

Vlerq & Ratcl

Jean-Claude Wippler
Equi4 Software, NL

da•ta |'datə; 'dātə|

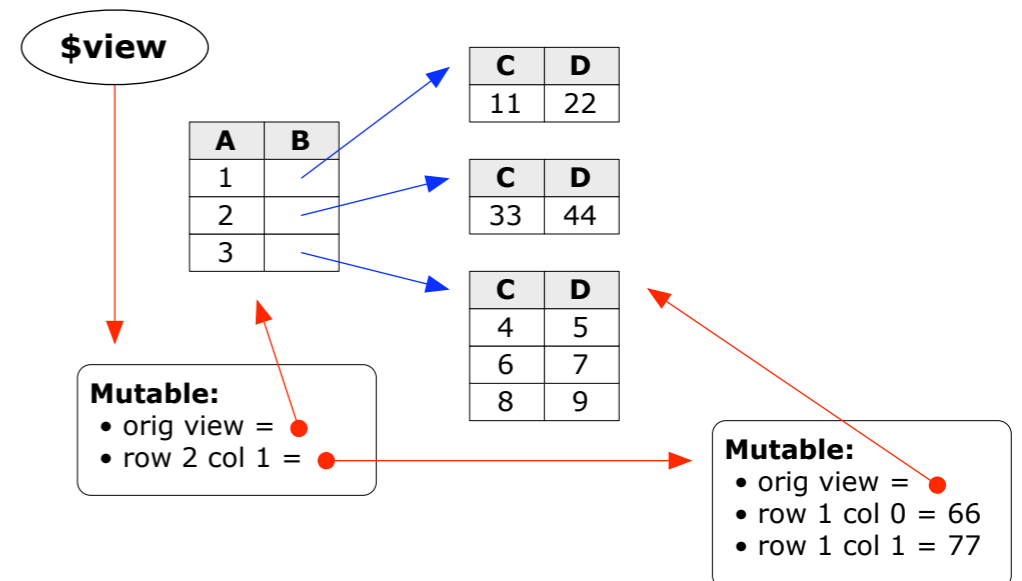
