildeten Domäne

-m*Domäne* : Domäne, die die alten Domänennamen enthält. Die neue Domäne hat den Datentyp, der mit ihr angegeben wurde.

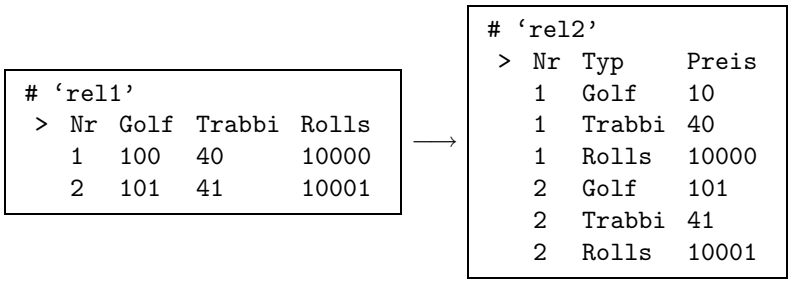
-i*infile* lies Eingabe aus *infile*. Default: **stdin**.

-o*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

-e*errfile* schreibe Fehlerausgabe nach *errfile*. Default: **stderr**.

BEISPIEL

```
trseper -dGolf,Trabbi,Rolls -nPreis -mTyp -irel1 -orel2
```



```
# 'rel1'
 >  Nr  Golf  Trabbi  Rolls
    1   100   40      10000
    2   101   41      10001
```

```
# 'rel2'
 >  Nr  Typ     Preis
    1   Golf    10
    1   Trabbi  40
    1   Rolls   10000
    2   Golf    101
    2   Trabbi  41
    2   Rolls   10001
```

Die drei Domänen `Golf`, `Trabbi` und `Rolls` verschwinden als eigene Domänen, ihre Werte erscheinen in der neuen Domäne `Preis`; und ihre alten Domänennamen erscheinen als `Typ`.

SIEHE EBENFALLS

trunsep

DATEIEN

trenums_t.dat – Testtypen.
textrel_t.dat – Testdaten.

SIEHE AUCH

trunsep

FEHLER

keiner.

NAME

trshrink, **trunshrk** – Daten komprimieren

SYNOPSIS

trshrink `-i`*infile* `-o`*outfile*

trunshrk `-i`*infile* `-o`*outfile*

BESCHREIBUNG

trshrink komprimiert die Textrelationen, indem es mehrere gleiche Werte in einer Domäne durch das besonderes Symbol '`@`' abkürzt (wie mit Gänsefüßchen).

Diese Dateien können dann mit anderen Archivprogrammen (z.B. **compress**) sehr effizient weiter komprimiert werden.

trunshrk stellt daraus wieder die orignalen Daten her.

OPTIONEN

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

```
trshrink <rel1 >rel2
```

```
# 'rel1'
 >  Dom1  Dom2
    aaa   bbb
    aaa   ccc
    bbb   bbb
    bbb   bbb
```

⟶

```
# 'rel1'
 >  Dom1  Dom2
    aaa   bbb
    @     ccc
    bbb   bbb
    @     @
```

FEHLER

keiner.

NAME

trsort – Textrelationen sortieren

SYNOPSIS

trsort *Domänen* `-l`*Domänen* `-nm` `-i`*infile* `-o`*outfile*

BESCHREIBUNG

trsort nimmt eine Liste von Textrelationen und sortiert die einzelnen Relationen darin, und löscht danach doppelte Tupel. Die Sortierreihenfolge der Domänen wird in der Commandline angegeben, die Ordnungsfunktion für die Werte einer Domäne hängt von ihrem Typ ab.

OPTIONEN

Domänen Angabe über die Sortierreihenfolge der Domäenen.

`-l`*Domänen* (least significant): Angabe von Domänen, die bei der Sortierung als letzte kommen (z.B. nötig für nachfolgendes **trcompress**).

`-nm` (no multis): mehrfach auftretende identische Tupel werden nicht gefiltert.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

```
trsort Dom2 Domx Dom1 <eingabe
```

bewirkt die Sortierreihenfolge: Dom2, Dom1, Dom3 (da Domx nicht vorhanden, aber Dom3).

```
# 'eingabe'
 > Dom1  Dom2  Dom3
   1     c     y
   02    a     y
   1     c     y
   3     b     z
```

⟶

```
# 'ausgabe'
 > Dom1  Dom2  Dom3
   02    a     y
   3     b     z
   1     c     y
```

Domänen, die in den Relationen nicht vorkommen, werden ignoriert.

