David Chen
Ryan Yun
CS 340 Project Step 3 Final Version

Link to website - http://flip1.engr.oregonstate.edu:5641/

Changes made for Project Step 3 Final

- Added Foreign Key to employee (recursive relationship) in our employee table data definition query (DDQ) SQL.
- Created the DDQ SQL command to create employee-to-employee table.
- Fixed a typo on line 78.
- Deleted duplicate DMQ SQL command that created, updated, and deleted branch entity.
- Added tables with sample data to better show intended design.
- Changed certain text input elements to select elements to better indicate intended design.

Project Step 3 Draft Review Feedback

Fauzi Kliman

Hello,

Your database outline is very thorough! I would recommend restructuring your Schema diagram and list which attributes are foreign keys. It is currently difficult to understand all the foreign key relationships looking at your Schema diagram.

Hey Ryan,

This is looking like a good rough draft! The following are some thoughts I had while I was reading through your documents. First, here are some notes on your DDQ:

- Your DDQ needs sample data.
- The order you create the tables needs to either be rearranged so that you are always referencing tables (for foreign keys) that have already been created, or you can add the foreign keys after all of the tables have been created. For example, when `employee_to_branch` is created, it can't reference the `branch` table because the `branch` table hasn't been created yet.
- `start_date` and `birthday` should probably be of data type "date"
- Is there a particular reason you have `employee_to_department` and `department_to_employee`? Couldn't those just be one table? Same goes for `department_to_branch` and `branch_to_department`.

Second, here are some ideas and questions I had while reading through your DMQ:

- Line 21 in your DMQ references an `employee_to_employee` table that is not created in your DDQ.
- I was struggling to wrap my head around what the DMQ would look like for filtering/searching a table, and it looks like you did that with your position and branch

data, so that looks good and was helpful to me. However, I think the project specs state that you need to be able to filter/search all tables, so I think you'll need to do that for your employee and department data as well.
- There is a tiny typo on line 78 with one of the instances of the table name `managerial_duties`
- Looks like you accidently have some of the manipulation queries for your branch data duplicated there at the end.
- I thought we still needed the back-ticks on the names of the tables and columns in the DMQ, is that not correct?
- And finally, I didn't see where you had manipulation queries for adding and removing from a many-to-many relationship, which is one of the requirements on the project specs. Did I just miss it?

Finally, it's hard to really see if your website has everything when some stuff won't show up until you have the database connected and have the back-end up and running, so I just had a couple questions:

- Will the tables show up automatically when the page loads?
- Will the update/delete buttons trigger forms or a drop down menu to show up when you click them? How will those work?

Okay, that's all I have for now. Again, this looks good, and was super helpful to see this, this step makes a lot more sense after looking at what other groups are doing.

Changes made for Project Step 2 Final

- Indicated participation and cardinality for Employee in ER Diagram in the Employee to Branch relationship
- Added notation for recursive relationship between Employee and Employee
- Removed not null constraint for Department attribute on Branch
- Fixed notation in ERD for Employee to Relationship relationship (reversed)
- Fixed notation in ERD for Employee to Branch (reversed)
- Changed Age attribute in Employee to Birthdate
- Split Name attribute in Employee into First Name and Last Name attributes

Project Step 2 Draft Review Feedback
from Navine Rai

Some ideas that came to mind:

- In your ER diagram: the start of the line from Employee to Branch at the Employee end has no notation indicating participation or cardinality of the Employee entity. Based on the text portion of your outline, should there be two lines here to indicate exactly one employee?

- it seems that the Employee to Employee relationship is a Recursive Relationship since managers are employees but also manage employees. This is explained in your text section but if you'd like to include it in the ER diagram, the video lecture Special Cases has an example of possible notation that could be used to indicate this at around 2:50.

- lastly, I would add some sort of notation in the schema to indicate which values are the primary keys. Perhaps underlining as is shown in the lecture videos or a short abbreviation such as PK.

