Thanks Corporation Database Project
CMSI 486 Enterprise Project
Fall 2014

Edward Bramanti

This project designs and implements a database for an employee recognition system for companies to use internally.

Contents

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

- Thanks Form
  - Name of Employee Being Thanked
  - Message
  - Company Value (optional)
    Represents a custom value that the company wants to encourage in their employees
- Employee Profile Page
  Allows for editing of a specific employee's profile. Some of the values stored in the database will not be able to be changed, such as Real Name and Job Title, and must be updated by an admin.
  - Edit Employee's Nickname
  - Edit Employee's Photo
- "Thanks Feed" of Company
  Allows employees to see the "Thanks Feed" of the company, a place where signed-in employees can like Thanks messages that have been given and comment/like those thanks messages.
- Thanks Item in "Thanks Feed"
  Allows an employee to take action on a Thanks item within the "Thanks Feed".
  - Like Current Message
    Allows employee to like a Thanks Given, whether addressed directly or external from the employee.
  - Comment on Current Message
    A comment is a text response to a Thanks Given, whether addressed directly or external from the employee.
- Admin Panel for Department
  Allows for department heads to manage information about their department.
  - Edit Department Title
    If the title of a department changes slightly, a department head is able to change this.

  - Edit Department Description
    Allows department head to change the department description information.
  - Edit Department Employee's Job Title
    Allows department head to select a specific employee's job title in their department and edit it.

- Admin Panel for Executive
  Allows for executives to manage information about their company, and to change even department-level information.
  - Edit Company Title
    Allows executives to change their company name if their corporation undergoes a name change.
  - Edit Company Founded Date
    Allows executives to set the founding date of the company.
  - Set Department Heads
    Allows executives to set the head of each department.

- Webmaster Panel for DB Manager
  Since Thanks Corporation is a business that provides its database service to other companies, a webmaster needs an admin panel to manage the many companies in the database.
  - View Companies in Database
    Allows webmaster to view all companies in the database
  - Edit Companies in Database
    Allows webmaster to deactivate companies or remove companies from database.

IV.2 Assumptions

1. Fellow employees address thanks to other employees.
2. Each employee is assigned an account, which has predefined data and some limited customization/personalization.
3. Employees can view a newsfeed-like interface of all thanks being given throughout the company.
4. Employees have the ability to view and like/comment on all Thanks company-wide.
5. Department heads will be able to manage their department info and certain employee info.
6. Executives will be able to manage department heads and company info.

IV.3 User-Oriented Data Dictionary

Datum	Information Definition
comment_data	Text data contained within a comment on a Thanks data type
companies	View of companies for a webmaster of Thanks
company_title	Title of a company using Thanks
company_value	Value being exuding that represents the company in the Thanks
department_description	Description of particular department in text form
department_title	The name of a particular department within the company
employee_department	Department employee works in
employee_name	Name of employee in the company, form
employee_nickname	Nickname of an employee
employee_photo	Photo of an employee for the Thanks database
employee_title	Job title of an employees position
founded_date	Date that a company was founded
like_data	Stored when an employee likes a Thanks another employee game
message_data	Text data contained within the body of a Thanks data type
thanks_feed	List of all Thanks in a corporation

IV.4 Cross-Reference Table

Datum	**Form/Screen**						
	Thanks Form	**Employee Profile**	**”Thanks Feed”**	**Thanks Item**	**Department Admin Panel**	**Executive Admin Panel**	**Webmaster Admin Panel**
comment_data			X	X			
companies							X
company_title			X			X	
company_value	X						
department_description					X		
department_title					X		
employee_department		X		X			
employee_name	X	X		X	X	X	
employee_nickname		X		X	X	X	
employee_photo		X		X			
employee_title		X		X	X	X	
founded_date						X	
like_data			X	X			
message_data	X		X	X			
thanks_feed			X				

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
  - 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

