

Ahsanullah University of Science & Technology



Course Name: Distributed Database Systems Lab

Course No: CSE 4126

Project Title: Hospital Management System

Submitted to:

Mr. Mohammad Imrul Jubair
Assistant Professor, CSE, AUST.

Submitted by:

Mohsena Ashraf - 15.01.04.012

Other members:

Tasnim Mashrur Mahee - 15.01.04.013

Atiqul Islam Chowdhury -15.01.04.014

Objective:

Hospitals are the essential part of our lives, providing best medical facilities to people suffering from various ailments. It is necessary for the hospitals to keep track of its day-to-day activities and records of its personnel.

The main aim of our project is to provide a paper-less hospital as much as possible. We want to computerize all details regarding patient, doctor and hospital details. The information of the hospital should be kept up to date and should be kept in the system for historical purposes.

We have done this project to ensure multiple copies of data in different fragments, ensure data recovery, to take the advantage of powers and computer resources at each site considering the storage limitation.

Allocation schema:

Table Name	Attributes	Allocation
Patient	p_id, p_name, p_phone, gender, age, weight	Server
Doctor	d_id, d_name, department, d_phone	Server
Outpatient	p_id, d_id, o_date	Server
Labreport	Lab_no, p_id, p_name, d_id, disease, test_category, r_date, r_time	Server
Bill	bill_no, p_id, room_charge, medicine_charge, operation_charge, doctor_fee, type	Site
Inpatient	p_id, p_name, room_no, dateOfAdmission, dateOfDischarge	Site
Appointment	d_id, p_id, serial, a_date	Site
Room	p_id, room_no, status, type	Site

Fragmentation Schema:

Type of fragmentation : Horizontal fragmentation

Doctor₁: SL_{department="medicine"} Doctor (at site)

Doctor₂: SL_{department="cardiologist"} Doctor (at site)

Patient₁: SL_{age<15} Patient (at site)

Patient₂: SL_{age>=15} Patient (at site)

Packages:

1. Knn:

This package contains two procedures (knn_algorithm, show_result). It takes input of n (the number of 'k' in k-nearest-neighbor algorithm), room charge, medicine charge, operation charge and doctor fee from the command prompt and tells that if the calculated bill is cheaper or expensive, based on k nearest neighbors.

2. Linear_regression:

This package takes input of the age of a patient from the command prompt and estimates that what will be the patient's weight by applying linear regression algorithm.

It uses the following formula:

$$y = b_0 + b_1 * x$$

Where 'y' represents the patient's weight to be estimated and 'x' represents the patient's age which is given.

3. Algebraic_relation:

This package contains two procedures (algebraic_relation_left, algebraic_relation_right) and shows the output that proves the following rule of algebra of qualified relation:

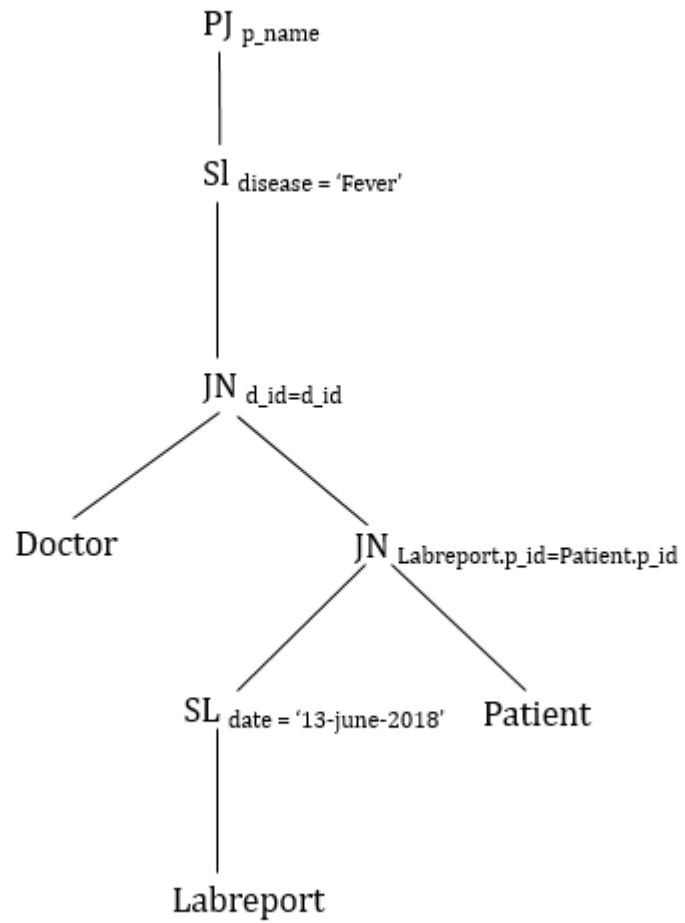
$$[R : qR] \text{ DF } [S : qS] \rightarrow [R \text{ DF } S : qR]$$