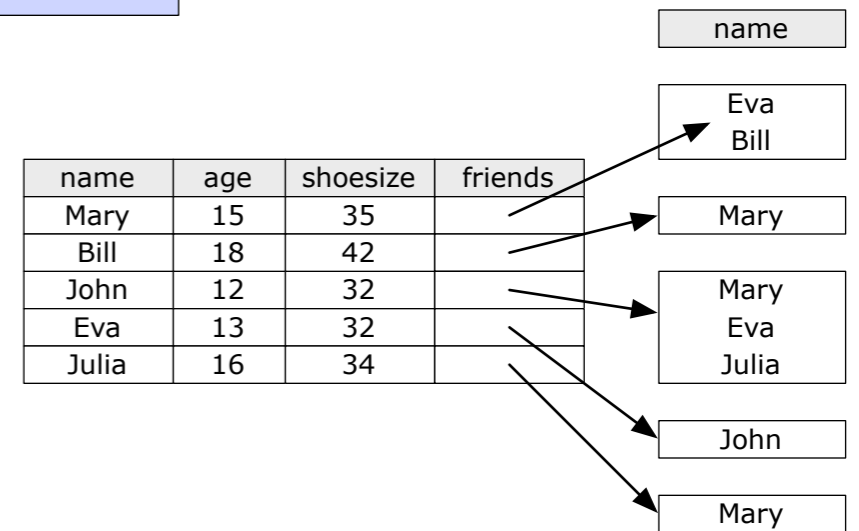
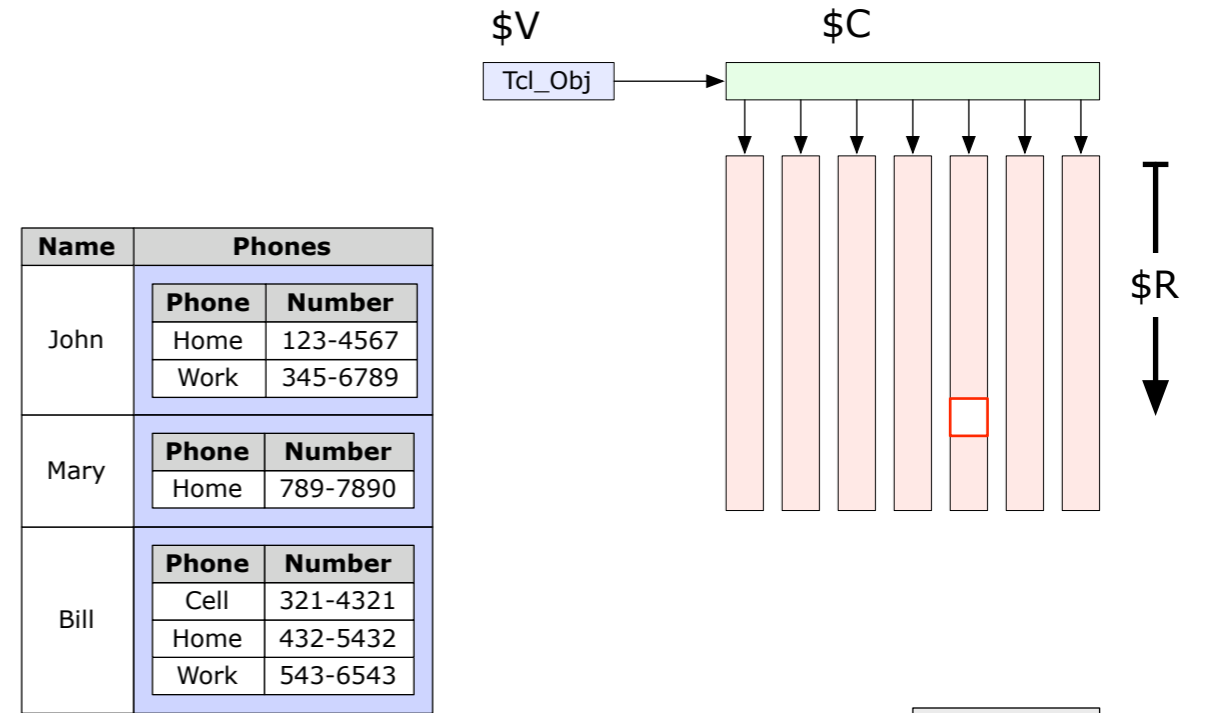
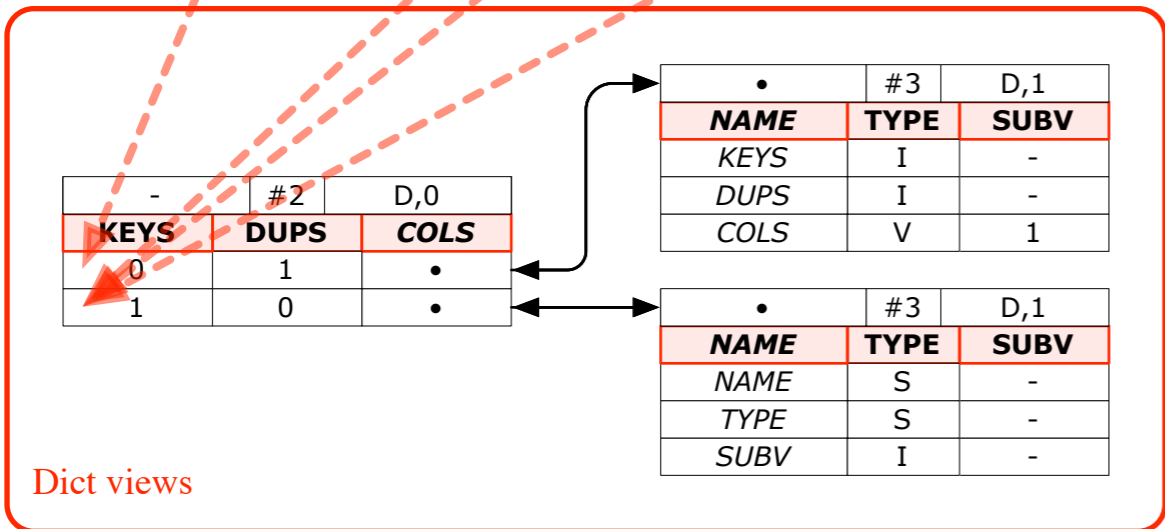
