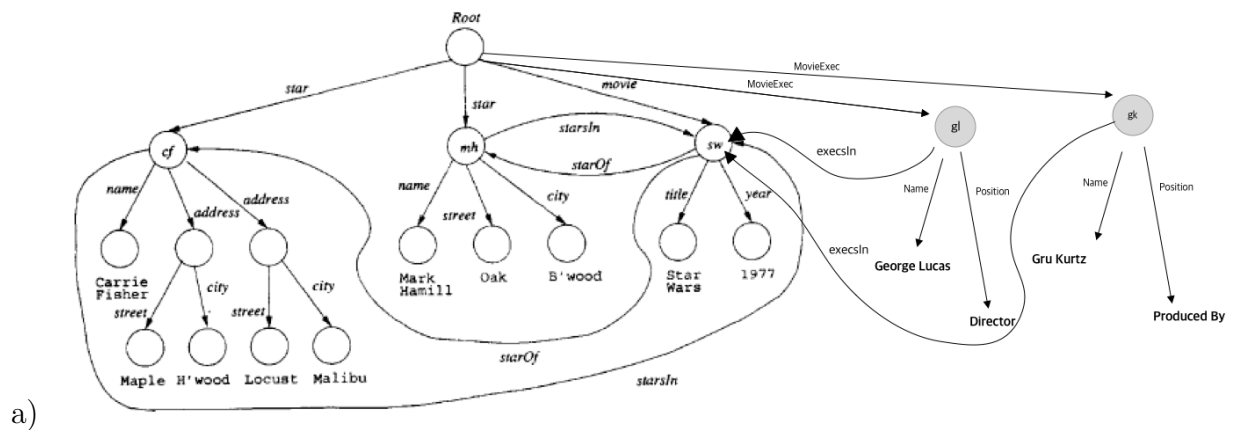


CSC343 Worksheet 9 Solution

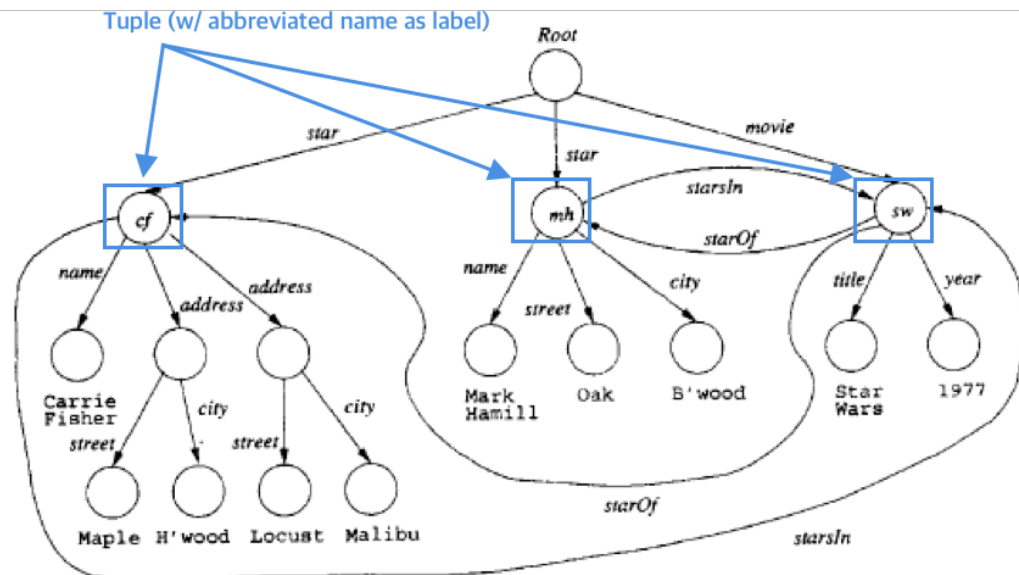
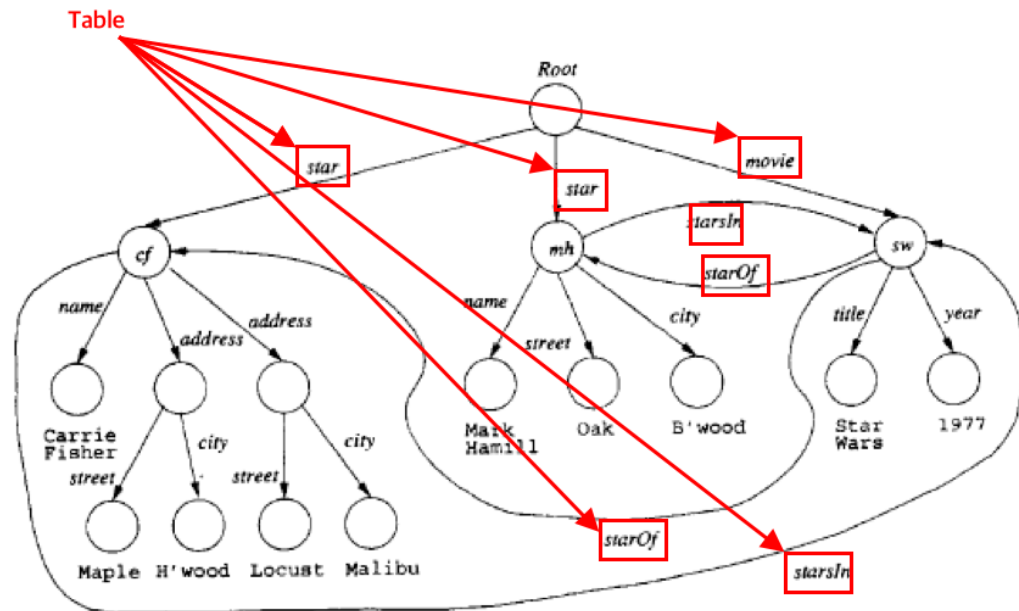
June 28, 2020

