IIITS/S-2019/End Semester Exams Date: 24th April 2019

Indian Institute of Information Technology, Sri City, Chittoor

Name of the Exam: Database Management Systems Duration: 3 hr Max. Marks: 40

Roll No.:____________________________________Room No.: _______________Seat No.: __________

Name: _____________________________________Invigilator's Signature: _______________________

Instructions: 1. All questions have to be answered in the box space provided only.
2. You have to do rough work in the question paper if required in the last sheet.

Q1. Multiple Choice Questions. Write the answer for the following questions in the space provided. Only one answer to be selected. (1x10=10 marks)

i) Atomicity of transactions refers to the requirement that
(a) each transaction updates exactly one indivisible entity of a database
(b) either all operations of the transaction are reflected in the database or none are
(c) each transaction preserves the correctness of the database
(d) when one transaction executes all other transactions should be made to wait

Ans: □

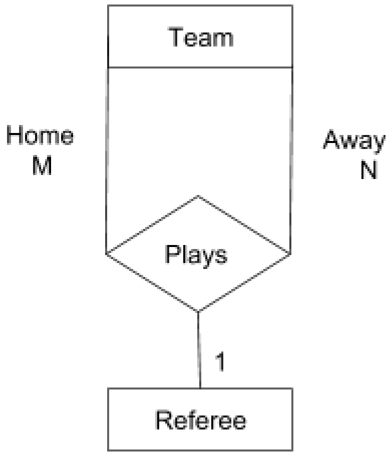
ii) Under the 2-phase locking protocol
(a) once a transaction has released a lock it can not acquire any more locks
(b) a transaction can release all its locks when it commits
(c) a transaction can acquire all the locks it needs when it begins
(d) all of the above

Ans: □

iii) Given a schedule for transactions t1 and t2 we can say that
(a) the schedule can be serialized if t1 and t2 resulted from the use of two-phase locking
(b) the transactions compute the correct result if t1 executed only after t2 committed
(c) the transactions compute the correct result if t1 and t2 executed concurrently but did not access any common data items
(d) all of the above statements are correct

Ans: □

iv) Consider the following ER Diagram



Which one of the following problem descriptions could have lead to the ER model above?
(a) We are modeling a volleyball tournament. A volleyball game is played by two teams. Multiple referees make sure during a single game that the rules are respected.
(b) A volleyball game in a tournament is played by two teams. A referee has to be present at each game to make sure the rules are respected.
(c) Two teams play volleyball in a tournament. In each game, one has the role of “home” team and the other one has the role of “away” team. Each play is supervised by at most one referee.
(d) None of the above

Ans:

v) Which of the following statements is true?
(a) Transactions do not need to be isolated in databases where a majority of applications are read-only and a minority of applications perform updates.
(b) If transactions are not isolated from one another, lost updates might occur for mixed read-write applications.
(c) A history with only read operations is not always serializable
(d) When determining the serializability of a history, only the fact whether transactions write or not is important, no matter which objects the different transactions write.

Ans:

Answer question vi) –viii) referring the following schema of an employee database.

departments: (dept_no, dept_name)
dept_emp: (emp_no, dept_no, from_date, to_date)
dept_manager: (emp_no, dept_no, from_date, to_date)
employees: (emp_no, birth_date, first_name, last_name, gender, hire_date)
salaries: (emp_no, salary, from_date, to_date)
title: (emp_no, title, from_date, to_date)

vi) The following queries try to compute the average salary of all current employees per department and to display department names and salary averages in descending order of the average salary. Which one of the following performs the intended task

```
(a) SELECT d.dept_name ,AVG(s.salary) AS avgs
FROM employees e
INNER JOIN salaries s ON e.emp_no = s.emp_no
INNER JOIN dept_emp de ON de.emp_no = s.emp_no
INNER JOIN departments d ON d.dept_no = de.dept_no
WHERE s.to_date > NOW()
        AND de.to_date > NOW()
GROUP BY d.dept_name
ORDER BY avgs DESC
```

```
(b) SELECT d.dept_name ,AVG(s.salary) AS avgs
FROM employees e
INNER JOIN salaries s ON e.emp_no = s.emp_no
INNER JOIN dept_emp de ON de.emp_no = s.emp_no
INNER JOIN departments d ON d.dept_no = de.dept_no
GROUP BY d.dept_name
ORDER BY avgs DESC
```

```
(c) SELECT e.dept_no ,AVG(s.salary) AS avgs
FROM employees e
INNER JOIN salaries s ON e.emp_no = s.emp_no
INNER JOIN dept_emp de ON de.emp_no = s.emp_no
WHERE s.to_date > NOW()
        AND de.to_date > NOW()
ORDER BY avgs DESC
```

(d) None of the above

Ans: □

vii) Consider the following query. Select the statement which describes what the query is meant to do.

```
SELECT e.first_name
        ,e.last_name
        ,d.dept_name
        ,s.salary
        ,avgst.avgs
FROM employees e
INNER JOIN salaries s ON s.emp_no = e.emp_no
INNER JOIN dept_manager dm ON dm.emp_no = e.emp_no
INNER JOIN departments d ON d.dept_no = dm.dept_no
INNER JOIN (
	SELECT de.dept_no
		,AVG(s.salary) AS avgs
	FROM employees e
	INNER JOIN salaries s ON e.emp_no = s.emp_no
	INNER JOIN dept_emp de ON de.emp_no = s.emp_no
	WHERE s.to_date > NOW()
		AND de.to_date > NOW()
	GROUP BY de.dept_no
	) avgst ON avgst.dept_no = dm.dept_no
WHERE dm.to_date > NOW()
	AND s.to_date > NOW()
	AND s.salary > avgst.avgs
```

(a) It returns names of managers (first_name and last_name) who earn more than the average salary of all employees that they have ever managed. It also returns department's name (dept_name), manager's salary and department's average salary.
(b) It returns names of employees (first_name and last_name) who earn the most in their department. It also returns department's name (dept_name), employees' salary and department's average salary.
(c) It returns names of managers (first_name and last_name) who earn more than the average salary of the current employees that they manage. It also returns department's name (dept_name), manager's salary and department's average salary.
(d) It returns names of employees (first_name and last_name) who earn no more than the average salary of the current employees of their same department. It also returns department's name (dept_name), and department's average salary.

Ans:

viii) Which query or queries return the correct count of employees in the database whose first name starts with the letter B?

(a)
```
SELECT COUNT(first_name)
FROM employees
WHERE first_name LIKE '%B%'
```

(b)
```
SELECT COUNT(first_name)
FROM employees
WHERE SUBSTRING(first_name, 0, 1) IN ('B')
```

(c)
```
SELECT COUNT(first_name)
FROM employees
WHERE first_name LIKE 'B'
	OR first_name LIKE 'b'
```

(d)
```
SELECT COUNT(first_name)
FROM employees
WHERE SUBSTRING(first_name, 1, 1) = 'B'
```

Ans:

Answer Question ix) and x) based on a metro rail transportation system. For reference, the schema of this database is shown below.

stops: (stop_id, stop_name, stop_lat, stop_lon)
stop_times: (trip_id, departure_time, arrival_time, stop_id, stop_sequence)
trips: (trip_id, train_number, trip_headsign)

ix) Given the following query which is the statement that describes best the operation it performs.

```
SELECT COUNT(*) AS count_trips , train_number
FROM trips
GROUP BY train_number
ORDER BY count_trips DESC
```

(a) Finds how many train lines (train_number) are in a trip and orders its result in descending order.
(b) Finds the number of trips each train line (train_number) makes and orders its result by number of trips in descending order.
(c) Finds and counts the number of train lines (train_number) and orders them in descending order
(d) Finds the trips each train line (train_number) makes and orders its result by number of trips in ascending order.

Ans:

x) The following queries try to compute the top 10 stations according to the number of times that any train stops at that station.

```
(a) SELECT s.stop_id
        ,s.stop_name
FROM stop_times st
        ,stops s
WHERE s.stop_id = st.stop_id
GROUP BY st.stop_id
        ,s.stop_name
ORDER BY count_stations DESC
```

```
(b) SELECT count_stations
        ,s.stop_name
FROM stop_times st
        ,stops s
WHERE s.stop_id = st.stop_id
GROUP BY st.stop_id
ORDER BY count_stations DESC LIMIT 10
```

```
(c) SELECT COUNT(*) AS count_stations
            ,s.stop_name
FROM stop_times st
            ,stops s
WHERE s.stop_id = st.stop_id
GROUP BY st.stop_id
            ,s.stop_name
ORDER BY count_stations DESC LIMIT 10
```

```
(d) SELECT COUNT(*) AS count_stations
            ,s.stop_name
FROM stop_times st
            ,stops s
WHERE s.stop_id = st.stop_id
GROUP BY st.stop_id
ORDER BY count_stations DESC
```

Ans:

Q2. Subjective Questions. Answer the following questions in the space provided only. (1x5=5 marks)

i) Draw the steps in high-level query processing.

ii) Explain RAID. Also draw and explain the working principle in RAID 0, RAID 1 and RAID 10 technologies.