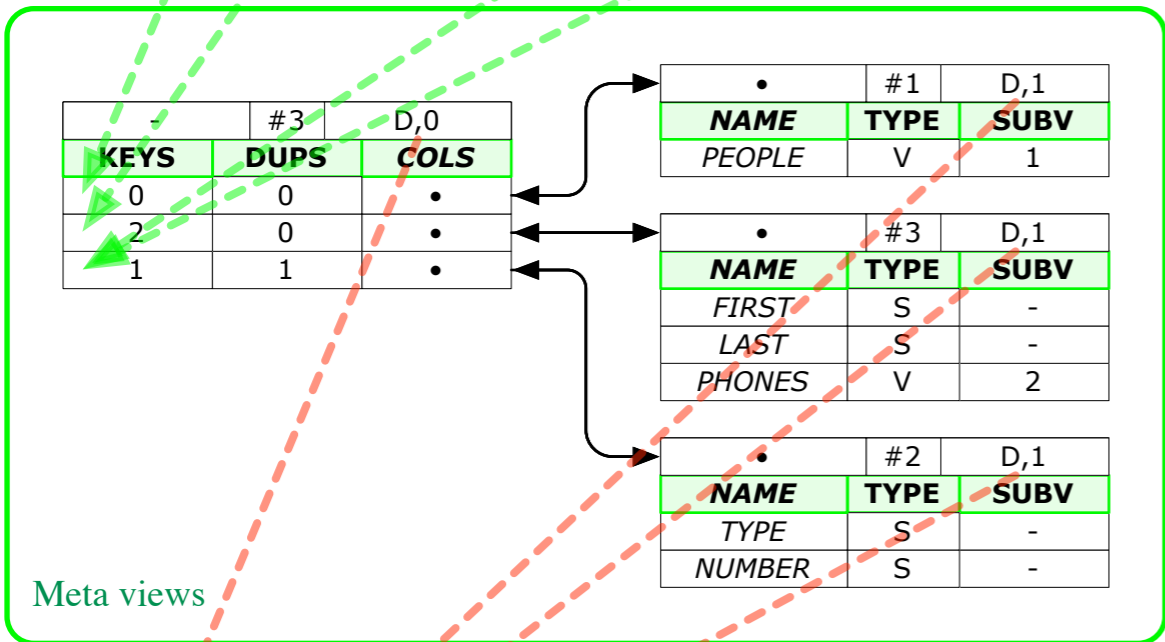
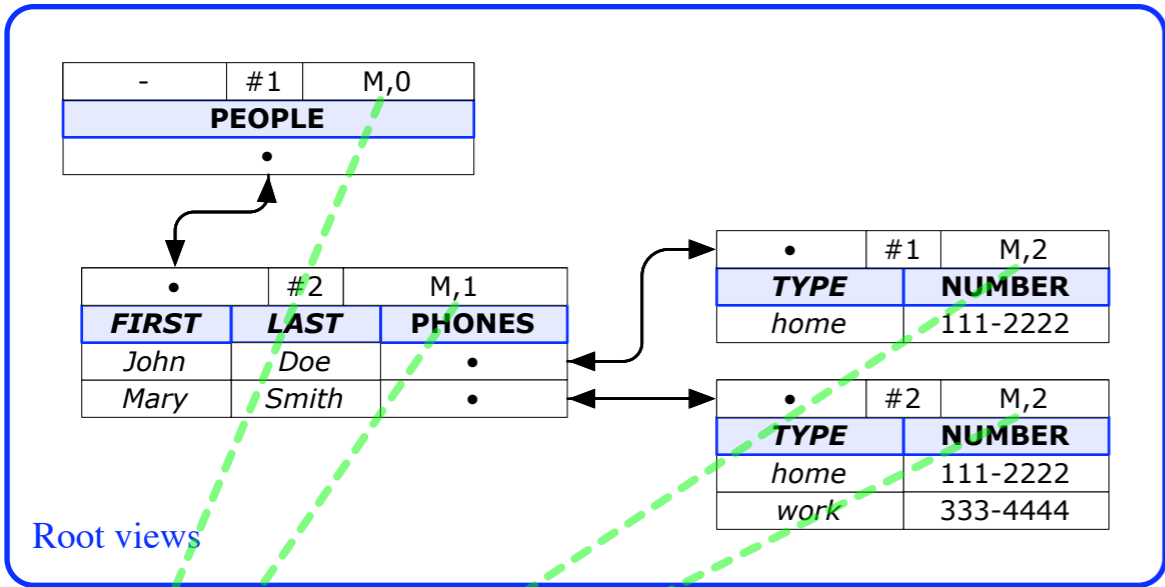
noun [treated as sing. or pl.]

