

## CSE 350: Advanced DB Topics

### Homework 3: OODBs, XML

#### SOLUTIONS

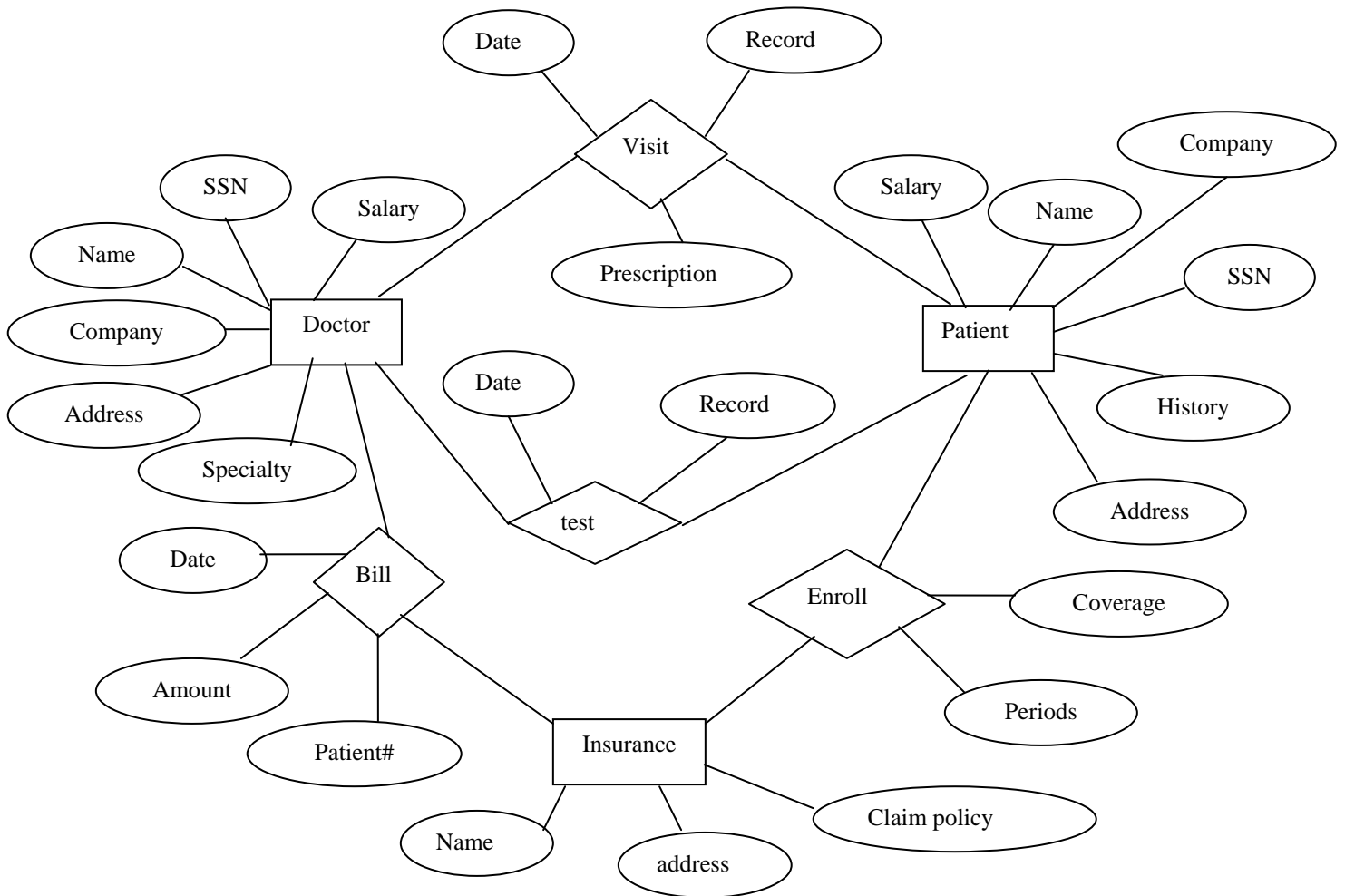
---

Pretend you work in a DB company, and your boss asked you to write reports for him on the following two issues (**problems 1 and 2**). The first one is a spec for a design of a new OODB for your customers, and the second one is more theoretical, because he does not know anything about OODBs and he wants to learn from you. Be thorough and explain everything well.

