CSE 532: Project 2
WOCO via the Object-Oriented Extensions of SQL
Fall 2017

Author(s): **Siddharth Jain** (111425239)
Shubham Kumar Jain (111482623)

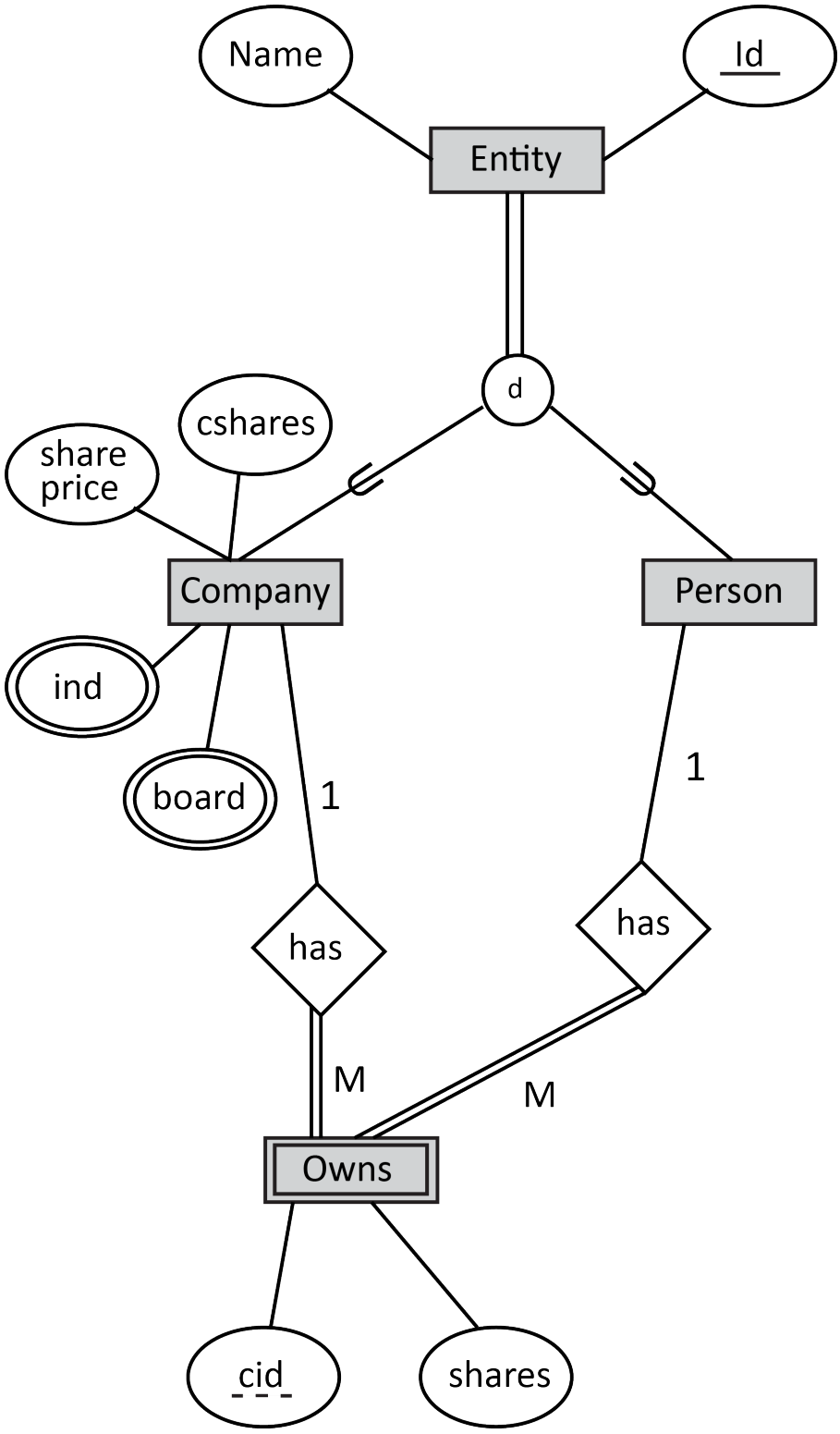
We, pledge our honor that all parts of this project were done by us alone and without collaboration with anybody else.

STONY BROOK UNIVERSITY

(I)

TABLE OF CONTENTS

1. ER Diagram


Name
Id
Entity
d
cshares
share price
Company
Person
ind
board
1
1
has
has
M
M
Owns
cid
shares

2. Description of the Database Scheme

Designing and implementing the WOCO database using the object-relational extensions of POSTGRESQL. Used Java/JSP/Servlets to build an application front end of the database. Database connectivity is done by using Java/JDBC.

DATABASE SCHEME

WOCO database is an object oriented database, with two basic types:-

1) Ownstype having the company id (CID) and no of shares of an owner (shares)
2) Entitytype having the id and name of the owner (person or company).

There are three basic tables:-

1) **Entity** which is of type – Entitytype , which contains id(primary key) and name.

