DBMS GATE Questions

Mohammad Asad Ali

Mohammad Najmuddin

Modabbir Farid

Osama Ayub Gazi

Suhel Khan

Syed Monis

Mohammad Irshad (CBCS)

Group 7

GATE

Year 2010

Q.1)

A relational schema for a train reservation database is given below. Passenger (pid, pname, age) Reservation (pid, class, tid)

Table: Passenger

pid	pname	age
0	Sachin	65
1	Rahul	66
2	Sourav	67
3	Anil	69

Table : Reservation

pid	class	tid
0	AC	8200
1	AC	8201
2	SC	8201
5	AC	8203
1	SC	8204
3	AC	8202

What pids are returned by the following SQL query for the above instance of the tables?

```
SLECT pid

FROM Reservation ,

WHERE class ‘AC’ AND

    EXISTS (SELECT *

       FROM Passenger

       WHERE age > 65 AND

       Passenger. pid = Reservation.pid)
```

A 1, 0

B 1, 2

C 1, 3

D 1, 5

Solution :

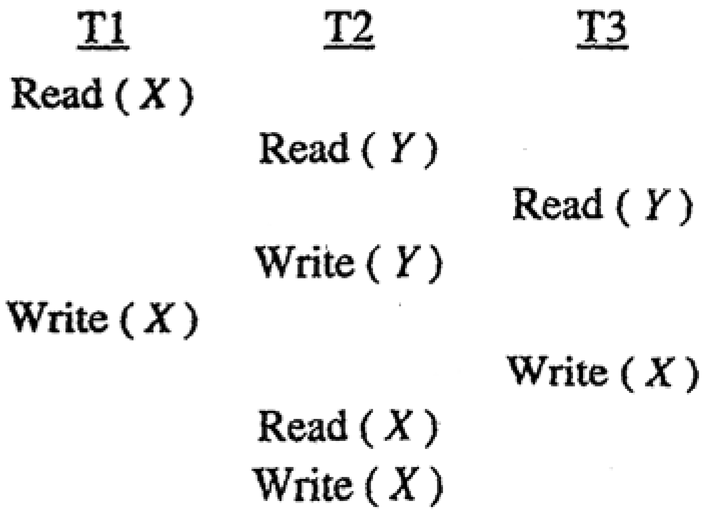
->Option C

When a subquery uses values from outer query, the subquery is called correlated subquery. The correlated subquery is evaluated once for each row processed by the outer query.

The outer query selects 4 entries (with pids as 0, 1, 5, 3) from Reservation table. Out of these selected entries, the subquery returns Non-Null values only for 1 and 3

Q.2)

Consider the following schedule for transactions T1, T2 and T3: Which one of the schedules below is the correct serialization of the above?



T1	T2	T3
Read (X)		
	Read (Y)	
		Read (Y)
	Write (Y)	
Write (X)		
		Write (X)
	Read (X)	
	Write (X)	

A T1->>T3->>T2

B T2->>T1->>T3

C T2->>T3->>T1

D T3->>T1->>T2

Solution :

->Option A

T1 can complete before T2 and T3 as there is no conflict between Write(X) of T1 and the operations in T2 and T3 which occur before Write(X) of T1 in the above diagram.
T3 should can complete before T2 as the Read(Y) of T3 doesn’t conflict with Read(Y) of T2.
Similarly, Write(X) of T3 doesn’t conflict with Read(Y) and Write(Y) operations of T2.
Another way to solve this question is to create a dependency graph and topologically sort the dependency graph. After topologically sorting, we can see the sequence T1, T3, T2.

Q.3)

Which of the following functional dependencies hold for relations R(A, B, C) and S(B, D, E):

```
B -> A

A -> C
```

The relation R contains 200 tuples and the rel ation S contains 100 tuples. What is the maximum number of tuples possible in the natural join of R and S (R natural join S)

A 100

B 200

C 300

D 2000

Solution:

->Option A

From the given set of functional dependencies, it can be observed that B is a candidate key of R. So all 200 values of B must be unique in R. There is no functional dependency given for S. To get the maximum number of tuples in output, there can be two possibilities for S.

