ACTIVIDAD LABORATORIO NO.1
ALMACENAMIENTO Y VALIDACIÓN DE FICHEROS XML EN ORACLE

PRESENTADO POR:
ALEJANDRO DE MENDOZA TOVAR

PRESENTADO AL PROFESOR:
ING DEIVIS EDUARD RAMIREZ MARTINEZ
PROFESOR

FUNDACIÓN UNIVERSITARIA INTERNACIONAL DE LA RIOJA
FACULTAD DE INGENIERÍA – INGENIERIA INFORMÁTICA
BOGOTÁ D.C.
3 DE NOVIEMBRE
2025

TABLA DE CONTENIDO

RESUMEN

Para el desarrollo de este laboratorio correspondiente al área de Bases de Datos y Programación en PL/SQL, se realizó la implementación y validación de un conjunto de procedimientos, consultas y triggers dentro del entorno Oracle Database, específicamente sobre el esquema HR. El objetivo principal del ejercicio fue integrar el manejo de datos en formato XML con las capacidades procedimentales de PL/SQL, mediante la creación de una tabla denominada empleados_xml y un trigger orientado a la validación de la estructura de los datos XML antes de su inserción o actualización.

Durante el proceso se implementó el trigger trg_validar_xml, encargado de verificar la presencia de elementos obligatorios dentro del XML almacenado, en particular el campo “apellido”. A través del uso de funciones nativas de Oracle como EXTRACTVALUE(), se comprobó de manera automática la integridad de los datos, generando un error personalizado en caso de omisión. Este enfoque permitió evidenciar la importancia de los mecanismos de validación interna y la capacidad de PL/SQL para gestionar estructuras semiestructuradas como XML.

A lo largo del desarrollo, se aplicaron conceptos fundamentales de arquitectura de bases de datos, modelado lógico y control de integridad, complementados con la interacción directa entre componentes del lenguaje SQL y PL/SQL. Se analizaron las ventajas del uso de triggers para garantizar consistencia, trazabilidad y calidad en los datos, así como su relación con los principios de programación reactiva en entornos de bases de datos empresariales.

El entorno Oracle SQL Developer fue esencial para la ejecución y prueba del código, permitiendo visualizar en tiempo real la compilación del trigger, la validación de inserciones y la estructura XML contenida en cada registro. Esta herramienta facilitó el seguimiento de errores y la observación de los resultados de cada operación, fortaleciendo la comprensión práctica del flujo de ejecución en PL/SQL y la integración de tecnologías de datos heterogéneas.

Con este trabajo se demuestra que la implementación de triggers sobre estructuras XML no solo responde a un requisito académico, sino que también constituye una práctica formativa de gran valor para la ingeniería informática y de datos. La actividad permitió reforzar conocimientos en validación de datos, control de errores y automatización de procesos dentro del motor Oracle, integrando teoría y práctica en un entorno profesional.

Finalmente, se concluye que el desarrollo del laboratorio HR_XML_LAB.sql representa un ejercicio clave para comprender la interacción entre estructuras de datos avanzadas (XML) y mecanismos de control interno (triggers). Este trabajo consolida competencias en diseño, análisis y ejecución de código PL/SQL, fortaleciendo la capacidad del estudiante para implementar soluciones eficientes, seguras y orientadas a la integridad de la información dentro de sistemas de bases de datos corporativos.

1. INTRODUCCIÓN

El presente laboratorio tiene como finalidad profundizar en la comprensión de los procesos de validación, control y manipulación de datos dentro de entornos de bases de datos relacionales, utilizando Oracle Database como plataforma de desarrollo. A través del lenguaje PL/SQL, se busca integrar conceptos teóricos y prácticos para la creación de estructuras que automaticen la verificación de datos en tiempo real, fortaleciendo la integridad y consistencia de la información almacenada.

En este contexto, Oracle se presenta como un entorno robusto y didáctico que permite al estudiante experimentar directamente con la creación de tablas, estructuras XML, y triggers. Estas herramientas constituyen elementos fundamentales en la administración de bases de datos modernas, ya que posibilitan la definición de reglas de negocio y la validación automática de los registros, sin necesidad de intervención manual. De esta manera, se fomenta el desarrollo de soluciones inteligentes, eficientes y orientadas a la calidad de los datos.

El desarrollo del laboratorio se centra específicamente en la implementación del proyecto HR_XML_LAB.sql, que incluye la creación de una tabla para almacenar información de empleados en formato XML, y la elaboración de un trigger denominado trg_validar_xml, encargado de verificar la existencia de campos obligatorios dentro del documento XML antes de permitir una operación de inserción o actualización. Este enfoque combina la manipulación de datos estructurados con el uso de

funciones nativas de Oracle, como EXTRACTVALUE(), reforzando la relación entre el modelo relacional y los formatos semiestructurados.

El laboratorio permite, además, comprender la relevancia del lenguaje PL/SQL como herramienta de automatización y control, destacando su capacidad para gestionar excepciones, aplicar reglas lógicas y garantizar la fiabilidad de los procesos. La ejecución paso a paso del código y su validación en Oracle SQL Developer proporcionan un entorno ideal para observar el comportamiento de los triggers, analizar errores de ejecución y comprender la lógica interna del motor de base de datos.

Asimismo, este ejercicio representa una oportunidad formativa para el fortalecimiento de competencias técnicas clave en la ingeniería informática, como el diseño de estructuras de validación, la implementación de soluciones basadas en XML y la comprensión de los mecanismos de integridad referencial. Estas habilidades resultan esenciales en entornos empresariales donde la calidad y la seguridad de la información constituyen pilares fundamentales para la toma de decisiones.

En síntesis, el laboratorio HR_XML_LAB.sql no se limita a la simple ejecución de comandos SQL o PL/SQL, sino que constituye un ejercicio integral que articula teoría, práctica y análisis crítico, permitiendo al estudiante afianzar su dominio sobre el manejo avanzado de datos y la automatización de procesos en bases de datos Oracle. Este trabajo sienta las bases para el desarrollo de sistemas más consistentes, eficientes y seguros en el ámbito profesional de la ingeniería informática.

2. DESARROLLO DEL LABORATORIO

El presente laboratorio tiene como objetivo aplicar los conceptos de integración y validación de datos XML dentro del entorno Oracle Database, utilizando PL/SQL como lenguaje de control y automatización. A través del script denominado HR_XML_LAB.sql, se diseñó un ejercicio completo que permite la extracción, estructuración, validación e inserción de datos XML provenientes del esquema de ejemplo HR (Human Resources).

El propósito fundamental del desarrollo fue construir un flujo funcional que integre distintas capacidades de Oracle, tales como:

- la generación de estructuras XML a partir de consultas SQL,
- la definición y registro de esquemas XML (XSD) dentro de la base de datos,
- la creación de tablas XMLTYPE con validación estructural,
- y la implementación de triggers en PL/SQL para verificar la integridad semántica de los datos antes de su inserción.

Para lograr este objetivo, se partió de la conexión con el usuario HR, cuyas tablas (EMPLOYEES, DEPARTMENTS, LOCATIONS, …) sirvieron como base de información para construir una jerarquía XML representando empleados y sus departamentos. Posteriormente, se implementaron rutinas PL/SQL para transformar estos datos en documentos XML bien formados, los cuales fueron validados mediante un esquema XSD registrado dentro del sistema.

El laboratorio integra de manera práctica los principios de modelado estructurado de datos, control transaccional, normalización, y validación XML, demostrando cómo Oracle puede actuar no solo como repositorio relacional, sino también como una plataforma híbrida capaz de gestionar información estructurada y semiestructurada.

