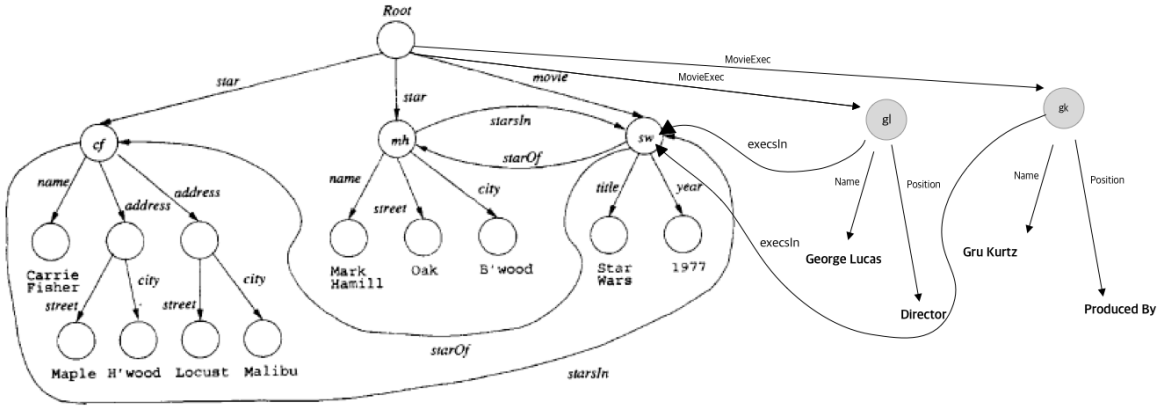
CSC343 Worksheet 9 Solution

June 27, 2020

1. **Exercise 11.1.1:**

a)



