Semistructured Data
Semistructured Data

- *Semistructured data* is data that has some structure, but it may be irregular and incomplete and does not necessarily conform to a fixed schema

  - World-Wide Web
  - Integration of data from heterogeneous sources.
Object Exchange Model (OEM)

- Nodes without outgoing edges are called atomic objects; the rest of the nodes are called complex objects.
- Atomic objects have a value of type integer, real, string, etc. Complex objects have the reserved value C.
An OEM database is a 4-tuple $O=(N,A,v,r)$, where:

- $N$ is a set of object identifiers;
- $A$ is a set of labeled, directed arcs $(p,l,c)$ where $p,c \in N$ and $l$ is a string;
- $v$ is a function that maps each node $n \in N$ to an atomic value or the reserved value $C$; and
- $r$ is a distinguished node in $N$ called the root of the database.
Lorel Example 1

Select  Guide.restaurant
where  guide.restaurant.restaurant.price < 20.5

• The result of the query is a singleton set containing the restaurant object for “Bangkok Cuisine.”
• Lorel coerces the values to a common same type before making the comparisons.
Lorel Example 2

Select Guide.restaurant
from Guide.restaurant.address.street
where Z = "Green"

select Guide.restaurant
from Guide.restaurant
where Guide.restaurant.address.street = "Green"
Basic Change Operations

- \textit{creNode}(n,v): creates a new object. The identifier \( n \) must be new.

- \textit{updNode}(n,v): changes the value of object \( n \), where \( v \) is an atomic value or the special symbol \( C \). Object \( n \) must be either an atomic object or a complex object without subobjects.

- \textit{addArc}(p,l,c): adds an arc labeled \( l \) from object \( p \) to object \( c \). The new arc must not already exist.

- \textit{remArc}(p,l,c): removes an arc \( (p,l,c) \).
Valid Change Sequence

• We say that a sequence $L = u_1, u_2, \ldots, u_n$ of basic change operations is valid for an OEM database $O$ if $u_i$ is valid for $O_{i-1}$ for all $i = 1, \ldots, n$, where $O_0 = O$, and $O_i = u_i(O_{i-1})$, for $i = 1, \ldots, n$.

• We use $L(O)$ to denote the OEM database obtained by applying the entire $L$ to $O$. 
Valid Changes

We say that a set $U = \{u_1, u_2, \ldots, u_n\}$ of basic change operations is valid for an OEM database $O$ if

- for some ordering $L$ of the changes in $U$, $L$ is a valid sequence of changes,
- for any two such valid sequences $L$ and $L'$, $L(O) = L'(O)$, and
- $U$ does not contain both $\text{addArc}(p,l,c)$ and $\text{remArc}(p,l,c)$ for any $p$, $l$, and $c$. 
OEM History

• **OEM history** is a sequence $H = (t_1, U_1), \ldots, (t_n, U_n)$, where $U_i$ is a set of basic change operations and $t_i$ is a timestamp, for $i = 1, \ldots, n$, and $t_i < t_{i+1}$ for $i = 1, \ldots, n-1$.

• We say $H$ is **valid** for an OEM database $O$ if, for all $i = 1, \ldots, n$, $U_i$ is valid for $O_{i-1}$, where $O_o = O$, and $O_i = U_i(O_{i-1})$ for $i = 1, \ldots, n$. 
OEM History Example

- We have the history $H = ((t_1, U_1), (t_2, U_2), (t_3, U_3))$, where $t_1 = 1Jan97$, $t_2 = 5Jan97$, $t_3 = 8Jan97$.

  - $U_1 = \{ \text{updNode}(n_1, 20), \text{creNode}(n_2, C), \text{creNode}(n_3, "Hakata"), \text{addArc}(n_4, "restaurant", n_2), \text{addArc}(n_2, "name", n_3) \}$

  - $U_2 = \{ \text{creNode}(n_5, "need info") \text{addArc}(n_2, "comment", n_5) \}$

  - $U_3 = \{ \text{remArc}(n_6, "parking", n_7) \}$
Annotations to the OEM graph

- Annotations are attached to the nodes and arcs to encode the history of basic change operations.

- Four types of annotations:
  - \textit{cret}(t): the node was created at time \( t \).
  - \textit{upd}(t,ov): the node was updated at time \( t \), \( ov \) is the old value.
  - \textit{add}(t): the arc was added at time \( t \).
  - \textit{rem}(t): the arc was removed at time \( t \).
OEM graph with Annotations
DOEM Database

- The set of all possible node annotations is denoted by \textit{node-annot}, and the set of all possible arc annotations is denoted by \textit{arc-annot}.

- A \textit{DOEM database} is a triple $D = (O,f_N,f_A)$, where $O = (N, A, v, r)$ is an OEM database, $f_N$ maps each node in $N$ to a finite subset of \textit{node-annot}, and $f_A$ maps each arc in $A$ to a finite subset of \textit{arc-annot}.
DOEM Database - Properties

• Given a DOEM database $D$, it is easy to obtain
  - the original snapshot, $O_o(D)$,
  - the snapshot at time $t$, $O_t(D)$, and
  - the current snapshot, $O_c(D)$. 
**Chorel Example 1**

**QUERY:** Find the names of all restaurants whose price ratings were updated on or after January 1st, 1997 to a value greater than 15, together with the time of the update and the new price.

Select $N, T, NV$
from $guide.restaurant.price <upd at T to NV>,$
$guide.restaurant.name N$
where $T >= 1Jan97$ and $NV > 15$

**Answer:**
name "Bangkok Cuisine"
new-value 20
update-time 1Jan97
**Chorel Example 2**

**QUERY:** Find the names of restaurants to which a “moderate” price subobject was added since January 1st, 1997.

```
Select N
from guide.restaurant R, R.name N
where R.<add at T> price = "moderate" and T>= 1Jan97
```
Syntax of Annotation Expression

- `<Annot [at timeV]>` if `Annot` is in `{ add, rem, cre }
  `<upd [at timeV] [from oldV] [to newV]>` for `upd`

- Arc annotation expressions must occur immediately before a label.
  Example: `guide.restaurant. <add at T> price`

- Node annotation expressions must occur immediately after a label.
  Example: `guide.restaurant.price <upd at T to NV>`