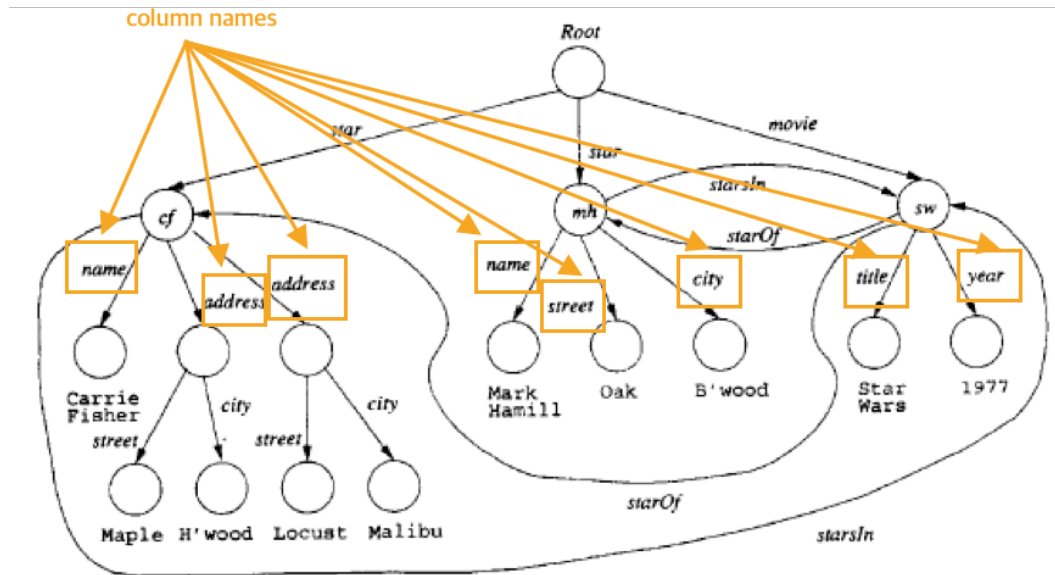
1. Exercise 11.1.1:



