**CSA0593**

**DATABASE MANAGEMENT SYSTEM**

*PRAKALYA P V*

*192311418*

*ASSIGNMENT - 3*

Healthcare Insurance Claims Processing System Database

- Design tables for patients, providers, claims, payments, and insurance plans.

- Implement stored procedures for claims submission, processing, and payment.

- Develop SQL queries for reporting claims status, provider performance, and payment summaries.

- Implement ACID properties to ensure reliable data management.

- Test using various failure scenarios.

1). Below is a database design for a **Healthcare Insurance Claims Processing System** with tables for managing **patients, providers, claims, payments,** and **insurance plans**. This design will support the submission, processing, and payment of healthcare insurance claims.

**Database Design:**

1. **Table for Patients:-**

CREATE TABLE Patients (

patient\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

date\_of\_birth DATE NOT NULL,

gender CHAR(1) CHECK (gender IN ('M', 'F', 'O')), -- M=Male, F=Female, O=Other

insurance\_plan\_id INT,

address VARCHAR(255),

phone\_number VARCHAR(20),

email VARCHAR(100),

FOREIGN KEY (insurance\_plan\_id) REFERENCES InsurancePlans(insurance\_plan\_id) ON DELETE SET NULL

);

1. **Table for Providers:-**

CREATE TABLE Providers (

provider\_id INT PRIMARY KEY AUTO\_INCREMENT,

provider\_name VARCHAR(100) NOT NULL,

specialty VARCHAR(50),

contact\_number VARCHAR(20),

email VARCHAR(100),

address VARCHAR(255),

provider\_type ENUM('Hospital', 'Clinic', 'Individual', 'Pharmacy') NOT NULL

);

1. **Table for Insurance Plans:-**

CREATE TABLE InsurancePlans (

insurance\_plan\_id INT PRIMARY KEY AUTO\_INCREMENT,

plan\_name VARCHAR(100) NOT NULL,

coverage\_percentage DECIMAL(5, 2) NOT NULL CHECK (coverage\_percentage BETWEEN 0 AND 100),

deductible DECIMAL(10, 2) NOT NULL,

max\_coverage\_amount DECIMAL(15, 2) NOT NULL,

description TEXT

);

**4.Table for Claims:-**

CREATE TABLE Claims (

claim\_id INT PRIMARY KEY AUTO\_INCREMENT,

patient\_id INT NOT NULL,

provider\_id INT NOT NULL,

service\_date DATE NOT NULL,

claim\_amount DECIMAL(15, 2) NOT NULL,

status ENUM('Submitted', 'Processing', 'Approved', 'Rejected', 'Paid') DEFAULT 'Submitted',

insurance\_coverage\_amount DECIMAL(15, 2),

patient\_responsible\_amount DECIMAL(15, 2),

submission\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id) ON DELETE CASCADE,

FOREIGN KEY (provider\_id) REFERENCES Providers(provider\_id) ON DELETE SET NULL

);

**5.Table for Payments:-**

CREATE TABLE Payments (

payment\_id INT PRIMARY KEY AUTO\_INCREMENT,

claim\_id INT NOT NULL,

payment\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

amount DECIMAL(15, 2) NOT NULL,

payment\_method ENUM('Bank Transfer', 'Check', 'Credit Card') NOT NULL,

FOREIGN KEY (claim\_id) REFERENCES Claims(claim\_id) ON DELETE CASCADE

);

**RELATIONS:-**

* **Patients and Providers**: Patients receive healthcare services from providers. Each claim ties a patient to a provider, helping track who rendered the service.
* **Insurance Plans and Claims**: Each patient is associated with an insurance plan. The insurance plan determines the coverage percentage, deductible, and maximum coverage amount for claims. These details are used to calculate the insurance and patient portions for each claim.
* **Claims and Payments**: A claim may result in one or multiple payments. Payments table links each payment to a specific claim, capturing the details of financial transactions for each claim.

2). Below are stored procedures for **claims submission**, **claims processing**, and **claims payment**. These procedures will automate the claim-related processes, such as submitting claims, processing them (approving/rejecting), and making payments.

