



## INFORME DE GUÍA PRÁCTICA

### I. PORTADA

Tema:	APE 4. REPLICACIÓN DE DATOS
Unidad de Organización Curricular:	PROFESIONAL
Nivel y Paralelo:	Quinto - A
Alumnos participantes:	Cholota Guaman Carlos Sebastian Mazabanda Pilamunga Diego Abraham Tixilema Puaquiza Kevin Alexander Tubon Chipantiza Danilo Alexander
Asignatura:	Sistemas de Base de Datos Distribuidos
Docente:	Ing. Rubén Caiza, Mg.

### II. INFORME DE GUÍA PRÁCTICA

#### 2.1 Objetivos

##### **General:**

Configurar una Réplica de Instantánea en una instancia diferente

##### **Específicos:**

- Implementar tablas temporales (staging) para recibir y organizar datos externos de pacientes y consultas, simulando un proceso real de integración hospitalaria.
- Aplicar procesos de sincronización con MERGE, logrando actualizaciones selectivas de registros existentes e inserciones controladas de nuevos pacientes y consultas.
- Verificar y documentar los cambios realizados, asegurando trazabilidad, integridad referencial y reproducibilidad académica mediante evidencias y consultas de validación.

#### 2.2 Modalidad

Presencial.

#### 2.3 Tiempo de duración

**Presenciales:** 4

**No presenciales:** 0

#### 2.4 Instrucciones

Conéctese al motor de base de datos

- Active la instancia principal o por defecto
- Configure la instancia para ser Distribuidor y Publicador
- Cree una Publicación de una tabla
- Cree una suscripción en la instancia Sitio\_A y/o Sitio\_B, a la publicación creada anteriormente
- Despliegue la publicación ? mostrar los registros
- Despliegue la suscripción y verifique que los datos son exactamente los mismos a la publicación

#### 2.5 Listado de equipos, materiales y recursos

Listado de equipos y materiales generales empleados en la guía práctica:

- **Computador**



TAC (Tecnologías para el Aprendizaje y Conocimiento) empleados en la guía práctica:

- ☐ Plataformas educativas
- ☐ Simuladores y laboratorios virtuales
- ☐ Aplicaciones educativas
- ☐ Recursos audiovisuales
- ☐ Gamificación
- ☒ Inteligencia Artificial

Otros (Especifique): \_\_\_\_\_

## 2.6 Actividades por desarrollar

Detalladas en la guía proporcionada por el docente.

## 2.7 Resultados obtenidos

### Ejercicios Prácticos en Google Cloud

Implementación de Instancias en Cloud SQL con Alta Disponibilidad (Multi-AZ/HA) y Gestión Avanzada de Réplicas de Lectura

Paso 1: Acceso a la Consola de Cloud SQL

Para iniciar el proceso de creación de instancias con configuración de alta disponibilidad, es necesario acceder al menú principal de la consola de administración de Google Cloud.

- Diríjase al Menú de navegación ubicado en la parte superior izquierda de la interfaz.
- Seleccione la opción Cloud SQL.
- Dentro de esta sección, haga clic en Instancias, donde se gestionan todas las bases de datos desplegadas en el entorno de Google Cloud.

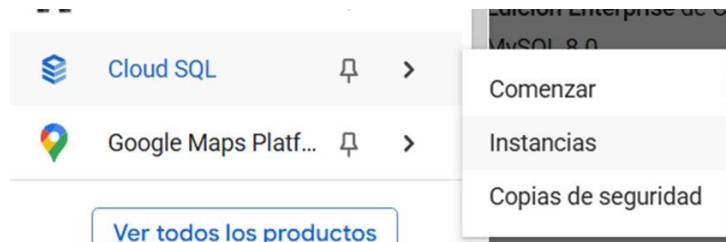


Ilustración 1. Google cloud

Paso 2: Creación de una Nueva Instancia

En la pantalla principal de Cloud SQL, seleccionar la opción Crear instancia.

Esta acción habilita el asistente de configuración, desde el cual se definen los parámetros iniciales de la instancia, incluyendo el motor de base de datos, la versión y las opciones de disponibilidad.

