NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

Practical Report

Database Management Systems

Computer Science Engineering (Internet of Things)

Semester 3

Kushagra Lakhwani

2021UCI8036

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

February 20, 2024

Abstract

The lab project report *Database Management System* is submitted by *Kushagra Lakhwani* [1] (Roll No. 2021UCI8036). The report is submitted to the *Mr. Vishal Gupta*, Department of Computer Science and Engineering, Netaji Subhas University of Technology in the fulfillment of the requirements for the course of *Database Management System* (semester 3).

Contents

1 Sailors

1.1 Schema

Consider the following relational schema:

```
SAILORS (sid, sname, rating, date_of_birth)

BOATS (bid, bname, color)

RESERVES (sid, bid, date, time_slot)
```

1.2 Table Definitions

a) Create the tables for the schema

```sql
CREATE TABLE SAILORS (
    sid INTEGER PRIMARY KEY,
    sname VARCHAR(20),
    rating INTEGER,
    date_of_birth DATE
);

CREATE TABLE BOATS (
    bid INTEGER PRIMARY KEY,
    bname VARCHAR(20),
    color VARCHAR(20)
);

CREATE TABLE RESERVES (
    sid INTEGER,
    bid INTEGER,
    date DATE,
    time_slot INTEGER,
    PRIMARY KEY (sid, bid, date, time_slot)
);
```

```
practicals=# \dt
      List of relations
Schema |    Name    | Type  |  Owner
--------+----------+-------+----------
public | boats     | table | postgres
public | reserves  | table | postgres
public | sailors   | table | postgres
(3 rows)
```

b) Add Foreign Key constraints

```
ALTER TABLE reserves
ADD FOREIGN KEY (sid) REFERENCES sailors(sid);

ALTER TABLE reserves
ADD FOREIGN KEY (bid) REFERENCES boats(bid);
```

c) Insert the following tuples into the tables

```
INSERT INTO Sailors VALUES (1, 'John', 7, '1999-01-03');
INSERT INTO Sailors VALUES (2, 'Rusty', 9, '1998-07-12');
INSERT INTO Sailors VALUES (3, 'Horatio', 9, '1996-05-22');
INSERT INTO Sailors VALUES (4, 'Zorba', 8, '1993-01-23');
INSERT INTO Sailors VALUES (5, 'Julius',8,'2001-09-01');

INSERT INTO Boats VALUES (101, 'Interlake', 'blue');
INSERT INTO Boats VALUES (102, 'Interlake', 'red');
INSERT INTO Boats VALUES (103, 'Clipper', 'green');
INSERT INTO Boats VALUES (104, 'Marine', 'red');

INSERT INTO Reserves VALUES (1, 101, '2017-10-10',1);
INSERT INTO Reserves VALUES (1, 102, '2017-10-10',2);
INSERT INTO Reserves VALUES (1, 103, '2017-10-10',2);
INSERT INTO Reserves VALUES (1, 104, '2017-10-10',2);
INSERT INTO Reserves VALUES (1, 101, '2019-10-10',1);
INSERT INTO Reserves VALUES (2, 102, '2011-03-01',3);
INSERT INTO Reserves VALUES (2, 102, '2019-11-07',3);
INSERT INTO Reserves VALUES (3, 101, '2017-11-07',2);
INSERT INTO Reserves VALUES (3, 102, '2017-08-07',2);
INSERT INTO Reserves VALUES (4, 103, '2017-03-19',1);
INSERT INTO Reserves VALUES (2, 103, '2017-03-19',3);
```

1.3 Queries

1. Find sailors who've reserved at least one boat
   (a) Relational Algebra

$$\pi_{sid,sname}(SAILORS \bowtie RESERVES))$$

   (b) SQL

```
SELECT sname
FROM SAILORS
WHERE sid IN (
    SELECT sid
    FROM RESERVES
);
```