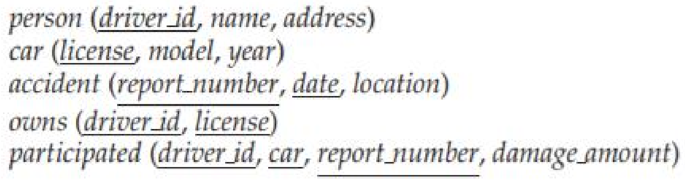
iii) Give 2 differences between Main Memory and Cache.

iv) What are materialized views in SQL? Give 2 difference between view and materialized views in database?

v) Explain CAP theorem in detail in NoSQL databases

Q3. SQL Queries (3+1+1=5 marks)

i) Consider the insurance database with the following schema where the primary keys are underlined. Construct the following SQL queries for this relational database.



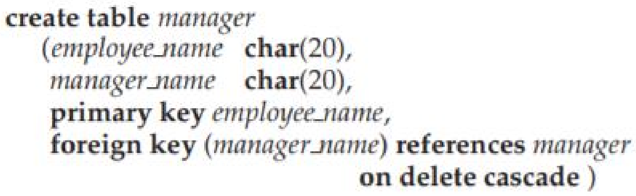
person (driver_id, name, address)
car (license, model, year)
accident (report_number, date, location)
owns (driver_id, license)
participated (driver_id, car, report_number, damage_amount)

a. Find the total number of people who owned cars that were involved in accidents in 1989.

b. Add a new accident to the database; assume any values for required attributes.

c. Delete the Mazda belonging to “John Smith”.

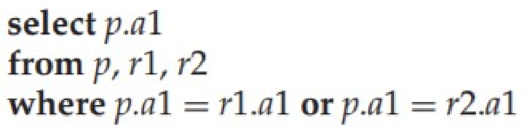
ii) SQL allows a foreign-key dependency to refer to the same relation, as in the following example:



```
create table manager
    (employee_name   char(20),
     manager_name    char(20),
     primary key employee_name,
     foreign key (manager_name) references manager
                                    on delete cascade )
```

Here, employee name is a key to the table manager, meaning that each employee has at most one manager. The foreign-key clause requires that every manager also be an employee. Explain exactly what happens when a tuple in the relation manager is deleted.

iii) Consider the SQL query:



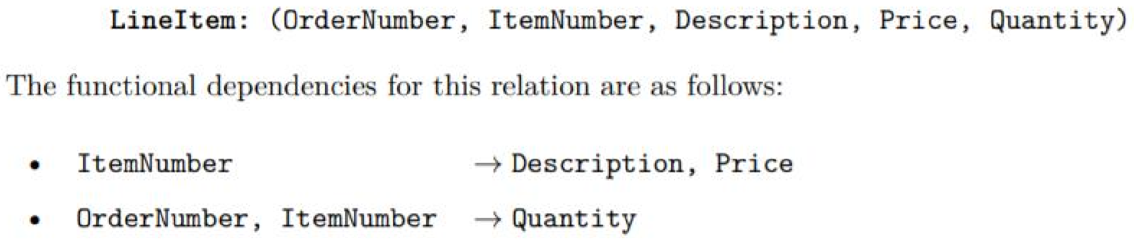
```
select p.a1
from p, r1, r2
where p.a1 = r1.a1 or p.a1 = r2.a1
```

Under what conditions does the preceding query select values of p.a1 that are either in r1 or in r2?

Examine carefully the cases where one of r1 or r2 may be empty.

Q4. Normalization (3+2=5 marks)

i) Consider the following relational schema:



LineItem: (OrderNumber, ItemNumber, Description, Price, Quantity)

The functional dependencies for this relation are as follows:

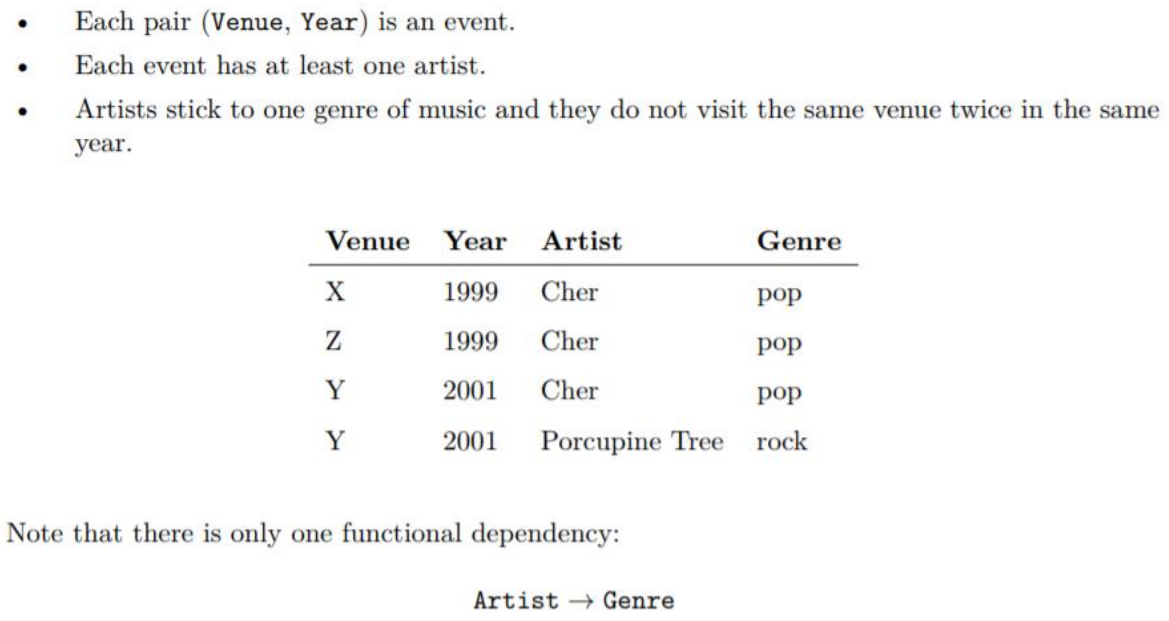
- ItemNumber → Description, Price
- OrderNumber, ItemNumber → Quantity

a) Find the candidate key(s) of the relation above.

b) What normal form is the above LineItem relation in?

c) What are some disadvantages of this choice of schema?

ii) Consider the following relational schema describing musical events in Switzerland. For the questions below, you can make the following assumptions:



- Each pair (`Venue`, `Year`) is an event.
- Each event has at least one artist.
- Artists stick to one genre of music and they do not visit the same venue twice in the same year.

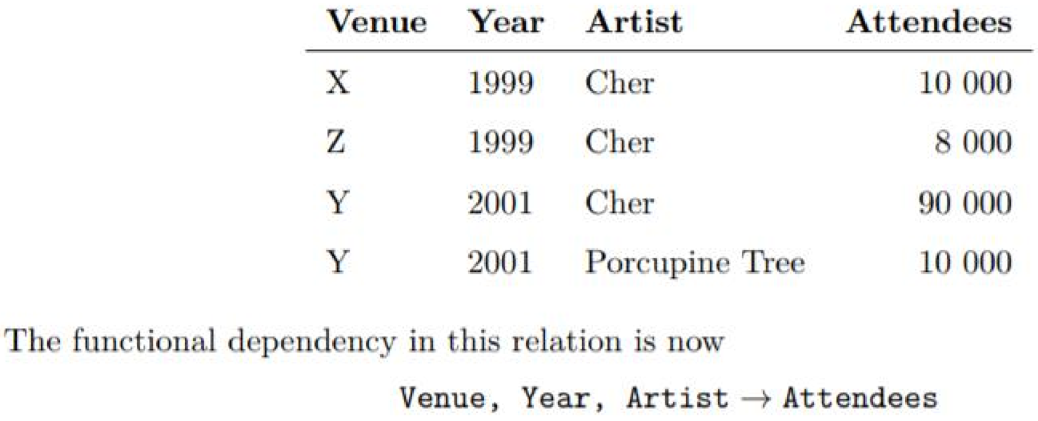
Venue	**Year**	**Artist**	**Genre**
X	1999	Cher	pop
Z	1999	Cher	pop
Y	2001	Cher	pop
Y	2001	Porcupine Tree	rock

Note that there is only one functional dependency:

`Artist` → `Genre`

a) Is this relation in 2NF? 3NF? Determine the keys and justify your answer.

b) We now modify the schema to include the number of attendees, but no genre:



Venue	Year	Artist	Attendees
X	1999	Cher	10 000
Z	1999	Cher	8 000
Y	2001	Cher	90 000
Y	2001	Porcupine Tree	10 000

The functional dependency in this relation is now

`Venue, Year, Artist` → `Attendees`

Is the new schema in 2NF or 3NF? Justify.

