Lab 3: E-R Design and Relational Database

50 points

In lecture 2 you were introduced to the three phases of database design: conceptual design, logical design, and physical design. Usually we are more concerned with conceptual design and logical design because the DBMS often takes care of physical design (e.g., how the database are actually stored in a particular machine) for us. After *conceptual design phase*, we should have the *Entity-Relationship (E-R) diagram*. In the following *logical design phase*, we need to turn the E-R diagram into a set of relations (tables) which are ready to be stored in a DBMS, and at the same time, maintain the entities and relationships in the E-R diagram by involving constraints (e.g., primary keys, foreign keys).
In this lab you will be using *pgModeler*, a graphical database designer, or an equivalent web-based drawing tool for database design and modeling. For exercise, the COMPANY database example is used to illustrate the process of database design. After you observe and practice with this exemplary database design process, you are required to design a database independently (see *Lab Assignment* section). As a start point, the data requirements for the COMPANY database are summarized as:

- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- We store each employee's name, social security number, address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee (who is another employee).
- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date, and relationship to the employee.

Objectives

The goals for you to take away from this lab are:

- To reinforce your understanding of Entity-Relationship (E-R) model;
- To understand the process of designing a relational database for an application based on ER model;
- To practice formal database design techniques;

Lab Data

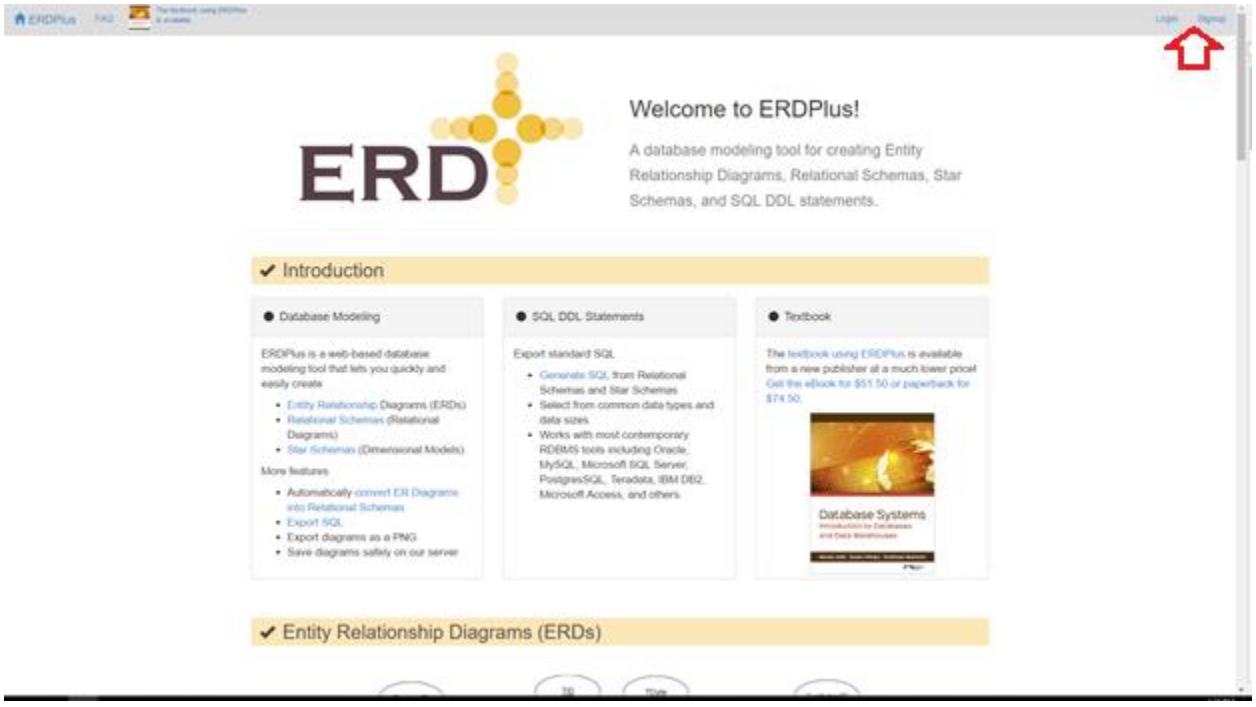
No data required for this lab.

