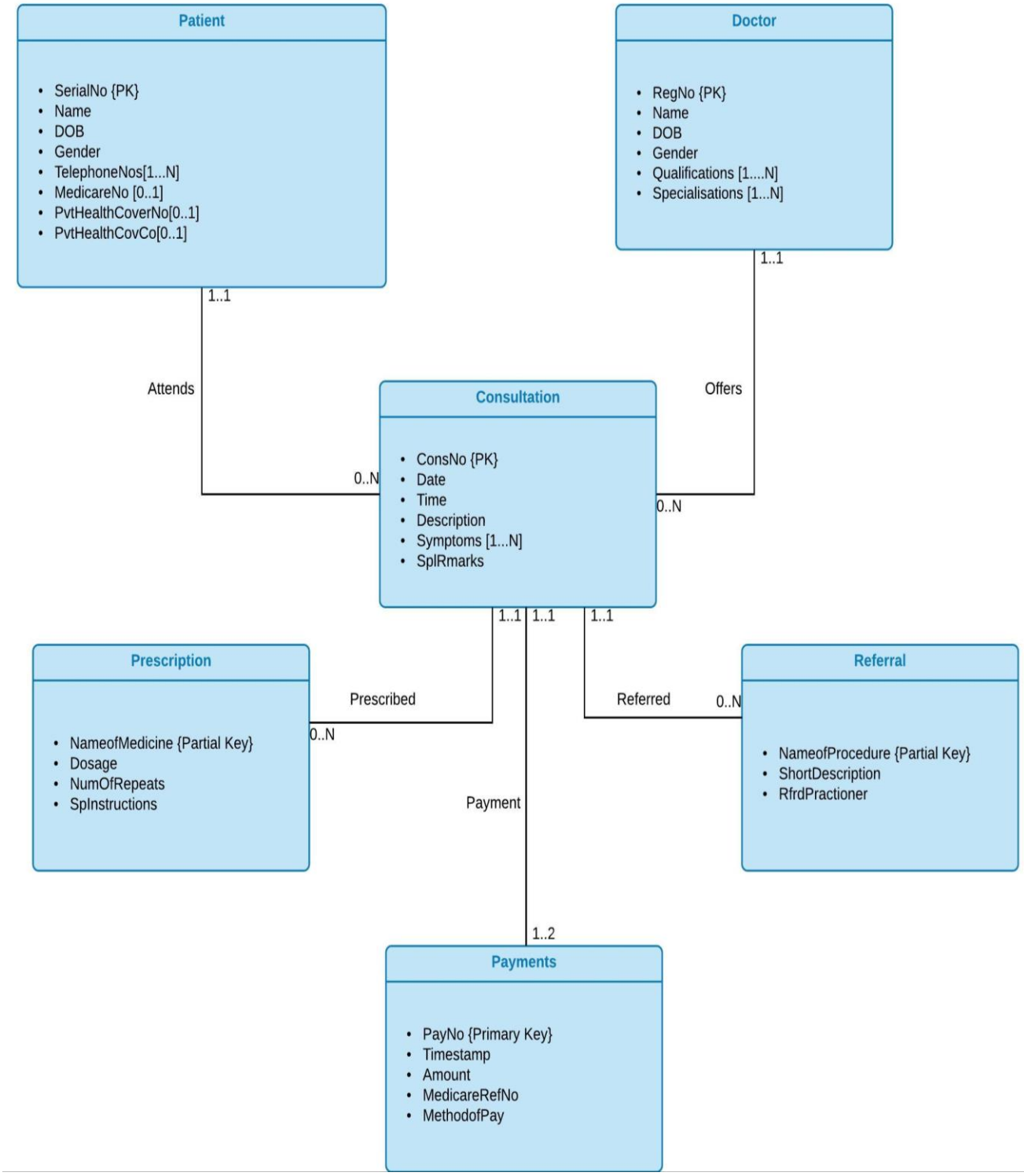
Part A :Entity Relationship Modelling



Note :In Entity Patients, attributes such as Medicare Number, Private Health Cover Number and Private Health Cover Company are optional, hence value (0..1)

MAPPING AN ER MODEL TO A RELATIONAL DATABASE SCHEMA:

STEP 1: STRONG ENTITIES:

AUTHOR (EMAIL, NAME, ADDRESS, TELEPHONE {1..N})

BOOK (ISBN, TITLE, EDITION, YEAR, LISTPRICE)

PUBLISHER (NAME, ADDRESS, URL, ABN)

WAREHOUSE (CODE, ADDRESS)

SHOPPINGCART (CARTID, TIMESTAMP)

CUSTOMER (EMAIL, NAME, ADDRESS)

STEP 2: WEAK ENTITIES.

