Thanks Corporation Database Project
CMSI 486 Enterprise Project — Fall 2014

Edward Bramanti

November 6, 2014

Contents

I Title Page **1**

II Table of Contents **2**

III Description of the Enterprise **3**

IV Definition of Environment **5**
IV.1 Input and Report Forms 5
IV.2 Assumptions 5
IV.3 User-Oriented Data Dictionary 5
IV.4 Cross-Reference Table 5

V Enterprise Database Design **6**
V.1 Logical Model of the Enterprise 6
V.1.1 List of Entities and Attributes 6
V.1.2 List of Relationships and Attributes 8
V.1.3 Entity-Relationship Diagram of the Enterprise 9
V.2 Conceptual model of the enterprise 10
V.3 Table dictionary 11
V.4 Attribute dictionary 11

VI Database and Query Definition **12**
VI.1 Database Definition 12
VI.2 Database Queries 19
VI.3 Design Tradeoffs and Limitations 22

Chapter III

Description of the Enterprise

Thanks is an effective, entirely digital, multi-purpose employee recognition system. The database will consist of many different companies who will pay for a service that makes recognizing employees simple and meaningful. This will allow the Thanks database to be used in a way that is unique to each company using the service.

The enterprise in question will make it much easier for employees to recognize one another across their company. As companies grow, it becomes difficult to maintain an atmosphere of employee worth. This growing size represents a problem as individual employees can feel like their work goes unnoticed in their company. Thanks provides a remedy for that by providing a digital way to send recognition to any employee quickly.

In an organization, there will be one user type with some special attributes. For example, an executive will be a special user type and will be denoted with special attributes. When opening the thanks form, it will be necessary to present a list of all employees in the company. A list of employees would appear with these attributes: name, position and department. This would provide employees with a clear snapshot on everyones role in the company if they saw a fellow employee do something amazing and was not sure what position they occupied in the company. Users will be able to send other users the primary data type known as Thanks. For each thank, we have a user that gives the Thanks and a user that receives the Thanks. This demonstrates the personal aspect of recognition, as an employee recognizes a specific employee directly.

A Thanks also contains an area for an employee to write a message so they can explain what they are thanking the employee for. This allows for personalization of the Thanks so that employees can be detailed on the outstanding work their coworkers are doing. Finally, companies will be able to define a custom attribute for the last part of the Thanks, which will be represented as values the company wants to promote. The advantage of this value is that companies, depending on their mission statements and core beliefs, will be able to tailor this field to encourage specific attributes that represent the company within employees.

Thanks represents a social network within a company than it is a private thanking system. Messages will display on a live feed all of the Thanks being given around the company. Employees will be able to easily see the interaction and encouragement being spread amongst their acquaintances. The potential of Thanks is enormous, which is why a specific structure around a public space of thanks combined with personalized instances and explanations of employee worth are necessary in this database design.

Here are a set of questions employees, department heads or executives may pose when retrieving data:

1. List the names of all employees in the company.
2. Which department has received the most Thanks in the company?
3. Return a list of all employees who have nicknames.
4. List all company values for the company.
5. Which department head has received the most Thanks in the company?
6. Which one of an employee's thanks they gave received the most likes?
7. Who has never received a Thanks within a company?
8. Who is the newest employee to have joined the company?
9. What Thanks were posted in the database on October 20, 2014?
10. List the executives in the company.

Chapter IV

Definition of Environment

IV.1 Input and Report Forms

IV.2 Assumptions

IV.3 User-Oriented Data Dictionary

IV.4 Cross-Reference Table

Chapter V

Enterprise Database Design

V.1 Logical Model of the Enterprise

V.1.1 List of Entities and Attributes

- Employee
  - eid: Employee ID
  - name: Employee Name
  - job_title: Employee Job Title
  - photo: Employee Photo
  - nickname: Employee Nickname
  - started: Date the Employee Started
- Department Head - inherits Employee
  - headid: Department Head ID
- Executive - inherits Employee
  - execid: Executive ID
- Department
  - did: Department ID
  - depTitle: Department Title
  - depDescription: Department Description
- Company
  - cid: Company ID
  - cTitle: Company Title
  - founded_date: Date Company was Founded
- Thanks
  - tid: Thanks ID