Part I: Exercise Tutorial

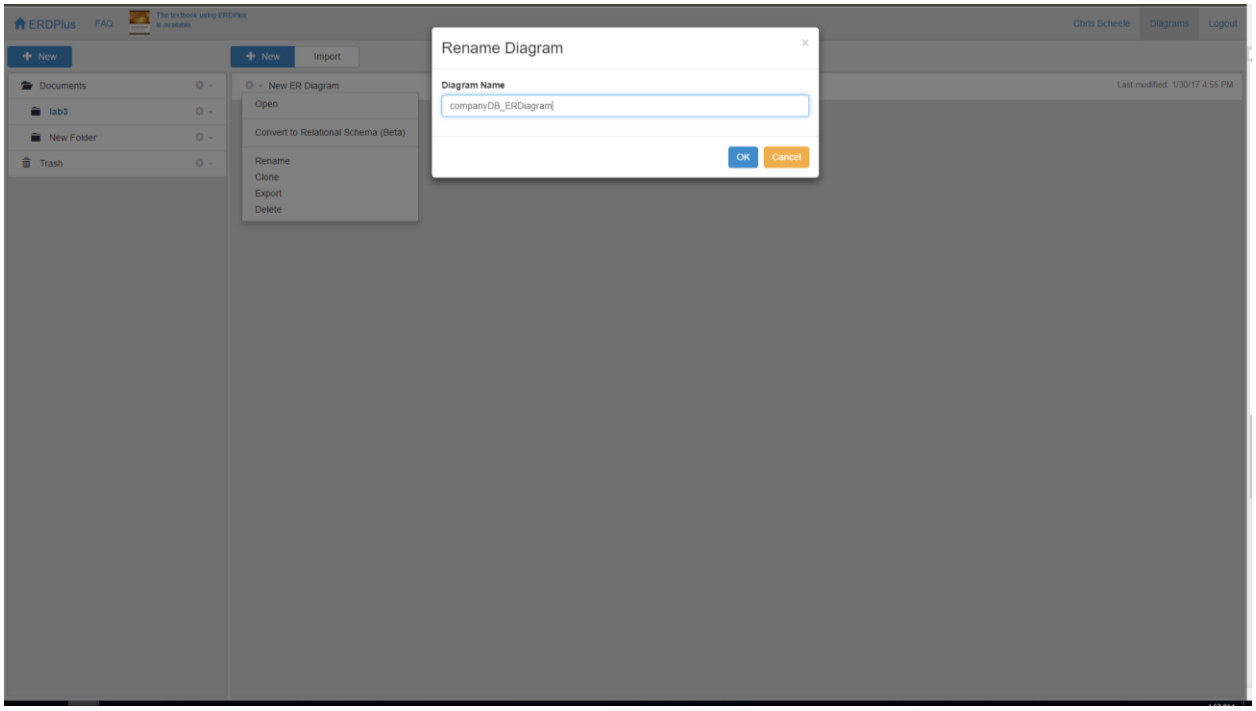
1. Starting with *ERDPlus*

Originally, this lab was written for *pgModeler*. However, due to a change in business model, this software is no longer free unless you install from source. To accommodate for the change, the exercise will be performed using *ERDPlus*, a web-based database modeling tool that allows you to create both ER and Relational Schema Diagrams and then generate SQL to load the schema into Postgres. Another option for this lab is to create the diagrams manually through a program like draw.io.

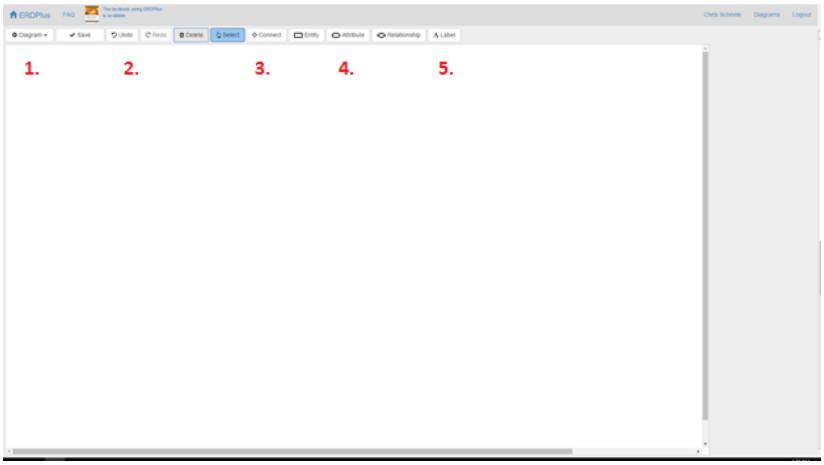
➔ Navigate to the ERDPlus website and create a login.



➔ Once an account is created, select *Diagrams* in the upper right corner. This window is the storage interface for all of your diagrams. Folders can be created in the menu on the left and diagrams can be created in the center pane. Add a new folder and rename it **lab3**. Next let's add a new ER-Diagram and give it a clever name like **companyDB_ERDiagram**. To add this diagram to the lab3 folder, simply drag the file.



➔ Now let's create the diagram! Clicking on the file will bring you to a new canvas page. The menu bar is fairly intuitive, but let's go over it anyways.



