DBMS Project Report

PES University

Database Management Systems

UE18CS252

Insurance Management System

Submitted By

Roshini Bhaskar K
PES2201800122

Insurance, by definition, is the group of non-life insurance policies that provide insurance cover for motor vehicles, houses, health, and travel. This system is recreated and managed using MySQL. The are 6 entities namely agent, customer, policy, premium, paid_premium and unpaid_premium which exhibit one-to-many, many-to-one, weak and many relationships. All the relations are checked for their normalization form and potential violations are discussed.

TABLE OF CONTENTS

1. INTRODUCTION

The world we live in is full of uncertainties and risks. Individuals, families, businesses, properties and assets are exposed to different types and levels of risks. These include risk of losses of life, health, assets, property, etc. While it is not always possible to prevent unwanted events from occurring, financial world has developed products that protect individuals and businesses against such losses by compensating them with financial resources. Insurance is a financial product that reduces or eliminates the cost of loss or effect of loss caused by different types of risks.

Apart from protecting individuals and businesses from many kinds of potential risks, the Insurance sector contributes significantly to the general economic growth of the nation by providing stability to the functioning of businesses and generating long-term financial resources for the industrial projects. Among other things, Insurance sector also encourages the virtue of savings among individuals and generates employments for millions, especially in a country like India, where savings and employment are important.

General insurance, by definition, is the group of non-life insurance policies that provide insurance cover for motor vehicles, houses, health, and travel. These policies have to be renewed according to the tenure prescribed in their respective policy documents, and these have a particular sum insured which is reimbursed depending on the loss from a particular financial event. In some countries like US, Dubai having health insurance is mandatory.

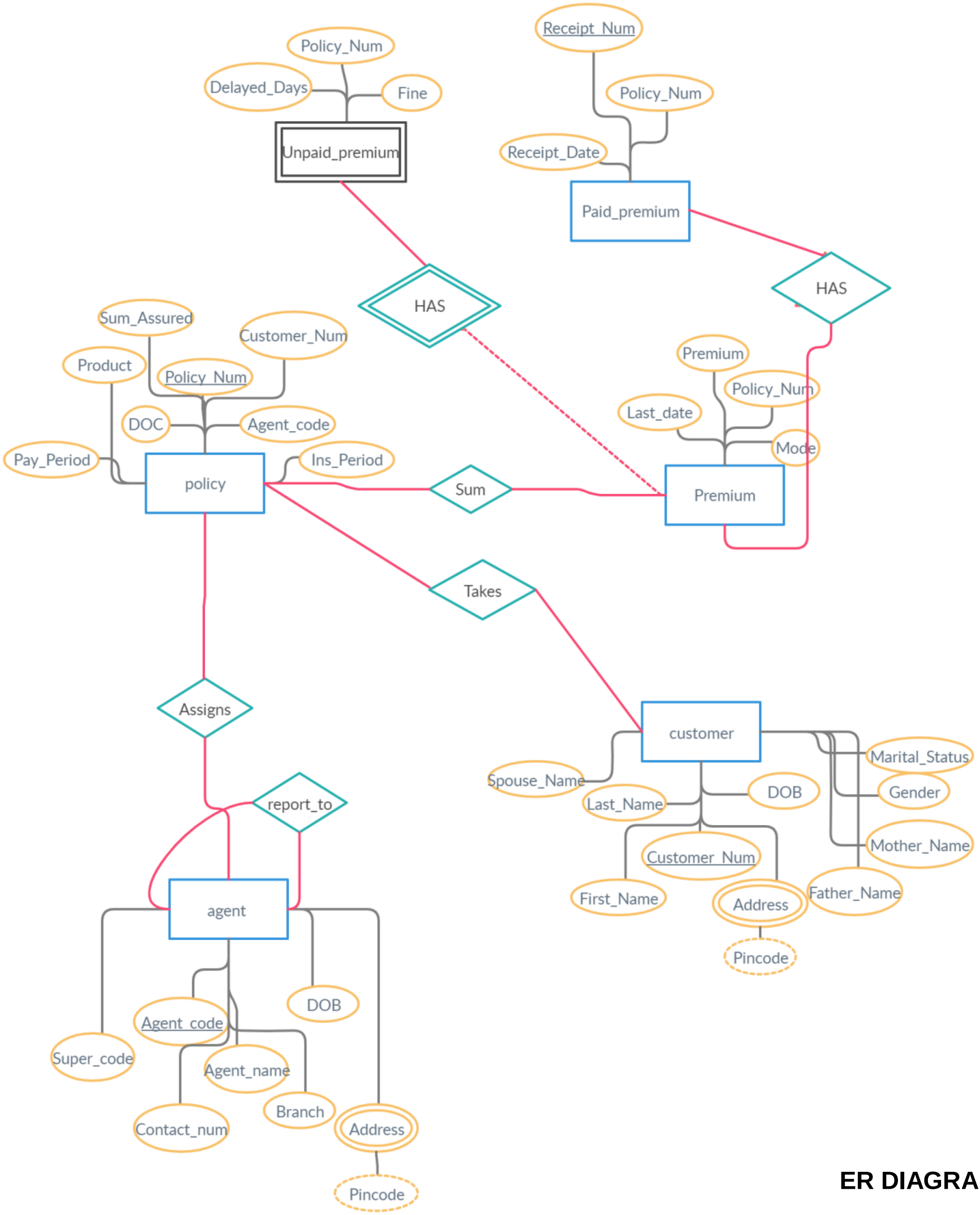
Having Insurance ensures:

- Financial Security
- Transfer of risk
- Complete protection of you and your family
- Peace of mind

This Insurance Management System is recreated using MySQL. There are totally six entity types for implementing this system.

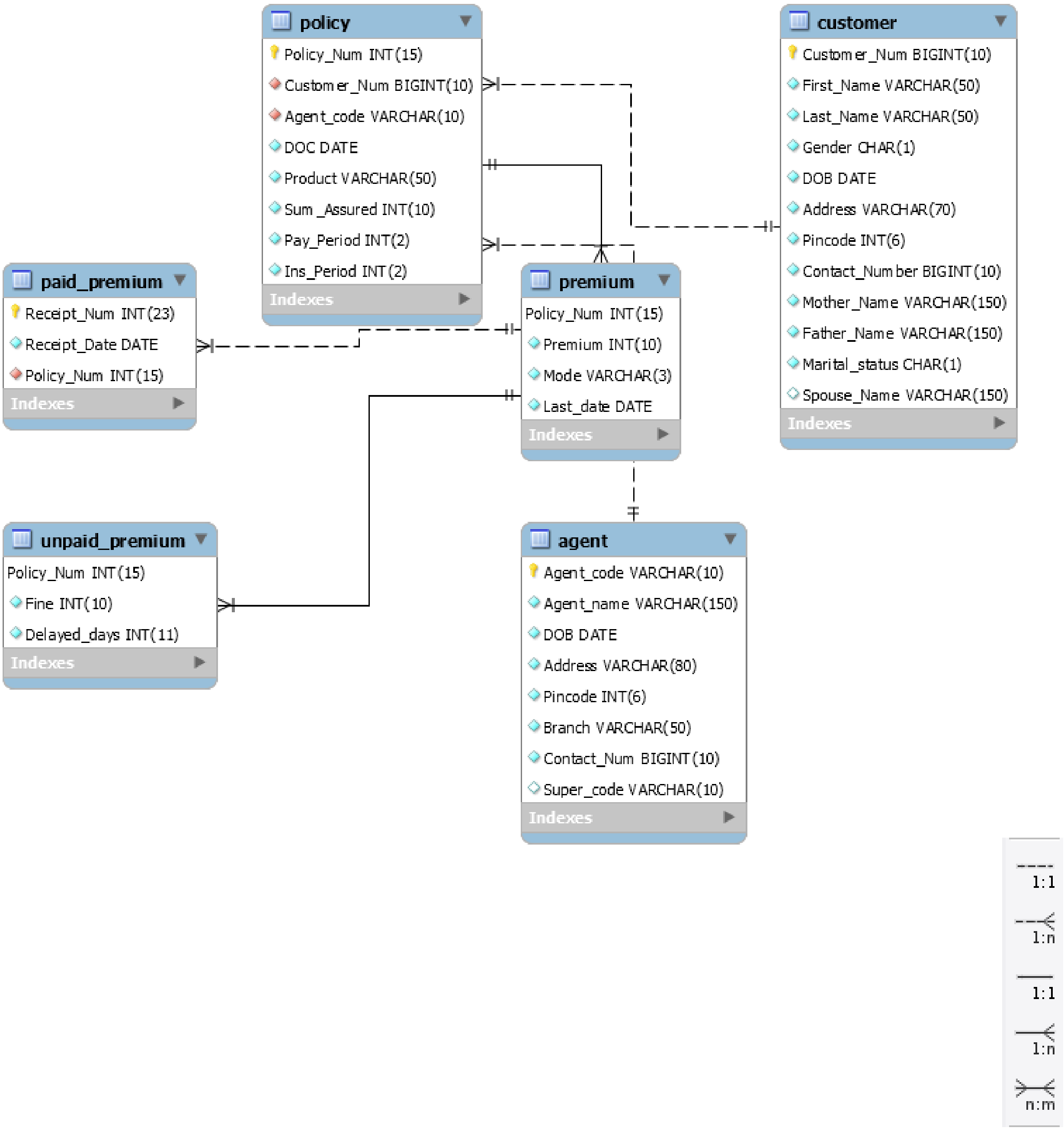
- Agent
- Customer
- Policy
- Premium
- Unpaid_premium
- Paid_premium

