

CSC343 Worksheet 9 Solution

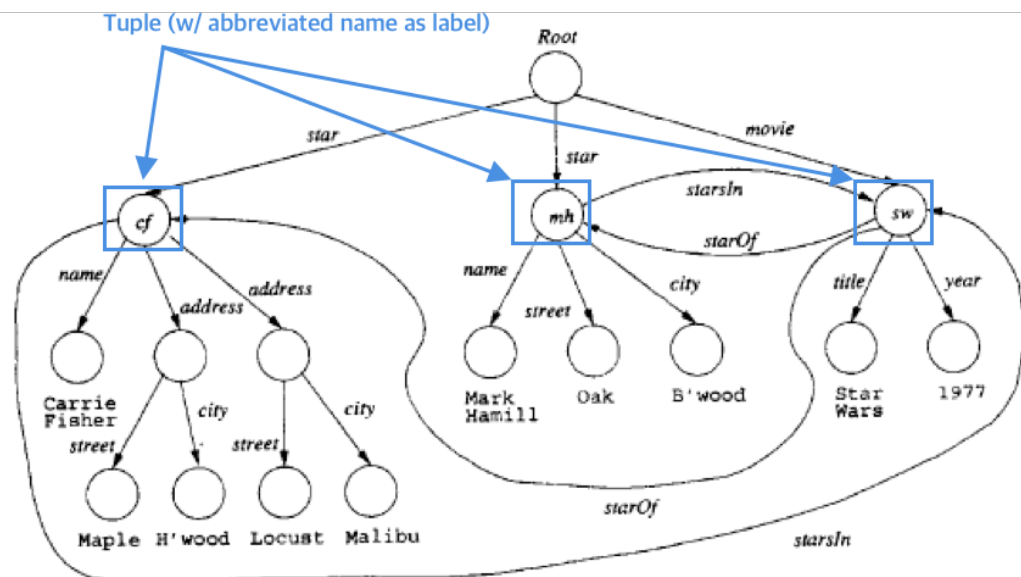
June 27, 2020

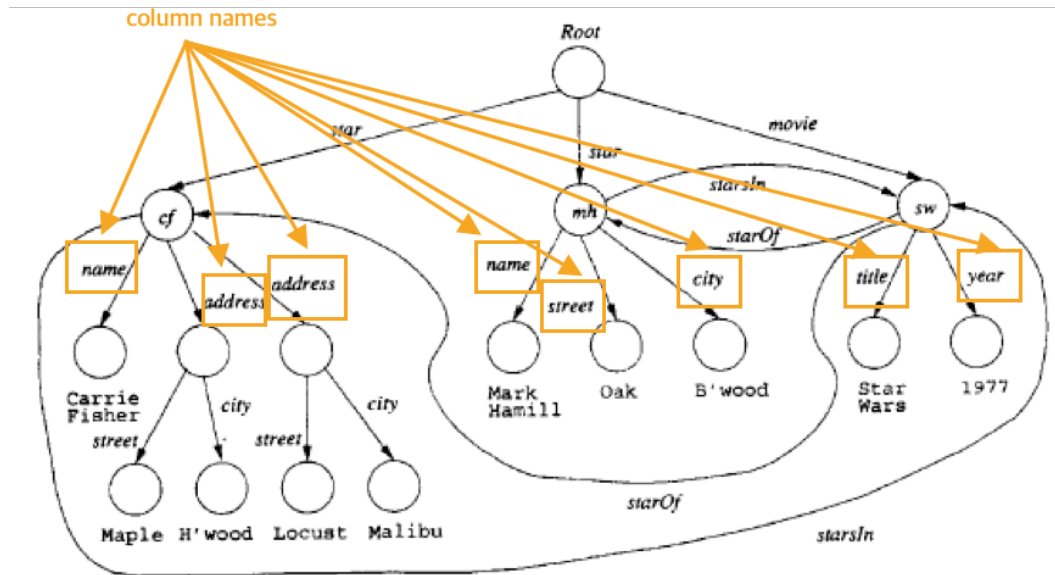
1. Exercise 11.1.1:



Notes:

- Semistructured data
 - serves as a model suitable for **databases integration**, that is, for describing the data contained in two or more databases that contain similar data with different schemas
 - It serves as the underlying model for notations such as XML, to be taken up in Section 2, that are being used to share information on the web.
- Semistructured Data Representation
 - is a collection of nodes





b)



c)





- 3.
4. The difference is that UML must fit data into its schema, where as the semi structured data allows whatever schema information that is appropriate to be attached to data

Notes:

- Semi-structured Data
 - Is schemaless

- Is motivated primarily by its flexibility
- One could enter data at will, and attach to the data whatever schema information you felt was appropriate for that data.
- Makes query processing harder
- Structured Data
 - Is rigid framework into which data is placed.
 - Data must fit into schema
 - Fixed schema allows data to be organized with data structures that support efficient answering of queries
 - e.g. UML, E/R, Relational, ODL