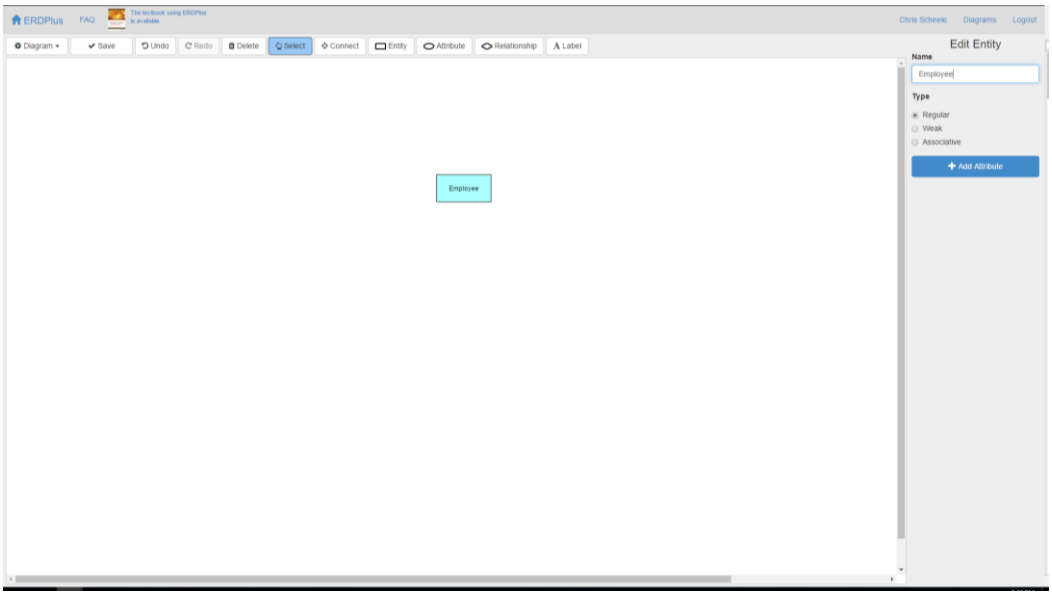
Each highlighted widget has special functions as follow:

1). **Diagram:** Allows you to export an image of your canvas (good for lab submissions), export the file of the diagram, or change your canvas size.
2). **Basic User functions:** The standard user controls for manipulating the data
3). **Connect:** The function to connect attributes or entities together.
4). **Entity, Attribute, Relationship:** Used to create a new entity, attribute, or relationship. The select function should be used to edit any already created elements.
5). **Label:** Add additional text to your diagram.

2. Adding Entity Types

➔ To add an entity type, select *Entity* from the menu and draw the entity on the canvas

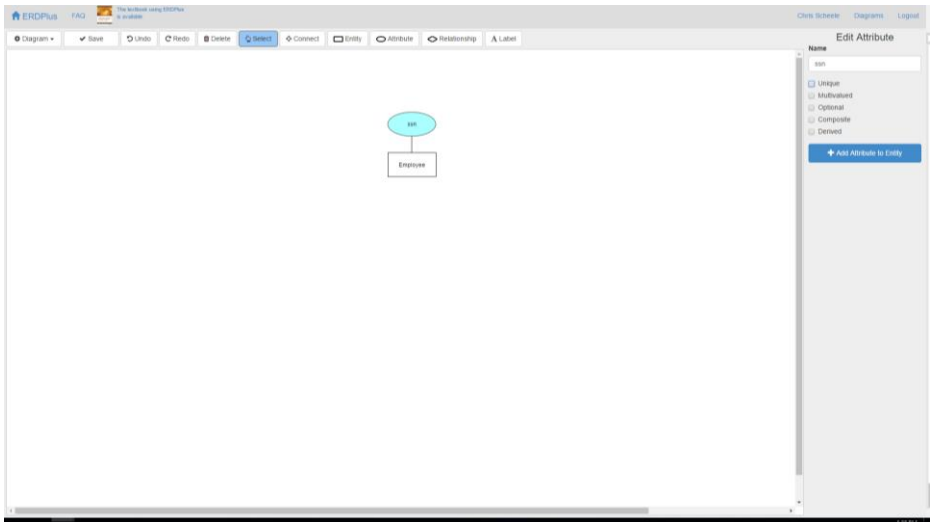
➔ Select the newly created entity and edit the name in the editing panel on the right side of the screen. Put the "employee" as the Entity name. The Type will be Regular.



3. **Adding Attributes**

➔ There are two ways to add attributes to an entity. First, you can use the "+ Add Attribute" button in the editing panel. The second way is to select *Attribute* from the menu, edit the information and then use *Connect* to attach it to an entity.

➔ In this case, let's first add employee social security number as the first attribute and call it **ssn**. For the time being, don't worry about the check boxes.



➔ Following similar steps, we can add all attributes for the employee entity. We'll revisit this table when we create the relational schema to input the data type and length information.

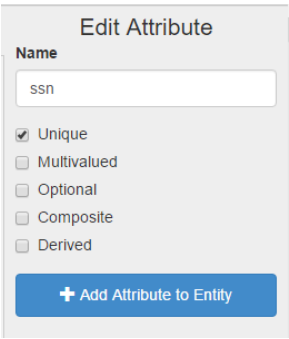
employee			
Attribute name	*Data type*	*Length*	*Description*
fname	varchar	15	First name
minit	varchar	1	Middle name initial
lname	varchar	15	
ssn	char	9	XXX-XXXX-XX
bdate	date		YYYY-MM-DD
address	varchar	50	Home address
sex	char	1	"F" or "M"
salary	decimal	Float	salary
superssn	char	9	XXX-XXXX-XX
dno	int		Department number(since there are only limited departments in a company, a smallint data type is enough)

Please refer here for description for a list of data type supported by PostgreSQL
http://www.postgresql.org/docs/9.3/static/datatype.html

4. **Specifying constraints: primary keys**

After adding all the attributes to the employee entity, we can specify the primary key, which is used to identify objects for the entity. In our case, the attribute "ssn" is our primary key.

➔ Select the ssn attribute and check the box next to Unique in the editing menu



5. Create all other entity types

Following the similar steps to create all other entity types, including *department*, *dept_locations*, *project*, *works_on* and *dependent* for the COMPANY database (primary keys are underscored).

