Distributed Databases

Tutorial Questions

1. What is a Distributed Database system? A single database host multiple DBMS
2. What advantages does a Distributed Database have over a Centralized Database? Local autonomy and performance
3. Why do some users require data from multiple sites in a Distributed Database? Collaborative Work
4. A heterogeneous distributed database is which of the following?
   A. The DBMS is identical at each site and data is not distributed across all sites.
   B. The DBMS is identical at each site and data is distributed across all sites.
   C. A different DBMS is used at each site and data is not distributed across all site.
   D. A different DBMS is used at each site and data is distributed across all sites. D
5. What is Horizontal Fragmentation within a DDBMS? Spread different rows from one table over anumber of different sites
6. Some columns of a table / relation are located at different sites is. This is an example of?
   A. Data Replication
   B. Horizontal Fragmentation
   C. Vertical Fragmentation C
   D. Horizontal and Vertical Fragmentation
7. A DDBMS often has replication. What is an advantage of replication? High Reliability
8. Which of the following is a disadvantage of replication?
   A. Reduced network traffic D
   B. If the database fails at one site, a copy can be located at another site.
   C. Each site must have the same storage capacity.
   D. Each transaction may proceed without coordination across the network.

9. A DDBMS has two tables
   - TableA
     - has a row size of roughly 100 bytes
     - Is stored in Melbourne
     - has around 500,000 rows
   - TableB
     - has a row size of roughly 100 bytes
     - Is stored in the New York
     - has around 10,000 rows
   - Imagine a query is written that requires data from about 10% of rows from both tables.

   a. "When a query that involves tables at more than one site, it will be performed at the local site".
      What does this mean? So the query will be processed at the local site
   b. Which will generally take longer?
      a) Running the query in Melbourne A
      b) Running the query in New York
   c. How can views and synonyms be used to get the queries to roughly execute in the same time, regardless of which city the query is run from?
      Create a view of NY in Melbourne

Database Triggers

Tutorial Questions

10. How is a trigger similar to a stored procedure? it is code stored in DBMS as an object
11. How is a trigger different to a stored procedure? We don thave to call
12. When writing a trigger, what are the :new and :old variables used for? the value before and after run
13. Are :old and :new variables available with statement-level triggers? No
14. Are :old and :new variables available with row-level triggers? Yes
15. Does an ON UPDATE trigger have access to :old and :new variables? Yes
16. Does an ON DELETE trigger have access to :old and :new variables? No, only :old
17. Does an ON INSERT trigger have access to :old and :new variables? No, only :new
18. What is a mutating table error? database server can not resolve the logical problem
19. How can a mutating table error be avoided? use trigger
20. A business rule says that an employee cannot earn more than his/her manager.
    Assume that each employee row has a foreign key which refers to a manger row.
    a. Can this business rule be implemented using a fixed format constraint? No
    b. Can this business rule be implemented using a trigger? Yes
21. Triggers should be created sparingly. Why? Stored in memory
22. Should you use a trigger to check the uniqueness of a primary key? Yes
23. Consider a trigger which archives deleted rows from a table into a separate archive table.
    a. Is using a trigger to achieve this using needless computation power? NO
    b. What is another way of implementing this feature without using triggers? through application logic level
    c. What are the arguments in favour of this solution? Flexibility, easier debugging
    d. What are the arguments against this solution?
       Complexity, reduced performance

Lab Tasks

Your first trigger is the auditing example from the lecture. Create the following tables first:

```
DROP TABLE CUST CASCADE CONSTRAINTS;
CREATE TABLE CUST
        (CUSTID    NUMBER PRIMARY KEY,
        CUSTNAME  VARCHAR2(20),
        BALANCE   NUMBER(8,2) );

DROP TABLE AUDITING CASCADE CONSTRAINTS;
CREATE TABLE AUDITING
        (AUDITID      number primary key,
         TABLENAME    varchar2 (15),
         OP_TYPE      varchar2 (10),
        OLD_BALANCE       NUMBER(8,2),
        new_balance       NUMBER(8,2),
        ACCESSED_BY   VARCHAR2(10),
        ACCESSED_TIME date );

DROP SEQUENCE  AUDITID_SEQ;
CREATE  SEQUENCE  AUDITID_SEQ;
```