De este modo, el proyecto HR_XML_LAB.sql representa una experiencia completa de integración entre SQL, PL/SQL y XML, consolidando competencias en el diseño, validación y automatización de estructuras de datos en entornos empresariales.

2.1 Conexión y preparación del entorno

En esta primera fase del laboratorio, se habilitó el entorno de trabajo en Oracle Database, utilizando el esquema HR (Human Resources) como base de datos principal. Este esquema contiene la estructura

de información típica de un sistema de gestión de recursos humanos, y sirve como referencia para la creación de consultas y la generación de documentos XML validados por esquema (XSD).

Inicialmente, el esquema HR se encontraba bloqueado, por lo cual fue necesario ejecutar el siguiente procedimiento desde una sesión con privilegios administrativos (usuario SYSTEM):

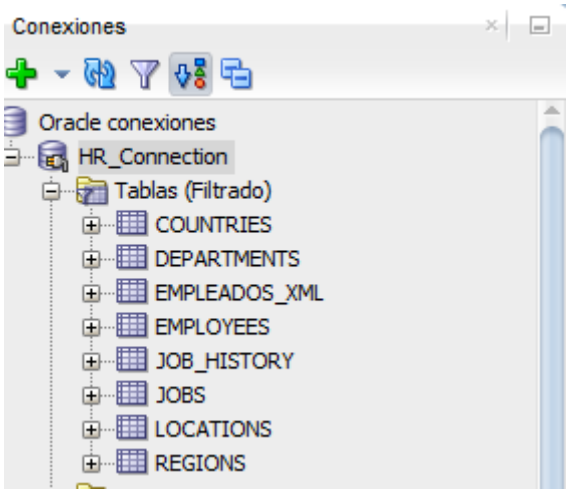
- Conexión como usuario SYSTEM: CONNECT system;
- Desbloqueo de la cuenta HR y asignación de contraseña: ALTER USER hr ACCOUNT UNLOCK IDENTIFIED BY hr;
- Concesión de privilegios de conexión y recursos: GRANT CONNECT, RESOURCE TO hr;
- Una vez realizado este proceso, se estableció la conexión al esquema con el comando: CONNECT hr/hr;

El esquema HR está conformado por varias tablas interrelacionadas que representan distintos componentes del sistema organizacional. Entre las más importantes se incluyen:

- EMPLOYEES: Contiene los datos personales y laborales de los empleados (nombres, apellidos, salario, departamento, etc.).
- DEPARTMENTS: Define los departamentos dentro de la organización.
- LOCATIONS: Especifica las ubicaciones físicas de los departamentos.
- JOBS: Lista los diferentes cargos y sus rangos salariales.
- COUNTRIES: Almacena los países asociados a las ubicaciones.
- REGIONS: Agrupa los países en regiones geográficas.
- JOB_HISTORY: Registra el historial de empleos anteriores de cada empleado.

Todas estas tablas están relacionadas mediante claves primarias y foráneas, lo que permite realizar uniones lógicas entre ellas.

Adicionalmente, durante el desarrollo se creó la tabla EMPLEADOS_XML (tipo XMLTYPE) destinada a almacenar documentos XML validados contra el esquema empleados_departamentos.xsd. Esta tabla actúa como repositorio híbrido dentro del esquema HR, permitiendo conservar información semiestructurada (XML) junto con las tablas relacionales tradicionales (EMPLOYEES, DEPARTMENTS, LOCATIONS, JOBS, COUNTRIES, REGIONS, JOB_HISTORY). La presencia de EMPLEADOS_XML se hizo visible en la arquitectura del esquema (captura incluida) y su creación se realizó con la instrucción CREATE TABLE ... XMLSCHEMA ... ELEMENT "empleados", garantizando una asociación formal entre la tabla y el XSD registrado. A continuación las imágenes respectivas de las tablas:



Para este laboratorio, se hizo énfasis en las tablas, las cuales fueron utilizadas para generar un documento XML estructurado con información jerárquica de los empleados y sus departamentos.

El propósito de esta primera parte fue garantizar que el entorno estuviera correctamente configurado, con acceso a todos los objetos del esquema HR y los privilegios necesarios para crear, registrar y manipular datos XML dentro de Oracle.

2.2 Generación y Validación del XML

En esta segunda fase del laboratorio se desarrolló el proceso de generación del archivo XML a partir de los datos almacenados en el esquema HR, con el fin de integrar información proveniente de las tablas relacionales EMPLOYEES y DEPARTMENTS en un solo documento estructurado conforme al esquema XML (XSD) definido.

2.2.1 Objetivo

Construir un documento XML llamado empleados.xml que contenga los datos de los empleados junto con la descripción del departamento al que pertenecen. Este XML se utilizará posteriormente para:

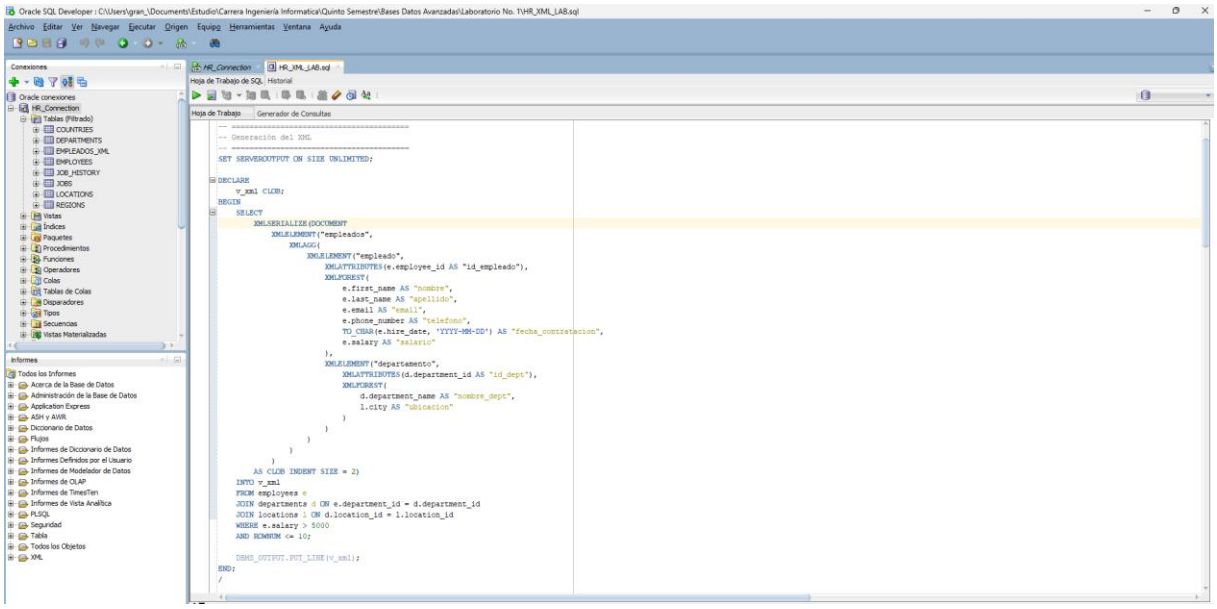
Validar su estructura con el esquema empleados_departamentos.xsd.

Cargarlo en la tabla EMPLEADOS_XML del esquema HR.

2.2.2 Generación del XML

Para la creación del XML se utilizó la función SQL/XML integrada en Oracle, específicamente las sentencias XMLELEMENT, XMLAGG y XMLFOREST, las cuales permiten transformar los resultados de una consulta SQL en un documento XML bien formado.