Q5. Transaction Management & Concurrency Control Techniques (5+5+5=15 marks)

i) Check whether the following schedules are conflict serializable or not (2.5x2=5 marks)

a) First Schedule

T1	T2	T3
R(a)		
	R(b)	
		R(c)
		W(c)
	W(b)	
W(a)		

b) Second Schedule

T1	T2	T3	T4
	R(x)		
		W(x)	
		Commit	
W(x)			
Commit			
	W(y)		
	R(z)		
	Commit		
			R(x)
			R(y)
			Commit

ii) Consider the timestamp-ordering protocol, and two transactions, one that writes two data items p and q, and another that reads the same two data items. Give a schedule whereby the timestamp test for a write operation fails and causes the first transaction to be restarted, in turn causing a cascading abort of the other transaction. Show how this could result in starvation in both transactions. (Such a situation, where two or more processes carry out actions, but are unable to complete their task because of interaction with the other processes, is called a livestock). (5 marks)

iii) Consider the following two transactions:

T_{31} : read(A);
      read(B);
      if A=0 then B:=B+1;
      write(B).

T_{32} : read(B);
      read(A);
      if B=0 then A:=A+1;
      write(A).

Add lock and unlock instructions to transactions T_{31} and T_{32}, so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock? Justify your answer how it may result in a deadlock. (5 marks)

ROUGH WORK

IIITS/S-2019/End Semester Exams Date: 24th April 2019

Indian Institute of Information Technology, Sri City, Chittoor

Name of the Exam: Database Management Systems Duration: 3 hr Max. Marks: 40

Roll No.:____Room No.: ________________Seat No.: ___________

Name: ____Invigilator's Signature: _________________________

Instructions: 1. All questions have to be answered in the box space provided only.
2. You have to do rough work in the question paper if required in the last sheet.

Q1. Multiple Choice Questions. Write the answer for the following questions in the space provided. Only one answer to be selected. (1x10=10 marks)

i) Atomicity of transactions refers to the requirement that
(a) each transaction updates exactly one indivisible entity of a database
(b) either all operations of the transaction are reflected in the database or none are
(c) each transaction preserves the correctness of the database
(d) when one transaction executes all other transactions should be made to wait

Ans: B

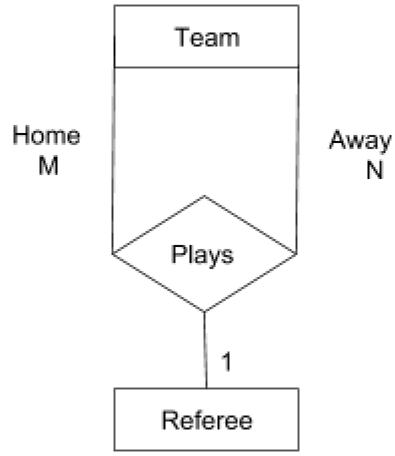
ii) Under the 2-phase locking protocol
(a) once a transaction has released a lock it can not acquire any more locks
(b) a transaction can release all its locks when it commits
(c) a transaction can acquire all the locks it needs when it begins
(d) all of the above

Ans: D

iii) Given a schedule for transactions t1 and t2 we can say that
(a) the schedule can be serialized if t1 and t2 resulted from the use of two-phase locking
(b) the transactions compute the correct result if t1 executed only after t2 committed
(c) the transactions compute the correct result if t1 and t2 executed concurrently but did not access any common data items
(d) all of the above statements are correct

Ans: D

iv) Consider the following ER Diagram



Which one of the following problem descriptions could have lead to the ER model above?
(a) We are modeling a volleyball tournament. A volleyball game is played by two teams. Multiple referees make sure during a single game that the rules are respected.
(b) A volleyball game in a tournament is played by two teams. A referee has to be present at each game to make sure the rules are respected.
(c) Two teams play volleyball in a tournament. In each game, one has the role of “home” team and the other one has the role of “away” team. Each play is supervised by at most one referee.
(d) None of the above

Ans: B

v) Which of the following statements is true?
(a) Transactions do not need to be isolated in databases where a majority of applications are read-only and a minority of applications perform updates.
(b) If transactions are not isolated from one another, lost updates might occur for mixed read-write applications.
(c) A history with only read operations is not always serializable
(d) When determining the serializability of a history, only the fact whether transactions write or not is important, no matter which objects the different transactions write.

Ans: B

Answer question vi) –viii) referring the following schema of an employee database.

departments: (dept_no, dept_name)
dept_emp: (emp_no, dept_no, from_date, to_date)
dept_manager: (emp_no, dept_no, from_date, to_date)
employees: (emp_no, birth_date, first_name, last_name, gender, hire_date)
salaries: (emp_no, salary, from_date, to_date)
title: (emp_no, title, from_date, to_date)

vi) The following queries try to compute the average salary of all current employees per department and to display department names and salary averages in descending order of the average salary. Which one of the following performs the intended task

```
(a) SELECT d.dept_name ,AVG(s.salary) AS avgs
FROM employees e
INNER JOIN salaries s ON e.emp_no = s.emp_no
INNER JOIN dept_emp de ON de.emp_no = s.emp_no
INNER JOIN departments d ON d.dept_no = de.dept_no
WHERE s.to_date > NOW()
        AND de.to_date > NOW()
GROUP BY d.dept_name
ORDER BY avgs DESC
```

```
(b) SELECT d.dept_name ,AVG(s.salary) AS avgs
FROM employees e
INNER JOIN salaries s ON e.emp_no = s.emp_no
INNER JOIN dept_emp de ON de.emp_no = s.emp_no
INNER JOIN departments d ON d.dept_no = de.dept_no
GROUP BY d.dept_name
ORDER BY avgs DESC
```

```
(c) SELECT e.dept_no ,AVG(s.salary) AS avgs
FROM employees e
INNER JOIN salaries s ON e.emp_no = s.emp_no
INNER JOIN dept_emp de ON de.emp_no = s.emp_no
WHERE s.to_date > NOW()
        AND de.to_date > NOW()
ORDER BY avgs DESC
```

(d) None of the above

Ans: A

vii) Consider the following query. Select the statement which describes what the query is meant to do.

```
SELECT e.first_name
        ,e.last_name
        ,d.dept_name
        ,s.salary
        ,avgst.avgs
FROM employees e
INNER JOIN salaries s ON s.emp_no = e.emp_no
INNER JOIN dept_manager dm ON dm.emp_no = e.emp_no
INNER JOIN departments d ON d.dept_no = dm.dept_no
INNER JOIN (
        SELECT de.dept_no
                ,AVG(s.salary) AS avgs
        FROM employees e
        INNER JOIN salaries s ON e.emp_no = s.emp_no
        INNER JOIN dept_emp de ON de.emp_no = s.emp_no
        WHERE s.to_date > NOW()
                AND de.to_date > NOW()
        GROUP BY de.dept_no
        ) avgst ON avgst.dept_no = dm.dept_no
WHERE dm.to_date > NOW()
        AND s.to_date > NOW()
        AND s.salary > avgst.avgs
```

(a) It returns names of managers (first_name and last_name) who earn more than the average salary of all employees that they have ever managed. It also returns department's name (dept_name), manager's salary and department's average salary.
(b) It returns names of employees (first_name and last_name) who earn the most in their department. It also returns department's name (dept_name), employees' salary and department's average salary.
(c) It returns names of managers (first_name and last_name) who earn more than the average salary of the current employees that they manage. It also returns department's name (dept_name), manager's salary and department's average salary.
(d) It returns names of employees (first_name and last_name) who earn no more than the average salary of the current employees of their same department. It also returns department's name (dept_name), and department's average salary.

Ans: C

viii) Which query or queries return the correct count of employees in the database whose first name starts with the letter B?

(a)
```
SELECT COUNT(first_name)
FROM employees
WHERE first_name LIKE '%B%'
```

(b)
```
SELECT COUNT(first_name)
FROM employees
WHERE SUBSTRING(first_name, 0, 1) IN ('B')
```

(c)
```
SELECT COUNT(first_name)
FROM employees
WHERE first_name LIKE 'B'
        OR first_name LIKE 'b'
```

(d)
```
SELECT COUNT(first_name)
FROM employees
WHERE SUBSTRING(first_name, 1, 1) = 'B'
```

Ans: D

Answer Question ix) and x) based on a metro rail transportation system. For reference, the schema of this database is shown below.

stops: (stop_id, stop_name, stop_lat, stop_lon)
stop_times: (trip_id, departure_time, arrival_time, stop_id, stop_sequence)
trips: (trip_id, train_number, trip_headsign)

ix) Given the following query which is the statement that describes best the operation it performs.

```
SELECT COUNT(*) AS count_trips , train_number
FROM trips
GROUP BY train_number
ORDER BY count_trips DESC
```

(a) Finds how many train lines (train_number) are in a trip and orders its result in descending order.
(b) Finds the number of trips each train line (train_number) makes and orders its result by number of trips in descending order.
(c) Finds and counts the number of train lines (train_number) and orders them in descending order
(d) Finds the trips each train line (train_number) makes and orders its result by number of trips in ascending order.