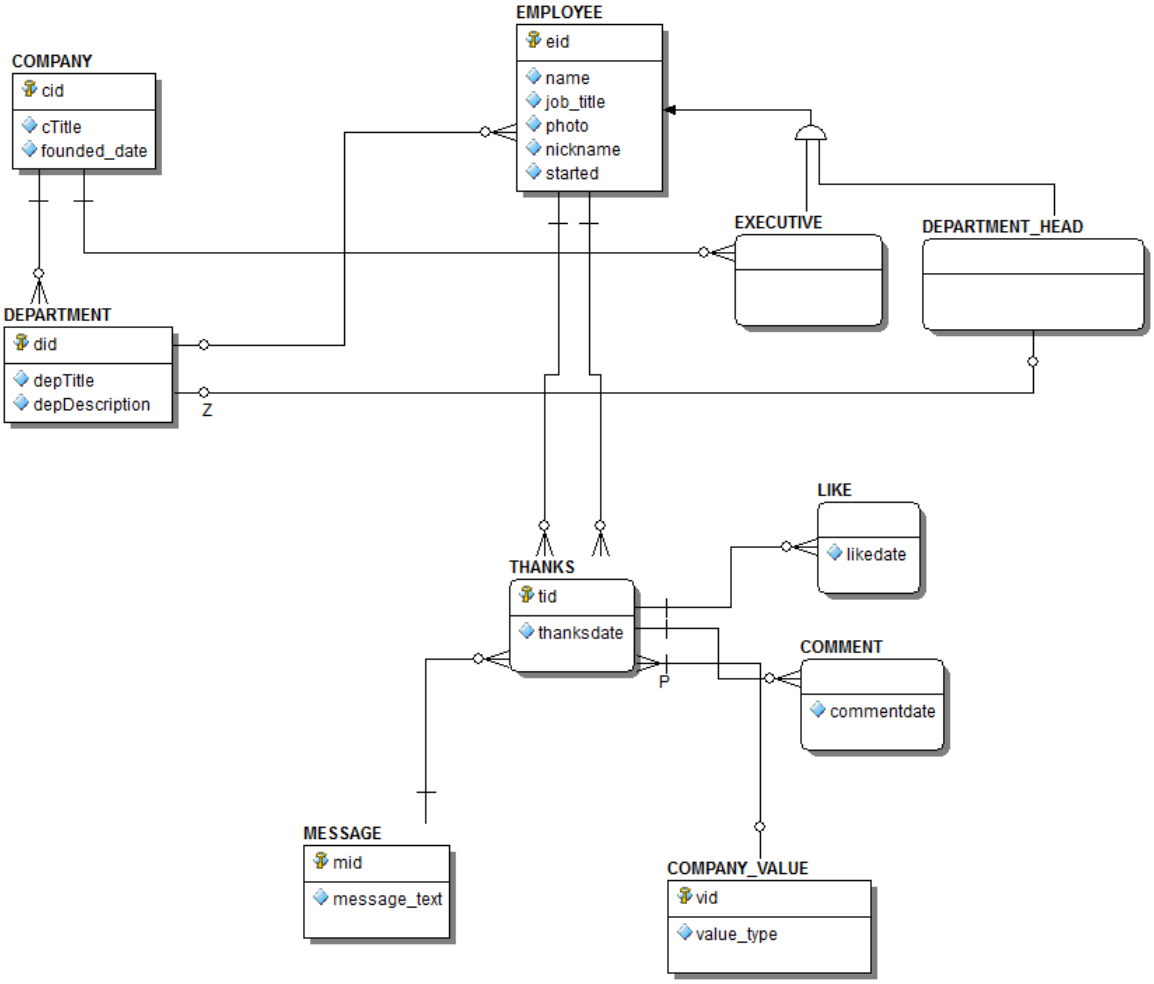
  - thanksdate: Date Thanks was Given

- Like
  - likedate: Date Like was Given
- Comment
  - commentdate: Date Comment was Given
- Message
  - mid: Message ID
  - message_text: Text of Message
- Company Value
  - vid: Company Value ID
  - value_type: type of company value

