Question 1

1) A transaction is an execution of a user program, and is seen by the DBMS as a series or list of actions. The actions that can be executed by a transaction include reads and writes of database objects, whereas actions in an ordinary program could involve user input, access to network devices, user interface drawing, etc.

2) a) Atomicity is the guarantee that series of database operations in
an atomic transaction will either all occur (a successful operation), or none will occur
b) Consistency in database systems refers to the requirement that any given database transaction must change affected data only in allowed ways
c) Isolation is the database-level property that controls how and when changes are made, and if they become visible to each other, users, and systems
d) Durability in databases is the property that ensures transactions are saved permanently and do not accidentally disappear or get erased, even during a database crash.
e) a schedule (or history) of a system is an abstract model to describe execution of transactions running in the system
f) A dirty read occurs when a transaction is allowed to read data from a row that has been modified by another running transaction and not yet committed.
g) a blind write occurs when a transaction writes a value without reading it
h) A schedule is called serializable whenever executing the transactions sequentially, in some order, could have left the database in the same state as the actual schedule

3) a) If the transaction T2 performed W(Y) before T1 performed R(Y), and then T2 aborted, the value read by T1 would be invalid and the abort would be cascaded to T1 (i.e. T1 would also have to abort.).
b) Strict 2PL would require T2 to obtain an exclusive lock on Y before writing to it. This lock would have to be held until T2 committed or aborted; this would block T1 from reading Y until T2 was finished, but there would be no interference.

Question 2

Write-read conflict (Reading uncommited data or Dirty Read) -
T1 R(X)
T1 R(Y)

T1 W(X)

T2 R(X) - Dirty Read

Read-Write Conflict (Unrepeatable reads) -

T1 R(X)

T1 R(Y)

T2 R(X) T2 R(Y)

T1 W(X)

...

Now T2 will get unrepeatable read.

Write-Write conflict (Overwriitng uncommited data) -

T1 R(X)

T1 R(Y)

T2 R(X)

T1 W(X) - Step 4 T2 R(Y)

T2 W(X) - Write-Write conflict

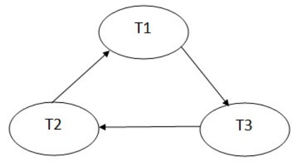
Write-read conflict - T2 will not get a Shared lock on X, untill T1 commits.

Read-Write Conflict - T1 will not get Exclusive lock on X untill T2 commits.

Write-Write conflict - T1 will not get Exclusive lock on X in Step 4, unitll T2 commits.

Precedence Graph or Serialization Graph is used commonly to test Conflict Serializability of a schedule.

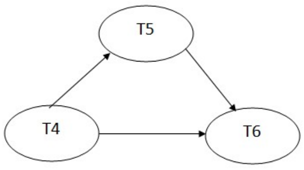
Schedule S1:



$$\begin{tikzcd}
 & T1 \arrow[dr] & \\
T2 \arrow[ur] & & T3 \arrow[ll]
\end{tikzcd}$$

The precedence graph for schedule S1 contains a cycle that's why Schedule S1 is non-serializable.

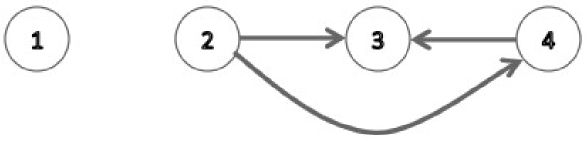
Schedule S2



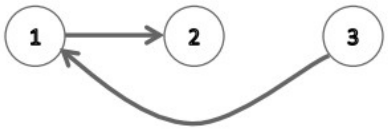
The precedence graph for schedule S2 contains no cycle that's why ScheduleS2 is serializable.

