

Project #1 - Backend of Jobster Report

Team Member: Yixuan Tang (yt1369@nyu.edu), Isha Chaturvedi (ic1018@nyu.edu)

1. Project Description

In this project, a relational database is designed using MySQL that can serve backend for the social networking site “Jobster”. The database is further loaded on the portable web application “phpMyAdmin” to be used to design the web front-end in the next part of the project.

2. Introduction

This is a relational database design project, to develop the backend of the social networking site called “Jobster”. Since the focus of the website is to provide an online portal for graduating students to look for jobs, the database is designed to facilitate the objective. The database is designed by keeping in mind the following general requirements:

- The site lets the students to sign up and add their resume and other relevant information to the website.
- The site is also a medium for the companies to connect with the students or future employees, and it lets the companies to sign up as well.
- The companies can also post their information and job announcements for the vacant positions.
- A student can follow the companies of his/her interest. Once, a student follows a company, he/she will start getting all the job announcements from that particular company.
- Apart from following companies, students can also apply for jobs (primary role of the website), and can connect with other students by being friends with them as well.
- Once a student get friends with someone, he/she can send and receive messages, forward tips, and job announcements to and from that friend.
- A company can have multiple followers, and a student can follow multiple companies. Similarly, a student can have multiple friends.
- Both companies and students can search for matching applicants and job descriptions.

3. Workflow

Here is the whole procedure of how relational backend of Jobster was developed:

- The database is developed using MySQL (MySQL workbench). It is further managed with phpMyAdmin, which is connected with MySQL database. This makes it easy to execute queries on the web application.
- All the data about students, companies, job announcements, connections (friendships and followings), notifications(message, friend request, job notification) and job applications are stored in a relational database.
- The database is first designed using Entity-Relationship (ER) model, and later it is translated to a Relational model.
- The database went through several iterations of testing (the database is populated with sample data to carry out the testing) and re-design phases, to make sure it is equipped to execute the required sql queries and operations.

4. Database Schema Design

The schema of the database is first designed using ER model and later relational model. The schema is designed in such a way that each column in the table is space efficient and normalized.

4.a Entity-Relationship (ER) model

The entity-relationship model, is typically used to represent the conceptual design⁴. The schema specifies the entities that are represented in the database, the attributes of the entities, the relationships among the entities, and constraints on the entities and relationships. This conceptual-design phase resulted in the creation of an entity-relationship diagram that provided a graphic representation of the schema (Figure 4.a.1).

In the ER diagram below, rectangles with titles represent the attributes of a relationship set. The attributes that are part of the primary key are underlined. The diamonds represent relationship sets (for example messages and friend requests sent by students). The dashed lines link attributes of a relationship set to the respective relationship set (in this case, the follower relationship has time attribute). This helps in removing redundancy and make the logical relationship between student and company explicit. This also show that student entity is related to the company entity set via follower relationship set.

The double lines indicate total participation of an entity in a relationship set. In this case application entity has a total participation in the position entity. Similarly, Message, Request and application entities have a total participation in student entity. The double diamonds represent the relationship sets that are linked to weak entity sets. A weak entity set is one that is dependent on another entity set (strong entity set) for its existence. Here, application is a weak entity set and is thus dependent on position entity set. Similarly, Message, Request, Notification and application entities are weak

entities and are dependent on student entity via send, send, apply, and to relationship sets respectively. The notification set is also a weak entity in notification - position relationship, and is dependent on position entity set.

The arrow between position and company entities directs to company, indicating that a company can post multiple positions, while a particular position can only be posted by a particular company (constraint). There's no arrow between student and company in the follower relationship indicating that a student can follow multiple companies and a company can have many student followers. Similarly, a student can send multiple friend requests or messages, but a particular message or friend request is sent by a particular student. The same is the case for application and notification, where a students can send multiple applications or notifications but vice-versa is not true. Similarly, there could be announcement (of job positions) of multiple applications, but a specific application belongs to a specific announcement of a job position.