2. DATA MODEL



ER DIAGRAM

RELATIONAL SCHEMA



Mini World Description

Agent Entity

This table stores all the details of the agent i.e the insurer. The agent in an insurance contract undertakes to pay compensation**.**Agent and customer share 1:N Relationship. Each customer can be assigned to only one agent. An agent can also be a customer, hence Super_code stores the agent's agent code. A recursive relation.

Customer Entity

Stores details of the insured, the one purchases the insurance policies to safeguard the assets. Customer and policy entity share a weak 1:N relationship.

Policy Entity

Contains the policy the customer opts and the money paid and the period of insurance. Policy and agent share a weak N:1 Relationship

Premium Entity

An insurance premium is the amount of money an individual or business pays for an insurance policy. Premium and policy share N:1 relationship.

Paid_Premium

This entity stores the details of all paid receipts of the customers. Premium and paid_premium share a weak 1: N relationship.

Unpaid_Premium

This weak entity stores the details of all the customers numbers who haven't paid the sum and their fine. Premium and unpaid_premium share 1:N relationship.

Data Types

INT : Integer data type used to represent numbers and can stored numbers in the range $-2e^{22}$ to $+2e^{22}$

BIGINT: Used to represent range of integers greater than int.

VARCHAR : Stores alphanumeric values. Memory consumed will be the length of the string.

CHAR : Stores only characters. A CHAR(100) takes up 100 bytes on disk, regardless of the string it holds.

DATE : The DATE data type stores the calendar date in YYYY-MM-DD format.