5. a)

```

1  <? xml version = "1.0" encoding="utf-8" standalone = "yes">
2  <StarMovieData>
3    <Star starID="cf" starredIn="sw">
4      <Name>Carrie Fisher</Name>
5      <Address>
6        <Street>123 Maple St.</Street>
7        <City>Hollywood</City>
8      </Address>
9      <Address>
10       <Street>5 Locust Ln.</Street>
11       <City>Malibu</City>
12     </Address>
13   </Star>
14   <Star starID="mh" starredIn="sw">
15     <Name>Mark Hamill</Name>
16     <Street>456 Oak Rd.</Street>
17     <City>Brentwood</City>
18   </Star>
19   <Movie movieID="sw" starsOf="cf", "mh">
20     <Title>Star Wars</Title>
21     <Year>1977</Year>
22   </Movie>
23   <MovieExec movieExecID="gl" execsIn="sw">
24     <Name>George Lucas</Name>
25     <Position>Director</Position>
26   </MovieExec>
27   <MovieExec movieExecID="gk" execsIn="sw">
28     <Name>Gru Kurtz</Name>
29     <Position>Produced By</Position>
30   </MovieExec>
31 </StarMovieData>
32

```

- XML
 - is called *Extensible Markup Language*
 - is an example of semistructured data
- XML with and without a Schema

- has two different types
 1. Well-formed XML
 - * allows to invent your own tags
 - * corresponds very-similarly to semi-structured data

Example:

```

1  <? xml version = "1.0" encoding="utf-8" standalone = "yes
   ">
2  <StarMovieData>
3    <Star>
4      <Name>Carrie Fisher</Name>
5      <Address>
6        <Street>123 Maple St.</Street>
7        <City>Hollywood</City>
8      </Address>
9      <Address>
10       <Street>5 Locust Ln.</Street>
11       <City>Malibu</City>
12     </Address>
13   </Star>
14   <Star>
15     <Name>Mark Hamill</Name>
16     <Street>456 Oak Rd.</Street>
17     <City>Brentwood</City>
18   </Star>
19   <Movie>
20     <Title>Star Wars</Title>
21     <Year>1977</Year>
22   </Movie>
23 </StarMovieData>
24

```

2. Valid XML
 - * Involves "Document Type Definition"
 - * specifies allowable tags and gives a grammar for how they may be nested

- Attributes

- is used to represent connections in a semistructured data graph

Example:

```

1  <? xml version = "1.0" encoding="utf-8" standalone = "yes">
2  <StarMovieData>
3    <Star starID="cf" starredIn="sw">
4      <Name>Carrie Fisher</Name>
5      <Address>
6        <Street>123 Maple St.</Street>
7        <City>Hollywood</City>

```



```

8         </Address>
9         <Address>
10             <Street>5 Locust Ln.</Street>
11             <City>Malibu</City>
12         </Address>
13     </Star>
14     <Star starID="mh" starredIn="sw">
15         <Name>Mark Hamill</Name>
16         <Street>456 Oak Rd.</Street>
17         <City>Brentwood</City>
18     </Star>
19     <Movie starID="sw" starOf="cf", "mh">
20         <Title>Star Wars</Title>
21         <Year>1977</Year>
22     </Movie>
23 </StarMovieData>
24

```

- Namespaces

- **Syntax:** xmlns:name:URI
- Is similar to import numpy as np in python
- Is used to distinguish tags coming from different sources, i.e. HTML

Example:

Retrieving element *StarMovieData* from document infolab.stanford.edu/movies.
Set md as the name of import

```

1     <md:StarMovieData xmlns:md="http://infolab.stanford.edu/
2     movies">

```

b)

```

2 <? xml version = "1.0" encoding="utf-8" standalone = "yes">
3 <StarMovieData>
4     <Star starID="cf" starredIn="sw">
5         <Name>Carrie Fisher</Name>
6         <Address>
7             <Street>123 Maple St.</Street>
8             <City>Hollywood</City>
9         </Address>
10        <Address>
11            <Street>5 Locust Ln.</Street>
12            <City>Malibu</City>
13        </Address>
14    </Star>
15    <Star starID="mh" starredIn="sw">
16        <Name>Mark Hamill</Name>
17        <Street>456 Oak Rd.</Street>
18        <City>Brentwood</City>
19    </Star>
20    <Movie movieID="sw" starsOf="cf", "mh">
21        <Title>Star Wars</Title>
22        <Year>1977</Year>

```

```
22     </Movie>
23     <Movie movieID="esb" starOf="cf", "mh">
24         <Title>Empire Strikes Back</Title>
25         <Year>1980</Year>
26     </Movie>
27     <Movie movieID="roj" starOf="cf", "mh">
28         <Title>Return of Jedi</Title>
29         <Year>1983</Year>
30     </Movie>
31 </StarMovieData>
32
```