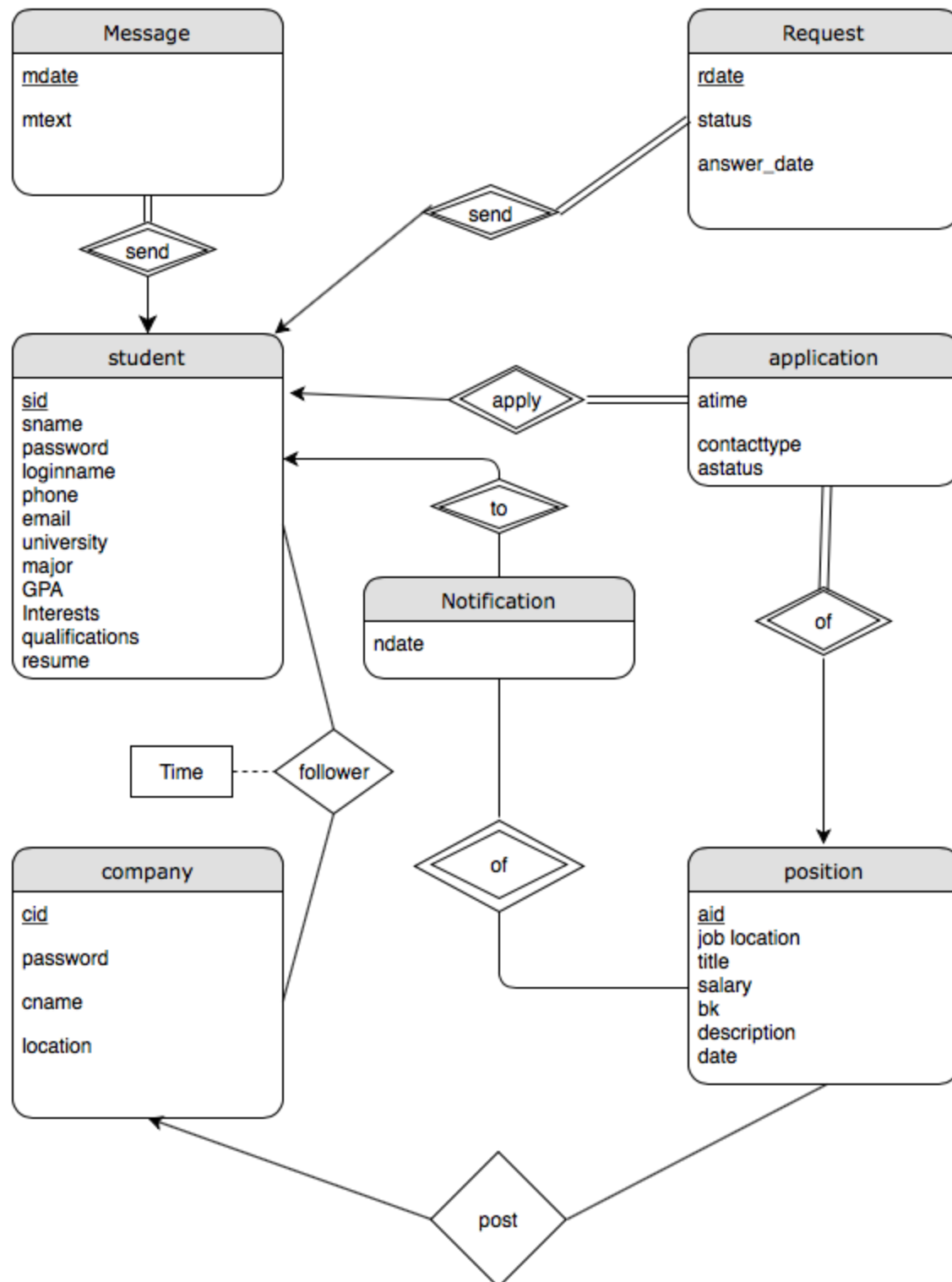


Figure 4.a.1: ER diagram of the database of Jobster website

Comment:

1. ~~两个foreign key 指向student 需要~~
2. ~~notification 写成relation~~
3. ~~弱实体 (application) 改成relation~~
4. student里加一个feature 限制别人看他主页 : why need a feature? Why not just validate through friend relationship?
5. Forward job 的时候加一个新的relation 在position和student之间
6. 双菱形表示弱实体与强实体之前的关系, 双线表示total participation
7. 弱实体的主键应该用虚线

Multiple errors in ER diagram against the schema, plz review previous ER sample to correct it. Suggest to have the feature that can tell if notification viewed or new. missing the features that user can limit access to their profile.(students should be able to restrict access to their information (say, their GPA and resume) so that it is only visible to their friends and to those companies that they have applied to; other students and companies can only see their name, major, and university.)

