AUTOESTUDIO 5

PL/ SQL Básico

ALUMNOS:

Ignacio Andrés Castillo Rendon

Anderson Fabian Garcia Nieto

Laboratorio-Modelos de bases de datos
2024-2

DOCENTE:
MARIA IRMA DIAZ ROZO

Bogotá D.C-Colombia

Investigación

Transacciones
A. Transacciones
1. ¿Cómo se define el comienzo y fin de una transacción en ORACLE?

RTA: El inicio se define con un BEGIN TRANSACTION, mientras que el final se marca con un COMMIT o ROLLBACK. El commit actualiza los datos que se modificaron/insertaron en la tabla de manera definitiva, mientras que el rollback devuelve todo tipo de cambio hecho.

2. ¿Cuáles son los diferentes tipos de aislamiento que soporta ORACLE? Para cada uno de ellos detalle, ¿cómo maneja los bloqueos? ¿Qué problemas resuelve?

RTA: Los niveles de aislamiento que ofrece SQL es: READ COMMITTED (Una transacción no podrá ver los cambios de otras conexiones hasta que no hayan sido confirmados o descartados), SERIALIZABLE (No se permitirá a otras transacciones la inserción, actualización o borrado de datos utilizados por nuestra transacción), READ ONLY (Solo permite operaciones de lectura).

3. ¿Cuál es el tipo de aislamiento por defecto en ORACLE?

RTA: El tipo de aislamiento por defecto es READ COMMITTED, es decir, Solo se pueden leer datos que han sido confirmados por otras transacciones, cada consulta ve los datos más recientes que han sido confirmados, proporciona un buen balance entre consistencia y concurrencia y no requiere configuración adicional pues viene por defecto.

FUENTE:

- campusMVP. (s. f.). Fundamentos de SQL: Transacciones - campusMVP.es. campusMVP.es. https://www.campusmvp.es/recursos/post/Fundamentos-de-SQL-Transacciones.aspx

B. Vistas
1. ¿Cuáles son los mecanismos para la creación y borrado de vistas en ORACLE?
Una vista se crea a partir de una tabla, o varias para esto es necesario de una consulta sobre esta.
Los mecanismos para la creación de la vistas en ORACLE son:

```
CREATE VIEW nombre_de_la_vista AS
SELECT columna1, columna2
FROM tabla
WHERE condiciones;
```

Los mecanismos para la eliminación de las vistas en ORACLE son:

```
DROP VIEW nombre_de_la_vista;
```

July2024. (2024, September 4). *CREATE VIEW*. Oracle Help Center. https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/CREATE-VIEW.html

July2024. (2024, September 4). *DROP VIEW*. Oracle Help Center. https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/DROP-VIEW.html

2. ¿Cuáles son las restricciones de las vistas en ORACLE?

Las vistas en bases de datos, aunque ofrecen una representación virtual de los datos, tienen limitaciones a la hora de realizar modificaciones directas. Estas restricciones se derivan de la complejidad de las consultas que definen a las vistas y de las relaciones entre las vistas y las tablas subyacentes.

Vistas de Solo Lectura:

Funciones y Expresiones: Vistas que incluyen cálculos, agrupaciones o subconsultas suelen ser de solo lectura.
Uniones y Subconsultas: La combinación de múltiples tablas o el uso de subconsultas en la definición de una vista restringe su modificabilidad.
Funciones de Ventana: Vistas que utilizan funciones como ROW_NUMBER() o RANK() para realizar cálculos sobre conjuntos de filas son inmutables.
Vistas Materializadas: Estas vistas, que almacenan los resultados de una consulta, tienen restricciones específicas en su actualización, dependiendo de su configuración.

Complejidad de la Vista:

Cuanto más compleja sea la definición de una vista, más difícil es garantizar la consistencia de los datos al realizar modificaciones.
Dependencia de Múltiples Tablas: Las vistas basadas en varias tablas pueden tener conflictos de actualización si se modifican datos en una tabla subyacente.
Naturaleza de las Funciones: Funciones como SUM, AVG o ROW_NUMBER() calculan valores basados en conjuntos de datos, lo que hace inviable modificarlos directamente.

C. Modularidad Paquetes

1. ¿Para qué sirve un paquete?

RTA: Los paquetes poseen varias características, entre ellas:

- Proporcionan una forma conveniente de organizar las funciones y procedimientos que tienen un propósito relacionado. El permiso para utilizar las funciones y procedimientos del paquete depende de un privilegio que se otorga a todo el paquete.
- Determinados elementos de un paquete pueden declararse públicos. Las entidades públicas son visibles y pueden ser referenciadas por otros programas que tienen el privilegio EXECUTE en el paquete
- Otros elementos de un paquete se pueden declarar privados. Las entidades privadas pueden ser referenciadas y utilizadas por funciones y procedimientos dentro del paquete, pero no por aplicaciones externas.

