INST 733 - Database Design

Topic
AWIC Workshop Database

Instructor: Prof. Katy Lawley

Submitted by-
[Harshith Sharma, Rahul Bahadur, Sheryl Mathias]

PROJECT REPORT

Problem Domain

The aim of this project is to create a database for USDA's National Agricultural Library (NAL).The Animal Welfare Information Center (AWIC), which is a part of NAL conducts Animal Welfare Act (AWA) workshops throughout the year for individuals who are responsible for providing information to meet the requirements of the AWA. The AWIC workshop is limited to 20 participants thus having 60 participants for a year. This workshop is conducted at NAL, Beltsville and other locations across the country. We will be creating a database to store information related to AWA workshops conducted by NAL as well as other organizations at other locations. While we are creating a database keeping workshops conducted by NAL in mind, this database can be extended and used for all the workshops conducted by various organizations/subsidiaries that are a part of USDA.

Client:
Our client is National Agricultural Library which is under United States Department of Agriculture(USDA). The Animal Welfare Information Center (AWIC) is mandated by the Animal Welfare Act (AWA) to provide information for improved animal care and use in research, testing, and teaching. For this, NAL conducts workshops at its office in Beltsville but also maintains information about similar workshops conducted elsewhere in the US.

Motivation

AWA workshops are being conducted at various locations throughout the year and at different locations. Employees from various organizations attend these workshops over the years and it is important to store this important data. Besides, attendees are given evaluation form to rate the workshop in various ways. It will be interesting to know workshops conducted by specific trainers have performed well in terms of presentation quality and overall quality. Also, the client wants to store information about all the trainers that conduct these workshops, and wants to ensure that not more than 10 attendees are registered for a workshop. A database will be the best way to store all this information and the queries that we have created will help us retrieve data in an organized manner.

Entities

Attendee: This entity stores all information related to the entity who registers for AWA workshop.

- Attendee_ID (PK)
- first_name
- last_name
- title
- city
- state

- zip code
- phone
- email
- IACUC_member_status
- principal_investigator
- experienced_db_searcher
- attendee_orgID (FK)

Organization: This entity stores all the information regarding the organizations that participate in the workshops.

- organization_id (PK)
- organization_name
- whether_host
- whether_sponsor

Workshop: This entity stores all the information related to the workshops conducted.

- workshop_id (PK)
- topic
- start_date
- end_date
- location_address
- zip_code
- workshop_type
- organization_id
- overall_rating
- presentation_quality
- Duration

Workshop_has_attendee: This entity stores mapping information between attendee and workshop.

- workshop_workshop_ID (PK)
- attendee_attendee_ID (PK)
- Reg_Time

Workshop_trainer: This entity stores mapping information between workshop and trainer.

- workshopID (PK)
- trainerID (PK)

awic_trainer: This entity stores all information related to trainers that conduct workshops.

