Salary Management System

Mini Project Report

MSc. Mathematics and computing

Database Management Systems

Semester –2

Submitted By : Mudita Sharma (302303008)

Mentored By : DR. ANIL VASHISHT
Professor
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

Date: 9/05/2024

CERTIFICATE

This is to certify that the Mudita Sharma(302303008) have successfully executed a mini project titled "**Salary Management System**" rightly brining fore the competencies and skill sets they have gained during the course-Database Lab, thereby resulting in the culmination of this project.

DR. ANIL VASHISHT
Professor
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

Contents

INTRODUCTION

The "Salary Management System" project aims to simplify and improve how companies handle employee salaries. By using this software, companies can overcome the challenges of manual methods. The system is designed to fit each company's unique needs, making operations smoother and more effective.

This web application is easy to use and minimizes mistakes when entering data. It provides clear error messages for any incorrect inputs. No special knowledge is needed to operate it.

In today's fast-paced business world, managing human resources is crucial. As companies grow, the demand for efficient employee and payroll management systems increases. Our focus is on creating a simple version of such a system, emphasizing the importance of fast and efficient performance, especially in larger organizations

Synopsis

Proposed System

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. Having such a Management system is well in demand.

Objectives

- To view employee details
- To add details of employees or salaries
- To carry out transactions

Normalisation

The provided database schema is already in a normalized state, specifically in the Third Normal Form (3NF). Here's a breakdown of the normalization:

First Normal Form (1NF):

- Each table has a primary key, which ensures that each row has a unique identifier.
- Each column contains only atomic values (single values, not lists or groups).

Second Normal Form (2NF):

- Each non-key attribute (column) in a table depends on the entire primary key.
- In the Cart table, the quantity depends on both the Customer_customer_id and Product_product_id.
- In the Order_Item table, the quantity depends on both the Order_order_id and Product_product_id.

Third Normal Form (3NF):

- If a table has a composite primary key (multiple columns), each non-key attribute depends on the entire primary key.
- In the Cart table, the Customer_customer_id and Product_product_id together determine the quantity.
- In the Order_Item table, the Order_order_id and Product_product_id together determine the quantity.

ER DIAGRAM

The EMPLOYEE table is related to the SALARY table through the EMPLOYEE_SALARY table. This is a many-to-many relationship, because an employee can have multiple salaries, and a salary can be associated with multiple employees.

The EMPLOYEE table is related to the LEAVE table through the eid column. This is a one-to-many relationship, because an employee can have multiple leaves, but a leave can only be associated with one employee.

The EMPLOYEE table is related to the TRANSECTION table through the eid column. This is a one-to-many relationship, because an employee can have multiple transactions, but a transaction can only be associated with one employee.

The EMPLOYEE_SALARY table is related to the EMPLOYEE_SALARY_Audit table through the new_sid and old_sid columns. This is a one-to-many relationship, because a salary can have multiple audit records, but an audit record can only be associated with one salary.

DATABASE OBJECT INVOLVED

TABLES

- Employee (EID, EName, Gender, Email, JoinDate)
- Salary (SID, Basic, Allowance)
- Employee_Salary (EID(FK), SID(FK),)
- Leave (LID, EID(FK), L_month, L_days)
- Transection (TID, EID(FK), Amount, T_Date, S_month)

TRIGGERS

. eid_sid_restrict

. Check_eid_in_leave

PROCEDURES

. ChangeEmployeeSalary

. AddLeave

INDEXES

. Transaction_date_idx

. Date_of_join_idx

View

. transaction_view

. Employee_view

Sequence

. transaction_sequence

. employee_salary_sequence

Function

. insert_audit_salary

. add_transaction

IMPLEMENTATION

TABLE CREATION

SQL Worksheet

Clear Find Actions Save Run

```
CREATE TABLE EMPLOYEE (
eid int,
ename varchar2(20),
gender varchar2(5) check (gender in('M','F','Male','Female')),
email varchar2(25) check (email like '%@%'),
join_date varchar2(20) ,
PRIMARY KEY(eid));


CREATE TABLE SALARY(
sid int,
basic int,
allowance int,
PRIMARY KEY(sid));
```

```
Table created.

Table created.
```

