Using Left-Right Trees for Hierarchic Data Storage

Version: 20 September 2011

Dale Chant, Roland Seidel,

Red Centre Software Pty Ltd

SSS Conference, Bristol, 2011

Red Centre Software
Data Comes Alive

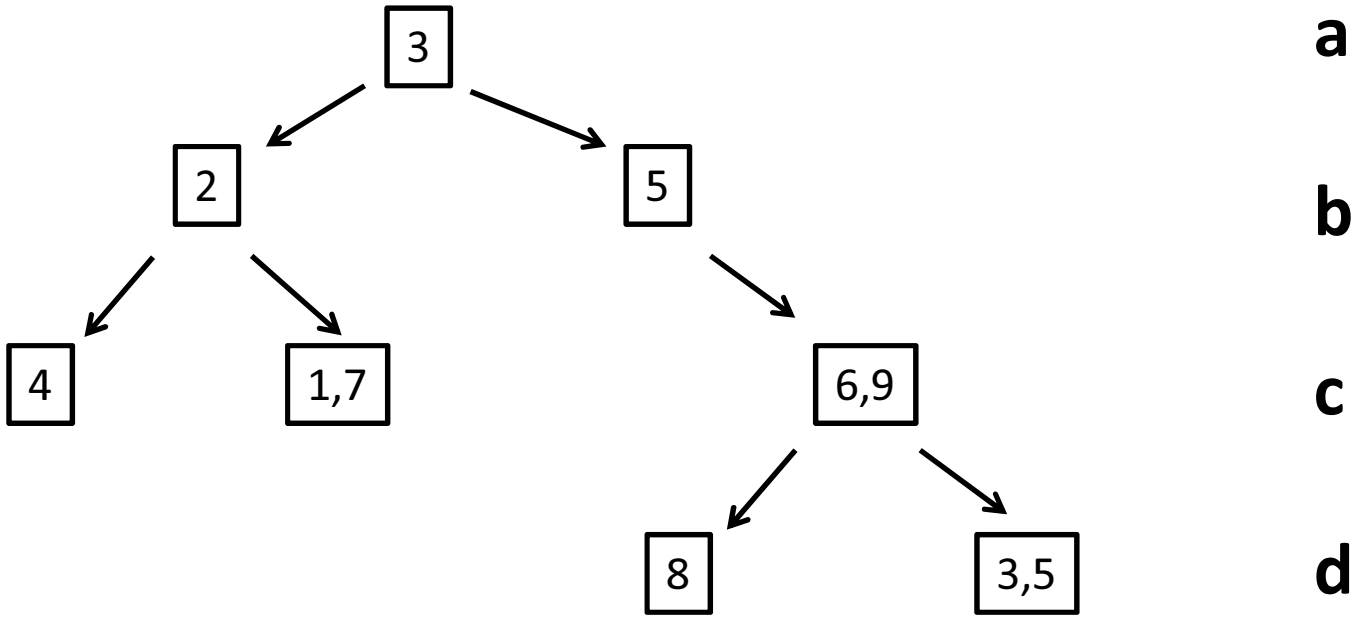
Abstract

- Hierarchies such as grids (Brand Image) or cubes (Brand/Statement/Rating) are levels where no levels are parallel , or, alternatively, all levels are mutually orthogonal at the origin.
- Such N-dimensional structures must presently be stored as either flat or as a SSS v2 <hierarchy>
- But if flat, then many columns, and if as hierarchy of surveys, then many files.
- For flat storage, the problem is acute on large brand lists with sparse code instantiation.
- 1,000 brands * 10 attributes * 10 rating points = 20,000 columns, even if most respondents skip or respond for only a few out of the 1,000 brands. And if 10 such questions, then 200,000 columns.
- For hierarchic storage, multiple files for simple grids and cubes is overkill, and conceptualising as a hierarchy of surveys can be counter-intuitive where the case is a single respondent.
- This proposal for the storage of such data as left-right trees (parsable by simply reading a string from the left) can hugely reduce the number of required columns.
- For fixed width, the number of columns is determined by the longest response in the record. For delimited storage, each respondent would require only as many characters as needed to record and structure just that respondent's answer set.
- The proposed storage could also be used to store any levels structure, but at the expense of needing to duplicate the upper paths for parallel (non-orthogonal) levels.

Red Centre Software
Data Comes Alive

Left Right Trees

Left-right trees are simply a way of representing data hierarchies as a strings which can be parsed from left to right.