2.2.3 Código SQL implementado:

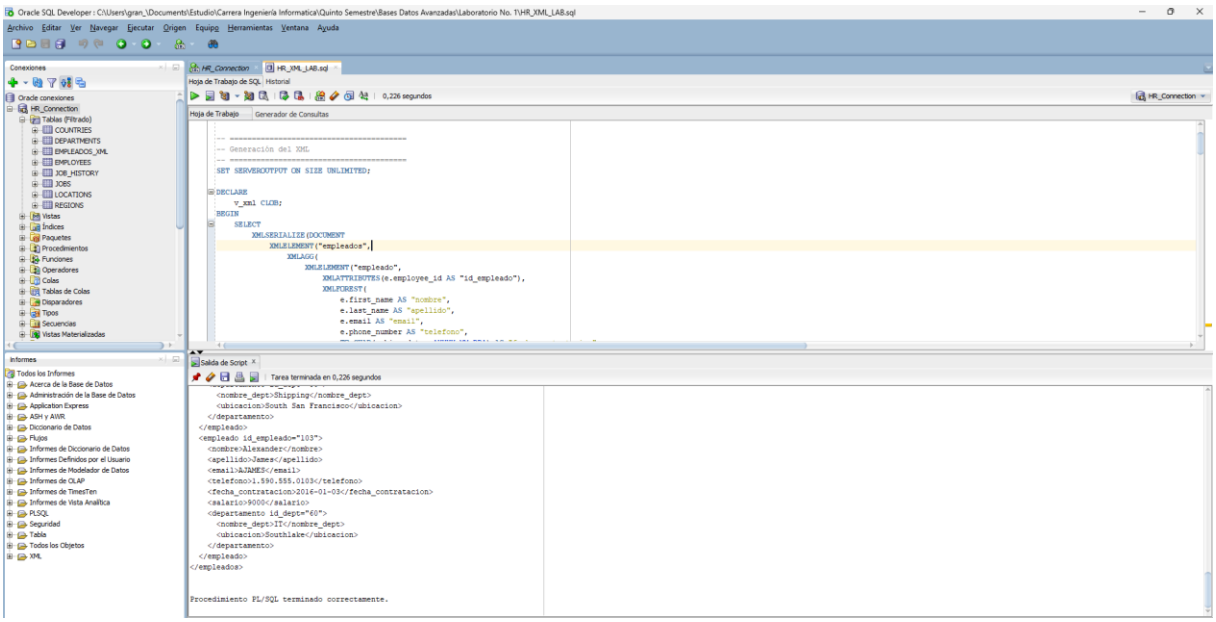


```
-- Generación del XML
SET SERVEROUTPUT ON SIZE UNLIMITED;

DECLARE
    v_xml CLOB;
BEGIN
    SELECT
        XMLSERIALIZE(DOCUMENT
            XMLELEMENT("empleados",
                XMLAGG(
                    XMLELEMENT("empleado",
                        XMLATTRIBUTES(e.employee_id AS "id_empleado"),
                        XMLFOREST(
                            e.first_name AS "nombre",
                            e.last_name AS "apellido",
                            e.email AS "email",
                            e.phone_number AS "telefono",
                            TO_CHAR(e.hire_date, 'YYYY-MM-DD') AS "fecha_contratacion",
                            e.salary AS "salario"
                        ),
                        XMLELEMENT("departamento",
                            XMLATTRIBUTES(d.department_id AS "id_dept"),
                            XMLFOREST(
                                d.department_name AS "nombre_dept",
                                l.city AS "ubicacion"
                            )
                        )
                    )
                )
            )
        AS CLOB INDENT SIZE = 2)
    INTO v_xml
    FROM employees e
    JOIN departments d ON e.department_id = d.department_id
    JOIN locations l ON d.location_id = l.location_id
    WHERE e.salary > 5000
    AND ROWNUM <= 10;

    DBMS_OUTPUT.PUT_LINE(v_xml);
END;
/
```

A continuación la salida de la consola:



2.2.4 Explicación línea por línea:

A continuación la explicación línea por línea del código:

- SET SERVEROUTPUT ON SIZE UNLIMITED;
    - Habilita la salida de DBMS_OUTPUT en SQL Developer y amplía el buffer para poder ver XML largos.
- DECLARE v_xml CLOB;
    - Se declara una variable v_xml de tipo CLOB para almacenar el XML resultante.
- SELECT XMLSERIALIZE(DOCUMENT XMLELEMENT("empleados", XMLAGG(...)) AS CLOB INDENT SIZE = 2) INTO v_xml FROM ...
    - XMLELEMENT("empleados", ...) crea el elemento raíz <empleados>.
    - XMLAGG(...) agrega/concatena todos los nodos <empleado> generados (uno por fila).
    - XMLELEMENT("empleado", XMLATTRIBUTES(...), XMLFOREST(...), XMLELEMENT("departamento", ...)) construye cada <empleado> con:
        - XMLATTRIBUTES(e.employee_id AS "id_empleado") → atributo id_empleado en la etiqueta <empleado>.
        - XMLFOREST(...) → convierte cada columna (nombre, apellido, email, telefono, fecha_contratacion, salario) en subelementos.
        - XMLELEMENT("departamento", XMLATTRIBUTES(d.department_id AS "id_dept"), XMLFOREST(...)) añade el nodo <departamento> con su id_dept como atributo y los subelementos nombre_dept y ubicacion.
    - XMLSERIALIZE(... AS CLOB INDENT SIZE = 2) convierte el resultado XML a CLOB con indentación para mejor lectura.
    - INTO v_xml almacena el CLOB en la variable local.
- FROM employees e JOIN departments d ... JOIN locations l ... WHERE e.salary > 5000 AND ROWNUM <= 10;
    - Filtra los empleados con salario mayor a 5000 y limita la salida a 10 filas (captura de muestra).
- DBMS_OUTPUT.PUT_LINE(v_xml);

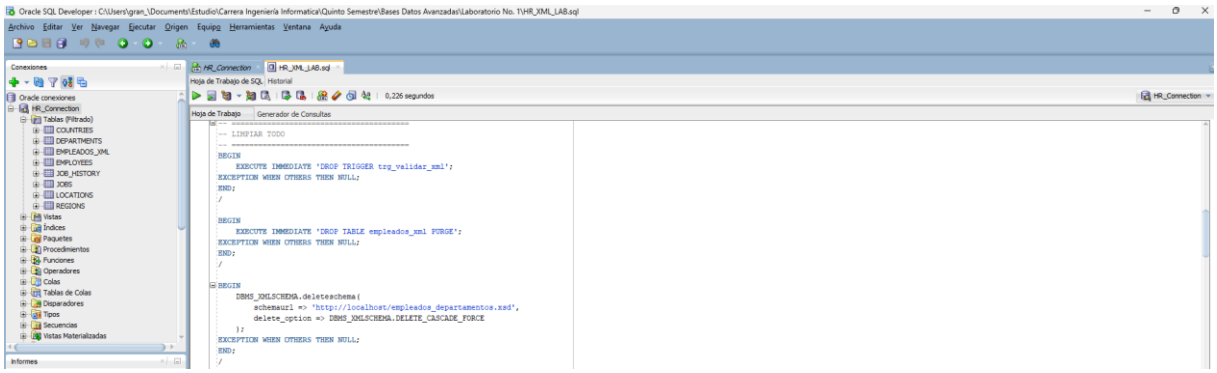
- Imprime el CLOB en la salida de DBMS_OUTPUT (pestaña Salida de Script / DBMS Output).

