



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

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).

Contents

1	Sailors	1
1.1	Schema	1
1.2	Table Definitions	1
1.3	Queries	2
2	Customers	7
2.1	Schema	7
2.2	Table Definitions	7
2.3	Queries	8

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

```
1 CREATE TABLE SAILORS (
2     sid INTEGER PRIMARY KEY,
3     sname VARCHAR(20),
4     rating INTEGER,
5     date_of_birth DATE
6 );
7
8 CREATE TABLE BOATS (
9     bid INTEGER PRIMARY KEY,
10    bname VARCHAR(20),
11    color VARCHAR(20)
12 );
13
14 CREATE TABLE RESERVES (
15     sid INTEGER,
16     bid INTEGER,
17     date DATE,
18     time_slot INTEGER,
19     PRIMARY KEY (sid, bid, date, time_slot)
20 );
```

```
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

```
1 ALTER TABLE reserves
2 ADD FOREIGN KEY (sid) REFERENCES sailors(sid);
3
4 ALTER TABLE reserves
5 ADD FOREIGN KEY (bid) REFERENCES boats(bid);
```

c) Insert the following tuples into the tables

```
1 INSERT INTO Sailors VALUES (1, 'John', 7, '1999-01-03');
2 INSERT INTO Sailors VALUES (2, 'Rusty', 9, '1998-07-12');
3 INSERT INTO Sailors VALUES (3, 'Horatio', 9, '1996-05-22');
4 INSERT INTO Sailors VALUES (4, 'Zorba', 8, '1993-01-23');
5 INSERT INTO Sailors VALUES (5, 'Julius', 8, '2001-09-01');
6
7 INSERT INTO Boats VALUES (101, 'Interlake', 'blue');
8 INSERT INTO Boats VALUES (102, 'Interlake', 'red');
9 INSERT INTO Boats VALUES (103, 'Clipper', 'green');
10 INSERT INTO Boats VALUES (104, 'Marine', 'red');
11
12 INSERT INTO Reserves VALUES (1, 101, '2017-10-10', 1);
13 INSERT INTO Reserves VALUES (1, 102, '2017-10-10', 2);
14 INSERT INTO Reserves VALUES (1, 103, '2017-10-10', 2);
15 INSERT INTO Reserves VALUES (1, 104, '2017-10-10', 2);
16 INSERT INTO Reserves VALUES (1, 101, '2019-10-10', 1);
17 INSERT INTO Reserves VALUES (2, 102, '2011-03-01', 3);
18 INSERT INTO Reserves VALUES (2, 102, '2019-11-07', 3);
19 INSERT INTO Reserves VALUES (3, 101, '2017-11-07', 2);
20 INSERT INTO Reserves VALUES (3, 102, '2017-08-07', 2);
21 INSERT INTO Reserves VALUES (4, 103, '2017-03-19', 1);
22 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

```
1 SELECT sname
2 FROM SAILORS
3 WHERE sid IN (
4     SELECT sid
5     FROM RESERVES
6 );
```

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

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

(b) SQL

```
1 SELECT sname
2 FROM SAILORS
3 WHERE sid IN
4     (SELECT sid
5      FROM RESERVES
6      WHERE bid IN
7          (SELECT bid
8           FROM BOATS
9           WHERE bname = 'red' OR bname = 'green'))
10 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

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

(b) SQL

```
1 SELECT DISTINCT S1.sname
2 FROM SAILORS S1, RESERVES R1, BOATS B1,
3 RESERVES R2, BOATS B2
4 WHERE S1.sid = R1.sid
5       AND R1.bid = B1.bid
6       AND S1.sid = R2.sid
7       AND R2.bid = B2.bid
8       AND B1.color = 'red'
9       AND B2.color = 'green';
```

(c) Output

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

5. Find SID of sailors who have not 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

```
1 SELECT sid FROM SAILORS
2 WHERE sid NOT IN
3       (SELECT sid FROM RESERVES
4        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

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

(b) SQL

```

1 SELECT sid, sname FROM SAILORS S1
2 WHERE S1.rating > ALL
3     (SELECT S2.rating FROM SAILORS S2
4      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

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

(b) SQL

```

1 SELECT S.sid, S.sname
2 FROM SAILORS S
3 WHERE NOT EXISTS
4     (SELECT B.bid
5      FROM BOATS B
6      WHERE NOT EXISTS
7          (SELECT R.sid, R.bid
8           FROM RESERVES R
9           WHERE R.sid = S.sid
10              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

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

(b) SQL

```
1 SELECT sname FROM SAILORS S1
2 WHERE S1.date_of_birth > ALL
3     (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

$$\pi_{\text{rating}, \text{minage}}(\sigma_{\text{no_of_sailors} \geq 1}(\rho_r(\text{rating}, \text{no_of_sailors}, \text{minage}) \mathcal{F}(\text{rating}, \text{count}(\text{sid}), \text{min}(\text{age}))(\text{SAILORS}))))$$

(b) SQL

```
1 SELECT rating, age FROM SAILORS S1
2 WHERE S1.date_of_birth > ALL AS minage
3     (SELECT S2.date_of_birth FROM SAILORS S2
4      WHERE S2.rating = S1.rating)
5 GROUP BY rating
6 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

```
1 CREATE TABLE CUSTOMERS
2 (
3     cust_num int PRIMARY KEY,
4     cust_lname varchar(20),
5     cust_fname varchar(20),
6     cust_balance int
7 );
8
9 CREATE TABLE PRODUCT
10 (
11     prod_num int PRIMARY KEY,
12     prod_name varchar(20),
13     price int
14 );
15
16 CREATE TABLE INVOICE
17 (
18     inv_num int,
19     prod_num int,
20     cust_num int,
21     inv_date date,
22     unit_sold int,
23     inv_amount int,
24     PRIMARY KEY (inv_num, prod_num, cust_num, inv_date),
25     FOREIGN KEY (prod_num) REFERENCES PRODUCT(prod_num),
26     FOREIGN KEY (cust_num) REFERENCES CUSTOMERS(cust_num)
27 );
```

```
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