Question 3

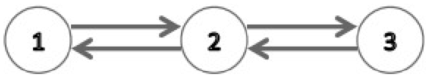
a) No. Snapshot isolation ignores read/write conflicts. Consider
T1: (R_A, W_B) and T2: (R_B, W_A).
These execute in the following order: R_{A0} (T1), R_{B0} (T2), W_{B1} (T1), W_{A2} (T2)
Under snapshot isolation, this order is acceptable and both transactions can commit.
However, this is neither serial orders T1, T2 (where T2 would R_{B1}) or T2, T1 (where T1 would R_{A2}).



b) **Yes this is serializable and one possible order is T1, T2, T4, T3. T1 can happen anywhere, but the ordering T2, T4, T3 is the only possible ordering of those three transactions.**

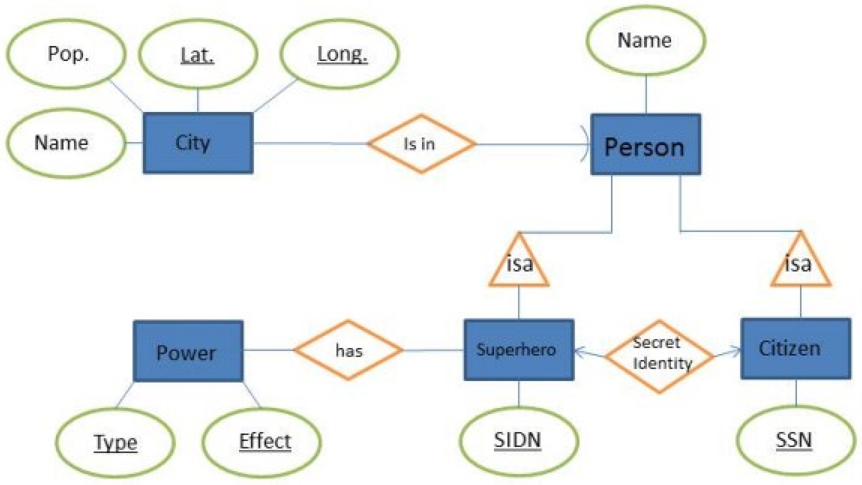


c) **This is also serializable. The only possible order is T3, T1, T2.**



d) **Not serializable. There is a cycle between T1 and T2 on A, and a cycle between T2 and T3 on B.**

Question 4



```
Create table City(name varchar(50), population INTEGER, latitude INTEGER, longitude INTEGEF
            primary key(latitude, longitude)
);

Create table Person(name varchar(50), latitude INTEGER, longitude INTEGER, foreign key
            (latitude, longitude) references City(latitude, longitude)
);

Create table Superhero(SIDN INTEGER primary key, foreign key (latitude, longitude) reference
            Person(latitude, longitude)
);

Create table Citizen(SSN INTEGER primary key, foreign key (latitude, longitude) references
           Person(latitude, longitude)

Create table Power(type varchar(20), effect varchar(50), SIDN INTEGER references
           Superhero(SIDN), primary key(type, effect)
);

Create table secretIdentity(
SIDN INTEGER references Superhero(SIDN), SSN INTEGER references Citizen(SSN), primary
           key(SIDN, SSN)
);
```

Question 5

- The data definition language is important in representing information because it is used to describe external and logical schemas.
- The data manipulation language is used to access and update data; it is not important for representing the data. (Of course, the data manipulation language must be aware of how data is represented, and reflects this in the constructs that it supports.)
- The buffer manager is not very important for representation because it brings arbitrary disk pages into main memory, independent of any data representation.
- The data model is fundamental to representing information. The data model determines what data representation mechanisms are supported by the DBMS.
- The data definition language is just the specific set of language constructs available to describe an actual application’s data in terms of the data model.

B) A database is an integrated collection of data, usually so large that it has to be stored on secondary storage devices such as disks or tapes. This data can be maintained as a collection of operating system files, or stored in a DBMS (database management system). The advantages of using a DBMS are:
Data independence and efficient access. Database application programs are independent of the details of data representation and storage. The conceptual and external schemas provide independence from physical storage decisions and logical design decisions respectively. In addition, a DBMS provides efficient storage and retrieval mechanisms, including support for very large files, index structures and query optimization.
Reduced application development time. Since the DBMS provides several important functions required by applications, such as concurrency control and crash recovery, high level query facilities, etc., only application-specific code needs to be written. Even this is facilitated by suites of application development tools available from vendors for many database management systems.