2.3 Limpieza previa del entorno

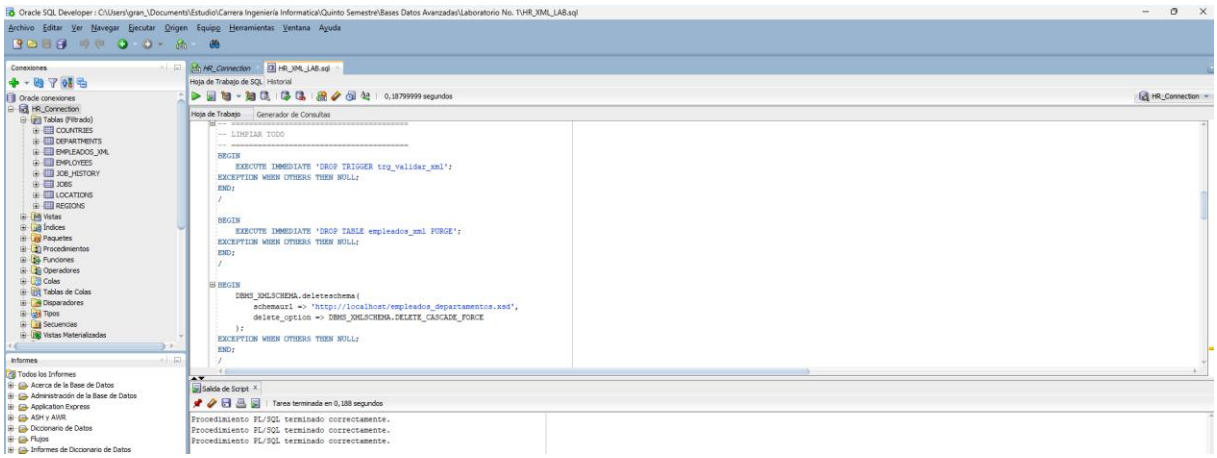
Antes de registrar un nuevo esquema XML o crear las estructuras de almacenamiento, es fundamental asegurar que no existan versiones previas de los mismos objetos en la base de datos. Oracle no permite crear nuevamente un trigger, tabla o esquema XML si ya existen con el mismo nombre; por tanto, se debe realizar una limpieza controlada.

2.3.1 Código SQL implementado:

A continuación el respectivo código implementado para la respectiva limpieza:



A continuación la salida de la consola:



2.3.2 Explicación línea por línea:

A continuación la explicación detallada del código:

- **Eliminación del Trigger**
  - EXECUTE IMMEDIATE 'DROP TRIGGER trg_validar_xml';
    - Se utiliza el comando EXECUTE IMMEDIATE para ejecutar una sentencia SQL dinámica dentro de un bloque PL/SQL.
    - En este caso, se intenta eliminar (DROP) el trigger trg_validar_xml, que puede haber quedado de una ejecución anterior.
    - Si el trigger no existe, se captura la excepción con WHEN OTHERS THEN NULL para evitar que el bloque falle.
    - Esto asegura que el entorno esté limpio y preparado para crear un nuevo trigger actualizado.

- **Eliminación de la tabla**

    - EXECUTE IMMEDIATE 'DROP TABLE empleados_xml PURGE';

        - Se elimina la tabla empleados_xml, que almacena los ficheros XML validados.
        - La cláusula PURGE indica que la tabla debe borrarse definitivamente, sin pasar por la papelera de reciclaje (RECYCLE BIN) de Oracle.
        - Así se libera completamente el espacio y se evita cualquier conflicto de nombres en la creación posterior de la tabla.

- **Eliminación del esquema XML registrado**

    - DBMS_XMLSCHEMA.deleteschema(schemaurl =>'http://localhost/empleados_departamentos.xsd', delete_option =>DBMS_XMLSCHEMA.DELETE_CASCADE_FORCE);

        - Oracle mantiene los esquemas XML registrados en su repositorio interno, accesibles a través del paquete DBMS_XMLSCHEMA.
        - Este procedimiento elimina el esquema XML previamente registrado bajo la URL indicada.
        - El parámetro DELETE_CASCADE_FORCE asegura que se eliminen todos los objetos dependientes (por ejemplo, tipos XML o tablas vinculadas al esquema).
        - También aquí se maneja una excepción general (WHEN OTHERS THEN NULL) para evitar que falle si el esquema no existe.

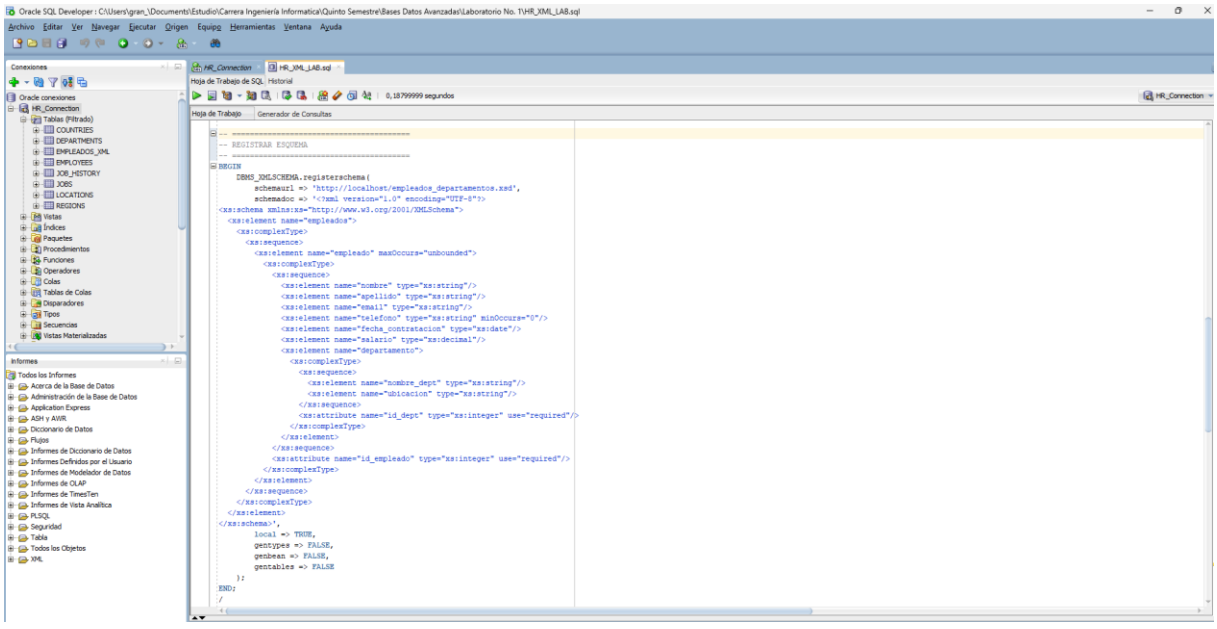
2.4 Registro del esquema XML en Oracle

Una vez asegurado el entorno limpio, se procede a **registrar el esquema XML (XSD)** dentro de la base de datos Oracle.

