Salary Management System

Abstraction

Salary Management System is aimed at efficient management of employee information, emoluments, expenses, net pay-outs, calculation salary based on workdays and pay salary etc. All of these are managed through a database. This project is terminal project, which implemented using PL/SQL and oracle database. The database is divided based on different conditions which are known as fragments and these fragments are kept at different locations which has Database Management System to deal with the data. The idea of dividing/fragmenting the data makes the system reliable, fast with better response. In case of database failures, the system remains functional though it may reduce performance.

Motivation

Companies, it may be public or private increasing to fast with population growth. And in every company, there is a must essential thing, a system which manage employee information and their payments. So having a well-organized Salary Management System is a market demand. So we tried to implement a smaller conceptual version of Salary Management System. And as there are some large companies has distributed database system, so we focus on distributed database system in small manner so that system can perform fast and efficient ways.

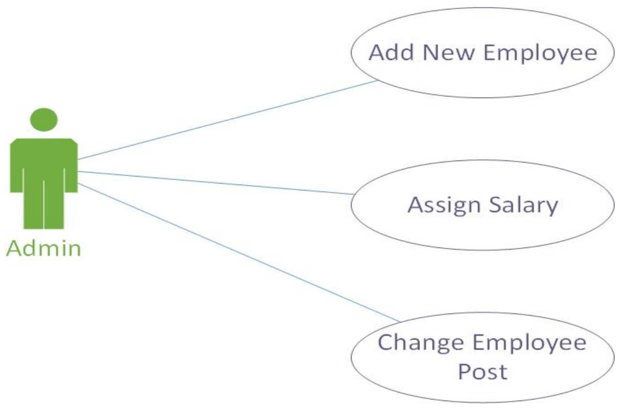
Types of user

There are 3 types of user for our project

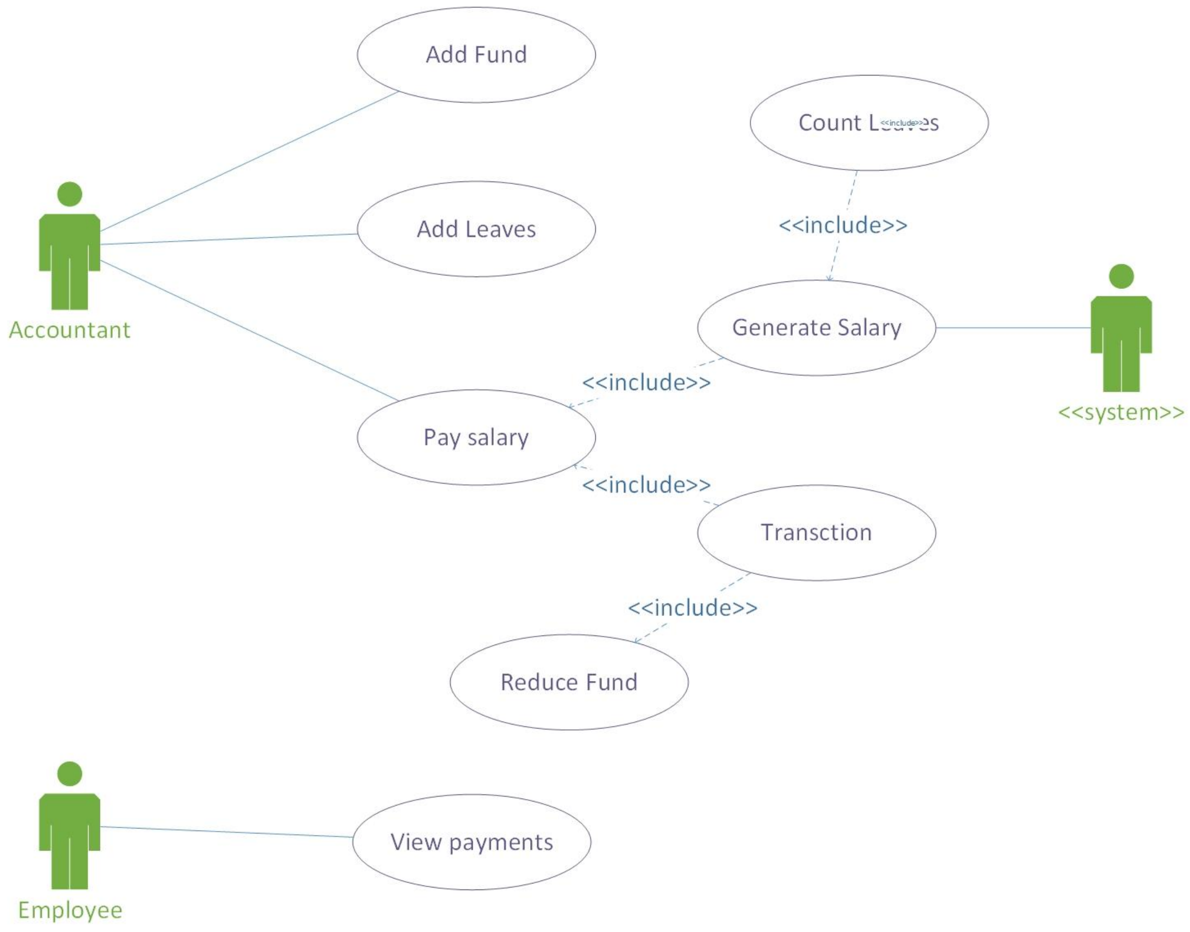
1. Admin: Employer or HR
2. Accountant
3. Employees