Attribute name	*Data type*	*Length*	*Description*
dname	varchar	25	department name
dnumber	int		Department number
mgrssn	char	9	Manger ssn

mgrstartdate	date		Manger start date

Question 1 (5 pts): Why the data type for dname should be varchar? Why do we set the length as 25? How about dnumber?

Attribute name	*Data type*	*Length*	*Description*
dnumber	int		Department number
dlocation	varchar	15	Department location (A department (e.g., sale) may locate at multiple locations)

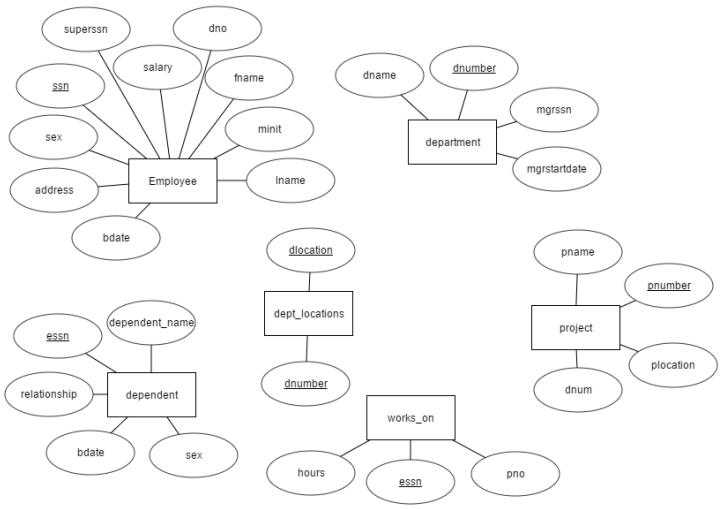
Question 2 (5 pts): Why do we need to use *dnumber* and *dlocation*, in combination, as primary key here?

Attribute name	*Data type*	*Length*	*Description*
pname	varchar	25	Project name
pnumber	int		Project number
plocation	varchar	15	Project location
dnum	int		Department number

Attribute name	*Data type*	*Length*	*Description*
essn	char	9	Employee ssn
pno	smallint		Project number
hours	float		Project location

Attribute name	*Data type*	*Length*	*Description*
essn	char	9	Employee ssn
dependent_name	varchar	15	department name
sex	char	1	“F” or “M”
bdate	date		YYYY-MM-DD
relationship	varchar	8	Relationship to the empolyee

We can also specify related primary keys for those entities as section 4. Then you will get E-R model as follows:



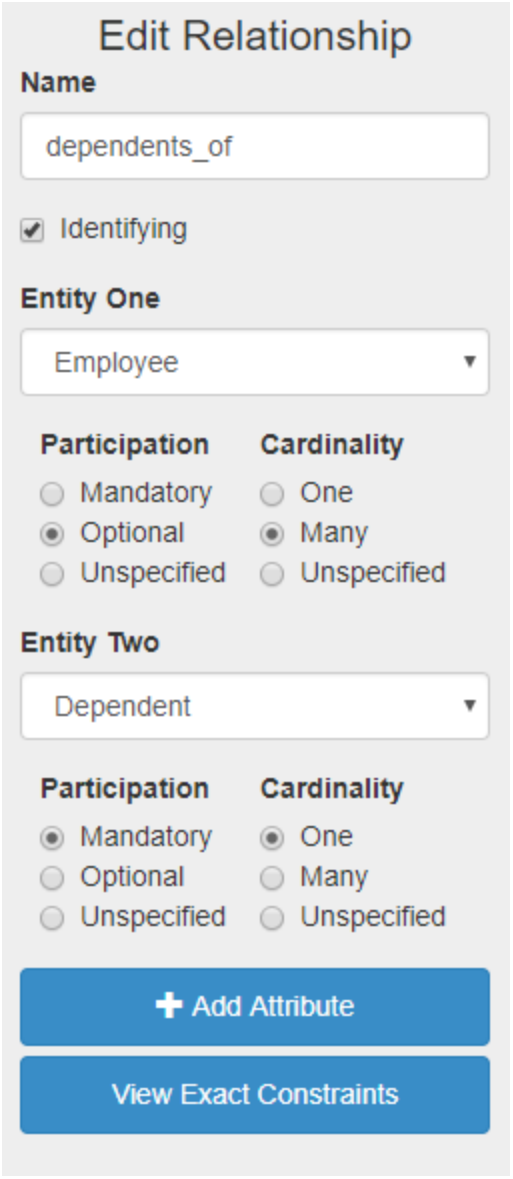
6. **Specifying constraints: foreign keys and relationships**

Three types of relationships are supported in ERDPlus: one-to-one, one-to-many, and many-to-many. In the case of the COMPANY database, we create the following relationships:

- One identifying relationship from *employee* to *dependent*.
- Two many-to-many relationships, one from *employee* to *project* and the other from *department* to *dept_locations*, and
- Four non-identifying relationships: from *employee* to *department* (one-to-one for manages), from *department* to *employee* (one-to-many for works for relationship), from *employee* to *employee* (one-to-many for supervisor/supervisee relationship), and from *department* to *project* (one-to-many for the controls relationship).

Now, let's add the identifying relationship from *employee* to *dependent*.

