Documentación Técnica

Descripción del caso práctico implementado

Se implementó un sistema de replicación bidireccional entre dos motores de base de datos heterogéneos: Oracle y MySQL. El objetivo es mantener sincronizadas las operaciones CRUD (Create, Read, Update, Delete) en ambas bases de datos, garantizando la integridad y coherencia de la información en un entorno donde ambas se consultan o modifican de forma concurrente.

La replicación está basada en el uso de una bitácora de operaciones. Cada operación relevante (INSERT, UPDATE, DELETE) es registrada en dicha bitácora con los datos necesarios para replicarla en la base remota. Los procedimientos programados y jobs se encargan de procesar esas entradas y aplicar los cambios.

Este sistema fue desarrollado para una base de datos clínica, con un conjunto de entidades como paciente, doctor, cita, receta, etc.

Diagrama lógico del esquema utilizado

Entidades principales:

paciente(id_paciente, nombre, fecha_nacimiento, telefono, email, genero)

clinica(id_clinica, nombre, ciudad, direccion)

doctor(id_doctor, nombre, especialidad, id_clinica)

cita(id_cita, id_paciente, id_doctor, fecha_cita, motivo, estado)

historial_medico(id_historial, id_paciente, descripcion, fecha_registro)

receta(id_receta, id_cita, fecha_emision, observaciones)

medicamento(id_medicamento, nombre, descripcion, laboratorio)

receta_medicamento(id_receta, id_medicamento, dosis, frecuencia, duracion)

bitacora(id_bitacora, tabla, operacion, id_registro, datos, origen, replicado, intentos, ultimo_error)

Relaciones:

Un paciente puede tener muchas citas.

Una cita es atendida por un doctor.

Un doctor trabaja en una clínica.

Un paciente tiene historial médico.

Una cita genera una receta que incluye múltiples medicamentos.

Justificación de la estrategia de replicación

Se optó por una replicación asincrónica basada en bitácora porque:

- Permite trazabilidad: cada cambio queda registrado y puede ser auditado.

- Evita conflictos de escritura simultánea: cada instancia actúa como master local y replica al otro extremo.

- Soporta entornos heterogéneos (Oracle y MySQL).

- Es escalable: pueden agregarse validaciones y filtros sin afectar la lógica general.

- Es segura: los triggers incluyen una bandera de control para evitar loops infinitos y mutaciones de tabla.

Para evitar replicación circular o recursiva, se implementó una variable de control (en Oracle, un paquete con bandera; en MySQL, una tabla de control) que indica cuándo un trigger debe evitar ejecutar su lógica.

Manual técnico con procedimientos, disparadores, vistas y transacciones

A. Triggers (Oracle & MySQL):

Se agregaron triggers AFTER INSERT/UPDATE/DELETE para cada tabla que escribe en la bitácora.

Cada trigger incluye una condición:

En Oracle: IF replicacion_ctx.replicando THEN RETURN; END IF;

En MySQL: consulta de la tabla replicacion_control.

B. Paquetes y Procedimientos:

aplicar_json_replica(tabla, operacion, id_registro, datos):

Aplica operaciones INSERT, UPDATE, DELETE a una tabla específica a partir de un JSON.

replicar_mysql_a_oracle():

Lee la bitácora de MySQL y aplica los cambios en Oracle.

Usa replicacion_ctx.replicando := TRUE para evitar que triggers se activen.

replicar_oracle_a_mysql():

Realiza lo inverso, usando un flag en MySQL (tabla replicacion_control).

marcar_replicado_mysql(id_bitacora):

Marca un registro como replicado en MySQL.

insertar_en_bitacora_mysql(...):

Llamado desde Oracle si se replica hacia MySQL (en el job o trigger Oracle).

C. Jobs:

Oracle: DBMS_SCHEDULER job llamado JOB_REPLICAR_MYSQL_A_ORACLE que ejecuta cada 5 minutos el procedimiento replicar_mysql_a_oracle.

D. Tablas de apoyo:

replicacion_ctx (Oracle) → paquete con bandera booleana.