Assign a depth delimiter to each level – eg a, b, c, d

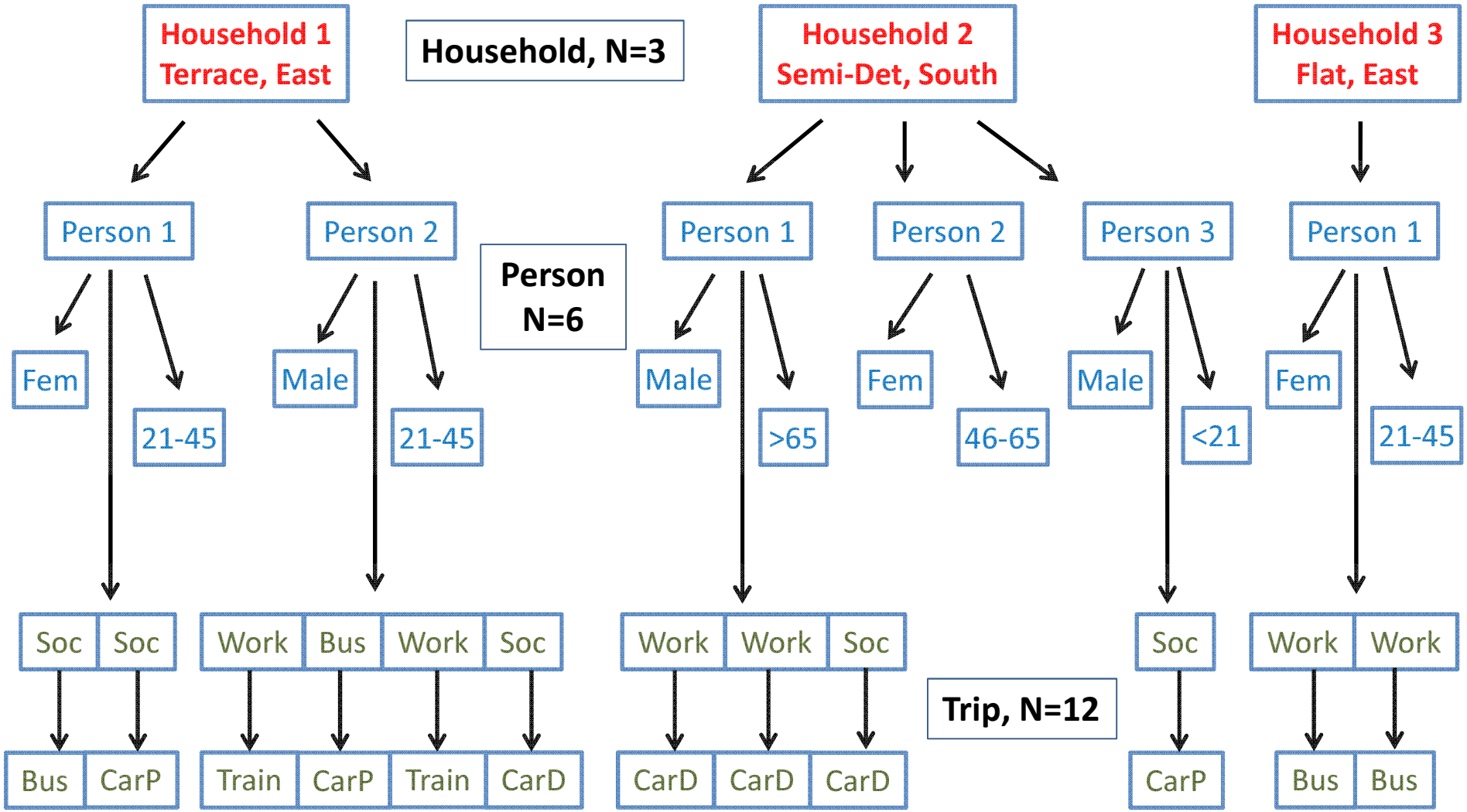
The top-down tree node structure **a b c c b c d d**

Store the data at each node as **a3b2c4c1,7b5c6,9d8d3,5**

(This is conceptually similar to Surveycraft loops)

