

## Chapter 2: Syntax for Data

Semistructured Data: self-describing or schemaless.

```
{name: "Alan", tel: 2157786, email: "agb@abc.com"}
```

set of label-value pairs;

The values themselves may be structures; ex:

```
{name: {first: "Alan", last: "Black"},  
  tel: 2157786,  
  email: "agb@abc.com"  
}
```

Graphical notation: nodes representing objects connected by edges  
(labeled with the label) to the values.

.

name tel email

"Alan" 2157786 "agb@abc.com"

.

name tel email

. 2157786 "agb@abc.com"

first last

"Alan" "Black"

Duplicate labels will be allowed:

```
{name: "Alan", tel: 2157786, tel: 2498762}
```

The syntax is easily generalized to describe sets of objects/tuples:

```
{ person:  
  {name: "Alan", tel: 2157786, email: "agb@abc.com"},  
  person:  
  {name: "Sara", tel: 2136877, email: "sara@math.xyz.com"},  
  person:  
  {name: "Fred", tel: 7786312, email: "fds@acme.co.uk"}  
}
```

All objects/tuples within a set need not have the same structure:

```
{ person:
  {name: "Alan", tel: 2157786, email: "agb@abc.com"},
  person:
  {name: {first: "Sara", last: "Green"},
   tel: 2136877,
   email: "sara@math.xyz.com"}
  },
  person:
  {name: "Fred", tel: 7786312, height: 183}
}
```

Base Types: numbers, strings, labels  
there could be others, but in this book we  
limit ourselves to these three

## Representing Relational Databases:

```
{r1: {row: {a: a1, b: b1, c: c1},  
      row: {a: a2, b: b2, c: c2}  
},  
r2: {row: {c: c2, d: d2},  
     row: {c: c3, d: d3},  
     row: {c: c4, d: d4}  
}  
}
```

This data represents two relations r1 and r2.

See Figure 2.3 on Page 15 for 3 tree structures representing the same data;

## Representing Object Databases:

```
{person: &o1{name: "Mary",
             age: 45,
             child: &o2,
             child: &o3
          },
 person: &o2{name: "John",
             age: 17,
             relatives: {mother: &o1,
                        sister: &o3}
          },
 person: &o3{name: "Jane",
             country: "Canada",
             mother: &o1
          }
}
```

See Figure 2.4 for graphical notation. The names  $&o1$ ,  $&o2$ ,  $&o3$  are called Object Identities (oids).

Each node has a unique oid. If a node is not assigned an oid by the user, the system automatically creates one for the node.

ex:             $\{a: &o1\{b: &o2\ 5\}\}$   
          and  $\{a: &o1\{b: 5\}\}$   
are isomorphic

In  $\{a: \{b: 3\}, a: \{b: 3\}\}$   
there are two objects with different oids!

Syntax for semi-structured data expressions (ssd):

```
<ssd-expr> ::= <value> | oid <value> | oid
```

```
<value> ::= atomicvalue | <complexvalue>
```

```
<complexvalue> ::= { label:<ssd-expr>, ..., label:<ssd-expr> }
```

oids are written with an ampersand followed by a string.



Def: An oid  $o$  is DEFINED in an ssd-expression  $s$  if  
either  $s$  is of the form  $o v$  for some atomic value  $v$   
or  $s$  is of the form  $\{l_1:e_1, \dots, l_n:e_n\}$  and  $o$  is defined  
in one of  $e_1, \dots, e_n$ .

If an oid occurs in  $s$  in any other way,  
we say that it is USED in  $s$ .

Def: ssd-expression  $s$  is CONSISTENT if

- (a) Any object identifier is defined at most once in  $s$  and
- (b) If an object identifier  $o$  is used in  $s$ , it must be defined  
in  $s$