Data integrity and security. The view mechanism and the authorization facilities of a DBMS provide a powerful access control mechanism. Further, updates to the data that violate the semantics of the data can be detected and rejected by the DBMS if users specify the appropriate integrity constraints.
Data administration. By providing a common umbrella for a large collection of data that is shared by several users, a DBMS facilitates maintenance and data administration tasks. A good DBA can effectively shield end-users from the chores of fine-tuning the data representation, periodic back-ups etc.
Concurrent access and crash recovery. A DBMS supports the notion of a transaction, which is conceptually a single user's sequential program. Users can write transactions as if their programs were running in isolation against the database. The DBMS executes the actions of transactions in an interleaved fashion to obtain good performance, but schedules them in such a way as to ensure that conflicting operations are not permitted to proceed concurrently. Further, the DBMS maintains a continuous log of the changes to the data, and if there is a system crash, it can restore the database to a transaction-consistent state. That is, the actions of incomplete transactions are undone, so that the database state reflects only the actions of completed transactions. Thus, if each complete transaction, executing alone, maintains the consistency criteria, then the database state after recovery from a crash is consistent.

C) Logical data independence means that users are shielded from changes in the logical structure of the data, while physical data independence insulates users from changes in the physical storage of the data.. We could choose to store Students tuples in a heap file, with a clustered index on the sname field. Alternatively, we could choose to store it with an index on the gpa field, or to create indexes on both fields, or to store it as a file sorted by gpa. These storage alternatives are not visible to users, except in terms of improved performance, since they simply see a relation as a set of tuples. This is what is meant by physical data independence.
QUESTION 6
a) The following SQL statements create the corresponding relations.

```
CREATE TABLE A ( a1 CHAR(10),
a2 CHAR(10),
b1 CHAR(10),
c1 CHAR(10),
PRIMARY KEY (a1),
UNIQUE (b1),
FOREIGN KEY (b1) REFERENCES B,
FOREIGN KEY (c1) REFERENCES C )
CREATE TABLE B ( b1 CHAR(10),
b2 CHAR(10),
PRIMARY KEY (b1) )
CREATE TABLE C ( b1 CHAR(10),
c2 CHAR(10),
PRIMARY KEY (c1) )
```

The first SQL statement folds the relationship R into table A and thereby guarantees the participation constraint.

The following SQL statements create the corresponding relations.

```
CREATE TABLE A ( a1 CHAR(10),
a2 CHAR(10),
b1 CHAR(10),
c1 CHAR(10),
PRIMARY KEY (a1),
```

```
UNIQUE (b1),
FOREIGN KEY (b1) REFERENCES B,
FOREIGN KEY (c1) REFERENCES C )
CREATE TABLE B ( b1 CHAR(10),
b2 CHAR(10),
PRIMARY KEY (b1) )
CREATE TABLE C ( b1 CHAR(10),
c2 CHAR(10),
PRIMARY KEY (c1) )
```

The first SQL statement folds the relationship R into table A and thereby guarantees the participation constraint.

b)

I)The system will do the following:

```
SELECT S.name FROM ( SELECT E.ename AS name, E.age, E.salary FROM Emp E
WHERE E.age > 50 ) AS S WHERE S.salary > 100000
```

II)The following view on Emp can be updated automatically by updating Emp:

```
CREATE VIEW SeniorEmp (eid, name, age, salary) AS SELECT E.eid, E.ename, E.age,
E.salary FROM Emp E WHERE E.age > 50
```