Red Centre Software
Data Comes Alive

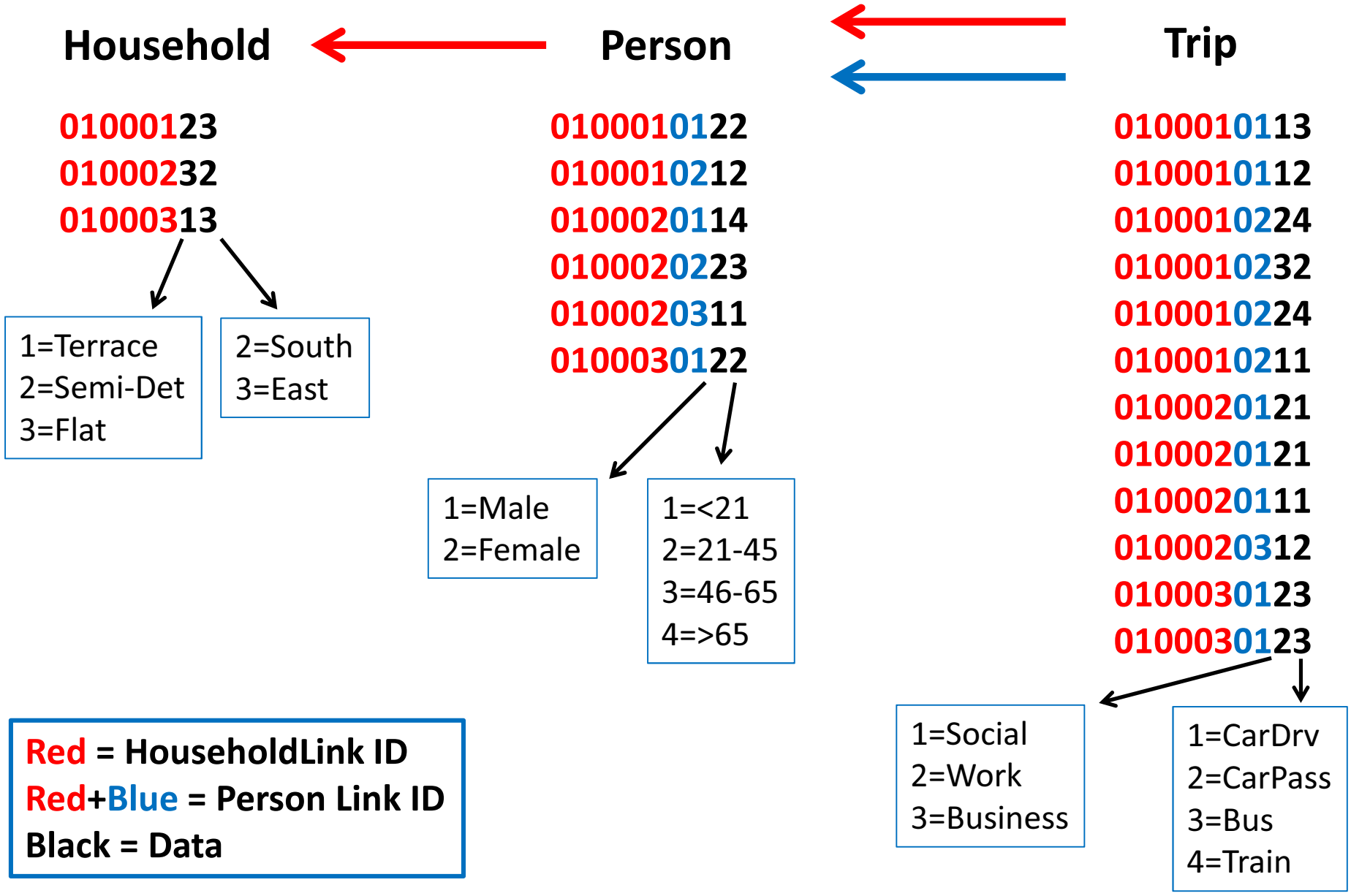
The SSS V2 Household Data



Triple-S XML version 2.0.001 (December 2006), pp 42 ff.