V.1.2 List of Relationships and Attributes

- Relationship

V.1.3 Entity-Relationship Diagram of the Enterprise



V.2 Conceptual model of the enterprise

employee(eid, name, job_title, photo, nickname, started)

CK - eid, name, photo
departmenthead(headid)

executive(execid)

department(did, depTitle, depDescription)

CK - did, depTitle
company(cid, cTitle, founded_date)

CK - cid, cTitle
thanks(tid, thanksdate)

CK - tid
like(tid, likedate)

CK - tid FK - tid REFERENCES thanks.tid
comment(tid, commentdate)

CK - tid FK - tid REFERENCES thanks.tid
message(mid, message_text)

CK - mid
company_value(vid, value_type)

CK - vid

V.3 Table dictionary

V.4 Attribute dictionary

Chapter VI

Database and Query Definition

VI.1 Database Definition

```
--
-- ER/Studio Data Architect 9.6 SQL Code Generation
-- Project :      ThanksCorp.DM1
--
-- Date Created : Tuesday, November 04, 2014 21:35:39
-- Target DBMS : MySQL 5.x
--

--
-- TABLE: COMMENT
--

CREATE TABLE COMMENT(
    commentid      VARCHAR(30)    NOT NULL,
    mid            VARCHAR(30)    NOT NULL,
    commentdate    DATETIME       NOT NULL,
    PRIMARY KEY (commentid, mid)
)ENGINE=INNODB
;



--
-- TABLE: COMPANY
--

CREATE TABLE COMPANY(
    cid             VARCHAR(30)    NOT NULL,
    cTitle          VARCHAR(30)    NOT NULL,
    founded_date    DATETIME       NOT NULL,
    PRIMARY KEY (cid)
)ENGINE=INNODB
```

```
;



--
-- TABLE: COMPANY_VALUE
--

CREATE TABLE COMPANY_VALUE(
    vid           VARCHAR(30)    NOT NULL,
    value_type    VARCHAR(30)    NOT NULL,
    PRIMARY KEY (vid)
)ENGINE=INNODB
;



--
-- TABLE: DEPARTMENT
--

CREATE TABLE DEPARTMENT(
    did               VARCHAR(30)     NOT NULL,
    depTitle          VARCHAR(30),
    depDescription    VARCHAR(255),
    headid            VARCHAR(30),
    cid               VARCHAR(30)     NOT NULL,
    PRIMARY KEY (did)
)ENGINE=INNODB
;



--
-- TABLE: DEPARTMENT_HEAD
--

CREATE TABLE DEPARTMENT_HEAD(
    headid    VARCHAR(30)    NOT NULL,
    PRIMARY KEY (headid)
)ENGINE=INNODB
;



--
-- TABLE: EMPLOYEE
--
```

```
CREATE TABLE EMPLOYEE(
    eid          VARCHAR(30)    NOT NULL,
    name         VARCHAR(30)    NOT NULL,
    job_title    VARCHAR(30)    NOT NULL,
    photo        BLOB,
    nickname     VARCHAR(30),
    started      DATETIME       NOT NULL,
    did          VARCHAR(30),
    PRIMARY KEY (eid)
)ENGINE=INNODB
;



--
-- TABLE: EXECUTIVE
--

CREATE TABLE EXECUTIVE(
    execid    VARCHAR(30)    NOT NULL,
    cid       VARCHAR(30)    NOT NULL,
    PRIMARY KEY (execid)
)ENGINE=INNODB
;



--
-- TABLE: LIKE
--

CREATE TABLE LIKE(
    likeid      VARCHAR(30)    NOT NULL,
    mid         VARCHAR(30)    NOT NULL,
    likedate    DATETIME       NOT NULL,
    PRIMARY KEY (likeid, mid)
)ENGINE=INNODB
;



--
-- TABLE: MESSAGE
--

CREATE TABLE MESSAGE(
    mid             VARCHAR(30)     NOT NULL,