Ans: B

x) The following queries try to compute the top 10 stations according to the number of times that any train stops at that station.

```
(a) SELECT s.stop_id
        ,s.stop_name
FROM stop_times st
        ,stops s
WHERE s.stop_id = st.stop_id
GROUP BY st.stop_id
        ,s.stop_name
ORDER BY count_stations DESC
```

```
(b) SELECT count_stations
        ,s.stop_name
FROM stop_times st
        ,stops s
WHERE s.stop_id = st.stop_id
GROUP BY st.stop_id
ORDER BY count_stations DESC LIMIT 10
```

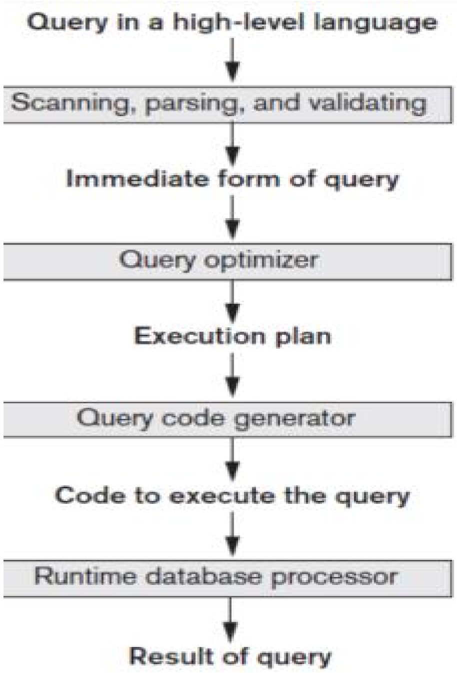
```
(c) SELECT COUNT(*) AS count_stations
        ,s.stop_name
FROM stop_times st
        ,stops s
WHERE s.stop_id = st.stop_id
GROUP BY st.stop_id
        ,s.stop_name
ORDER BY count_stations DESC LIMIT 10
```

```
(d) SELECT COUNT(*) AS count_stations
        ,s.stop_name
FROM stop_times st
        ,stops s
WHERE s.stop_id = st.stop_id
GROUP BY st.stop_id
ORDER BY count_stations DESC
```

Ans: C

Q2. Subjective Questions. Answer the following questions in the space provided only. (1x5=5 marks)(1/2:Partial Correct; 1:Complete Correct)

i) Draw the steps in high-level query processing.


Query in a high-level language
Scanning, parsing, and validating
Immediate form of query
Query optimizer
Execution plan
Query code generator
Code to execute the query
Runtime database processor
Result of query

ii) Explain RAID. Also draw and explain the working principle in RAID 0, RAID 1 and RAID 10 technologies.

Redundant arrays of independent disks (RAID). Goal: improve disk speed and access time

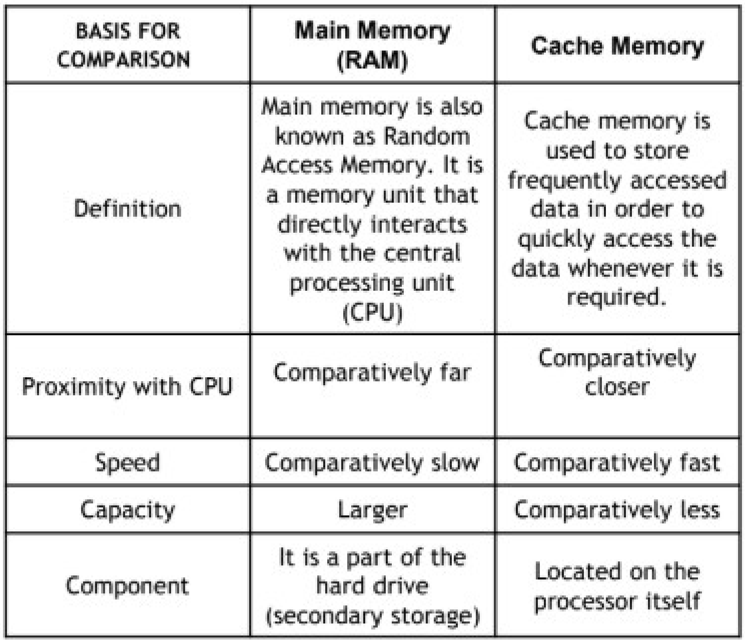
RAID 0: Striped disk array without fault tolerance
This configuration has striping, but no redundancy of data. It offers the best performance, but no-fault tolerance.

RAID 1: Mirroring and duplexing.
Also called disk mirroring, this configuration consists of at least two drives that duplicate the storage of data.
Read performance is improved since either disk are often scan at an equivalent time.
Write performance is that the same as for single disk storage.

RAID 10 (RAID 1+0): A stripe of mirrors
Combining RAID one and RAID zero, this level is commonly stated as RAID ten, that offers higher performance than RAID one, however at a way higher value. In RAID 1+0, the information is reflected and therefore the mirrors are striped.

Pros: Higher performance. Fault tolerance.
Cons: Limited scalability. Lower usable capacity/High cost.
Ideal use: Extremely utilized database servers/ servers performing a lot of write operations.

iii) Give 2 differences between Main Memory and Cache.



BASIS FOR COMPARISON	Main Memory (RAM)	Cache Memory
Definition	Main memory is also known as Random Access Memory. It is a memory unit that directly interacts with the central processing unit (CPU)	Cache memory is used to store frequently accessed data in order to quickly access the data whenever it is required.
Proximity with CPU	Comparatively far	Comparatively closer
Speed	Comparatively slow	Comparatively fast
Capacity	Larger	Comparatively less
Component	It is a part of the hard drive (secondary storage)	Located on the processor itself

iv) What are materialized views in SQL? Give 2 difference between view and materialized views in database?

Materialized Views:

Materialized views are also the logical view of our data-driven by the select query but the result of the query will get stored in the table or disk, also the definition of the query will also store in the database.

When we see the performance of Materialized view it is better than normal View because the data of materialized view will be stored in table and table may be indexed so faster for joining also joining is done at the time of materialized views refresh time so no need to every time fire join statement as in case of view.

Difference:

1) The first difference between View and materialized view is that In Views query result is not stored in the disk or database but Materialized view allow to store the query result in disk or table.

2) Another difference between View vs materialized view is that, when we create a view using any table, rowid of view is same as the original table but in case of Materialized view rowid is different.

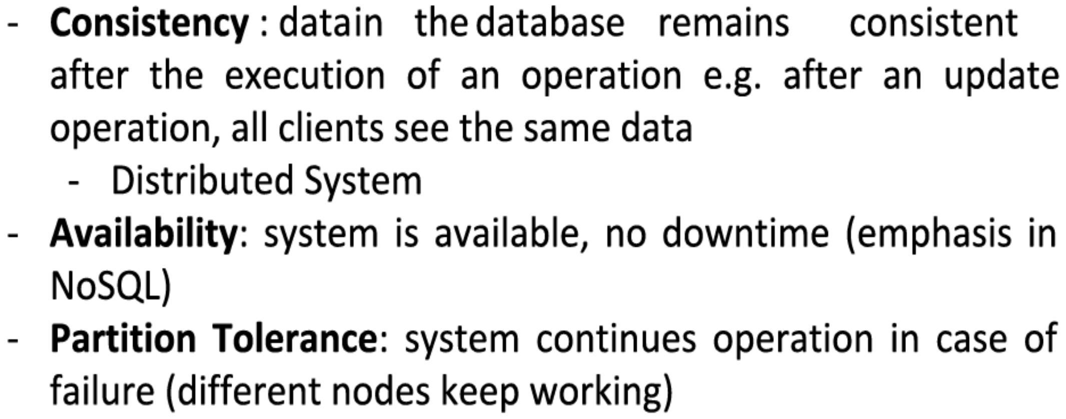
3) One more difference between View and materialized view in the database is that In case of View we always get latest data but in case of Materialized view we need to refresh the view for getting latest data.

4) Performance of View is less than Materialized view.

5) This is continuation of first difference between View and Materialized View, In case of view its only the logical view of table no separate copy of table but in case of Materialized view we get physically separate copy of table

6) Last difference between View vs Materialized View is that In case of Materialized view we need an extra trigger or some automatic method so that we can keep MV refreshed, this is not required for views in the database.