II)The following view cannot be updated automatically because it is not clear which employee records will be affected by a given update:
CREATE VIEW AvgSalaryByAge (age, avgSalary) AS SELECT E.eid, AVG (E.salary) FROM Emp E GROUP BY E.age
c)
I)The system will do the following:
SELECT S.name FROM (SELECT E.ename AS name, E.age, E.salary FROM Emp E WHERE E.age > 50) AS S WHERE S.salary > 100000
II)The following view on Emp can be updated automatically by updating Emp:
CREATE VIEW SeniorEmp (eid, name, age, salary) AS SELECT E.eid, E.ename, E.age, E.salary FROM Emp E WHERE E.age > 50
III)The following view cannot be updated automatically because it is not clear which employee records will be affected by a given update:
CREATE VIEW AvgSalaryByAge (age, avgSalary) AS SELECT E.eid, AVG (E.salary) FROM Emp E GROUP BY E.age

QUESTION 7

a)The DBA is responsible for:

- Designing the logical and physical schemas, as well as widely-used portions of the external schema.
- Security and authorization.
- Data availability and recovery from failures.
- Database tuning: The DBA is responsible for evolving the database, in particular the conceptual and physical schemas, to ensure adequate performance as user requirements change.

A DBA needs to understand query optimization even if s/he is not interested in running his or her own queries because some of these responsibilities (database design and tuning) are related to query optimization. Unless the DBA understands the performance needs of widely used queries, and how the DBMS will optimize and execute these queries, good design and tuning decisions cannot be made.
b)
I)A transaction is any one execution of a user program in a DBMS. This is the basic unit of change in a DBMS.
II)A DBMS is typically shared among many users. Transactions from these users can be interleaved to improve the execution time of users' queries. By interleaving queries,

users do not have to wait for other user's transactions to complete fully before their own transaction begins. Without interleaving, if user A begins a transaction that will take 10 seconds to complete, and user B wants to begin a transaction, user B would have to wait an additional 10 seconds for user A's transaction to complete before the database would begin processing user B's request.

III)A user must guarantee that his or her transaction does not corrupt data or insert nonsense in the database. For example, in a banking database, a user must guarantee that a cash withdraw transaction accurately models the amount a person removes from his or her account. A database application would be worthless if a person removed 20 dollars from an ATM but the transaction set their balance to zero! A DBMS must guarantee that transactions are executed fully and independently of other transactions. An essential property of a DBMS is that a transaction should execute atomically, or as if it is the only transaction running. Also, transactions will either complete fully, or will be aborted and the database returned to it's initial state. This ensures that the database remains consistent.

IV)Strict two-phase locking uses shared and exclusive locks to protect data. A transaction must hold all the required locks before executing, and does not release any lock until the transaction has completely finished.

V)The WAL property affects the logging strategy in a DBMS. The WAL, WriteAhead Log, property states that each write action must be recorded in the log (on disk) before the corresponding change is reflected in the database itself. This protects the database from system crashes that happen during a transaction's execution. By recording the change in a log before the change is truly made, the database knows to undo the changes to recover from a system crash. Otherwise, if the system crashes just after making the change in the database but before the database logs the change, then the database would not be able to detect his change during crash recover

QUESTION 8

a)

In the answers below RA refers to Relational Algebra, TRC refers to Tuple Relational Calculus and DRC refers to Domain Relational Calculus.

I)RA

π sname(π sid((π pid σ (color = 'red') Parts) X Catalog) X Suppliers)

- TRC

  {T | ∃T 1 ∈ Suppliers(∃X ∈ Parts(X.color = 'red' ∧ ∃Y ∈ Catalog (Y.pid = X.pid ∧ Y.sid = T 1.sid)) ∧ T.sname = T 1.sname)}

- DRC
  { | <X, Y, Z> ∈ Suppliers ∧ ∃P, Q, R(<P, Q, R> ∈ Parts ∧ (R = 'red') ∧ ∃I, J, K(<I, J, K> ∈ Catalog ∧ J = P ∧ I = X))}
- SQL
  SELECT S.sname FROM Suppliers S, Parts P, Catalog C WHERE P.color='red' AND C.pid=P.pid AND C.sid=S.sid

II)RA

π sid(π pid(σ (color='red' v color='green') Parts) X catalog)