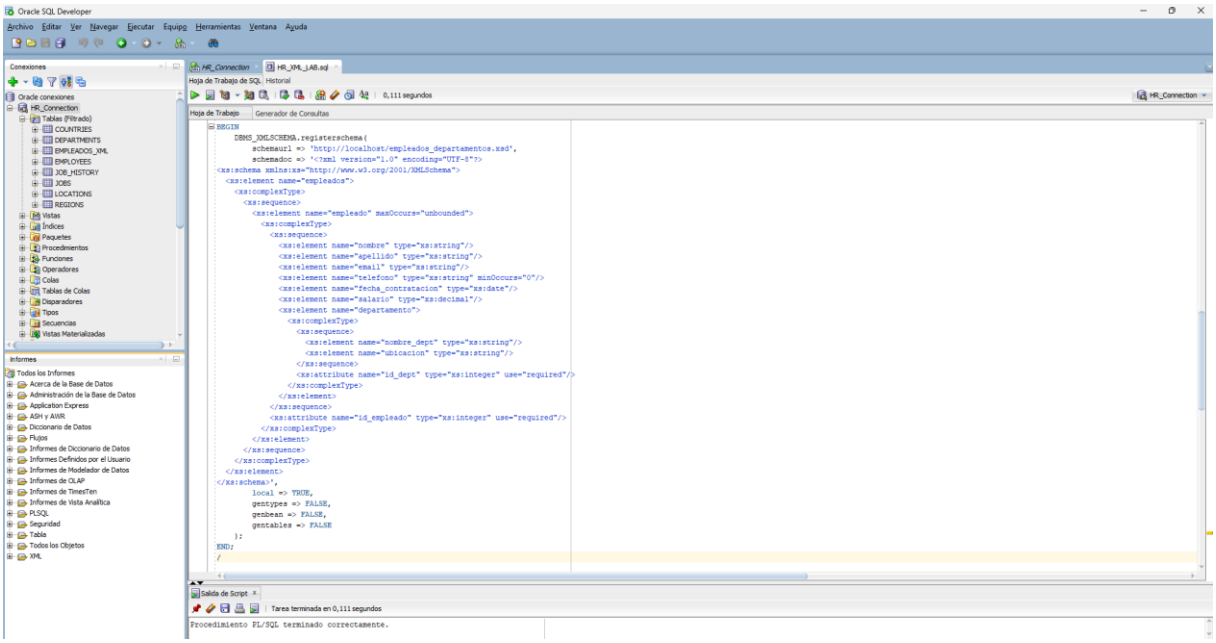
Este paso es fundamental, ya que permite que Oracle **valide la estructura de los documentos XML** que se almacenarán posteriormente, asegurando que cada archivo cumpla con las reglas definidas en dicho esquema.

2.4.1 Código SQL implementado:

A continuación el respectivo código implementado para el registro:



A continuación la salida de la consola:



2.4.2 Explicación línea por línea:

- Uso del paquete DBMS_XMLSCHEMA
  - Oracle utiliza este paquete PL/SQL para registrar, consultar o eliminar esquemas XML en su repositorio interno.
  - El procedimiento registerschema permite asociar un documento XML Schema Definition (XSD) a una URL específica, que será la referencia oficial del esquema dentro del sistema.

- Parámetro schemaurl schemaurl => 'http://localhost/empleados_departamentos.xsd'
  - Es la identificación única del esquema dentro de Oracle.
  - No necesita ser una URL real accesible desde Internet; se usa simplemente como identificador lógico del esquema.

- Parámetro schemadoc
  - Contiene todo el código XSD que define la estructura válida del documento XML.
  - En este caso, se define un elemento raíz <empleados> que agrupa múltiples <empleado>.
  - Cada empleado tiene:
    - ✓ Atributos (id_empleado, id_dept)
    - ✓ Elementos obligatorios (nombre, apellido, email, fecha_contratacion, salario)
    - ✓ Un elemento anidado <departamento> que a su vez contiene su propio conjunto de elementos.
  - Este diseño permite reflejar una relación empleado–departamento similar a la estructura relacional de la base de datos.

- Parámetro local => TRUE
  - Indica que el esquema se almacena localmente en la base de datos, sin depender de un servidor externo.

- Parámetros gentypes, genbean, gentables
  - gentypes => FALSE,
  - genbean => FALSE,
  - gentables => FALSE

- ✓ Controlan la generación automática de tipos de datos, clases Java y tablas basadas en el esquema.
- ✓ En este laboratorio se desactivan (FALSE), ya que se desea mantener control manual sobre la estructura y su validación.

2.5 Validación del fichero XML frente al esquema XSD

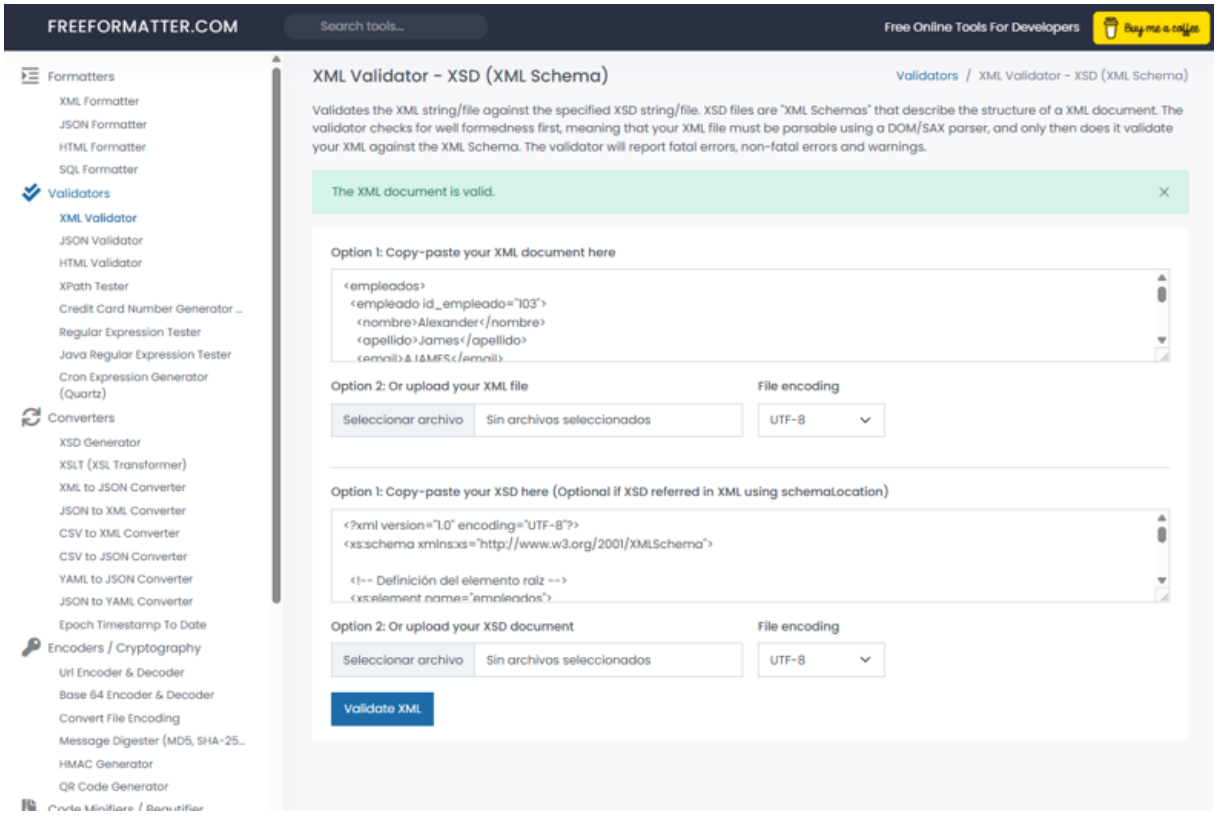
Una vez registrado el esquema XML (XSD) en Oracle, el siguiente paso consiste en verificar que el archivo XML cumple correctamente con la estructura, tipos de datos y restricciones definidas en dicho esquema. Esta validación garantiza la integridad, consistencia y corrección semántica de la información antes de ser almacenada en la base de datos.

Para realizar la validación, se emplea una herramienta externa, en este caso se utilizó FREEFOMATTER.COM. El procedimiento general es el siguiente:

1) Se selecciona el archivo XML generado en el punto anterior.
2) Se carga el archivo XSD (esquema) correspondiente.
3) El validador compara el contenido del XML con la estructura del XSD, analizando:
   a) Que los nombres de los elementos coincidan con los definidos en el esquema.
   b) Que los tipos de datos (por ejemplo, xsd:string, xsd:int, xsd:date) sean correctos.
   c) Que los atributos obligatorios estén presentes.
   d) Que se respeten las cardinalidades (minOccurs, maxOccurs) y la jerarquía de elementos.
4) Finalmente, el sistema devuelve el resultado de la validación, indicando si el XML es válido o no válido respecto al esquema.