1) All 100 values of B in S are same and there is an entry in R that matches with this value. In this case, we get 100 tuples in output.

2) All 100 values of B in S are different and these values are present in R also. In this case also, we get 100 tuples.

GATE

Year 2011

Q.1)

Consider a relational table with a single record for each registered student with the following attributes.

1. *Registration_Num*: Unique registration number of each registered student
2. *UID*: Unique identity number, unique at the national level for each citizen
3. *BankAccount_Num*: Unique account number at the bank. A student can have multiple accounts or join accounts. This attribute stores the primary account number.
4. *Name*: Name of the student
5. *Hostel_Room*: Room number of the hostel

Which one of the following option is **INCORRECT**?

A BankAccount_Num is candidate key

B Registration_Num can be a primary key

C UID is candidate key if all students are from the same country

D If S is a superkey such that S∩UID is NULL then S∪UID is also a superkey

Solution :

->Option A

A Candidate Key value must uniquely identify the corresponding row in table. BankAccount_Number is not a candidate key. As per the question “A student can have multiple accounts or joint accounts. This attributes stores the primary account number”. If two students have a joint account and if the joint account is their primary account, then BankAccount_Number value cannot uniquely identify a row.

Q.2)

Consider a database table T containing two columns X and Y each of type integer. After the creation of the table, one record (X=1, Y=1) is inserted in the table. Let MX and My denote the respective maximum values of X and Y among all records in the table at any point in time. Using MX and MY, new records are inserted in the table 128 times with X and Y values being MX+1, 2*MY+1 respectively. It may be noted that each time after the insertion, values of MX and MY change. What will be the output of the following SQL query after the steps mentioned above are carried out?

```
SELECT Y FROM T WHERE X=7;
```

A 127

B 255

C 129

D 257

Solution :

->Option A

First entry of the table will be

X	Y
1	1
2	3
3	7
4	15
5	31
6	63
7	127

So, when X=7 the value of Y is 127

Q.3)

Database table by name Loan_Records is given below.

Borrower	Bank_Manager	Loan_Amount
Ramesh	Sunderajan	10000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	7000.00

What is the output of the following SQL query?

```
SELECT Count(*)
FROM  ( ( SELECT Borrower, Bank_Manager
          FROM Loan_Records) AS S
          NATURAL JOIN ( SELECT Bank_Manager,
Loan_Amount
                         FROM Loan_Records) AS T );
```

A 3

B 9

C 5

D 6

Solution :

->Option C

The two given subquery brings the table S and T on there cross product we get the 3X3 = 9 tuples. Making the Natural Join we will get only those values of the Bank_Manager which are matching. This will produce a table of 5 tuples. Aggregate Function Count will give us total number of tuples that is equal to 5

Q.4)

Consider a relational table r with sufficient number of records, having attributes A1, A2,…, An and let 1 <= p <= n. Two queries Q1 and Q2 are given below.

Q1 : $\pi_{A1 \ldots A_n}\left(\sigma_{A_p = c}(r)\right)$ where c is a const

Q2: $\pi_{A1 \ldots A_n}\left(\sigma_{c_1 \leq A_p \leq c_2}(r)\right)$ where c_1 and c_2 are constants

The database can be configured to do ordered indexing on Ap or hashing on Ap. Which of the following statements is TRUE?

A Ordered indexing will always outperform hashing for both queries

B Hashing will always outperform ordered indexing for both queries

C Hashing will outperform ordered indexing on Q1, but not on Q2

D Hashing will outperform ordered indexing on Q2, but not on Q1.

Solution :

->Option C

If record are accessed for a particular value from table, hashing will do better. If records are accessed in a range of values, ordered indexing will perform better.