Where, R represents the fragment Doctor1 and S represents the fragment Doctor2.
qR: d_name='Dr. Mamun'
qS: d_name='Dr. Asif'

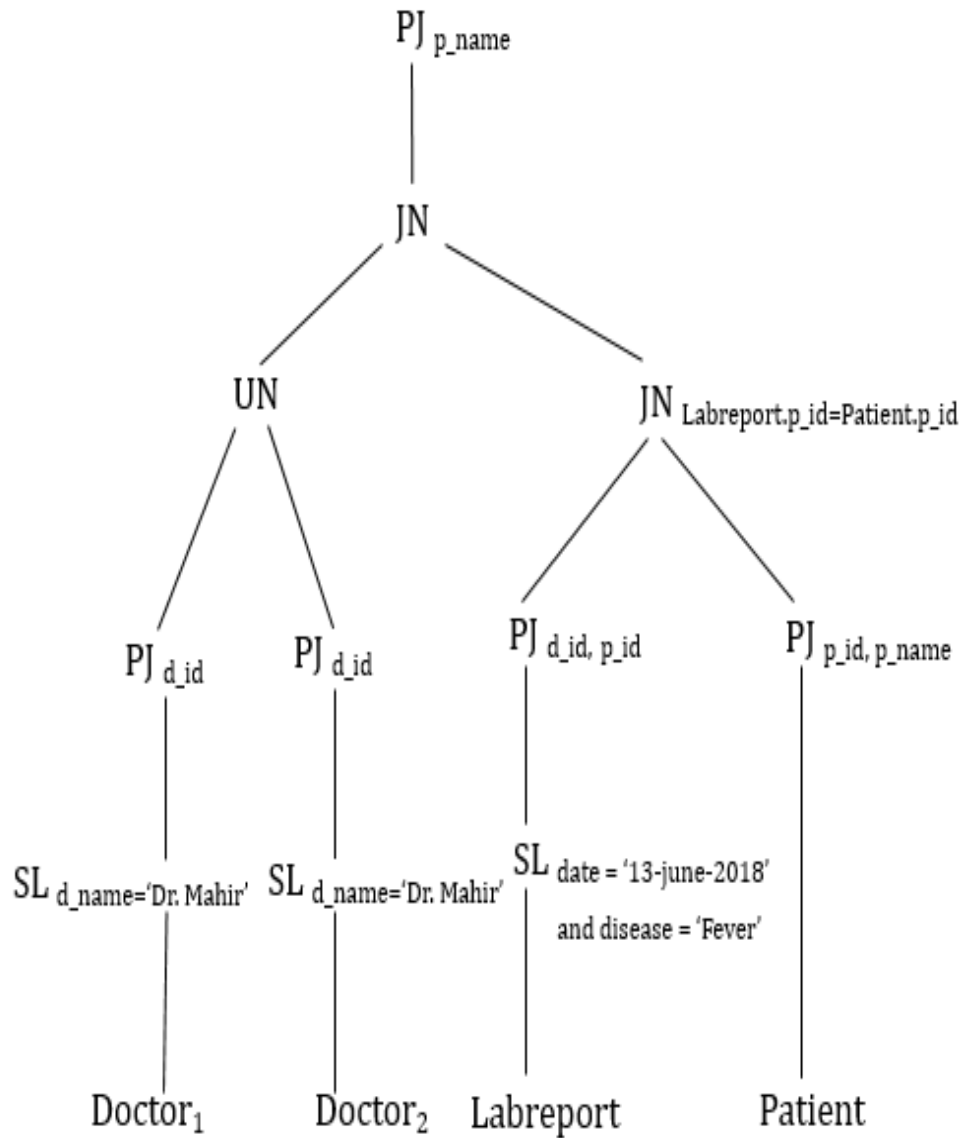
4. canonical_operator_simulation:

This package contains two procedures (operator_tree_canonical, operator_tree_simplified) and shows that result of the query of an operator tree and the result of that query with its simplified version with canonical expression yield to the same result.

The actual operator tree is given below:



The simplified operator tree with canonical expression is given below:



5. semi_join_proof:

This package contains two procedures (semi_join_left, semi_join_right) and shows that the given formula is true, i.e. the left hand side which contains a normal join operation and the right hand side which contains a semi join yields to the same result.

$$R \text{ JN }_{C=A} S \leftrightarrow (R \text{ SJ}_{C=A} \text{ PJ}_A S) \text{ JN}_{C=A} S$$

Where **R** represents the table 'labreport' and **S** represents the table 'Outpatient'.

Triggers:

1. insert_delete_update_for_trigger:

Input: An option to insert, delete or update; and a patient's id.

Output: This trigger works for the 'patient' table for insert, delete and update. It does the following operations:

- If the user selects 1, the patient info having the patient's id is inserted.
- If he selects 2, the patient having the given id is deleted.
- If he selects 3, the patients age having that id is updated.

2. Trigger_restricted:

This trigger doesn't take any input, but it works as a restriction for inserting, deleting or updating.

It restricts the option of manipulating the patient table. A user can manipulate patient table only between 4:00 am and 6:00 am.

3. Trigger_for_inserting:

This trigger works on inserting or deleting in the 'patient' table. If we insert the info of a patient in the 'patient' table whose age is less than 15, it is automatically inserted in 'patient1' fragment which contains patients having age less than 15.

Again, if we insert the info of a patient in the 'patient' table whose age is greater than or equal to 15, it is automatically inserted in 'patient2' fragment which contains patients having age greater than or equal to 15.

4. Trigger_for_updating:

This trigger works for updating on the 'patient' table. This has 4 cases:

- I. If the new age is less than 15, and the old age is less than 15, value is just updated on 'patient1' fragment automatically.

- II. If the new age is greater than or equal to 15, and the old age is less than 15, value is deleted from 'patient1' fragment and inserted on 'patient2' fragment automatically.
- III. If the new age is less than 15, and the old age is greater than or equal to 15, value is deleted from 'patient2' fragment and inserted on 'patient1' fragment automatically.
- IV. If the new age is greater than or equal to 15, and the old age is greater than or equal to 15, value is just updated on 'patient2' fragment automatically.

Functions:

1. **patientTimeRoom**(date1 in date, date2 in date,rno in int):

Inputs: Two dates, one room number.

Output: Shows the patient's name who was admitted in that room in that time interval.

2. **shift_appointment**(searchName in varchar2, date1 in date, date2 out date, patientId out number, doctorId out number):

Inputs: name of a doctor, an appointment date.

Output: shows the name of the patient, his id, and the id of that doctor. It also updates the appointment of that doctor and that patient to a given date.

Procedures:

1. **billsWithDiscount**(SearchPatientID IN int):

Inputs: Id of a patient.

Output: this procedure calculates the total bill of the patient. If the total bill is less than 15000, final bill is calculated with 10% discount; if it is in between 15000 to 30000, it calculates the final bill using 20%; if it is in between 30000 to 60000, it calculates the final bill using 30%, else it calculates the final bill using 50% discount.

2. **db_profile:**

This procedure proves the correctness of the estimating of profiles of results of join operation.