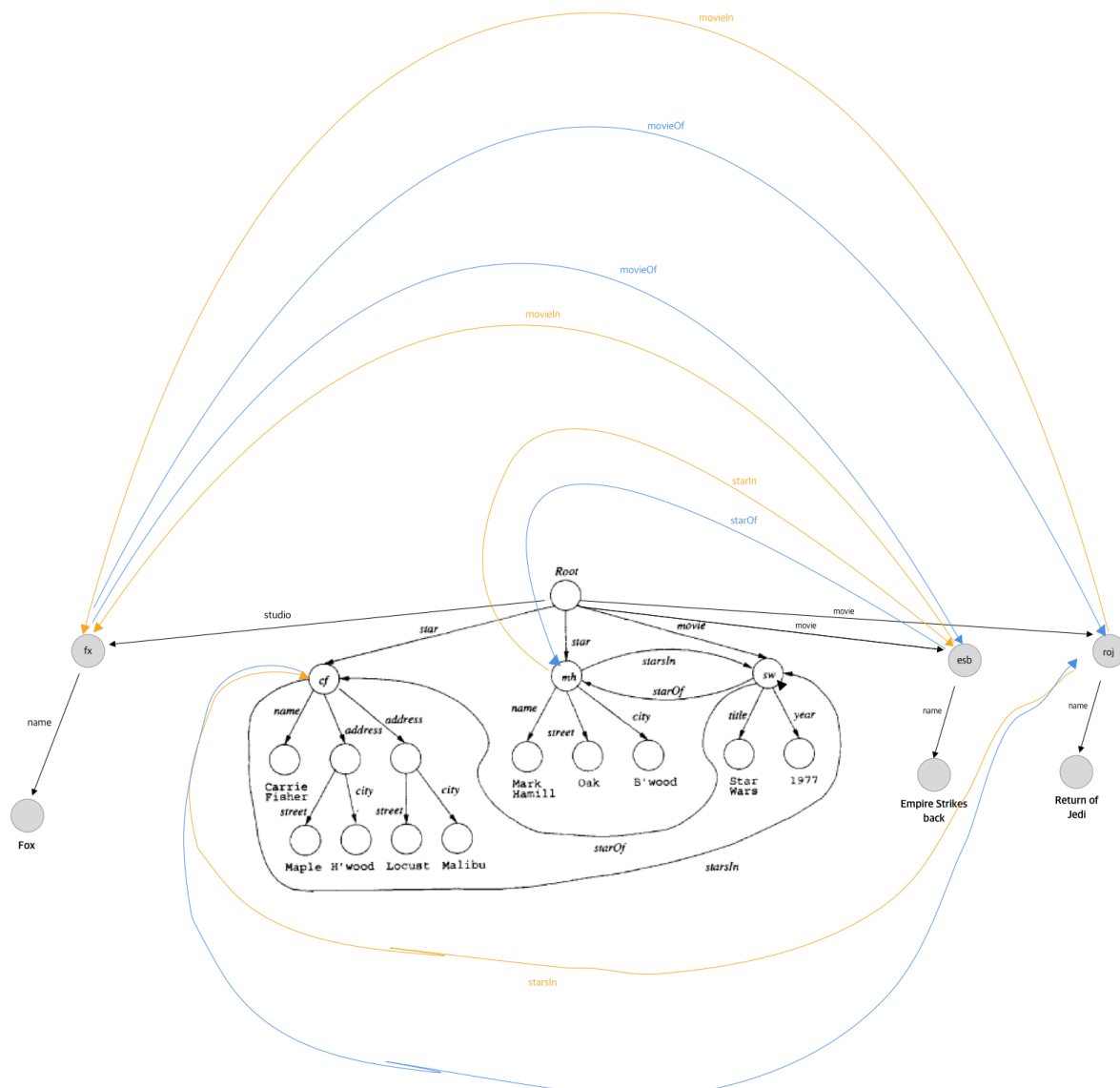
Notes:

- 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

```

c)

```

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" movieIn="fx">
21         <Title>Star Wars</Title>
22         <Year>1977</Year>
23     </Movie>
24     <Movie movieID="esb" starOf="cf", "mh" movieIn="fx">
25         <Title>Empire Strikes Back</Title>
26         <Year>1980</Year>
27     </Movie>
28     <Movie movieID="roj" starOf="cf", "mh" movieIn="fx">
29         <Title>Return of Jedi</Title>
30         <Year>1983</Year>
31     </Movie>
32     <Studio studioID="fx" movieOf="esb", "roj", "sw">
33         <Name>Fox</Name>
34         <Address>Hollywood</Address>
35     </Studio>
36 </StarMovieData>

```

6. Consider the following relation Classes:

<i>class</i>	<i>type</i>	<i>country</i>	<i>numGuns</i>	<i>bore</i>	<i>displacement</i>
Bismarck	bb	Germany	8	15	42000
Iowa	bb	USA	9	16	46000
Kongo	bc	Japan	8	14	32000
North Carolina	bb	USA	9	16	37000
Renown	bc	Gt. Britain	6	15	32000
Revenge	bb	Gt. Britain	8	15	29000
Tennessee	bb	USA	12	14	32000
Yamato	bb	Japan	9	18	65000

(a) Sample data for relation Classes