4.b Relational model

The above ER model (Figure 4.a.1) is translated to a relational model. All the entities are translated into tables in the relational model. The relationship "follower" is also converted to a table. Initially, a separate table was kept for user login (one login table for student and one for company separately), that is the Student and Company tables didn't have user login info feature within the same tables. However this created redundancy, and hence the two separate user login info tables for student and company were removed, and their columns were instead combined with the Student and Company tables respectively. Similarly, a separate Friends table was removed, and its columns were combined with the Request table. These changes were then reflected both in the ER diag (Figure 4.a.1 is the updated ER diag) and the relational model. The above changes made the schema look condensed and non-redundant.

The updated relational model:

Student (sid, sname, password, loginname, phone, email, university, major, GPA, interests, qualifications, Resume)

Company (cid, password, cname, location, industry)

Position (aid, cid, joblocation, title, salary, bk, description, date)

Follower (cid, sid, time)

Request (rdate, sid, Friendid, status, answer_date)

Message (mdate, sid1, sid2, mtext)

Notification (aid, sid, ndate)

Application (aid, sid, atime, contacttype, astatus)

Here, the columns that are underlined remember represent the primary keys (unique identity) for the respective tables.

Primary Keys:

Student: sid

Company: cid

Position: aid

Follower: cid and sid (together)

Request: rdate, sid, Friendid (together)

Message: mdate, sid1, sid2 (together)

Notification: aid and sid (together)

Application: aid, sid (together)

Foreign Keys:

position.cid REFERENCES Company.cid;

Follower.cid REFERENCES Company.cid,

Follower.sid REFERENCES Student.sid;

Request.sid REFERENCES Student.sid,

Request.Friendid REFERENCES Student.sid;

Message.sid1 REFERENCES Student.sid,

Message.sid2 REFERENCES Student.sid;

Notification.aid REFERENCES position.aid,

Notification.sid REFERENCES Student.sid;

Application.aid REFERENCES position.aid,

Application.sid REFERENCES Student.sid;

Detailed explanation:

- Student: sid is an id to identify each student, and is thus unique. sname is student name, loginname and password are the login info of the student. phone and email are the contact info for the student, and the rest other columns serve as the profile of the student.
- Company: cid is an id to identify each company, and is unique, that is no two companies can have same cid. cname is company name, and a company has to register with cname and password. location column represents the headquarters of the company, and industry is the type or category of industry the company belong to.
- Position: aid is the position id (The position for which the company has given the announcement of hiring) and is unique. date is the announcement date of the position. bk is

the required academic background (e.g., an MS in Physics, or a BS in any field) for the position, and description is longer textual description of the position.

- Follower: time is the exact date and time when a student followed a company. Here, no two tuples can have same cid and sid together.
- Request: rdate is the date when the friendship request was sent. Sid is the id of the user who sent the request, Friendid is the id of the user who received the request. status is the friendship request status: pending, accepted, rejected, and answer_date is the date, when the student either accepts or rejects a request. This column is left NULL if the status is still “pending”. Here no two tuples can have same rdate, sid and Friendid.
- Message: mdate is the date when the message was sent. sid1 and sid2 are the two id’s of the students among whom the message conversation was exchanged. mtext is the message that sid1 sent to sid2. No two tuples in this table can have same sid1, sid2 and mdate together.
- Notification: ndate is the date when the notification was given out. Here, no two tuples in the table can have same aid and sid.
- Application: atime is the time when a student apply for the particular position. contacttype is the way a company can reach out to a student (phone or email). It is assumed that a student can only apply for a particular position once. Hence, no two tuples in this table can have same aid and sid.