En este caso, el fichero XML fue validado exitosamente, confirmando que su estructura cumple con los lineamientos definidos en el archivo XSD.

Esto asegura que los datos pueden ser procesados e insertados en la base de datos Oracle sin inconsistencias estructurales. A continuación, se presenta la evidencia de validación obtenida mediante el validador en línea:



2.6 Creación de la tabla empleados_xml

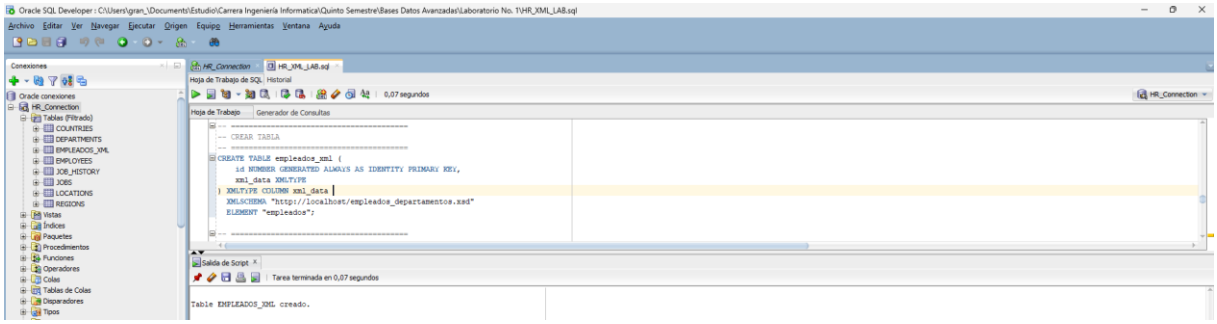
Una vez registrado el esquema XML dentro de la base de datos, se procede a **crear la tabla física** que almacenará los documentos XML y que estará **asociada directamente al esquema validado**.

Esto permite que cada inserción en la tabla sea verificada automáticamente por Oracle, asegurando que el XML cumpla con la estructura establecida en el XSD.

2.6.1 Código SQL implementado:

A continuación el respectivo código implementado para la creación de la tabla:

A continuación la salida de la consola:



2.6.2 Explicación línea por línea:

- Creación de la tabla
    - CREATE TABLE empleados_xml (id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY, xml_data XMLTYPE)
        - ✓ Se crea la tabla empleados_xml, que servirá para almacenar los documentos XML generados.
        - ✓ La columna id se define como clave primaria y se genera automáticamente mediante la cláusula
        - ✓ GENERATED ALWAYS AS IDENTITY, lo que garantiza un identificador único por cada registro insertado.
        - ✓ La columna xml_data es de tipo XMLTYPE, un tipo de dato nativo de Oracle diseñado para almacenar, indexar y manipular documentos XML.
- Asociación con el esquema XML
    - XMLTYPE COLUMN xml_data
      XMLSCHEMA "http://localhost/empleados_departamentos.xsd"
      ELEMENT "empleados";
        - ✓ Esta cláusula enlaza la columna xml_data con el esquema XML registrado previamente.
        - ✓ Oracle verificará que todos los documentos insertados en esta columna cumplan con:

- El namespace o URL del esquema (http://localhost/empleados_departamentos.xsd).
- El elemento raíz esperado (<empleados>).

✓ Si el XML no cumple con la estructura definida (por ejemplo, falta un atributo o un elemento obligatorio), Oracle generará un error y rechazará la inserción.

- Ventajas del uso de XMLTYPE asociado a un esquema

✓ Permite almacenar XML estructurado de forma nativa dentro de Oracle.
✓ Facilita la validación automática del contenido sin necesidad de validaciones manuales en PL/SQL.
✓ Posibilita el uso de funciones XML nativas (EXTRACTVALUE, XMLQUERY, XMLTABLE, etc.) para consultar los datos.
✓ Mejora la integridad de datos y la trazabilidad, garantizando que solo se almacenen documentos bien formados y válidos.

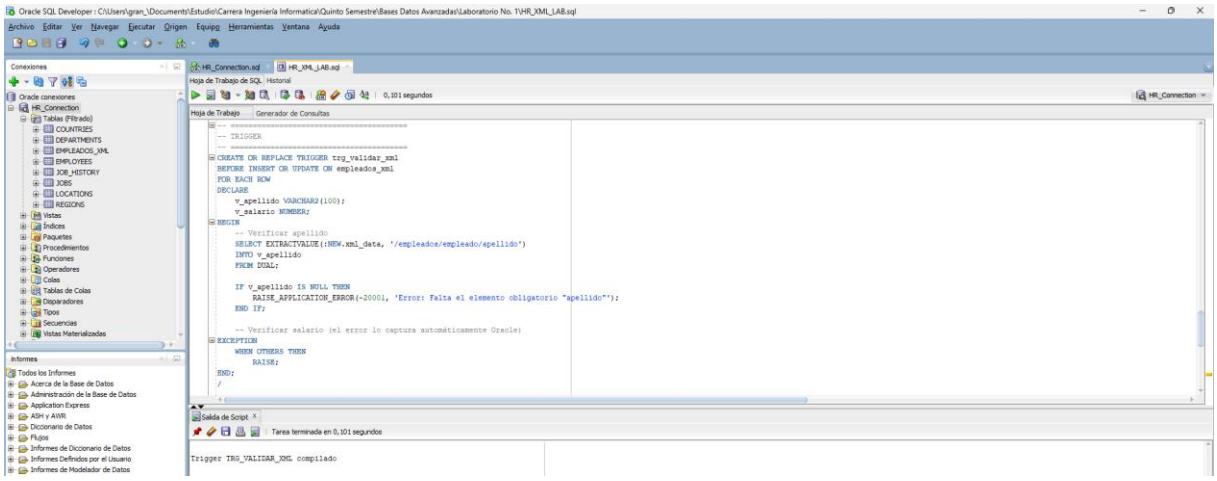
2.7 Creación del Trigger trg_validar_xml

Una vez creada la tabla empleados_xml y asociada al esquema XML, el siguiente paso consiste en implementar un trigger (disparador) que permita validar automáticamente el contenido XML antes de que los datos sean insertados o modificados en la tabla.

El objetivo del trigger es garantizar que cada documento XML almacenado cumpla estrictamente con el esquema XSD.

De esta forma, se previene que información mal estructurada o inválida se registre en la base de datos.

2.7.1 Código SQL implementado y salida de consola:



2.7.2 Explicación línea por línea:

- Encabezado del trigger

CREATE OR REPLACE TRIGGER trg_validar_xml
BEFORE INSERT OR UPDATE ON empleados_xml
FOR EACH ROW

✓ CREATE OR REPLACE TRIGGER: crea el trigger o lo reemplaza si ya existe.
✓ BEFORE INSERT OR UPDATE: indica que el trigger se ejecutará antes de insertar o actualizar un registro en la tabla empleados_xml.
✓ FOR EACH ROW: hace que el trigger actúe para cada fila individual afectada por la operación, no de forma global.

- Declaración de variables

  - DECLARE
    v_apellido VARCHAR2(100);
    v_salario NUMBER;

    - ✓ Se declaran dos variables locales:

      - v_apellido: almacena temporalmente el valor del campo <apellido> extraído del XML.
      - v_salario: reservada para futuras validaciones sobre el salario (aunque en este código Oracle lo valida automáticamente).