SQL Worksheet

Clear Find Actions Save Run

```
CREATE TABLE EMPLOYEE_SALARY(
eid int,
sid int,
FOREIGN KEY(eid) REFERENCES EMPLOYEE(eid), FOREIGN KEY(sid) REFERENCES SALARY(sid));


CREATE TABLE LEAVE(
CREATE TABLE EMPLOYEE (
eid int,
ename varchar2(20),
gender varchar2(5) check (gender in('M','F','Male','Female')),
email varchar2(25) check (email like '%@%'),
join_date varchar2(20) ,
PRIMARY KEY(eid));

```

```
Table created.
```

SQL Worksheet

Clear Find A

```
5  CREATE TABLE TRANSECTION (
6  tid int,
7  eid int,
8  ammount int,
9  t_date date,
0  s_month varchar2(15),
1  PRIMARY KEY(tid),
2  FOREIGN KEY(eid) REFERENCES EMPLOYEE(eid));
3
4  CREATE TABLE EMPLOYEE_SALARY_Audit(
5  new_sid int,
6  old_sid int,
7  Changing_date varchar2(30));
8
```

able created.

able created.

SQL Worksheet

Clear

```
48
49 insert into employee values (1, 'Sajid Abdullah', 'M','sajid@gamil.com', '1/1/2019');
50 insert into employee values (2, 'Samia Zahan','F', 'samia@gamil.com', '1/1/2019');
51 insert into employee values (3, 'Muna Saha','F', 'muna@gmail.com', '1/1/2019');
52 insert into employee values (4, 'Robiul Hasan','M', 'nowshad@gamil.com', '1/1/2019');
53 insert into employee values (5, 'Apoorva Mitai','M', 'apoorv@yahoo.com', '1/1/2019');
54
55
56
```

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

INSERTION

SQL Worksheet

```
55
56
57  insert into salary values(1, 18000,5000);
58  insert into salary values(2, 20000,5000);
59  insert into salary values(3, 22000,6000);
60  insert into salary values(4, 35000,6500);
61  insert into salary values(5, 50000,7000);
62
63
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

SQL Worksheet

```
2
3
4  insert into employee_salary values(1,1);
5  insert into employee_salary values(2, 3);
6  insert into employee_salary values(3,5);
7  insert into employee_salary values(4,2);
8  insert into employee_salary values(5,1);
9
0
```

```
l row(s) inserted.

l row(s) inserted.

l row(s) inserted.

l row(s) inserted.
```

SQL Worksheet

```
70
71
72  insert into leave values(1,1, 'Jan/19', 3);
73  insert into leave values(2,3, 'Jan/19', 4);
74  insert into leave values(3,2, 'Jan/19', 5);
75  insert into leave values(4,6, 'Jan/19', 3);
76  insert into leave values(5,4, 'Jan/19', 1);
77
78
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

TRIGGER CREATION

SQL Worksheet

```
77
78  CREATE OR REPLACE TRIGGER eid_sid_restrict
79  BEFORE INSERT OR UPDATE OF eid, sid ON EMPLOYEE_SALARY
80  FOR EACH ROW
81  DECLARE
82      check_exists NUMBER(1);
83  BEGIN
84      SELECT 1 INTO check_exists FROM DUAL
85      WHERE EXISTS (
86          SELECT 1 FROM EMPLOYEE WHERE eid = :NEW.eid
87      ) AND EXISTS (
88          SELECT 1 FROM SALARY WHERE sid = :NEW.sid );
89  EXCEPTION
90      WHEN NO_DATA_FOUND THEN
91          dbms_output.put_line ('Invalid eid or sid');
92  END;
```

```
Trigger created.
```

```
126 CREATE OR REPLACE TRIGGER check_eid_in_leave
127 BEFORE INSERT ON LEAVE
128 FOR EACH ROW
129 DECLARE
130     check_exists NUMBER(1);
131 BEGIN
132     SELECT 1 INTO check_exists FROM DUAL
133     WHERE EXISTS (
134         SELECT 1 FROM EMPLOYEE WHERE eid = :NEW.eid
135     );
136 EXCEPTION
137     WHEN NO_DATA_FOUND THEN
138      dbms_output.put_line( 'invalid eid');
139 END;
140
```

```
Trigger created.
```