5. Database Tables

Following are the database tables along with the respective columns and their data types:

```

• DROP TABLE IF EXISTS `Student`;
• CREATE TABLE `Student` (
  `sid` VARCHAR(10) NOT NULL,
  `sname` VARCHAR(20) NOT NULL,
  `password` VARCHAR(20) NOT NULL,
  `loginname` VARCHAR(20) NOT NULL,
  `phone` VARCHAR(10),
  `email` VARCHAR(45),
  `university` VARCHAR(40),
  `major` VARCHAR(15),
  `GPA` VARCHAR(10),
  `interests` VARCHAR(40),
  `qualifications` VARCHAR(30),
  `Resume` BLOB,
  UNIQUE(`sid`),
  UNIQUE(`loginname`),
  UNIQUE(`email`),
  UNIQUE(`phone`),
  PRIMARY KEY (`sid`));

```

Table 5.1 Student table

```

• DROP TABLE IF EXISTS `Company`;
• CREATE TABLE `Company` (
  `cid` VARCHAR(10) NOT NULL,
  `password` VARCHAR(10) NOT NULL,
  `cname` VARCHAR(20) NOT NULL,
  `location` VARCHAR(30),
  `industry` VARCHAR(45),
  UNIQUE(`cid`),
  UNIQUE(`cname`),
  PRIMARY KEY (`cid`));

```

Table 5.2 Company table

```

• DROP TABLE IF EXISTS `position`;
• CREATE TABLE `position` (
  `aid` VARCHAR(10) NOT NULL,
  `cid` VARCHAR(10) NOT NULL,
  `joblocation` VARCHAR(10) NOT NULL,
  `title` VARCHAR(25) NOT NULL,
  `salary` VARCHAR(20) NOT NULL,
  `bk` VARCHAR(30) NOT NULL,
  `description` VARCHAR(45) NOT NULL,
  `date` DATETIME NOT NULL,
  UNIQUE(`aid`),
  PRIMARY KEY(`aid`),
  FOREIGN KEY(`cid`) REFERENCES `Company`(`cid`));

```

Table 5.3 Position table

```

• DROP TABLE IF EXISTS `Follower`;
• CREATE TABLE `Follower` (
  `cid` VARCHAR(10) NOT NULL,
  `sid` VARCHAR(10) NOT NULL,
  `time` DATETIME NOT NULL,

  PRIMARY KEY(`cid`,`sid`),
  FOREIGN KEY(`cid`) REFERENCES `Company`(`cid`),
  FOREIGN KEY(`sid`) REFERENCES `Student`(`sid`));

```

Table 5.4 Follower table

```

• DROP TABLE IF EXISTS `Request`;
• CREATE TABLE `Request` (
  `sid` VARCHAR(10) NOT NULL,
  `Friendid` VARCHAR(10) NOT NULL,
  `status` VARCHAR(10) NOT NULL,
  `rdate` DATETIME NOT NULL,
  `answer_date` DATETIME,
  PRIMARY KEY(`sid`,`Friendid`,`rdate`),
  FOREIGN KEY(`sid`) REFERENCES `Student`(`sid`),
  FOREIGN KEY(`Friendid`) REFERENCES `Student`(`sid`));

```

Table 5.5 Request table

```

• DROP TABLE IF EXISTS `Message`;
• CREATE TABLE `Message` (
  `sid1` VARCHAR(10) NOT NULL,
  `sid2` VARCHAR(10) NOT NULL,
  `mtext` VARCHAR(100) NOT NULL,
  `mdate` DATETIME NOT NULL,

  PRIMARY KEY(`sid1`,`sid2`,`mdate`),
  FOREIGN KEY(`sid1`) REFERENCES `Student`(`sid`),
  FOREIGN KEY(`sid2`) REFERENCES `Student`(`sid`));

```

Table 5.6 Message table

```

• DROP TABLE IF EXISTS `Application`;
• CREATE TABLE `Application` (
  `aid` VARCHAR(10) NOT NULL,
  `sid` VARCHAR(10) NOT NULL,
  `atime` DATETIME NOT NULL,
  `contacttype` VARCHAR(10) NOT NULL,
  `astatus` VARCHAR(10) NOT NULL,
  PRIMARY KEY(`aid`,`sid`),
  FOREIGN KEY(`aid`) REFERENCES `position`(`aid`),
  FOREIGN KEY(`sid`) REFERENCES `Student`(`sid`));

