Sri Lanka Institute of Information Technology

IT2040 – Database Management System

Year 2, Semester I, 2021

Topic 2

Employee Leave Management System

Name: **Jayakody J.A.M.G.**

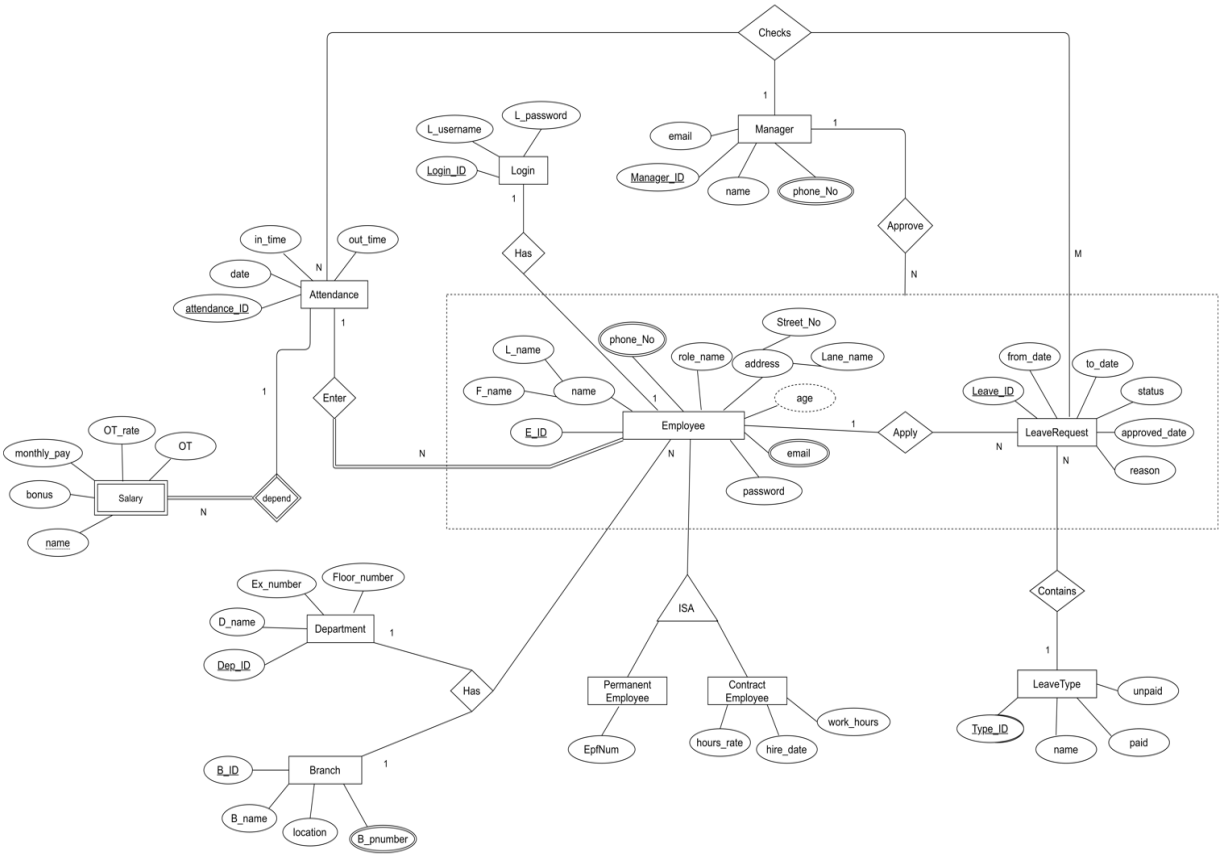
Registration Number: **IT20150648**

Group: **Y2S1 01.2**

1. The data requirements of Employee Leave management System

- An employee has an employee ID (unique), name (first name, last name composite attributes), age, role name, password, address (street no, lane name composite attributes), multiple phone numbers and multiple email addresses.
- Permanent employee and contract employee are types of employees.
- Permanent employee has epf number.
- Contract employee has work hours, hire date, hours rate as its attributes.
- Every employee has a department and a branch.
- Department has department ID (unique), name, floor number and extension number.
- Branch has a branch ID (unique), name, location, phone number.
- Employee has only one login and login contain login ID, login username, login password.
- Employee must enter their attendance. Attendance has attendance ID, date, in time and out time for its attributes.
- One employee can enter one attendance at once and one attendance can enter by many employees at one.
- Salary is a weak entity having attributes name, bonus, monthly payment. It depends on attendance.
- Every employee can apply leave by a leave request and it has leave id(unique), from date, to date, reason, approved date and status.
- One employee can apply for many leave requests.
- Leave request contains leave type having attributes type ID, name, unpaid and paid.
- Manager checks attendance and leave requests. Manager has manager ID (unique), name, email and multiple phone numbers.
- While applying for leave request, manager approves the request.

2. EER Diagram for Employee Leave Management System



3. Mapping of the EER model in to relational model

Employee (E_ID, F_name, L_name, age, Sreet_No, Lane_name, role_name, password, EpfNo, hours_rate, work_hours, hire_date, type, Dep_ID, B_ID, attendance_ID)

- Foreign key (Dep_ID) references Department (Dep_ID)
- Foreign key (B_ID) references Branch (B_ID)
- Foreign key (attendance_ID) references Attendance (attendance_ID)

Employee_phone_No (E_ID, phone_No)

- Foreign key (E_ID) references Employee (E_ID)

Employee_email (E_ID, email)

- Foreign key (E_ID) references Employee (E_ID)

Login (Login_ID, L_username, L_password, E_ID)

- Foreign key (E_ID) references Employee (E_ID)

Department (Dep_ID, D_name, Ex_number, Floor_number)

Branch (B_ID, B_name, location)