```

1  <? xml version = "1.0" encoding="utf-8" standalone = "yes">
2  <shipsData>
3    <Classes>
4      <Class>Bismarck</Class>
5      <Type>bb</Type>
6      <Country>Germany</Country>
7      <NumGuns>8</NumGuns>
8      <Bore>15</Bore>
9      <Displacement>42000</Displacement>
10   </Classes>
11   <Classes>
12     <Class>Iowa</Class>
13     <Type>bb</Type>
14     <Country>USA</Country>
15     <NumGuns>9</NumGuns>
16     <Bore>16</Bore>
17     <Displacement>46000</Displacement>
18   </Classes>
19   <Classes>
20     <Class>Kongo</Class>
21     <Type>bc</Type>
22     <Country>Japan</Country>
23     <NumGuns>8</NumGuns>
24     <Bore>14</Bore>
25     <Displacement>32000</Displacement>
26   </Classes>
27   <Classes>
28     <Class>North Carolina</Class>
29     <Type>bb</Type>
30     <Country>USA</Country>
31     <NumGuns>9</NumGuns>
32     <Bore>16</Bore>
33     <Displacement>37000</Displacement>
34   </Classes>
35   <Classes>
36     <Class>Renown</Class>

```

```

37         <Type>bc</Type>
38         <Country>Gt. Britain</Country>
39         <NumGuns>6</NumGuns>
40         <Bore>15</Bore>
41         <Displacement>32000</Displacement>
42     </Classes>
43     <Classes>
44         <Class>Revenge</Class>
45         <Type>bb</Type>
46         <Country>Gt. Britain</Country>
47         <NumGuns>8</NumGuns>
48         <Bore>15</Bore>
49         <Displacement>29000</Displacement>
50     </Classes>
51     <Classes>
52         <Class>Tennessee</Class>
53         <Type>bb</Type>
54         <Country>USA</Country>
55         <NumGuns>12</NumGuns>
56         <Bore>14</Bore>
57         <Displacement>32000</Displacement>
58     </Classes>
59     <Classes>
60         <Class>Yamato</Class>
61         <Type>bb</Type>
62         <Country>Japan</Country>
63         <NumGuns>9</NumGuns>
64         <Bore>18</Bore>
65         <Displacement>65000</Displacement>
66     </Classes>
67 </shipsData>
68

```

7. • DocRoot

```

1     <DocRoot docID="" rootElementID="" />
2

```

• SubElement

```

1     <Subelement parentID="" childID="" position="" />
2

```

• ElementAttribute

```

1     <ElementAttribute elementID="" name="" value="" />
2

```

• ElementValue

```

1     <ElementValue elementID="" name="" value="" />
2

```

Notes:

- Empty element
 - is an element that is complete by itself
 - rather than being composed of a start tag, data and an end tag, the empty element is a combined start and end tag.

Example:

```

1  <phone>
2    <entry>
3      <name>Chip</name>
4      <extension number="3"/>
5    </entry>
6  </phone>
7

```

- * Phone is not an empty element
- * extension is an empty element

8. The following will be used as an example.



```

1  <? xml version = "1.0" encoding="utf-8" standalone = "yes">
2  <DocRoot docID="root" rootElementID="">
3    <SubElement parentID="root" childID="cf" position="1">
4      <ElementAttribute elementID="cf-starsIn" name="starsIn"
value="sw"/>

```

