CSE 532 LinkedOut Project 2

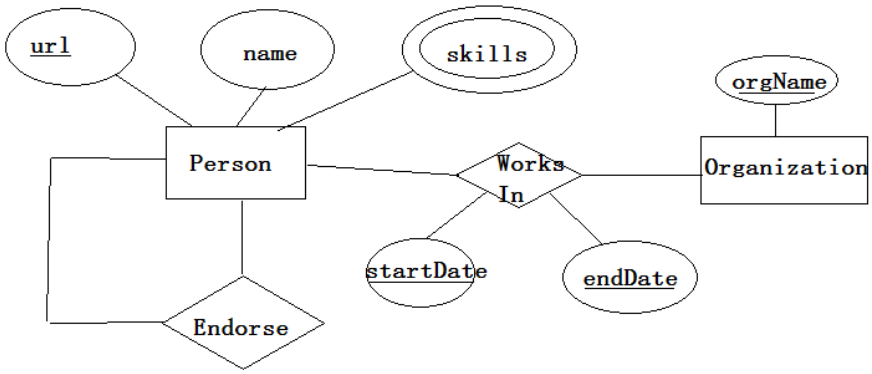
Team 06

Benjamin X. Lin (106186822) and Wenbin Lin (107851206)

WE Pledge Our Honor that All Parts of This Project were done by us alone and without collaboration with anybody else.

Table of Contents of this Project Report

1. Entity-Relationship Diagram



2. Description of Our Database Schema

** Due to the limitations of PostgreSQL 9.3, the following schema design is the best we generated. **

Based on the ER diagram above, we decided to create 3 tables: Person, Organization and Endorsement.

Person Table has 3 attributes (url, name, and skills). We set url to be the primary key for url uniquely identifies a person. We put the other two attributes “name” and “skills” into a Type called “person_type” in order best simulate object oriented design. Also, since “skills” is a set type, we set “skills” to be a text array in our design.

Organization Table is the table absorbed WorksIn and Organization entities in the ER diagram above, for the concern of data succinctness and query convenience. “url” is the foreign key in Organization Table, and the combination of “orgName”, “startDate” and “endDate” is the primary key in Organization Table.

Endorsement Table is the table representing one person endorsing another one for a set of skills. “fromURL” (the endorser) and “toURL” (the endorsee) are the foreign keys in Endorsement Table, and the combination of “fromURL” and “toURL” serves as the primary key for this table.

3. Description of Integrity Constraints

3.1 Referential Integrity

The “url” field in Organization Table is the foreign key referring to the “url” field, the primary key in Person Table. Doing this is because one same person can work in different organizations in different periods.

The “fromURL” and “toURL” are the foreign keys in Endorsement Table referring to the “url” field, the primary key in Person Table. Doing this is because one person can endorse multiple people (including himself) and himself can also be endorsed by multiple people (including himself) for various skills.

3.2 CHECK-Constraints

In Organization Table, we need to check “orgName”, “startDate” and “endDate” are not null. The reasons for this design are as follows:

(1) The organization one person works in must have a name.
(2) There must be a date on which the person starts to work
(3) The endDate should not be null for there would ultimately be a date a person leaves his job; if not, we can use today’s date to be endDate. In all, this field is not null primarily for the data manipulation reasons.