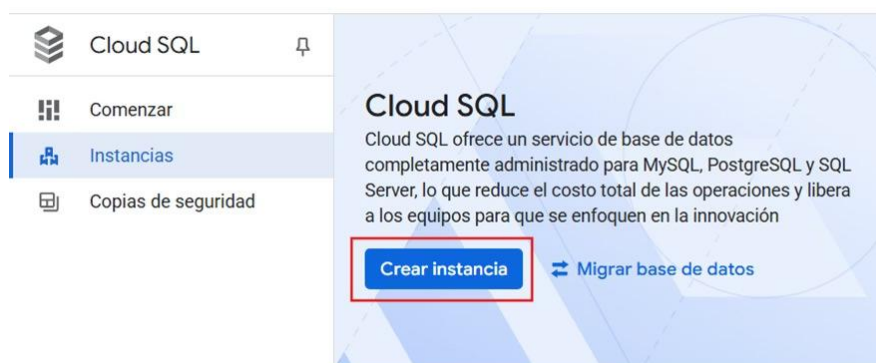
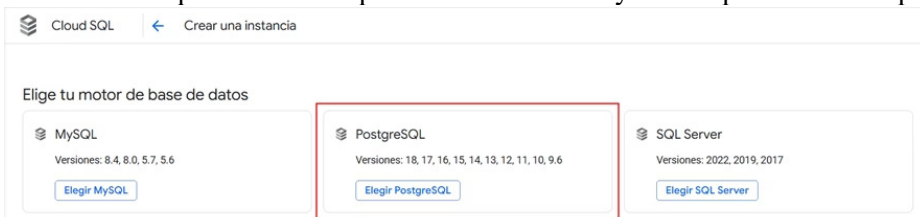


Ilustración 2. Creación de instancia



### Paso 3: Selección del Sistema de Gestión de Base de Datos

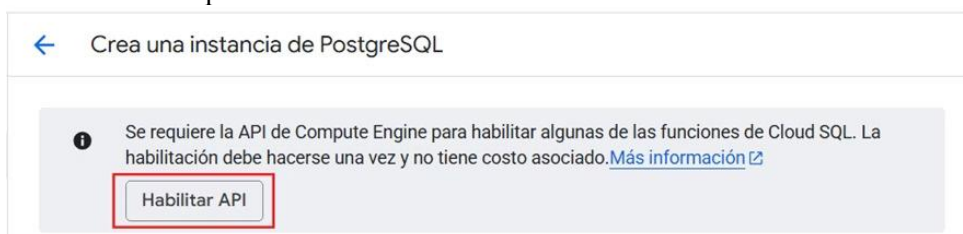
En el asistente de creación de instancia, elegir el motor de base de datos que se utilizará. Las opciones disponibles incluyen MySQL, PostgreSQL y SQL Server. La selección debe responder a los requerimientos técnicos y de compatibilidad del proyecto.



*Ilustración 3. Selección de sistema*

### Paso 4: Habilidadación de la API de Compute Engine

Antes de continuar con la configuración de alta disponibilidad, es necesario habilitar la API de Compute Engine, ya que esta proporciona la infraestructura subyacente para la creación de instancias en múltiples zonas.



*Ilustración 4. Habilidadación de instancia*

### Paso 5: Selección de la Edición de Cloud SQL

En el asistente de configuración, elegir la edición de Cloud SQL que definirá las características fundamentales de la instancia:

- Enterprise Plus: Garantiza un SLA del 99.99%, con mantenimiento en menos de un segundo y capacidad de escalación vertical prácticamente sin tiempo de inactividad.
- Enterprise: Ofrece un SLA del 99.95%, con un máximo de 60 segundos de inactividad planificada durante tareas de mantenimiento

<b>Enterprise Plus</b> <ul style="list-style-type: none"><li>• 99.99% disponibilidad</li><li>• Mantenimiento &lt;1 segundo</li><li>• Máquinas optimizadas</li></ul>	<b>Enterprise</b> <ul style="list-style-type: none"><li>• 99.95% disponibilidad</li><li>• Mantenimiento &lt;60 segundos</li><li>• Máquinas de uso general</li></ul>
---	---