Branch_B_phone (B_ID, B_pnumber)

- Foreign key (B_ID) references Branch (B_ID)

Has (E_ID, Dep_ID, B_ID)

- Foreign key (E_ID) references Employee (E_ID)
- Foreign key (Dep_ID) references Department (Dep_ID)
- Foreign key (B_ID) references Branch (B_ID)

Attendance (attendance_ID, E_ID, date, in_time, out_time, Manager_ID)

- Foreign key (Manager_ID) references Manager (Manager_ID)
- Foreign key (E_ID) references Employee (E_ID)

LeaveType (Type_ID, name, paid, unpaid)

LeaveRequest (Leave_ID, from_date, to_date, reason, status, approved_date, Manager_ID, attendance_ID, E_ID, Type_ID)

- Foreign key (Manager_ID) references Manager (Manager_ID)
- Foreign key (attendance_ID) references Attendance (attendance_ID)
- Foreign key (E_ID) references Employee (E_ID)
- Foreign key (Type_ID) references LeaveType (Type_ID)

Checks (attendance_ID, Leave_ID)

- Foreign key (attendance_ID) references Attendance (attendance_ID)
- Foreign key (Leave_ID) references LeaveRequest (Leave_ID)

Manager (Manager_ID, name, email)

Manager_phone (Manager_ID, phone_No)

- Foreign key (Manager_ID) references Manager (Manager_ID)

Salary (attendance_ID, name, bonus, monthly_pay, OT_rate, OT)

- Foreign key (attendance_ID) references Attendance (attendance_ID)

3. C) Justification as to why the option selected is used to map the ISA hierarchy

According to my diagram:

- Overlapping - When one person is considered at the same time, the individual might be both a permanent and a contract employee. Employees who are both permanent and contract employees are overlapping.
- Disjoint - It means that if we consider one person at a time, that individual cannot be both a permanent and a contract employee at the same time. Employees might be permanent or contract employees at any given moment.
- Partial – In addition to permanent and contract employees, there are several other types of employees. We haven't mentioned that in our model/design. There are a few different sorts of employees. However, only permanent and contract employees are mentioned here. There may be employees in addition to permanent and contract employees
- Total (covering constraint) – Simply put, it states that no other employee types exist. The subclasses encompass all of the employee kinds. Other than permanent and contract employees, there are no other types of employees.

Therefore, I choose option 3, because according to the diagram permanent employee and contract employee cannot be overlapping. The specialization relationship must be disjoint. Here, Subclasses have few attributes. This relation subclasses covers the superclass.

.