FEHLER

keiner.

NAME

trsplit – Textrelationen auftrennen

SYNOPSIS

trsplit *Dateimaske* `-n`*Zeilenzahl* `-i`*infile*

BESCHREIBUNG

trsplit teilt einen große Textrelation in mehrere kleine Dateien auf. Eine neue Ausgabedatei wird immer bei einer neuen Relation begonnen, optional zusätzlich nach einer gewissen Zeilenanzahl.

OPTIONEN

Dateimaske Namensschema für die neu erzeugten Dateien. Ein „`%d`“ steht für eine automatisch hochgezählte Nummer.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

BEISPIEL

```
trsplit outfile%d.tr -n2 <eingabe
```

trennt die Eingabedatei in Dateien auf, die nicht mehr als zwei Datenzeilen (Tupel) enthalten.

```
# 'eingabe'
 > Dom1  Dom2  Dom3
   1     c     y
   02    a     y
   3     c     y
   4     b     z
```

⟶

```
# 'outfile1.tr'
 > Dom1  Dom2  Dom3
   1     c     y
   02    a     y
```

```
# 'outfile2.tr'
 > Dom1  Dom2  Dom3
   3     c     y
   4     b     z
```

FEHLER

keiner.

NAME

trstree – Baumdarstellung graphisch darstellen

SYNOPSIS

trstree `-wd`*Spalten* `-pg`*Zeilen* `-dd`*Domänen* `-i`*infile* `-o`*outfile*

BESCHREIBUNG

trstree nimmt eine geordnete Liste von Tupeln, die LAPIS-Pfade beschreiben, und malt daraus den Analysebaum.

- Die Ausgabe erfolgt seitenweise als Folge von ASCII-Bildern.
- Pro Wechsel der ersten Domäne wird ein neuer Baum gemalt.
- Bäume gehen nicht über Seitengrenzen (außer, sie gehen gar nicht auf ein Blatt).
- Datendomänen werden rechts neben den Baum geschrieben.
- die zur Verfügung stehende Blattbreite wird optimal genutzt.
- Am Kopf jeden Blattes steht eine Zeile mit den Domänennamen.
- Seitenvorschub mit “Formfeed”-Zeichen.

Der Kern von **trstree** wurde von Fr. Bäumer (Prager Schule) entwickelt.

OPTIONEN

`-dd`*Domänen* Domänen, die als Nutzdaten ganz rechts neben den Baum geschrieben werden

`-wd`*Spalten* : soviele Druckspalten stehen zur Verfügung. Default: 80.

`-pg`*Zeilen* soviele Zeilen pro Seite. Default: 66.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

```
trstree -ddEndwert <eingabe >ausgabe
```

```
# 'eingabe'
 >  PNr          AKArt  AMethode           MMethode  Endwert
    B_25923_92   AKB    AKE1.1             HH+1      0.13
    B_25923_92   AKB    AKE1.1             HH+2      0.14
    B_25923_92   AKB    AKE1.1             NaNages   1.12
    B_25923_92   AKB    AKE1.1             KKges     2.05
    B_25923_92   AKB    AKE1.1             FeFeges   9.744
    B_25923_92   AKB    AKE1.1             MnMnges   3.1
    B_25923_92   AKB    AKE1.1             MgMgges   2.9
    B_25923_92   AKB    AKE1.1             CaCages   1.115
    B_25923_92   AKB    AKE1.1             AlAlges   5.5
    B_25923_92   AKB    DAN1.1             PPges     6.6
    B_25923_92   AKB    pHCaCl2/0.01/1.1   HH+       0.02
    B_25923_92   AKB    ANULL              NNges     176.76
    B_25923_92   AKB    ANULL              CCges     188.09
    B_25923_92   AKB    ANULL              SSges     55.5
```

```
     PNr         AKArt       AMethode                MMethode      Endwert
-------------------------------------------------------------------------
                                              ,-----> HH+1          0.13
                                              +-----> HH+2          0.14
                                              +-----> NaNages       1.12
                                              +-----> KKges         2.05
                          ,-> AKE1.1 ---------+-----> FeFeges       9.744
                          |                   +-----> MnMnges       3.1
                          |                   +-----> MgMgges       2.9
                          |                   +-----> CaCages       1.115
 B_25923_92 ----> AKB - --+                   '-----> AlAlges       5.5
                          +-> DAN1.1 ----------->  PPges         6.6
                          +-> pHCaCl2/0.01/1.1 ->  HH+           0.02
                          |                   ,-----> NNges       176.6
                          '-> ANULL ----------+-----> CCges       188.09
                                              '-----> SSges       55.5
```

FEHLER

Dieses Programm existiert nicht in einer lauffähigen Version.