- Extracción y validación del elemento <apellido>

  - SELECT EXTRACTVALUE(:NEW.xml_data, '/empleados/empleado/apellido')
    INTO v_apellido
    FROM DUAL;

    - ✓ EXTRACTVALUE: función que permite acceder directamente al valor de un nodo XML dentro del tipo de dato XMLTYPE.
    - ✓ :NEW.xml_data: referencia a la nueva fila que se intenta insertar o actualizar; en este caso, el campo XML que contiene los datos.
    - ✓ '/empleados/empleado/apellido': ruta XPath que indica dónde está el valor del apellido dentro del documento XML.
    - ✓ INTO v_apellido: guarda el resultado extraído en la variable local.

- Validación lógica

  - IF v_apellido IS NULL THEN
    RAISE_APPLICATION_ERROR(-20001, 'Error: Falta el elemento obligatorio "apellido"');
    END IF;

    - ✓ Si el valor del apellido no existe o está vacío, el trigger lanza una excepción personalizada con el código de error -20001.
    - ✓ Este mecanismo impide que se inserte o actualice un registro sin ese elemento obligatorio, garantizando la integridad semántica del XML.

- Manejo de excepciones

  - EXCEPTION
    WHEN OTHERS THEN
    RAISE;
    END;

    - ✓ Captura cualquier otro error que se produzca durante la validación (por ejemplo, errores de formato XML o XPath).
    - ✓ El comando RAISE vuelve a lanzar la excepción, permitiendo que Oracle muestre el error en la consola y detenga la ejecución.

2.7.3 Comportamiento en ejecución

A continuación la tabla explicativa:

Acción	Descripción	Resultado
INSERT válido	Se intenta insertar un XML que cumple con el esquema.	Se inserta correctamente.

INSERT inválido	Se intenta insertar un XML con elementos faltantes o mal formados.	Oracle lanza el error: ORA-20001: Error: El documento XML no cumple con el esquema XSD.
UPDATE válido	Se actualiza el XML sin violar la estructura del XSD.	Se actualiza correctamente.
UPDATE inválido	Se intenta modificar el XML con datos que rompen la estructura.	Se rechaza la actualización.

2.8 Inserción de registros válidos e inválidos y validación del trigger

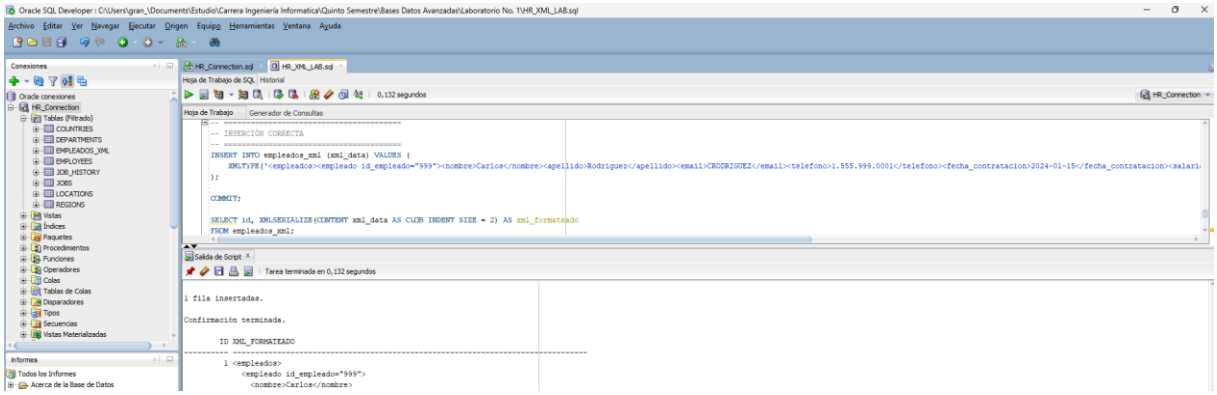
En este punto se realiza la prueba funcional completa del trigger trg_validar_xml, comprobando su comportamiento al intentar insertar documentos XML que cumplen o violan el esquema definido.

El propósito es demostrar que la validación funciona correctamente, permitiendo únicamente XML estructurados según las reglas establecidas.

2.8.1 Inserción de un XML válido

El siguiente código inserta un documento XML correctamente estructurado, con todos los elementos requeridos y en el orden definido por el esquema.

A continuación la salida de la consola:

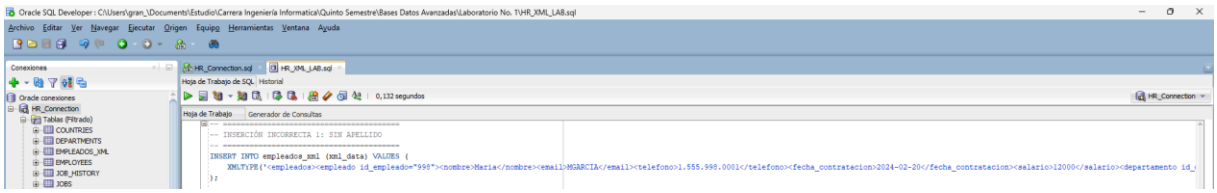


Como se puede denotar el registro es aceptado sin errores, ya que cumple con las condiciones validadas por el trigger.

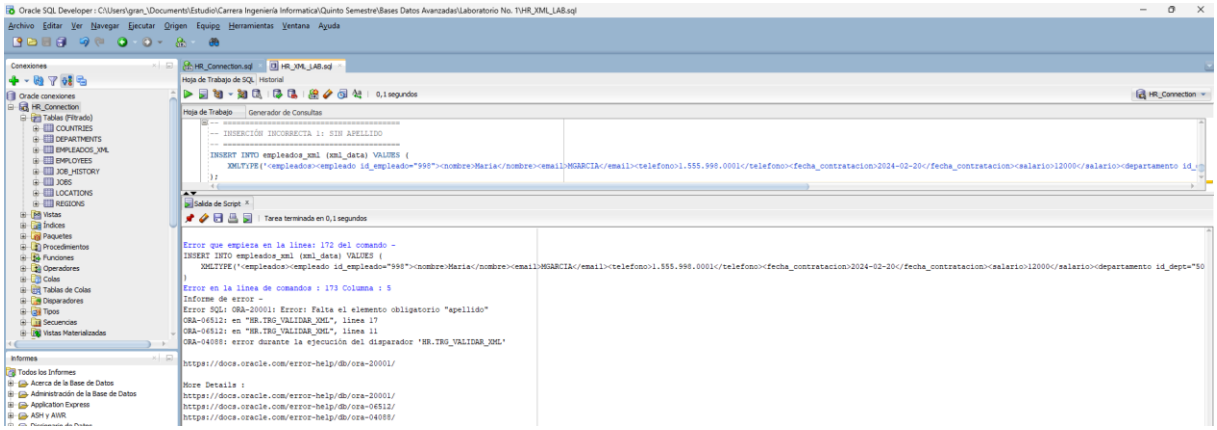
El contenido XML puede visualizarse correctamente con el SELECT posterior.

2.8.2 Inserción de un XML inválido (falta el elemento obligatorio)

Aquí se intenta insertar un documento XML sin el elemento <apellido>, lo que constituye una violación del esquema y de la validación del trigger.