4. All SQL CREATE TABLE/VIEW/TYPE Commands

```
CREATE TYPE person_type AS (
        name   varchar(30),
        skills   text[]
);

CREATE TABLE Person
(
        url      varchar primary key,   /* Primary key */
        personInfo      person_type
);
/* create both TYPE person_type and TABLE Person
        in order to use the primary key feature */

CREATE TABLE Organization (
        url    varchar    references Person(url),        /* foreign key constraint */
        orgName        varchar(30) NOT NULL,
        startDate      date    NOT NULL,
        endDate        date    NOT NULL
);

CREATE TABLE Endorsement (
        fromURL  varchar     references Person(url), /* foreign key constraint */
        toURL      varchar     references Person(url), /* foreign key constraint */
        skills       text[]
);

CREATE VIEW single_endorse AS
        SELECT
                P1.url as from, P2.url as to, (P1.personInfo).name as from_name,
        (P2.personInfo).name as to_name, UNNEST(E.skills) as skill
        FROM   Endorsement E, Person P1, Person P2
        WHERE   E.fromURL = P1.url
                AND   E.toURL = P2.url;

CREATE VIEW single_skill AS
        SELECT  P.url AS url, UNNEST((P.personInfo).skills) as skill
        FROM     Person P;

CREATE VIEW endorsed_eachskill AS
        SELECT    DISTINCT SS.url AS url
        FROM    single_skill SS, single_endorse SE
        WHERE  SS.url<>SE.from
                AND SS.url = SE.to
```

```
        AND SS.skill = SE.skill;



CREATE VIEW   more_skill AS
        SELECT      P1.url AS U1, P2.url AS U2
        FROM        Person P1, Person P2
        WHERE       P1.url <> P2.url        /* P1 and P2 are different persons */
                AND ( (P1.personInfo).skills IS NOT NULL AND
(P2.personInfo).skills IS NULL )
                                        /* P1 has any skills while P2 has nothing */
                OR  (
                        (P1.personInfo).skills @> (P2.personInfo).skills
                                        /* P1 has every skill that P2 has */
                        AND array_length((P1.personInfo).skills, 1) >
array_length((P2.personInfo).skills, 1)
                                        /* P1 has a skill that that P2 does Not have */
        );


CREATE VIEW   DirectEndorse(fromURL, toURL)   AS
        SELECT   fromURL, toURL
        FROM     Endorsement
        EXCEPT
        SELECT  E1.fromURL, E1.toURL
        FROM    Endorsement E1, Endorsement E2
        WHERE  E1.fromURL = E2.toURL
                AND E2.fromURL = E1.toURL;


CREATE RECURSIVE VIEW IndirectEndorse(fromURL, toURL) AS
        SELECT   *
        FROM     DirectEndorse
        UNION
        SELECT   DE.fromURL, I.toURL
        FROM     DirectEndorse DE, IndirectEndorse I
        WHERE   DE.toURL = I.fromURL;

CREATE VIEW DirectEndorse_MoreSkill(fromURL, toURL) AS
        SELECT  fromURL, toURL
        FROM    Endorsement, more_skill MS
        WHERE  fromURL = MS.U1 AND toURL = MS.U2
        EXCEPT
        SELECT  E1.fromURL, E1.toURL
        FROM    Endorsement E1, Endorsement E2
        WHERE  E1.fromURL = E2.toURL
                AND E2.fromURL = E1.toURL;
```

```
CREATE RECURSIVE VIEW IndirectEndorse_MoreSkill(fromURL, toURL) AS
        SELECT   *
        FROM     DirectEndorse_MoreSkill
        UNION
        SELECT   DEMS.fromURL, IM.toURL
        FROM     DirectEndorse_MoreSkill DEMS, IndirectEndorse_MoreSkill IM
        WHERE    DEMS.toURL = IM.fromURL;
```

5. Seven Query Statements for the 7 Questions

Q1: Endorsement pairs sharing a common organization on 09/18/2013:

```
SELECT  (P1.personInfo).name AS from_name, (P2.personInfo).name AS to_name
FROM     Endorsement E, Organization O1, Organization O2, Person P1, Person P2
WHERE         E.fromURL = O1.url
          AND   E.toURL = O2.url
          AND   E.fromURL <> E.toURL
          AND   O1.orgName = O2.orgName
          AND   O1.startDate < '2013-09-18'
          AND   O1.endDate > '2013-09-18'
          AND   O2.startDate < '2013-09-18'
          AND   O2.endDate > '2013-09-18'
          AND   E.fromURL = P1.url
          AND   E.toURL = P2.url
          ORDER BY  from_name, to_name
```

Q2: Highly qualified endorsements

```
/* Note: The following SQL code plus the Java code in
src/DB/LinkedOutDbHelpers.java form the final correct answers */

SELECT   SE.from_name, SE.to_name, SE.skill
FROM      single_endorse SE, single_skill SS
WHERE   SS.skill = SE.skill AND SE.from = SS.url AND SE.from <> SE.to AND
          /* person endorses the other for this skill is endorsed by a third person */
          0 < (SELECT count(*) FROM single_endorse SEE
                    WHERE SE.from <> SEE.from AND SE.to <> SEE.from AND
SE.skill = SEE.skill AND SEE.to = SE.from) AND
          /* person endorses the other for this skill is endorsed by a third person */
          0 < (SELECT count(*) FROM single_endorse SEE
        WHERE SE.from <> SEE.from AND SE.to <> SEE.from AND SE.skill =
SEE.skill AND SEE.to = SE.to)
```