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

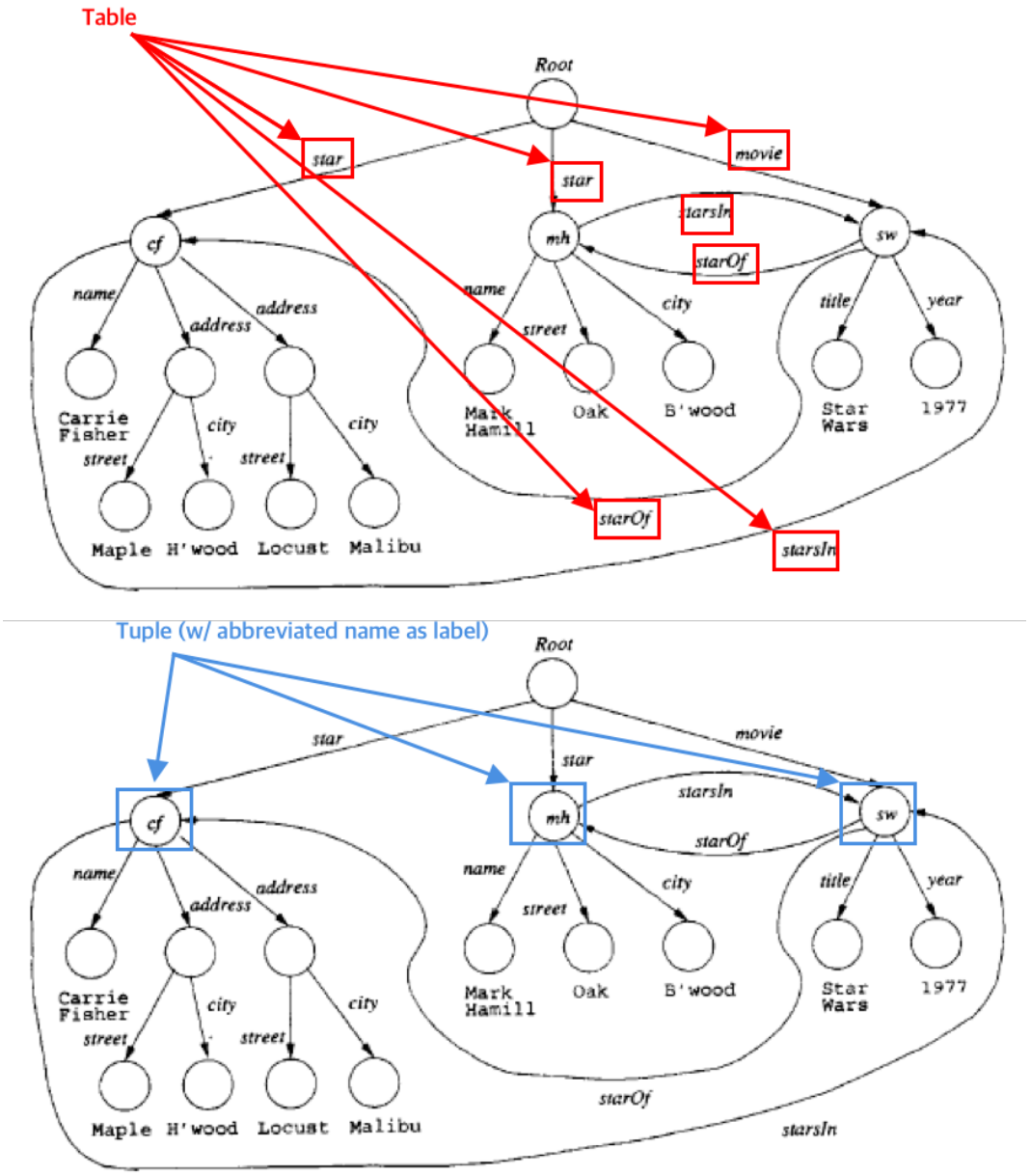
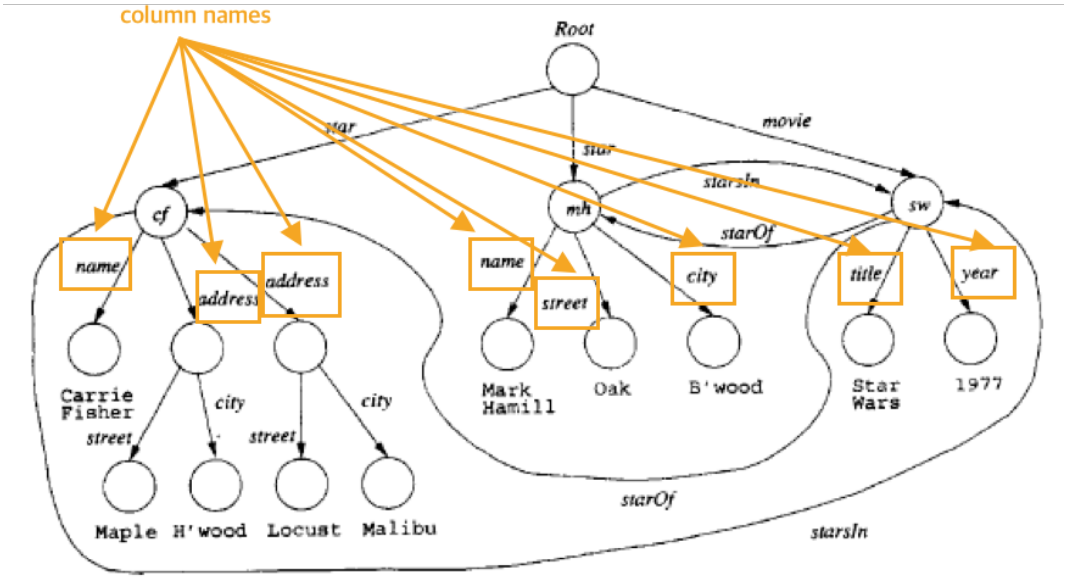