2. ¿Cuáles son los mecanismos para la creación, invocación, modificación y borrado de paquetes en ORACLE?

RTA: En la parte de la creación, se divide por dos partes:

- Crear la cabecera del paquete donde se definen que procedimientos, funciones, variables, cursores, etc. Disponibles para su uso posterior fuera del paquete. En esta parte solo se declaran los objetos, no se implementa el código.
- Crear el cuerpo del paquete, donde se definen los bloques de código de las funciones y procedimientos definidos en la cabecera del paquete.

Para la parte de la invocación se debe hacer llamado a un procedimiento para acceder al paquete.

A la hora de la modificación, se maneja la siguiente estructura:

```
-- Modificar la especificación
CREATE OR REPLACE PACKAGE nombre_paquete AS
   -- Nueva especificación
END nombre_paquete;
/

-- Modificar el cuerpo
CREATE OR REPLACE PACKAGE BODY nombre_paquete AS
   -- Nuevo cuerpo
END nombre_paquete;
/
```

Por último, para la eliminación del paquete, se hace lo siguiente:

```
-- Eliminar solo el cuerpo
DROP PACKAGE BODY nombre_paquete;

-- Eliminar el paquete completo (especificación y cuerpo)
DROP PACKAGE nombre_paquete;
```

FUENTES:

- DB2 11.1. (s. f.). https://www.ibm.com/docs/es/db2/11.1?topic=support-packages-plsql
- El Baúl del Programador. (2016, 1 enero). PL/SQL. paquetes (Packages). PL/SQL. Paquetes (Packages). https://elbauldelprogramador.com/plsql-paquetes-packages/

D. SYS_REFCURSOR

1. ¿Qué es un SYS_REFCURSOR?¿Para qué sirve?

SYS_REFCURSOR es un tipo predefinido en Oracle que permite trabajar con cursores de referencia sin necesidad de declarar un tipo personalizado para cada cursor. Simplifica la creación y el uso de cursores de referencia, ya que permite manejar dinámicamente resultados de consultas.

Sirve para:

Retornar Conjuntos de Resultados: Se utiliza frecuentemente en procedimientos y funciones para devolver conjuntos de resultados de manera flexible.

Consultas dinámicas: Es útil cuando no se conoce la estructura exacta del conjunto de resultados al momento de escribir el código PL/SQL. Puedes ejecutar consultas dinámicas y retornar el resultado a aplicaciones o interfaces de usuario externas.

Interoperabilidad con Aplicaciones Externas: Muchas aplicaciones, como clientes de Oracle o interfaces de usuario desarrolladas en otras plataformas, pueden procesar un SYS_REFCURSOR para obtener datos de un procedimiento almacenado en Oracle.

How and when to use sys_refcursor in oracle. (n.d.). Database Administrators Stack Exchange. https://dba.stackexchange.com/questions/34180/how-and-when-to-use-sys-refcursor-in-oracle

2. ¿Cómo se define, se asigna y se retorna?

Como se define:

```
CREATE OR REPLACE PROCEDURE get_data(result OUT SYS_REFCURSOR) AS
BEGIN
   -- Lógica del procedimiento
END;
```

Cómo se asigna:

```
CREATE OR REPLACE PROCEDURE get_data(result OUT SYS_REFCURSOR) AS
BEGIN
   OPEN result FOR
   SELECT column1, column2
   FROM table_name
   WHERE condition;
END;
```

Cómo se retorna:

```
CREATE OR REPLACE PROCEDURE get_musicians_by_band (
   band_id IN NUMBER,
   result OUT SYS_REFCURSOR
) AS
BEGIN
   OPEN result FOR
   SELECT m.m_name, m.born
   FROM musician m
   JOIN plays_in p ON m.m_no = p.player
   WHERE p.band_id = band_id;
END;
```

PRACTICANDO

A. OFRECIENDO SERVICIOS

1) Implemente los paquetes de componentes necesarios para ofrecer las funciones básicas y consultas del ciclo actual del sistema (CRUD).

ESPECIFICACIONES

ad(name: VARCHAR, type : VARCHAR, b_date : DATE) : NUMBER
Esta es una función que permite agregar una nueva banda, ya que inserta una banda en la base de datos, en esta inserción se agrega el nombre de la banda (name), el género (type) y la fecha de creación (b_date).