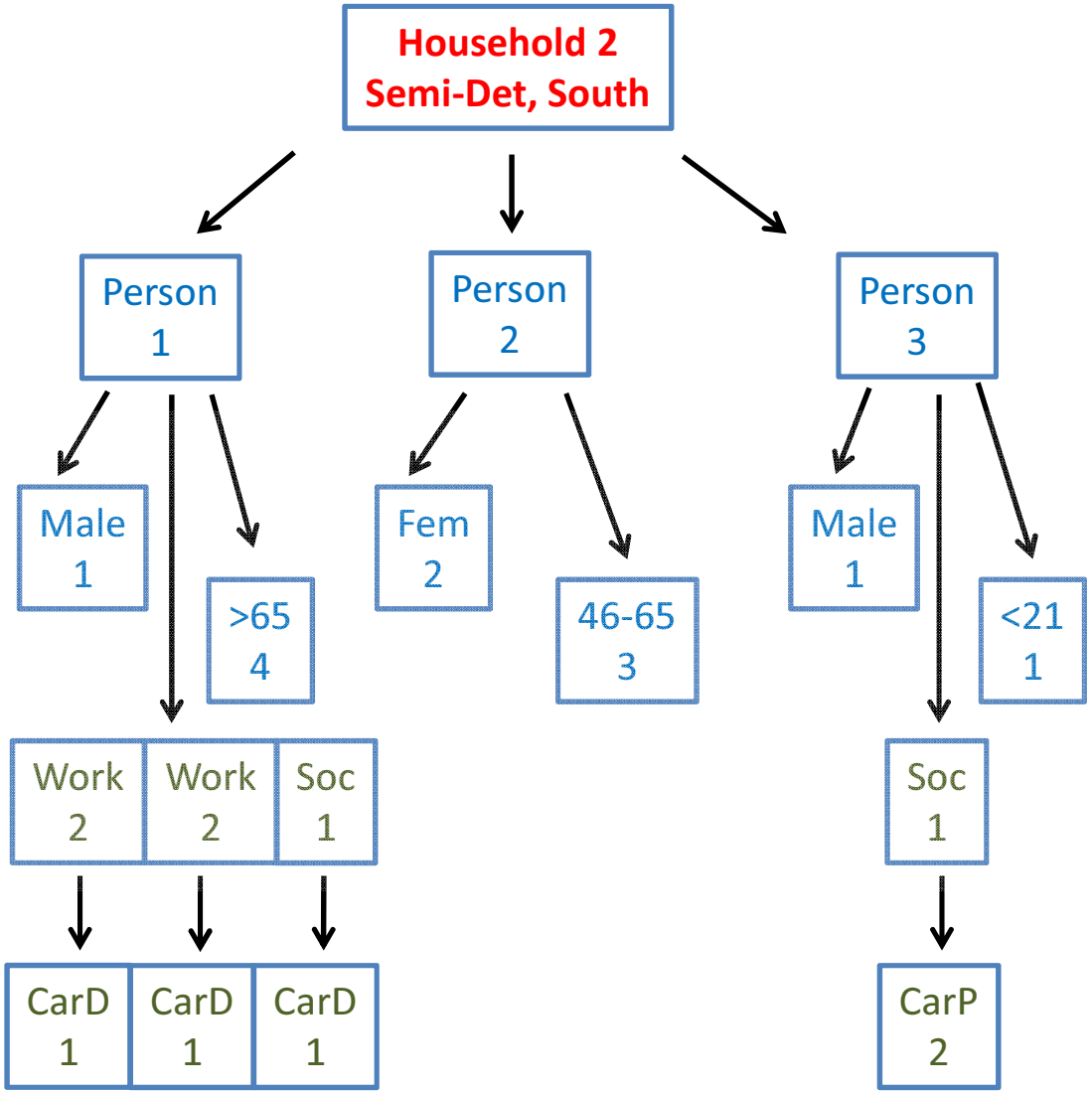
Red Centre Software
Data Comes Alive

SSS Data Storage: Hierarchy of Surveys



Red Centre Software
Data Comes Alive

Household #2 as 5 LR Trees



One tree per level requires 3 parallel b levels

a: Person: a1a2a3

b: Gender: ab1ab2ab1

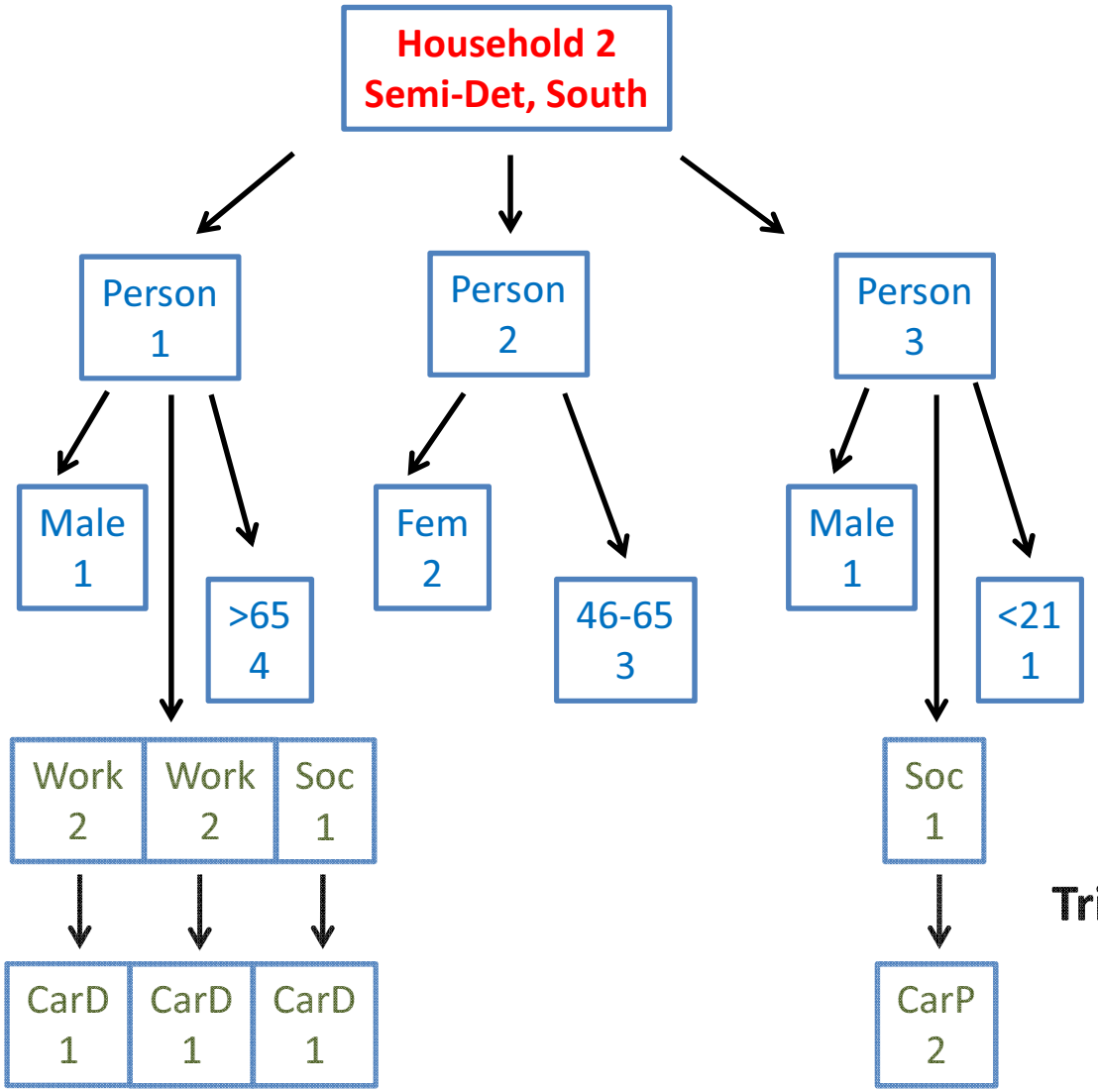
b: Age: ab4ab3ab1

b: Purpose: ab2b2b1aab1

c: Mode: abc1bc1bc1aabc2

Red Centre Software
Data Comes Alive

Household #2 as 3 LR Trees



- Store upper level data instead of just the nodes.
- 3 parallel b levels, so need at least 3 trees

Gender: a1b1a2b2a3b1

Age: a1b4a2b3a3b1

Trips: a1b2c1b2c1b1c1a2a3b1c2

Red Centre Software
Data Comes Alive

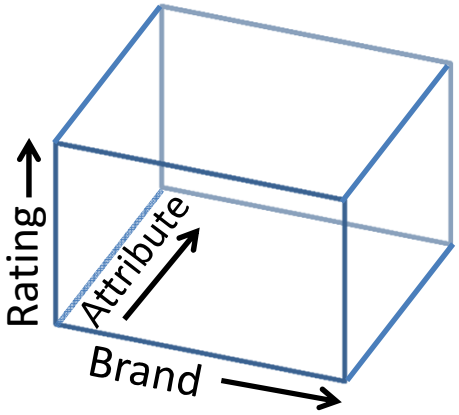
Tree vs Hierarchy of Surveys

- The three parallel levels mandate three storage instances for both – either three trees, or three survey files
- Left-right trees need to duplicate the upper paths for parallel levels
- But for circumstances where there are no parallel levels, such as Brand/Attribute/Ratings or Brand Image, left-right trees offer several advantages.
- The primary advantage is dramatically reduced storage requirements for typical brand-oriented consumer surveys

Red Centre Software
Data Comes Alive

Grids, Cubes, As LR Trees

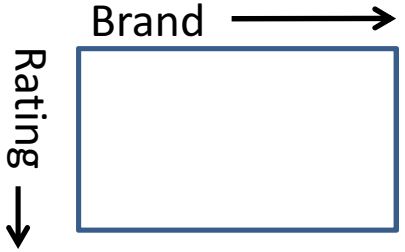
Left-right trees can also be used to store grids, cubes, or any N-dimensional data structure.



- 1 BrandX — a1
  - 1 Relevance — b1
    - 8 — c8
  - 2 Quality — b2
    - 6 — c6
  - 3 Price — b3
    - 5 — c5
- 2 BrandY — a2
  - 1 Relevance — b1
    - 6 — c6
  - 2 Quality — b2
    - 7 — c7
  - 3 Price — b3
    - 2 — c2
- 3 BrandZ — a3
  - 1 Relevance — b1
    - 7 — c7
  - 2 Quality — b2