#### Problem 1

We have a small medical office, with *employees*, *doctors*, and *patients*. (Note: Doctors are employees; employees can be patients, too). We also have *office visits*, *prescriptions*, and *bills*.

a) Create an ER model for this scenario, then translate it into an OODB schema. What are the top-level objects? Who has what attributes? Who has what methods? Who inherits what from whom?



The corresponding OODB Schema:

```
schema MedicalOffice;

class Address:
  type tuple (
    country: string,
    city: string,
    street: string,
    zip: integer);

class person
  type tuple (
    SSN: string,
    public name: string,
    public addr: Address
with extent;

class Employee inherit Person
  type tuple (
    public emp_no: integer,
    salary: integer)
  method public
    set_Salary(salary: integer),
    get_Salary: integer
with extent;

class Doctor inherit Employee
  type tuple (
    public Specialty: string) //dentist, dermatologist, ...
with extent;

class Bill // things that insurance has to pay for
  type tuple (
    date: Date,
    patient: Patient,
    doctor: Doctor
    amount: number
  )
end;

class Patient inherit Person
  type tuple (
    History: setof(Bill)), //medical history
    Insured_by: Insurance,
    Claim_no: int
  )
  method public add_history(Bill) //add the bill to patient's history
end;

class Visit inherit Bill
  type tuple (
    Record: string,
    Prescription: string,
  )
  method public
    get_Record: string,
    set_record(string),
    set_Prescription(string),
    get_Prescription: string,
with extent;

class Test inherit Bill
  type tuple
  (
    test_type: string,
```

```

        result: string
    )
end;

class Insurance
    type tuple (
        public Name: string,           //the name of the insurance company
        public Addr: Address,
        clients setof(Enroll)
    }
with extent;

class Enroll
    type tuple (
        time_begin: date,
        time_end: date,
        coverage_type: string,
        patient: patient,
    )
end;

```

b) Come up several typical reports that you would want to run for such an office. (Note: choose reports that will highlight the non-relational features of the data model.) Make sure your schema is sufficient to handle them (redo part a if necessary).

c) Implement these reports in O2 (on paper).

1. Find patients who cost their insurance more than \$5000 from Oct 1<sup>st</sup> to Oct 31<sup>th</sup>

```

Select p.name           // an inherited attribute
From p in Patient
Where sum (select x.amount
           from x in p.History           // history is an attr of type 'set'
           where x.date between (Oct1, Oct31) ) > $5000

```

2. Find patients who were examined by doctor "David" between Oct 1<sup>st</sup> to Oct 31<sup>th</sup>

```

Select x.patient.name
From x in visit        // 'visit' needs to be with extent
Where (x.doctor.name="David") and
      x.date between (Oct1, Oct31)

```

## **Problem 2**

The issue of keys and functional dependencies is very important for schema design of RDBs. Should we be concerned about it for ODBs as well? Discuss the motivation for this issue, and how it applies in the context of ODBs (if at all). What are the common and different aspects in the two contexts?

The keys defined on the table identify unique entities. They assure that every entity appears no more than once in the table, to avoid data duplication. When there are functional dependencies in the schema, anomalies may arise in the database; there are three types of anomalies (insertion, deletion and update anomalies). The schema must be normalized to avoid anomalies, which is achieved with schema decomposition. Keys serve a crucial role in schema decomposition, by allowing attributes in multiple tables to refer to they same entity, identified by its key.

In an OODB, when we insert a new object into the database, the system will automatically generate a unique identifier for it. One can then refer to the same object in many places without any data duplication. Values that serve as keys are not needed for identifying the object in question, because the object identifier serves this role.

As for functional dependencies, they are an important consideration in OODB design. Data anomalies are still possible, and the schema designer must be sure that his design decisions (such as which attributes to group together, and where to include copies of references to an object) avoid these anomalies.