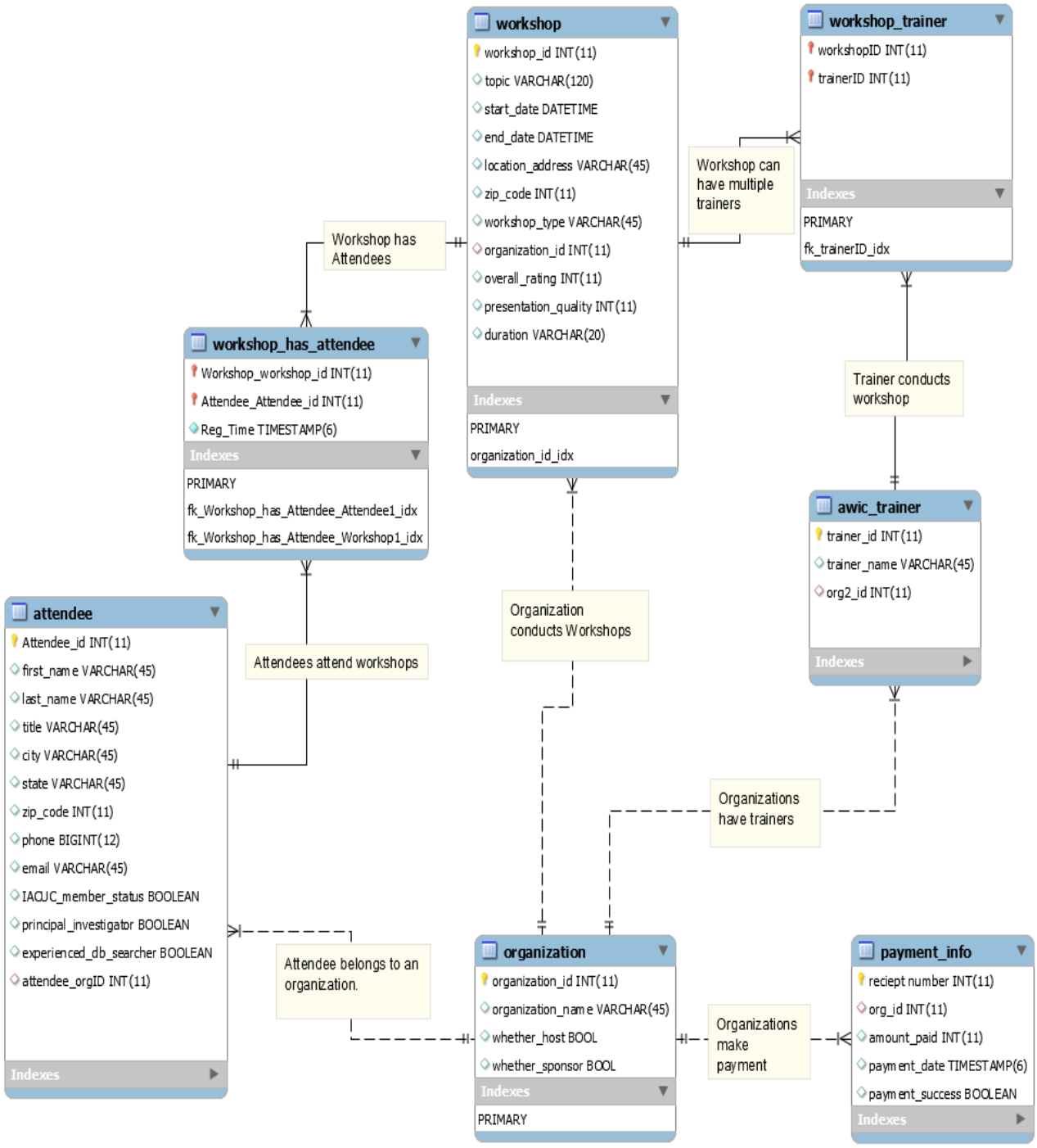
- trainer_id (PK)
- trainer_name

- org2_id

Payment_info: This entity stores all the payment information related to organizations who have paid to organizations that conduct the workshops on behalf of attendees.

- receipt_number (PK)
- org_ID
- amount_paid
- payment_date
- payment_success

Logical Design



We have tried to simplify this logical design as much as possible. During the initial phases of the project we had thought of having a separate table for waitlisted attendees. However, later on, we

thought we can have a query or a View which displays the first 20 attendees for each workshop as confirmed and the rest as waitlisted. Moreover, we have created just a single "organization" table which has all relevant details about all organizations involved whether it is the host, sponsor or merely sending its employees to attend the workshop.

The "payment" table has triggers associated with it. Whenever someone tries to enter "amount paid" as 0 or negative the system throws an error that "amount cannot be 0 or negative".

Implementation/Physical Design

After creating the logical design that is our Entity Relationship diagram in MySQL Workbench, we used the forward engineer option to create the physical design for our database.