- COMPANY_OF_DEPARTMENT(did, cid)
  CK - did

- DEPARTMENT_OF_EMPLOYEE(eid, did)
  CK - eid

- THANKS_TO(to,eid)
  CK - to

- THANKS_FROM(from,eid)
  CK - from

- THANKS_MESSAGE(tid,mid)
  CK - tid

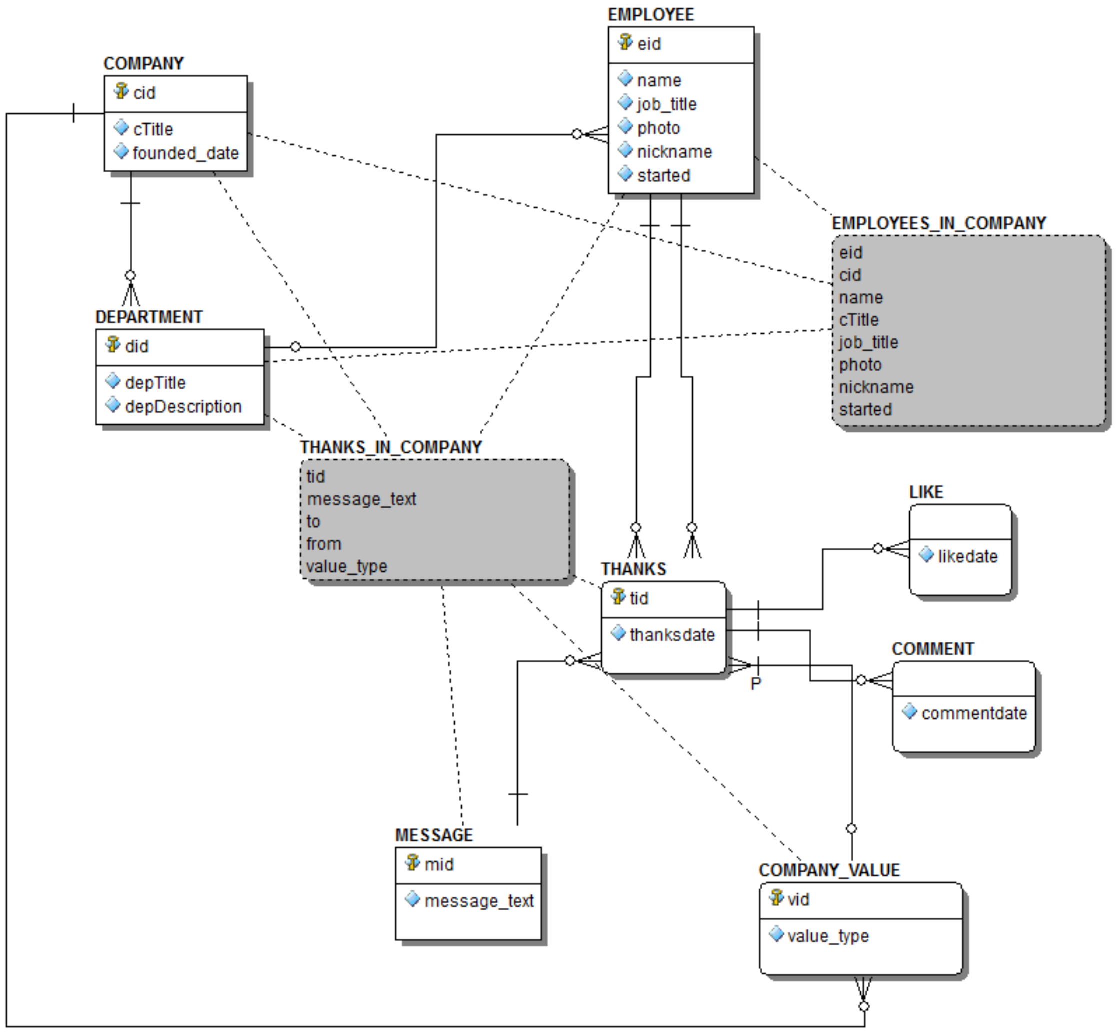
- THANKS_LIKE(likeid,mid)
  CK - likeid

- THANKS_COMMENT(commentid,mid)
  CK - commentid

- THANKS_VALUE(thanks.vid,company_value.vid)
  CK - thanks.vid

- VALUES_OF_COMPANY(cid, vid)
  CK - cid

V.1.3 Entity-Relationship Diagram of the Enterprise



V.2 Conceptual Model of the Enterprise

A conceptual model of the Thanks database.

company(<u>cid</u>, cTitle, founded_date)
PK - cid
CK - cid, cTitle
AK - cTitle

department(<u>did</u>, depTitle, depDescription)
PK - did
CK - did
FK - department.headid REFERENCES departmenthead.headid
department.cid REFERENCES company.cid

employee(<u>eid</u>, name, job_title, photo, nickname, started)
PK - eid
CK - eid, name, photo
AK - name, photo
FK - employee.did REFERENCES department.did

thanks(<u>(tid, mid)</u>, thanksdate)
PK - (tid, mid)
CK - tid
FK - thanks.mid REFERENCES message.mid
thanks.to REFERENCES employee.eid
thanks.from REFERENCES employee.eid
thanks.vid REFERENCES company_value.vid

like(<u>(likeid, mid)</u>, likedate)
PK - (likeid, mid)
CK - likeid
FK - like.mid REFERENCES message.mid

comment(<u>(commentid, mid)</u>, commentdate)
PK - (commentid, mid)
CK - commentid
FK - comment.mid REFERENCES message.mid

message(<u>mid</u>, message_text)
PK - mid
CK - mid

company_value(<u>vid</u>, value_type)
PK - vid
CK - vid
FK - company_value.cid REFERENCES company.cid

V.3 Table Dictionary

Table	Attributes	Definition
COMPANY	cid, cTitle, founded_date	Represents a company, which is the starting data point
DEPARTMENT	did, depTitle, depDescription, headid, cid	Department in a company
EMPLOYEE	eid, name, job_title, photo, nickname, started, did	Employee in a company
THANKS	tid, thanksdate	Data type used to recognize another employee
LIKE	likeid, mid, likedate	A like on a Thanks post
COMMENT	commentid, mid, comment-date	A comment on a Thanks post
MESSAGE	mid, message_text	Message of a Thanks
COMPANY_VALUE	vid, value_type	Company value of a Thanks