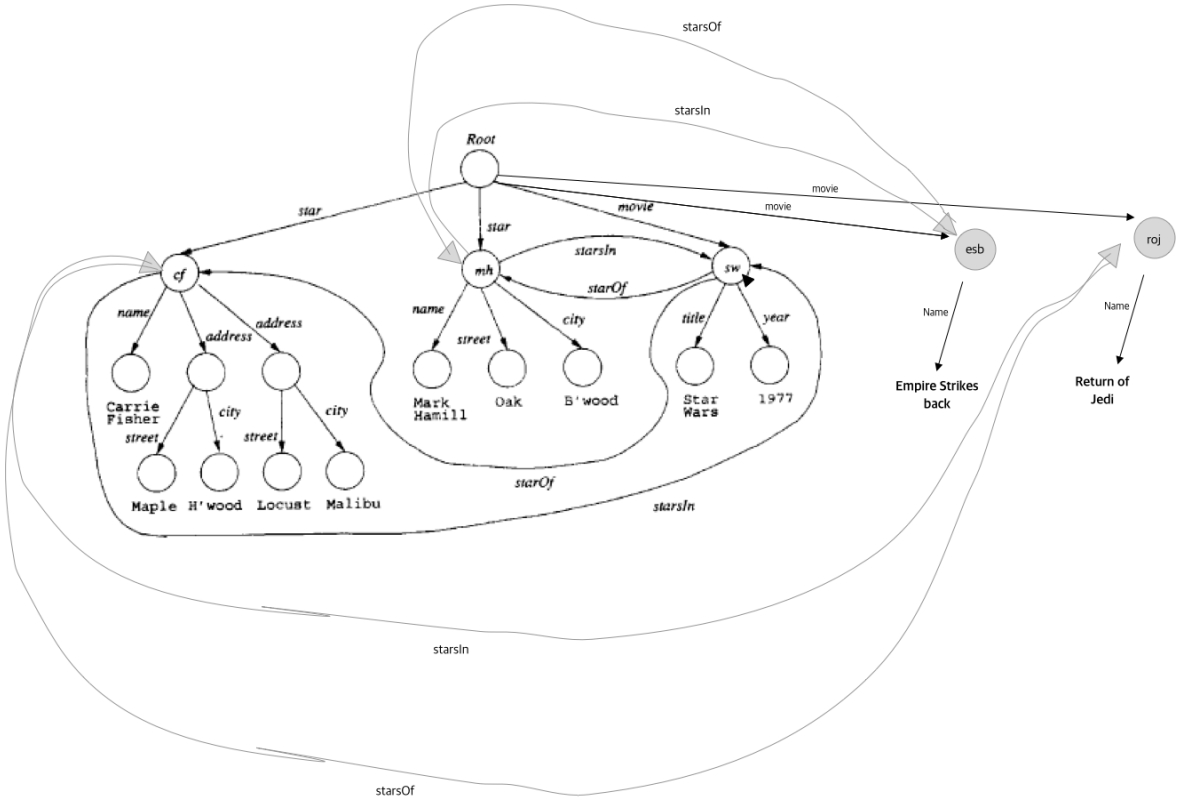
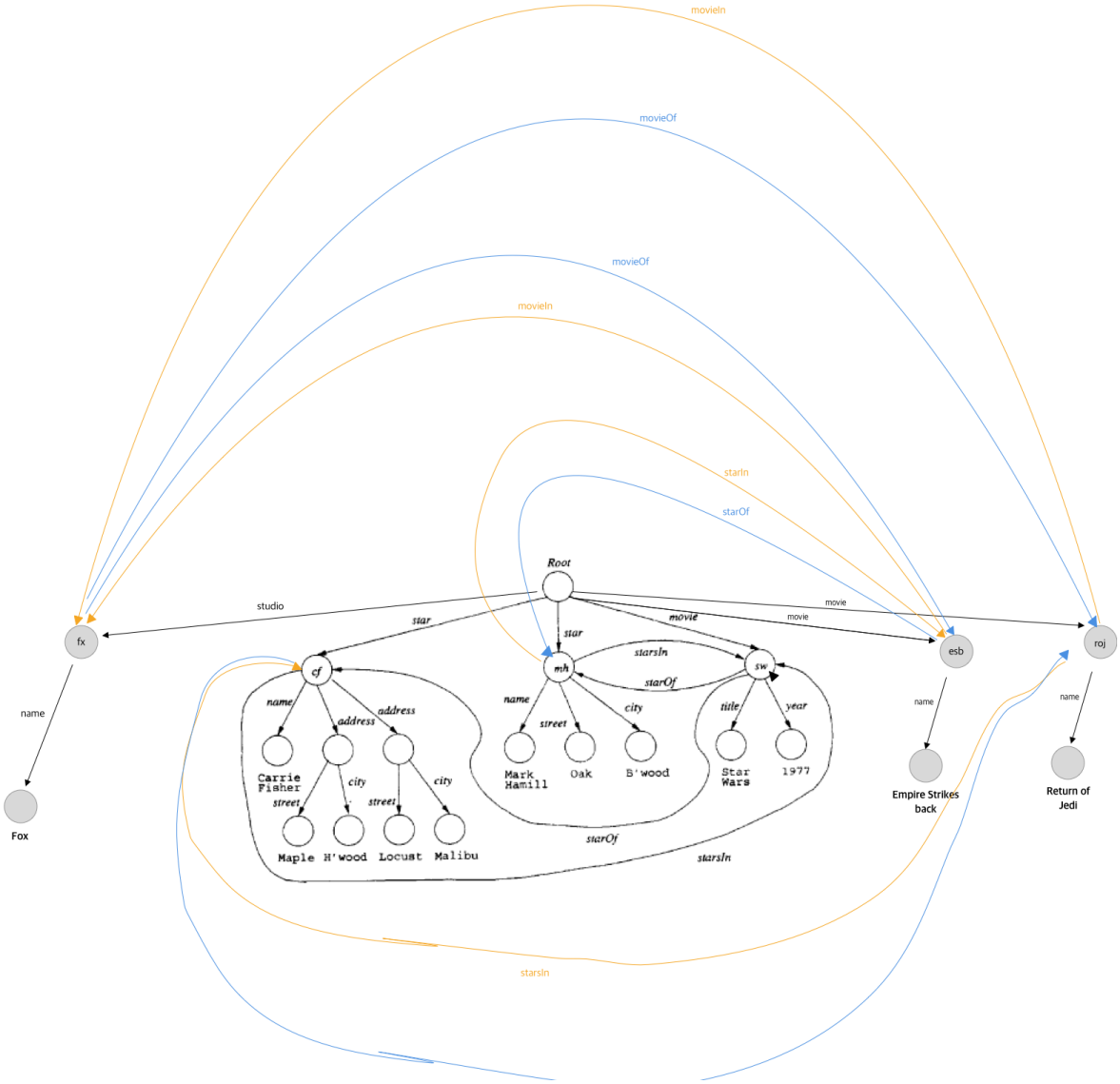