To insert the data in our tables of our database,

Table Name	SQL Querry	Description
attendee	CREATE TABLE IF NOT EXISTS `nal`.`attendee` (`Attendee_id` INT(11) NOT NULL, `first_name` VARCHAR(45) NULL DEFAULT NULL, `last_name` VARCHAR(45) NULL DEFAULT NULL, `title` VARCHAR(45) NULL DEFAULT NULL, `city` VARCHAR(45) NULL DEFAULT NULL, `state` VARCHAR(45) NULL DEFAULT NULL, `zip_code` INT(11) NULL DEFAULT NULL, `phone` BIGINT(12) NULL DEFAULT NULL, `email` VARCHAR(45) NULL DEFAULT NULL, `IACUC_member_status` TINYINT(1) NULL DEFAULT NULL, `principal_investigator` TINYINT(1) NULL DEFAULT NULL, `experienced_db_searcher` TINYINT(1) NULL DEFAULT NULL, `attendee_orgID` INT(11) NULL, PRIMARY KEY (`Attendee_id`)) ENGINE = InnoDB	Each row contains the list of the attendee's ID, their first and last name, title, city, state, Zip_code, phone number, email address, whether the person is a current member of IACUC_member_status, whether the person is a principal_investigator,an experienced_db_searcher and a foreign key linking the attendee to the organization from which he is associated.

	DEFAULT CHARACTER SET = utf8;	
workshop_has_attendee	CREATE TABLE IF NOT EXISTS `nal`.`workshop_has_attendee` (`Workshop_workshop_id` INT(11) NOT NULL, `Attendee_Attendee_id` INT(11) NOT NULL, `Reg_Time` TIMESTAMP(6) NOT NULL, PRIMARY KEY (`Workshop_workshop_id`, `Attendee_Attendee_id`)) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;	It is a join table which links the workshop_id from the workshop table to the attendee_id of the attendee table. Each workshop can have more than one attendee.
workshop	CREATE TABLE IF NOT EXISTS `nal`.`workshop` (`workshop_id` INT(11) NOT NULL, `topic` VARCHAR(120) NULL DEFAULT NULL, `start_date` DATETIME NULL DEFAULT NULL, `end_date` DATETIME NULL DEFAULT NULL, `location_address` VARCHAR(45) NULL DEFAULT NULL, `zip_code` INT(11) NULL DEFAULT NULL, `workshop_type` VARCHAR(45) NULL DEFAULT NULL, `organization_id` INT(11) NULL DEFAULT NULL, `overall_rating` INT(11) NULL DEFAULT NULL, `presentation_quality` INT(11) NULL DEFAULT NULL, `duration` VARCHAR(20) NULL, PRIMARY KEY (`workshop_id`)) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;	The workshop table has the workshop_id as the primary key, the topic, start date and end date, the location address, Zip_code, workshop_type, organization_id, overall_rating, presentation_quality and the duration.
workshop_trainer	CREATE TABLE IF NOT EXISTS `nal`.`workshop_trainer` (`workshopID` INT(11) NOT NULL, `trainerID` INT(11) NOT NULL,	This join table has workshopID, trainerID.

	PRIMARY KEY (`workshopID`, `trainerID`)) ENGINE = InnoDB;	
payment_info	CREATE TABLE IF NOT EXISTS `nal`.`payment_info` (`reciept number` INT(11) NOT NULL, `org_id` INT(11) NULL DEFAULT NULL, `amount_paid` INT(11) NULL DEFAULT NULL, `payment_date` TIMESTAMP(6) NULL DEFAULT NULL, `payment_success` TINYINT(1) NULL DEFAULT NULL, PRIMARY KEY (`reciept number`)) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;	This table has reciept number as primary key along with org_id, amount_paid, payment_date, payment_success.
awic_trainer`	CREATE TABLE IF NOT EXISTS `nal`.`awic_trainer` (`trainer_id` INT(11) NOT NULL, `trainer_name` VARCHAR(45) NULL DEFAULT NULL, `org2_id` INT(11) NULL DEFAULT NULL, PRIMARY KEY (`trainer_id`)) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;	This table has the details of the trainers who are conducting the workshop. The trainer_id is the primary key. This table also has the trainer_name along with org2_id as the foreign key linking this table with the organization table.
organization	CREATE TABLE IF NOT EXISTS `nal`.`organization` (`organization_id` INT(11) NOT NULL, `organization_name` VARCHAR(45) NULL DEFAULT NULL, `whether_host` TINYINT(1) NULL, `whether_sponsor` TINYINT(1) NULL, PRIMARY KEY (`organization_id`)) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;	This table records all the detailsof the organizations associated with the workshops – those who are the host as well as those which are attending it. It has the organization_id as the primary key, along with the organization_name and whether the organization is a host or a sponsor (boolean).

- Entered data from sample test data provided by the client using Microsoft Excel and saved it n csv format.