SQL Queries for Creating Tables

```
create table Department(
	Dep_ID varchar(10)not null,
	D_name char(30),
	Ex_number varchar(20),
	Floor_number varchar(20)
	constraint Department_PK primary key(Dep_ID)

);




CREATE TABLE Employee(
E_ID varchar(10)not null,
F_name varchar(255),
L_name varchar(255),
age int not null,
Street_No varchar(10)not null,
Lane_name varchar(30),
role_name char(100),
password varchar(10),
EpfNum varchar(50),
hours_rate int not null,
work_hours int not null,
hire_date DATE not null,
type char(50),
Dep_ID varchar(10)not null,
B_ID varchar(10)not null,
attendance_ID varchar(10)not null,
CONSTRAINT Employee_pk primary key (E_ID),
CONSTRAINT Employee_fk foreign key(Dep_ID) references Department(Dep_ID)

);




create table Employee_phone(
	E_ID varchar(10) NOT NULL,
	phone_No char(30),

	constraint Employee_phone_PK primary key(E_ID,phone_No),
	constraint Employee_phone_FK foreign key(E_ID) references Employee(E_ID),

);
```

```sql
create table Employee_Email(
	E_ID varchar(10) NOT NULL,
	e_mail varchar(30),

	constraint Employee_Email_PK primary key(E_ID,e_mail),
	constraint Employee_Email_FK foreign key(E_ID) references Employee(E_ID),
	constraint Employee_Email_CK check(e_mail LIKE '%_@_%')

);




create table Login(
	Login_ID varchar(10) NOT NULL,
	L_username varchar(30),
	L_password varchar(30),
	E_ID varchar(10),
	constraint Login_PK primary key(Login_ID),
	constraint Login_FK foreign key(E_ID) references Employee(E_ID)


);



create table Branch(
	B_ID varchar(10)not null,
	B_name varchar(20),
	location varchar(30),
	constraint Branch_PK primary key(B_ID)

);




create table Branch_B_phone(
	B_ID varchar(10)not null,
	B_pnumber char(30),
	constraint Branch_B_phone_PK primary key(B_ID, B_pnumber),
	constraint Branch_B_phone_FK foreign key(B_ID) references Branch(B_ID)

);
```

```sql
create table Has(
	E_ID varchar(10)not null,
	Dep_ID varchar(10)not null,
	B_ID varchar(10)not null,
	constraint Has_PK primary key(E_ID, Dep_ID, B_ID),
	constraint Has_FK1 foreign key(E_ID) references Employee(E_ID),
	constraint Has_FK2 foreign key(Dep_ID) references Department(Dep_ID),
	constraint Has_FK3 foreign key(B_ID) references Branch(B_ID)

);




Create table Manager(
	Manager_ID varchar(10)not null,
	name varchar(30),
	email varchar(30),
	constraint Manager_PK primary key(Manager_ID),
	constraint Manager_CK check(email LIKE '%_@_%')

);




create table Attendance(
	attendance_ID varchar (10)not null,
	E_ID varchar(10)not null,
	date DATE not null,
	in_time TIME not null,
	out_time TIME not null,
	Manager_ID varchar(10)not null,
	constraint Attendance_PK primary key(attendance_ID),
	constraint Attendance_FK1 Foreign key (E_ID) references Employee (E_ID),
	constraint Attendance_FK2 Foreign key (Manager_ID) references Manager(Manager_ID)

);
```

```
create table LeaveType(
	Type_ID varchar (10)not null,
	name char(100),
	paid varchar(10),
	unpaid varchar(10),
	constraint LeaveType_PK primary key(Type_ID)

);




create table LeaveRequest(
	Leave_ID varchar(10)not null,
	from_date DATE not null,
	to_date DATE not null,
	reason varchar(30),
	status varchar(10),
	approved_date DATE not null,
	Manager_ID varchar(10),
	attendance_ID varchar(10),
	E_ID varchar(10)not null,
	Type_ID varchar(10)not null
	constraint LeaveRequest_PK primary key(Leave_ID),
	constraint LeaveRequest_FK1 Foreign key (Manager_ID) references Manager (Manager_ID),
	constraint LeaveRequest_FK2 Foreign key (attendance_ID) references Attendance
(attendance_ID),
	constraint LeaveRequest_FK3 Foreign key (E_ID) references Employee (E_ID),
	constraint LeaveRequest_FK4 Foreign key (Type_ID) references LeaveType (Type_ID)

);




create table Checks(
	attendance_ID varchar(10)not null,
	Leave_ID varchar(10)not null,
	constraint Check_PK primary key(Leave_ID),
	constraint Check_FK1 Foreign key (attendance_ID) references Attendance (attendance_ID),
	constraint Check_FK2 Foreign key (Leave_ID) references LeaveRequest (Leave_ID)


);
```

```sql
create table Manager_phone(
	Manager_ID varchar(10),
	phone_No char(30),
	constraint Manager_phone_PK primary key (Manager_ID,phone_No),
	constraint Manager_phone_FK foreign key (Manager_ID) references Manager(Manager_ID)


);




create table Salary(
	attendance_ID varchar(10)not null,
	name varchar(30),
	bonus real,
	monthly_pay real,
	OT_rate int,
	OT real,
	constraint Salary_PK primary key(attendance_ID),
	constraint Salary_FK Foreign key (attendance_ID) references Attendance (attendance_ID)
);
```