Table
Root
star
star
movie
cf
mh
sw
starsIn
starOf
name
address
address
name
street
city
title
year
Carrie Fisher
city
city
Mark Hamill
Oak
B'wood
Star Wars
1977
street
street
Maple
H'wood
Locust
Malibu
starOf
starsIn
Tuple (w/ abbreviated name as label)
Root
star
star
movie
cf
mh
sw
starsIn
starOf
name
address
address
name
street
city
title
year
Carrie Fisher
city
city
Mark Hamill
Oak
B'wood
Star Wars
1977
street
street
Maple
H'wood
Locust
Malibu
starOf
starsIn


column names
Root
star
movie
star
starsIn
cf
mh
sw
starOf
name
address
address
name
street
city
title
year
Carrie Fisher
city
city
street
street
Mark Hamill
Oak
B'wood
Star Wars
1977
Maple
H'wood
Locust
Malibu
starOf
starsIn

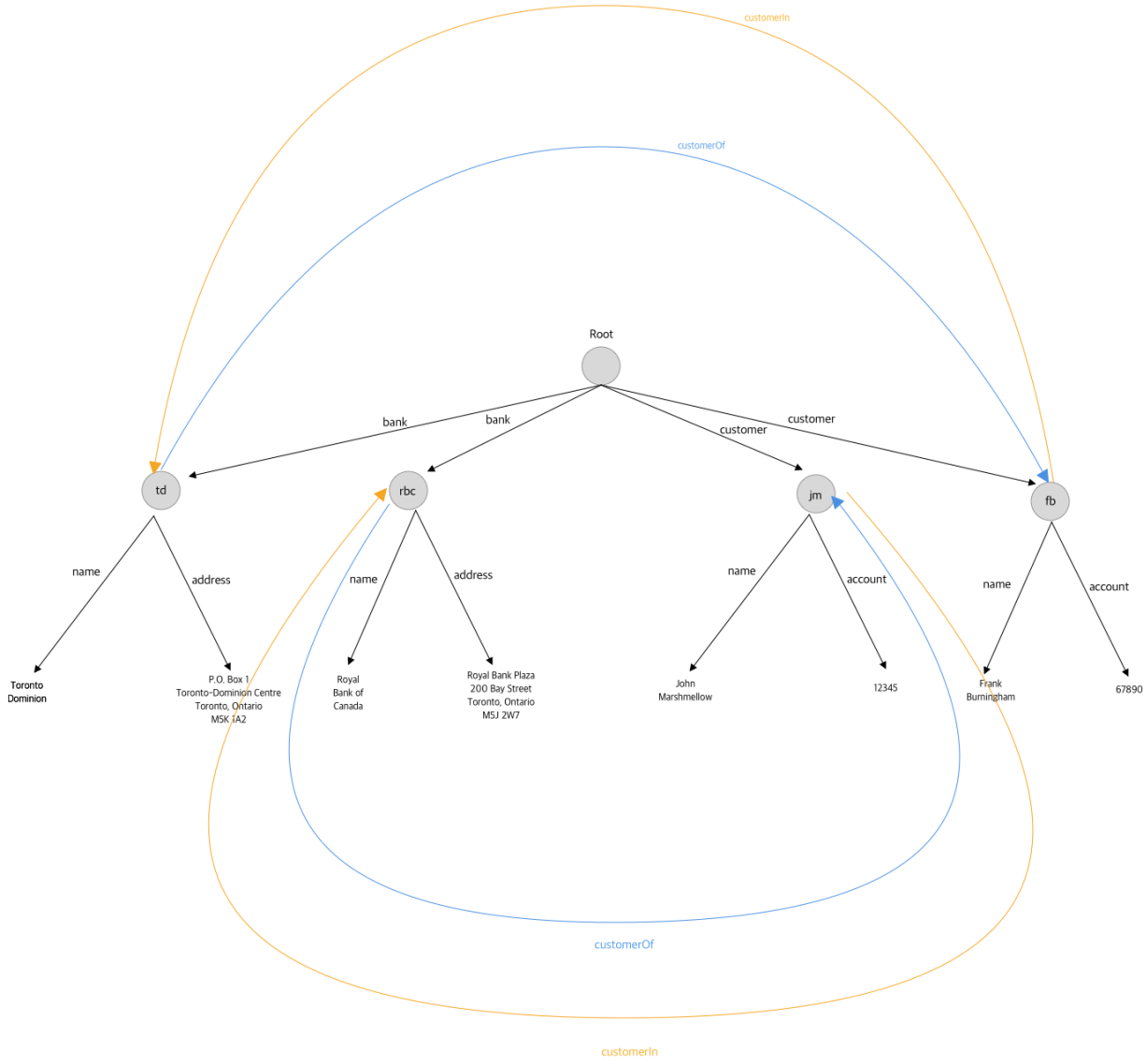
- b)