➔ Select the *Connect* function in the main menu bar. Then select employee and drag the connection line to dependent. A new relationship will connect the two. Let's rename it to something specific, **dependents_of**. In the editing panel, let's add the cardinality. Thinking about the relationship, an employee may have zero, one or many dependents. As for a dependent, they must be related to one and only one employee. Therefore all dependents have mandatory participation in the relationship. Since the existence of dependent entity depends on employee entity, this relationship is also an identifying relationship. Here's what that looks like in the editing panel:

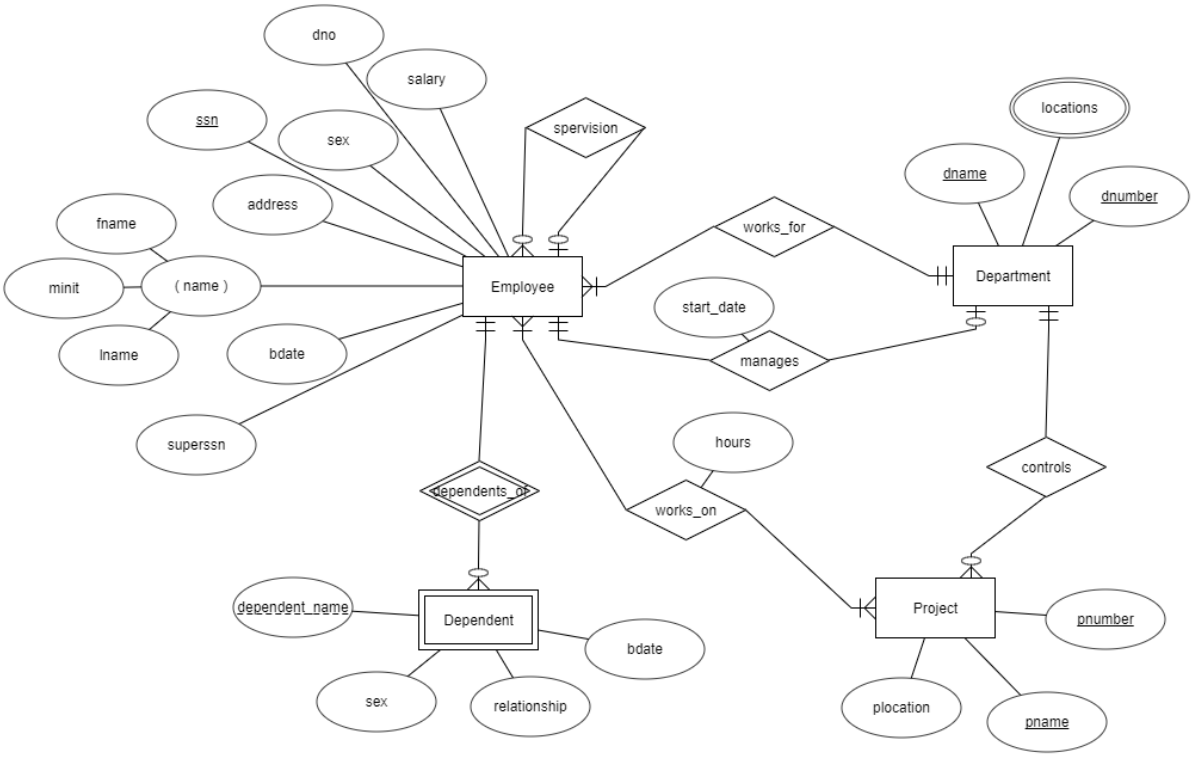


Following similar steps add the following foreign keys. We'll revisit this table when we create the relational schema to input the foreign and primary key information.

Foreign key	*Table*	*Referenced column*	*Referenced Table*	*Relationship*
essn	dependent	ssn	employee	one-to-many (one employee may have several dependents)
superssn	employee	ssn	employee	one-to-many for supervisor / supervisee relationship
dno	employee	dnumber	department	one-to-many (one department may have many employees)
mgrssn	department	ssn	employee	one-to-one (one department only has one manager)

dnum	project	dnumber	department	one-to-many (one department controls many projects)
essn	works_on	ssn	empolyee	one-to-many (one employee works on several projects)
pno	works_on	pnumber	project	one-to-many (one project includes many employees)
dnumber	dept_locations	dnumber	department	one-to-many (one department has many locations)

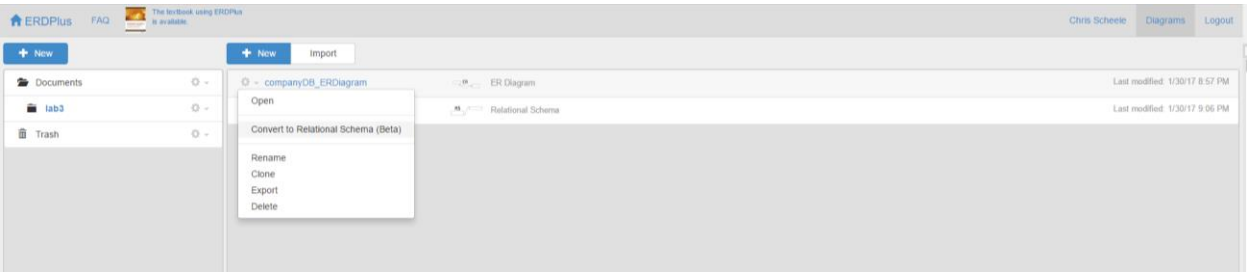
Up to now, you E-R model should look like as follows:



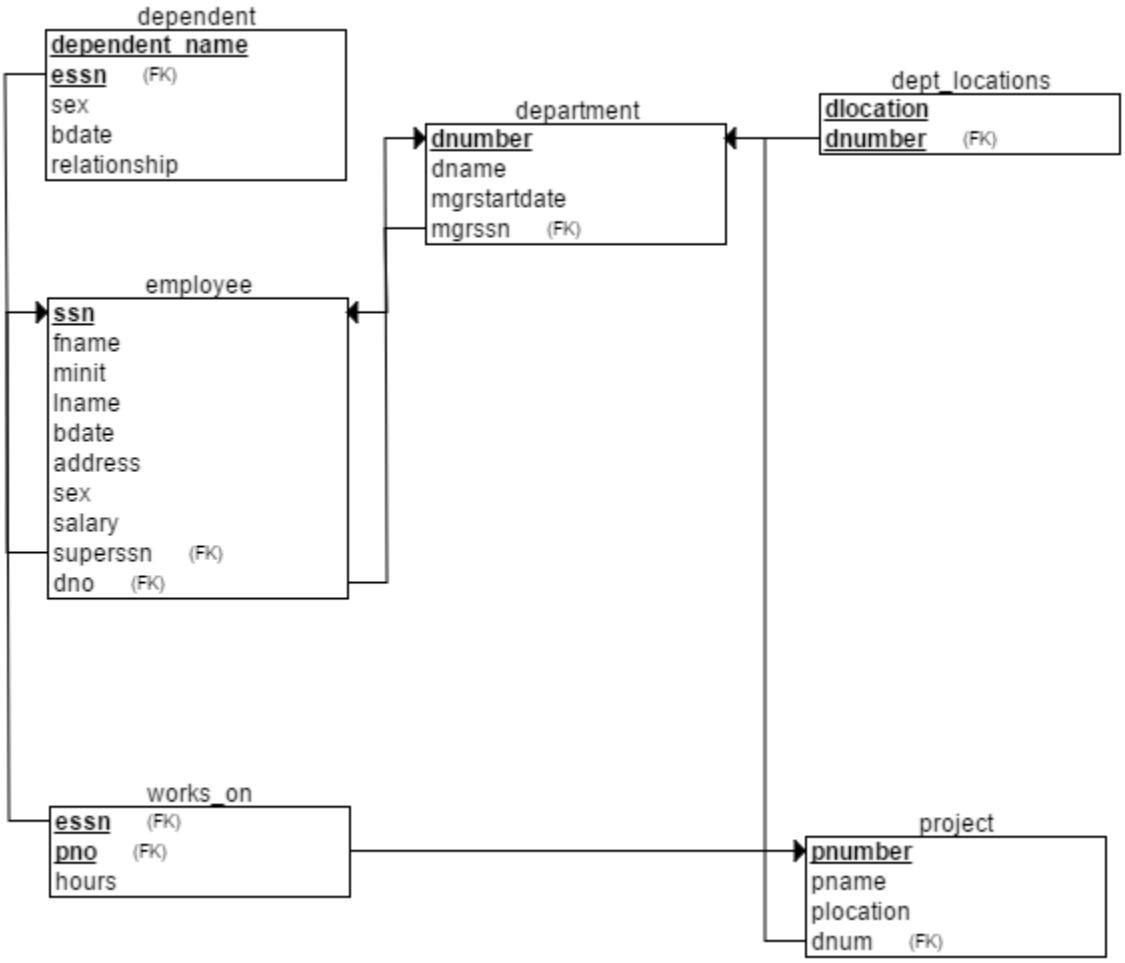
7. **Creating the Relational Schema from the ER-Diagram**

The next task is to create a Relational Schema which we will then use to import into Postgres. In the relational schema, we will specify datatypes and keys for the attributes and relationships we just created in the ER-Diagram. Luckily, ERDPlus has functionality (in beta) that can convert an ER to Relational

Diagram. To do this, select Diagram in the upper right corner to get back to your folders. Then select the cog on *companyDB_ERDiagram* and select **Convert to Relational Schema**. Now you should have a new Relational Schema file which you can rename to **companyDB_relationalDiagram**.