---

from Cherie Young

Good details in the draft. Some ideas below.

The Department and Branch entities have circular references that may be hard to implement in a live database. Both entities have each other as the foreign key, and they are both have constraints so that they cannot be null. How would you be able to add data to each table? For example, how would you be able to insert into your Department table if the item in the Branch table does not yet exist? Since the Branch foreign key cannot be null, you would be unable to create a row in the Department table while temporarily setting the Branch foreign to null. One way around this is to remove the "not null" constraint of one of the entity's foreign key.

Employee entity:

- The "Employee ID Number" attribute
    - I suspect this will be your primary key?
- The "Start Date" attribute
    - How will it be stored as an int? Will it be an epoch format?

Position entity:

- The "Position ID" attribute
    - I suspect this will be your primary key?
- The "Title" attribute
    - You might want to specify the max character of the varchar

Department entity:

- The "Department ID" attribute
    - I suspect "Department ID" is your primary key?
- The "Department Name" attribute
    - You might want to specify the max character of the varchar

Branch entity:

- The "Branch ID" attribute
    - I suspect "Branch ID" is your primary key?
- The "City Name" and "Country" attribute
    - You might want to specify the max character of the varchar

---

from Harrison Latimer

Hello David and Ryan,

Great detail with your diagrams and overall schemas! The only thing I was wondering is if it would be easier to just merge "Work At", "Hold Position", and "At Location" tables. It seems like this is getting at storing information about the specific location a person works. Why not just make an Entity called "Store Location" and have an store id primary key that could map to a specific employee that way instead of having to use three separate tables to do the same thing. I think you could also probably do something similar with the three tables labeled "Works In", "Manages" and "Has Department". It could also be that you simply listing your composite relationships between tables so let me know If I am misunderstanding what it is your tables represent. Again great job overall!

---

from Kristopher Hill

Hi Ryan,

After reading through your database design and the other databases in this group, I have some good ideas of where my partner and I need to improve our own. Your list of entities and their attributes and those attributes' data types is laid out very clearly and succinctly. Overall, I think your database design looks good.

One area where I think you will want to take a look at are the relationships in your ER diagram. I could be misunderstanding how the relationships are supposed to be read, but it seems like on er diagram, they are a bit mixed up. For example, in your description of the relationship between the employee entities and the department entities, you say it's a one-to-many relationship because "employees can only be in one department, whereas multiple employees can be in a department," but the way I think we are supposed to read the ER diagram, you've drawn the relationship between employees and departments to say, "an employee belongs to one or departments and a department has exactly one employee." Likewise, your employee-branch relationship is described as a similar one-to-many relationship, but your ER diagram shows, "an employee belongs to one or many branches" and there is no description of the participation or cardinality of the branch-to-employee relationship.

This brings up something that I mentioned to Danny with his project ER diagram, which is that I'm pretty sure we should have cardinality and participation for both parts of all relationships. I could be wrong that it's not a requirement for the class, but it seems like a good idea anyway. For example, the branch-to-department relationship has cardinality but not participation, and the relationship between positions and employees has no participation shown for either side of the relationship. Honestly, I can't remember if this was necessary in the first draft and it looks like you posted the first draft, so you might have already fixed this, and if that's the case, just ignore this. Actually, as I was looking at these relationships, I noticed that you also describe the

employee-position relationship as being one-to-many, but your ER diagram shows it as being many-to-many, so that's another thing you are going to want to fix.

Another aspect of your database to consider looking at again is some of the attributes of your entities. Honestly, I've only spotted a couple things that might cause trouble/be unnecessary, so don't take this to mean you need to rework a whole bunch of your attributes. One example is the age attribute for employees, which might present a problem later, since it will have to be changed once a year and on different dates for each employee. It might be better to use a birthdate attribute instead and derive their age from that (there is a specific term for this kind of attribute, but I can't remember it now). Also, have you thought about instances where you would have an employee in your database with an employment status of false? Are you planning on keeping a record of all employees after they no longer work for the business? That's fine, just something I noticed.