NAME

trtypchk – Typenprüfung von Feldinhalten

SYNOPSIS

trtypchk -i*infile* -o*outfile* -e*errfile*

BESCHREIBUNG

trtypchk liest eine Liste von Textrelationen und formatiert die Feldinhalte entsprechend den Domänentypen neu.

Treten in einem Tupel fehlerhafte Feldinhalte auf, wird das Tupel mit einer Beschreibung aller Fehler in die fehlerausgabe geschrieben.

OPTIONEN

-i*infile* lies Eingabe aus *infile*. Default: **stdin**.

-o*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

-e*errfile* schreibe Ausgabe nach *errfile*. Default: **stderr**.

BEISPIEL

```
# 'eingabe'
 >  Dom1/i  Dom2/T  Dom3/D
    1       cc      010195
    002     AA      020202
    -001    c       1.1.95
    3       b       1.1.1995
```

```
# 'ausgabe'
 >  Dom1/i  Dom2/T  Dom3/D
    1       cc      1.1.1995
    2       AA      2.2.2002
    -1      c       1.1.1995
    3       b       1.1.1995
```

FEHLER

keiner.

NAME

trunify – Textrelationen in einer Datei vereinen

SYNOPSIS

trunify -i*infile* -o*outfile*

BESCHREIBUNG

trunify macht aus mehreren Textrelationen ein große, mit allen vorkommenden Domänen. Die Tupelanzahl bleibt gleich. Die überzähligen Domänen werden mit „“ (Leerstring) gefüllt.

OPTIONEN

-i*infile* lies Eingabe aus *infile*. Default: **stdin**.

-o*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

```
trunify <eingabe >ausgabe
```

```
# 'eingabe'
 >  Marke  Typ
    VW     Golf
    VW     Passat
    Opel   Kadett

 >  Marke  Bemerkung
    VW     volksnah
    Opel   poplig
```

⟶

```
# 'ausgabe'
 >  Marke  Typ     Bemerkung
    VW     Golf
    VW     Passat
    Opel   Kadett
    VW             volksnah
    Opel           poplig
```

FEHLER

keiner.

NAME

trunsep – fasse Datentupel zu einer Tabelle zusammen.

SYNOPSIS

trunsep `-d`*BlockDomänen* `-k`*KeyDomänen* `-v` *WertDomäne* `-i`*infile* `-o`*outfile*

BESCHREIBUNG

trunsep macht aus mehreren Tupeln mit einer Domäne ein Tupel mit mehreren Domänen. Es ist die Umkehroperation zu **trseper**.

KeyDomänen: Domänen, aus deren Werten die neuen Domänen gebildet werden.
WertDomäne: Domäne, deren Werte den neuen Domänen zugeordnet werden.

Die Eingaberelation wird in Blöcke gegliedert, bei denen jeweils die sog. 'Blockdomänen' identisch sind. Jeder Block wird zu einem Tupel zusammengefaßt.

'Blockdomänen' sind entweder die angegebenen Domänen (Option `-d`), oder alle Domänen außer Key- und Wertdomänen.

Key- und Wertdomänen sind in der Ausgabe nicht mehr vorhanden.

Die Werte der 1. KeyDomäne bildet den Anfang der neuen Domänennamen, die Werte der folgenden KeyDomänen werden angehängt, wenn mehrere Tupel mit denselben KeyDoms vorkommen.

Alle Domänen der Eingaberelation, die nicht zu den Blockdomänen gehören und innerhalb eines Blocks für verschiedene Tupel verschiedene Werte annehmen, werden in der Ausgabe gelöscht (d.h. auf einen Leerstring gesetzt). Ist der Domänenwert für alle Tupel im Block gleich, wird er übernommen.

Die hier durchgeführte Operation könnte 'tabelize' ('tabellisiere') heißen: es werden nämlich explizit aufgeführte Kombinationen in einer Tabelle zusammengefaßt.