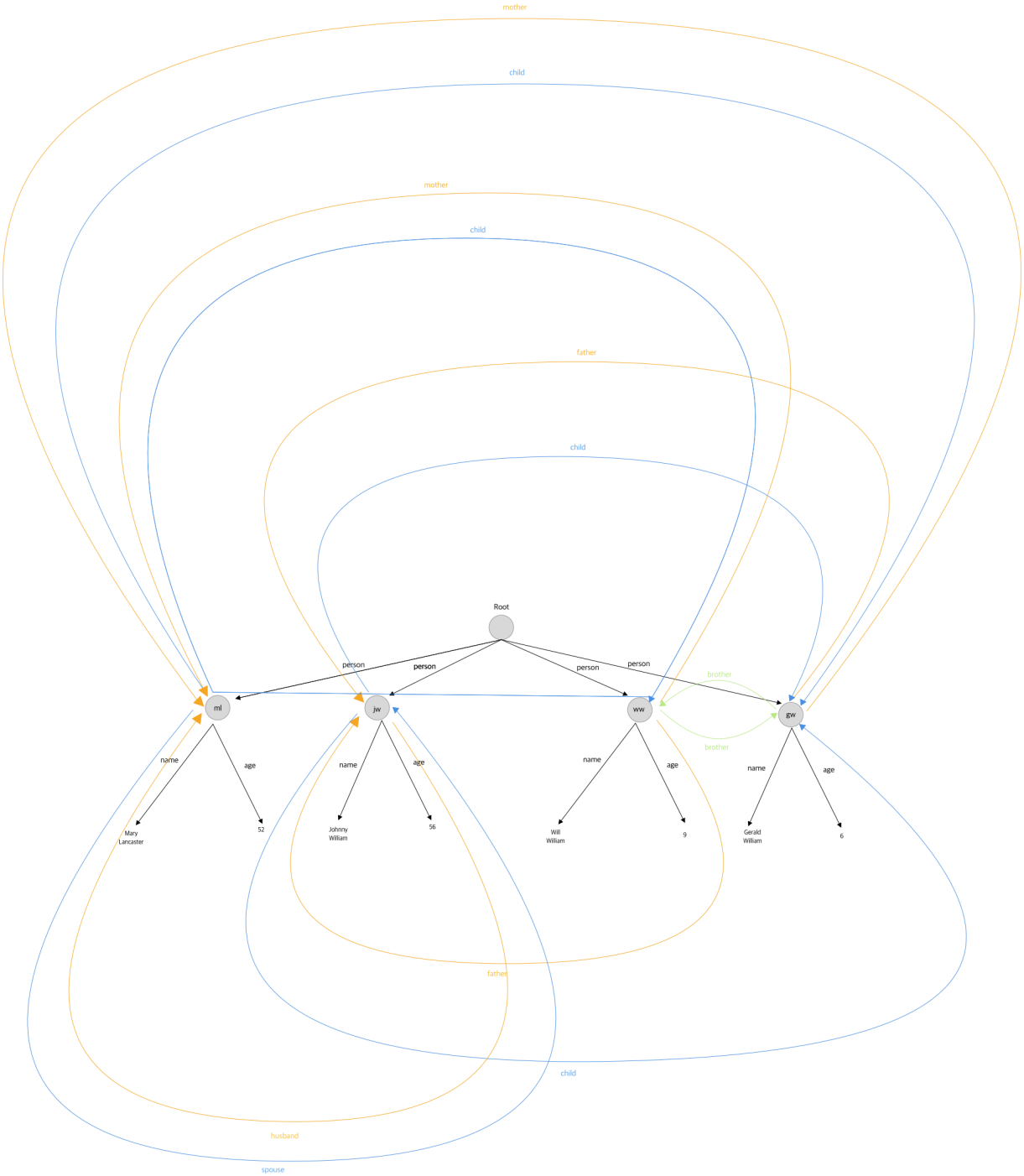

starsOf
starsIn
Root
movie
movie
movie
star
star
cf
mh
sw
starsIn
starOf
esb
roj
name
address
address
name
street
city
title
year
Name
Name
Carrie Fisher
city
city
street
street
Mark Hamill
Oak
B'wood
Star Wars
1977
Empire Strikes back
Return of Jedi
Maple
H'wood
Locust
Malibu
starOf
starsIn
starsIn
starsOf


movieIn
movieOf
movieOf
movieIn
starIn
starOf
Root
studio
star
star
movie
movie
movie
fx
cf
mh
sw
esb
roj
starsIn
starOf
name
name
address
address
city
city
street
street
name
street
city
title
year
name
name
Carrie Fisher
Maple
H'wood
Locust
Malibu
Mark Hamill
Oak
B'wood
Star Wars
1977
Fox
Empire Strikes back
Return of Jedi
starOf
starsIn
starsIn

c)

2.