V.4 Attribute Dictionary

Attribute	Tables Used In	Description
eid	EMPLOYEE	Unique identifier of an employee
name	EMPLOYEE	Name of an employee
job_title	EMPLOYEE	Job title of employee
photo	EMPLOYEE	Photo of employee
nickname	EMPLOYEE	Nickname of employee
started	EMPLOYEE	Date that the employee started at their company
did	DEPARTMENT	Unique identifier of the department of a company
depTitle	DEPARTMENT	Name (title) of the department
depDescription	DEPARTMENT	Description of the department
cid	COMPANY	Unique identifier of a company
cTitle	COMPANY	Name (title) of a company
founded_date	COMPANY	Date a company was founded
tid	THANKS	Unique identifier of a Thanks
thanksdate	THANKS	Date the Thanks was given
likeid	LIKE	Unique identifier of a like on a Thanks
likedate	LIKE	Date the like was given
commentid	COMMENT	Unique identifier of a comment
commentdate	COMMENT	Date the comment was added to the Thanks
mid	MESSAGE	Unique identifier of the message of a Thanks
message_text	MESSAGE	Body text of the message of a Thanks
vid	COMPANY_VALUE	Unique identifier of a company value
value_type	COMPANY_VALUE	Name (type) of company value

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 25, 2014 23:37:27
-- 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,
    cid           VARCHAR(30)    NOT NULL,
    value_type    VARCHAR(30)    NOT NULL,
    PRIMARY KEY (vid, cid)
)ENGINE=INNODB
;



--
-- TABLE: DEPARTMENT
--

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

--
-- VIEW: EMPLOYEES_IN_COMPANY
```

```
--

CREATE VIEW EMPLOYEES_IN_COMPANY AS (
SELECT e.eid
         , c.cid
         , e.name
         , c.cTitle
         , e.job_title
         , e.photo
         , e.nickname
         , e.started
FROM company as c
        INNER JOIN department AS d ON c.cid = d.cid
        INNER JOIN employee AS e ON e.did = d.did
)

--
-- VIEW: THANKS_IN_COMPANY
--

CREATE VIEW THANKS_IN_COMPANY AS (
    SELECT t.tid
         , m.message_text
         , t.to
         , t."from"
         , cv.value_type
    FROM company as c
        INNER JOIN department AS d ON c.cid = d.cid
        INNER JOIN employee AS e ON e.did = d.did
        INNER JOIN thanks AS t ON t.to = e.eid
        INNER JOIN message AS m ON t.mid = m.mid
        INNER JOIN company_value as cv ON t.vid = cv.vid
)

--
-- TABLE: COMMENT
--

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


--
-- TABLE: COMPANY_VALUE
--
```

```
ALTER TABLE COMPANY_VALUE ADD CONSTRAINT RefCOMPANY33
    FOREIGN KEY (cid)
    REFERENCES COMPANY(cid)
;


--
-- TABLE: DEPARTMENT
--

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


--
-- TABLE: EMPLOYEE
--

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


--
-- 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, cid)
    REFERENCES COMPANY_VALUE(vid, cid)
;

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.

Chapter VII

Database Integrity and Security

VII.1 Functional Dependencies

- cid $\rightarrow$ cTitle, founded_date
- did $\rightarrow$ depTitle, depDescription
- eid $\rightarrow$ name, job_title, photo, nickname, started
- tid $\rightarrow$ thanksdate
- likeid $\rightarrow$ likedate
- commentid $\rightarrow$ commentdate
- mid $\rightarrow$ message_text
- vid $\rightarrow$ value_type

VII.2 Adjustments for Normalization

According to these functional dependencies and after looking at the ERD, it is clear that the database has already been normalized. Therefore, no adjustments will be needed on top of what already is in normalizing this database.

VII.3 Integrity and Security

1. Users and Granted Privileges
   There will multiple types of access privileges to the database. These access types are listed below.

   - Webmaster User
     A webmaster represents access privileges for the owner of Thanks Corporation. He has complete access to all data at any time.
   - Executive User
     Allows access to an executive's own company entity, and allows for modification of its attributes.

- Department Head User
  Allows access to a department head's own department entity, and allows for modification of its attributes.
- Employee User
  Allows normal access for an employee: modification of own employee entity and insert/update/delete privileges for own Thanks and related entities.

2. Assertion
   MySQL does not directly provide assertions, however triggers have been used to provide assertion-like functionality in MySQL. Assertions the Thanks Corporation needs are listed below.

   - Determine whether an employee giving a Thanks with a company value is using a company value from the employee's own company.

3. Views
   Two views are utilized in this database.

   - EMPLOYEES_IN_COMPANY
     Returns employees from all departments associated with a company.
   - THANKS_IN_COMPANY
     Returns all thanks from within a company.

Chapter VIII

Implementation Notes

VIII.1 Indices

VIII.2 Data

VIII.3 Query Trace

VIII.4 Implementation Assessment