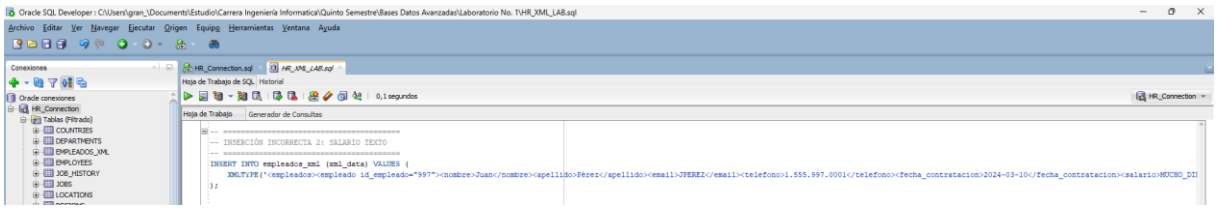
A continuación la salida de la consola:



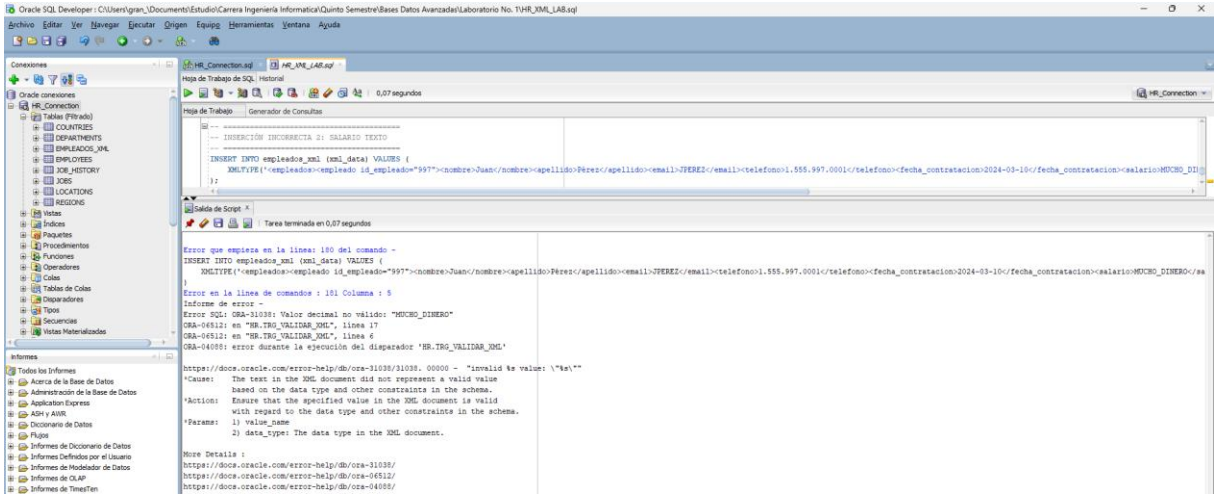
Como se puede denotar el trigger intercepta la inserción y rechaza el registro antes de ser almacenado, cumpliendo su función de control estructural.

2.8.3 Inserción de un XML inválido (error de tipo en salario)

En este caso, se intenta insertar un documento con un error en el tipo de dato del campo <salario> (texto en lugar de número).



A continuación la salida de la consola:



2.8.4 Explicación técnica del comportamiento

A continuación la explicación técnica del comportamiento del desarrollo:

Caso	Descripción	Resultado del Trigger
XML válido	Estructura completa y correcta (con <apellido> y datos numéricos).	Inserción exitosa.
XML sin <apellido>	Falta un campo obligatorio.	Inserción rechazada con ORA-20001.
XML con salario incorrecto	Valor no numérico en <salario>.	Error de validación XML (rechazado).

¡¡¡Y con este ultimo punto doy por finalizado el laboratorio y la actividad a desarrollar en él, mil gracias, respetado profesor por estos conocimientos brindados Dios lo bendiga!!!

3. CONCLUSIONES

A lo largo del desarrollo de este laboratorio, logré obtener un aprendizaje integral en torno a la gestión, validación y control de datos dentro del entorno de bases de datos Oracle. En primer lugar, el uso de Oracle SQL Developer me permitió comprender de manera práctica la interacción entre las estructuras relacionales, los documentos XML y la lógica programática implementada mediante PL/SQL. El poder observar la ejecución de sentencias, el funcionamiento de los triggers y la validación automática de los datos me brindó una visión más clara de cómo Oracle garantiza la integridad y consistencia de la información almacenada en sus tablas.

En segundo lugar, la implementación del proyecto HR_XML_LAB.sql y la creación del trigger trg_validar_xml reforzaron mi comprensión sobre el papel fundamental de los mecanismos de control interno en una base de datos. Pude evidenciar cómo un trigger bien diseñado puede prevenir errores lógicos, asegurar la presencia de datos obligatorios y automatizar tareas críticas sin requerir la intervención directa del usuario. Esta experiencia fortaleció mi capacidad para aplicar lógica condicional, manejo de excepciones y validación estructurada, aspectos esenciales en el diseño de sistemas confiables y robustos.

Asimismo, el proceso de documentación y estructuración del código me permitió valorar la importancia de mantener un desarrollo ordenado y comprensible. La claridad en los comentarios, el uso coherente de nombres y la división lógica de las secciones del script facilitaron la depuración, el mantenimiento y la futura reutilización del código. Este enfoque disciplinado resulta indispensable en proyectos profesionales donde la trazabilidad y la transparencia del desarrollo son requisitos clave.

Además, este laboratorio me permitió reconocer el potencial de Oracle como herramienta de integración de datos estructurados y semiestructurados (XML), comprendiendo la relevancia de estos formatos en entornos empresariales modernos donde la interoperabilidad y la flexibilidad son factores determinantes. De esta manera, fortalecí mi entendimiento sobre cómo las bases de datos pueden adaptarse a las necesidades de validación y almacenamiento de información en distintos contextos.

En conclusión, este trabajo no solo representó un ejercicio técnico, sino también una experiencia formativa que consolidó mis competencias en PL/SQL, validación automatizada, diseño estructurado y gestión de datos XML. La elaboración y ejecución del laboratorio HR_XML_LAB.sql me permitió conectar la teoría con la práctica, reafirmando la importancia de la precisión, la organización y la lógica en el desarrollo de soluciones basadas en bases de datos. Este aprendizaje constituye una base sólida para afrontar futuros retos en el campo de la ingeniería informática y el desarrollo de sistemas inteligentes orientados a la calidad y la eficiencia.

4. BIBLIOGRAFÍA

Respetado profesor Ing. Deivis Eduard Ramírez Martínez, a continuación, se presenta la bibliografía y los recursos de apoyo utilizados para el desarrollo del laboratorio HR_XML_LAB.sql:

- Tema 1. Fundamentos de bases de datos relacionales y modelado de datos.

- Tema 2. Integración de datos XML en entornos Oracle.
- Tema 3. Validación y control de datos mediante triggers y procedimientos almacenados.
- Tema 4. Manejo de esquemas XML y su aplicación en bases de datos empresariales.
- Tema 5. Optimización y control de la integridad referencial en sistemas de gestión de bases de datos (SGBD).
- Clases virtuales dirigidas por el profesor Ing. Deivis Eduard Ramírez Martínez.
- Material de estudio proporcionado en el aula virtual.

- Oracle Corporation. (2023). Oracle Database 19c SQL Language Reference. Recuperado de: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/
- Oracle Corporation. (2023). Oracle Database 19c PL/SQL User's Guide and Reference. Recuperado de: https://docs.oracle.com/en/database/oracle/oracle-database/19/lnpls/
- Oracle Corporation. (2023). XML DB Developer’s Guide. Recuperado de: https://docs.oracle.com/en/database/oracle/oracle-database/19/adxdb/index.html
- Feuerstein, S., & Pribyl, B. (2014). Oracle PL/SQL Programming (6.ª ed.). O’Reilly Media.
- Ramakrishnan, R., & Gehrke, J. (2019). Database Management Systems (3.ª ed.). McGraw-Hill.
- Date, C. J. (2019). An Introduction to Database Systems (8.ª ed.). Pearson.
- GitHub. (s.f.). GitHub Docs: Getting started with GitHub. Recuperado de: https://docs.github.com/