PROCEDURE CREATION

SQL Worksheet Clear

```
92  END;
93
94
95
96  CREATE OR REPLACE PROCEDURE ChangeEmployeeSalary(v_eid IN NUMBER, v_sid IN NUMBER)
97  IS
98  BEGIN
99      UPDATE EMPLOYEE_SALARY
100     SET sid = v_sid
101     WHERE eid = v_eid;
102 EXCEPTION
103     WHEN OTHERS THEN
104         dbms_output.put_line( 'An error occurred while changing the employee salary');
105 END;
106
107
```

```
Procedure created.
```

```
110
111
112
113 BEGIN
114     ChangeEmployeeSalary(4, 3);
115 END;
116
117
```

```
Statement processed.
```

SQL Worksheet

```
142
143
144 create or replace Procedure
145 AddLeave(v_lid in number, v_eid in number,v_l_month in varchar2,v_l_day in number)
146 is begin
147 insert into leave values(v_lid,v_eid, v_l_month, v_l_day);
148 end;
149
150
151
152
153
154
155
156
```

```
Procedure created.
```

```
154
155
156
157 BEGIN
158     AddLeave(100, 1, 'monday', 4);
159 END;
160
161
162
163
```

```
Statement processed.
```

INDEX CREATION

```
161 Create index
162     date_of_join_idx on
163        EMPLOYEE(join_date);
164
165 Create index
166     transaction_date_idx on
167         TRANSECTION(t_date );
168
169
```

```
Index created.

Index created.
```

VIEW CREATION

SQL Worksheet

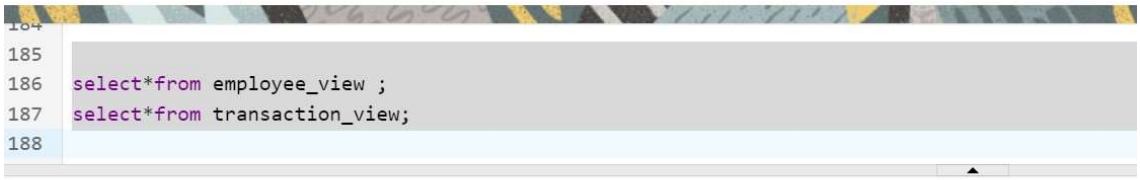
```
175
176 Create view employee_view as
177 Select eid ,ename, join_date
178 From employee;
179
180
181 Create view transaction_view as
182 Select tid ,eid, ammount
183 From transection;
184
185
```

```
View created.

View created.
```

