Database Management System
Assignment-4 Solution

1. What are the roles of Assertions and Triggers in SQL? Consider following bank database
   Branch-schema = (*branch-name, branch-city, assets*)
   Loan-schema = (*loan-number, branch-name, amount*)
   Write an assertion for the bank database to ensure that the Assets value for the Perryridge branch is equal to the sum of all the amounts lent by the Perryridge branch.

Ans: For 1st part see class note. 2nd part answer is

```
create assertion perry check
   (not exists (select *
      from branch
      where branch-name = 'Perryridge' and
         assets <> (select sum (amount)
            from loan
            where branch-name = 'Perryridge')))
```

Note: Here assertion name is arbitrary. I have chosen the name 'perry'.

2. What is Normalization and why it is done? Give an example of a relation schema R and a set of dependencies such that R is not in 2NF and normalize it into 2NF.

Ans: Refer class note.

3. What do you mean by Integrity Constraints? Explain its types.

Ans: Refer class note.

4. Give an example of a relation schema R and a set of dependencies which illustrates the importance of 4NF.

Ans: Refer class note.

5. State and explain with example about functional dependency, transitive dependency and multivalued dependency.

Ans: Refer class note.

6. Give an example of a relation schema R and a set of dependencies such that R is not in 3NF and normalize it into 3NF.

Ans: Refer class note.

7. How security can be granted using view explain. What are two advantages of encrypting data stored in the database?

Ans: For 1st part refer class note. For 2nd part:

The two advantages of encrypting data stored in the database are:

a. Encrypted data allows authorized users to access data without worrying about other users or the system administrator gaining any information.

b. Encryption of data may simplify or even strengthen other authorization mechanisms. For example, distribution of the cryptographic key amongst only trusted users is both, a simple way to control read access, and an added layer of security above that offered by views.

8. What is Log-Based Recovery? Compare the deferred- and immediate-modification versions of the log-

based recovery scheme in terms of ease of implementation and overhead cost.

Ans: Refer class note.

9. What is Log-Based Recovery? Explain Deferred Database modification and Immediate Database modification with an illustration.

Ans: Refer class note.

10. Explain the basic steps in query processing.

11. Explain the distinction between the terms *serial schedule* and serializable *schedule*. 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.

Ans: A schedule in which all the instructions belonging to one single transaction appear together is called a *serial schedule*. A *serializable schedule* has a weaker restriction that it should be *equivalent* to some serial schedule. There are two definitions of schedule equivalence – conflict equivalence and view equivalence.

For 2nd part:

Lock and unlock instructions:

*T*31: **lock-S**(*A*)
read(*A*)
lock-X(*B*)
read(*B*)
if $A = 0$
then $B := B + 1$
write(*B*)
unlock(*A*)
unlock(*B*)

*T*32: **lock-S**(*B*)
read(*B*)
lock-X(*A*)
read(*A*)
if $B = 0$
then $A: = A + 1$
write(*A*)
unlock(*B*)
unlock(*A*)

12. What do you mean schedule and serializability? What are view serialization schedules?
13. What do you mean by a schedule? When schedule is called serializable? What is conflict serialization schedules?
14. What do you mean by concurrency control? Describe Two phase locking Protocol.
15. Construct a B+-tree for the following set of key values:
(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the tree is initially empty and values are added in ascending order. Construct B+-trees for the case where the number of pointers that will fit in one node is **Six**. Also show the form of the tree after insertion of **9**.

16. Construct a B+-tree for the following set of key values:

(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the tree is initially empty and values are added in ascending order. Construct B+-trees for the case where the number of pointers that will fit in one node is **Four**. Also show the form of the tree after deletion of 23.

17. Explain about sequential file access and hash index with examples.
18. Define Referential-integrity constraints. Consider a database that includes the following relations:

salaried-worker (*name, office, phone, salary*)
hourly-worker (*name, hourly-wage*)
address (*name, street, city*)

Suppose that we wish to require that every name that appears in *address* appear in either *salaried-worker* or *hourly-worker*, but not necessarily in both.

a. Propose a syntax for expressing such constraints.
b. Discuss the actions that the system must take to enforce a constraint of this form.

Ans: a. First create table salaried-worker and hourly-worker with fields mentioned above with **name** as primary key in both tables. As part of the **create table** expression for *address* we include **foreign key** (*name*) **references** *salaried-worker* or *hourly-worker.*

i.e. *create table address (name char(20), street varchar(30), city char(20),*
foreign key(name) references salaried-worker(name) or hourly-worker(name))

b. To enforce this constraint, whenever a tuple is inserted into the *address* relation, a lookup on the *name* value must be made on the *salaried-worker* relation and (if that lookup failed) on the *hourly-worker* relation (or vice-versa).

19. What is importance of trigger? Write an SQL trigger to carry out the following action: On **delete** of an account, for each owner of the account, check if the owner has any remaining accounts, and if she does not, delete her from the *depositor* relation.

Ans: For 1st part refer class note. For 2nd part

create trigger *check-delete-trigger* **after delete on** *account*
referencing old row as *orow*
for each row
delete from *depositor*
where *depositor.customer-name* **not in**
(**select** *customer-name* **from** *depositor*
where *account-number <> orow.account-number*)
end

Note: The questions whose answer I have not provided are available on class note. So please refer class note for the answer of those questions.

Prepared By: Amit K. Shrivastava
NCIT, Balkumari