3. FD AND NORMALIZATIONS

Functional Dependencies

Agent entity
Primary Key: Agent_code
Candidate key: Contact_no

- Agent_code ⟶ { Agent_Name, DOB , Address , Branch , Pincode, Contact_no, Super_code}
- Contact_no ⟶ { Agent_Name, DOB , Address , Branch , Pincode, Agent_code, Super_code}

Customer entity
Primary Key: Customer_no
Candidate Key: Contact_no,Address,Spouse_Name

- Customer_no ⟶ { First_Name, Last_Name ,DOB,Gender,Address, Contact_no, Mother_Name , Father_Name , Marital_Status , Spouse_Name}
- Contact_no ⟶ { First_Name, Last_Name ,DOB,Gender,Address, Customer_no, Mother_Name , Father_Name , Marital_Status , Spouse_Name}
- (Address, Spouse_Name) ⟶ { First_Name, Last_Name ,DOB,Gender,Contact_no,Customer_no, Mother_Name , Father_Name , Marital_Status}

Policy entity
Primary Key : Policy_no
Candidate Keys: Agent_code, Customer_no

- Policy_no ⟶ { Customer_no , agent_code , DOC , product , sum_assured , pay_period, ins_period }
- (Agent_code , Customer_no) ⟶ { Policy_no , DOC , product , sum_assured , pay_period,ins period}

Premium entity
Primary key: Policy_no

- Policy_no ⟶ { premium, mode, last_date }

Paid_premium
Primary Key: Receipt_no
Candidate Key: Policy_no

- Receipt_no ⟶ { Receipt_date , Policy_no}
- Policy_no ⟶ { Receipt_date,Receipt_no}

Unpaid_premium
Primary Key: Policy_num

- Policy_num ⟶ { Fine, delayed_days}

Normalizations

The normal form of a relation refers to the highest normal form condition that it meets. It indicates the degree to which it has been normalized. There are totally 5 Normal forms. First three normal forms are based on the functional dependencies among the attributes of a relation.

First Normal Form

It states that the domain of an attribute must include only atomic values. ER Mapping -to-Relational Mapping Algorithm ensures that all the multivalued attributes are converted into a new set of independent tables, thus the ‘Pincode’ attribute which is derived from the ‘Address’ attribute is given a separate column in the Relational Schema. Hence all the relations have no multivalued attributes and satisfy 1NF.

Second Normal Form

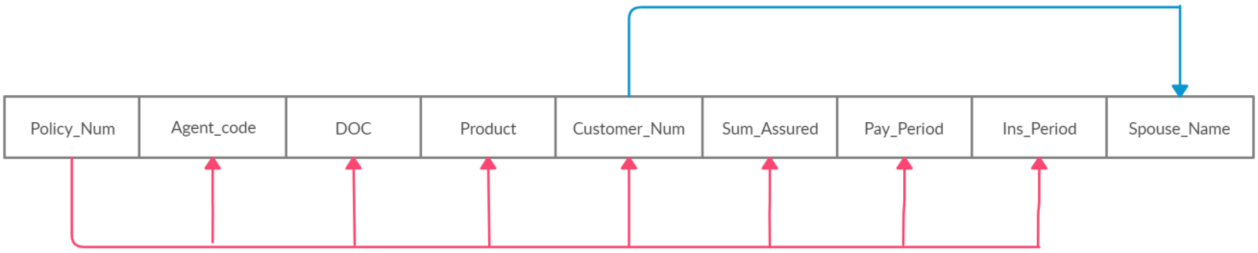
It states that every nonprime attribute A in R is fully functionally dependent on the primary key of R. Every non candidate attribute of the relation is dependent on the primary key and there exists no situation where a nonprime attribute is functionally dependent on a part of primary key. Hence all relations satisfy 2NF.

Third Normal Form

It should satisfy 2NF and also the condition that no nonprime attribute of R is transitively dependent on the primary key. The relation has no nonprime attribute functionally dependent on a nonprime attribute. There is no transitive dependency of a nonprime attribute on a primary key. Hence all relations satisfy 3NF.