Q3: Users endorsed for unclaimed skills

```
/* Users who do not have a certain skills
        but were endorsed by at least two other users for that skill */
SELECT      DISTINCT (P.personInfo).name AS name
FROM        Person P, single_endorse SE1
WHERE       P.url = SE1.to
        AND  2 <= (SELECT count(SE2.to)
                    FROM   single_endorse SE2
                   WHERE   SE1.to = SE2.to
                      AND   SE1.skill = SE2.skill
                      AND   SE2.from <> SE2.to
                            /* Users that were endorsed by at least two other users
for that skill */
                     AND SE1.skill NOT IN (
                            SELECT  SS.skill
                            FROM      single_skill SS
                            WHERE   SE1.to = SS.url
                     )              /* Users do not have that certain skill */
                )
        ORDER BY name
```

Q4: Strictly more skilled users

```
SELECT      (P1.personInfo).name AS from_name,
            (P2.personInfo).name AS to_name
FROM        Person P1, Person P2
WHERE       P1.url <> P2.url                /* P1 and P2 are different persons */
        AND  ( (P1.personInfo).skills IS NOT NULL AND (P2.personInfo).skills IS
NULL )
                                            /* P1 has any skills while P2 has nothing */
        OR  (
            (P1.personInfo).skills @> (P2.personInfo).skills
                                            /* P1 has every skill that P2 has */
            AND
                array_length((P1.personInfo).skills, 1) >
            array_length((P2.personInfo).skills, 1)
                                            /* P1 has a skill that that P2 does Not have */
        )
        ORDER BY from_name, to_name
```

Q5: Strictly more certified users

```
SELECT   DISTINCT   (P1.personInfo).name AS from_name,
                        (P2.personInfo).name AS to_name
FROM       Person P1, Person P2, endorsed_eachskill EES, more_skill MS
WHERE          P1.url = EES.url          /* P1 was endorsed for each of his skills */
         AND   P1.url = MS.U1
         AND   P2.url = MS.U2          /* P1 is more skilled than P2 */
         ORDER BY   from_name, to_name
```

Q6: Indirect endorsements

```
SELECT   (P1.personInfo).name AS from_name, (P2.personInfo).name AS to_name
FROM      Person P1, Person P2, IndirectEndorse IE
WHERE   P1.url = IE.fromURL
         AND    P2.url = IE.toURL
ORDER BY    from_name, to_name
```

Q7: Skill-descending indirect endorsements

```
SELECT   (P1.personInfo).name AS from_name, (P2.personInfo).name AS to_name
FROM      Person P1, Person P2, IndirectEndorse_MoreSkill IEMS
WHERE   P1.url = IEMS.fromURL
         AND   P2.url = IEMS.toURL
ORDER BY   from_name, to_name
```

6. Brief User Guide

Prepare Stage:

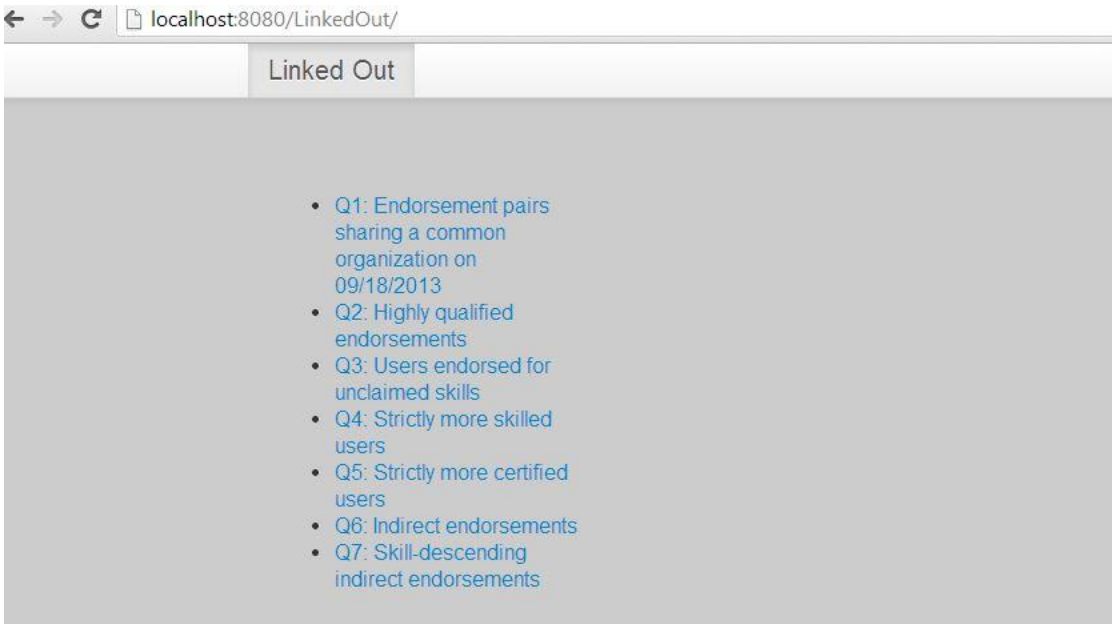
1. Install PostgreSQL 9.3 on your local machine
2. Install Eclipse EE (ie. eclipse-jee-juno-SR1-win32-x86_64) on your machine
3. Install Apache Tomcat 7.0 on your local machine
4. Download postgresql-9.3-1100.jdbc4.jar