```
1  INSERT INTO CUSTOMERS VALUES
2  (1, 'Smith', 'John', 0),
3  (2, 'Jones', 'Mary', 2000),
4  (3, 'Brown', 'Peter', 3000),
5  (4, 'Smith', 'Mary', 0),
6  (5, 'Brown', 'John', 5000),
7  (6, 'Smith', 'Peter', 6000),
8  (7, 'Jones', 'John', 7000),
9  (8, 'Brown', 'Mary', 8000),
10 (9, 'Smith', 'John', 9000),
11 (10, 'Jones', 'Mary', 10000);
12
13 INSERT INTO PRODUCT VALUES
14 (1, 'Laptop', 1000),
15 (2, 'Desktop', 2000),
16 (3, 'Tablet', 3000),
17 (4, 'Mobile', 4000),
18 (5, 'Printer', 5000),
19 (6, 'Scanner', 6000),
20 (7, 'Monitor', 7000),
21 (8, 'Keyboard', 8000),
22 (9, 'Mouse', 9000),
23 (10, 'Speakers', 10000);
24
25 INSERT INTO INVOICE VALUES
26 (1, 1, 1, '2015-01-01', 1, 1000),
27 (2, 2, 1, '2015-02-01', 2, 4000),
28 (3, 3, 1, '2015-03-01', 3, 9000),
29 (4, 4, 1, '2015-04-01', 4, 16000),
30 (5, 5, 1, '2015-05-01', 5, 25000),
31 (6, 6, 1, '2015-06-01', 6, 36000),
32 (7, 7, 1, '2015-07-01', 7, 49000),
33 (8, 8, 1, '2015-06-01', 8, 64000),
34 (9, 9, 1, '2015-04-01', 9, 81000),
35 (10, 10, 1, '2015-10-01', 10, 100000),
36 (11, 1, 2, '2015-11-01', 1, 2000),
37 (13, 3, 2, '2015-01-01', 3, 6000),
38 (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

```
1 SELECT concat(cust_lname , ' ' , cust_fname) as name
2 FROM CUSTOMERS
3 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.

```
1 CREATE TRIGGER update_cust_balance
2 AFTER INSERT ON INVOICE
3 FOR EACH ROW
4 BEGIN
5     UPDATE CUSTOMERS
6     SET cust_balance = cust_balance + NEW.inv_amount
7     WHERE cust_num = NEW.cust_num;
8 END;
```

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

(a) Relational Algebra

$$\pi_{\text{cust_lname} + " " + \text{cust_fname}}(\sigma_{\text{unit_sold} \geq 3}((\text{CUSTOMER} \bowtie \sigma_{\text{unit_sold}}((\text{cust_num}, \text{inv_date} > \text{prod_num} \mathcal{F}_{\text{sum}(\text{unit_sold})}(\text{INVOICE}))))))$$

(b) SQL

```
1 SELECT concat(cust_lname , ' ' , cust_fname) as name
2 FROM CUSTOMERS
3 WHERE cust_num IN
4 (
5     SELECT cust_num
6     FROM INVOICE
7     GROUP BY cust_num, inv_date, prod_num
8     HAVING sum(unit_sold) >= 3
9 );
```

(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

$CUSTOMER] \bowtie INVOICE$

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

(b) Right Outer Join

$CUSTOMER \bowtie [INVOICE$

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

(c) Full Outer Join

$CUSTOMER] \bowtie [INVOICE$

```
1 SELECT CONCAT(C.cust_fname, " ", C.cust_lname) as name
2 LEFT JOIN INVOICE i
3 ON C.cust_num=i.cust_num
4 UNION
5 SELECT CONCAT(C.cust_fname, " ", C.cust_lname) as name, i.inv_amount
6 RIGHT JOIN INVOICE i
7 ON C.cust_num=i.cust_num
```

5. Count number of products sold on each date.

(a) Relational Algebra

$$\pi_{\text{inv_date}, \text{sum}(\text{unit_sold})}(\text{inv_date } \mathcal{F}_{\text{sum}(\text{unit_sold})}(\text{INVOICE}))$$

(b) SQL

```
1 SELECT inv_date, sum(unit_sold)
2 FROM INVOICE
3 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

```
1 CREATE TABLE GOLD_CUSTOMER
2 (
3     cust_num int,
4     PRIMARY KEY (cust_num),
5     FOREIGN KEY (cust_num) REFERENCES CUSTOMERS (cust_num)
6 )
```

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

```
1 CREATE TRIGGER update_gold_customer
2 AFTER INSERT ON INVOICE
3 FOR EACH ROW
4 BEGIN
5     IF NEW.cust_balance > 100000
6     AND NEW.cust_num NOT IN (SELECT cust_num FROM GOLD_CUSTOMER) THEN
7         INSERT INTO GOLD_CUSTOMER VALUES (NEW.cust_num);
8     END IF;
9 END
```

7. Add a new attribute CUST_DOB in customer table

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

References

- [1] KorigamiK, “Dbms lab practical.” <https://github.com/korigamik/semester-3>, 2022.