- TRC
  {T | ∃T 1 ∈ Catalog(∃X ∈ P arts((X.color = 'red ∨ X.color = 'green) ∧X.pid = T 1.pid) ∧ T.sid = T 1.sid)}
- DRC
  { X | X, Y, Z ∈ Catalog ∧ ∃A, B, C(A, B, C ∈ P arts ∧(C = red ∨ C = green) ∧ A = Y)}
- SQL
  SELECT C.sid FROM Catalog C, Parts P WHERE (P.color = 'red' OR P.color = 'green') AND P.pid = C.pid

III)RA

ρ(R1, πsid((πpidσcolor= red P arts) Catalog)) ρ(R2, πsidσaddress= 221P ackerStreet Suppliers) R1 ∪ R2

- TRC
  {T | ∃T 1 ∈ Catalog(∃X ∈ P arts(X.color = 'red ∧ X.pid = T 1.pid) ∧T.sid = T 1.sid) ∨∃T 2 ∈ Suppliers(T 2.address = 221P ackerStreet ∧ T.sid = T 2.sid)}
- DRC
  { X | X, Y, Z ∈ Catalog ∧ ∃A, B, C(A, B, C ∈ P arts ∧C = red ∧ A = Y) ∨∃P, Q(X, P, Q ∈ Suppliers ∧ Q = 221P ackerStreet)}
- SQL
  SELECT S.sid FROM Suppliers S WHERE S.address = '221 Packer street' OR S.sid IN (SELECT C.sid FROM Parts P, Catalog C WHERE P.color='red' AND P.pid = C.pid)

IV)RA

ρ(R1, πsid((πpidσcolor= red P arts) Catalog)) ρ(R2, πsid((πpidσcolor= green P arts) Catalog)) R1 ∩ R2

- TRC
  {T | ∃T 1 ∈ Catalog(∃X ∈ P arts(X.color = 'red ∧ X.pid = T 1.pid) ∧∃T 2 ∈ Catalog(∃Y ∈ P arts(Y.color = green ∧ Y.pid = T 2.pid) ∧T 2.sid = T 1.sid) ∧ T.sid = T 1.sid)}

- DRC

  { X | X, Y, Z ∈ Catalog ∧ ∃A, B, C(A, B, C ∈ P arts ∧C = red ∧ A = Y) ∧∃P, Q, R(P, Q, R ∈ Catalog ∧ ∃E, F, G(E, F, G ∈ P arts ∧G = green ∧ E = Q) ∧ P = X)}
- SQL

  SELECT C.sid FROM Parts P, Catalog C WHERE P.color = 'red' AND P.pid = C.pid AND EXISTS (SELECT P2.pid FROM Parts P2, Catalog C2 WHERE P2.color = 'green' AND C2.sid = C.sid AND P2.pid = C2.pid)

V)RA

(πsid,pidCatalog)/(πpidP arts)

- TRC

  {T | ∃T 1 ∈ Catalog(∀X ∈ P arts(∃T 2 ∈ Catalog (T 2.pid = X.pid ∧ T 2.sid = T 1.sid)) ∧ T.sid = T 1.sid)}
- DRC

  { X | X, Y, Z ∈ Catalog ∧ ∀ A, B, C ∈ P arts (∃ P, Q, R ∈ Catalog(Q = A ∧ P = X))}
- SQL

  SELECT C.sid FROM Catalog C WHERE NOT EXISTS (SELECT P.pid FROM Parts P WHERE NOT EXISTS (SELECT C1.sid FROM Catalog C1 WHERE C1.sid = C.sid AND C1.pid = P.pid))

VI)RA

(πsid,pidCatalog)/(πpidσcolor= red P arts)

- TRC

  {T | ∃T 1 ∈ Catalog(∀X ∈ P arts(X.color = 'red ∨∃T 2 ∈ Catalog(T 2.pid = X.pid ∧ T 2.sid = T 1.sid)) ∧T.sid = T 1.sid)}
- DRC

  { X | X, Y, Z ∈ Catalog ∧ ∀ A, B, C ∈ P arts (C = 'red ∨ ∃ P, Q, R ∈ Catalog(Q = A ∧ P = X))}
- SQL