*Ilustración 5. Selección de edición*



### Configuración de Instancia y Región

Define el nombre de tu instancia (mínimo 8 caracteres) y selecciona la región más cercana a tus servicios. Luego, seleccionar el ajuste de edición, en este caso: Zona de pruebas

#### Información de la instancia

Versión de la base de datos \*

PostgreSQL 17

ID de instancia \*

postgres1

Usa letras minúsculas, números y guiones. Comienza con una letra.

Contraseña \*

JosueG\_17

Generar

Establece una contraseña para el usuario administrador predeterminado "postgres". [Más información](#)

☒ Debe contener una letra mayúscula, una letra minúscula, un número y un carácter no alfanumérico

☒ No debe contener el nombre de usuario

☒ Debe tener 8 caracteres como mínimo

*Ilustración 6. Configuración de instancia*

Después, elige disponibilidad zonal múltiple (HA) que aplica conmutación por error automática a otra zona. Aunque aumenta costos, es esencial para garantizar continuidad operativa y cumplir con Multi-AZ.

#### Elige la región y la disponibilidad zonal

Para obtener un mejor rendimiento, mantén tus datos cerca de los servicios que los necesitan. La región es permanente, mientras que la zona se puede cambiar en cualquier momento.

Región

us-central1 (Iowa)

Disponibilidad zonal

☐ Zona única

En caso de interrupción del servicio, no se aplica la conmutación por error. No se recomienda esta opción para la producción.

☒ Varias zonas (con alta disponibilidad)

La conmutación por error automática se aplica a otra zona en la región que seleccionaste. Esta opción se recomienda para las instancias de producción. Aumenta el costo.

Zona principal

Cualquiera

Zona secundaria

Cualquiera (diferente ...)

*Ilustración 7. Disponibilidad zonal*



### Configuración de Máquina y Almacenamiento

En la configuración de la instancia, definir los recursos de cómputo y almacenamiento:

- Tipo de máquina: Seleccionar según los requerimientos de procesamiento y memoria del proyecto.

#### Personaliza tu instancia

También puedes personalizar las opciones de configuración de instancias más adelante

#### Configuración de la máquina

##### Máquina

Elige la máquina que mejor se adapte a las necesidades de recursos de tu instancia. La selección de la máquina determina el tipo de almacenamiento y las opciones de configuración.

De uso general: núcleo compartido

- ☒ 1 CPU virtuales, 0.614 GB
- ☐ 1 CPU virtuales, 1.7 GB

*Ilustración 8. Configuración de maquina*

- Disco de almacenamiento: Optar por SSD, recomendado por su menor latencia y mayor capacidad de consultas por segundo (QPS) en comparación con HDD.
- Capacidad: Establecer un tamaño entre 10 y 250 GB, o definir un valor personalizado según la carga esperada.
- Ajustes de crecimiento: Deshabilitar los aumentos automáticos de almacenamiento para mantener control sobre el espacio y evitar consumos inesperados.

#### Almacenamiento

##### Tipo de almacenamiento ?

- ☒ SSD  
La opción más popular. Menor latencia que un HDD, con mayor QPS y capacidad de procesamiento de datos.
- ☐ HDD  
Menor rendimiento que SSD con tarifas de almacenamiento más bajas.

##### Capacidad de almacenamiento

10 - 3,054 GB. Las capacidades mayores mejoran el rendimiento hasta el límite que impone el tipo de máquina. La capacidad no puede disminuirse más adelante.

- ☒ 10 GB
- ☐ 20 GB
- ☐ 100 GB
- ☐ 250 GB
- ☐ Personalizado

- ☐ Habilitar los aumentos de almacenamiento automáticos  
Si la opción está habilitada, cuando te acerques al límite de capacidad, el almacenamiento se incrementará de manera progresiva (y permanente). [Más información](#)

*Ilustración 9. Capacidad de almacenamiento*





## Configuración de Conexiones y Seguridad

En la sección de red y seguridad de la instancia, establecer los siguientes parámetros:

- IP pública: Configurar una dirección IP pública para permitir el acceso externo a la base de datos.

