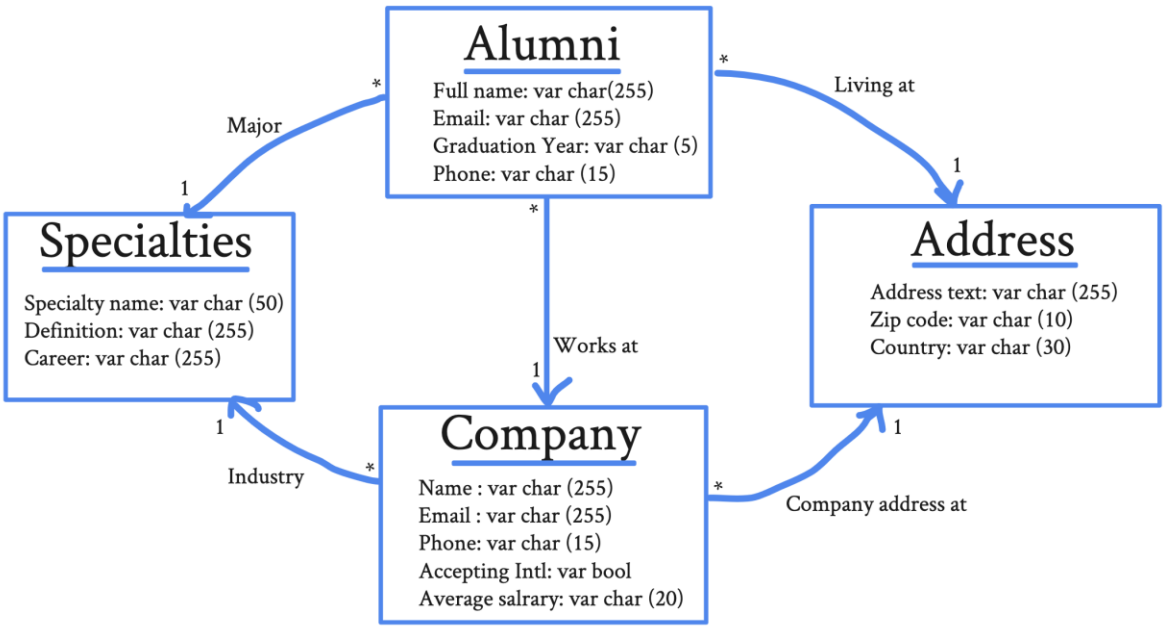
Model Database Design Project: Alumni Network Database

Duc Nguyen, Spring 2019

This is a simple database design for an alumni network that stores data and information about the alumni, companies, their addresses and specialties. The class diagram is shown below:

I am using the regular arrowheads to highlight the foreign key relationships.



The given relationships are adequate to represent all of the information needed for the application: the user can get information about alumni, where he/she works at, where he/she lives at, what did he/she studied as a major, what does his/her company do, where does the company locate at, etc. The database also gives full information in each and every table:

- Alumni: the full name, email are each set at 255 characters, with a checking system for the format of the email. Graduation year should be only 4 digits, and 15 characters for phone number. Deleting an alumnus will be cascaded through all their foreign keys.
- Address: address text (which combines street address with city, state, etc. for flexibility). Zipcode is set at 10 characters and Country is set at 30. Zip code and Country can later be used to find particular information like companies/ alumni network/ career opportunities about an area.
- Company: Name of the company and email are each set at 255 characters, with a checking system for the format of the email. 15 characters for phone numbers, a Boolean variable to see if the company is accepting internationals or not, and 20 characters for the company's average salary.
- Specialties: Name should only take 50 characters with a definition (description) of 255 characters. Gives information about the field/industry/major with its description and a career opportunities and jobs of 255 characters.

Note: An attempt is made to check that a customer has a valid email address (the "like" operator here says that the email field has to match the pattern "_%@_%._%", which is three groups of one or more characters each, separated first by an at sign and then by a dot; the pattern "_%" means "an arbitrary character followed by zero or more additional characters"); however, the proper way to check for a valid email address is probably just to send a verification email….

There are no redundant relationships in the diagram. Although there are several places where multiple paths connect the same tables, in each case they give different information:

- Alumni -> Company -> Address: gives the information about the location/ address of the company where he/she works at, which is not related to his/her living address.
- Alumni -> Company -> Specialties: gives the information about what industry is the alumni's company is specialized in, which is not related to his/her education or major when he/she was still in school (people may end up working in a different field/ industry than what they chose to study in school).

Here is the SQL schema produced from the above diagram:

```
create table Alumni (
        alumniID integer not null,
        fullname varchar(255) not null,
        email varchar(255) not null,
        gradyear varchar(4) not null,
        phone varchar(15),

        primary key (alumniID),
        foreign key (livingaddr) references Address
                on delete cascade,
        foreign key (workingcompany) references Company
                on delete set cascade,
        foregin key (major) references Specialties
                on delete set cascade,
        check (email like '_%@_%._%')
        );
```

```
create table Company (
        companyID integer not null,
        name varchar(255) not null,
        email varchar(255) not null,
        phone varchar(15) not null,
        acceptingintl bool not null,
        avgSalary varchar(20) not null,

        primary key (companyID),
        foreign key (industry) references Specialties
                on delete cascade,
        foreign key (companyaddress) references Address
                on delete cascade,
        check (email like '_%@_%._%')
        );

create table Address (
        addressID integer not null,
        addrtext varchar(255) not null,
        zipcode varchar(10) not null,
        country varchar(30) not null,

        primary key (addressID)
        );

create table Specialties (
        specialtyID integer not null,
        name varchar(50) not null,
        definition varchar(255),
        career varchar(255),
```

```
primary key (specialtyID)
);
```

Four examples of useful queries:
Given a customer id, retrieve lists of all previous shipping addresses and credit cards that they have used.

- Given a company ID, retrieve a list of all the alumni who is currently working at that specific company.
- Given a specialty ID (Major), retrieve a list of all the alumni who studied that major while in school with their contact information.
- Given an alumni ID, retrieve a list of all companies that he/she has worked for.
- Given a specialty ID (Industry), retrieve a list of all the companies in the industry that have alumni working at.

One possible view could be used to restricted access to my database for some users is maybe a feature that only faculties/ staffs can see personal information (emails, phone, …) of alumni, but not all students and others who have access to the database can have this personal information. In order to have access to alumni's contact, user needs permission from the system/ admin.