2. Output

```
practicals=# \i /home/sql/01.sql
  sname
---------
 John
 Rusty
 Horatio
 Zorba
(4 rows)
```

3. Find names of sailors who've reserved a red or a green boat in the month of March.
   (a) Relational Algebra

$$\pi_{sname}(SAILORS \bowtie RESERVES \bowtie BOATS) \bowtie \sigma_{\text{bname=red}\vee\text{bname=green}}(\sigma_{\text{date=March}}(BOATS \bowtie RESERVES))$$

   (b) SQL

```
SELECT sname
FROM SAILORS
WHERE sid IN
    (SELECT sid
    FROM RESERVES
    WHERE bid IN
        (SELECT bid
        FROM BOATS
        WHERE bname = 'red' OR bname = 'green')
    AND (SELECT extract(month FROM date) FROM RESERVES) = 3)
```

   (c) Output

```
practicals=# \i /home/sql/sea2.sql
sname
-------
Rusty
Zorba
(2 rows)
```

4. Find names of sailors who've reserved a red and a green boat
   (a) Relational Algebra

$$\pi_{sname}(SAILORS \bowtie RESERVES \bowtie (\sigma_{\text{color}=red}(BOATS))) \cap \pi_{sname}(SAILORS \bowtie RESERVES \bowtie (\sigma_{\text{color}=green}(BOATS)))$$

(b) SQL

```
SELECT DISTINCT S1.sname
FROM SAILORS S1, RESERVES R1, BOATS B1,
RESERVES R2, BOATS B2
WHERE S1.sid = R1.sid
    AND R1.bid = B1.bid
    AND S1.sid = R2.sid
    AND R2.bid = B2.bid
    AND B1.color = 'red'
    AND B2.color = 'green';
```

(c) Output

```
sname
-------
John
Rusty
(2 rows)
```

5. Find SID of sailors who have <u>not</u> reserved a boat after Jan 2018.

(a) Relational Algebra

$$\pi_{\text{sid}} - \pi_{\text{sid}}(SAILORS \bowtie \sigma_{\text{date\_of\_birth} > \text{Jan 2018}}(RESERVES))$$

(b) SQL

```
SELECT sid FROM SAILORS
WHERE sid NOT IN
    (SELECT sid FROM RESERVES
    WHERE date_of_birth > '2018-01-01')
```

(c) Output

```
practicals=# \i /home/sql/sea4.sql
sid
-----
3
4
5
(3 rows)
```

6. Find sailors whose rating is greater than that of all the sailors named "John"

(a) Relational Algebra

$$\pi_{\text{sid,sname}}(SAILORS)$$
$$- \pi_{S_2.\text{sid},S_2.\text{sname}}(\sigma_{S_2.\text{rating}<S.\text{rating}}(\rho_{S_2}(SAILORS) \times \rho_S(SAILORS)))$$

(b) SQL

```
SELECT sid, sname FROM SAILORS S1
WHERE S1.rating > ALL
        (SELECT S2.rating FROM SAILORS S2
        WHERE S2.sname = 'John')
```

(c) Output

```
practicals=# \i /home/sql/sea5.sql
sid |  sname
-----+---------
   2 | Rusty
   3 | Horatio
   4 | Zorba
   5 | Julius
(4 rows)
```

7. Find sailors who've reserved all boats
   (a) Relational Algebra

$$\pi_{\text{sid,sname}}(\pi_{\text{sid,bid}}(RESERVES) \div \pi_{\text{bid}}(BOATS)) \bowtie SAILORS$$

   (b) SQL

```
SELECT S.sid, S.sname
FROM SAILORS S
WHERE NOT EXISTS
      (SELECT B.bid
      FROM BOATS B
      WHERE NOT EXISTS
            (SELECT R.sid, R.bid
            FROM RESERVES R
            WHERE R.sid = S.sid
                  AND R.bid = B.bid))
```

   (c) Output

```
practicals=# \i /home/sql/sea6.sql
sid | sname
-----+-------
   1 | John
(1 row)
```