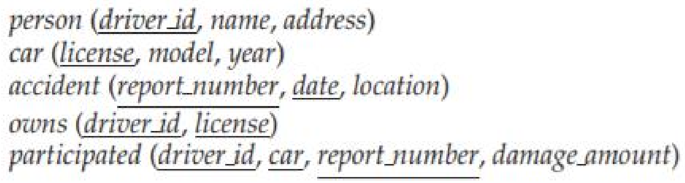
v) Explain CAP theorem in detail in NoSQL databases



- **Consistency** : datain the database remains consistent after the execution of an operation e.g. after an update operation, all clients see the same data
  - Distributed System
- **Availability**: system is available, no downtime (emphasis in NoSQL)
- **Partition Tolerance**: system continues operation in case of failure (different nodes keep working)

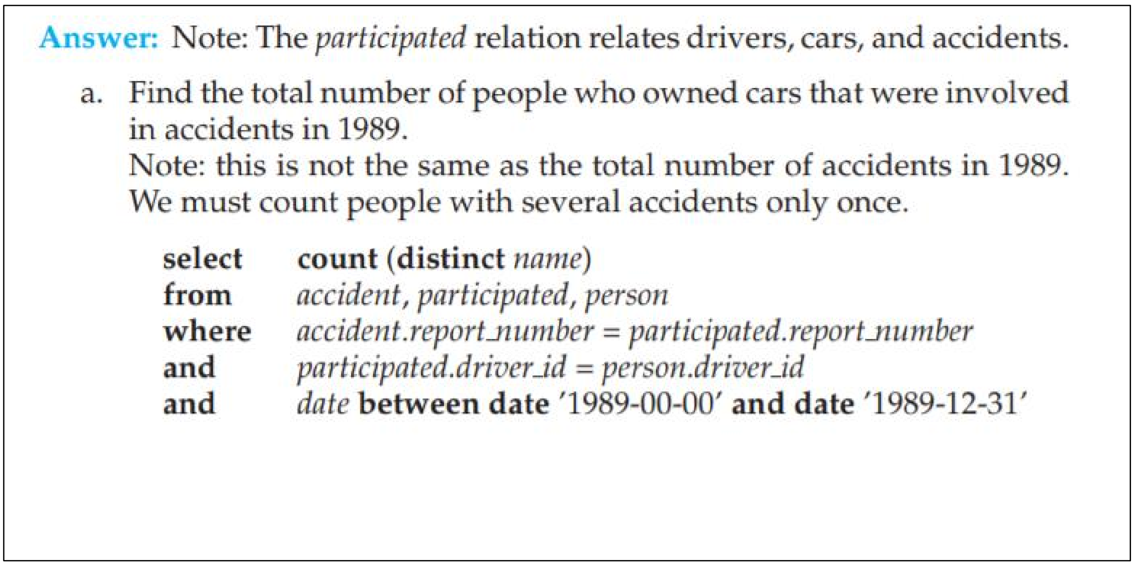
Q3. SQL Queries (3+1+1=5 marks)

i) Consider the insurance database with the following schema where the primary keys are underlined. Construct the following SQL queries for this relational database. (1x3=3 marks, 1 marks for complete correct)



person (*driver_id*, *name*, *address*)
car (*license*, *model*, *year*)
accident (*report_number*, *date*, *location*)
owns (*driver_id*, *license*)
participated (*driver_id*, *car*, *report_number*, *damage_amount*)

a. Find the total number of people who owned cars that were involved in accidents in 1989.

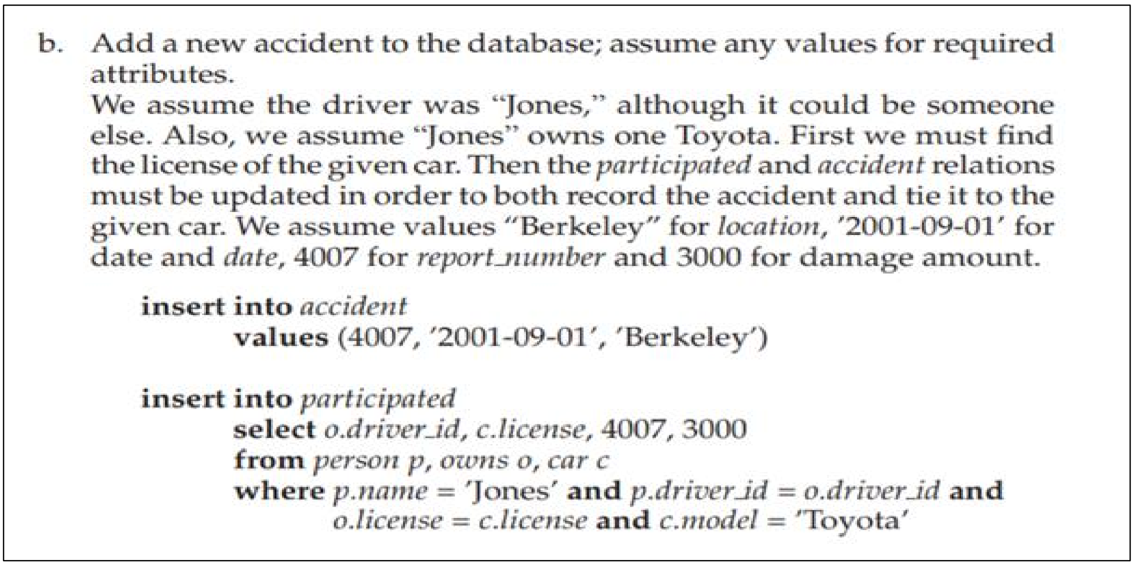


Answer: Note: The *participated* relation relates drivers, cars, and accidents.

a. Find the total number of people who owned cars that were involved in accidents in 1989.
Note: this is not the same as the total number of accidents in 1989. We must count people with several accidents only once.

```
select   count (distinct name)
from     accident, participated, person
where    accident.report_number = participated.report_number
and      participated.driver_id = person.driver_id
and      date between date '1989-00-00' and date '1989-12-31'
```

b. Add a new accident to the database; assume any values for required attributes.

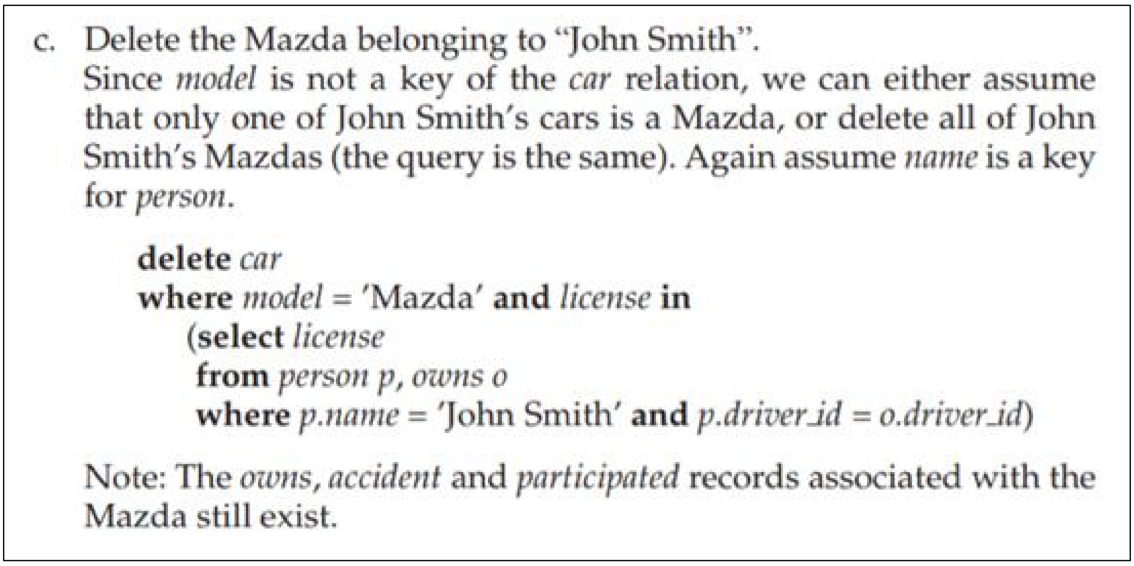


b. Add a new accident to the database; assume any values for required attributes.
We assume the driver was "Jones," although it could be someone else. Also, we assume "Jones" owns one Toyota. First we must find the license of the given car. Then the *participated* and *accident* relations must be updated in order to both record the accident and tie it to the given car. We assume values "Berkeley" for *location*, '2001-09-01' for date and *date*, 4007 for *report_number* and 3000 for damage amount.

```
insert into accident
        values (4007, '2001-09-01', 'Berkeley')

insert into participated
        select o.driver_id, c.license, 4007, 3000
        from person p, owns o, car c
        where p.name = 'Jones' and p.driver_id = o.driver_id and
              o.license = c.license and c.model = 'Toyota'
```

c. Delete the Mazda belonging to “John Smith”.