There is no weak entity in the ER Model.

STEP 3: ONE TO ONE RELATIONSHIPS

One to One Relationship does not exists in the ER Model.

STEP 4: ONE TO MANY RELATIONSHIPS

SHOPPINGCART (CARTID, EMAIL*,TIMESTAMP)

STEP 5: MANY TO MANY RELATIONSHIP

WRITTENBY (EMAIL*,ISBN*)

PUBLISHEDBY (NAME*,ISBN*)

STOCKEDAT (ISBN*,CODE*,STOCKQTY)

ADDEDTO (ISBN*,CARDID*,BUYPRICE,QTY)

STEP 6: MULTIVALUED ATTRIBUTE

AUTHORTELEPHONE (EMAIL*, TELEPHONE)

COMPLETE RELATIONAL MODEL:

AUTHOR (EMAIL, NAME, ADDRESS, TELEPHONE {1..N})

BOOK (ISBN, TITLE, EDITION, YEAR, LISTPRICE)

PUBLISHER (NAME, ADDRESS, URL, ABN)

WAREHOUSE (CODE, ADDRESS)

SHOPPINGCART (CARTID, TIMESTAMP)

CUSTOMER (EMAIL, NAME, ADDRESS)

SHOPPINGCART (CARTID, EMAIL*,TIMESTAMP)

WRITTENBY (EMAIL*,ISBN*)

PUBLISHEDBY (NAME*,ISBN*)

STOCKEDAT (ISBN*,CODE*,STOCKQTY)

ADDEDTO (ISBN*,CARDID*,BUYPRICE,QTY)

AUTHORTELEPHONE (EMAIL*, TELEPHONE)

SQL Queries:

--Q1 Display the titles of all books on the subject "DataBases". Your result set should be sorted on the alphabetical order of the titles.

```
SELECT title Databases_Book
  FROM book b
       JOIN
       subject s ON b.subjectid = s.subjectid
 WHERE lower(subjecttype) = 'databases'
 ORDER BY title;
```

-- Q2a Display a. the number of books on the subject "DataBases".

```
SELECT count( * ) No_Books_On_Databases
  FROM book b
       JOIN
       subject s ON b.subjectid = s.subjectid
 WHERE lower(subjecttype) = 'databases';
```

-- Q2b the number of book copies on the subject "DataBases".

```
SELECT count(a.bookdescid) BookCopieS_on_DatabaseS
  FROM book a
```

```
    JOIN
    book_copy b ON a.bookdescid = b.bookdescid
    JOIN
    subject c ON a.subjectid = c.subjectid
 WHERE lower(subjecttype) = 'databases';
```

-- Q3a Done Display the firstname and lastname of the authors who wrote books on the subject"DataBases" .a. Write your query without using NATURAL JOINs.

```
SELECT c.firstname Given_Name,
    c.lastname Family_Name
  FROM book a
    JOIN
    written_by b ON a.bookdescid = b.bookdescid
    JOIN
    author c ON b.authorid = c.authorid
    JOIN
    subject d ON a.subjectid = d.subjectid
 WHERE lower(subjecttype) = 'databases';
```

-- Q3 b. Write your query using NATURAL JOINs.

```
SELECT firstname Given_Name,
```

```
    lastname Family_Name
  FROM book
    NATURAL JOIN
    written_by
    NATURAL JOIN
    author
    NATURAL JOIN
    subject
 WHERE lower(subjecttype) = 'databases';
```

