ENTITY RELATION DIAGRAM

Entities

a. Employees
b. Job Title
c. Departments
d. Department Managers
e. Salary record

1. Employees

a. Employee ID - (PK)
b. Date of Birth – Simple Attribute
c. Name – Composite Attribute (Might have First, Middle, Surname etc.)
d. Gender – Simple Attribute
e. Salary – My input for salary is a simple attribute, but Employee Entity probably will use just salary_id after normalization.
f. Hire Date – Simple Attribute

2. Job Title

1. Job Title ID – PK
2. Job Title Name – Simple Attribute

3. Departments

a. Department ID – PK
b. Department name – Simple Attribute

4. Department Manager

Department Manager Entity is a weak entity. Therefore, relationship between the department and department manager is a weak relationship. PK of the entity belongs to Departments Entity. Therefore, Department Manager Entity existence related with existence of Departments Entity.

I thought for a while about the entity. It is kind of a middle table between employees and departments. I will explain further in the relations section.

a. Department Number – PK
b. Employee Number – Employee ID
c. Dates Managed – Composite Attribute – Start Date, End Date

5. Salary History

a. Date : A simple attribute
b. Salary : A simple attribute
c. Description : If there is any changes for the salary explanation for that.

Relationship Between The Entities

1. EMPLOYEES has a JOB TITLE
2. EMPLOYEES earn SALARY
3. EMPLOYEES work in DEPARTMENTS
4. MANAGERS manage DEPARTMENTS

The explanation for Department Manager Entity – Departments – Employee relation.

Department ID (PK) and Employee ID (PK) should be compound key for Department Manager Entity. Therefore, it is a weak entity and a middle table. However, provided information for the Department Manager Entity requires a unique Department Number which is (PK). It is also added as an PK attribute. It will be checked in normalization part.

CARDINALITY

Employees - Job Title : Each employee MUST have a job title. It is total participation. Each employee can have only one title and many employees has title. It's 1 to N

Department Manager – Department : A department can have several manager and a manager can manage only one department. It's also 1 to N.

Employees – Salary_ History : Each employee can have many salary records because of salary increments. But more than one employee cannot have same salary records. It's unique for each employee. It's 1 to N.

Employees – Department : There are many departments can have many employees, and an employee can belong to many departments. It's M to N.

ENTITY RELATION DATA MODEL

Changes Implemented to ERD (THERE IS ONLY IMPORTANT CHANGES MENTIONED COMPARED TO ERD TO AVOID OVERWRITING)

1. EMPLOYEES ENTITY

a. **Salary attribute:** It's replaced with salary_id(FK). There will be an additional salary table that contains salary_id(PK) and salary.
b. **Job_Title attribute:** In order to achieve a job title for each employee it was mandatory to have a job title attribute with job_title_id(FK).
c. **Employee_status:** It's a Boolean data type. It has two values. Active(true) and Inactive(false). When an employee's status is changed to inactive, relative information about the employee won't be displayed anywhere. However, related employee data will remain in the database.

2. SALARY HISTORY ENTITY

a. **Salary History ID:** Must be added as a new attribute for having salary records (PK).
b. **Salary_ID**: It's a foreign key from a middle table(Salary).
c. **Salary**: Simple Attribute. It's replaced with the salary_amount_history attribute.
d. **Sorted_counter**: Multi-Value Attribute. It's an auto-increment attribute. Start from "0" as a default value. It's added for precise filtering and sorting.
e. **Date**: New or Initial salary start date. I didn't add an end date since the start date of the new salary value will be the end date of the old salary value.

I can add a description for this entity. It might have a default value of New Employee if it has a new salary_history_id. Increments or promotions must be added to a description by the end user. Although, I wanted to keep it simple.

3. SALARY ENTITY

It's a new table to manage recording salary changes in Salary History Entity.

a. **Salary ID:** Primary Key of the table.
b. **Salary**: Stores recent salary amount of employees.

4. DEPARTMENT ENTITY

a. Department_ID: A PK for the entity.
b. Department_Name: A simple attribute.

5. DEPARTMENT MANAGER ENTITY

There will be an additional table to store manager history. Therefore below changes were implemented.

a. **Department_ID and Employee_ID:** Compound key for the recent managers of the department. There is already a manager history entity. Also, it's an associative entity.

6. DEPARMENT MANAGER HISTORY

a. **Manager_History_ID:** A PK for the entity.
b. **Department_ID:** FK from department entity.
c. **Employee_ID:** FK from employee entity.
d. **Sorted_Counter:** Auto increment attribute. Defaults start from "0". It's added for precise filtering and sorting.
e. **Management_Start:** A date simple attribute.
f. **Management_end:** A date optional attribute and also a derived attribute. It's optional because once a manager started to manage a department start date must be recorded in the entity and the record cannot have an initial end date.

7. EMPLOYEE DEPARMENT

a. **Department_ID and Employee_ID:** They are compound keys for the entity. It's an associative entity.

Relationship Between The Entities and Cardinality

5. **EMPLOYEES has a JOB TITLE**: Every employee must have a job title. But every job title needn't be captured by an employee. Many employees can have the same job title, but one employee can have only one job title. It's a total partition of many-to-one N:1.
6. **EMPLOYEES earn SALARY**: One employee can have only one salary, and each salary can belong to one employee. It's one-to-one 1:1.
7. **SALARY records in SALARY HISTORY:** Every salary change event gets recorded in the salary history entity. Each salary can have many records. But each record can have only one. It's many-to-one N:1.
8. **EMPLOYEES has DEPARTMENTS :** Many departments can have many employees. It's many-to-many relations N:M. (There is an associative entity between two entities. Since there is a middle table between the two entities indicates which employee works where.)
9. **EMPLOYEES manage DEPARTMENTS:** Managers are also an employee. Hence, there is no need for a manager table. Between two entities there is an associative entity with a compound key that identifies managers' responsible departments with a

management task. Each employee can manage one department. Each department can have many managers. It's one-to-many 1:N.

