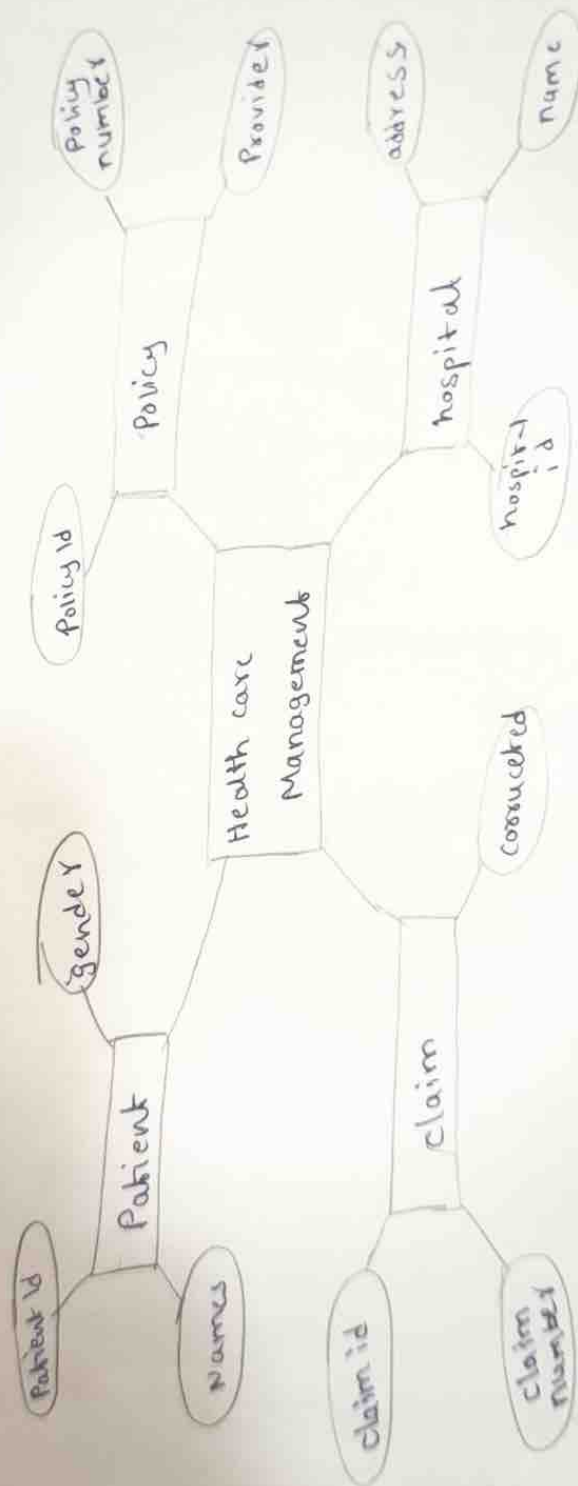


Task-12  
Mini-Project

ER Diagram



## 2.1 Normalization steps:-

First Normal form:- Ensure atomic attribute values for each <sup>tuple</sup>

Second Normal form:- Remove partial dependencies; each non-key attributes must depend on the whole primary key

Third Normal form:- Remove transitive dependencies; non-key attrib: depend only on the primary key

Candidate keys examples

- Patient:- patientid (unique)
- Policy:- policy Number (unique)
- Claim:- claimid (unique)
- Hospital:- Hospitalid (unique)

## 2.2 SQL Queries to process, approve and track insurance claims

• Insert new claim;

```
INSERT INTO claim (claimid, policyid, patientid,  
hospitalid, claim Number, amount)
```

```
VALUES (1001, 'PN123', 1, 101, 22, 2000)
```

• Approve a claim

```
update claim SET status = 'Approval';
```

```
Approval Date = '2023-02-01' WHERE claimid = 1001;
```

• Track claims for a given policy:

```
SELECT * FROM claim WHERE policy Number = 'PN123';
```

• Calculate total approved claims per patient:

```
SELECT patientid, sum(amount) AS Total Approved
```

```
FROM claim
```

```
WHERE status = 'Approved'
```

```
GROUP BY - Patientid;
```

### 3 Transaction Management during concurrent claim approvals

#### 1. optimistic concurrency control

UPDATE claim

SET status = 'Approved', version = version + 1

WHERE claim\_id = 'uid-123u' AND status = 'IN-Review'  
AND version = 3;

#### 2. pessimistic locking

BEGIN;

SELECT status FROM claim WHERE claim-id = 'uid-123u'  
FOR UPDATE; UPDATE claim SET status = 'Approved' WHERE  
claim-id = 'uid-123u'; COMMIT;

#### 3. Idempotency & business-level compensation

Make approval operations idempotent

SUBMITTED → IN-Review → Approved.

#### 4. isolation levels

Use read committed typically, or REPEATABLE READ

SERIALIZE if strict serializability is required

### 4. Performing CRUD operations in MongoDB to Manage claims records.

• Basic MongoDB CRUD operations for claim records:-

• Create:-

db.claims.insertOne({

claimid: 1001,

Policy Number = 'PN123';

Patient id = 1,

```

"timestamp": "2025-01-15 10:30:00",
"sensors": {
  "temperature": 23.5,
  "humidity": 45.5,
  "status": "active",
  "metadata": {
    "manufacturer": "AcmeDevices",
    "firmware version": "1.2.5"
  }
}

```

### How indexing steps

db: devices data - create index [ { "device id": 1 } ];  
 db: device Data - Create index [ { "location - 'location id'": 1 } ];  
 db: devices Data - Create index [ { "device id": 1 } "location - 'location id'": 1 } ];

### Benefits

- Fetch all records of a device in different location
- Query all devices in a location
- Retrieve temperature readings across all thermostats

### Result

Thus the user case is successfully verified and executed successfully