VIEW DISPLAYED

SQL Worksheet

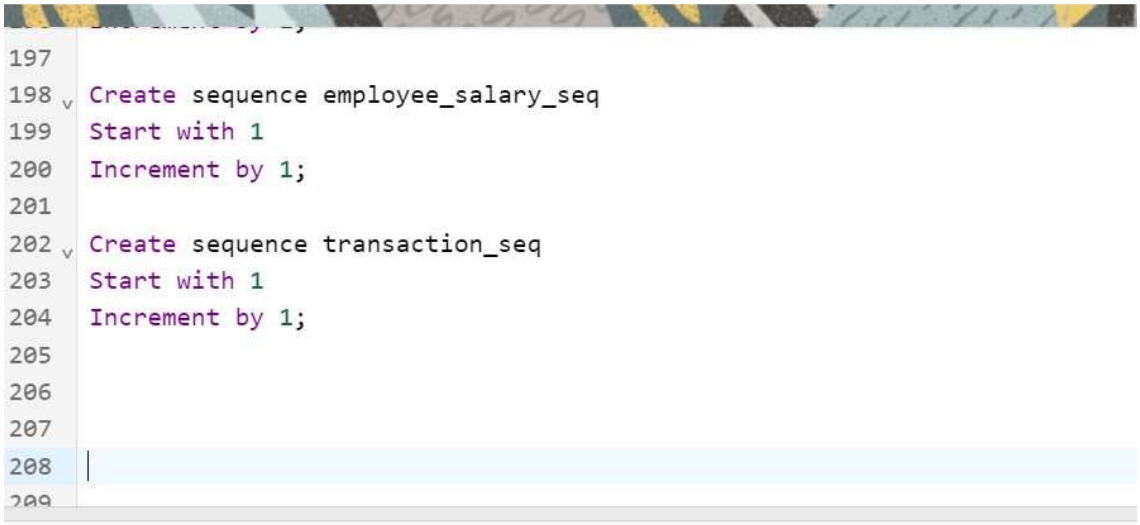


```
185
186 select*from employee_view ;
187 select*from transaction_view;
188
```

EID	ENAME	JOIN_DATE
1	Sajid Abdullah	1/1/2019
2	Samia Zahan	1/1/2019
3	Muna Saha	1/1/2019
4	Robiul Hasan	1/1/2019
5	Apoorva Mitai	1/1/2019

SEQUENCE CREATION

SQL Worksheet



```
197
198 Create sequence employee_salary_seq
199 Start with 1
200 Increment by 1;
201
202 Create sequence transaction_seq
203 Start with 1
204 Increment by 1;
205
206
207
208
```

```
Sequence created.

Sequence created.
```

FUNCTION CREATION

```
CREATE OR REPLACE FUNCTION insert_audit_salary(
    p_new_sid IN EMPLOYEE_SALARY_Audit.new_sid%TYPE,
    p_old_sid IN EMPLOYEE_SALARY_Audit.old_sid%TYPE,
    p_changing_date IN EMPLOYEE_SALARY_Audit.Changing_date%TYPE
) RETURN NUMBER IS
    v_audit_id EMPLOYEE_SALARY_Audit.audit_id%TYPE;
BEGIN
    v_audit_id := salary_audit_seq.NEXTVAL;
    INSERT INTO EMPLOYEE_SALARY_Audit (audit_id, new_sid, old_sid, Changing_date)
    VALUES (v_audit_id, p_new_sid, p_old_sid, p_changing_date);
    RETURN v_audit_id;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('An error occurred while adding the audit salary record: ' || SQLERRM);
        RETURN NULL;
END;
```

SQL Worksheet

```
238 CREATE OR REPLACE FUNCTION add_transaction(
239     p_eid IN EMPLOYEE.eid%TYPE,
240     p_amount IN TRANSECTION.ammount%TYPE,
241     p_t_date IN TRANSECTION.t_date%TYPE,
242     p_s_month IN TRANSECTION.s_month%TYPE
243 ) RETURN TRANSECTION.tid%TYPE IS
244     v_tid TRANSECTION.tid%TYPE;
245 BEGIN
246     v_tid := transaction_seq.NEXTVAL;
247  INSERT INTO TRANSECTION (tid, eid, ammount, t_date, s_month)
248     VALUES (v_tid, p_eid, p_amount, p_t_date, p_s_month);
249     RETURN v_tid;
250 EXCEPTION WHEN OTHERS THEN
251         dbms_output.put_line('An error occurred while adding the transaction: ' || SQLERRM);
252         RETURN NULL;
253 END;
```

```
Function created.
```

Conclusion

Established an idea of how distributed database work in real life scenario. Worked through the management system and discovered how the working takes place. We look forward to implement the project in future on a larger scale.