Case Study: Friends of the Aquarium

Use-Cases

Consider a volunteer organization (Friends of the Aquarium). Create a conceptual database

design for a Volunteer Schedule using the following information.

* The person on duty in the volunteers' office receives request for volunteer services from

the aquarium's administrative office. Some of the jobs are regularly scheduled (for

example, staffing the gift shop and admissions desk). Others are ad hoc, such as a

request from a teacher to bring her class for a tour.

* The volunteer doing the scheduling checks the list of volunteers to see who is trained to

do the job requested. Each volunteer's information is recorded in a data file that

contains the volunteer's contact data along with the volunteer's skills. The volunteer

information also includes an indication of when that person is available to work.

* The volunteer doing the scheduling searches for those people who have the required

skills and have indicated that they are available at the required time. Most volunteers

work on a regularly scheduled basis either at the admissions desk or in the gift shop.

However, for ad hoc jobs, the person doing the scheduling must start making telephone

calls until someone who is willing and able to do the job is found.

* The volunteer organization would like to keep track of the work record of each

individual volunteer. There is a volunteer recognition lunch once a year, and the

volunteer organization would like to find an easy way to identify volunteers who have

put in an extra effort so they can be recognized at that event. The volunteer

organization would also like to identify volunteers who rarely participate, as well as

people who make commitments to work but do not show up.

Please be sure that your attributes have data types associated with them. Here's a starting tip:

The administrator needs to track volunteers, their schedules, and what jobs they can do. Each

of these entities has attributes which you can either find in the text above, or in the case of

schedule, assume a Monday-Friday week with time slots from 9am to 4pm and no one works

from 12pm to 1pm (thus this one hour does not appear in the list of possible time slots).

Describe your entities and attributes below, and specify the data type for each attribute. If an

attribute has a fixed list of values (like Regular, AdHoc, or days of the week), then list all

possible values. Unlimited attributes like Names or TotalHours do not need examples or

values listed, but you do need to identify the data types.

Volunteers (from the use case):

Volunteers are those who are willing to help at the aquarium's shop and with other

tasks that need to be completed and their skills to get the job done.

The attributes involved are

* VolunteerID: integer
* VolunteerFirstName : string
* VolunteerLastName: string
* VolunteerPhoneNumber:Long="10"
* VolunteerAddress : String
* VolunteerSkills : String
* NumberOfDays: integer
* AvailableTime : DateTime(9am to 4pm)
* Volunteer Work record : String

Schedule (from the use case):

This object keeps track of all the jobs that need to be completed in a table. It also

contains information on the volunteer who is currently working on the Job and whether

whether the volunteer had shown up or not.

* SchedulelD:int
* ScheduleHours:DateTime
* lobID:int
* VolunteerID:int
* VolunteerShownUp:Boolean

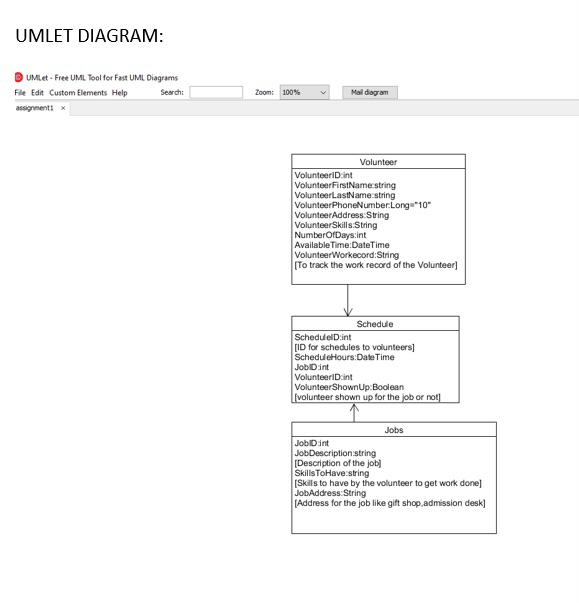
Jobs (from the use case):

It describes the Job description that the aquarium's administration department has