```

5      <ElementValue elementID="cf-name" name="name" value="Carrie
    Fisher"/>
6      <SubElement parentID="cf" childID="cf-add_1" position="1">
7          <City elementID="cf-add_1-city" value="Hollywood">
8              <Street elementID="cf-add_1-street" value="Maple">
9                  </SubElement>
10             <SubElement parentID="cf" childID="cf-add_2" position="2">
11                 <ElementValue elementID="cf-add_2-city" value="Malibu">
12                     <ElementValue elementID="cf-add_1-street" value="Locust"
13                 >
14             </SubElement>
15         </SubElement>
16         <SubElement parentID="root" childID="mh" position="2">
17             <ElementAttribute elementID="mh-starsIn" name="starsIn"
18             value="sw"/>
19             <ElementValue elementID="mh-name" value="Mark Hamill"/>
20             <SubElement parentID="mh" childID="mh-add_2" position="1">
21                 <ElementValue elementID="mh-add_2-city" value="Bollywood
22             >
23                 <ElementValue elementID="mh-add_2-street" value="Oak">
24             </SubElement>
25         </SubElement>
26         <SubElement parentID="root" childID="sw" position="3">
27             <ElementAttribute elementID="mh-starsOf_1" name="starsOf"
28             value="cf"/>
29             <ElementAttribute elementID="mh-starsOf_2" name="starsOf"
30             value="mh"/>
31             <ElementValue elementID="sw-title" value="Star Wars" />
32             <ElementValue elementID="sw-year" value="1977" />
33         </SubElement>
34     </DocRoot>

```

9. 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          <Address>
17              <Street>456 Oak Rd.</Street>
18              <City>Brentwood</City>
19          </Address>
20      </Star>
21      <Movie movieID="sw" starsOf="cf mh">

```

```

22         <Title>Star Wars</Title>
23         <Year>1977</Year>
24     </Movie>
25     <Movie movieID="esb" starsOf="cf mh">
26         <Title>The Empire Strikes Back</Title>
27         <Year>1980</Year>
28     </Movie>
29     <Movie movieID="roj" starsOf="cf mh">
30         <Title>Return of the Jedi</Title>
31         <Year>1983</Year>
32     </Movie>
33 </StarMovieData>
34

```

```

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="hf" starredIn="sw fw">
15         <Name>Harrison Ford</Name>
16     </Star>
17
18     <Star starID="mh" starredIn="sw">
19         <Name>Mark Hamill</Name>
20         <Address>
21             <Street>456 Oak Rd.</Street>
22             <City>Brentwood</City>
23         </Address>
24     </Star>
25     <Movie movieID="sw" starsOf="cf mh hf">
26         <Title>Star Wars</Title>
27         <Year>1977</Year>
28     </Movie>
29     <Movie movieID="esb" starsOf="cf mh hf">
30         <Title>The Empire Strikes Back</Title>
31         <Year>1980</Year>
32     </Movie>
33     <Movie movieID="roj" starsOf="cf mh hf">
34         <Title>Return of the Jedi</Title>
35         <Year>1983</Year>
36     </Movie>
37     <Movie movieID="fw" starsOf="hf">
38         <Title>Firewall</Title>
39         <Year>2006</Year>
40     </Movie>

```

```

41     </StarMovieData>
42

```

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="hf" starredIn="sw fw">
16         <Name>Harrison Ford</Name>
17     </Star>
18
19     <Star starID="mh" starredIn="sw">
20         <Name>Mark Hamill</Name>
21         <Address>
22             <Street>456 Oak Rd.</Street>
23             <City>Brentwood</City>
24         </Address>
25     </Star>
26     <Movie movieID="sw" starsOf="cf mh hf">
27         <Title>Star Wars</Title>
28         <Year>1977</Year>
29     </Movie>
30     <Movie movieID="esb" starsOf="cf mh hf">
31         <Title>The Empire Strikes Back</Title>
32         <Year>1980</Year>
33     </Movie>
34     <Movie movieID="roj" starsOf="cf mh hf">
35         <Title>Return of the Jedi</Title>
36         <Year>1983</Year>
37     </Movie>
38     <Movie movieID="fw" starsOf="hf">
39         <Title>Firewall</Title>
40         <Year>2006</Year>
41     </Movie>
42     <Movie movieID="hhs" starsOf="cf">
43         <Title>Hannah and Her Sisters</Title>
44         <Year>1985</Year>
45     </Movie>
46 </StarMovieData>

```

d)