    - 5 — c5
  - 3 Price — b3
    - 2 — c2

a1b5a2b3a3b7

- 1 BrandX — **BrandX rated 5**
  - 5
- 2 BrandY — **BrandY rated 3**
  - 3
- 3 BrandZ — **BrandZ rated 7**
  - 7



Red Centre Software
Data Comes Alive

Multi-response

Brand Image **a1b1;2;3;4;5;6;7;8a2b5;6;7;8a3b2;3;5;6**

- 1 BrandX
  - 1 Well-Known
  - 2 Trustworthy
  - 3 Easy to get
  - 4 High quality
  - 5 Good reputation
  - 6 Used by Professionals
  - 7 Modern
  - 8 Value for Money
- 2 BrandY
  - 5 Good reputation
  - 6 Used by Professionals
  - 7 Modern
  - 8 Value for Money
- 3 BrandZ
  - 2 Trustworthy
  - 3 Easy to get
  - 5 Good reputation
  - 6 Used by Professionals

- **Note the ; delimiter to avoid confusion with European , as decimal place**
- **Any level (or dimension) can be multi-response, eg a1;2b3;4c5;6;7**
- **For 10 statements coded 1 to 10, the flat storage for 3 brands (spread format) requires 60 columns**
- **Can have multi-response at any level, eg a1;2b3;4;5**

Red Centre Software
Data Comes Alive

Current Grid/Cube Storage

The implementer must choose between

- traditional flat storage, or
- SSS ver 2.0 hierarchic storage

But a typical brand tracker will have many grids, cubes, etc – a random sample of 3 jobs gives, 15, 42, and 37 instances. The cost is either

- A large number of columns (if flat), or
- A large number of files (if SSS hierarchic)

And with internet collection now dominant, the tendency to allow responses for any subset of brands for which there is awareness (rather than just the traditional main brand list) can result in combinatorial explosions which impose a heavy burden on storage, RAM and CPU. International jobs also can have very large brand lists.