### **Problem 3**

We have the following IllustraSQL code:

```
create table businesses of type business-t;
create table stores of new type store-t
{
...
}
under businesses;

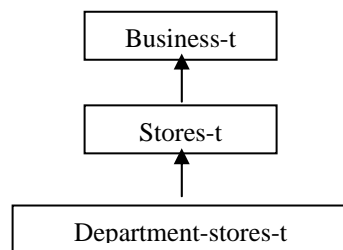
create table department-stores of new type dept-store-t
{
...
}
under stores;
```

(a) We add 3 tuples to the *stores* table and 1 tuple to the *department-stores* table, then we ask the following 2 queries:

- (i) select \* from stores;
- (ii) select \* from department-stores;

Which query will return more tuples? Which tuples will have more attributes? Why?

The inheritance relationships are as following according to the schema.



The first query will return 4 tuples (because all department stores are considered as stores) and the second one will return 1 tuple (just the department store).

Tuples in the second query will have more attributes than in the first one. Since department-stores inherit from stores, the 'department-stores' relation contains all its own attributes and methods as well as attributes and methods from 'stores' relation.

(b) Next, we create a method *change-name* for objects of type *store-t*. Will we get an error if we call *change-name* for tuples in *businesses*? What if we call it for tuples in *department-stores*? Please explain.

If we create a method *change-name* for objects of type *store-t*, we will get an error if we call it for tuples in *businesses* because *business-t* is the supertype of *stores-t* and the method *change-name* in *stores-t* is invisible to *businesses-t*. But we can call it for tuples in *department-stores* since *department-stores* inherit from *stores*. *Department-stores* will get all the data items and methods from *stores*, including the method *change-name*.

## Problem 4

a) What are the important characteristics shared by the complex object data model and the semistructured data model which are not present in the relational data model?

- 1) Values are not required to be atomic (like in relational model) but can be themselves sets, tuples or relations (sets of tuples) objects have identity
- 2) in both the complex object and semistructured data model the data can be viewed as a graph with an arbitrary structure whereas in the relational data model we have a tree of uniform structure - DB is a set of relations, relation is a set of tuples, tuples have attributes that are atomic values

b) In what way does the semistructured data model differ from the complex object data model? What are the implications of these differences for data representation and querying?

- 1) In the semistructured data model, the data is self-describing (the schema is described as part of data)
- 2) The complex object data model is ordered whereas the semistructured data model is set-based throughout.
- 3) The complex object data model implies only an arbitrary nesting of the data. The values inserted have to conform to this nested structure. In the semistructured data model, we can cope with irregular structure. elements may be missing, or different elements may have a different structure.

Facilities like regular path expressions have to be provided in order to query "uncertain" or "irregular" structure.

For example if we have defined a complex object consisting of a set of tuples with a certain tuple structure then all tuples inserted for this object are of the same structure. This is not the case in semistructured data. E.g in a complex object data model, when we define a nested structure like

Bibliography: { <Title, Authors: {<Firstname,Lastname>}> }

then the values for all tuples <Title, Authors: {<Firstname,Lastname>}> will have to fit into the same structure

Title	Authors	
	First	Last
"abc"	"f1"	"l1"
	"f2"	"l2"
"xyz"	"f1"	"l1"
	"f2"	"l2"

with semistructured data, it is possible to define tuples of (title,author) that do not all have the same structure

```

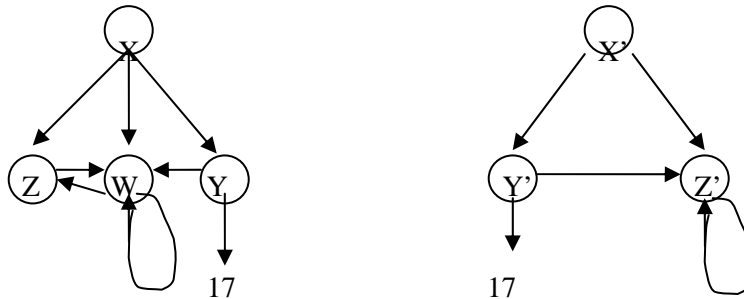
{ Bibliography
  { Title: "abc"
    Authors : {fistname: "f", lastname "l"}
    Authors "anonymous"
  }
  { Title: "xyz"
    Authors : {fistname: "f", middlename "m", lastname "l"}
    Authors "anonymous"
  }
}

```

**Problem 5 at end**

**Problem 6**

Are the following two graphs bisimilar? *Prove your answer.* (Assume all edges have the same label "L").