8. Find name and age of the oldest sailor(s)
   (a) Relational Algebra

$$\begin{aligned}&\pi_{\text{sname,age}}(\pi_{sid}(SAILORS) - \\ &\qquad \pi_{S_2.sid}(\sigma_{S_2.\text{age}<S.\text{age}}(\rho_{S_2}(SAILORS) \times \rho_S(SAILORS)))) \\ &\qquad\qquad\qquad\qquad \bowtie SAILORS\end{aligned}$$

(b) SQL

```
SELECT sname FROM SAILORS S1
WHERE S1.date_of_birth > ALL
        (SELECT S2.date_of_birth FROM SAILORS S2)
```

9. Find the age of the youngest sailor for each rating with at least 2 such sailors
   (a) Relational Algebra

$$\pi_{\text{rating,minage}}(\sigma_{\text{no\_of\_sailors}>1}(\rho_{r(\text{rating,no\_of\_sailors,minage})}\ \mathcal{F}_{(\text{rating},count(\text{sid}),min(\text{age}))}(SAILORS)))$$

   (b) SQL

```
SELECT rating, age FROM SAILORS S1
WHERE S1.date_of_birth > ALL AS minage
            (SELECT S2.date_of_birth FROM SAILORS S2
            WHERE S2.rating = S1.rating)
GROUP BY rating
HAVING COUNT(*) >= 2
```

2 Customers

2.1 Schema

Consider the following relational schema:

```
CUSTOMERS (cust_num, cust_lname, cust_fname, cust_balance)

PRODUCT (prod_num, prod_name, price)

INVOICE (inv_num, prod_num, cust_num, inv_date, unit_sold, inv_amount)
```

2.2 Table Definitions

a) Create the tables for the schema

```sql
CREATE TABLE CUSTOMERS
(
    cust_num int PRIMARY KEY,
    cust_lname varchar(20),
    cust_fname varchar(20),
    cust_balance int
);

CREATE TABLE PRODUCT
(
    prod_num int PRIMARY KEY,
    prod_name varchar(20),
    price int
);

CREATE TABLE INVOICE
(
    inv_num int,
    prod_num int,
    cust_num int,
    inv_date date,
    unit_sold int,
    inv_amount int,
    PRIMARY KEY (inv_num, prod_num, cust_num, inv_date),
    FOREIGN KEY (prod_num) REFERENCES PRODUCT(prod_num),
    FOREIGN KEY (cust_num) REFERENCES CUSTOMERS(cust_num)
);
```

```
practicals=# \dt
     List of relations
Schema |    Name    | Type  |  Owner
--------+-----------+-------+----------
public | customers | table | postgres
public | invoice   | table | postgres
public | product   | table | postgres
(3 rows)
```

b) Insert the following tuples into the tables

```
INSERT INTO CUSTOMERS VALUES
(1, 'Smith', 'John', 0),
(2, 'Jones', 'Mary', 2000),
(3, 'Brown', 'Peter', 3000),
(4, 'Smith', 'Mary', 0),
(5, 'Brown', 'John', 5000),
(6, 'Smith', 'Peter', 6000),
(7, 'Jones', 'John', 7000),
(8, 'Brown', 'Mary', 8000),
(9, 'Smith', 'John', 9000),
(10, 'Jones', 'Mary', 10000);

INSERT INTO PRODUCT VALUES
(1, 'Laptop', 1000),
(2, 'Desktop', 2000),
(3, 'Tablet', 3000),
(4, 'Mobile', 4000),
(5, 'Printer', 5000),
(6, 'Scanner', 6000),
(7, 'Monitor', 7000),
(8, 'Keyboard', 8000),
(9, 'Mouse', 9000),
(10, 'Speakers', 10000);

INSERT INTO INVOICE VALUES
(1, 1, 1, '2015-01-01', 1, 1000),
(2, 2, 1, '2015-02-01', 2, 4000),
(3, 3, 1, '2015-03-01', 3, 9000),
(4, 4, 1, '2015-04-01', 4, 16000),
(5, 5, 1, '2015-05-01', 5, 25000),
(6, 6, 1, '2015-06-01', 6, 36000),
(7, 7, 1, '2015-07-01', 7, 49000),
(8, 8, 1, '2015-06-01', 8, 64000),
(9, 9, 1, '2015-04-01', 9, 81000),
(10, 10, 1, '2015-10-01', 10, 100000),
(11, 1, 2, '2015-11-01', 1, 2000),
(13, 3, 2, '2015-01-01', 3, 6000),
(14, 4, 2, '2015-01-01', 4, 8000);
```