4. Query

a.) --------------/*Query 01*/--------------

Below query shows the employees who has taken more than 3 leaves.

```
SELECT E.E_ID, E.F_Name, D.D_Name AS 'Name of the Department', B.B_Name, COUNT(L.Leave_ID) AS
'Number of Leaves'
FROM Employee E, LeaveRequest L, Has H, Branch B, Department D
WHERE E.E_ID = L.E_ID AND
      H.B_ID = B.B_ID AND
      H.E_ID = E.E_ID AND
      H.Dep_ID = D.Dep_ID
GROUP BY E.E_ID,E.F_Name,D.D_Name, B.B_Name
Having COUNT(L.Leave_ID) > 3
```

b.) --------------/*Query 02*/--------------

Below query shows the all the approved leave requests.

```
SELECT E.E_ID, E.F_Name, E.L_Name, D.D_Name AS 'Name of the Department', B.B_Name AS 'Name of
the Branch'
FROM Employee E, Department D, Has H, Branch B
WHERE H.B_ID = B.B_ID AND
      H.Dep_ID = D.Dep_ID AND
      H.E_ID = E.E_ID AND
      E.E_ID IN (SELECT L.E_ID
                 FROM LeaveRequest L
                 WHERE L.Status = 'Approved')
```

5. Function/Procedure

Employees can view their leave details with following procedure.

(Employee should Input Employee id and password)

```
--------------/*CREATE PROCEDURE*/--------------




CREATE PROCEDURE getLeaveStatus
(
@eid varchar(10),
@password varchar(10),
@leaveId_ varchar(10) OUTPUT,
@type_ varchar(30) OUTPUT,
@status_ varchar(10) OUTPUT
)
AS
BEGIN

SELECT @leaveId_ = L.Leave_ID, @type_ = LT.name ,@status_ =L.status
FROM Employee E, LeaveRequest L,LeaveType LT
WHERE E.E_ID = L.E_ID AND
L.Type_ID = LT.Type_ID AND
E.E_ID = @eid AND
E.password = @password

END


DECLARE @employeeLeaveID varchar(10),@employeeLeaveType
varchar(30),@leaveStatus varchar(10)
EXEC getLeaveStatus 'E1001','qwer',
@leaveId_ = @employeeLeaveId OUTPUT ,
@type_ = @employeeLeaveType OUTPUT,
@status_ = @leaveStatus output
PRINT @employeeLeaveId
PRINT @employeeLeaveType
PRINT @leaveStatus
```

6. Trigger

An employee gets the message 'You can not apply for more leaves, it is greater than 5' when Employee applies more than five leaves.

```
--------------/*CREATE TRIGGER*/--------------



CREATE TRIGGER applyLeaveRequest
ON LeaveRequest
FOR INSERT,UPDATE
AS
Begin
	DECLARE @eid varchar(10)
	DECLARE @leaveCount int

	SELECT @eid = E_ID
	FROM Inserted

	SELECT @leaveCount = COUNT(LR.Leave_ID)
	FROM LeaveRequest LR
	WHERE @eid = LR.E_ID

	IF @leaveCount > 5
	BEGIN
		PRINT 'You can not apply for more leaves, it is greater than 5'
		ROLLBACK TRANSACTION
	END
END
```