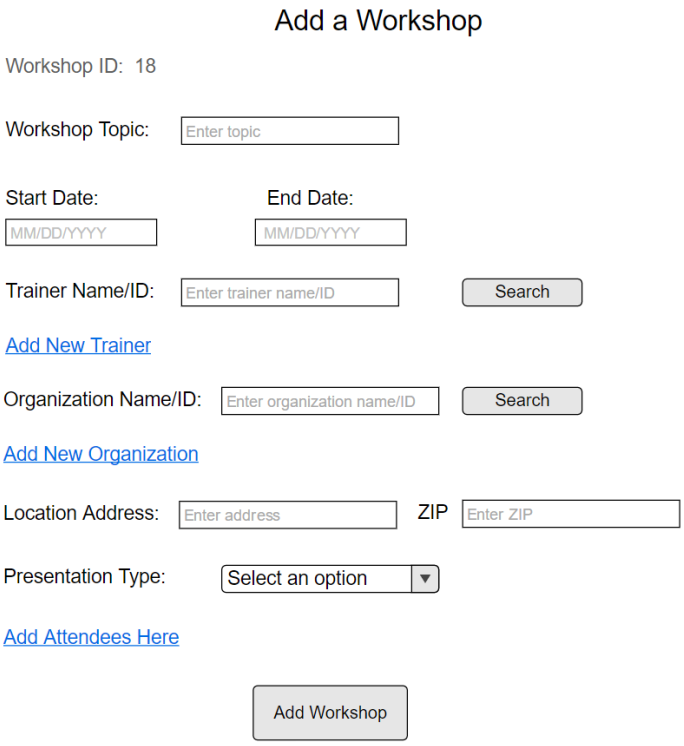
- Created a separate csv file for every table in our database and entered values accordingly.
- Executed SELECT command for retrieving all the rows from Attendee table using the following query SELECT * FROM nal.attendee;
- We used the 'Import records from an external file' option in MySQL Workbench and selected the respective csv file to import the data.
- Clicked on 'Apply' which then gives a summary of all the INSERT queries for all the rows in the Attendee table. Finally clicked on 'Finish' which enters all the data from our csv to the database in MySQL.
- Repeated all the above steps for other tables too.

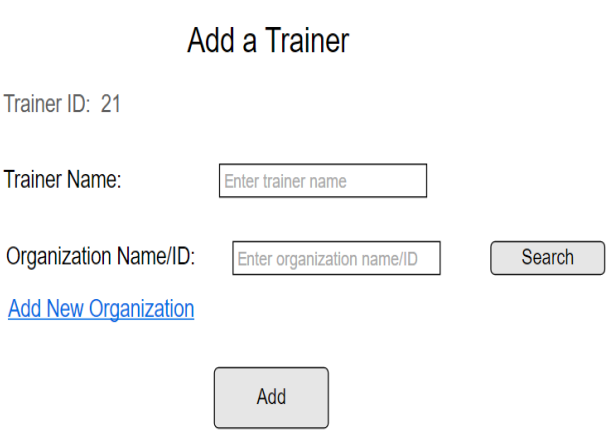
Sample Data

The sample data was provided by the client. It was kept as realistic as possible with dummy values for email IDs and phone numbers for all the attendees for the purpose of confidentiality. We stored all the data in separate csv files for all the entities.

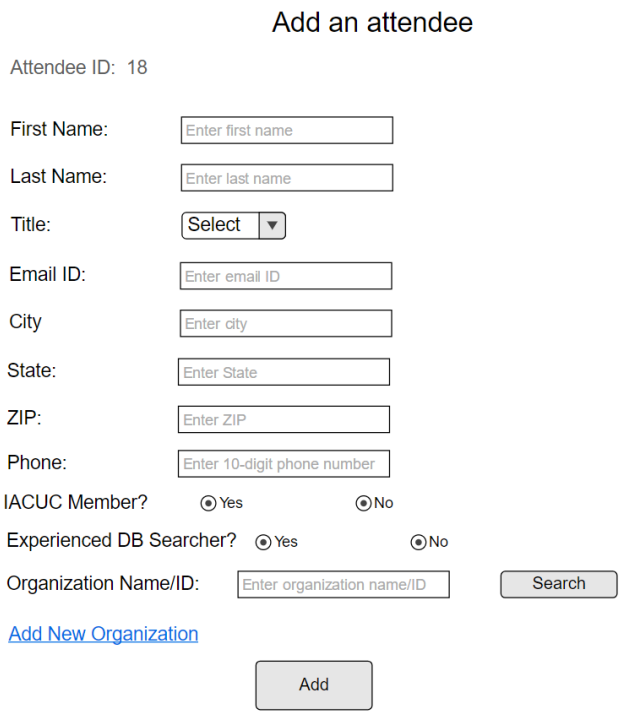
User Interaction (Forms):

