Distributed Databases

Tutorial Questions

1. **What is a Distributed Database system?**
   It is a single physically distributed database system that spans across many geographic locations but is logically centralized from the perspective of the users and application that access the data.
2. **What advantages does a Distributed Database have over a Centralized Database?**
   Some of the noteworthy advantages of Distributed Database are scalability, reliability and performance.
3. **Why do some users require data from multiple sites in a Distributed Database?**
   It may be because of the partition and distribution of distributed databased across multiple nodes in the network, as data is stored across multiple geographic location in distributed database. This will promote scalability, reliability and performance.
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.
5. **What is Horizontal Fragmentation within a DDBMS?**
   It is a strategy used in DDBMS to divide a table into multiple fragments or subsets of rows, each of which is assigned to a different site or node in the network.
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
   D. Horizontal and Vertical Fragmentation
7. **A DDBMS often has replication. What is an advantage of replication?**
   The advantage of replication is to promote data availability and reliability, as it is stored on multiple nodes or sites in the network, meaning that if 1 nodes fails the data can still be accessed from other sites. It also improve performance as load-balancing between each node can be performed.
8. **Which of the following is a disadvantage of replication?**
   A. Reduced network traffic
   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 thelocal site".**
   What does this mean?
   When a query is executed that requires data from tables located on multiple sites in DDBMS, the query will be executed at the "master" site, meaning the site in which the query is initiated, and then it will fetch data from other required file to compose a comprehensive result for the user.
   In this case, if a query requires data from both Table A and B, and the query is initiated at the Melbourne site, then the query will be executed at the Melbourne site but it will also retrieve data from the New York site.

b. **Which will generally take longer?**
   a) Running the query in Melbourne
   b) Running the query in New York
   It may take additional information to determine which one will take longer such as connectivity, hardware and software configuration. However, the site that stores the larger table or has a larger number of rows may take longer to process the query. Table A may need to retrieve 50000 rows and Table B may need to retrieve 1000 rows, so Table A may take longer.

c. **How can views and synonyms be used to get the queries to roughly execute in thesame time, regardless of which city the query is run from?**

By using views and synonyms, queries can be optimized and customized to address the specific requirements of the query. In this example, we can create views for Table A and Table B that are customized for the specific queries that will be run or create synonyms for the views that provide a standardized interface for accessing the data.

Database Triggers

Tutorial Questions

10. **How is a trigger similar to a stored procedure?**
    They are both database object that can be used to execute a set of instructions or actions.
11. **How is a trigger different to a stored procedure?**
    A trigger is a special type of stored procedure that is automatically executed in response to specific database event, such as an update or insert operation.
12. **When writing a trigger, what are the :new and :old variables used for?**
    It refers to the new and old values of a row that is being affected by insert, update or delete operation.
13. **Are :old and :new variables available with statement-level triggers?**
    As statement-level trigger is executed once for each row, the :old and :new variables are not available in these trigger to reference the old and new value of the rows.
14. **Are :old and :new variables available with row-level triggers?**
    :old and :new variables are available with row-level triggers to reference the old and new values of the row.
15. **Does an ON UPDATE trigger have access to :old and :new variables?**
    When an ON UPDATE trigger is defined on a table, it is executed in response to an update operation on a table, therefore it have access to the variables.
16. **Does an ON DELETE trigger have access to :old and :new variables?**
    When an ON DELETE trigger is defined on a table, it is executed in response to an delete operation on a table, therefore it have access to the :old variables as there is no :new variable.
17. **Does an ON INSERT trigger have access to :old and :new variables?**
    When an ON INSERT trigger is defined on a table, it is executed in response to an INSERT operation on a table, therefore it have access to the :new variables as there is no :old variable.
18. **What is a mutating table error?**
    Mutating table error occurs when a trigger or a SQL statement attempts to read or modify a table that is currently being modified by another transaction or statement.
19. **How can a mutating table error be avoided?**
    Compound triggers can be used to avoid mutating table error.
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, as fixed format constraint, such as a check constraint, can only reference values within the same row. In this example, Employee and Manager may not be in the same table.
    b. **Can this business rule be implemented using a trigger?**
       Yes, whenever an INSERT statement into Employee table is performed, a trigger can be implemented to check the :new and :old variable.
21. **Triggers should be created sparingly. Why?**
    Trigger can have a significant impact on data consistency, especially if they interact with other triggers or database objects, therefore, they should be created sparingly.
22. **Should you use a trigger to check the uniqueness of a primary key?**
    As trigger can have a significant impact on data consistency, which is the very idea behind uniqueness of primary key, using them to check the uniqueness of primary key would

somehow be counter intuitive.

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?**
    It is, as trigger will happen each time a row is deleted from a table and it will be computationally expensive. Hence, the delete operation and overall database performance will be affected.
    b. **What is another way of implementing this feature without using triggers?**
    We can use stored procedure to periodically archive deleted rows from the table into a separate archive table.
    c. **What are the arguments in favour of this solution?**
    As trigger is performed each time a row is deleted, the following may be argued in favor of the solution:
    + Real-time archiving.
    + Simplified design.
    d. **What are the arguments against this solution?**

    As trigger is performed each time a row is deleted, the following may be argued against the solution:

    + Performance impact

    + Data consistency

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.