**Stored Procedure for Claims Submission.**

DELIMITER $$

CREATE PROCEDURE SubmitClaim (

IN p\_patient\_id INT,

IN p\_provider\_id INT,

IN p\_service\_date DATE,

IN p\_claim\_amount DECIMAL(15, 2)

)

BEGIN

DECLARE v\_coverage\_percentage DECIMAL(5, 2);

DECLARE v\_deductible DECIMAL(10, 2);

DECLARE v\_max\_coverage DECIMAL(15, 2);

DECLARE v\_insurance\_coverage DECIMAL(15, 2);

DECLARE v\_patient\_responsible DECIMAL(15, 2);

-- Fetch insurance plan details

SELECT coverage\_percentage, deductible, max\_coverage\_amount

INTO v\_coverage\_percentage, v\_deductible, v\_max\_coverage

FROM Patients p

JOIN InsurancePlans i ON p.insurance\_plan\_id = i.insurance\_plan\_id

WHERE p.patient\_id = p\_patient\_id;

-- Calculate insurance coverage and patient responsibility

IF p\_claim\_amount > v\_deductible THEN

SET v\_insurance\_coverage = LEAST(p\_claim\_amount - v\_deductible, v\_max\_coverage) \* (v\_coverage\_percentage / 100);

SET v\_patient\_responsible = p\_claim\_amount - v\_insurance\_coverage;

ELSE

SET v\_insurance\_coverage = 0;

SET v\_patient\_responsible = p\_claim\_amount;

END IF;

-- Insert the claim into Claims table

INSERT INTO Claims (patient\_id, provider\_id, service\_date, claim\_amount, insurance\_coverage\_amount, patient\_responsible\_amount)

VALUES (p\_patient\_id, p\_provider\_id, p\_service\_date, p\_claim\_amount, v\_insurance\_coverage, v\_patient\_responsible);

END $$

DELIMITER ;

**Stored Procedure for Claims Processing.**

DELIMITER $$

CREATE PROCEDURE ProcessClaim (

IN p\_claim\_id INT

)

BEGIN

DECLARE v\_claim\_amount DECIMAL(15, 2);

DECLARE v\_max\_coverage DECIMAL(15, 2);

DECLARE v\_patient\_id INT;

DECLARE v\_status ENUM('Submitted', 'Processing', 'Approved', 'Rejected', 'Paid');

-- Fetch the claim details

SELECT claim\_amount, patient\_id

INTO v\_claim\_amount, v\_patient\_id

FROM Claims

WHERE claim\_id = p\_claim\_id;

-- Fetch the insurance plan details (e.g., max coverage amount)

SELECT max\_coverage\_amount

INTO v\_max\_coverage

FROM Patients p

JOIN InsurancePlans i ON p.insurance\_plan\_id = i.insurance\_plan\_id

WHERE p.patient\_id = v\_patient\_id;

-- Determine claim status based on amount and max coverage

IF v\_claim\_amount <= v\_max\_coverage THEN

SET v\_status = 'Approved';

ELSE

SET v\_status = 'Rejected';

END IF;

-- Update claim status

UPDATE Claims

SET status = v\_status

WHERE claim\_id = p\_claim\_id;

-- Optionally, update patient last activity (or create a trigger for this)

UPDATE Users

SET last\_activity = NOW()

WHERE user\_id = v\_patient\_id;

END $$

DELIMITER ;

**Stored Procedure for Claims Payment.**

DELIMITER $$

CREATE PROCEDURE ProcessPayment (

IN p\_claim\_id INT,

IN p\_payment\_method ENUM('Bank Transfer', 'Check', 'Credit Card'),

IN p\_payment\_amount DECIMAL(15, 2)

)

BEGIN

DECLARE v\_claim\_amount DECIMAL(15, 2);

DECLARE v\_patient\_responsible DECIMAL(15, 2);

DECLARE v\_insurance\_coverage DECIMAL(15, 2);

DECLARE v\_status ENUM('Paid', 'Processing', 'Approved', 'Rejected');