Let us consider a join operation:

$$T = R \text{ JOIN}_{R.A = S.B} S$$

Now 'db_profile' procedure proves the following formula:

$$\text{card}(T) = \rho * (\text{card}(R) \times \text{card}(S))$$

Where **R** represents the table 'inpatient' and **S** represents the table 'labreport'.

3. diseaseReportDate(searchName IN varchar2, searchDate in date):

Inputs: disease name (cancer), a date (18 january,2018).

Output: This procedure shows the name and admission date of the patients in the hospital who have the given disease and whose lab report was checked on that given date.

4. effect_of_update (doctor_id in int, dept in varchar2):

Inputs: The id of a doctor (9), his department (cardiologist).

Output: This procedure updates the department of the given doctor id and inserts it into the fragment having that department, deleting him from the previous fragment. This illustrates the effect of update for fragments.

5. insert_doctor1 (searchDept IN varchar2):

Inputs: A department of doctors (medicine).

Output: This procedure does the fragmentation of doctor table. It inserts the doctors having department 'medicine' into the *Doctor1* fragment.

6. insert_doctor2 (searchDept IN varchar2):

Inputs: A department of doctors (Cardiologist).

Output: This procedure does the fragmentation of doctor table. It inserts the doctors having department 'Cardiologist' into the 'Doctor2' fragment.

7. patientUnderDoctor(searchName IN varchar2):

Inputs: The name of a doctor (Dr. Mahir).

Output: It is a procedure which displays the name of the patients who have appointments under the given doctor.

My Contribution:

I have implemented and done the following things:

1. **Packages** for knn, linear regression, semi join, proof of algebraic relation, canonical expression and simplification of operator tree, database profile.
2. **Triggers** for trigger_restriction, insert on fragments on inserting the main table, update on fragments on updating the main table.
3. **Fragments** for Doctor₁ and Doctor₂.
4. **Functions** for shifting appointment.
5. **Procedures** for effect of update, calculating bills with vat and discount, finding the patients under a doctor.

Conclusion:

We have implemented all the features in our project. We will try to implement more machine learning algorithms such as logistic regression, decision tree etc in future.

Some screenshots and the ERD of our project are given on the next pages.

Screenshots of the project:

Some significant screenshots of our project are given below:

1. Knn:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Knn_package\knn.sql"
Enter the value of n: 5
Enter the room_charge: 45000
Enter the medicine_charge: 10000
Enter the operation_charge: 150000
Enter the doctor_fee: 2000

Package created.

Package body created.

4 expensive
3 expensive
2 cheaper
5 expensive
1 cheaper
The new bill type is expensive

PL/SQL procedure successfully completed.
```

2. Linear Regression

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Linear_regression package\linear_regression.sql"
Enter the age of the patient: 50

Package created.

Package body created.

140 398 3078 7979 7 .07 55.46 58.96
Predicted weight of a patient having age 50 is: 58.96

PL/SQL procedure successfully completed.
```