Claim: These graphs are bisimilar.

Proof: Let G1 and G2 be the graphs above, and let R be the following relation over the nodes of G1 and G2 (described as a set of pairs):

$(X, X'), (Y, Y'), (Z, Z'), (W, Z')$

It is easy to check that the following three statements are true:

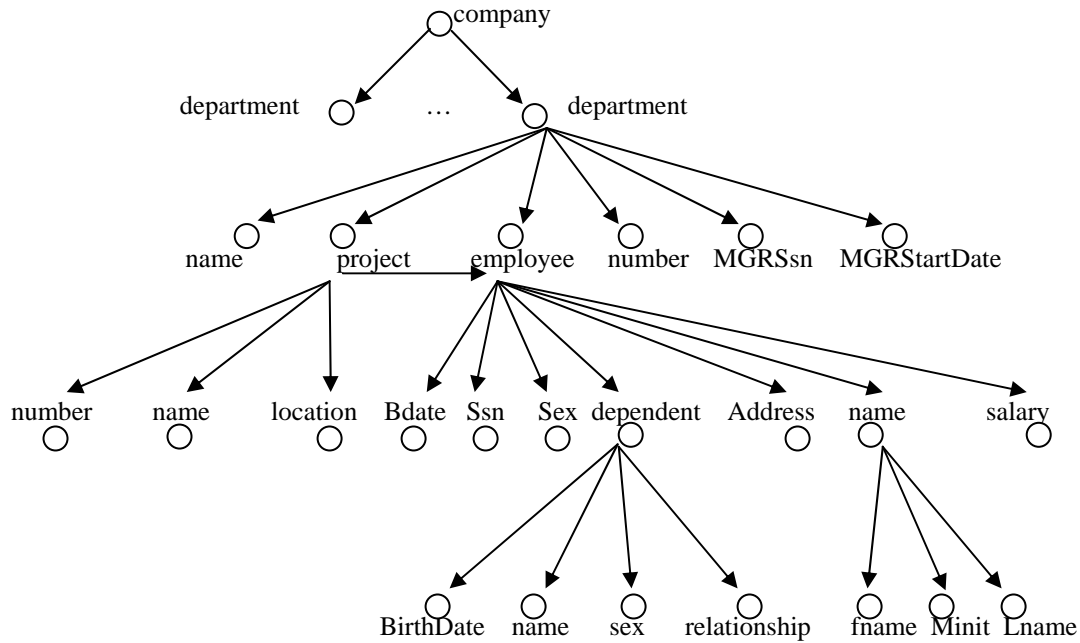
- (a) R is maximal (all nodes of G1 and G2 are involved in R)
- (b) For each pair  $(t,s)$  in R, and for each edge  $(t,a,t')$  in G1, there exists an edge  $(s,a,s')$  in G2 such that  $(t',s')$  is in R
- (c) [reverse of (b)]  
For each pair  $(t,s)$  in R, and for each edge  $(s,a,s')$  in G2, there exists an edge  $(t,a,t')$  in G1 such that  $(t',s')$  is in R

Therefore R is a desired relation that establishes bimilarity of G1 and G2.

**Problem 5**

This problem uses the QuiP system. Feel free to find and emulate other data/queries provided with QuiP when working on this problem. *Print out your files with data and queries and turn them in with your homework. Also turn in a copy of your output from running the queries on the data.*