customerIn
customerOf
Root
bank
bank
customer
customer
td
rbc
jm
fb
name
address
name
address
name
account
name
account
Toronto
Dominion
P.O. Box 1
Toronto-Dominion Centre
Toronto, Ontario
M5K 1A2
Royal
Bank of
Canada
Royal Bank Plaza
200 Bay Street
Toronto, Ontario
M5J 2W7
John
Marshmellow
12345
Frank
Burningham
67890
customerOf
customerIn



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

<u>Notees:</u>

- 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)

```
<? xml version = "1.0" encoding="utf-8" standalone = "yes">
<StarMovieData>
    <Star starID="cf" starredIn="sw">
        <Name>Carrie Fisher</Name>
        <Address>
            <Street>123 Maple St.</Street>
            <City>Hollywood</City>
        </Address>
        <Address>
            <Street>5 Locust Ln.</Street>
            <City>Malibu</City>
        </Address>
    </Star>
    <Star starID="mh" starredIn="sw">
        <Name>Mark Hamill</Name>
        <Street>456 Oak Rd.</Street>
        <City>Brentwood</City>
    </Star>
    <Movie movieID="sw" starsOf="cf", "mh">
        <Title>Star Wars</Title>
        <Year>1977</Year>
    </Movie>
    <MovieExec movieExecID="gl" execsIn="sw">
        <Name>George Lucas</Name>
        <Position>Director</Position>
    </MovieExec>
    <MovieExec movieExecID="gk" execsIn="sw">
        <Name>Gru Kurtz</Name>
        <Position>Produced By</Position>
    </MovieExec>
</StarMovieData>

```

- 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:

```
<? xml version = "1.0" encoding="utf-8" standalone = "yes
">
<StarMovieData>
    <Star>
        <Name>Carrie Fisher</Name>
        <Address>
            <Street>123 Maple St.</Street>
            <City>Hollywood</City>
        </Address>
        <Address>
            <Street>5 Locust Ln.</Street>
            <City>Malibu</City>
        </Address>
    </Star>
    <Star>
        <Name>Mark Hamill</Name>
        <Street>456 Oak Rd.</Street>
        <City>Brentwood</City>
    </Star>
    <Movie>
        <Title>Star Wars</Title>
        <Year>1977</Year>
    </Movie>
</StarMovieData>

```

  2. Valid XML
     * Involes "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:

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

```
            </Address>
            <Address>
                <Street>5 Locust Ln.</Street>
                <City>Malibu</City>
            </Address>
        </Star>
        <Star starID="mh" starredIn="sw">
            <Name>Mark Hamill</Name>
            <Street>456 Oak Rd.</Street>
            <City>Brentwood</City>
        </Star>
        <Movie starID="sw" starOf="cf", "mh">
            <Title>Star Wars</Title>
            <Year>1977</Year>
        </Movie>
    </StarMovieData>

```

- 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:

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

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

```

b)

```
<? xml version = "1.0" encoding="utf-8" standalone = "yes">
<StarMovieData>
    <Star starID="cf" starredIn="sw">
        <Name>Carrie Fisher</Name>
        <Address>
            <Street>123 Maple St.</Street>
            <City>Hollywood</City>
        </Address>
        <Address>
            <Street>5 Locust Ln.</Street>
            <City>Malibu</City>
        </Address>
    </Star>
    <Star starID="mh" starredIn="sw">
        <Name>Mark Hamill</Name>
        <Street>456 Oak Rd.</Street>
        <City>Brentwood</City>
    </Star>
    <Movie movieID="sw" starsOf="cf", "mh">
        <Title>Star Wars</Title>
        <Year>1977</Year>
```

```
        </Movie>
        <Movie movieID="esb" starOf="cf", "mh">
            <Title>Empire Strikes Back</Title>
            <Year>1980</Year>
        </Movie>
        <Movie movieID="roj" starOf="cf", "mh">
            <Title>Return of Jedi</Title>
            <Year>1983</Year>
        </Movie>
    </StarMovieData>
```

c)