  SELECT C.sid FROM Catalog C WHERE NOT EXISTS (SELECT P.pid FROM Parts P WHERE P.color = 'red' AND (NOT EXISTS (SELECT C1.sid FROM Catalog C1 WHERE C1.sid = C.sid AND C1.pid = P.pid)))

VII)RA

(πsid,pidCatalog)/(πpidσcolor= red ∨color= green P arts) TRC {T | ∃T 1 ∈ Catalog(∀ X ∈ P arts((X.color = 'red ∧X.color = 'green) ∨ ∃T 2 ∈ Catalog (T 2.pid = X.pid ∧ T 2.sid = T 1.sid)) ∧ T.sid = T 1.sid)}

- DRC
  { X | X, Y, Z ∈ Catalog ∧ ∀ A, B, C ∈ P arts ((C = ′red ∧ C = ′green) ∨ ∃ P, Q, R ∈ Catalog (Q = A ∧ P = X))}
- SQL
  SELECT C.sid FROM Catalog C WHERE NOT EXISTS (SELECT P.pid FROM Parts P WHERE (P.color = 'red' OR P.color = 'green') AND (NOT EXISTS (SELECT C1.sid FROM Catalog C1 WHERE C1.sid = C.sid AND C1.pid = P.pid)))

VIII)RA ρ(R1,((πsid,pidCatalog)/(πpidσcolor= red P arts))) ρ(R2,((πsid,pidCatalog)/(π pidσcolor= green P arts))) R1 ∪ R2

b) TRC

{T | ∃T 1 ∈ Catalog((∀X ∈ P arts (X.color = ′red ∨ ∃Y ∈ Catalog(Y.pid = X.pid ∧ Y.sid = T 1.sid)) ∨∀Z ∈ P arts(Z.color = ′green ∨ ∃P ∈ Catalog (P.pid = Z.pid ∧ P.sid = T 1.sid))) ∧ T.sid = T 1.sid)}

- DRC
  { X | X, Y, Z ∈ Catalog ∧ (∀ A, B, C ∈ P arts (C = ′red ∨ ∃ P, Q, R ∈ Catalog(Q = A ∧ P = X)) ∨∀ U, V, W ∈ P arts(W = ′green ∨ M,N,L ∈ Catalog (N = U ∧ M = X)))}
- SQL
  SELECT C.sid FROM Catalog C WHERE (NOT EXISTS (SELECT P.pid FROM Parts P WHERE P.color = 'red' AND (NOT EXISTS (SELECT C1.sid FROM Catalog C1 WHERE C1.sid = C.sid AND C1.pid = P.pid)))) OR (NOT EXISTS (SELECT P1.pid FROM Parts P1 WHERE P1.color = 'green' AND (NOT EXISTS (SELECT C2.sid FROM Catalog C2 WHERE C2.sid = C.sid AND C2.pid = P1.pid))))

IX)RA ρ(R1, Catalog) ρ(R2, Catalog) πR1.sid,R2.sid(σR1.pid=R2.pid∧R1.sid =R2.sid∧ R1.cost>R2.cost(R1 × R2))

- TRC
  {T | ∃T 1 ∈ Catalog(∃T 2 ∈ Catalog (T 2.pid = T 1.pid ∧ T 2.sid = T 1.sid ∧T 2.cost < T 1.cost ∧ T.sid2 = T 2.sid) ∧T.sid1 = T 1.sid)}
- DRC { X, P | X, Y, Z ∈ Catalog ∧ ∃P, Q, R (P, Q, R ∈ Catalog ∧ Q = Y ∧ P = X ∧ R<Z)}
- SQL
  SELECT C1.sid, C2.sid FROM Catalog C1, Catalog C2 WHERE C1.pid = C2.pid AND C1.sid = C2.sid AND C1.cost > C2.cost