### Conexiones

Elige cómo quieres que se conecte la fuente a esta instancia y, luego, define las redes que estarán autorizadas para conectarse. [Más información](#)

Puedes usar el proxy de Cloud SQL para aumentar la seguridad con cualquiera de las opciones. [Más información](#)

#### Asignación de IP de la instancia

☐ IP privada

Asigna una dirección IP de VPC interna alojada en Google. Se requieren API y permisos adicionales. No se puede inhabilitar una vez que se habilitó. [Más información](#)

☒ IP pública

Asigna una dirección IP externa a la que se puede acceder a través de Internet. Se requiere el uso de una red autorizada o el proxy de Cloud SQL para conectarse a esta instancia. [Más información](#)

*Ilustración 10. Seguridad de instancia*

- Seguridad SSL: Habilitar la opción de SSL obligatorio en todas las conexiones públicas.
- Certificados: Asegurar que los clientes dispongan de certificados válidos para establecer conexiones seguras.

### Seguridad

#### Modo SSL

Transmisión segura de datos entre el cliente y esta instancia aplicando forzosamente la SSL encryption and certificates. Se recomienda la encriptación SSL para todas las conexiones que usan redes de IP públicas. [Más información](#)

☐ Permitir el tráfico de red sin encriptar (no recomendado)

☒ Solo permitir conexiones SSL

Solo permite conexiones usando encriptación SSL/TLS. No se verificarán los certificados de cliente.

☐ Exigir certificados de cliente de confianza

Solo permite conexiones de clientes que usan un certificado de cliente y encriptación SSL válidos. La autenticación basada en IAM requiere de conectores de Cloud SQL (proxy de autenticación o bibliotecas de lenguajes) para la aplicación de la verificación de certificados.

#### Modo de autoridad certificadora del servidor

Elige el tipo de autoridad certificadora que firma el certificado del servidor para esta instancia de Cloud SQL. [Más información](#)

El modo de autoridad certificadora del servidor no se puede cambiar una vez que se elige para la instancia.

☒ Autoridad certificadora interna administrada por Google

Una autoridad certificadora interna por instancia es el ancla de confianza de la instancia.

*Ilustración 11. Opcion SSL*



### Configuración Adicional y Creación de Instancia

En la fase final de configuración:

- Eliminación de instancia: Definir las políticas de eliminación, asegurando que la instancia pueda ser removida de forma controlada cuando ya no sea necesaria.

#### Protección contra la eliminación de instancias

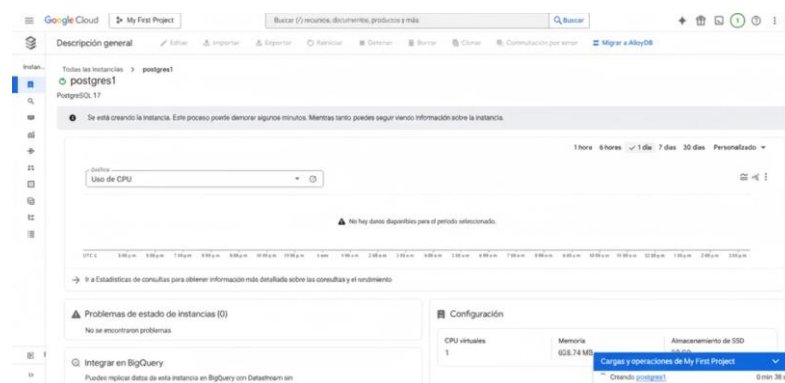
Protégete contra la eliminación accidental y la pérdida de datos. [Más información](#)



- ☐ Impedir la eliminación de instancias  
Impide la eliminación accidental o no autorizada de esta instancia. Inhabilita este parámetro de configuración antes de intentar borrarla
- ☐ Retener copias de seguridad después de la eliminación de la instancia  
Las copias de seguridad automáticas se conservan según tu configuración, mientras que las copias de seguridad a demanda se conservan hasta que se borran de forma manual. El almacenamiento se factura según el uso.
- ☐ Copia de seguridad final en la eliminación de la instancia  
Las copias de seguridad finales se crearán automáticamente durante la eliminación de la instancia. La copia de seguridad final se almacena durante 30 días después de la eliminación de forma predeterminada.