Real-world examples follow:

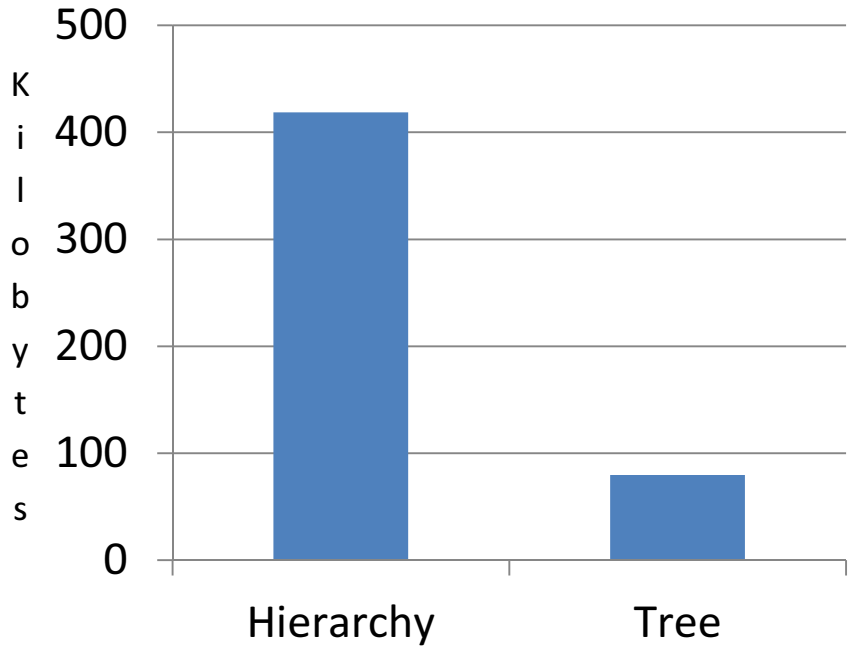
Red Centre Software
Data Comes Alive

FMCG (1): Hierarchy of Surveys

SSS fixed-width export from Confirmit, 180 respondents, 12 brands, 10 grids and 5 cubes requires 15*2 = 30 files (15 XML, 15 ASC)

Comparing storage requirements:

ASC	Bytes	Tree	Bytes
Data_0	15,747	B32	15,755
Data_1	14,728	B41	1,181
Data_2	38,523	B42a	492
Data_3	12,549	KC32	11,333
Data_4	9,218	KC41	862
Data_5	55,215	KC42a	537
Data_6	17,031	M32	14,469
Data_7	11,308	M41	975
Data_8	86,031	M42a	657
Data_9	18,321	P32	17,417
Data_10	18,528	P41	1,349
Data_11	68,055	P42a	594
Data_12	11,325	SP32	12,448
Data_13	9,978	SP41	968
Data_14	32,103	SP42a	465
total	418,660		79,502



A small number of brands, and high instantiation, but still five times less space