- a) Translate the ER diagram for the Company DB (attached) to semistructured data model. (Note the Min/Max values on the edges)



b) Create an XML file “company.xml” with the above data model, using data values from attached sheet.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<company>
  <department id="5">
    <name>Research</name>
    <MGRSSN idref="333445555"/>
    <MGRStartDate>1988-05-22</MGRStartDate>
    <project id="1">
      <PName>ProductX</PName>
      <PLocation>Bellaire</PLocation>
    </project>
    <project id="2">
      <PName>ProductY</PName>
      <PLocation>Sugarland</PLocation>
    </project>
    <project id="3">
      <PName>ProductZ</PName>
      <PLocation>Houston</PLocation>
    </project>
    <employee id="123456789">
      <Name>
        <FName>John</FName>
        <Minit>B</Minit>
        <LName>Smith</LName>
      </Name>
      <BDate>1965-01-09</BDate>
      <Address>731 Fondren, Houston, TX</Address>
      <Sex>M</Sex>
      <Salary>30000</Salary>
      <SuperSSN idref="333445555"/>
      <project idref="1">
        <hours>32.5</hours>
      </project>
      <dependent>
        <dependent_name>Michael</dependent_name>
        <Sex>M</Sex>
        <BDate>1988-01-04</BDate>
      </dependent>
    </employee>
  </department>
</company>
  
```

```

        <Relationship>SON</Relationship>
    </dependent>
</dependent>
    <dependent_name>Alice</dependent_name>
    <Sex>F</Sex>
    <BDate>1988-12-30</BDate>
    <Relationship>DAUGHTER</Relationship>
</dependent>
</dependent>
    <dependent_name>Elizabeth</dependent_name>
    <Sex>F</Sex>
    <BDate>1967-05-05</BDate>
    <Relationship>SPOUSE</Relationship>
</dependent>
</employee>
<employee id="333445555">
    <Name>
        <FName>Fanklin</FName>
        <Minit>T</Minit>
        <LName>Wong</LName>
    </Name>
    <BDate>1955-12-08</BDate>
    <Address>638 Voss, Houston, TX</Address>
    <Sex>M</Sex>
    <Salary>40000</Salary>
    <SuperSSN idref="888665555"/>
    <project idref="2">
        <hours>10.0</hours>
    </project>
    <project idref="3">
        <hours>10.0</hours>
    </project>
    <project idref="10">
        <hours>10.0</hours>
    </project>
    <project idref="20">
        <hours>10.0</hours>
    </project>
    <dependent>
        <dependent_name>Alice</dependent_name>
        <Sex>F</Sex>
        <BDate>1986-04-05</BDate>
        <Relationship>DAUGHTER</Relationship>
    </dependent>
    <dependent>
        <dependent_name>Theodore</dependent_name>
        <Sex>M</Sex>
        <BDate>1983-10-25</BDate>
        <Relationship>SON</Relationship>
    </dependent>
    <dependent>
        <dependent_name>Joy</dependent_name>
        <Sex>F</Sex>
        <BDate>1958-05-03</BDate>
        <Relationship>SPOUSE</Relationship>
    </dependent>
</employee>
<employee id="666884444">
    <Name>
        <FName>Ramesh</FName>
        <Minit>K</Minit>
        <LName>Narayan</LName>
    </Name>
    <BDate>1962-09-15</BDate>
    <Address>975 Fire Oak, Humble, TX</Address>
    <Sex>M</Sex>

```

```

    <Salary>38000</Salary>
    <SuperSSN idref="33344555"/>
    <project idref="3">
      <hours>40.0</hours>
    </project>
  </employee>
<employee id="453453453">
  <Name>
    <FName>Joyce</FName>
    <Minit>A</Minit>
    <LName>English</LName>
  </Name>
  <BDate>1972-07-31</BDate>
  <Address>5631 Rice, Houston, TX</Address>
  <Sex>F</Sex>
  <Salary>25000</Salary>
  <SuperSSN idref="33344555"/>
  <project idref="1">
    <hours>20.0</hours>
  </project>
  <project idref="2">
    <hours>20.0</hours>
  </project>
</employee>
</department>
<department id="4">
  <name>Administration</name>
  <MGRSSN idref="987654321"/>
  <MGRStartDate>1995-01-01</MGRStartDate>
  <project id="10">
    <PName>Computerization</PName>
    <PLocation>Stafford</PLocation>
  </project>
  <project id="30">
    <PName>NewBenefits</PName>
    <PLocation>Stafford</PLocation>
  </project>
<employee id="999887777">
  <Name>
    <FName>Alicia</FName>
    <Minit>J</Minit>
    <LName>Zelaya</LName>
  </Name>
  <BDate>1968-07-19</BDate>
  <Address>3321 Castle, Spring, TX</Address>
  <Sex>F</Sex>
  <Salary>25000</Salary>
  <SuperSSN idref="987654321"/>
  <project idref="30">
    <hours>30.0</hours>
  </project>
  <project idref="10">
    <hours>10.0</hours>
  </project>
</employee>
<employee id="987654321">
  <Name>
    <FName>Jennifer</FName>
    <Minit>S</Minit>
    <LName>Wallace</LName>
  </Name>
  <BDate>1941-06-20</BDate>
  <Address>291 Berry, Bellaire, TX</Address>
  <Sex>F</Sex>
  <Salary>43000</Salary>
  <SuperSSN idref="888665555"/>

```