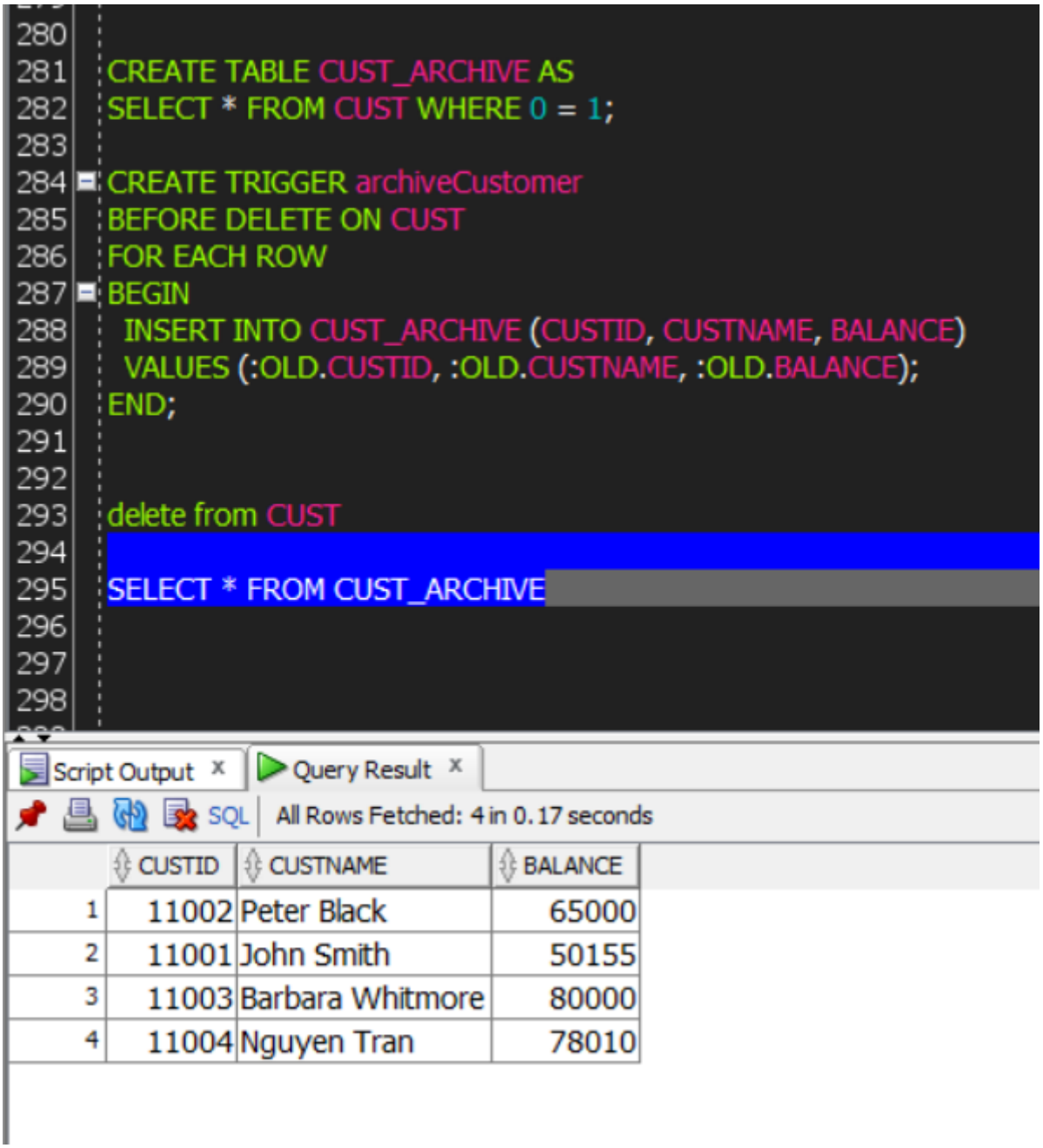
Now test the Trigger.

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.



```
280
281  CREATE TABLE CUST_ARCHIVE AS
282  SELECT * FROM CUST WHERE 0 = 1;
283
284  CREATE TRIGGER archiveCustomer
285  BEFORE DELETE ON CUST
286  FOR EACH ROW
287  BEGIN
288    INSERT INTO CUST_ARCHIVE (CUSTID, CUSTNAME, BALANCE)
289    VALUES (:OLD.CUSTID, :OLD.CUSTNAME, :OLD.BALANCE);
290  END;
291
292
293  delete from CUST
294
295  SELECT * FROM CUST_ARCHIVE
296
297
298
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.17 seconds

	CUSTID	CUSTNAME	BALANCE
1	11002	Peter Black	65000
2	11001	John Smith	50155
3	11003	Barbara Whitmore	80000
4	11004	Nguyen Tran	78010