```

Table 5.7 Application table

```

• DROP TABLE IF EXISTS `Application`;
• CREATE TABLE `Application` (
  `aid` VARCHAR(10) NOT NULL,
  `sid` VARCHAR(10) NOT NULL,
  `atime` DATETIME NOT NULL,
  `contacttype` VARCHAR(10) NOT NULL,
  `astatus` VARCHAR(10) NOT NULL,
  UNIQUE(`aid`,`sid`),
  PRIMARY KEY(`aid`,`sid`),
  FOREIGN KEY(`aid`) REFERENCES `position`(`aid`),
  FOREIGN KEY(`sid`) REFERENCES `Student`(`sid`));

```

Table 5.8 Notification table

Detailed explanation:

- When a new student user register in this website, he/she needs to set student name, loginname and password, and system would assign a default sid in database to identify this user, the user can later update his/her information(phone, email, university etc.), the email and phone should be unique for each user. Here the datatype of Resume is kept BLOB (65535 bytes maximum) as it is expected that Resume is a large object.
- When a company user register in this website, it first needs to set password and cname first, and then it can further add more information(location, industry).
- A company can announce positions for hiring by giving details like title, joblocation, salary, background requirements (bk), and further description about the position. Each announced positions have a unique aid.
- A student can follow multiple companies, and this is tracked by Follower table.
- A student can send friend request to other students. The status of the friend request is tracked through the status column in the Request table.

- A student can also send messages or some announcements to other students. The Message table keeps the record of this data. The limit on the messages is 100 characters.
- Application table keeps the record of the applications that students have filled.
- A student can receive notifications about job announcements from the companies it has followed. The record of the notifications is kept through Notifications table.
- The data type (and its size) of each column is set to make sure the tables are space efficient, and allow sufficient room for the users (students and companies) to input their information or search through the database.

6. Assumptions

The database design described in Section 4.5 and 5 have following assumptions:

- A student can send messages, forward tips or job announcements to another students only if they are friends.
- A company can send announcements to a particular student (that is the announcement is not global), only if the student follows that company.

These assumptions will be handled with the SQL queries while developing the frontend in the next project.

Further assumptions:

- It is assumed in the university column in the Student table, that students enter the university name in which they are currently enrolled in.
- A student can apply for one particular position only once.

7. Constraints

The database design follows some constraints (mentioned in Section 5) as well:

- Every student user should have unique: student id, login name, email address and phone number, that is there cannot be two students with same login name and contact information (email address and phone number).
- Every company should have unique: company id and company name.

These constraints makes sure that a particular student or a company don't register into the database repeatedly. The constraints are handled through the keyword "UNIQUE".

Along with the UNIQUE constraint mentioned above each column of the table in the database has its constraints:

- NOT NULL - In Student table (5.1) the values in sid, sname, password, loginname must not be null. In Company table (5.2) the values in cid, cname, password cannot be null. In Position table (5.3), all the columns cannot have null values. Similar is the case in the Follower table (5.4), Message table (5.6), Application table (5.7) and Notification table (5.8). In Request table (5.5), all the values except answer_date must not be null.
- UNIQUE - The UNIQUE constraint for Student and Company tables is already mentioned above. The Position table (5.3) has a UNIQUE constraint on aid column. The rest other columns don't have any UNIQUE constraint.
- PRIMARY KEY and FOREIGN KEY constraint - These constraints are already discussed in Section 4.b.

8. SQL Query Tests

Each table in the database is populated with interesting and meaningful sample data (Table 8.1 - Table 8.8).

sid	sname	password	loginname	phone	email	university	major	GPA	interests	qualifications	Resume
U1245	Joan Arc	MarkTwain	Joan	5535	joan@nyu.edu	NYU	CS	3.34	playing games	BS in Computer Science	[BLOB - 34 B]
U1246	Yu Chen	sql2018	Yu	5270	YU@fordham.edu	Fordham	EE	3.9	soccer	MS in Marketting	[BLOB - 27 B]
U2384	Gaurav Bhardwaj	FrenchD	Gaurav Bhardwaj	5327	gb242@nyu.edu	NYU	CS	3.3	soccer	BS in Electronics	[BLOB - 55 B]
U3984	Charles Moffett	CharlesS	Charles Moffett	5273	cm6353@columbia.edu	Columbia	Marketing	3.9	soccer	MS in Marketting	[BLOB - 53 B]
U4785	Isha Chaturvedi	Joshua	Isha Chaturvedi	5353	ic1018@nyu.edu	NYU	CS	3.5	dancing	MS in Data Science	[BLOB - 59 B]
U7234	Yixuan Tang	Dorling	Tang	5674	yt41@stanford.edu	Stanford	CS	3.9	singing	MS in Urban Data Science	[BLOB - 58 B]