Assembly Stage:

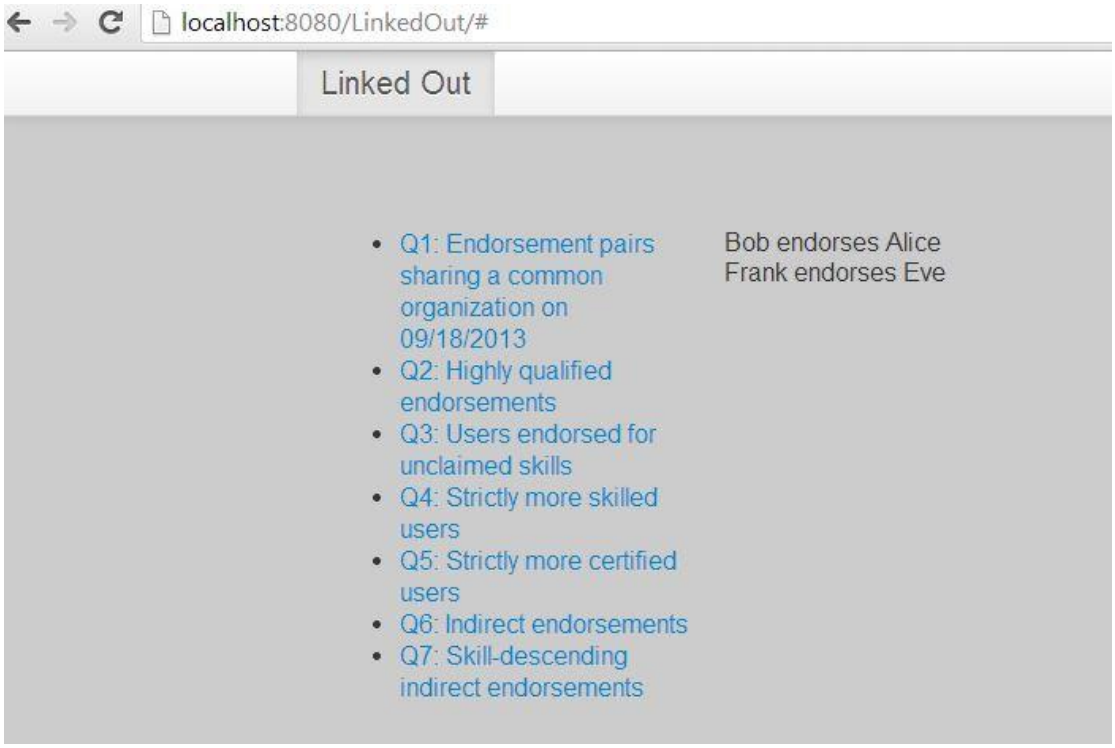
1. Open Eclipse, and import the LinkedOut project (the "linkedout" folder) from the cse532-team06 repository
2. Embed Apache Tomcat v7.0 to LinkedOut project
3. Embed postgresql-9.3-1100.jdbc4.jar into Web App Libraries
4. Run this project on server, choosing Tomcat v7.0 as the local server for running process
5. You may copy the URL from the Eclipse internal browser into the address bar of a browser you would prefer to use in order to run this application

Usage:

This is the GUI of our application:



As you can see, there are 7 bullet points indicating all the 7 questions we want to query. Click each of them, the answer will be displayed on the right side of those questions, as shown below:



7. Contribution

7.1 Design Stage:

Benjamin: Discuss with TA Jon and make a final design of the Database Schema
Wenbin: Verify the Schema Design

7.2 Implementation:
7.2.1. Front-End:
Wenbin: Using AJAX and JavaScript to make the single dynamic page to present the question and queried answers; write the PunchServelet class
Benjamin: Being knowledge transferred about simple JavaScript and AJAX; write the HTML code for those 7 questions.

7.2.2 Back-End:
Benjamin: Write Query 1, 3, 4, 5, 7 in SQL and put them in LinkedOutDBDriver class
Wenbin: Write the frame of LinkedOutDBDriver class; write Query 2 and its help class LinkedOutDBHelper; write Query 6 in SQL.

7.3 Documentation:
Benjamin: Write this project report.
Wenbin: Proofread this project report.