Use Case Diagram

Admin use case:



Accountant and Employee use case:



Procedures and Functions For The Project

1. Employee
   Module
   - ViewDetails()
     - ✓ Usage : Show employee his/her details with payment history
     - ✓ Parameter: EmployeeID
     - ✓ Output : Show details of given employee ID
2. Accountant
   Module
   - PaySalary()
     - ✓ Usage : Pay Employees Salaries by a function GenerateSalary() and a procedure TransectSalary() call
     - ✓ Parameter: Employee ID, Month of Giving salary
     - ✓ Output : Pay salary of given EID and month

   Procedures
   - AddFund()
     - ✓ Usage : Increase fund
     - ✓ Parameter: Amount
     - ✓ Output : Increase fund with given amount
   - AddLeave()
     - ✓ Usage : Add leaves of Employees of specific month
     - ✓ Parameter: EID, S_month, L_days
     - ✓ Output : Add leaves of given Employee ID, Month
   - TransectSalary()
     - ✓ Usage : It's the transection procedure for paymets use function ChechValid() and procedure UpdateFund() to complete
     - ✓ Parameter: EID, amount, month
     - ✓ Output : Transection for payments
   - UpdateFund()

- ✓ Usage : After transection the amount of payments will be reduce by this procedure
- ✓ Parameter: Amount
- ✓ Output : Updated fund after transection

Functions

- GenerateSalary()
  - ✓ Usage : Generate salary calculate with leaves of given month
  - ✓ Parameter: EID, month
  - ✓ Return : Generated salary
- CheckValid()
  - ✓ Usage : This says either its payable or not
  - ✓ Parameter: EID, month
  - ✓ Return : 1 for valid 2 for invalid

3. Admin

   Procedures

   - AddEmployee()
     - ✓ Usage : Add new employee and assign his salary
     - ✓ Parameter: Name, gender, email, joiningDate, SID
     - ✓ Output : Add the person as employee and assigned his salary
   - ChangeEmpPost()
     - ✓ Usage : Employee promote or demote
     - ✓ Parameter: EID,SID
     - ✓ Output : Changed Pay Scale with given EID, SID

