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

December 20, 2022

Contents

1 Sailors **2**
1.1 Schema 2
1.2 Queries 2

2 Customers **5**
2.1 Schema 5
2.2 Queries 5

Abstract

This is my [1] *Database Management System* lab project report. 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).

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 Queries

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

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

   (b) SQL

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

2. 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

```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)
```

3. 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

```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";
```

4. 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

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

5. 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

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

6. 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

```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))
```

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

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

   (b) SQL

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

8. 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

```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 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

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

   (b) SQL

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

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

```sql
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

```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
)
```

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

$$CUSTOMER] \bowtie INVOICE$$

```sql
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

$$CUSTOMER \bowtie [INVOICE$$

```sql
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

$$CUSTOMER] \bowtie [INVOICE$$

```sql
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
```

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.

```sql
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

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

References

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