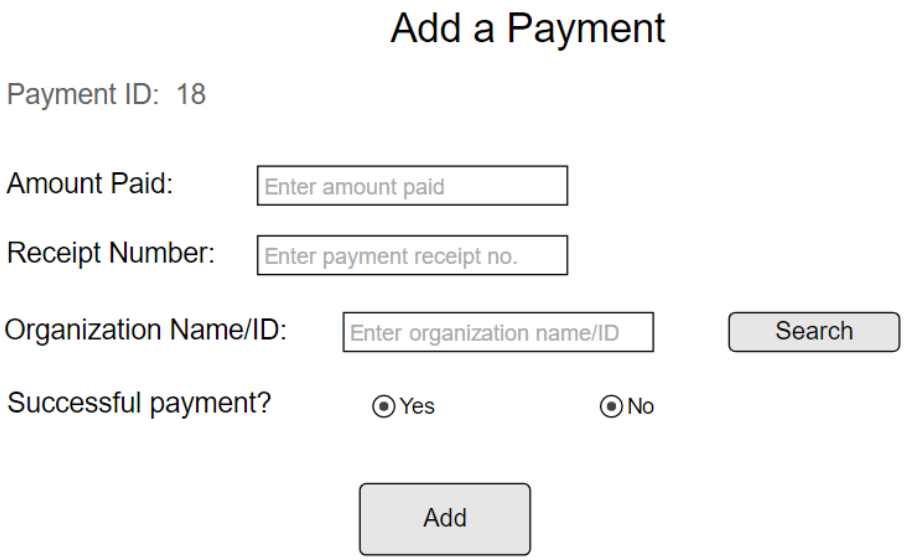
We have come up with some mockups for the forms that can be used to enter the data into our database using a user interface. The interface can be a standalone application, or an application deployed over the world wide web. The NAL staff are the ones that would be updating the forms based on the information received from the various organizations that conduct workshops, and also send attendees to attend. The below mockup shows the form for adding a new workshop, which is the most important entity in our database.


Add a Workshop
Workshop ID: 18
Workshop Topic:
Enter topic
Start Date:
End Date:
MM/DD/YYYY
MM/DD/YYYY
Trainer Name/ID:
Enter trainer name/ID
Search
Add New Trainer
Organization Name/ID:
Enter organization name/ID
Search
Add New Organization
Location Address:
Enter address
ZIP
Enter ZIP
Presentation Type:
Select an option
Add Attendees Here
Add Workshop


Add a Trainer
Trainer ID: 21
Trainer Name:
Enter trainer name
Organization Name/ID:
Enter organization name/ID
Search
Add New Organization
Add


Add an Organization
Organization ID: 200
Organization Name:
Enter organization name
Whether host?
Yes
No
Whether sponsor?
Yes
No
Add





As shown above, there are basically four important forms that would help the user populate data into the database: Organization, Workshop, Payment, Trainer and Attendee.

Queries:

We used queries to extract information that might make our client's day to day work and analysis easier. Below is a brief summary of what each of out query does:

Query1:

Workshop ratings are really important for our client to assess how much of use the attendees are getting from the conducted workshops. One of the important measures according to them to measure the same is how the attendees have rated the quality of the presentation. This would help our client to decide if there are any changes that need to be done (i.e. change the trainers/inform the organizations about the poor presentation quality). This query hence returns a sorted list of all the workshops in the order of the presentation quality rating received from the attendees.

