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 additional salary table which contains salary_id(PK) and salary.
b. Job_Title attribute : In order to achieve job title for each employee it was mandatory to have an 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 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 replaces with 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 start date of new salary value will be end date of the old salary value.

I can add a description for this entity. It might have a default value New Employee if it has a new salary_history_id. For increment or promotions must be added a description by 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 implemented.

a. Department_ID and Employee_ID : Compound key for the recent managers of the department. There is already a manager history entity. Also its and 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 starts 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 of once a manager started to manage a department start date must be recorded in the entity and record cannot have an initial end date.

7. EMPLOYEE DEPARMENT

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

Relationship Between The Entities and Cardinality

1. **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.
2. **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.
3. **SALARY records in SALARY HISTORY :** Every salary change event gets recorded in the salary history entity. Each salary can have many records. But each records can have only one. It's many-to-one N:1.
4. **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.**)**

5. **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.
6. **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

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) initializes by itself. Initially, the end date will be null. If there is a new record with the relative manager sorted counter will increase by one and the end date attribute of the previous record will be equal to the start date of the new record. If managers' employee status is changed to inactive it will trigger department manager history entity. Thus related manager management end date will be updated upon employee status change date.