Table 8.1 Student table

cid	password	cname	location	industry
C1245	db2016	Google	Mt View	Technology
C2384	Yahoo2	Yahoo	Sunntvale	Webservices
C3984	cc2009	Facebook	Menlo Park	Social Media
C4785	jobs111	Apple	Cupertino	Technology
C7234	Mic2008	Microsoft	Washington	Technology

Table 8.2 Company table

aid	cid	joblocation	title	salary	bk	description	date
A1245	C1245	Mt View	Developer	80000	BS in Computer Science	Application Development	2017-02-20 08:06:13
A2384	C2384	Sunnyvale	Data Engineer	75000	Masters in Data Science	Create API	2017-12-20 08:06:13
A3984	C3984	Menlo Park	UX Writer	65000	BS in any field	Writing Story	2018-02-20 18:06:13
A3985	C3984	Menlo Park	UI developer	65000	MS in Computer Science	Front-end developer	2018-04-19 08:26:13
A4785	C4785	Cupertino	Software Engineer	90000	BS in Computer Science	Debugging, Testing	2017-06-20 08:06:13
A7234	C7234	Washington	Data Scientist	100000	Masters in Data Science	Gesture Recognition	2017-08-20 08:06:13

Table 8.3 Position table

cid	sid	time
C1245	U1245	2017-10-20 08:16:10
C2384	U4785	2018-01-20 08:16:10
C2384	U7234	2017-12-20 08:16:10
C3984	U1245	2017-10-15 08:16:10
C4785	U4785	2017-10-20 08:12:10
C7234	U1245	2017-10-20 22:16:10

aid	sid	ndate
A1245	U1245	2017-10-20 08:06:13
A1245	U1246	2018-03-20 08:16:13
A3984	U4785	2018-02-20 22:06:22
A4785	U4785	2017-11-20 08:22:13
A7234	U7234	2017-12-20 18:06:13

Table 8.4 Follower table

Table 8.5 Notification table

sid	Friendid	status	rdate	answer_date
U1245	U2384	pending	2018-03-20 08:26:12	NULL
U1245	U4785	accepted	2017-10-20 08:06:10	2017-10-20 08:16:10
U1246	U1245	accepted	2018-04-01 11:26:10	2018-04-02 11:26:10
U3984	U1246	pending	2018-01-15 00:00:00	NULL
U4785	U2384	rejected	2017-11-20 08:06:10	2017-11-20 18:06:10
U4785	U7234	accepted	2018-02-20 08:26:10	2018-03-20 08:26:10
U7234	U1245	pending	2017-12-20 08:06:10	NULL

Table 8.6 Request table

sid1	sid2	mtext	mdate
U1245	U4785	hi	2017-10-20 08:06:10
U1245	U4785	how is it going?	2018-03-20 12:06:10
U4785	U1245	hello	2017-11-20 09:06:10
U4785	U7234	thanks	2018-02-20 11:06:10
U7234	U4785	your profile is good	2017-12-20 10:06:10

Table 8.7 Message table





aid	sid	atime	contacttype	astatus
A1245	U1245	2017-10-20 08:06:13	email	success
A2384	U3984	2017-12-10 08:34:13	email	pending
A3984	U4785	2018-02-20 15:06:13	email	success
A4785	U4785	2017-12-20 09:06:13	phone	success
A7234	U7234	2018-01-20 12:06:13	phone	pending

Table 8.8 Application table

Following queries are tested to make sure the database performs the required functionality:

- Create a record for a new student account, with a name, a login name, and a password.