```
<? xml version = "1.0" encoding="utf-8" standalone = "yes">
<StarMovieData>
    <Star starID="cf" starredIn="sw">
        <Name>Carrie Fisher</Name>
        <Address>
            <Street>123 Maple St.</Street>
            <City>Hollywood</City>
        </Address>
        <Address>
            <Street>5 Locust Ln.</Street>
            <City>Malibu</City>
        </Address>
    </Star>
    <Star starID="mh" starredIn="sw">
        <Name>Mark Hamill</Name>
        <Street>456 Oak Rd.</Street>
        <City>Brentwood</City>
    </Star>
    <Movie movieID="sw" starsOf="cf", "mh" movieIn="fx">
        <Title>Star Wars</Title>
        <Year>1977</Year>
    </Movie>
    <Movie movieID="esb" starOf="cf", "mh" movieIn="fx">
        <Title>Empire Strikes Back</Title>
        <Year>1980</Year>
    </Movie>
    <Movie movieID="roj" starOf="cf", "mh" movieIn="fx">
        <Title>Return of Jedi</Title>
        <Year>1983</Year>
    </Movie>
    <Studio studioID="fx" movieOf="esb", "roj", "sw">
        <Name>Fox</Name>
        <Address>Hollywood</Address>
    </Studio>
</StarMovieData>
```

6. Consider the following relation Classes:

class	*type*	*country*	*numGuns*	*bore*	*displacement*
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**

```
<? xml version = "1.0" encoding="utf-8" standalone = "yes">
<shipsData>
    <Classes>
        <Class>Bismarck</Class>
        <Type>bb</Type>
        <Country>Germany</Country>
        <NumGuns>8</NumGuns>
        <Bore>15</Bore>
        <Displacement>42000</Displacement>
    </Classes>
    <Classes>
        <Class>Iowa</Class>
        <Type>bb</Type>
        <Country>USA</Country>
        <NumGuns>9</NumGuns>
        <Bore>16</Bore>
        <Displacement>46000</Displacement>
    </Classes>
    <Classes>
        <Class>Kongo</Class>
        <Type>bc</Type>
        <Country>Japan</Country>
        <NumGuns>8</NumGuns>
        <Bore>14</Bore>
        <Displacement>32000</Displacement>
    </Classes>
    <Classes>
        <Class>North Carolina</Class>
        <Type>bb</Type>
        <Country>USA</Country>
        <NumGuns>9</NumGuns>
        <Bore>16</Bore>
        <Displacement>37000</Displacement>
    </Classes>
    <Classes>
        <Class>Renown</Class>
```

```
            <Type>bc</Type>
            <Country>Gt. Britain</Country>
            <NumGuns>6</NumGuns>
            <Bore>15</Bore>
            <Displacement>32000</Displacement>
        </Classes>
        <Classes>
            <Class>Revenge</Class>
            <Type>bb</Type>
            <Country>Gt. Britain</Country>
            <NumGuns>8</NumGuns>
            <Bore>15</Bore>
            <Displacement>29000</Displacement>
        </Classes>
        <Classes>
            <Class>Tennessee</Class>
            <Type>bb</Type>
            <Country>USA</Country>
            <NumGuns>12</NumGuns>
            <Bore>14</Bore>
            <Displacement>32000</Displacement>
        </Classes>
        <Classes>
            <Class>Yamato</Class>
            <Type>bb</Type>
            <Country>Japan</Country>
            <NumGuns>9</NumGuns>
            <Bore>18</Bore>
            <Displacement>65000</Displacement>
        </Classes>
    </shipsData>

```

7.

- DocRoot

```
    <DocRoot docID="" rootElementID="" />

```

- SubElement

```
    <Subelement parentID="" childID="" position="" />

```

- ElementAttribute

```
    <ElementAttribute elementID="" name="" value="" />

```

- ElementValue

```
    <ElementValue elementID="" name="" value="" />

```

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:

```
    <phone>
        <entry>
            <name>Chip</name>
            <extension number="3"/>
        </entry>
    </phone>

```

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

```
<? xml version = "1.0" encoding="utf-8" standalone = "yes">
<DocRoot docID="root" rootElementID="">
    <Subelement parentID="" childID="" position="">
        <ElementAttribute elementID="" name="" value="">

        </ElementAttribute>
        <ElementValue elementID="" name="" value="">

        </ElementValue>
    </Subelement>
</DocRoot>

```