Copy/paste the create trigger code into SQL Developer and then execute it.

```
CREATE OR REPLACE TRIGGER LOGOPERATION
BEFORE INSERT OR UPDATE OR DELETE ON CUST
FOR EACH ROW
DECLARE
        vOP_TYPE  AUDITING.OP_TYPE%TYPE;
BEGIN
        IF INSERTING THEN
                vOP_TYPE := 'INSERT';
        ELSIF UPDATING THEN
                vOP_TYPE := 'UPDATE';
        ELSE                    -- MUST BE DELETING
                vOP_TYPE := 'DELETE';
        END IF;
        INSERT INTO AUDITING   (AUDITID,
                TABLENAME,
                OP_TYPE,
                OLD_BALANCE,
                NEW_BALANCE,
                ACCESSED_BY,
                ACCESSED_TIME)
        VALUES
                (AUDITID_SEQ.NEXTVAL,
                'CUST',
                vOP_TYPE,
                :OLD.BALANCE,
                :NEW.BALANCE,
                USER,
                SYSDATE);
END;
```

Fire the trigger by performing some changes on the CUST table:

```
INSERT INTO CUST (CUSTID, CUSTNAME, BALANCE) VALUES (11001, 'John Smith', 50000);
UPDATE CUST SET BALANCE = BALANCE + 155 WHERE CUSTID = 11001;
INSERT INTO CUST (CUSTID, CUSTNAME, BALANCE) VALUES (11002, 'Peter Black', 65000);
DELETE FROM CUST where CUSTID = 11002;
INSERT INTO CUST (CUSTID, CUSTNAME, BALANCE) VALUES (11003, 'Barbara Whitmore', 75000);
UPDATE CUST SET BALANCE = BALANCE + 5000 WHERE CUSTID = 11003;
INSERT INTO CUST (CUSTID, CUSTNAME, BALANCE) VALUES (11004, 'Nguyen Tran', 77777);
UPDATE CUST SET BALANCE = BALANCE + 233 WHERE CUSTID = 11004;
UPDATE CUST SET BALANCE = BALANCE + 1;
DELETE FROM CUST;
```

Observe the effects by SELECTing all rows from the CUST and AUDITING tables.
The date in the auditing table looks more interesting if you format it to show the time:

```
SELECT AUDITID, TABLENAME, OP_TYPE,
    OLD_BALANCE, NEW_BALANCE, ACCESSED_BY ,
    TO_CHAR(ACCESSED_TIME, 'DD-MON-YYYY HH24:MI:SS')
FROM AUDITING
```

Task 2

Write your own trigger for the following scenario:

Enterprises are often interested in historical information and like to archive rather than lose information. So, when a customer is deleted, we require a trigger which copies each customer row into another table as it is deleted from the original table.

Start by creating an archiving table CUST_ARCHIVE that is an exact copy of the CUST table. Copy the table structure (was discussed in earlier lectures).
Remember to add the 'WHERE 0 = 1' clause to the statement if you want to stop data rows from being copied to the new table.

Write a trigger archiveCustomer that copies each CUST row into CUST_ARCHIVE before it is deleted from CUSTOMER.

- You should use BEFORE DELETE ON CUST
- The trigger should process FOR EACH ROW deleted
- The trigger requires an INSERT statement to add a row to the CUST_ARCHIVE table.
- The data VALUES need to be the :OLD values from the CUST row. (e.g. :old.custid…)

Remember that there are no :new values in the operation as a DELETE only references :old values.

Now test the Trigger.

Try to delete a customer that exists.
Try to delete a customer that doesn't exist.

Use statement such as:

```
SELECT * FROM CUST;
   and
SELECT * FROM CUST_ARCHIVE;
```

to observe the effects of your actions.