Before insert:

Result Grid	  Filter Rows:		<input type="text" value="Search"/>	Edit:	  Export/Import:							
	sid	sname	password	loginname	phone	email	university	major	GPA	interests	qualifications	Resume
▶	U1245	Joan Arc	MarkTwain	Joan	5535	joan@nyu.edu	NYU	CS	3.34	playing games	BS in Computer Science	BLOB
	U1246	Yu Chen	sql2018	Yu	5270	YU@fordham.edu	Fordham	EE	3.9	soccer	MS in Marketing	BLOB
	U2384	Gaurav Bhardwaj	FrenchD	Gaurav Bhardwaj	5327	gb242@nyu.edu	NYU	CS	3.3	soccer	BS in Electronics	BLOB
	U3984	Charles Moffett	CharlesS	Charles Moffett	5273	cm6353@columbia.edu	Columbia	Marketing	3.9	soccer	MS in Marketing	BLOB
	U4785	Isha Chaturvedi	Joshua	Isha Chaturvedi	5353	ic1018@nyu.edu	NYU	CS	3.5	dancing	MS in Data Science	BLOB
	U7234	Yixuan Tang	Dorling	Tang	5674	yt41@stanford.edu	Stanford	CS	3.9	singing	MS in Urban Data Science	BLOB
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

After insert:

6

7

8

Create a record for a new student account, with a name, a login name, and a password.

• INSERT INTO 'Student' VALUES('U7235', 'Justin', 'code110','Justin',NULL,NULL,NULL,NULL,NULL,NULL,NULL);

100%

↕

1:5

Result Grid

Filter Rows:

Q

Search

Edit:

Export/Import:

sid	sname	password	loginname	phone	email	university	major	GPA	interests	qualifications	Resume
▶ U1245	Joan Arc	MarkTwain	Joan	5535	joan@nyu.edu	NYU	CS	3.34	playing games	BS in Computer Science	BLOB
U1246	Yu Chen	sql2018	Yu	5270	YU@fordham.edu	Fordham	EE	3.9	soccer	MS in Marketing	BLOB
U2384	Gaurav Bhardwaj	FrenchD	Gaurav Bhardwaj	5327	gb242@nyu.edu	NYU	CS	3.3	soccer	BS in Electronics	BLOB
U3984	Charles Moffett	CharlesS	Charles Moffett	5273	cm6353@columbia.edu	Columbia	Marketing	3.9	soccer	MS in Marketing	BLOB
U4785	Isha Chaturvedi	Joshua	Isha Chaturvedi	5353	ic1018@nyu.edu	NYU	CS	3.5	dancing	MS in Data Science	BLOB
U7234	Yixuan Tang	Dorling	Tang	5674	yt41@stanford.edu	Stanford	CS	3.9	singing	MS in Urban Data Science	BLOB
U7235	Justin	code110	Justin	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- List the names of all friends of a particular user.



Find all friends of user with id U1245:


```
10 # List the names of all friends of a particular user.
11 #user U1245
12 • select student.sname from
13 (
14   select sid, Friendid from Request
15   where status = 'accepted' and (sid = 'U1245' OR Friendid = 'U1245')
16 )
17 as a,
18 student
19 where student.sid != 'U1245' and (a.Friendid = student.sid or a.sid = student.sid);|
20
```

100%

86:19

Result Grid

 Filter Rows:

Export: 

sname
▶ Isha Chaturvedi
Yu Chen

- Delete all friendship requests that are older than one month and that have not been answered.

Before delete:

Result Grid					
Filter Rows: <input type="text" value="Search"/>					
Edit:					
sid	Friendid	status	rdate	answer_date	
U1245	U2384	pending	2018-03-20 08:26:12	NULL	
U1245	U4785	accepted	2017-10-20 08:06:10	2017-10-20 08:16:10	
U1246	U1245	accepted	2018-04-01 11:26:10	2018-04-02 11:26:10	
U3984	U1246	pending	2018-01-15 00:00:00	NULL	
U4785	U2384	rejected	2017-11-20 08:06:10	2017-11-20 18:06:10	
U4785	U7234	accepted	2018-02-20 08:26:10	2018-03-20 08:26:10	
U7234	U1245	pending	2017-12-20 08:06:10	NULL	
NULL	NULL	NULL	NULL	NULL	

After delete:

```

21  # Delete all friendship requests that are older than one month and that have not been answered.
22  • set sql_safe_updates = 0;
23  # remember to set this back with sql_safe_updates = 1;
24
25  • DELETE FROM request
26    WHERE rdate <= now() - interval 3 month and status = 'pending';
27
28  • select * from request;
29