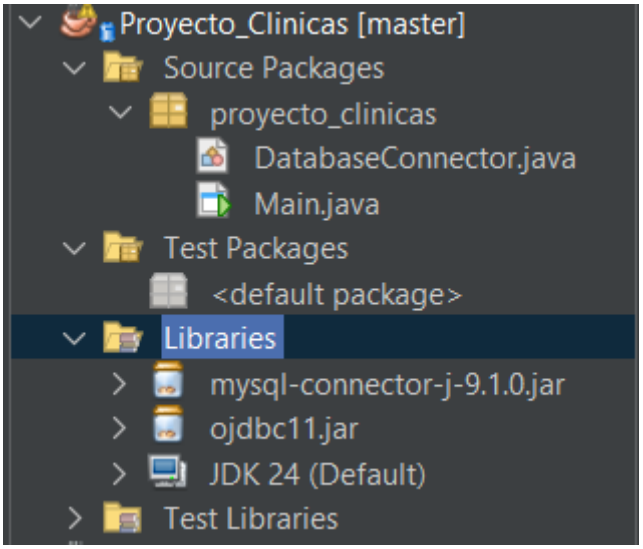
replicacion_control (MySQL) → tabla con campo replicando TINYINT(1).

E. Control de errores:

En caso de error durante replicación, el error se guarda en el campo ultimo_error (VARCHAR2(2000)) y se incrementa el contador de intentos.

Manual de Aplicativo Forzador de Replicación

-Asegurarse de tener las librerías y jars adecuados en la carpeta “Libraries” del aplicativo:



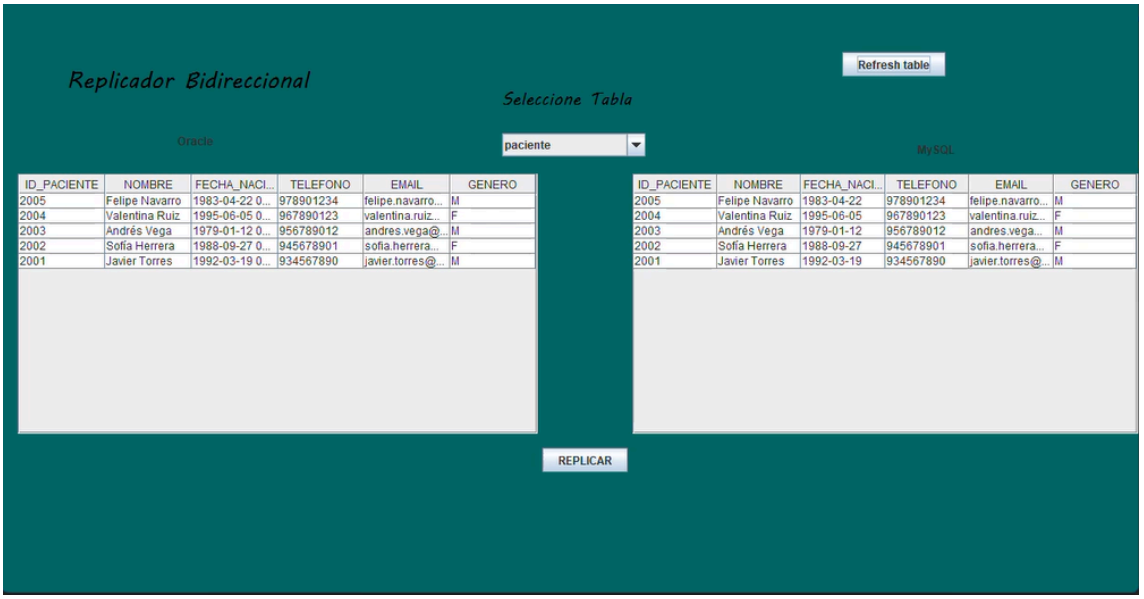
-Asegurarse de que las credenciales en la clase “DatabaseConnector” esten ingresadas correctamente.

```
// Configuración para Oracle
private static final String ORACLE_JDBC_URL = "jdbc:oracle:thin:@//20.185.224.121:1521/ORCLPDB";
private static final String ORACLE_USER = "administrador";
private static final String ORACLE_PASSWORD = "W8xd7YKmk9sYWwJn09SgJTMGHw0qvR";

// Configuración para MySQL
private static final String MYSQL_JDBC_URL = "jdbc:mysql://clinica-mysql.mysql.database.azure.com:3306/clinica?useSSL=true&requireSSL=true&verify

private static final String MYSQL_USER = "userclinicasql";
private static final String MYSQL_PASSWORD = "W8xd7YKmk9sYWwJn09SgJTMGHw0qvR";
```

-Dentro del aplicativo se puede escoger la tabla sobre la cual se quiere forzar la replicación. Esto se escoge con el Combo box al centro arriba de la pantalla



-Una vez realizadas las acciones deseadas en una base, Seleccionar la Tabla sobre la cual se realizó la acción y oprimir el botón de Replicar. Los cambios deberían de ser aplicados en la base opuesta.

-Oprimir el Botón Refresh para visualizarlo en el aplicativo