2.3 Queries

Write SQL queries and relational algebraic expression for the following:

1. Find the names of the customer who have purchased no item. Set default value of cust_balance as 0 for such customers.
   (a) Relational Algebra

$$\pi_{\text{cust\_lname}+\text{""}+\text{cust\_fname}}(\sigma_{\text{cust\_balance}=0}(CUSTOMERS))$$

(b) SQL

```
SELECT concat(cust_lname , ' ' , cust_fname) as name
FROM CUSTOMERS
WHERE cust_balance = 0;
```

(c) Output

```
practicals=# \i /home/sql/customers1.sql
    name
------------
 Smith John
 Smith Mary
(2 rows)
```

2. Write the trigger to update the CUST_BALANCE in the CUSTOMER table when a new invoice record is entered for the customer.

```
CREATE TRIGGER update_cust_balance
AFTER INSERT ON INVOICE
FOR EACH ROW
BEGIN
    UPDATE CUSTOMERS
    SET cust_balance = cust_balance + NEW.inv_amount
    WHERE cust_num = NEW.cust_num;
END;
```

3. Find the customers who have purchased more than three units of a product on a day.
   (a) Relational Algebra

$$\pi_{\text{cust\_lname+""+cust\_fname}}(\sigma_{\text{unit\_sold}\geq 3}$$
$$(CUSTOMER \bowtie \sigma_{\text{unit\_sold}}$$
$$(_{\text{cust\_num, inv\_date>prod\_num}} \mathcal{F}_{sum(\text{unit\_sold})}(INVOICE))))$$

   (b) SQL

```
SELECT concat(cust_lname , ' ' , cust_fname) as name
FROM CUSTOMERS
WHERE cust_num IN
(
    SELECT cust_num
    FROM INVOICE
    GROUP BY cust_num, inv_date, prod_num
    HAVING sum(unit_sold) >= 3
);
```

(c) Output

```
practicals=# \i /home/sql/customers3.sql
    name
------------
 Smith John
 Jones Mary
(2 rows)
```

4. Write a query to illustrate Left Outer, Right Outer and Full Outer Join.

   (a) Left Outer Join

$$CUSTOMER] \bowtie INVOICE$$

```
SELECT CONCAT(C.cust_fname, c.cust_lname) as name,
LEFT JOIN INVOICE i
ON C.cust_num=i.cust_num
```

   (b) Right Outer Join

$$CUSTOMER \bowtie [INVOICE$$

```
SELECT CONCAT(C.cust_fname, c.cust_lname) as name,
RIGHT JOIN INVOICE i
ON C.cust_num=i.cust_num
```

   (c) Full Outer Join

$$CUSTOMER] \bowtie [INVOICE$$

```
SELECT CONCAT(C.cust_fname, " ", C.cust_lname) as name
LEFT JOIN INVOICE i
ON C.cust_num=i.cust_num
UNION
SELECT CONCAT(C.cust_fname, " ", C.cust_lname) as name, i.inv_amount
RIGHT JOIN INVOICE i
ON C.cust_num=i.cust_num
```