Esta función retorna un número que podría ser el ID, también un número que indica el estado de éxito de la creación o varias cosas

mo(band: NUMBER, b_date : DATE) : void
Esta función permite modificar la fecha de creación de la banda indicada, band representa el ID de la banda y b_date es la nueva fecha de formación.

La función al finalizar su ejecución no retorna nada(void), y actualiza la fecha en la base de datos.

ad_musician(band : NUMBER, musician : NUMBER) : void
Esta función permite agregar un nuevo músico a la base de datos en la banda especificada.

La función al finalizar su ejecución no retorna nada, y actualiza la relación del músico en la base de datos

del_musician(band : NUMBER, musician : NUMBER) : void
Esta función permite eliminar un músico de la banda indicada, y a sí mismo de la base de datos.

La función al finalizar su ejecución no retorna nada, y elimina la relación del músico de la base de datos

del(band : NUMBER) : void
La función permite eliminar una banda de la base de datos
La función al finalizar no retorna ningún valor, borra el registro de la banda y así mismo sus relaciones con los músicos

co() : SYSREFCURSOR
La función permite ver una consulta a través de un cursor con la lista de los músicos en la base de datos, con sus detalles, como el nombre, instrumento y los atributos relacionados

Retorna un cursor que permite iterar sobre las bandas

co_musicians() : SYSREFCURSOR
La función permite ver una consulta a través de un cursor con la lista de los músicos en la base de datos, con sus detalles, como el nombre, instrumento y los atributos relacionados.
Esta es la función que representa a la consulta "Consult the number of instruments and the number of compositions of musicians"

Retorna un cursor con la consulta pedida

co_bands() : SYSREFCURSOR
La función permite ver una consulta a través de un cursor como el nombre de las bandas y sus músicos asociados.
Esta es la función que representa a la consulta "Consult the bands that have more musicians than average"

Retorna un cursor con la consulta pedida

IMPLEMENTACIONES

1. **Implemente los paquetes de componentes necesario para ofrecer las funciones básicas y consultas del ciclo actual del sistema (CRUD).**

1) **ad(name: VARCHAR, type : VARCHAR, b_date : DATE) : NUMBER**

```
CREATE OR REPLACE FUNCTION ad(name IN VARCHAR2, type IN
VARCHAR2, b_date IN DATE)
RETURN NUMBER IS
   band_id NUMBER;
BEGIN
   INSERT INTO band (band_no, band_name, band_type, b_date, band_home,
band_contact)
   VALUES (num_sequence.NEXTVAL, name, type, b_date, NULL, NULL)
   RETURNING band_no INTO band_id;

   COMMIT;

   RETURN band_id;
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error adding band: ' || SQLERRM);
      RETURN NULL;
END;
```

2) **mo(band: NUMBER, b_date : DATE) : void**

```
CREATE OR REPLACE PROCEDURE mo(band IN NUMBER, b_date IN DATE)
IS
BEGIN
   UPDATE band
   SET b_date = b_date
   WHERE band_no = band;

   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error modifying band date: ' || SQLERRM);
END;
```

3) **ad_musician(band : NUMBER, musician : NUMBER) : void**

```
CREATE OR REPLACE PROCEDURE ad_musician(band IN NUMBER, musician
IN NUMBER) IS
BEGIN
   INSERT INTO plays_in (player, band_id)
   VALUES (musician, band);

   COMMIT;
```

```
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error adding musician to band: ' || SQLERRM);
END;
/
```

4) del_musician(band : NUMBER, musician : NUMBER) : void

```
CREATE OR REPLACE PROCEDURE del_musician(band IN NUMBER, musician
IN NUMBER) IS
BEGIN
   DELETE FROM plays_in
   WHERE band_id = band AND player = musician;

   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error deleting musician from band: ' ||
SQLERRM);
END;
```

5) del(band : NUMBER) : void

```
CREATE OR REPLACE PROCEDURE del(band IN NUMBER) IS
BEGIN
   DELETE FROM plays_in WHERE band_id = band;
   DELETE FROM band WHERE band_no = band;

   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error deleting band: ' || SQLERRM);
END;
/
```

6) co() : SYSREFCURSOR

```
CREATE OR REPLACE PROCEDURE co(bands OUT SYS_REFCURSOR) IS
BEGIN
```

```
    OPEN bands FOR
    SELECT * FROM band;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error retrieving bands: ' || SQLERRM);
END;
```

7) co_musicians() : SYSREFCURSOR

```
CREATE OR REPLACE PROCEDURE co_musicians(musicians OUT
SYS_REFCURSOR) IS
BEGIN
    OPEN musicians FOR
    SELECT * FROM musician;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error retrieving musicians: ' || SQLERRM);
END;
```