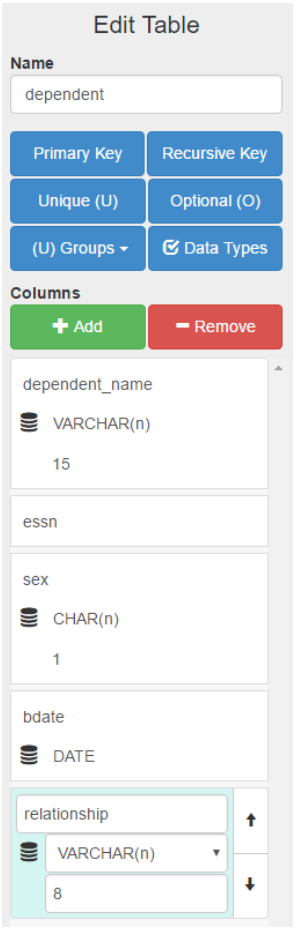


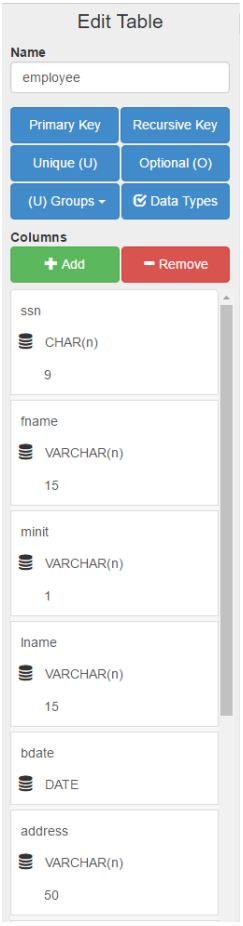
Now open up the diagram and see how everything looks. Primary keys are bolded and underlined. Foreign keys have "(FK)" next to them. The conversion saved us a lot of work, but it wasn't perfect because in many instances we already created the foreign key attribute in the entity, but the tool added another one labeled with "(FK)". So we'll need to do some cleanup using the edit panel to rename attributes, delete attributes, assign the correct primary keys, and so on. To redo a foreign key, simply use the *Connect* function at the top to redraw your connection. In the end, you want your diagram to look like this:



Question 3 (5 pts): The referencing column of a foreign key must be a primary key of the referenced table. Is this statement True or False? Why do you think so?

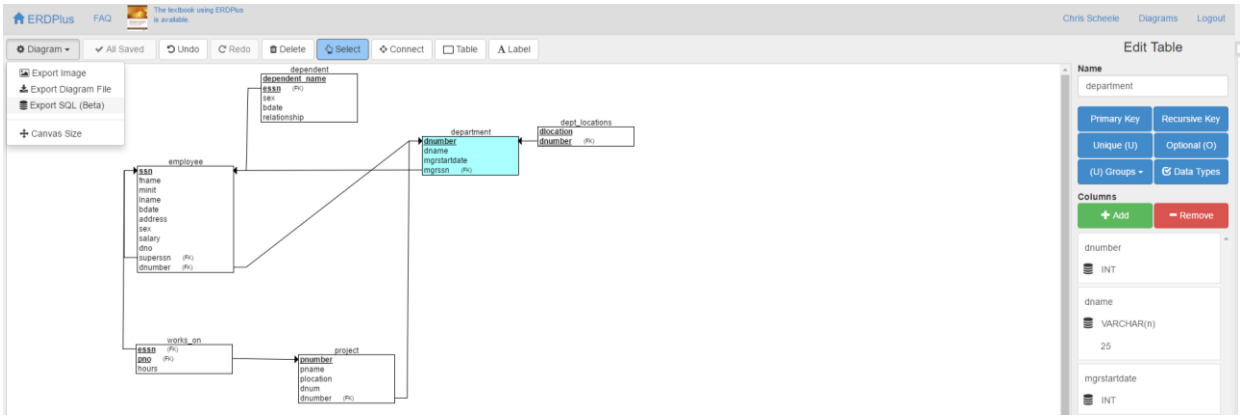
The final step before export is to go back to the tables in the begin and assign data types. To do this in ERDPlus, select an entity and click on the Data Types button in the edit panel. Then select the appropriate data type and length from the drop down for each attribute. Here are a couple of examples:



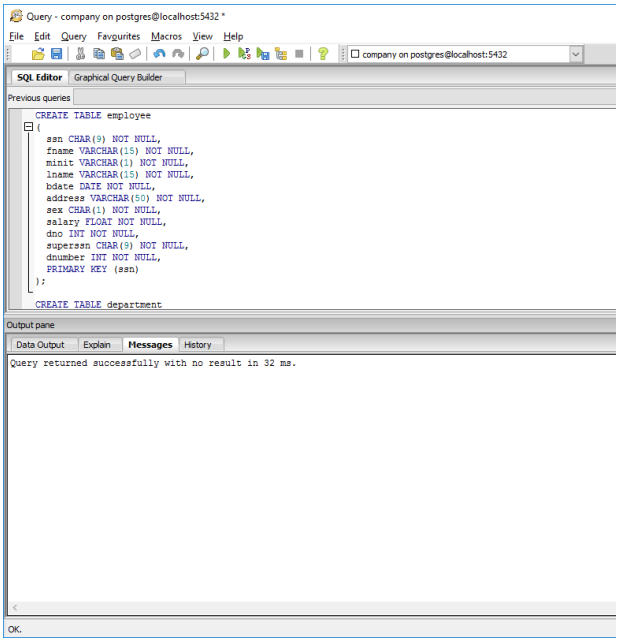


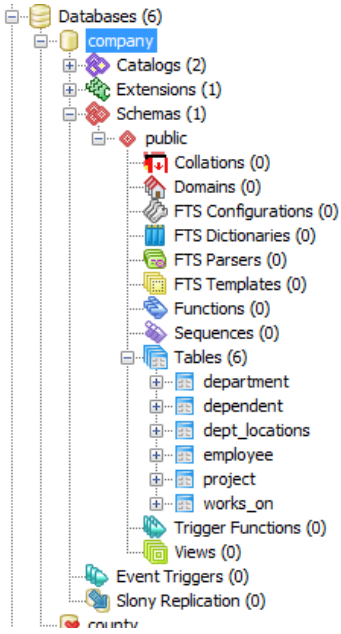
8. **Exporting the diagram to create a database**

Once you have completed adding the data types, it is time to implement it as the database in the DBMS. In the upper left corner select *Diagram* > Export SQL



A window will appear with SQL statements you can copy. Open up pgAdminIII and create a new database called **company**. Open the database and go to the query builder (SQL magnify glass). Paste the SQL statements into the SQL Editor and run the query (green arrow). Since this is a beta, there might be errors when running this command, specifically with the foreign keys. I removed them from my statements and added them manually.