So, those are some things I noticed about your database that might need to be looked at and worked on. I might come on later and some other comments if I have the time, and if you have any questions regarding these notes, feel free to let me know.

---

from Phi Luu

Hello Ryan,

I like the idea of your group going for a near real-world project (the Employee system). Your choice of entities are very good, as the system has plenty of hierarchies and relationships. Here are a few suggestions I have for your project:

- Split Employee's *Name* attribute into *firstName* and *lastName*, as explained in the lecture. Also, doing so will help you sort the employees' names easier in the future.
- Change *StartDate* attribute of Employee to *date* type.
- Based on what you wrote in Relationships.1.d, it seems like Employee has a special relationship with itself, namely a recursive relationship. If that was what you meant, then you should add a self-loop to the Employee entity of your ERD.
- Clarify the ERD's *many* symbol, as they could mean "*zero-or-many*" or "*one-or-many*," like follows: That's about it! Good job on the idea and the draft!

Project Outline

This project is a company database. The four entities in this database are Employee, Position, Department, and Branch. Each entity contains various attributes including foreign keys that are used to link them to other entities. The primary goal of this database is to give the user access to all of the company's critical data such as employee's name, the head of a department, department branch, number of employees, etc.

Database Outline

Entities

1. **Employee**
    a. **Employee ID Number (type: int / constraints: not null, unique / autoincrement)**
    b. **First Name (type: varchar(40) / constraints: not null)**
    c. **Last Name (type: varchar(40) / constraints: not null)**
    d. **Birthday (type: date / constraint: greater than or equal to 18)**
    e. **Branch: Foreign Key to Branch (type: int / constraint: not null)**
    f. **Department: Foreign Key to Department (type: int / constraint: not null)**
    g. **Position : Foreign Key to Position (type: int / constraint: not null)**
    h. **Salary (type: int / constraint: not null)**
    i. **Start Date (type: datetime / constraint: not null)**
    j. **Employment Status (type: boolean / constraint: not null)**
    k. **Manager: Foreign Key to Employee (type: int)**

2. **Position**
    a. **Position ID (type: int / constraint: not null, unique)**
    b. **Title (type: varchar(40) / constraint: not null, unique)**
    c. **Has Managing Duties (type: boolean / constraint: not null)**

3. **Department**
    a. **Department ID (type: int / constraint: not null, unique)**
    b. **Head of Department: Foreign Key to Employee (type: int)**
    c. **Department Name (type: varchar(40) / constraint: not null)**
    d. **Number of Employees (type: int / constraint: not null)**
    e. **Branch: Foreign Key to Branch (type: int / constraint: not null)**

4. **Branch**
    a. **Branch ID (type: int / constraint: not null, unique)**
    b. **City Name (type: varchar(50) / constraint: not null)**
    c. **Country (type: varchar(50) / constraint: not null)**
    d. **Employee Count (type: int / constraint: not null)**
    e. **Departments: Foreign Key to Department (type: int)**