8) co_bands() : SYSREFCURSOR

```
CREATE OR REPLACE PROCEDURE co_bands(bands_musicians OUT
SYS_REFCURSOR) IS
BEGIN
    OPEN bands_musicians FOR
    SELECT b.band_name, m.m_name
    FROM band b
    JOIN plays_in p ON b.band_no = p.band_id
    JOIN musician m ON p.player = m.m_no;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error retrieving bands and musicians: ' ||
SQLERRM);
END;
```

2. Proponga un caso de prueba exitoso por subprograma. (son ocho)

```
--1. AD
BEGIN
    INSERT INTO musician (m_no, m_name, born) VALUES (1, 'John Doe',
TO_DATE('1985-05-20', 'YYYY-MM-DD'));
    COMMIT;
END;
/
```

--2. MO

```
BEGIN
   PC_BANDS.MO(2, TO_DATE('2024-01-01', 'YYYY-MM-DD'));
   DBMS_OUTPUT.PUT_LINE('Fecha de banda modificada exitosamente');
END;
/
```

--3. AD MUSICIAN

```
BEGIN

   PC_BANDS.ad_musician(
      p_m_no => 2,
      p_m_name => 'Jane Smith',
      p_born => TO_DATE('1990-07-15', 'YYYY-MM-DD'),
      p_died => NULL,
      p_born_in => 1,
      p_living_in => 2
   );

   DBMS_OUTPUT.PUT_LINE('Músico agregado exitosamente');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error al agregar el músico: ' || SQLERRM);
END;
/
```

--4. DEL_MUSICIAN

```
BEGIN
   PC_BANDS.DEL_MUSICIAN(1, 2);
   DBMS_OUTPUT.PUT_LINE('Músico eliminado de la banda exitosamente');
END;
/
```

--5. DEL

```
BEGIN
   PC_BANDS.DEL(1);
   DBMS_OUTPUT.PUT_LINE('Banda eliminada exitosamente');
END;
/
```

--6. CO

```
DECLARE
   v_count NUMBER;
BEGIN
```

```
   v_count := PC_BANDS.CO;
   DBMS_OUTPUT.PUT_LINE(v_count);
END;
/
```

--7. CO_MUSICIANS

```
DECLARE
   v_cursor SYS_REFCURSOR;
   v_musician_name VARCHAR2(100);
BEGIN
   v_cursor := PC_BANDS.CO_MUSICIANS;
   IF v_cursor  % ISOPEN THEN
      DBMS_OUTPUT.PUT_LINE('Músicos listados:');
      DBMS_OUTPUT.PUT_LINE(v_musician_name);
   ELSE
      DBMS_OUTPUT.PUT_LINE('No hay músicos disponibles.');
   END IF;
END;
/
```

--8. co_bands() : SYSREFCURSOR

```
DECLARE
   v_cursor SYS_REFCURSOR;
   v_band_name VARCHAR2(100);
   v_musician_name VARCHAR2(100);
BEGIN
   PC_BANDS.co_bands(v_cursor);

   LOOP

      FETCH v_cursor INTO v_band_name, v_musician_name;
      EXIT WHEN v_cursor%NOTFOUND;


      DBMS_OUTPUT.PUT_LINE('Banda: ' || v_band_name || ', Músico: ' ||
v_musician_name);
   END LOOP;

   CLOSE v_cursor;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error al listar bandas y músicos: ' || SQLERRM);
END;
/
```

3. Proponga tres casos en los que el subprograma no se puede ejecutar.

```
-- 1)
BEGIN
   -- Intento de insertar una banda con un número fuera del rango permitido
   PC_BANDS.AD(15, 'Rock Band', 'rock', TO_DATE('2022-01-01', 'YYYY-MM-DD'), 10,
1);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
-- 2)
BEGIN
   -- Intento de insertar una banda con un tipo no permitido
   PC_BANDS.AD(2, 'Reggae Band', 'reggae', TO_DATE('2023-01-01', 'YYYY-MM-DD'),
5, 1);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/

-- 3)
BEGIN
   -- Intento de insertar una banda con un contacto de músico inexistente
   PC_BANDS.AD(3, 'Jazz Band', 'jazz', TO_DATE('2023-05-01', 'YYYY-MM-DD'), 5,
999);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

4. Escriba las instrucciones necesarias para eliminar los paquetes.

```
DROP PROCEDURE mo;
DROP PROCEDURE ad_musician;
DROP PROCEDURE del_musician;
DROP PROCEDURE del;
DROP PROCEDURE co;
DROP PROCEDURE co_musicians;
```

```
DROP PROCEDURE co_bands;
DROP FUNCTION ad;
```