OPTIONEN

`-d`*BlockDomänen* : nur diese Domänen müssen für alle Tupel gleich sein, die in einen Block kommen.

`-k`*KeyDomänen* : Name der Domänen, aus deren Werten die neuen Domänen gebildet weren.

`-v`*WertDomäne* : Domäne, die die Werte für die neuzubildenden Domänen enthalten. Die neue Domänen haben den Datentyp der Domäne 'KeyDom'.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

Ordne Preise den Auto-Typen zu. Hat ein Typ mehrere Preise, unterscheide sie über 'Var'.

```
trunsep -vPreis -kTyp,Var <eingabe | trunify >ausgabe
```

```
# 'eingabe'
 >  Nr  Typ     Var  Preis
    1   Golf         100
    1   Trabbi       40
    1   Rolls   V1   10000
    2   Golf         1001
    2   Trabbi       401
    2a  Rolls   V1   100001
    2a  Rolls   V2   100002
```

⟶

```
# 'ausgabe'
 >  Nr  Golf  Trabbi  Rolls_V1  Rolls_V2
    1   100   40      10000
    2   1001  401
    2a                100001    100002
```

SIEHE AUCH

trseper

EINSCHRÄNKUNGEN

- Soll durch den Expansionsmechanismus eine Domäne erzeugt werden, die es schon gibt, so wird diese ignoriert (wie an anderer Stelle auch schon: lieber Funktionen unvollständig ausführen, als Daten verlieren).
- Programm bricht zusammen, wenn in einem Tupel der Eingabe alle Keydomänen leer sind (eigentlich sollten dann einfach keine neuen Ausgabedomänen erzeugt werden).
- Typenangaben der Keydomänen in der Eingabe könnten durchaus beachtet werden, wenn z.B. nur 1 Keydomäne angegeben wurde. Stattdessen haben die neuen Domänen immer den Typ der Wertdomäne (ist ja auch nicht verkehrt).

NAME

trwc – zählt die Tupel von Textrelationen

SYNOPSIS

trwc `-s -i`*infile*

BESCHREIBUNG

trwc zählt die Tupel in den Textrelationen. Es gibt für jede einzelne Relation die Zahl der Tupel nach **stdout** aus. Die Anzahl der Zahlen ist gleichzeitig die Anzahl der einzelnen Relationen.

OPTIONEN

`-s` : gib nur die Summe aller Tupel aus.

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

DATEIEN

trenum_t.dat – Testtypen
textrel_t.dat – Testdaten

FEHLER

keiner.

NAME

trxpose – transponiere Textrelationen.

SYNOPSIS

trxpose `-d`*Domänen* `-n` `-i`*infile* `-o`*outfile*

BESCHREIBUNG

trxpose transponiert jede der Textrelationen, d.h., es vertauscht Zeilen und Spalten, und erzeugt evtl. auch neue Domänen.

OPTIONEN

`-d`*Domänen* : Die neuen Domänenüberschriften sind die angegebenen.

`-d` Die Werte der ersten Spalte werden die neuen Domänen.

`-n` behandle die alten Domänennamen auch als Werte. (Sie landen dann in der 1. Spalte).

`-i`*infile* lies Eingabe aus *infile*. Default: **stdin**.

`-o`*outfile* schreibe Ausgabe nach *outfile*. Default: **stdout**.

BEISPIEL

```
trxpose '-dVar Val opt' <eingabe
```

```
# 'eingabe'
 >  Dom1  Dom2  Dom3
    a     b     c
    vx    vy    vz
```

⟶

```
# 'ausgabe'
 >  Var  Val  opt
    a    vx
    b    vy
    c    vz
```

```
trxpose '-dVar Val opt' -n <eingabe
```

```
# 'eingabe'
 >  Dom1  Dom2  Dom3
    a     b     c
    vx    vy    vz
```

⟶

```
# 'ausgabe'
 >  Var   Val  opt
    Dom1  a    vx
    Dom2  b    vy
    Dom3  c    vz
```

```
trxpose -d <eingabe
```

```
# 'eingabe'
 >  Dom1  Dom2  Dom3
    Var   b     c
    Val   vy    vz
```

⟶

```
# 'ausgabe'
 >  Var  Val
    b    vy
    c    vz
```

DATEIEN

trenum_t.dat – Testtypen.
textrel_t.dat – Testdaten.

FEHLER

keiner, aber lange nicht eingesetzt.