Red Centre Software
Data Comes Alive

FMCG (2) Flat: Brand Image

323 brands by 58 statements (multi-response) over 69,841 cases

- **Spread format:**

Requires 323*58*2 = 37,468 columns
columns * cases = 2,496 meg

- **Bit format (divide by 2):**

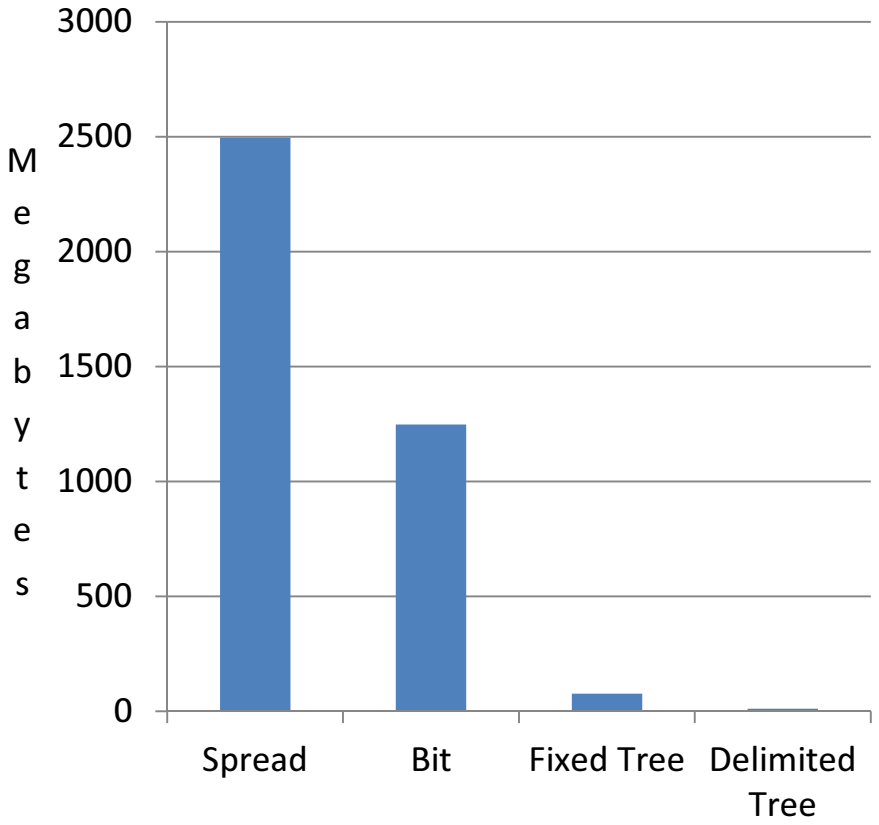
Requires 323*58 = 18,734 columns
columns * cases = 1,248 meg

- **Tree as Fixed Width:**

Longest response = 1150 characters
chars * cases = 76.6 meg

- **Tree as Delimited:**

Sum of response lengths = 11.33 meg



Red Centre Software
Data Comes Alive

FMCG (3) Fixed Width: Brand Statement Rating

204 brands by 4 statements by 5 ratings over 1,530 cases

- **Bit format:**

Requires 204*4*5 = 4,080 columns
columns * cases = 6,096 k

- **Spread format:**

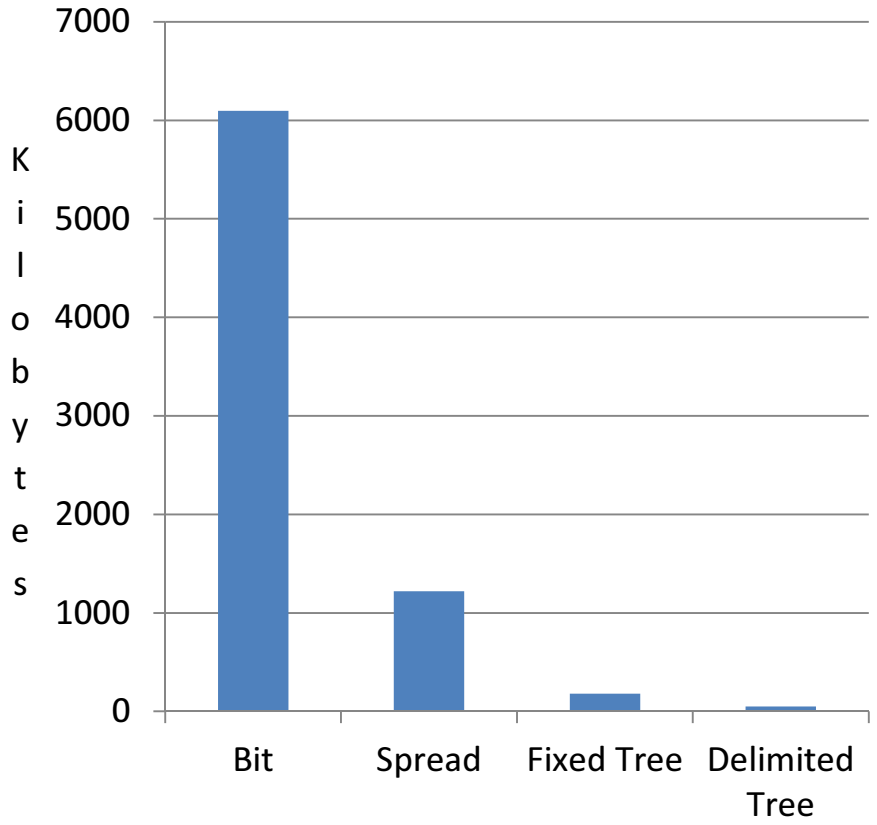
Requires 204*4 = 816 columns
columns * cases = 1,219 k

- **Tree as Fixed Width:**

Longest response = 120 characters
chars * cases = 179.3 k

- **Tree as Delimited:**

Sum of response lengths = 51.5 k



Red Centre Software
Data Comes Alive

Proposed SSS Storage: Fixed Width Single

- New tag type, tree
- Different context for the <level> tag
- No href or parent, so the levels are subordinate

Brand Rating:

```
<tree ident="BRAT">
    <position start="3" finish="10"/>
    <level ident="Brand" type="single">
        <values>
            <value code="1">AMEX</value>
            <value code="2">Visa</value>
        </values>
    </level>
    <level ident="Rating" type="single">
        <values>
            <value code="1">1</value>
            <value code="2">2</value>
            <value code="3">3</value>
        </values>
    </level>
</tree>
```

```
                      11
Column:   12345678901
Case#1:   xxa1b3a2b1x
Case#2:   xxa2b2     x
Case#3:   xx         x
Case#4:   xxa1b1a2b3x
```