5. Count number of products sold on each date.
   (a) Relational Algebra

$$\pi_{\text{inv\_date}, sum(\text{unit\_sold})}(\, _{\text{inv\_date}}\mathcal{F}_{sum(\text{unit\_sold})}(INVOICE))$$

   (b) SQL

```sql
SELECT inv_date, sum(unit_sold)
FROM INVOICE
GROUP BY inv_date
```

   (c) Output

```
practicals=# SELECT inv_date, sum(unit_sold)
    FROM INVOICE
    GROUP BY inv_date;
  inv_date   | sum
------------+-----
 2015-10-01 |  10
 2015-07-01 |   7
 2015-03-01 |   3
 2015-02-01 |   2
 2015-01-01 |   8
 2015-06-01 |  14
 2015-05-01 |   5
 2015-11-01 |   1
 2015-04-01 |  13
(9 rows)
```

6. As soon as customer balance becomes greater than Rs. 100,000, copy the customer_num in new table called "GOLD_CUSTOMER"
   (a) Create table GOLD_CUSTOMER

```sql
CREATE TABLE GOLD_CUSTOMER
(
    cust_num int,
    PRIMARY KEY (cust_num),
    FOREIGN KEY (cust_num) REFERENCES CUSTOMERS (cust_num)
)
```

(b) Create a trigger to update the GOLD_CUSTOMER table when a new invoice record is entered for the customer.

```
CREATE TRIGGER update_gold_customer
AFTER INSERT ON INVOICE
FOR EACH ROW
BEGIN
    IF NEW.cust_balance > 100000
    AND NEW.cust_num NOT IN (SELECT cust_num FROM GOLD_CUSTOMER) THEN
        INSERT INTO GOLD_CUSTOMER VALUES (NEW.cust_num);
    END IF;
END
```

7. Add a new attribute CUST_DOB in customer table

```
ALTER TABLE CUSTOMERS
ADD COLUMN cust_dob date
```

3 Case Study: Library Management System

3.1 Introduction

Online Library Management System is a system which maintains the information about the books present in the library, their authors, the members of library to whom books are issued, library staff and all.

This is very difficult to organize manually. Maintenance of all this information manually is a very complex task. Owing to the advancement of technology, organization of an Online Library becomes much simple.

The Online Library Management has been designed to computerize and automate the operations performed over the information about the members, book issues and returns and all other operations. This computerization of library helps in many instances of its maintenances. It reduces the workload of management as most of the manual work done is reduced.

3.2 Objective

The main objective of this project is to develop a system that will help in maintaining the information about the books present in the library, their authors, the members of library to whom books are issued, library staff and all.

The library management system keeps track of reader with the following considerations:

1. The system keeps track of the staff with a single point authentication system comprising login ID and password.
2. Staff maintains the book catalog with its ISBN, Book title, price(in INR), category(novel, general, story), edition, author Number and details.
3. A publisher has publisher ID, Year when the book was published, and name of the book.
4. Readers are registered with their user ID, email, name (first name, last name), Phone no (multiple entries allowed), communication address. The staff keeps track of readers.

5. Readers can return/reserve books that stamps with issue date and return date. If not returned within the prescribed time period, it may have a due date too.
6. Staff also generate reports that have readers ID, registration number of report, book number and return/issue info.

3.3 Design

The following are the entities of the system:

3.3.1 Entities

- **Book**
  It has authno, isbn number, title, edition, category, price. ISBN is the Primary Key for Book Entity.
- **Reader**
  It has UserId, Email, address, phone no, name. Name is composite attribute of firstname and lastname. Phone No. is multivalued attribute. UserId is the Primary Key for Readers entity.
- **Publisher**
  It has PublisherID, Year of publication, name. PublisherID is the Primary Key.
- **Authentication System**
  It has LoginID and password with LoginID as Primary Key.
- **Reports**
  It has UserId, Reg_no, Book_no, Issue/Return date. Reg_no is the Primary Key of reports entity.
- **Staff**
  It has name and staff_id with staff_id as Primary Key.
- **Reserve/Return Relationship Set**
  It has three attributes: Reserve date, Due date, Return date.