2) **Person** which inherits id and name from table entity. It contains id (primary key) and an object **owns** (an array of ownstype).

3) **Company** which inherits id and name from table entity. It contains id (primary key), shareprice of the company, ind (array of the industry which it belongs to), board (array of the board members of the particular company), an object **owns** (array of ownstype) and cshares (no of shares of that company).

The tables are normalized and are in 4NF for quality database schema. The database has two one to many relationships (Company can have many owns and a person can have many owns). There are two arrays (industry and board)which are in accordance to the object oriented model scheme.

Database is designed in such a manner that the structure is minimalistic. The model is structured in a manner that it allows the user to export the quality of its data with certain level of flexibility.

3. INTEGRITY CONSTRAINTS

In designing the database schema various integrity constraints are considered as stated below-

- **Primary Key** – (cid in ownstype, id in entitytype, id in entity, id in person, id in company
- **Not Null** – name in entity table, shareprice in table company, cshares in table company
- **Check** - shareprice>0 in table company, cshares>0 in table company

INDEXING OF THE DATABASE

```
CREATE INDEX idx_person
ON person(id,name,owns);

CREATE INDEX idx_company
ON company(id,name,shareprice,ind,board,owns,cshares);
```

4. All SQL CREATE TABLE/VIEW/TYPE Queries (PostgreSQL queries for creating the Schema of the required database)

```
CREATE TYPE ownstype AS
(
        cid TEXT,
        shares DOUBLE PRECISION
);

CREATE TYPE entitytype AS
(
        id TEXT,
        name TEXT
```

```
);

CREATE TABLE entity OF entitytype
(
        id PRIMARY KEY,
        name NOT NULL
);

CREATE TABLE person
(
        PRIMARY KEY (id),
        owns ownstype[]
) INHERITS (entity);

CREATE TABLE company
(
        PRIMARY KEY (id),
        shareprice Integer NOT NULL CHECK(shareprice>0),
        ind TEXT[],
        board TEXT[],
        owns ownstype[],
        cshares DOUBLE PRECISION NOT NULL CHECK (cshares>0)
) INHERITS (entity);
```

5. PostgreSQL Statements (PostgreSQL queries for the data insertion in the database)

Inserting into the person database

```
INSERT INTO person VALUES('p1','Bill Doe',
```

```
ARRAY[('c5',30000),('c8',100000)]::ownstype[]);

INSERT INTO person
VALUES('p2','Bill Seth',
ARRAY[('c7',40000),('c4',20000)]::ownstype[]);

INSERT INTO person
VALUES('p3','John Smyth',
ARRAY[('c1',20000),('c2',20000),('c5',800000)]::ownstype[]);

INSERT INTO person
VALUES('p4','Anne Smyle',
ARRAY[('c2',30000),('c5',40000),('c3',500000)]::ownstype[]);

INSERT INTO person
VALUES('p5','Steve Lamp',
ARRAY[('c8',90000),('c1',50000),('c6',50000),('c2',70000)]::ownstype[]);

INSERT INTO person
VALUES('p6','May Serge',
ARRAY[('c8',-10000),('c9',-40000),('c3',500000),('c2',40000)]::ownstype[]);

INSERT INTO person
VALUES('p7','Bill Public',
ARRAY[('c7',80000),('c4',30000),('c1',30000),('c5',300000),('c2',-9000)]::ownstype[]);

INSERT INTO person
VALUES('p8','Muck Lain',
ARRAY[('c2',60000),('c6',-40000),('c9',-80000),('c8',30000)]::ownstype[]);
```

Inserting into the company database.

```
INSERT INTO company  VALUES ('c1','QUE',30,ARRAY[('Software'),('Accounting')]::TEXT[],
ARRAY(SELECT id FROM person WHERE id ='p3' or id = 'p1' or id =
'p4'),ARRAY[('c2',10000),('c4',20000),('c8',30000)]::ownstype[],150000);

INSERT INTO company VALUES ('c2','RHC',20,ARRAY[('Accounting')]::TEXT[],ARRAY(SELECT id
FROM person WHERE id ='p2' or id = 'p1' or id = 'p5'),ARRAY[]::ownstype[],250000);

INSERT INTO company VALUES
('c3','Alf',700,ARRAY[('Software'),('Automotive')]::TEXT[],ARRAY(SELECT id FROM person
WHERE id ='p6' or id = 'p1' or id = 'p7'),ARRAY[('c9',-
100000),('c4',400000),('c8',100000)]::ownstype[],10000000);

INSERT INTO company VALUES
('c4','Elgog',400,ARRAY[('Software'),('Search')]::TEXT[],ARRAY(SELECT id FROM person WHERE
id ='p6' or id = 'p7' or id = 'p5'),ARRAY[('c6',5000)]::ownstype[],1000000);

INSERT INTO company VALUES
('c5','Tfos',300,ARRAY[('Software'),('Hardware')]::TEXT[],ARRAY(SELECT id FROM person
WHERE id ='p2' or id = 'p5' or id =
'p4'),ARRAY[('c1',200000),('c6',30000),('c7',50000)]::ownstype[],10000000);

INSERT INTO company VALUES ('c6','Ohay',50,ARRAY[('Search')]::TEXT[],

ARRAY(SELECT id FROM person WHERE id ='p2' or id = 'p8' or id = 'p4'),

ARRAY[]::ownstype[],180000);

INSERT INTO company VALUES ('c7','Gnow',300,ARRAY[('Search')]::TEXT[],

ARRAY(SELECT id FROM person WHERE id ='p2' or id = 'p3' or id = 'p4'),

ARRAY[]::ownstype[],150000);

INSERT INTO company VALUES ('c8','Elpa',300,ARRAY[('Software'),('Hardware')]::TEXT[],

ARRAY(SELECT id FROM person WHERE id ='p2' or id = 'p3' or id = 'p8'),

ARRAY[('c5',20000),('c4',30000)]::ownstype[],9000000);
```

```
INSERT INTO company VALUES ('c9','Ydex',100,ARRAY[('Software'),('Search')]::TEXT[],

ARRAY(SELECT id FROM person WHERE id ='p6' or id = 'p3' or id = 'p8'),

ARRAY[]::ownstype[],

5000000);
```