X)RA ρ(R1, Catalog) ρ(R2, Catalog) πR1.pidσR1.pid=R2.pid∧R1.sid =R2.sid(R1 × R2)

c) TRC {T | ∃T 1 ∈ Catalog(∃T 2 ∈ Catalog (T 2.pid = T 1.pid ∧ T 2.sid = T 1.sid) ∧T.pid = T 1.pid)}

- DRC { X | X, Y, Z ∈ Catalog ∧ ∃A, B, C (A, B, C ∈ Catalog ∧ B = Y ∧ A = X)}
- SQL

  SELECT C.pid FROM Catalog C WHERE EXISTS (SELECT C1.sid FROM Catalog C1 WHERE C1.pid = C.pid AND C1.sid = C.sid)

XI)RA

ρ(R1, πsidσsname= Y osemiteSham Suppliers) ρ(R2, R1 Catalog) ρ(R3, R2) ρ(R4(1 → sid, 2 → pid, 3 → cost), σR3.cost<R2.cost(R3 × R2)) πpid(R2 − πsid,pid,costR4)

- TRC

  {T | ∃T 1 ∈ Catalog(∃X ∈ Suppliers (X.sname = Y osemiteSham ∧ X.sid = T 1.sid) ∧ ¬(∃S ∈ Suppliers (S.sname = Y osemiteSham ∧ ∃Z ∈ Catalog (Z.sid = S.sid ∧ Z.cost > T 1.cost))) ∧ T.pid = T 1.pid)
- DRC

  { Y | X, Y, Z ∈ Catalog ∧ ∃A, B, C (A, B, C ∈ Suppliers ∧ C = Y osemiteSham ∧ A = X) ∧¬(∃P, Q, R(P, Q, R ∈ Suppliers ∧ R = Y osemiteSham ∧∃I, J, K(I, J, K ∈ Catalog(I = P ∧ K>Z))))}
- SQL

  SELECT C.pid FROM Catalog C, Suppliers S WHERE S.sname = ‘Yosemite Sham’ AND C.sid = S.sid AND C.cost ≥ ALL (Select C2.cost FROM Catalog C2, Suppliers S2 WHERE S2.sname = ‘Yosemite Sham’ AND C2.sid = S2.sid)
-

  QUESTION 9

a) A spatial database is a database that is optimized for storing and querying data that represents objects defined in a geometric space. Most spatial databases allow the representation of simple geometric objects such as points, lines and polygons.With support for different geometry types, the PostGIS spatial database allows querying and managing information about locations and mapping. Other database examples include SQL Server (where geometry is just another data type, like char and int) and Microsoft Access (known as a personal geodatabase in ArcGIS). Spatial data refers to the shape, size and location of the feature. Non- spatial data refers to other attributes associated with the feature such as name, length, area, volume, population, soil type, etc ..

b)

1.RA

πeid(σaname=‘Boeing (Aircraf t Certif ied))

- TRC
  {C.eid | C ∈ Certif ied ∧ ∃A ∈ Aircraf t(A.aid = C.aid ∧ A.aname = ‘Boeing)}
- DRC
  { Ceid | Ceid, Caid ∈ Certif ied ∧ ∃Aid, AN, AR(Aid, AN, AR ∈ Aircraf t ∧Aid = Caid ∧ AN = ‘Boeing)}
- SQL
  SELECT C.eid FROM Aircraft A, Certified C WHERE A.aid = C.aid AND A.aname = ‘Boeing’

2.RA

πename(σaname=‘Boeing (Aircraf t Certif ied Employees))

- TRC {E.ename | E ∈ Employees ∧ ∃C ∈ Certif ied (∃A ∈ Aircraf t(A.aid = C.aid ∧ A.aname = ‘Boeing ∧ E.eid = C.eid))}
- DRC
  { EN | Eid, EN, ES ∈ Employees∧ ∃Ceid, Caid(Ceid, Caid ∈ Certif ied∧ ∃Aid, AN, AR(Aid, AN, AR ∈ Aircraf t∧ Aid = Caid ∧ AN = ‘Boeing ∧ Eid = Ceid)}
- SQL
  SELECT E.ename FROM Aircraft A, Certified C, Employees E WHERE A.aid = C.aid AND A.aname = ‘Boeing’ AND E.eid = C.eid