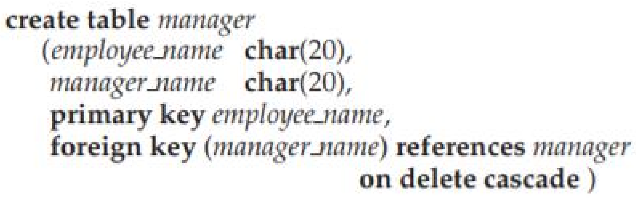


c. Delete the Mazda belonging to “John Smith”.
Since *model* is not a key of the *car* relation, we can either assume that only one of John Smith’s cars is a Mazda, or delete all of John Smith’s Mazdas (the query is the same). Again assume *name* is a key for *person*.

```
delete car
where model = 'Mazda' and license in
    (select license
     from person p, owns o
     where p.name = 'John Smith' and p.driver_id = o.driver_id)
```

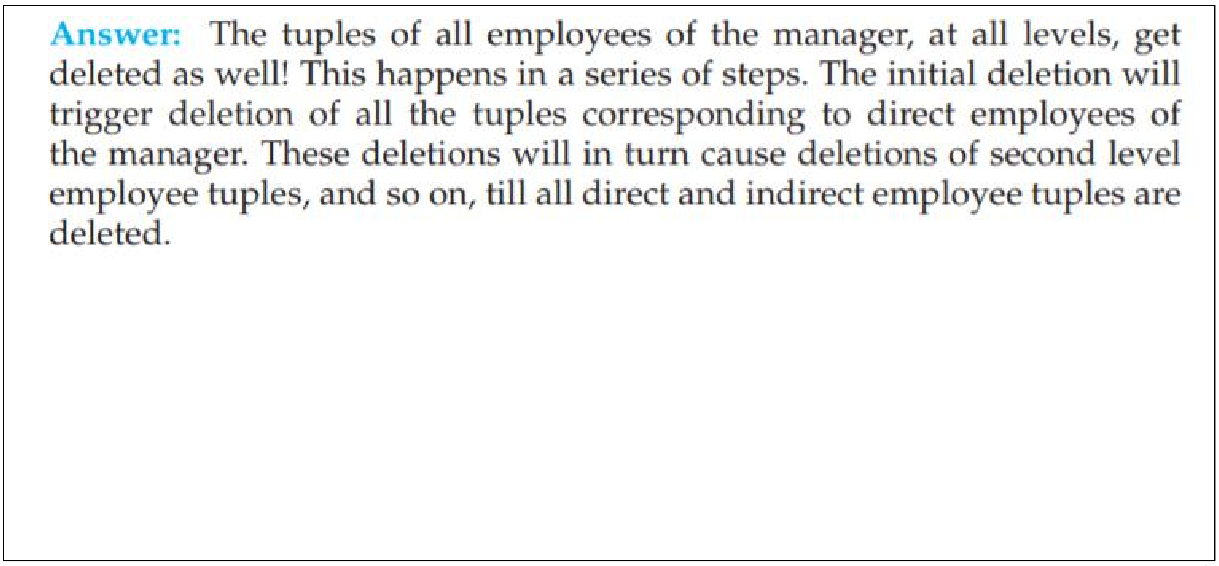
Note: The *owns*, *accident* and *participated* records associated with the Mazda still exist.

ii) SQL allows a foreign-key dependency to refer to the same relation, as in the following example: (1/2 partial correct; 1:Complete correct)



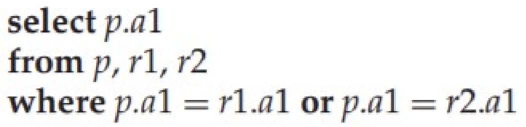
```
create table manager
    (employee_name   char(20),
     manager_name    char(20),
     primary key employee_name,
     foreign key (manager_name) references manager
                               on delete cascade )
```

Here, employee name is a key to the table manager, meaning that each employee has at most one manager. The foreign-key clause requires that every manager also be an employee. Explain exactly what happens when a tuple in the relation manager is deleted.



Answer: The tuples of all employees of the manager, at all levels, get deleted as well! This happens in a series of steps. The initial deletion will trigger deletion of all the tuples corresponding to direct employees of the manager. These deletions will in turn cause deletions of second level employee tuples, and so on, till all direct and indirect employee tuples are deleted.

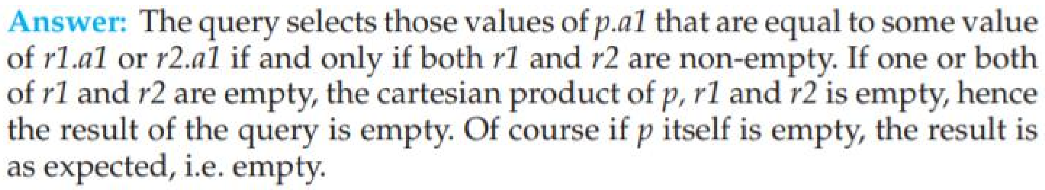
iii) Consider the SQL query:



```
select p.a1
from p, r1, r2
where p.a1 = r1.a1 or p.a1 = r2.a1
```

Under what conditions does the preceding query select values of p.a1 that are either in r1 or in r2?

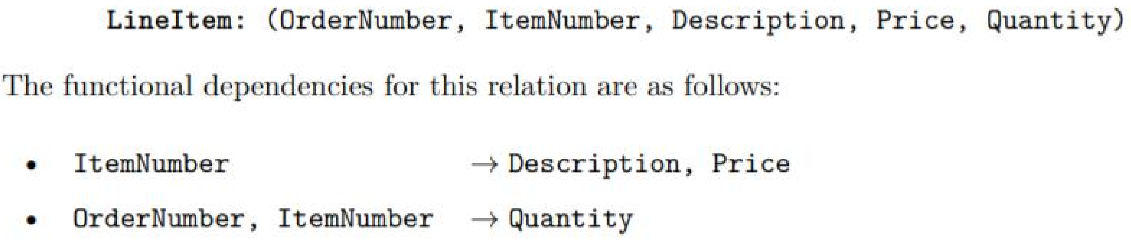
Examine carefully the cases where one of r1 or r2 may be empty. (1/2 partial correct; 1:Complete correct)



Answer: The query selects those values of *p.a1* that are equal to some value of *r1.a1* or *r2.a1* if and only if both *r1* and *r2* are non-empty. If one or both of *r1* and *r2* are empty, the cartesian product of *p*, *r1* and *r2* is empty, hence the result of the query is empty. Of course if *p* itself is empty, the result is as expected, i.e. empty.

Q4. Normalization (3+2=5 marks)

i) Consider the following relational schema: (1x3=3 marks, 1: Complete Correct)

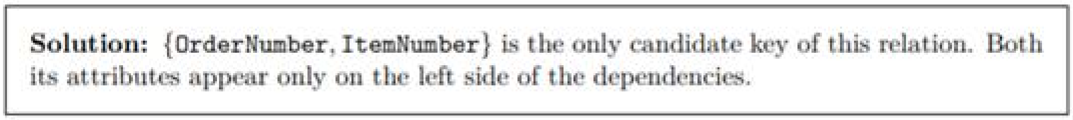


`LineItem: (OrderNumber, ItemNumber, Description, Price, Quantity)`

The functional dependencies for this relation are as follows:

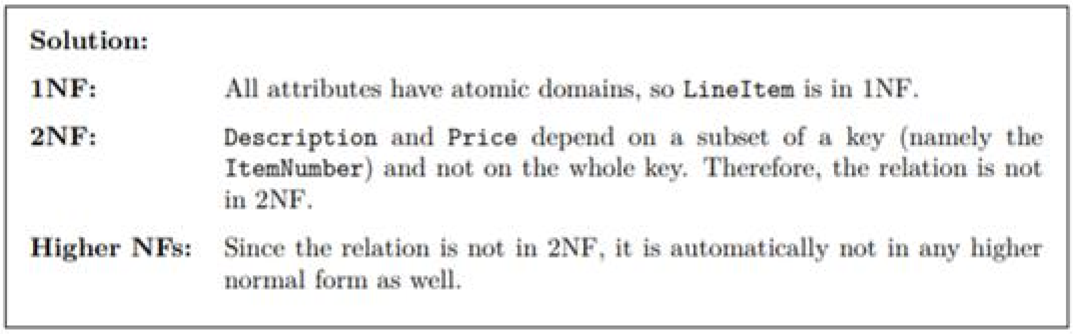
- `ItemNumber` → `Description, Price`
- `OrderNumber, ItemNumber` → `Quantity`

a) Find the candidate key(s) of the relation above.



Solution: {`OrderNumber`, `ItemNumber`} is the only candidate key of this relation. Both its attributes appear only on the left side of the dependencies.

b) What normal form is the above LineItem relation in?