Query2:

As mentioned above, various organizations pay NAL in order to conduct workshops. The organizations are responsible for the attendees fees as well, and hence would be paying the same as well. Our client wanted an easy way to find out which organization is paying the highest amount at present, and also an ordered list of the total amount paid by each organization. This query returns an ordered list of all the organizations in connection with NAL and the total amount they are paying.

Query3:

As mentioned above, each of the organization that conducts a workshop is being rated overall (i.e. based on presentation, trainer knowledge, and so on) by the various attendees. This query returns the average overall rating of each workshop that is being conducted by the various organizations.

Query4:

This query gives us the total number of attendees that have registered for each workshop. While we have a separate procedure written to calculate how many people are on the waitlist, this query returns details about all the attendees who have registered (including waitlist) so far for the workshop. The results are grouped by the unique workshop ID.

Query5:

As mentioned above, there is a limit of only ten students who are eligible to attend. Rest of the students would be automatically put on the waitlist with the help of one of our stored procedures (details given later). This query helps us to retrieve the first ten attendees that the organization registered. We filter these out by using the timestamp data in our database. This data will be useful to check how many people to actually contact regarding the various details about the workshops.

Query6:

As we have mentioned above, attendees are sent through organizations to participate in the workshops. Our client wanted a way to find out the total number of attendees each organization is sending. They would further like to use this data to do a comparative analysis across various months of the year, in order to decide when to hold what workshops. This query returns a list of organizations that are conducting workshops, and the total number of registered applicants for each (including waitlist).

Query7:

One of the important things our client needs to know about the attendees is whether an attendee is experienced with databases or not. Our last query aims to find out the ratio of the number of attendees who are experienced in databases to the number of them who aren't. We are retrieving this data for every workshop that is being conducted at the given time. This would be very helpful for the faculty to design the workshop material, since it gives insights about whether most of the attendees are already comfortable with database concepts.

Query	Requirement A	Requirement B	Requirement C	Requirement D
Query1	X	X		X
Query2	X	X	X	X
Query3	X	X	X	X
Query4	X	X	X	X
Query5	X	X	X	X
Query6	X	X	X	X
Query7	X	X	X	X

Trigger and Stored Procedure:

In addition to the queries mentioned before, we have also added a trigger and a stored procedure to our database in order to make our client's work easier. While the trigger takes care of the payments section, the procedure helps in handling the waitlist.

Trigger – Do not allow insert for invalid payment entries:

The trigger we have come up with is used to disallow any entries in the payment table if the payment amount is negative, or zero. This is basically to keep the data secure, and ensure that no payment data is unnecessarily modified or wrongly inserted. Since payment table is a very important part of our database and for the client, we wanted to add some sort of security to it and prevent it from being mishandled or prone to mistakes. The trigger is added in the SQL file attached with this report (along with the queries).

Stored Procedure - Find out who are on the wait list:

One important requirement that the client communicated to us is the maximum number of attendees that would be allowed to attend a workshop. At present, a workshop can have only ten attendees attending it. However, the client wanted no limit on the number of the students that register. The seats would be given on the basis o f when the organizations requested to register an attendee (timestamp). Hence it is important to also notify people who would not be able to attend the workshop, and also keep their contact details in case anyone in the main list cancels registration. We came up with a stored procedure that, at the point of execution gives us the list of attendees who are on the waitlist (sorted based on the timestamp when they were registered by their respective organizations). The procedure takes in the workshop ID as the input, and returns the waitlist for that particular workshop at that particular time. The procedure is added in the SQL file attached with this report (along with the queries).

Lessons Learned

We learned a lot about database normalization and creation of Entity Relationship diagrams using MySQL Workbench. Besides, initially in the first round robin our requirements looked simple and our ER diagram had the basic entities. However, after subsequent client meetings, we modified our ER diagram based on our requirements. Designing a database while considering every requirement is a very important skill and we have learnt this during our project. After two round robins, we were able to enhance our ER diagrams and understand our Workshop database requirements well.

Future Improvements

We can create a database connectivity for the existing website which has the AWIC workshop registration form displayed on it. After clicking on submit, the database entity "Attendee" will be populated with all the information entered in the registration form. We can expand upon the feedback column for the workshops. Perhaps a separate table that records the feedback of each and every attendee.

This database can be used in conjunction with a web application which can further make the process of creating and adding the workshop easier and more intuitive.