3.RA

ρ(BonnT oM adrid, σf rom=‘Bonn ∧to=‘Madrid (Flights)) πaid(σ cruisingrange>distance(Aircraf t × BonnT oM adrid))

- TRC
  {A.aid | A ∈ Aircraf t ∧ ∃F ∈ Flights (F.from = ‘Bonn ∧ F.to = ‘M adrid ∧ A.cruisingrange > F.distance)}
- DRC
  {Aid | Aid, AN, AR ∈ Aircraf t∧ (∃F N, F F, F T, FDi, FDe, F A(F N, F F, F T, FDi, FDe, F A ∈ Flights∧ F F = ‘Bonn ∧ F T = ‘M adrid ∧ F Di < AR))}
- SQL
  SELECT A.aid FROM Aircraft A, Flights F WHERE F.from = ‘Bonn’ AND F.to = ‘Madrid’ AND A.cruisingrange > F.distance

4.RA

πf lno(σdistance<cruisingrange∧salary>100,000(Flights Aircraf t Certif ied Employees)))

- TRC {F.flno | F ∈ Flights ∧ ∃A ∈ Aircraf t∃C ∈ Certif ied ∃E ∈ Employees(A.cruisingrange > F.distance ∧ E.salary > 100, 000∧ A.aid = C. 40 Chapter 4

5.RA ρ(R1, πeid(σcruisingrange>3000(Aircraf t   Certif ied))) πename(Employees   (R1 − πeid(σaname='Boeing   (Aircraf t   Certif ied))))

- TRC
  {E.ename | E ∈ Employees ∧ ∃C ∈ Certif ied(∃A ∈ Aircraf t (A.aid = C.aid ∧ E.eid = C.eid ∧ A.cruisingrange > 3000))∧ ¬(∃C2 ∈ Certif ied(∃A2 ∈ Aircraf t(A2.aname = 'Boeing   ∧ C2.aid = A2.aid ∧ C2.eid = E.eid)))}
- DRC
  { EN | Eid, EN, ES   ∈ Employees∧ ∃Ceid, Caid(Ceid, Caid   ∈ Certif ied∧ ∃Aid, AN, AR(Aid, AN, AR   ∈ Aircraf t∧ Aid = Caid ∧ Eid = Ceid ∧ AR > 3000))∧ ¬(∃ Aid2, AN2, AR2(Aid2, AN2, AR2   ∈ Aircraf t∧ ∃Ceid2, Caid2(Ceid2, Caid2   ∈ Certif ied ∧Aid2 = Caid2 ∧ Eid = Ceid2 ∧ AN2='Boeing   )))}
- SQL
  SELECT E.ename FROM Certified C, Employees E, Aircraft A WHERE A.aid = C.aid AND E.eid = C.eid AND A.cruisingrange > 3000 AND E.eid NOT IN (SELECT C2.eid FROM Certified C2, Aircraft A2 WHERE C2.aid = A2.aid AND A2.aname = 'Boeing')

c) Relational completeness means that a query language can express every query that can be written in relational algebra.Yes it can

QUESTION 10

a)

The statements can be interpreted as:

1. Find the Supplier names of the suppliers who supply a red part that costs less than 100 dollars.
2. This Relational Algebra statement does not return anything because of the sequence of projection operators. Once the sid is projected, it is the only field in the set. Therefore, projecting on sname will not return anything.
3. Find the Supplier names of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.
4. . Find the Supplier ids of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.

b)

I)

- User defined data types
- User defined indexes
- User defined query optimization

II) User defined query optimization

c)An unsafe query is a query in relational calculus that has an infinite number of results. An example of such a query is:
{S | ¬(S ∈ Sailors)}
The query is for all things that are not sailors which of course is everything else. Clearly there is an infinite number of answers, and this query is unsafe. It is important to disallow unsafe queries because we want to be able to get back to users with a list of all the answers to a query after a finite amount of time.