Question 4 (5 pts): Export your completed ER Diagram and Relational Diagram as images.

Part II: Lab Assignment

Please submit all the required write-up and files in a single .zip file

- Answer ***Question 1 – 4*** (20 pts) listed in Exercise Tutorial.

- ***Question 5*** (30 pts): Pick one of the five databases, design an Entity-Relationship diagram for it, an E-R relational logical schema, and enter the design using pgModeler.
  Submit:
  1) E-R diagram (following E-R diagram notations);
  2) E-R relational logical schema (like the one on slide #7 in "Lab 3 Guidelines" powerpoint)

1. Consider a *mail order* database in which employees take orders for parts from customers. The data requirements are summarized as follows:
   - The mail order company has employees identified by a unique employee number, their first and last names, and a zip code where they are located.
   - Customers of the company are uniquely identified by a customer number. In addition, their first and last names and a zip code where they are located are recorded.
   - The parts being sold by the company are identified by a unique part number. In addition, a part name, their price, and quantity in stock are recorded.
   - Orders placed by customers are taken by employees and are given a unique order number. Each order may contain certain quantities of one or more parts and their received date as well as a shipped date is recorded.
2. Consider a movie database in which data is recorded about the movie industry. The data requirements are summarized as follows:
   - Movies are identified by their title and year of release. They have a length in minutes. They also have a studio that produces the movie and are classified under one or more genres (such as horror, action, drama etc.). Movies are directed by one or more directors and have one or more actors acting in them. The movie also has a plot outline. Each movie also has zero or more quotable quotes that are spoken by a particular actor acting in the movie.
   - Actors are identified by their names and date of birth and act in one or more movies. Each actor has a role in the movie.
   - Directors are also identified by their names and date of birth and direct one or more movies. It is possible for a director to act in a movie (not necessarily in a movie they direct).
   - Studios are identified by their names and have an address. They produce one or more movies.
3. Consider a conference review system database in which researchers submit their research papers for consideration. The database system also caters to reviewers of papers who make recommendations on whether to accept or reject the paper. The data requirements are summarized as follows:
   - Authors of papers are uniquely identified by their email id. Their first and last names are also recorded.
   - Papers are assigned unique identifiers by the system and are described by a title, an abstract, and a file name containing the actual paper.
   - Papers may have multiple authors, but one of the authors is designated as the contact author.
   - Reviewers of papers are uniquely identified by their email id. Their first and last names are also recorded.
   - Each paper is assigned between two and four reviewers. The reviewer rates the paper assigned to him on a scale of 1 to 10.

- Each review contains two types of written comments: one to be seen by the review committee only and the other by the author(s) as well.

4. Consider the following set of requirements for a UNIVERSITY database that is used to keep track of students' transcripts.
   - The university keeps track of each student's name, student number, Social Security number, current address and phone number, permanent address and phone number, birth date, sex, class (freshman, sophomore, ..., graduate),major department, minor department (if any), and degree program (B.A., B.S., ..., Ph.D.). Some user applications need to refer to the city, state, and ZIP Code of the student's permanent address and to the student's last name. Both Social Security number and student number have unique values for each student.
   - Each department is described by a name, department code, office number, office phone number, and college. Both name and code have unique values for each department.
   - Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of the course number is unique for each course.
   - Each section has an instructor, semester, year, course, and section number. The section number distinguishes sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.
   - A grade report has a student, section, letter grade, and numeric grade (0,1, 2, 3, or 4).
5. Consider an online auction database system in which members (buyers and sellers) participate in the sale of items. The data requirements for this system are summarized as follows:
   - The online site has members who are identified by a unique member id and are described by an email address, their name, a password, their home address, and a phone number.
   - A member may be a buyer or a seller. A buyer has a shipping address recorded in the database. A seller has a bank account number and routing number recorded in the database.
   - Items are placed by a seller for sale and are identified by a unique item number assigned by the system. Items are also described by an item title, an item description, a starting bid price, bidding increment, the start date of the auction, and the end date of the auction.
   - Items are also categorized based on a fixed classification hierarchy (for example a modem may be classified as /COMPUTER/HARDWARE/MODEM).
   - Buyers make bids for items they are interested in. A bidding price and time of bid placement is recorded. The person at the end of the auction with the highest bid price is declared the winner and a transaction between the buyer and the seller may proceed soon after.
   - Buyers and sellers may place feedback ratings on the purchase or sale of an item. The feedback contains a rating between 1 and 10 and a comment. Note that the rating is placed by the buyer or seller involved in the completed transaction.