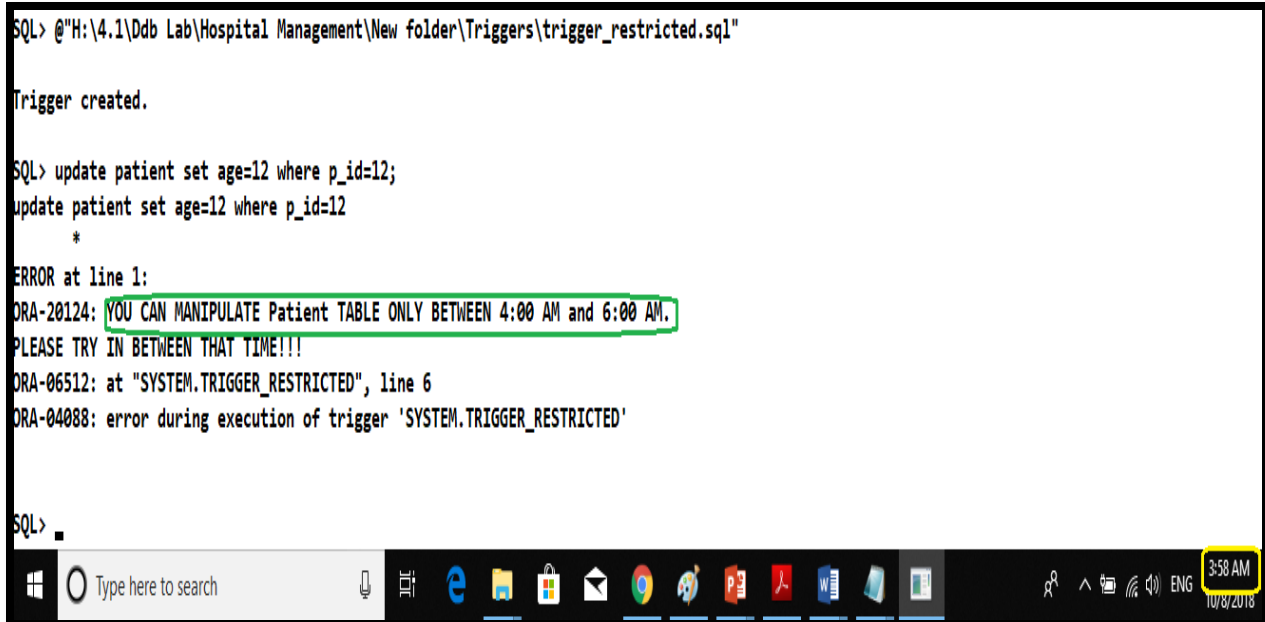
3. Trigger_restricted:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Triggers\trigger_restricted.sql"

Trigger created.

SQL> update patient set age=12 where p_id=12;
update patient set age=12 where p_id=12
*
ERROR at line 1:
ORA-20124: YOU CAN MANIPULATE Patient TABLE ONLY BETWEEN 4:00 AM and 6:00 AM.
PLEASE TRY IN BETWEEN THAT TIME!!!
ORA-06512: at "SYSTEM.TRIGGER_RESTRICTED", line 6
ORA-04088: error during execution of trigger 'SYSTEM.TRIGGER_RESTRICTED'

SQL> .
```



4. canonical_operator_simulation:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Canonical expression and simplified operator tree
e_package\Canonical Expression and operator tree simplified_package.sql"

Package created.

Package body created.

FOR ACTUAL QUERY, THE RESULT IS:
Result Name is : Mahee Mashrur
FOR SIMPLIFIED OPERATOR TREE WITH CANONICAL EXPRESSION, THE RESULT IS:
Result Name is : Mahee Mashrur

PL/SQL procedure successfully completed.
```

5. semi_join_proof:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\semi join\semi join proof.sql"

Package created.

Package body created.

LAB_NO PATIENT ID(L) PATIENT NAME DOCTOR_ID(L) DISEASE TEST_CATEGORY DATE(L) TIME PATIENT ID(0) DOCTOR ID(0) DATE(0)
1 1 Mohsena Ashraf 1 cancer Blood test 10-JAN-18 03:35 pm 1 2 10-JAN-18
2 1 Mohsena Ashraf 2 tumor Blood test 10-JAN-18 03:45 pm 1 2 10-JAN-18
3 2 Mahee Mashrur 2 fever Blood test 13-JAN-18 03:57 pm 2 2 16-JAN-18
4 3 Atiqul islam 1 cancer Blood test 18-JAN-18 04:21 pm 3 1 15-JAN-18
5 3 Atiqul islam 1 tumor Blood test 21-JAN-18 04:45 pm 3 1 15-JAN-18
6 3 Atiqul islam 1 tumor Blood test 21-JAN-18 04:45 pm 3 1 15-JAN-18
LAB_NO PATIENT ID(L) PATIENT NAME DOCTOR_ID(L) DISEASE TEST_CATEGORY DATE(L) TIME PATIENT ID(0) DOCTOR ID(0) DATE(0)
1 1 Mohsena Ashraf 1 cancer Blood test 10-JAN-18 03:35 pm 1 2 10-JAN-18
2 1 Mohsena Ashraf 2 tumor Blood test 10-JAN-18 03:45 pm 1 2 10-JAN-18
3 2 Mahee Mashrur 2 fever Blood test 13-JAN-18 03:57 pm 2 2 16-JAN-18
4 3 Atiqul islam 1 cancer Blood test 18-JAN-18 04:21 pm 3 1 15-JAN-18
5 3 Atiqul islam 1 tumor Blood test 21-JAN-18 04:45 pm 3 1 15-JAN-18
6 3 Atiqul islam 1 tumor Blood test 21-JAN-18 04:45 pm 3 1 15-JAN-18