```

```
    message_text    VARCHAR(255),
    PRIMARY KEY (mid)
)ENGINE=INNODB
;



--
-- TABLE: THANKS
--

CREATE TABLE THANKS(
    tid           VARCHAR(30)    NOT NULL,
    mid           VARCHAR(30)    NOT NULL,
    to            VARCHAR(30)    NOT NULL,
    from          VARCHAR(30)    NOT NULL,
    vid           VARCHAR(30),
    thanksdate    DATETIME       NOT NULL,
    PRIMARY KEY (tid, mid)
)ENGINE=INNODB
;



--
-- INDEX: Ref720
--

CREATE INDEX Ref720 ON COMMENT(commentid, mid)
;
--
-- INDEX: Ref1529
--

CREATE INDEX Ref1529 ON DEPARTMENT(headid)
;
--
-- INDEX: Ref332
--

CREATE INDEX Ref332 ON DEPARTMENT(cid)
;
--
-- INDEX: Ref148
--

CREATE INDEX Ref148 ON DEPARTMENT_HEAD(headid)
;
```

```
--
-- INDEX: Ref430
--

CREATE INDEX Ref430 ON EMPLOYEE(did)
;
--
-- INDEX: Ref147
--

CREATE INDEX Ref147 ON EXECUTIVE(execid)
;
--
-- INDEX: Ref328
--

CREATE INDEX Ref328 ON EXECUTIVE(cid)
;
--
-- INDEX: Ref719
--

CREATE INDEX Ref719 ON LIKE(likeid, mid)
;
--
-- INDEX: Ref617
--

CREATE INDEX Ref617 ON THANKS(vid)
;
--
-- INDEX: Ref1422
--

CREATE INDEX Ref1422 ON THANKS(to)
;
--
-- INDEX: Ref1424
--

CREATE INDEX Ref1424 ON THANKS(from)
;
--
-- INDEX: Ref531
--

CREATE INDEX Ref531 ON THANKS(mid)
;
```

```
--
-- TABLE: COMMENT
--

ALTER TABLE COMMENT ADD CONSTRAINT RefTHANKS20
    FOREIGN KEY (commentid, mid)
    REFERENCES THANKS(tid, mid)
;


--
-- TABLE: DEPARTMENT
--

ALTER TABLE DEPARTMENT ADD CONSTRAINT RefDEPARTMENT_HEAD29
    FOREIGN KEY (headid)
    REFERENCES DEPARTMENT_HEAD(headid)
;

ALTER TABLE DEPARTMENT ADD CONSTRAINT RefCOMPANY32
    FOREIGN KEY (cid)
    REFERENCES COMPANY(cid)
;


--
-- TABLE: DEPARTMENT_HEAD
--

ALTER TABLE DEPARTMENT_HEAD ADD CONSTRAINT RefEMPLOYEE8
    FOREIGN KEY (headid)
    REFERENCES EMPLOYEE(eid)
;


--
-- TABLE: EMPLOYEE
--

ALTER TABLE EMPLOYEE ADD CONSTRAINT RefDEPARTMENT30
    FOREIGN KEY (did)
    REFERENCES DEPARTMENT(did)
;


--
-- TABLE: EXECUTIVE
--
```

```
ALTER TABLE EXECUTIVE ADD CONSTRAINT RefEMPLOYEE7
    FOREIGN KEY (execid)
    REFERENCES EMPLOYEE(eid)
;

ALTER TABLE EXECUTIVE ADD CONSTRAINT RefCOMPANY28
    FOREIGN KEY (cid)
    REFERENCES COMPANY(cid)
;


--
-- TABLE: LIKE
--

ALTER TABLE LIKE ADD CONSTRAINT RefTHANKS19
    FOREIGN KEY (likeid, mid)
    REFERENCES THANKS(tid, mid)
;


--
-- TABLE: THANKS
--

ALTER TABLE THANKS ADD CONSTRAINT RefCOMPANY_VALUE17
    FOREIGN KEY (vid)
    REFERENCES COMPANY_VALUE(vid)
;

ALTER TABLE THANKS ADD CONSTRAINT RefEMPLOYEE22
    FOREIGN KEY (to)
    REFERENCES EMPLOYEE(eid)
;

ALTER TABLE THANKS ADD CONSTRAINT RefEMPLOYEE24
    FOREIGN KEY (from)
    REFERENCES EMPLOYEE(eid)
;

ALTER TABLE THANKS ADD CONSTRAINT RefMESSAGE31
    FOREIGN KEY (mid)
    REFERENCES MESSAGE(mid)
;
```

VI.2 Database Queries

Given below are 11 example English queries, with their SQL DML used to retrieve the necessary data.