Postgresql query for Question 1

```
SELECT C.name
FROM company C, unnest(C.board) AS x, person P, unnest(P.owns) AS y
WHERE x = P.id AND C.id=y.cid AND y.shares > 0
ORDER by C.name;
```

Postgresql query for Question 2

```
Select Per.name, SUM(x.shares*C.shareprice)
From person Per, company C, unnest(Per.owns) AS x
WHERE x.cid = C.id
        AND x.shares>0
GROUP BY Per.name
ORDER BY Per.name;
```

Postgresql query for Question 3

```
SELECT C.name, P.name
FROM company C, Person P, unnest(C.board) AS x, unnest(P.owns) AS y
WHERE x=P.id
        AND y.cid=C.id
        AND y.shares>0
        AND y.shares=(
                SELECT max(z.shares)
                FROM person P, unnest(P.owns) AS z
                WHERE x=P.id
                        AND z.cid=C.id
                        AND z.shares>0 )
ORDER BY C.name;
```

Postgresql query for Question 4

```
SELECT c1.name,c2.name
FROM company c1, company c2, unnest(c1.ind) AS i1, unnest(c2.ind) AS i2
WHERE c1.id != c2.id
        AND i1=i2
        AND NOT EXISTS(
                SELECT *
                        FROM person P2, unnest(c2.board) AS x, unnest(P2.owns) AS y
                        WHERE x = P2.id
                                AND y.shares >0
                                AND NOT EXISTS (
                                SELECT *
                                FROM person P1, unnest(c1.board) AS w, unnest(P1.owns) AS z
                                WHERE w = P1.id
                                        AND z.cid=y.cid
                                        AND z.shares >= y.shares ) )
GROUP BY c1.name,c2.name;
```

Postgresql query for Question 5

```
CREATE RECURSIVE VIEW direct(oid,cid,Perc) AS
(SELECT C2.id, C1.id, cast(cast(y.shares as Float)/cast(C1.cshares as Float) as numeric(10,8))
        FROM company C1, company C2, unnest(C2.owns) AS y
        WHERE y.cid = C1.id AND y.shares>0) UNION
        (SELECT P.id, C1.id,
        cast(cast(x.shares as Float)/cast(C1.cshares as Float) as numeric(10,8))
                FROM company C1, person P, unnest(P.owns) AS x
                WHERE x.cid = C1.id
                        AND x.shares>0);


CREATE RECURSIVE VIEW indirect(Entity,Cid,Per) AS
        (SELECT * FROM direct D)
        UNION
        SELECT D.oid, I.Cid, cast((D.Perc*I.Per) as numeric(10,8))
                FROM direct D, indirect I
                WHERE D.cid = I.Entity ;


SELECT P.name, C.name,Cast(SUM(I.Per*100) as numeric(10,4)) AS Percentage
FROM person P, company C, indirect I
WHERE P.id = I.Entity AND C.id = I.Cid
        AND I.Per*100>10
GROUP BY P.name, C.name
ORDER BY P.name;
```

6. User Guide

1. **Setup the WOCO database**
   Database Name – WOCO
   Database username – postgres
   Database password – password
   (Note:- if you change the username and password for the database, the changes need to be done in the Provider.java file as well).

2. **Copy the sql, java and jsp files to the workspace.**

3. **Setup the Frontend**
   a). Run the index.jsp to run the project.

7. Division of Work

We have collaborated on each part of the project and the work is on top of each other. The division of work is done in the vertical manner and is as follows.

Siddharth Jain	Shubham Kumar Jain
Database Schema design and implementation.	Data insertion and verification
Implementation of Postgres queries for the project.	Implementation of Postgres queries for the project
Connection establishment between database and servlet using JAVA.	Connection between all the Postgres queries and the JDBC application.
Index page creation using servlets in JSP.	Reformatting the queries to display the required result.