```

    <project idref="30">
      <hours>20.0</hours>
    </project>
  <project idref="20">
    <hours>15.0</hours>
  </project>
  <dependent>
    <dependent_name>Abner</dependent_name>
    <Sex>M</Sex>
    <BDate>1942-02-28</BDate>
    <Relationship>SPOUSE</Relationship>
  </dependent>
</employee>
<employee id="987987987">
  <Name>
    <FName>Ahmad</FName>
    <Minit>V</Minit>
    <LName>Jabber</LName>
  </Name>
  <BDate>1969-03-29</BDate>
  <Address>980 Dallas, Houston, TX</Address>
  <Sex>M</Sex>
  <Salary>25000</Salary>
  <SuperSSN idref="987654321"/>
  <project idref="10">
    <hours>35.0</hours>
  </project>
  <project idref="30">
    <hours>5.0</hours>
  </project>
  <dependent>
    <dependent_name>Abner</dependent_name>
    <Sex>M</Sex>
    <BDate>1942-02-28</BDate>
    <Relationship>SPOUSE</Relationship>
  </dependent>
</employee>
</department>
  <department id="1">
    <name>Headquarters</name>
    <MGRSSN idref="888665555"/>
    <MGRStartDate>1981-06-19</MGRStartDate>
    <project id="20">
      <PName>Reorganization</PName>
      <PLocation>Houston</PLocation>
    </project>
  <employee id="888665555">
    <Name>
      <FName>James</FName>
      <Minit>E</Minit>
      <LName>Borg</LName>
    </Name>
    <BDate>1937-11-10</BDate>
    <Address>450 Stone, Houston, TX</Address>
    <Sex>M</Sex>
    <Salary>55000</Salary>
    <project idref="20"></project>
  </employee>
</department>
</company>

```

c) Write the following XPATH queries and run them on “company.xml”:

1. Locations of all departments

Query:

```
distinct-values(collection("company")/company/department/project/PLocation)
```

Result:

```
<?xml version="1.0"?>
<quip:result xmlns:quip="http://namespaces.softwareag.com/tamino/quip/">
  <PLocation>Bellaire</PLocation>
  <PLocation>Sugarland</PLocation>
  <PLocation>Houston</PLocation>
  <PLocation>Stafford</PLocation>
</quip:result>
```

2. Last names of all employees with 2 or more female children.

Xpath query:

```
distinct-values(collection("company")/
company/department/employee[count(dependent[Sex="F"]>1)/Name/LName)
```

Result:

```
<?xml version="1.0"?>
<quip:result xmlns:quip="http://namespaces.softwareag.com/tamino/quip/">
  <LName>Smith</LName>
  <LName>Wong</LName>
</quip:result>
```

d) Write the following XQUERY query, and run it on “company.xml”:

For each department with 3 or more people, provide the name of the department manager and the department projects, and for each project, provide the names of the people working on it, and for each one, the number of his/her dependents.

Query:

```
<p5d>
{
  for $x in distinct-values(collection("company")/company/department[count(employee)>2])
  let $y := $x/employee[@id=$x/MGRSSN/@idref]
  return
    <department>
      {
        $x/name, <ManagerName>{$y/Name}</ManagerName>,
        for $p in $x/project
        return
          <project>
            {
              $p/PName,
              for $e in $x/employee[project/@idref = $p/@id]
              let $d := count($e/dependent)
              return
                <employee>
                  {
                    $e/Name,
                    <dependentNumber>{$d}</dependentNumber>
                  } </employee>
            } </project>
          } </department>
} </p5d>
```

Output:

```
<?xml version="1.0"?>
<quip:result xmlns:quip="http://namespaces.softwareag.com/tamino/quip/">
  <p5d>
    <department>
      <name>Research</name>
      <ManagerName>
        <Name>
          <FName>Fanklin</FName>
          <Minit>T</Minit>
          <LName>Wong</LName>
        </Name>
      </ManagerName>
      <project>
        <PName>ProductX</PName>
        <employee>
          <Name>
            <FName>John</FName>
            <Minit>B</Minit>
            <LName>Smith</LName>
          </Name>
          <dependentNumber>3</dependentNumber>
        </employee>
        <employee>
          <Name>
            <FName>Joyce</FName>
            <Minit>A</Minit>
            <LName>English</LName>
          </Name>
          <dependentNumber>0</dependentNumber>
        </employee>
      </project>
    </department>
    <department>
      <name>Research</name>
      <ManagerName>
        <Name>
          <FName>Fanklin</FName>
          <Minit>T</Minit>
          <LName>Wong</LName>
        </Name>
      </ManagerName>
      <project>
        <PName>ProductY</PName>
        <employee>
          <Name>
            <FName>Fanklin</FName>
            <Minit>T</Minit>
            <LName>Wong</LName>
          </Name>
          <dependentNumber>3</dependentNumber>
        </employee>
        <employee>
          <Name>
            <FName>Joyce</FName>
            <Minit>A</Minit>
            <LName>English</LName>
          </Name>
          <dependentNumber>0</dependentNumber>
        </employee>
      </project>
    </department>
    <department>
      <name>Research</name>
      <ManagerName>
        <Name>
          <FName>Fanklin</FName>
          <Minit>T</Minit>
          <LName>Wong</LName>
        </Name>
      </ManagerName>
      <project>
        <PName>ProductZ</PName>
        <employee>
          <Name>
            <FName>Fanklin</FName>
            <Minit>T</Minit>
            <LName>Wong</LName>
          </Name>
          <dependentNumber>3</dependentNumber>
        </employee>
        <employee>
          <Name>
            <FName>Ramesh</FName>
            <Minit>K</Minit>
          </Name>
          <dependentNumber>0</dependentNumber>
        </employee>
      </project>
    </department>
  </p5d>
</quip:result>
```

```

        <LName>Narayan</LName>
      </Name>
      <dependentNumber>0</dependentNumber>
    </employee>
  </project>
</department>
<department>
  <name>Administration</name>
  <ManagerName>
    <Name>
      <FName>Jennifer</FName>
      <Minit>S</Minit>
      <LName>Wallace</LName>
    </Name>
  </ManagerName>
  <project>
    <PName>Computerization</PName>
    <employee>
      <Name>
        <FName>Alicia</FName>
        <Minit>J</Minit>
        <LName>Zelaya</LName>
      </Name>
      <dependentNumber>0</dependentNumber>
    </employee>
    <employee>
      <Name>
        <FName>Ahmad</FName>
        <Minit>V</Minit>
        <LName>Jabber</LName>
      </Name>
      <dependentNumber>1</dependentNumber>
    </employee>
  </project>
</project>
  <PName>NewBenefits</PName>
  <employee>
    <Name>
      <FName>Alicia</FName>
      <Minit>J</Minit>
      <LName>Zelaya</LName>
    </Name>
    <dependentNumber>0</dependentNumber>
  </employee>
  <employee>
    <Name>
      <FName>Jennifer</FName>
      <Minit>S</Minit>
      <LName>Wallace</LName>
    </Name>
    <dependentNumber>1</dependentNumber>
  </employee>
  <employee>
    <Name>
      <FName>Ahmad</FName>
      <Minit>V</Minit>
      <LName>Jabber</LName>
    </Name>
    <dependentNumber>1</dependentNumber>
  </employee>
</project>
</department>
</p5d>
</quip:result>

```