-- Fetch claim details

SELECT claim\_amount, patient\_responsible\_amount, insurance\_coverage\_amount, status

INTO v\_claim\_amount, v\_patient\_responsible, v\_insurance\_coverage, v\_status

FROM Claims

WHERE claim\_id = p\_claim\_id;

-- Ensure the claim is approved before payment

IF v\_status = 'Approved' THEN

-- Insert payment record

INSERT INTO Payments (claim\_id, payment\_date, amount, payment\_method)

VALUES (p\_claim\_id, NOW(), p\_payment\_amount, p\_payment\_method);

-- Update claim status to Paid

UPDATE Claims

SET status = 'Paid'

WHERE claim\_id = p\_claim\_id;

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Claim is not approved or is in invalid state for payment.';

END IF;

END $$

DELIMITER ;

* These stored procedures help automate critical processes in the healthcare insurance claims workflow, ensuring that claims are handled efficiently and accurately.

3). Here are several SQL queries for reporting on **claims status**, **provider performance**, and **payment summaries** within a Healthcare Insurance Claims Processing System. These reports will help administrators and managers track the status of claims, analyse the performance of providers, and summarize payment information.

**1. Query for Reporting Claims Status**

This query generates a summary of claims grouped by their status (Submitted, Processing, Approved, Rejected, Paid). It provides the total number of claims and the total claim amounts for each status.

SELECT

status AS claim\_status,

COUNT(\*) AS total\_claims,

SUM(claim\_amount) AS total\_claim\_amount,

SUM(insurance\_coverage\_amount) AS total\_insurance\_coverage,

SUM(patient\_responsible\_amount) AS total\_patient\_responsible

FROM Claims

GROUP BY status

ORDER BY total\_claims DESC;

### 2. ****Query for Reporting Provider Performance****

This query provides a report on how each healthcare provider is performing by summarizing the number of claims they’ve processed, the total claim amounts, and the total insurance coverage amounts.

SELECT

p.provider\_name,

COUNT(c.claim\_id) AS total\_claims,

SUM(c.claim\_amount) AS total\_claim\_amount,

SUM(c.insurance\_coverage\_amount) AS total\_insurance\_coverage,

AVG(c.insurance\_coverage\_amount) AS avg\_coverage\_per\_claim

FROM Claims c

JOIN Providers p ON c.provider\_id = p.provider\_id

GROUP BY p.provider\_name

ORDER BY total\_claims DESC;

### 3. ****Query for Reporting Payment Summaries****

This query provides a summary of payments made, including the total payment amount, grouped by payment method (Bank Transfer, Check, Credit Card). It shows the total payments for each method.