-- Q4 Who translated the book "American Electrician's Handbook"? Display the firstname,middlenames, and lastname of the translator.

```
SELECT c.firstname,
    c.middlename,
    c.lastname
  FROM book a
    JOIN
    written_by b ON a.bookdescid = b.bookdescid
    JOIN
    author c ON b.authorid = c.authorid
 WHERE title LIKE '%AMERICAN ELECTRICIAN''S
HANDBOOK%' AND
```

```
    role LIKE 'TRANSLATOR';


-- Q5 Display the firstname and lastname of the people who
returned books late.
SELECT DISTINCT a3.firstname ,
                a3.lastname
  FROM person a3
    JOIN
    borrow a4 ON a3.personid = a4.personid
 WHERE returndate > duedate;


-- Q6 Display the firstname and lastname of the people who
returned books more than 7 days late.
SELECT DISTINCT p.firstname ,
                p.lastname
  FROM person p
    JOIN
    borrow b ON p.personid = b.personid
 WHERE returndate - duedate > 7;


-- Q7 Display the titles of books that haven't been borrowed.
```

```
SELECT title Books_Not_Borrowed
  FROM book
 WHERE bookdescid NOT IN (
          SELECT bc.bookdescid
            FROM book_copy bc
                 JOIN
                 borrow_copy br ON bc.bookid = br.bookid
       );


-- Q8a Using partial matching of the book title -- note that the borrower is interested in a "DATABASE" book.
SELECT title
  FROM book
 WHERE title NOT IN ('PRINCIPLES AND PRACTICE OF DATABASE SYSTEMS') AND
       title LIKE '%Database%';


-- Q8b By searching of other books written by the same author (i.e. the author of "PRINCIPLES AND PRACTICE OF DATABASE SYSTEMS"
SELECT b.title
  FROM written_by wb
```

```
    JOIN
    book b ON b.BOOKDESCID = wb.BOOKDESCID
 WHERE b.title <> 'PRINCIPLES AND PRACTICE OF
DATABASE SYSTEMS' AND
    wb.authorid IN (
       SELECT a.authorid
        FROM author a
           JOIN
           written_by wb ON a.authorID = wb.authorID
           JOIN
           book b ON wb.bookdescID = b.bookdescID
         WHERE b.title = 'PRINCIPLES AND PRACTICE OF
DATABASE SYSTEMS'
    );



-- Q9 Display the list of publishers who have published books on
the subject "DataBases". Your query should display publisher's
full name, along with "DataBases" book titles they published.
SELECT publisherfullname,
    c.title DATABASES_BOOK
  FROM publisher a
    JOIN
    published_by b ON a.publisherid = b.publisherid
```

```
    JOIN
    book c ON b.bookdescid = c.bookdescid
    JOIN
    subject d ON c.subjectid = d.subjectid
WHERE subjecttype LIKE 'DATABASES';
```

-- Q10 List the full names of publishers who have not published books on “Databases"

```
SELECT publisherfullname
  FROM publisher
WHERE publisherid NOT IN (
        SELECT p.publisherid
          FROM publisher p
              JOIN
              published_by pb ON p.publisherid = pb.publisherid
              JOIN
              book bk ON bk.bookdescid = pb.bookdescid
              JOIN
              subject s ON bk.subjectid = s.subjectid
         WHERE subjecttype = 'DataBases'
    );
```

Part C: Relational Database Model

Solution 1a:
They didn't work as it results in violation of Unique Constraint of primary key in a table and both of the above sql statements effect the same column as well as the same row. Running the first sql statement value of SSN will be set to '666884444' where the SSN was '123456789', which is not possible as the SSN is the primary key in the Employee and each column component must be unique in the primary key column.

Solution 1b:
Since SSN is a primary key, hence changing it will result in Integrity Constraint. Instead of swapping SSN, we can swap all other attributes except SSN.

Solution 1c:
SQL QUERY 1:

```
UPDATE EMPLOYEE
SET FNAME = 'JOHN',
        MINIT = 'B',
        LNAME = 'SMITH',
        BDATE = '1965-01-09',
        ADDRESS = '731 FORDREN, HOUSTEN, TX',
        SEX = 'M',
        SALARY = 30000
WHERE SSN = 666884444;
```

SQL QUERY 2:

```
UPDATE EMPLOYEE
SET FNAME = 'RAMESH',
        MINIT = 'K',
        LNAME = 'NARAYAN',
        BDATE = '1962-09-15',
        ADDRESS = '975 FIRE OAK, HUMBLE, TX',
        SEX = 'M',
        SALARY = 38000
WHERE SSN = 123456789;
```

Solution 2:
There is no constraints violation.

Solution 3:
There is no constraints are violation.

Solution 4:
Entity Integrity Constraint is violated as primary key cannot be null. Pnumber is a primary key and it is taken as null. Hence, the violation.