The values themselves may be structures:

set of label-value pairs:

```
{name: "Alan", tel: 2157786, email: "agBabC.com"}
```

Semistructured data: self-describing or schemaless.

Chapter 2: Syntax for Data
"Alan" "Black"

First Last

2157786 "abc@abc.com"

name tel email

"Alan" 2157786 "abc@abc.com"

name tel email

(Tabled with the table) to the values.

Graphical notation: nodes representing objects connected by edges
The syntax is easily generalized to describe sets of objects/tuples:

```
{ name: "Fred", tel: 7786312, email: "f@acme.co.uk" },
{ name: "Sara", tel: 2136877, email: "s@math.xyz.com" },
{ name: "Alan", tel: 215786, email: "a@abcc.com" },

Duplicate tables will be allowed:
```
Limit ourselves to these three
there could be others, but in this book we
Base types: numbers, strings, labels

```json
{
  {name: "Fred", tel: 7786312, height: 183}
  {person: {
    {email: "saramath.xyz.com"
    tel: 2136877,
    {name: "Sara", last: "Green"}
    {person: {
      {name: "Anna", tel: 215786, email: "alphabet.com"
      }}
    }}
  }
}
```

All objects/tuples within a set need not have the same structure.
This data represents two relations $R_1$ and $R_2$.

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

Representing Relational Databases:
```
{  
  {  
    mother: "Joel"
    country: "Canada"
    person: {name: "Jane"}
  },  
  {  
    sister: "Sarah"
    relatives: {  
      mother: "Joel"
      age: 17
      person: {name: "John"}
    }
    child: [  
      {  
        child: "Sara"
        age: 45
        person: {name: "Mary"}
      }
    ]
  }
}

Representing Object Databases:
there are two objects with different ids!

In

are isomorphic

and

ex:

user, the system automatically creates one for the node.

Each node has a unique oid. If a node is not assigned an oid by the

called Object Identifiers (oids).

See Figure 2.4 for graphical notation. The names $x_01$, $x_02$, $x_03$ are
The syntax for semi-structured data expressions (ssd):

```plaintext
{ <expr>: ssd-expr } ::= ::
  | \texttt{atom}\texttt{value}
  | \texttt{value} \texttt{old} \texttt{value} \texttt{old} \texttt{value}
```
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 if of the form {l1:e1, ..., ln:en} and o is defined
       in one of e1, ..., en.
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