SELECT

payment\_method,

COUNT(\*) AS total\_payments,

SUM(amount) AS total\_payment\_amount,

AVG(amount) AS avg\_payment\_amount

FROM Payments

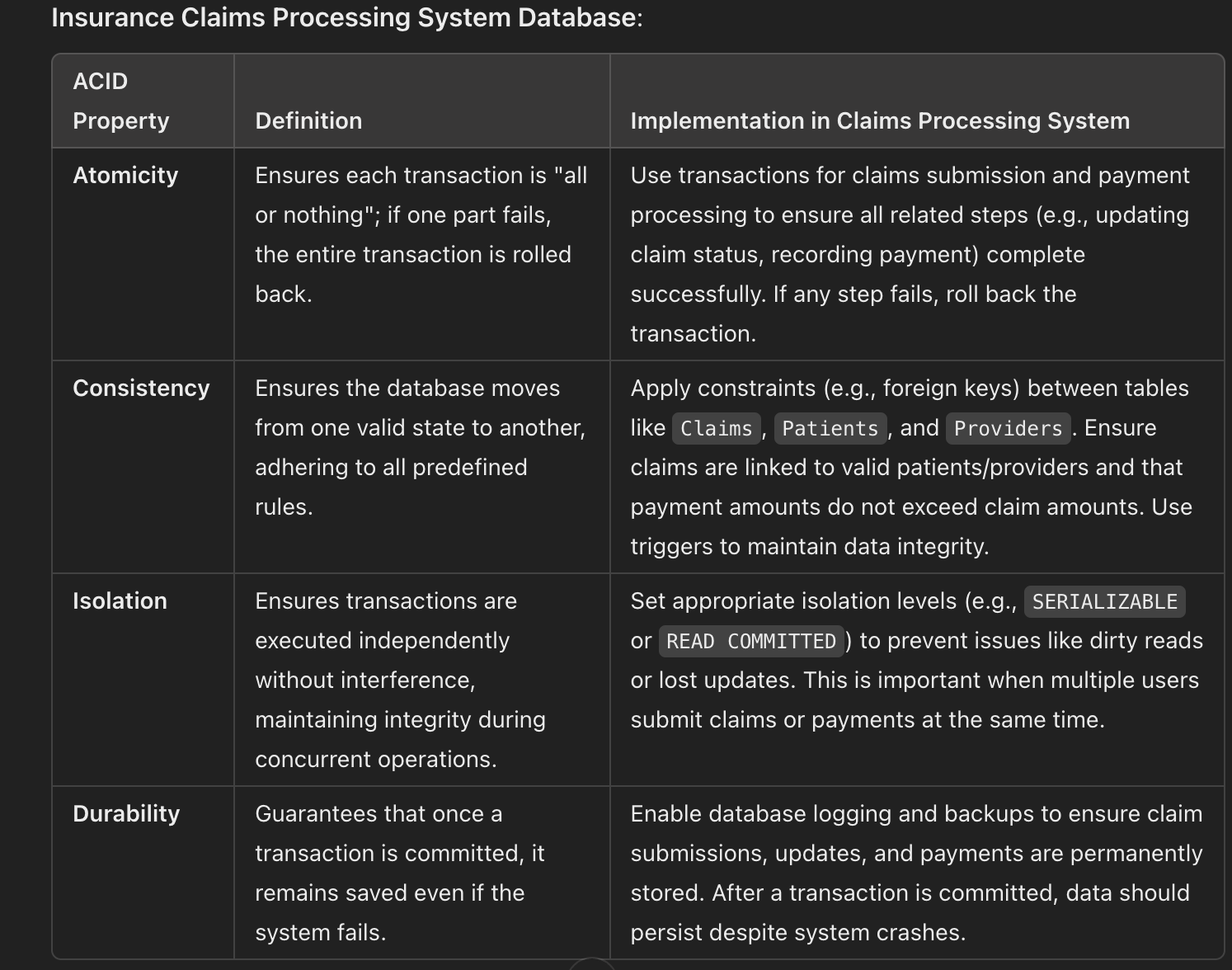
GROUP BY payment\_method

ORDER BY total\_payment\_amount DESC;

* These SQL queries allow you to generate various reports that help administrators manage healthcare insurance claims, track provider performance, and summarize payment activity. They offer insights into the overall claims process, from submission to payment, and are critical for effective decision-making, billing, and auditing within a healthcare insurance system.