Red Centre Software
Data Comes Alive

Proposed SSS Storage: Delimited Single

Brand Rating:

```
<tree ident="BRAT">
    <position start="3"/>
    <level ident="Brand" type="single">
        <values>
            <value code="1">AMEX</value>
            <value code="2">Visa</value>
        </values>
    </level>
    <level ident="Rating" type="single">
        <values>
            <value code="1">1</value>
            <value code="2">2</value>
            <value code="3">3</value>
        </values>
    </level>
</variable>
```

```
                    11111
Column:   12345678901234
Case#1:   x,x,a1b3a2b1,x
Case#2:   x,x,a2b2,x
Case#3:   x,x,,x
Case#4:   x,x,a1b1a2b3,x
```

Red Centre Software
Data Comes Alive

Proposed SSS Storage: Fixed Width Multi

```
                        1111
Column:   1234567890123
Case#1:   xxa1b1;3a2b1x
Case#2:   xxa2b1;2;3  x
Case#3:   xx          x
Case#4:   xxa1b1a2b1;2x
```

Brand Image:

```
<tree ident="BIM">
    <position start="3" finish="12"/>
    <level ident="Brand" type="single">
        <values>
            <value code="1">AMEX</value>
            <value code="2">Visa</value>
        </values>
    </level>
    <level ident="Image" type="multiple">
        <values>
            <value code="1">Cool</value>
            <value code="2">Relevant</value>
            <value code="3">Popular</value>
        </values>
    </level>
</variable>
```

Red Centre Software
Data Comes Alive

Proposed SSS Storage: Delimited Multi

Brand Image:

```
<tree ident="BIM">
   <position start="3"/>
   <level ident="Brand" type="single">
      <values>
         <value code="1">AMEX</value>
         <value code="2">Visa</value>
      </values>
   </level>
   <level ident="Image" type="multiple">
      <values>
         <value code="1">Cool</value>
         <value code="2">Relevant</value>
         <value code="3">Popular</value>
      </values>
   </level>
</tree>
```

```
                    1111111
Column:  123456789012345 6
Case#1:  x,x,a1b1;3a2b1,x
Case#2:  x,x,a2b1;2;3,x
Case#3:  x,x,,x
Case#4:  x,x,a1b1a2b1;2,x
```

Red Centre Software
Data Comes Alive

Pros and Cons

Pro:

- 2 files only always (one XML, one ASC)
- No need for Link IDs
- No need for <parent> and <href> tags
- No need for a Definition XML
- The number of cases (across all data) remains constant
- The base counts at each level are simply the number of a-nodes, b-nodes, c-nodes etc
- Data is implicitly ordered, so do not need *order* attribute
- Dramatic reduction in space requirements for grids/cubes with large codeframes when only a subset have responses, especially under CSV
- The current <Hierarchy> tags are unaffected
- A levels job can store grids and cubes asked at different levels as trees, avoiding levels within levels conundrums

Con:

- Could cost more characters than fixed-width for node-complete (all codes at all levels are instantiated)
- Position is recorded only for the start/end of the tree

Red Centre Software
Data Comes Alive

Household Data Storage

XML could look like this:

```
<trees ident="HHTrips">
   <level ident="Person" type="single">
      <position start="1"/>
      <values>
         <range from="1" to="10"/>
      </values>
  </level>
  <level ident="Gender" type="single" parent="Person">
      <position start="2"/>
      <values>
         <value code="1">Male</value>
         <value code="2">Female</value>
      </values>
  </level>
  <level ident="Age" type="single" parent="Person">
      <position start="3"/>
      <values>
         <value code="1">Under 21</value>
         <value code="2">21-45</value>
         <value code="3">46-65</value>
         <value code="4">Over 65</value>
      </values>
  </level>
  <level ident="Purpose" type="single" parent="Person">
      <position start="4"/>
      <values>
         <value code="1">Social</value>
         <value code="2">Work</value>
         <value code="3">Business</value>
       </values>
  </level>
  <level ident="Method" type="single" parent="Purpose">
      <position start="5"/>
      <values>
         <value code="1">Car Driver</value>
         <value code="2">Car Passenger</value>
         <value code="3">Bus</value>
         <value code="4">Train</value>
       </values>
  </level>
</trees>
```

HH#1: a1a2,ab2ab1,ab2ab2,ab1b1,abc3bc2
HH#2: a1a2a3, ab1ab2ab1, ab4ab3ab1, ab2b2b1aab1, abc1bc1bc1aabc2
HH#3: a1,ab2,ab2,ab2b2,abc3bc3

- **Trees tag because a set of trees is described**
- **The top level *Person* has no parent**
- **The parent attribute allows parallelism**
- **If no parents assigned then same as <tree>**

End