```

2  <? xml version = "1.0" encoding="utf-8" standalone = "yes">
3  <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="hf" starredIn="sw fw">
15         <Name>Harrison Ford</Name>
16     </Star>
17     <Star starID="mh" starredIn="sw">
18         <Name>Mark Hamill</Name>
19         <Address>
20             <Street>456 Oak Rd.</Street>
21             <City>Brentwood</City>
22         </Address>
23     </Star>
24     <Star starID="md" starredIn="bi">
25         <Name>Matt Damon</Name>
26     </Star>
27     <Movie movieID="sw" starsOf="cf mh hf">
28         <Title>Star Wars</Title>
29         <Year>1977</Year>
30     </Movie>
31     <Movie movieID="esb" starsOf="cf mh hf">
32         <Title>The Empire Strikes Back</Title>
33         <Year>1980</Year>
34     </Movie>
35     <Movie movieID="roj" starsOf="cf mh hf">
36         <Title>Return of the Jedi</Title>
37         <Year>1983</Year>
38     </Movie>
39     <Movie movieID="fw" starsOf="hf">
40         <Title>Firewall</Title>
41         <Year>2006</Year>
42     </Movie>
43     <Movie movieID="hhs" starsOf="cf">
44         <Title>Hannah and Her Sisters</Title>
45         <Year>1985</Year>
46     </Movie>
47     <Movie movieID="bi" starsOf="md">
48         <Title>The Bourne Identity</Title>
49         <Year>2002</Year>
50     </Movie>
51 </StarMovieData>
52

```

```

10< !DOCTYPE BankData [
2    < !ELEMENT BankData (Accounts*, Customers*)>
3    < !ELEMENT Accounts(acctNo, type, balance)>

```

```

4      <!--ATTLIST Accounts
5          acctNo ID #REQUIRED
6          acctIn IDREFS #IMPLIED
7      >
8      <!--ELEMENT Customers(firsrName, lastName, idNo, account)>
9          <!--ATTLIST Accounts
10             idNo ID #REQUIRED
11             account IDREFS #IMPLIED
12         >
13         <!--ELEMENT acctNo (#PCDATA)>
14         <!--ELEMENT type (#PCDATA)>
15         <!--ELEMENT balance (#PCDATA)>
16         <!--ELEMENT firstName (#PCDATA)>
17         <!--ELEMENT lastName (#PCDATA)>
18         <!--ELEMENT idNo (#PCDATA)>
19         <!--ELEMENT account (#PCDATA)>
20     ]>
21

```

Correct Solution:

```

1      <!--DOCTYPE BankData [
2          <!--ELEMENT BankData (Accounts*, Customers*)>
3          <!--ELEMENT Accounts(acctNo, type, balance)>
4              <!--ATTLIST Accounts
5                  acctNo ID #REQUIRED
6                  acctIn IDREFS #IMPLIED
7              >
8          <!--ELEMENT Customers(firsrName, lastName, idNo, account)>
9              <!--ATTLIST Accounts
10                  idNo ID #REQUIRED
11                  account IDREFS #IMPLIED
12              >
13          <!--ELEMENT acctNo (#PCDATA)>
14          <!--ELEMENT type (#PCDATA)>
15          <!--ELEMENT balance (#PCDATA)>
16          <!--ELEMENT firstName (#PCDATA)>
17          <!--ELEMENT lastName (#PCDATA)>
18          <!--ELEMENT idNo (#PCDATA)>
19          <!--ELEMENT account (#PCDATA)>
20      ]>
21

```

```

11     <!--DOCTYPE TeamData [
12         <!--ELEMENT TeamData (Players*, Teams*, Fans*)>
13         <!--ELEMENT Players(FirstName, LastName, Team)>
14             <!--ATTLIST Players
15                 playerId ID #REQUIRED
16                 playerIn IDREFS #IMPLIED
17             >
18         <!--ELEMENT FirstName (#PCDATA)>

```

```

9      <![ELEMENT LastName (#PCDATA)]>
10     <![ELEMENT Team (#PCDATA)]>
11     <![ELEMENT Teams(Name)]>
12         <![ATTLIST Teams
13             teamId ID #REQUIRED
14             playerOf IDREFS #IMPLIED
15             fanOf IDREFS #IMPLIED
16         >
17     <![ELEMENT Name (#PCDATA)]>
18     <![ELEMENT Fans(FirstName, LastName)]>
19         <![ATTLIST Fans
20             faId ID #REQUIRED
21             fanIn IDREFS #IMPLIED
22         >
23 ]>
24

```

Notes:

- There is no need to write same tags twice. (e.g. FirstName, LastName)