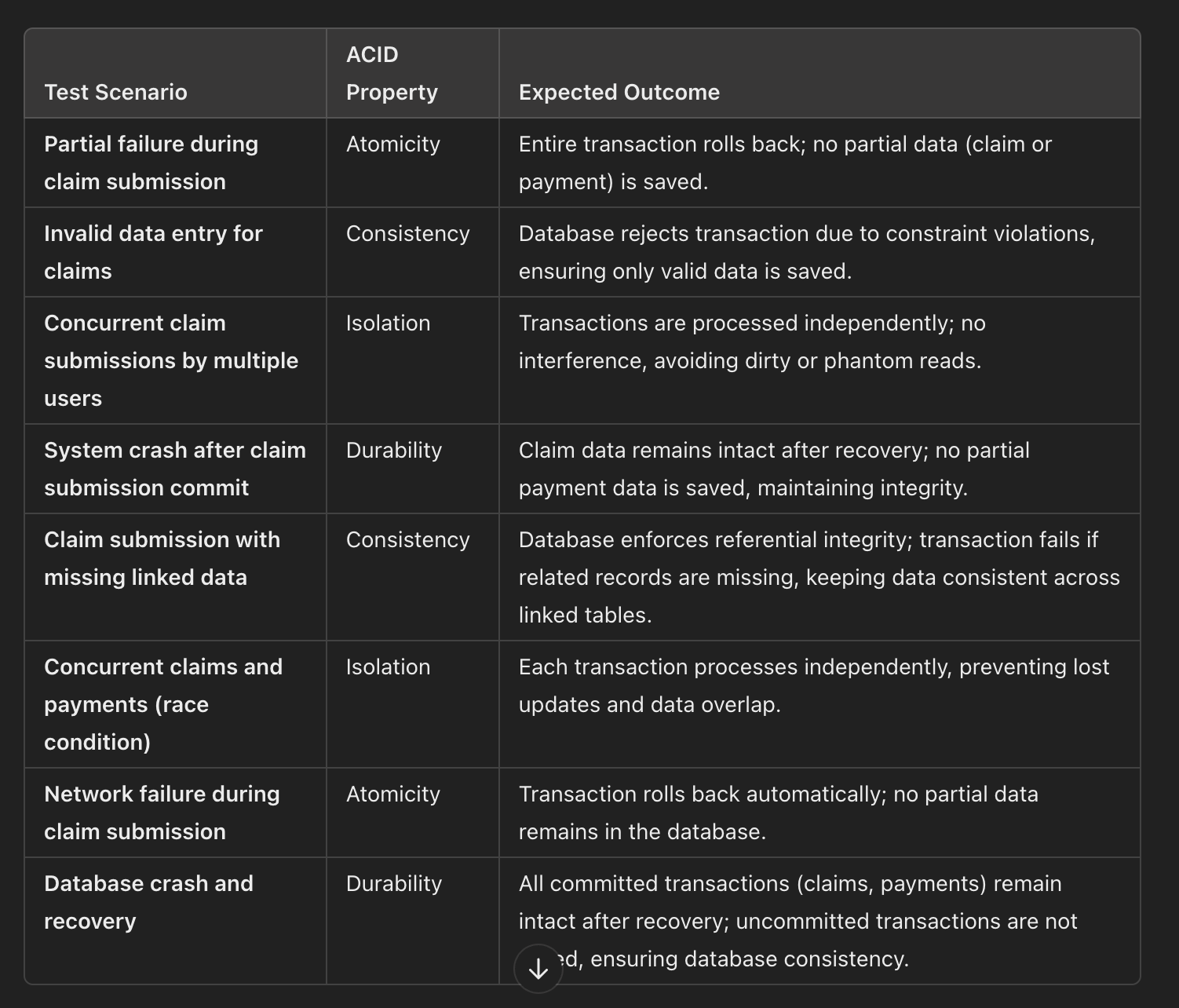
4) **Implement ACID properties to ensure reliable data management** :

To ensure reliable data management in a **Healthcare Insurance Claims Processing System Database**, implementing **ACID properties** is essential.



5.) **Test using various failure scenarios:**

To ensure the **Healthcare Insurance Claims Processing System Database** meets ACID properties, we can simulate various failure scenarios



**CONCLUSION :**

In conclusion, implementing and rigorously testing ACID properties in the **Healthcare Insurance Claims Processing System Database** ensures a robust and reliable system for managing sensitive healthcare data. By addressing **Atomicity**, **Consistency**, **Isolation**, and **Durability**, the database can handle various failure scenarios gracefully, maintaining data integrity and reliability even under unexpected conditions. Atomicity guarantees that each transaction completes fully or not at all, avoiding partial data issues. Consistency enforces valid states by upholding constraints and relationships across tables, ensuring data accuracy. Isolation enables concurrent processing without conflicts, essential for high-transaction environments, while Durability ensures data permanence and recoverability in case of system failures.

These ACID-compliant strategies provide a strong foundation for secure, dependable claims processing, making the system well-suited for critical healthcare operations where data integrity is paramount.