10. **DEPARTMENT MANAGE HISTORY:** The entity records managers' management history. It gets the compound key from the associative entity. It works in the same way as salary records apart from the relation and cardinality (it will be clarified in the how it works section.). Each manager can have only one management history. Many managers can have a management history. It's one-to-many 1:N.

HOW IT WORKS AFTER NORMALIZATION

THE SUM OF ALL STUDIES MUST BE A MAXIMUM OF 5 PDF PAGES. THEREFORE, THIS PART IS NOT INCLUDED IN TO REPORT FOR SUBMISSION OR EXTRA MARKS. ADDITIONAL INFORMATION ABOUT HOW IT WORKS IN REAL IMPLEMENTATION. JUST AN OPTIONAL PART FOR FURTHER INFORMATION

The description of DB is handled with a real example scenario. There will be different examples for different cases below:

New entry for HR DB: A new employee is being registered to their system by the HR end user. None of the employee attributes can be null including salary_id(FK). Otherwise, the system will pop an error. If everything is okay then identified salary will be registered to the salary entity in advance and the system stores a PK for the record. Thus all attributes will be ready to record a new employee. If there are any changes in the salary entity, it triggers a new record for the salary history entity. The new record has a PK, salary_id(FK), date, sorted_counter(auto-increment attribute), and amount of the salary. Amount_of_the salary is assigned by salary_id salary value. A description might be added for the history entity as an attribute with a default value of New Employee. If there is a new record attribute value will be New Employee by default. If there is an increment in the existing salary then the end user should enter a description for that increment and it will be stored for that record as a description.

Promotion Increment: If an employee's salary will be updated it will affect directly the salary entity. New salary value will be updated in the employee entity via salary id(FK). As mentioned above, the salary entity updates will trigger the salary history entity. If there is already a record for the related salary id, the new record attribute sorted_counter will increase by one. And the record will be occurred as per as above steps.

Employee's Job Title: There are already ready-to-use titles for the employees. If not it can be added through the system. Each employee must have a title.

Employees' Department: The employee department identifier is a middle table between the department and the employees' entity. Works with a compound key and identifies who works in which department. Through the HR system, an employee can be assigned to several departments. That will add another tuple to the middle table between the department and employees with their ids'. Thus employees' departments can be quarried.

Departments managers: Departments managers are also an employee. Department id and Employee id will be sufficient to mark managers. It looks the same as the Employees' Department. But it's a

completely different table and related to the management history table. It works in the same way as salary records. Every change triggers a new record in the department's management history table.

Department's Management History: All records work by themselves. If a manager is assigned to a department to manage the entity gets a new record by fetching the employee's id and the department's id. Unless a manager leaves his job start date, sorted counter, management history id (PK) initial

ER MODEL CROW'S FOOT NOTATION

Cardinality

Cardinality is the only important point to be explained for Crow's Foot notation. There is also middle tables we are not able to see as clear as Crow's Foot notation compared to Chen notation. Instead associative entities used for middle tables with Chen notation.(Employee Department and Department Manager).

I would like to explain cardinalities between the entities as far as I understood from my researches.

Employee – Job Title: Employees has a job title. Each employee only can have one job title and it's mandatory. On the other hand, many job titles captures many employee, however there might be some job titles are not captured. It's optional many. So the cardinality between to entity is ONE MANDATORY – MANY OPTIONAL.

Employee – Salary: Employees earns Salary. One employee can have only one salary and it's mandatory on both sides. ONE MANDATORY TO ONE MANDATORY.

Salary – Salary History: One salary can have many records. And even if there is a new employee initial salary will be recorded. It's ONE MANDATORY – MANY MANDATORY.

Employee – Department Manager: One department manager can be only one employee and many employees are department managers but there are employees and they are not a manager. MANY OPTIONAL - ONE MANDATORY.

Department Manager – Department: Each department can have many managers and each department must have a manager. A manager can manage only one department and all managers must manage a department. It's ONE MANDATORY – MANY MANDATORY.

Department Manager – Department Manager History: Each department manager can have many Department Manager History. Every manager must have a department manager

history. Every Department Manager History must belong to a manager. It's ONE MANDATORY – MANY MANDATORY.

Department_id and Employee_id just indicates employees' department.

Employee department and Employee: Each department can have many employees and each employee can work in many departments. Each department must have employee and each employee must be in a department. It's MANY MANDATORY – MANY MANDATORY

Employee Department and Department: It's same as above cardinality. Just indicates employee departments by a compound key.

GITHUB LINK

https://github.com/aybatu/webdev_ca1

REFERENCES

Constantine Nalimov, Sep 2, 2020.
https://www.gleek.io/blog/crows-foot-notation.html

1.1.1 Patrycja Dybka, Java Software Developer.

https://vertabelo.com/authors/patrycja-dybka/

raman_257, DBMS-ER MODEL
https://www.geeksforgeeks.org/attributes-to-relationships-in-er-model/

reaanb, Jul 14, 2016
https://stackoverflow.com/questions/38366466/when-to-use-associative-entities