```

12<![DOCTYPE GenealogyData [
2    <![ELEMENT GenealogyData (Person)]>
3    <![ELEMENT Person(FirstName, LastName, Age)]>
4        <![ATTLIST Person
5            personId ID #REQUIRED
6            spouse IDREFS #IMPLIED
7            husband IDREFS #IMPLIED
8            child IDREFS #IMPLIED
9            brother IDREFS #IMPLIED
10        >
11    <![ELEMENT FirstName (#PCDATA)]>
12    <![ELEMENT LastName (#PCDATA)]>
13 ]>
14

```

Notes:

- PCDATA
 - Is text that will be parsed by a parser.
 - Tags inside the text will be treated as markup and entities will be expanded.
- CDATA
 - Is text that will not be parsed by a parser.
 - Tags inside the text will not be treated as markup and entities will not be expanded
 - e.g. attributes

```

1      <Movie title="Star Wars" year="1977" genre="sciFi">
2
3      <!--ELEMENT Movie EMPTY-->
4          <!--ATTLIST Movie
5              title CDATA #REQUIRED
6              year  CDATA #REQUIRED
7              genre (comedy | drama | sciFi | teen) #IMPLIED
8          -->
9
10

```

```

13<!--DOCTYPE ShipBattleData [
2
3    ]>
4

```

14. • An example of a document that conforms to the XML Schema

```

1      <? xml version = "1.0" encoding = "utf-8" ?>
2      <Movies>
3          <Movie>
4              <Title>Star Wars</Title>
5              <Year>1977</Year>
6          </Movie>
7          <Movie>
8              <Title>Return of the Jedi</Title>
9              <Year>1983</Year>
10         </Movie>
11     </Movies>
12

```

- An example of one that has all the elements mentioned, but does not conform to definition

```

1      <? xml version = "1.0" encoding = "utf-8" ?>
2      <Movies>
3          <Movie>
4              <Title>Star Wars</Title>
5              <Year>1977</Year>
6          </Movie>
7          <Movie>
8              <Title>Return of the Jedi</Title>
9              <Year>This is not correct</Year>
10         </Movie>
11     </Movies>
12

```

Notes:

- XML Schema
 - is an alternative way to provide a schema for XML documents

- is more powerful than DTD
 - * allows to declare types such as integer or float
 - * gives ability to declare keys and foreign keys

- The Form of an XML Schema

- Each XML-Schema document has the form

```
<? xml version = "1.0" encoding = "utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  ...
</xs:schema>
```

- Elements

- **Syntax:**

```
<xs:element name = element name type = element type>
  constraints and / or structure information
</xs:element>
```

Example:

```
1 <xs:element name = "title" type = "xs:string" />
2 <xs:element name = "Year" type = "xs:integer" />
3
4
```

- Complex Types

- is an XML element that contains other elements and/or attributes. ^[1]
- **Syntax:**

```
<xs:complexType name = type name>
  <xs:sequence>
    list of element definitions
  </xs:sequence>
</xs:complexType>
```

Example:

```
1 <? xml version = "1.0" encoding = "utf-8" ?>
2 <xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
3
4   <xs:complexType name = "movieType">
5     <xs:sequence>
6       <xs:element name = "Title" type = "xs:string" />
7       <xs:element name = "year" type = "xs:integer" />
8     </xs:sequence>
9   </xs:complexType>
```

```

10
11     <xs:element name = "Movies">
12         <xs:complexType>
13             <xs:sequence>
14                 <xs:element name="Movie" type="movieType"
minOccurs="0" maxOccurs = "unbounded" />
15             </xs:sequence>
16         </xs:complexType>
17     </xs:element>
18 </xs:schema>
19

```

- Complex Types Attributes

- **Syntax:**

`<xs:attribute name = attribute name type = type name other information about the attribute />`

* *other information* includes

1. default value
2. required
3. type

Example:

```

1) <? xml version = "1.0" encoding = "utf-8" ?>
2) <xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">

3)   <xs:complexType name = "movieType">
4)       <xs:attribute name = "title" type = "xs:string"
        use = "required" />
5)       <xs:attribute name = "year" type = "xs:integer"
        use = "required" />
6)   </xs:complexType>

7)   <xs:element name = "Movies">
8)       <xs:complexType>
9)         <xs:sequence>
10)            <xs:element name = "Movie" type = "movieType"
minOccurs = "0" maxOccurs = "unbounded" />
11)        </xs:sequence>
12)    </xs:complexType>
13) </xs:element>

14) </xs:schema>

```

Above allows us to have the tag `<Movies title="value" year="1">`

- Restricted Simple Types

- Can be used as an attribute or element

1. Restricting numerical values by setting lower bound and upperbound

* **Syntax:**

`<xs:simpleType name = type name>`