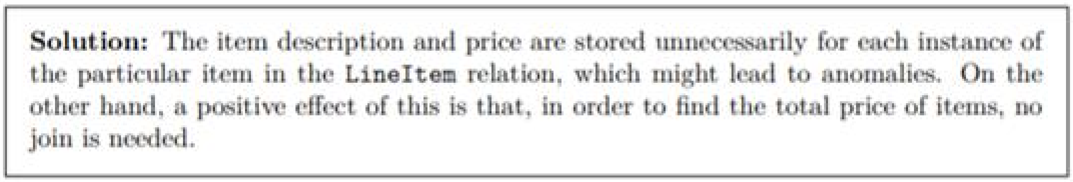
Solution:

1NF: All attributes have atomic domains, so `LineItem` is in 1NF.

2NF: `Description` and `Price` depend on a subset of a key (namely the `ItemNumber`) and not on the whole key. Therefore, the relation is not in 2NF.

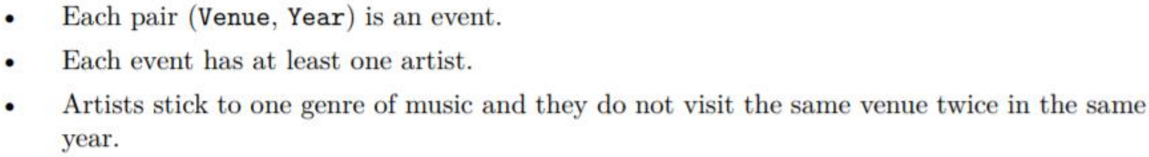
Higher NFs: Since the relation is not in 2NF, it is automatically not in any higher normal form as well.

c) What are some disadvantages of this choice of schema?

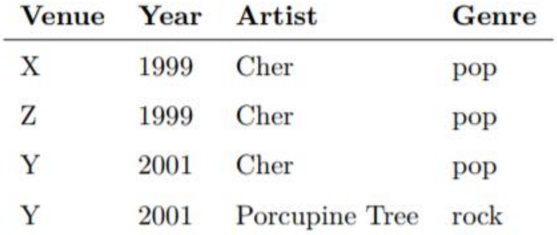


Solution: The item description and price are stored unnecessarily for each instance of the particular item in the `LineItem` relation, which might lead to anomalies. On the other hand, a positive effect of this is that, in order to find the total price of items, no join is needed.

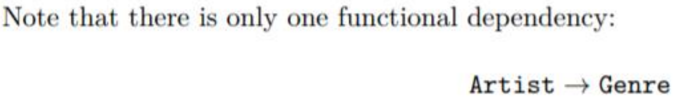
ii) Consider the following relational schema describing musical events in Switzerland. For the questions below, you can make the following assumptions: (1x2=2 marks, 1: Complete Correct)



- Each pair (`Venue`, `Year`) is an event.
- Each event has at least one artist.
- Artists stick to one genre of music and they do not visit the same venue twice in the same year.



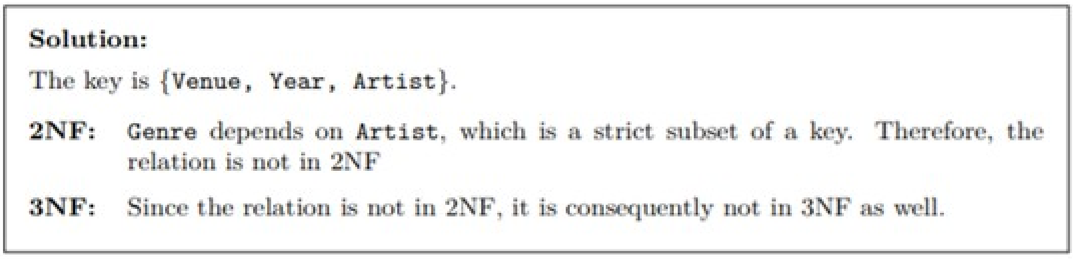
Venue	Year	Artist	Genre
X	1999	Cher	pop
Z	1999	Cher	pop
Y	2001	Cher	pop
Y	2001	Porcupine Tree	rock



Note that there is only one functional dependency:

`Artist` → `Genre`

a) Is this relation in 2NF? 3NF? Determine the keys and justify your answer.



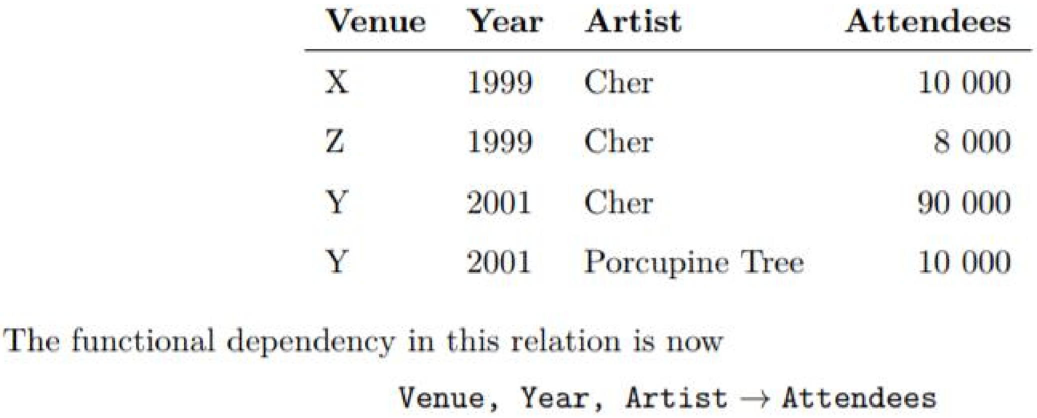
Solution:

The key is {`Venue, Year, Artist`}.

2NF: `Genre` depends on `Artist`, which is a strict subset of a key. Therefore, the relation is not in 2NF

3NF: Since the relation is not in 2NF, it is consequently not in 3NF as well.

b) We now modify the schema to include the number of attendees, but no genre:

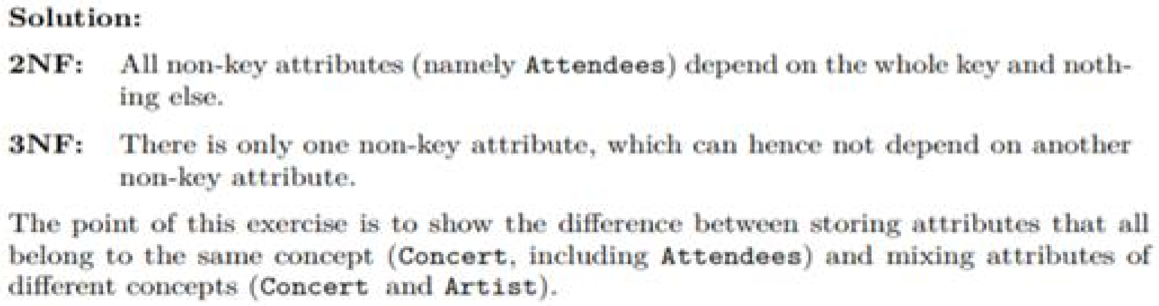


Venue	Year	Artist	Attendees
X	1999	Cher	10 000
Z	1999	Cher	8 000
Y	2001	Cher	90 000
Y	2001	Porcupine Tree	10 000

The functional dependency in this relation is now

$$\texttt{Venue, Year, Artist} \rightarrow \texttt{Attendees}$$

Is the new schema in 2NF or 3NF? Justify.



Solution:

2NF: All non-key attributes (namely `Attendees`) depend on the whole key and nothing else.

3NF: There is only one non-key attribute, which can hence not depend on another non-key attribute.

The point of this exercise is to show the difference between storing attributes that all belong to the same concept (`Concert`, including `Attendees`) and mixing attributes of different concepts (`Concert` and `Artist`).

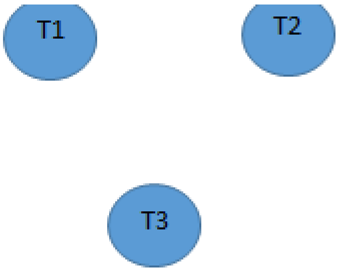
Q5. Transaction Management & Concurrency Control Techniques (5+5+5=15 marks)

i) Check whether the following schedules are conflict serializable or not (2.5x2=5 marks, 1: Diagram+1.5: Conclusion if conflict serializable or not)

a) First Schedule

T1	T2	T3
R(a)		
	R(b)	
		R(c)
		W(c)
	W(b)	
W(a)		

Yes conflict serializable since no cycle.