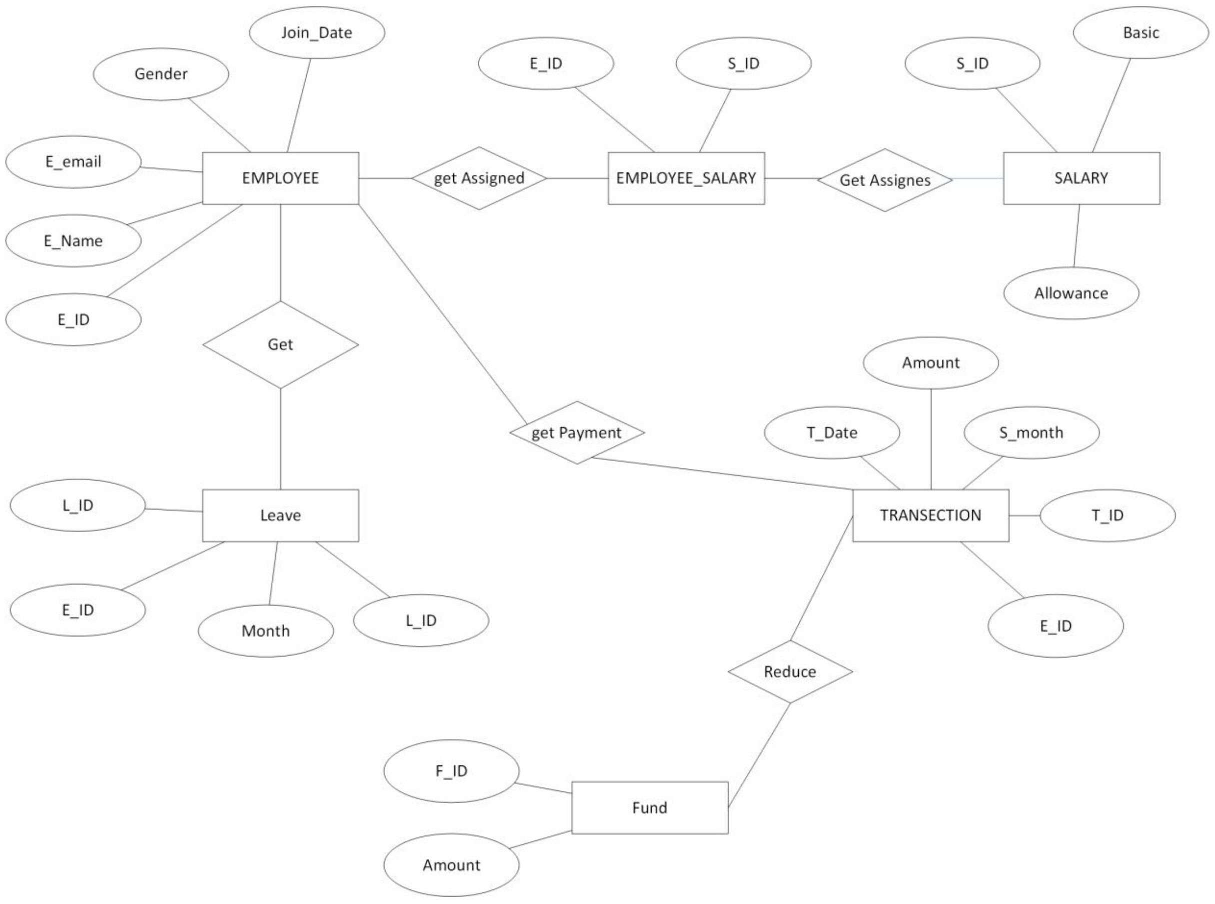
Triggers

- AddLeaves: Triggered if data insert into Leave table
- ChangeEmpSalary: Triggered if data update in Employee_salary Table and there is an **audit table** to track changes.
- AddEmployee: Triggered if data insert into Employee table
- AddEmpSalary: Triggered if new employee get assigned salary
- Transect: Triggered when transection occurs
- UpdateFund: Triggered id data update in Fund Table and there is an **audit table** to track changes.

Entities

- Employee(<u>EID</u>, EName, Gender, Email, JoinDate)
- Salary(<u>SID</u>, Basic, Allowance)
- Employee_Salary(<u>EID</u>(FK), <u>SID</u>(FK),)
- Leave(<u>LID</u>, EID(FK), L_month, L_days)
- Transection(<u>TID</u>, EID(FK), Amount, T_Date, S_month)
- Fund(<u>FID</u>, Fund_amount)

ER Diagram



Transparency

There are three levels of transparency in distributed database.

- Level 1: Location Transparency
- Level 2: Fragmentation Transparency
- Level 3: Local Mapping Transparency

In Level 3 transparency, location and fragmentation details are available to application user. In our project we divided the data into different fragments and kept them in different locations. All fragmentations are Horizontal fragmentation based on the ID. These fragments were kept in different sites. For example:

Emp1: Select * from Employee where eid <= 5;

Emp2: Select * from Employee where eid > 5;

Contribution

Throughout the project we divided our responsibilities and completed our task to make a demo of the project. The whole project has several phases.

- Creating an Entity Relationship Diagram
- Creating fragments
- Linking up all the sites.
- Inserting data to different sites according to the fragments
- Creating PL/SQL functions, procedures, control statement, triggers, packages
- Testing

My parts of this project was to cover **Admin/HR functionality and following triggers** discussed above and **linked up, and a part of fragmentation**, , though we did everything together.

Conclusion

In this project, we tried to implement based on some high quality Salary Management System. Though we couldn’t implement it as a whole but tried to give an idea how distributed database work in real life scenario. We look forward to implement the project in future on a larger scale.