facts and statistics collected together for reference or analysis. See also DATUM .

- Computing the quantities, characters, or symbols on which operations are performed by a computer, being stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media.
- Philosophy things known or assumed as facts, making the basis of reasoning or calculation.

ORIGIN mid 17th cent.(as a term in philosophy): from Latin, plural of DATUM .

USAGE **Data** was originally the plural of the Latin word: *datum*, 'something (e.g., a piece of information) given.' **Data** is now used as a singular where it means 'information': | *this data was prepared for the conference*. It is used as a plural in technical contexts and when the collection of bits of information is stressed: | *all recent data on hurricanes are being compared*. Avoid | *datas* and | *datae*, which are false plurals, neither English nor Latin.



Dealing with data

Tcl

lists, arrays, namespaces

each value ≥ 24 bytes, malloc'ed in memory

does not scale, can only dump/restore as strings

Set-wise operations

relational algebra: select, project, join

set union, intersection, exclusion

vector slice, reverse, concatenate

flair | fler |

noun

- 1 [in sing.] a special or instinctive aptitude or ability for doing something well : *she had a flair for languages* | *none of us had much artistic flair.*
- 2 stylishness and originality : *she dressed with flair.*

ORIGIN late 19th cent.: from French, from *flairer* 'to smell,' based on Latin *fragrare* 'smell sweet.' Compare with FRAGRANT .

Data in Vlerq

Views - everything is tabular

Column access by name or by position

Typed columns, optional

Nesting: views can contain sub-views

You've seen all this before: Metakit

Millions of rows, thousands of columns

rad·i·cal |'radikəl|

adjective

- 1** (esp. of change or action) relating to or affecting the fundamental nature of something; far-reaching or thorough : *a radical change*
 - forming an inherent or fundamental part of the nature of someone or something : *the assumption of radical differences between the two groups*
 - (of surgery or medical treatment) thorough and intended to be completely curative.
 - characterized by departure from tradition; innovative or progressive : *a radical approach to electoral reform.*
- 2** advocating thorough or complete political or social reform; representing or supporting an extreme section of a political party
 - (of a measure or policy) following or based on such principles.
- 3** of or relating to the root of something, in particular
 - Mathematics of the root of a number or quantity.
 - denoting or relating to the roots of a word.
 - denoting the semantic or functional class of a Chinese character.
 - Music belonging to the root of a chord.
 - Botany of, or springing direct from, the root or stem base of a plant.
- 4** [usu. as exclam.] informal very good; excellent : *Okay, then. Seven o'clock. Radical!*