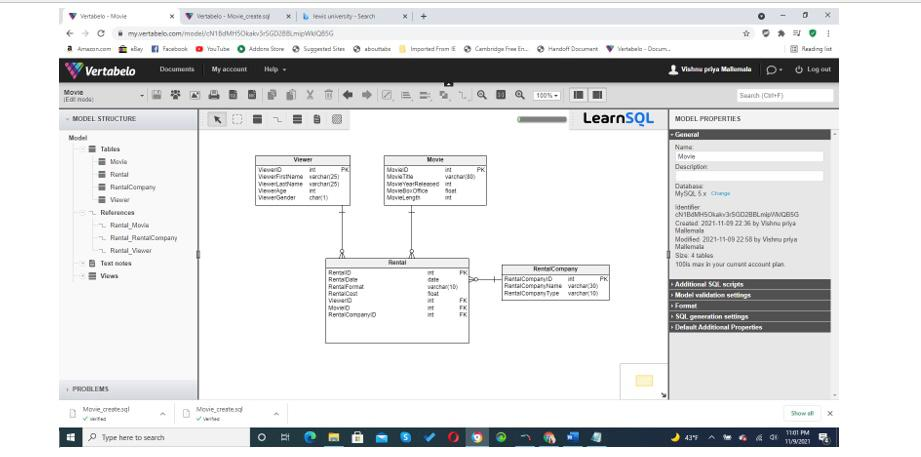
requested. It comprises a description of the job to be done as well as the qualifications that the

volunteers should possess it in order to execute it.

* JobID:int
* JobDescription:string
* SkillsToHave:string
* JobAddress:String



[**https://github.com/bharath4568/umlet.git**](https://github.com/bharath4568/umlet.git)

****

- Created by Vertabelo (<http://vertabelo.com>)

- Last modification date: 2021-04-03 04:58:06.23

--tables

- Table: Movie

CREATE TABLE Movie (

MovielD it NOT NULL AUTO\_INCREMENT,

MovieTitle varchar(80) NOT NULL,

MovieYearReleased it NOT NULL,

MovieBoxOffice float NOT NULL,

MovieLength int NOT NULL,

CONSTRAINT Movie pk PRIMARY KEY (MovielD)

);

- Table: Rental

CREATE TABLE Rental (

RentalID int NOT NULL AUTO\_ INCREMENT,

RentalDate date NOT NULL,

RentalFormat varchar(10) NOT NULL,

RentalCost float NOT NULL,

ViewerID int NOT NULL,

MovielD int NOT NULL,

RentalCompanyID int NOT NULL,

CONSTRAINT Rental pk PRIMARY KEY (RentalID)

);

- Table: RentalCompany

CREATE TABLE RentalCompany (

RentalCompanyID int NOT NULL AUTO\_INCREMENT,

RentalCompanyName varchar(30) NOT NULL,

RentalCompanyType varchar(10) NOT NULL,

CONSTRAINT RentalCompany pk PRIMARY KEY (RentalCompanyID)

);

Table: Viewer

CREATE TABLE Viewer (

ViewerID int NOT NULL AUTO\_INCREMENT,

ViewerFirstName varchar(25) NOT NULL,

ViewerLastName varchar(25) NOT NULL,

ViewerAge int NOT NULL,

ViewerGender char(1) NOT NULL,

CONSTRAINT Viewer pk PRIMARY KEY (ViewerID)

);

- foreign keys

- Reference: Rental Movie (table: Rental)

ALTER TABLE Rental ADD CONSTRAINT Rental Movie FOREIGN KEY Rental Movie (MovielD)

REFERENCES Movie (MovielD);

- Reference: Rental RentalCompany (table: Rental)

ALTER TABLE Rental ADD CONSTRAINT Rental\_RentalCompany FOREIGN KEY Rental\_RentalCompany

(RentalCompanyID)

REFERENCES RentalCompany (RentalCompanyID);

Reference: Rental Viewer (table: Rental)

ALTER TABLE Rental ADD CONSTRAINT Rental Viewer FOREIGN KEY Rental Viewer (ViewerID)

REFERENCES Viewer (ViewerID);

End of file.