PL/SQL procedure successfully completed.
```

Activate Windows
Go to Settings to activate Windows.

6. Algebric_relation:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\algebric_relation_proof_package\AlgebricRelation
_package.sql"

Package created.

Package body created.

FOR LEFT HAND SIDE:
Doctor Id: 11 and Doctor Name: Dr. Mamun
Doctor Id: 12 and Doctor Name: Dr. Mamun
Doctor Id: 13 and Doctor Name: Dr. Mamun
Doctor Id: 14 and Doctor Name: Dr. Mamun
FOR RIGHT HAND SIDE:
Doctor Id: 11 and Doctor Name: Dr. Mamun
Doctor Id: 12 and Doctor Name: Dr. Mamun
Doctor Id: 13 and Doctor Name: Dr. Mamun
Doctor Id: 14 and Doctor Name: Dr. Mamun

PL/SQL procedure successfully completed.
```

7. insert_delete_update_for_trigger:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Triggers\insert_delete_update_for_trigger.sql"
Press 1 to INSERT , Press 2 to DELETE: , Press 3 to UPDATE: 2
Enter the patient id to be inserted or deleted or updated: 7
Value deleted from Patient1.

PL/SQL procedure successfully completed.

Commit complete.
```

8. Trigger_for_inserting:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Triggers\Trigger_for_inserting.sql"
Trigger created.

SQL> insert into patient values(8,'Mahiya chowdhury',01961696961,'Female',27,59);
New value inserted in Patient2.

1 row created.
```

9. Trigger_for_updating:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Triggers\trigger_for_updating.sql"
```

```
Trigger created.
```

```
SQL> update patient set age=10 where p_id=7;
```

```
VALUE UPDATED FOR PATIENT1
```

```
1 row updated.
```

10. patientTimeRoom:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Functions\function patientTimeRoom.sql"
```

```
Function created.
```

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Functions\function patientTimeRoom_main.sql"
```

```
The patient who was admitted is:
```

```
patient Name is: Shovan Chowdhury
```

```
PL/SQL procedure successfully completed.
```

10. shift_appointment:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Functions\ShiftingAppointment.sql"

Function created.

SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Functions\shiftingAppointment_main.sql"
Shifted patient information is given below:
patients NAmE is: Shovan Chowdhury  20-JUL-18  3 4

PL/SQL procedure successfully completed.
```

11. billsWithDiscount:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Procedures
\billsWithVat.sql"

Procedure created.

Total bill: 51450   Final Bill: 36015

PL/SQL procedure successfully completed.

SQL>
```

12. db_profile:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\functions and procedures\Procedure\db_profile.sql"

Procedure created.

Patient ID: 2   and Patient Name: Mahee Mashrur and disease: fever
Patient ID: 3   and Patient Name: Atiqul islam and disease: tumor
Patient ID: 3   and Patient Name: Atiqul islam and disease: tumor
Patient ID: 3   and Patient Name: Atiqul islam and disease: cancer
Patient ID: 1   and Patient Name: Mohsena Ashraf and disease: tumor
Patient ID: 1   and Patient Name: Mohsena Ashraf and disease: cancer
card of result table is: 6
card of R(Inpatient) is:8
card of S(Labreport) is:6
distinct-VAL(A[R]) is:8
card (T) = row * (card(R) * card (S)) is proved.

PL/SQL procedure successfully completed.
```

13. patientUnderDoctor:

```
SQL> @"H:\4.1\Ddb Lab\Hospital Management\New folder\Procedures\patientUnderDoctor.sql"

Procedure created.

The patients under the doctor is:
Mohsena Ashraf
Mahee Mashrur
Pia Chowdhury

PL/SQL procedure successfully completed.
```