1. List the names of all employees in the company "I Love Thanks".

```
SELECT e.name
FROM company as c
INNER JOIN department as d
ON c.cid = d.cid
INNER JOIN employee as e
ON d.did = e.did
WHERE c.cTitle = "I Love Thanks"
;
```

2. Show all department names in the corporation "Blitz".

```
SELECT d.depTitle
FROM company AS c
INNER JOIN department as d
ON c.cid = d.cid
WHERE c.cTitle = "Blitz"
;
```

3. Return a list of all employees in the database who have nicknames.

```
SELECT e.name, e.nickname
FROM employee AS e
WHERE EXISTS (
    SELECT e.nickname
    FROM employee
)
;
```

4. Show names of employees who do not have photos in the "I Love Thanks" company.

```
SELECT e.name
FROM company as c
INNER JOIN department as d
ON c.cid = d.cid
INNER JOIN employee as e
ON d.did = e.did
WHERE c.photo IS NULL
AND c.cTitle = "I Love Thanks"
;
```

5. Show all department heads in “Insomniac Corporation”.

```
SELECT dh
FROM company as c
INNER JOIN department as d
ON c.cid = d.cid
INNER JOIN department_head as dh
ON d.headid = dh.headid
WHERE c.cTitle = "Insomniac Corporation"
;
```

6. Show all of the Thanks that Julia Crow from “Playa Medical” gave.

```
SELECT t
FROM company as c
INNER JOIN department as d
ON c.cid = d.cid
INNER JOIN employee as e
ON d.did = e.did
INNER JOIN thanks as t
ON t.from = e.eid
WHERE c.cTitle = "Playa Medical"
AND e.name = "Julia Crow"
;
```

7. Show all of the Thanks that Emma Cross from “Boeing” received.

```
SELECT t
FROM company as c
INNER JOIN department as d
ON c.cid = d.cid
INNER JOIN employee as e
ON d.did = e.did
INNER JOIN thanks as t
ON t.from = e.eid
WHERE c.cTitle = "Boeing"
AND e.name = "Julia Crow"
;
```

8. Show all thanks that have been given in the corporation “First America”.

```
SELECT t
FROM company as c
INNER JOIN department as d
ON c.cid = d.cid
```

```
INNER JOIN employee as e
ON d.did = e.did
INNER JOIN thanks as t
ON t.to = e.eid
WHERE c.cTitle = "First America"
;
```

9. Who is the newest employee to have joined the company “Lightning Corporation”?

```
SELECT e.name
FROM company as c
INNER JOIN department as d
ON c.cid = d.cid
INNER JOIN employee as e
ON d.did = e.did
WHERE c.cTitle = "Lightning Corporation"
AND e.started = (
    SELECT MAX(e.started)
    FROM company as c
    INNER JOIN department as d
    ON c.cid = d.cid
    INNER JOIN employee as e
    ON d.did = e.eid
    WHERE c.cTitle = "Lightning Corporation"
)
;
```

10. List all thanks posted in the database in October.

```
SELECT t
FROM thanks as t
WHERE MONTHNAME(t.thanksdate) = "October"
;
```

11. List the executives in the company “I Love Thanks”.

```
SELECT e
FROM company as c
INNER JOIN executive as e
ON c.cid = e.cid
WHERE c.cTitle = "I Love Thanks"
;
```

VI.3 Design Tradeoffs and Limitations

One design tradeoff in its current state is the lack of Thanks being tied to the company. In order to gain access to all thanks by company, multiple joins must be performed instead of having a relationship directly between company and the thanks given within a company. This may be remedied in the future by adding a relationship between THANKS and COMPANY.

Another design tradeoff is the lack of relationship between COMPANY and COMPANY_VALUE. This issue is very similar to the tradeoff listed above: multiple joins are required in order to get the COMPANY_VALUE when a relationship could be tied between COMPANY and COMPANY_VALUE.

Finally, one design difficulty was in how employees are linked to thanks. Two foreign keys are required in a Thanks entity, which have been aliased to 'to' and 'from'. Both of these reference an employee, and can create difficulty due to the aliasing of the foreign keys. However, this is necessary in the current implementation.