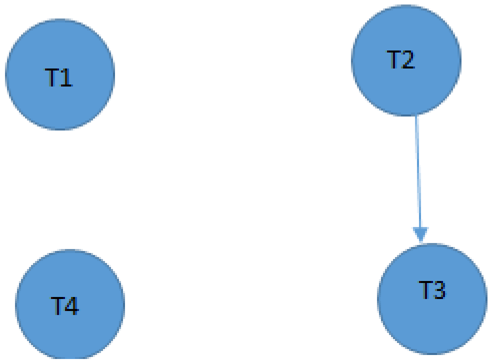


b) Second Schedule

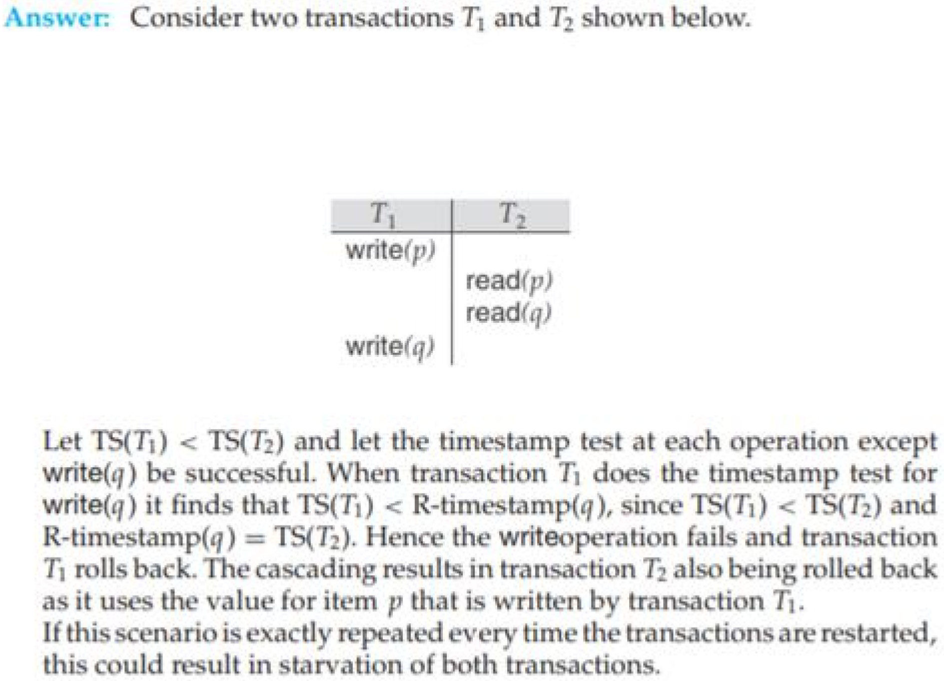
T1	T2	T3	T4
	R(x)		
		W(x)	
		Commit	
W(x)			
Commit			
	W(y)		
	R(z)		
	Commit		
			R(x)
			R(y)

			Commit

Yes conflict serializable since no cycle. Clue: After commit no conflict can happen.



ii) Consider the timestamp-ordering protocol, and two transactions, one that writes two data items p and q, and another that reads the same two data items. Give a schedule whereby the timestamp test for a write operation fails and causes the first transaction to be restarted, in turn causing a cascading abort of the other transaction. Show how this could result in starvation in both transactions. (Such a situation, where two or more processes carry out actions, but are unable to complete their task because of interaction with the other processes, is called a livestock). (5 marks, 2.5 for schedule , 2.5->Justification)



Answer: Consider two transactions T_1 and T_2 shown below.

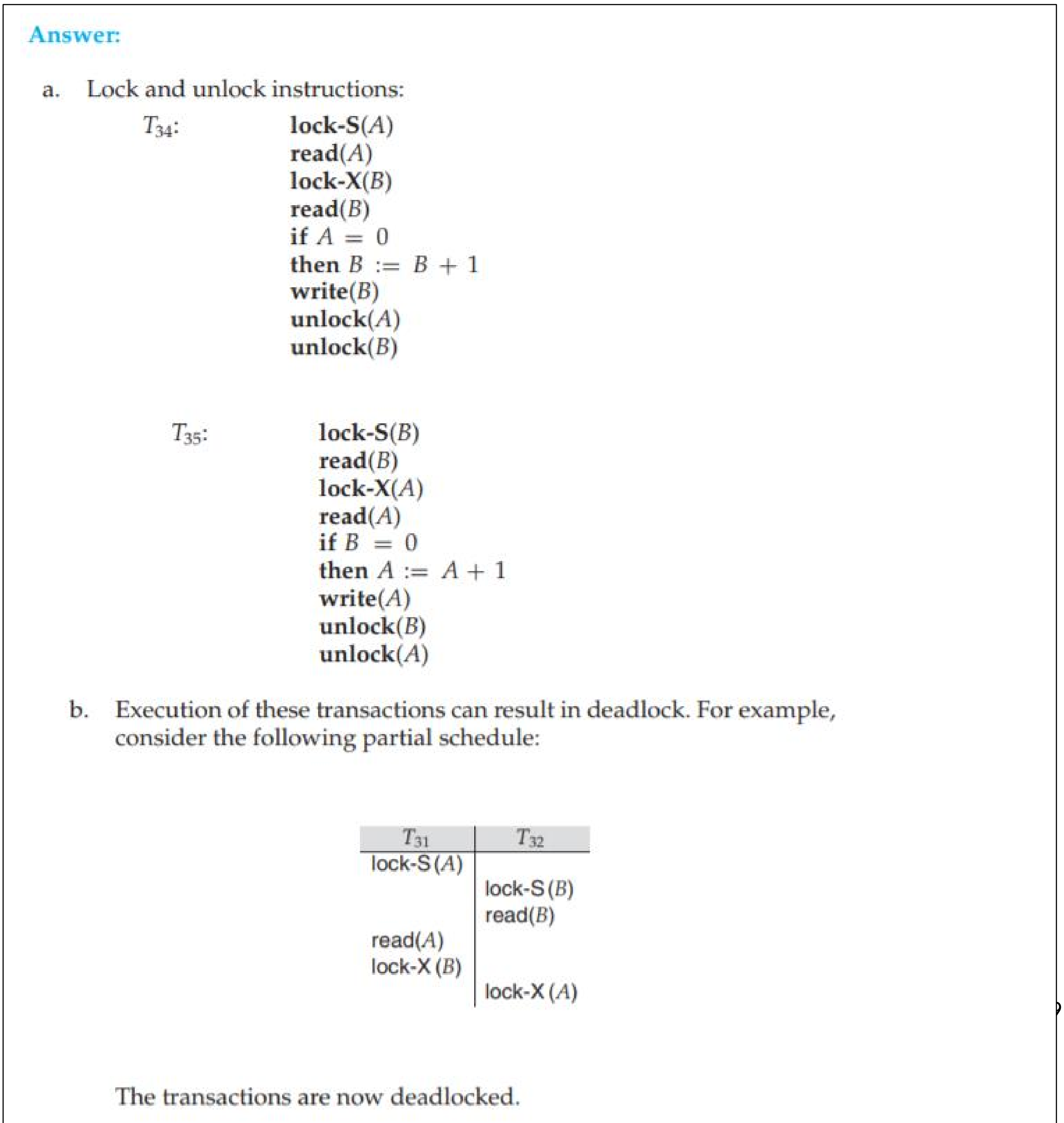
T_1	T_2
write(p)	
	read(p)
	read(q)
write(q)	

Let TS(T_1) < TS(T_2) and let the timestamp test at each operation except write(q) be successful. When transaction T_1 does the timestamp test for write(q) it finds that TS(T_1) < R-timestamp(q), since TS(T_1) < TS(T_2) and R-timestamp(q) = TS(T_2). Hence the writeoperation fails and transaction T_1 rolls back. The cascading results in transaction T_2 also being rolled back as it uses the value for item p that is written by transaction T_1.
If this scenario is exactly repeated every time the transactions are restarted, this could result in starvation of both transactions.

iii) Consider the following two transactions:

```
T31 : read(A);
      read(B);
      if A=0 then B:=B+1;
      write(B).
T32 : read(B);
      read(A);
      if B=0 then A:=A+1;
      write(A).
```

Add lock and unlock instructions to transactions T_{31} and T_{32}, so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock? Justify your answer how it may result in a deadlock. (5 marks, 2.5 for schedule , 2.5->Justification for deadlock)



Answer:

a. Lock and unlock instructions:

```
T34:    lock-S(A)
        read(A)
        lock-X(B)
        read(B)
        if A = 0
        then B := B + 1
        write(B)
        unlock(A)
        unlock(B)

T35:    lock-S(B)
        read(B)
        lock-X(A)
        read(A)
        if B = 0
        then A := A + 1
        write(A)
        unlock(B)
        unlock(A)
```

b. Execution of these transactions can result in deadlock. For example, consider the following partial schedule:

T_{31}	T_{32}
lock-S(A)	
	lock-S(B)
	read(B)
read(A)	
lock-X(B)	
	lock-X(A)

The transactions are now deadlocked.

ROUGH WORK