```

Result Grid					
Filter Rows: <input type="text" value="Search"/>					
Edit: Export/Import:					
sid	Friendid	status	rdate	answer_date	
U1245	U2384	pending	2018-03-20 08:26:12	NULL	
U1245	U4785	accepted	2017-10-20 08:06:10	2017-10-20 08:16:10	
U1246	U1245	accepted	2018-04-01 11:26:10	2018-04-02 11:26:10	
U4785	U2384	rejected	2017-11-20 08:06:10	2017-11-20 18:06:10	
U4785	U7234	accepted	2018-02-20 08:26:10	2018-03-20 08:26:10	
NULL	NULL	NULL	NULL	NULL	

- List all students from NYU that are following Microsoft.

```

30  #List all students from NYU that are following Microsoft.
31  • select s.sid, s.sname
32    from
33    follower f join student s on f.sid = s.sid join company c on f.cid = c.cid
34    where cname = 'Microsoft';
35

```

Result Grid		
Filter Rows: <input type="text" value="Search"/>		
Export:		
sid	sname	
U1245	Joan Arc	

- List all job announcements posted in the last week that are looking for someone with an MS in CS.

36

#List all job announcements posted in the last week that are looking for someone with an MS in CS.

37

•

select * from position

38

WHERE date >= curdate() - INTERVAL DAYOFWEEK(curdate())+6 DAY

39

AND date < curdate() - INTERVAL DAYOFWEEK(curdate())-1 DAY

40

AND bk = 'MS in Computer Science';

41

100%

↕

35:40

Result Grid

Filter Rows:

Search

Edit:

Export/Import:

aid	cid	joblocation	title	salary	bk	description	date
▶ A3985	C3984	Menlo Park	UI developer	65000	MS in Computer Science	Front-end developer	2018-04-19 08:26:13
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- For each student with GPA > 3.5 whose resume contains the keywords “database systems”, create a notification telling the student about a particular new job announcement that a company has posted.

Before creation:

52

53

select * from Notification;

100%

28:52

Result Grid

Filter Rows:

aid	sid	ndate
▶ A1245	U1245	2017-10-20 08:06:13
A1245	U1246	2018-03-20 08:16:13
A3984	U4785	2018-02-20 22:06:22
A4785	U4785	2017-11-20 08:22:13
A7234	U7234	2017-12-20 18:06:13
NULL	NULL	NULL

After creation:

```

42  #For each student with GPA > 3.5 whose resume contains the keywords "database systems", create a
43  #notification telling the student about a particular new job announcement that a company has posted.
44  #A1245
45  • set sql_safe_updates = 0;
46  • INSERT INTO Notification (aid, sid, ndate)
47    select aid, sid, now()
48    from
49    (select aid from position where aid = 'A1245') A,
50    (select sid from student where Resume like "%database systems%" and GPA > 3.5) b;
51
52  • select * from Notification;
53  |

```

100% 1:53

Result Grid Filter Rows: Search Edit: Export/Import:

aid	sid	ndate
A1245	U1245	2017-10-20 08:06:13
A1245	U1246	2018-03-20 08:16:13
A3984	U4785	2018-02-20 22:06:22
A4785	U4785	2017-11-20 08:22:13
A1245	U7234	2018-04-22 17:43:03
A7234	U7234	2017-12-20 18:06:13
NULL	NULL	NULL

These queries were used to improve the schema design. The design was tested repeatedly after every iteration of redesigning.

9. Further Discussion and Conclusions

In this project, a database was designed to serve as a relational backend for the “Jobster” which is a job hunting and networking site for university students. The database was designed keeping in mind that it is space efficient, non-redundant and functional. The resultant schema produced the desired results upon executing the SQL queries. Although, it will be used to carry forward the next part of the project, further improvements in the database would be identified upon actual implementation of the frontend.

10. References/Tools Used

1. MySQL Workbench
 2. phpMyAdmin on MAMP
 3. Entity-Relationship Model
 4. Abraham Silberschatz, H. F. (n.d.). *Database System Concepts, 6th Edition*. McGraw-Hill.
-