The relationships between the entities are as follows:

3.3.2 Relationships

- A reader can reserve N books, but one book can be reserved by only one reader. The relationship 1:N.
- A publisher can publish many books, but a book is published by only one publisher. The relationship 1:N.
- Staff keeps track of readers. The relationship is M:N.
- Staff maintains multiple reports. The relationship 1:N.
- Staff maintains multiple Books. The relationship 1:N.
- Authentication system provides login to multiple staffs. The relation is 1:N.

3.3.3 ER Diagram

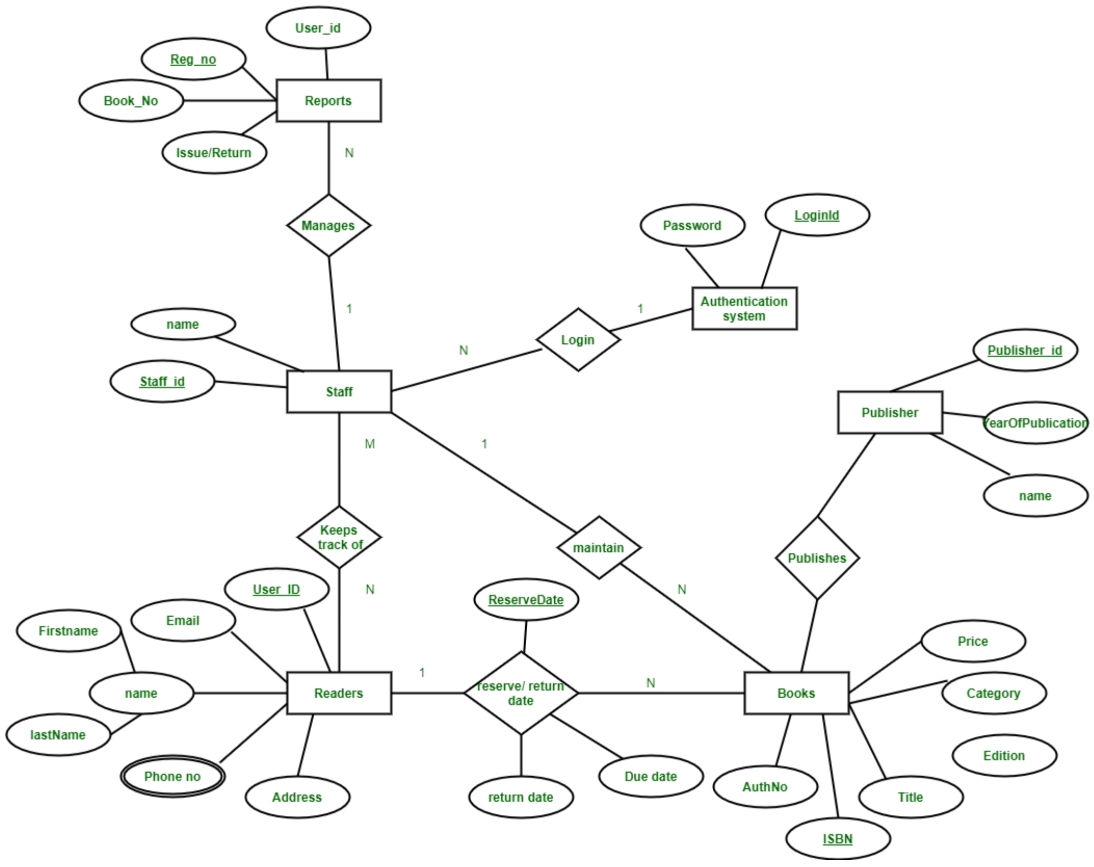

Figure 1: ER Diagram

References

[1] KorigamiK, "Dbms lab practical." https://github.com/korigamik/semester-3, 2022.