noun

- 1** a person who advocates thorough or complete political or social reform; a member of a political party or part of a political party
- 2** Chemistry a group of atoms behaving as a unit in a number of compounds. See also FREE RADICAL . [ORIGIN: early 17c.]
- 3** the root or base form of a word.
 - any of the basic set of 214 Chinese characters constituting semantically or functionally significant elements in the classification of Chinese characters in dictionaries.
- 4** Mathematics a quantity forming or expressed as the root of another.
 - a radical sign.

Ratcl

Adds a command to Tcl to work with views

```
package require ratcl

view {Name Age} vdef {Joe 12 Mary 15} | to v

proc odd {x} {
    expr {$x % 2 == 1}
}

puts [view $v where { [odd $(Age)] } | dump]
```

Output:

```
Name  Age
----  ---
Mary  15
```

```
view 1000000 collect {int(rand()*10)} | \
  asview Rand:I | group Rand Counts | sort
```

```
Rand  Counts
----  -
```

0	#100151
1	#99956
2	#99750
3	#99707
4	#99559
5	#100710
6	#100065
7	#99823
8	#100007
9	#100272

```
view 1000000 debug {
  collect {int(rand()*10)}
  asview Rand:I
  group Rand Counts
  sort
}
```

rows-in	col	msec	view-operation
-----	---	----	-----
1000000	0	795	collect {int(rand()*10)}
		192	asview Rand:I
1000000	1	182	group Rand Counts
10	2	0	sort
10	2	1	dump
-----	---	----	-----

Why views?

Relational & set & vector operations

Loop-less programming - think “*wham!*”

Performance: million-row join is sub-second

Persistence for free: views can be on disk

Acts like in-memory, but it scales far beyond

And...

Dataflow

Changes propagate across view operations

```
% view $o | sort | to s  
% puts [view $s | dump]
```

```
A B  
- -  
1 2  
3 4
```

```
% view $o set 1 A 0
```

```
% puts [view $s | dump]
```

```
A B  
- -  
0 4  
1 2
```

So?

Selections, joins, sorts, ... become dynamic

Tcl traces can be tied to view changes

View-aware widgets: auto-update the GUI

Propagate over the net: free replication

Implication: views automatically track remote changes

Remember the spreadsheet?

Implementation

Vlerq is a C (not C++!) extension for Tcl

Ratcl is a wrapper script, written in Tcl

Data files are compatible with Metakit

Open source, MIT-license

Vlerq 3 out now, TEA3, 500+ tests

Vlerq 4 in progress, adding dataflow

Values

A view is a value, not a reference

Pass views around as arguments

Massive internal data sharing

Sharing includes disk: memory-mapped files

Efficient dual representation via Tcl_Obj's

The Tcl way

Values clean up automatically

No side-effects (copy-on-write)

Views tied to variables are mutable

Changes are tracked as compact difference sets

You can treat any view as a string

(that's usually not the most efficient approach)

Going deeper

A simple example

```
% puts [view {A B} vdef {3 4 1 2} | sort | dump]
A  B
-  -
1  2
3  4
```

With a surprise

```
% puts [view {A B} vdef {3 4 1 2} | sort]
sort {vdef {A B} {3 4 1 2}}
```

A view is a lazily-evaluated description

Dataflow ... how?

Tcl_Obj points to the string-like description

(small but crucial detail: it's a list, not a string)

Second pointer caches the Vlerq view

On changes, the cached data is invalidated

Next use from Tcl will cache the new view

Lots of dependency tracking going on

Thank you

for listening ...

Mike Doyle and Eolas Technologies Inc.

for funding all work on Vlerq since 2005

`www.vlerq.org`