ERD of Hospital Management System

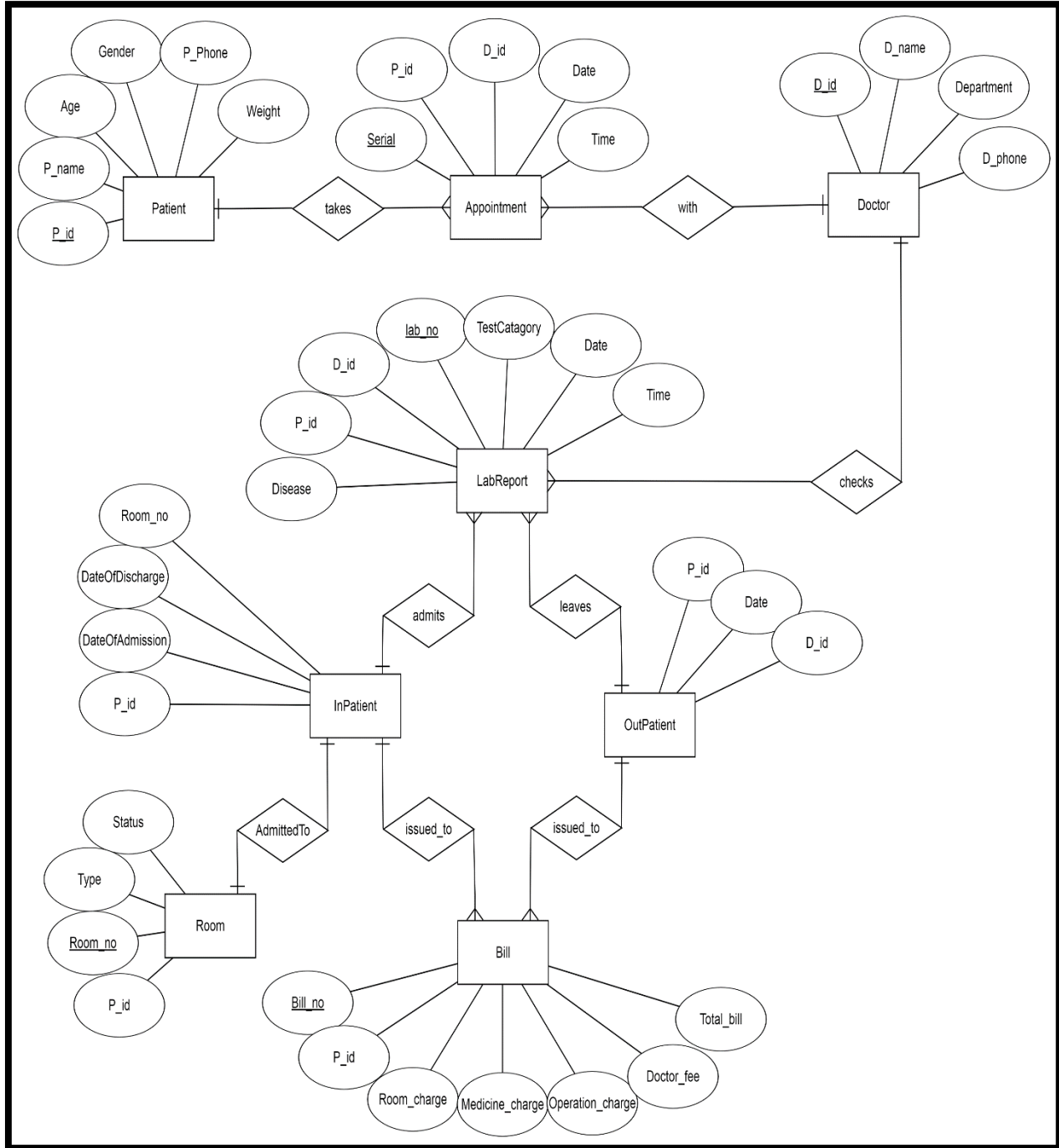


Fig: ERD of Hospital Management System