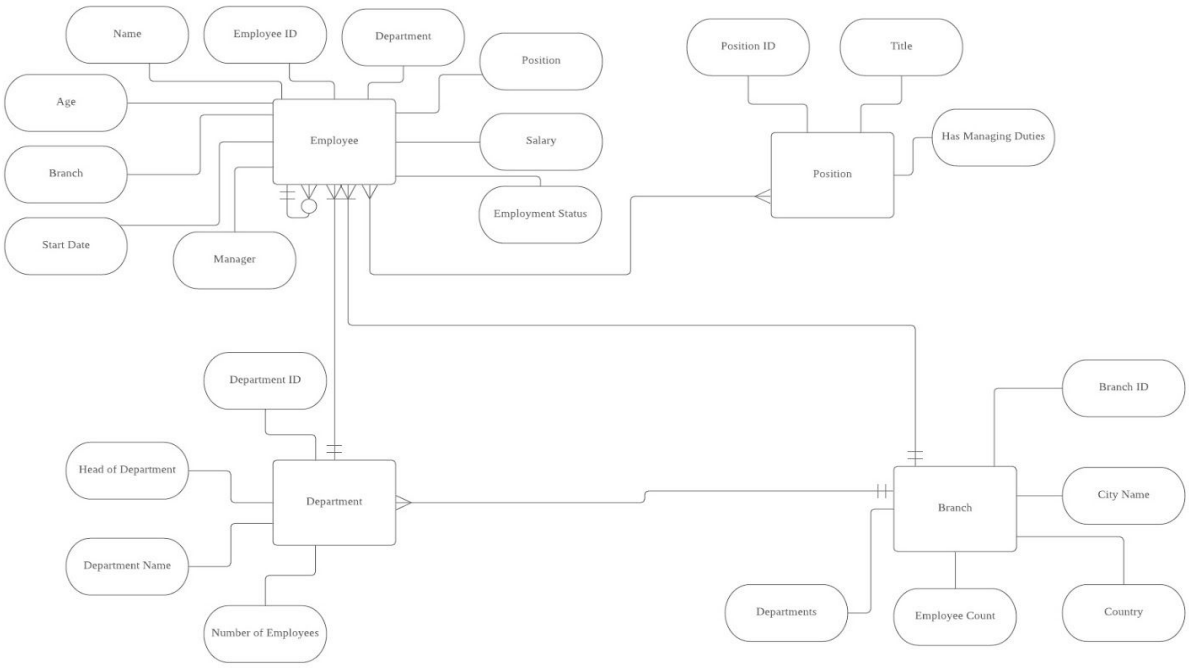
<u>Relationships</u>

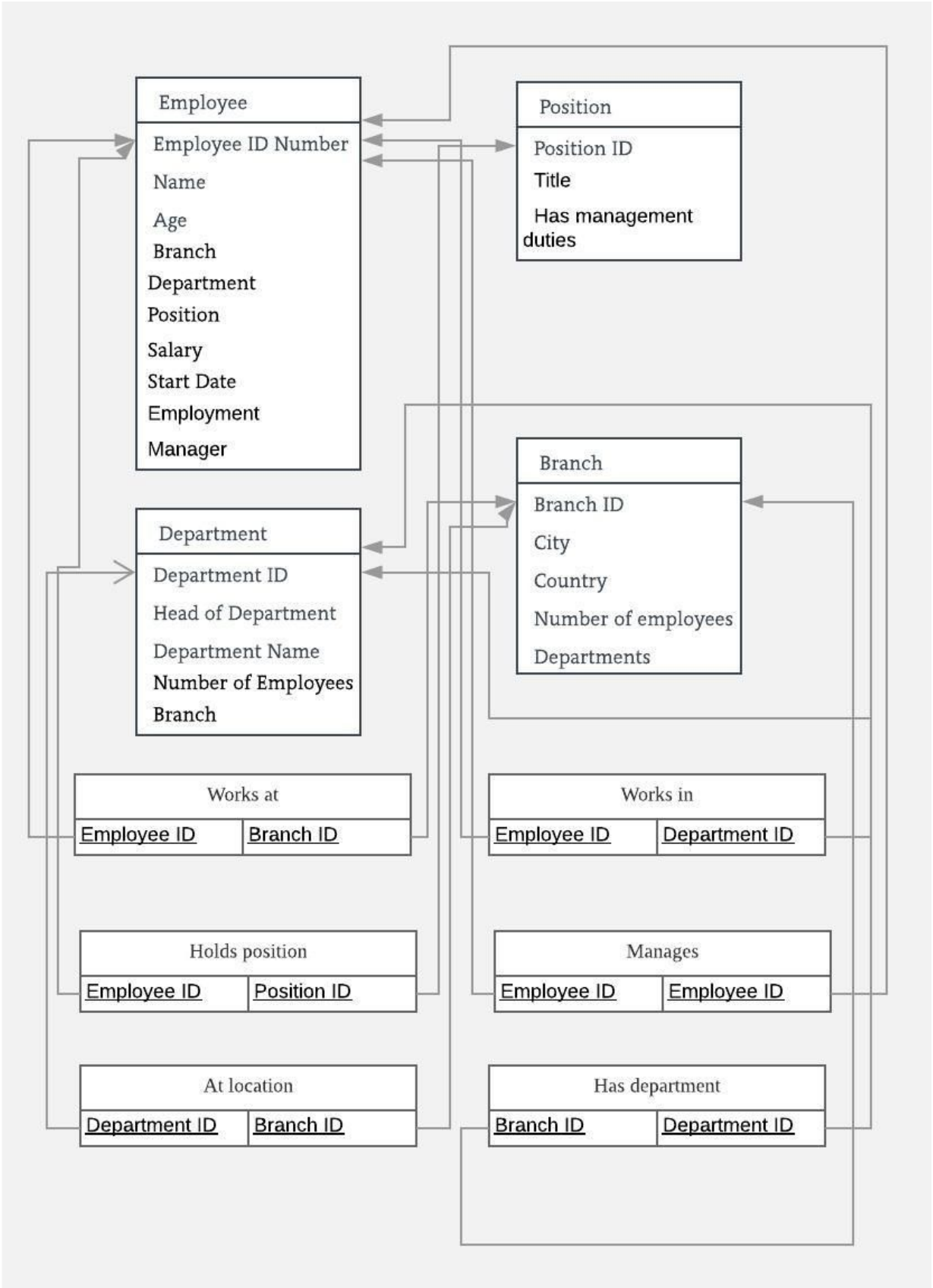
1. **Employee**
    a. **Employee - Branch: This is a many to one relationship because employees can only work at one branch, whereas multiple employees can work at a branch.**

b. **Employee - Department: This is a many to one relationship because employees can only be in one department, whereas multiple employees can be in a department.**
c. **ID Employee - Position: This is a many to one relationship because each employee can only have one position, whereas several employees can have the same position.**
d. **Employee - Employee: This is a many to many relationship because employees can be managed by several different employees who are managers and employees who are managers can manage several different different employees.**

2. **Position**
    a. **Position - Employee: Every employee has at most one job. This is a one to many relationship because multiple employees can have the same job but one employee can only have at most one job.**
3. **Department**
    a. **Department - Employee: Every employee belongs to a department. This is a one to many relationship because multiple employees can belong to one department but department can only belong to one employee.**
    b. **Department - Branch: This is a many to one relationship because departments can only belong to at most one branch, and each branch can contain several departments.**
4. **Branch**
    a. **Branch - Department: This is a one to many relationship because each branch can have multiple departments, but each department can only belong to a single branch.**

ERD Diagram


Name
Employee ID
Department
Position
Age
Employee
Salary
Branch
Employment Status
Start Date
Manager
Position ID
Title
Position
Has Managing Duties
Department ID
Branch ID
Head of Department
Department
Branch
City Name
Department Name
Departments
Employee Count
Country
Number of Employees


Employee
Employee ID Number
Name
Age
Branch
Department
Position
Salary
Start Date
Employment
Manager
Position
Position ID
Title
Has management duties
Branch
Branch ID
City
Country
Number of employees
Departments
Department
Department ID
Head of Department
Department Name
Number of Employees
Branch
Works at
Employee ID
Branch ID
Works in
Employee ID
Department ID
Holds position
Employee ID
Position ID
Manages
Employee ID
Employee ID
At location
Department ID
Branch ID
Has department
Branch ID
Department ID