```

    <xs:restriction base = base type >
        upper and/or lower bounds
    </xs:restriction >
</xs:simpleType>
* Use minInclusive to state the lower bound
* Use maxInclusive to state the upper bound

```

Example:

```

1      <xs:simpleType name = "movieYearType">
2          <xs:restriction base = "xs:integer">
3              <xs:minInclusive value = "1915" />
4          </xs:restriction>
5      </xs:simpleType>
6

```

2. Restricting values to an enumerated type* **Syntax:**

```

<xs:simpleType name = "movieYearType" >
    <xs:restriction base = "xs:string">
        <xs:enumeration value = "comedy" / >
        <xs:enumeration value = "drama" / >
        <xs:enumeration value = "sciFi" / >
        <xs:enumeration value = "teen" / >
    </xs:restriction>
</xs:simpleType>

```

Example:

```

1      <xs:simpleType name = "genreType">
2          <xs:restriction base = "xs:string">
3              <xs:enumeration value = "comedy"/>
4              <xs:enumeration value = "drama"/>
5              <xs:enumeration value = "sciFi"/>
6              <xs:enumeration value = "teen"/>
7          </xs:restriction>
8      </xs:simpleType>
9

```

• Keys in XML Schema

– **Syntax:**

```

<xs:key name = key name>
    <xs:selector xpath= path description >
    <xs:field xpath = path description >
</xs:key>

```

Example:

```

1) <? xml version = "1.0" encoding = "utf-8" ?>
2) <xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">

3) <xs:simpleType name = "genreType">
4)   <xs:restriction base = "xs:string">
5)     <xs:enumeration value = "comedy" />
6)     <xs:enumeration value = "drama" />
7)     <xs:enumeration value = "sciFi" />
8)     <xs:enumeration value = "teen" />
9)   </xs:restriction>
10) </xs:simpleType>

11) <xs:complexType name = "movieType">
12)   <xs:sequence>
13)     <xs:element name = "Title" type = "xs:string" />
14)     <xs:element name = "Year" type = "xs:integer" />
15)     <xs:element name = "Genre" type = "genreType"
16)       minOccurs = "0" maxOccurs = "1" />
17)   </xs:sequence>
18) </xs:complexType>

19) <xs:element name = "Movies">
20)   <xs:complexType>
21)     <xs:sequence>
22)       <xs:element name = "Movie" type = "movieType"
23)         minOccurs = "0" maxOccurs = "unbounded" />
24)     </xs:sequence>
25)   </xs:complexType>
26)   <xs:key name = "movieKey">
27)     <xs:selector xpath = "Movie" />
28)     <xs:field xpath = "Title" />
29)     <xs:field xpath = "Year" />
30)   </xs:key>
31) </xs:element>

32) </xs:schema>

```

where key is attached to (e.g. Movies)

the values which are set as primary key

– XPath of element → *element name*


```

11)    <xs:complexType name = "movieType">
12)      <xs:sequence>
13)        <xs:element name = "Title" type = "xs:string" />
14)        <xs:element name = "Year" type = "xs:integer" />
15)        <xs:element name = "Genre" type = "genreType"
16)          minOccurs = "0" maxOccurs = "1" />
17)      </xs:sequence>
18)    </xs:complexType>

```

```

24)    <xs:key name = "movieKey">
25)      <xs:selector xpath = "Movie" />
26)      <xs:field xpath = "Title" />
27)      <xs:field xpath = "Year" />
28)    </xs:key>

```

xpath of element
uses name of element

– XPath of attribute → @*attribute name*

```

<xs:complexType>
  <xs:attribute name = "title"
    type = "xs:string" />
  <xs:attribute name = "year"
    type = "xs:integer" />
</xs:complexType>

```

```

22)    <xs:keyref name = "movieRef" refers = "movieKey">
23)      <xs:selector xpath = "Star/StarredIn" />
24)      <xs:field xpath = "@title" />
25)      <xs:field xpath = "@year" />
26)    </xs:keyref>

```

xpath of attribute name
has @ at front

- Foreign Keys in XML Schema

– Syntax:

```

<xs:keyref name = foreign-key name refer = key name >
  <xs:selector xpath = path description >
  <xs:field xpath = path description >

```

</xs:keyref >

References:

1. w3School : Complex Type, link