Situations leading to violations of these Normal Forms

If a column named ‘Spouse_Name’ from the customer table is added to the policy table, then Policy_Num being the primary key and Spouse_Name being a nonprime key will show transitive dependency as Policy_Num⟶ {Customer_Num} and Customer_Num⟶ {Spouse_Name} , thus Policy_Num ⟶ {Spouse_Name}. Hence it violates the rules of 3NF.



Policy_Num	Agent_code	DOC	Product	Customer_Num	Sum_Assured	Pay_Period	Ins_Period	Spouse_Name

4. DATA DEFINITION LANGUAGE

```
use insurance;

CREATE TABLE agent (
  Agent_code varchar(10) NOT NULL,
  Agent_name varchar(150) NOT NULL,
  DOB date NOT NULL,
  Address varchar(80) NOT NULL,
  Pincode int(6) NOT NULL,
  Branch varchar(50) NOT NULL,
  Contact_Num bigint(10) NOT NULL
);

CREATE TABLE customer (
  Customer_Num bigint(10) NOT NULL,
  First_Name varchar(50) NOT NULL,
  Last_Name varchar(50) NOT NULL,
  Gender char(1) NOT NULL,
  DOB date NOT NULL,
  Address varchar(70) NOT NULL,
  Pincode int(6) NOT NULL,
  Contact_Number bigint(10) NOT NULL,
  Mother_Name varchar(150) NOT NULL,
  Father_Name varchar(150) NOT NULL,
  Marital_status char(1) NOT NULL,
  Spouse_Name varchar(150) DEFAULT NULL
);

CREATE TABLE policy (
  Policy_Num int(15) NOT NULL,
  Customer_Num bigint(10) NOT NULL,
  Agent_code varchar(10) NOT NULL,
  DOC date NOT NULL,
  Product varchar(50) NOT NULL,
  Sum_Assured int(10) NOT NULL,
  Pay_Period int(2) NOT NULL,
  Ins_Period int(2) NOT NULL
);

CREATE TABLE premium (
  Policy_Num int(15) NOT NULL,
  Premium int(10) NOT NULL,
  Mode varchar(3) NOT NULL,
  Last_date date NOT NULL
);
```

```
CREATE TABLE unpaid_premium (
  Policy_Num int(15) NOT NULL,
  Fine int(10) NOT NULL,
  Delayed_days int(11) NOT NULL
);

CREATE TABLE paid_premium (
  Receipt_Num int(23) NOT NULL,
  Receipt_Date date NOT NULL,
  Policy_Num int(15) NOT NULL,
);

#adding keys

ALTER TABLE agent
  ADD PRIMARY KEY (Agent_code);

ALTER TABLE customer
  ADD PRIMARY KEY (Customer_Num);

ALTER TABLE policy
  ADD PRIMARY KEY (Policy_Num),
  ADD KEY Agent_code (Agent_code),
  ADD KEY Customer_Num (Customer_Num);

ALTER TABLE premium
  ADD PRIMARY KEY (Policy_Num);


ALTER TABLE unpaid_premium
  ADD PRIMARY KEY (Policy_Num);

ALTER TABLE paid_premium
  ADD PRIMARY KEY (Receipt_Num),
  ADD KEY Policy_Num (Policy_Num);

#foreign key

ALTER TABLE paid_premium
  ADD CONSTRAINT Policy_Num FOREIGN KEY (Policy_Num) REFERENCES premium
(Policy_Num) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE policy
  ADD CONSTRAINT Agent_code FOREIGN KEY (Agent_code) REFERENCES agent
(Agent_code) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT Customer_Num FOREIGN KEY (Customer_Num) REFERENCES
customer (Customer_Num) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE premium
  ADD CONSTRAINT Policy FOREIGN KEY (Policy_Num) REFERENCES policy
(Policy_Num) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE unpaid_premium
  ADD CONSTRAINT Policy_Unpaid FOREIGN KEY (Policy_Num) REFERENCES premium
(Policy_Num) ON DELETE CASCADE ON UPDATE CASCADE;
```