*Ilustración 12. Configuración de eliminación*

- Creación de instancia: Una vez revisados todos los parámetros, hacer clic en Crear instancia.
- Tiempo de aprovisionamiento: El proceso de despliegue tarda aproximadamente 15 minutos, durante los cuales se habilitan los recursos y configuraciones seleccionadas.



*Ilustración 13. Habilitación de recursos*



### Configuración de Redes Autorizadas

En el menú lateral de la instancia, ingresar a la opción Conexiones y establecer las redes que tendrán acceso:

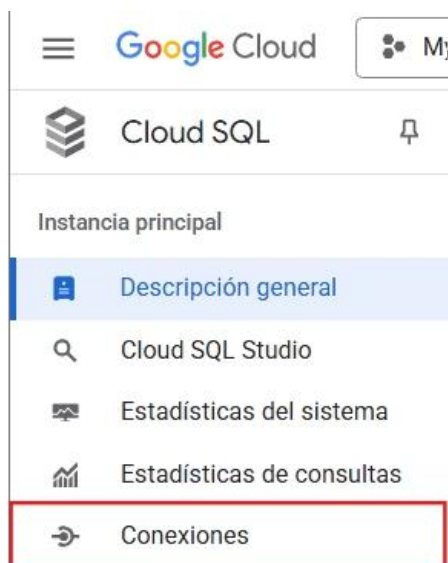


Ilustración 14. Configuración de redes

- Redes autorizadas: Definir rangos CIDR específicos para controlar qué direcciones IP pueden conectarse.
- Buenas prácticas: Evitar el uso de 0.0.0.0/0 en entornos de producción, ya que expone la base de datos a cualquier origen.
- Seguridad avanzada: Implementar el proxy de Cloud SQL como mecanismo recomendado para conexiones seguras y gestionadas.

#### Nueva red

Nombre

RedBDD

Rango de IP \*

0.0.0.0/0

Usa la notación CIDR. Ejemplo: 199.27.25.0/24

Si permites 0.0.0.0/0, se abrirá tu instancia a Internet. Esta no es una práctica recomendada de seguridad. Usa rangos de IP más reducidos o el proxy de autenticación de Cloud SQL para mejorar la seguridad. Si debes permitir 0.0.0.0/0, asegúrate de que las contraseñas sean seguras. [Más información](#)

☒ Reconozco los riesgos

Listo

Ilustración 15. Nueva red





## Conexión desde pgAdmin

Para establecer conexión con la instancia de Cloud SQL (PostgreSQL) utilizando pgAdmin, seguir los pasos:

- Abrir pgAdmin en el equipo cliente.
- En el panel lateral, seleccionar Add New Server.
- En la pestaña General, asignar un nombre descriptivo al servidor.
- En la pestaña Connection, ingresar los parámetros de acceso:
  - Host: Dirección IP pública de la instancia de Cloud SQL.
  - Port: Puerto de conexión (por defecto, 5432).
  - Username: Usuario administrador definido en la creación de la instancia.
  - Password: Contraseña correspondiente al usuario.
- Habilitar la opción de SSL y cargar los certificados requeridos (cliente y CA), previamente descargados desde la consola de Cloud SQL.

Register - Server

General Connection Parameters SSH Tunnel Advanced Post Connection SQL Tags

Host name/address

136.119.137.39

Port

5432

Maintenance database

postgres

Username

postgres

Kerberos authentication?

☐

Password

\*\*\*\*\*

Close Reset Save

Ilustración 16. Conexión de postgresql (pgadmin)

Verificación de conexión a la instancia en Google Cloud.

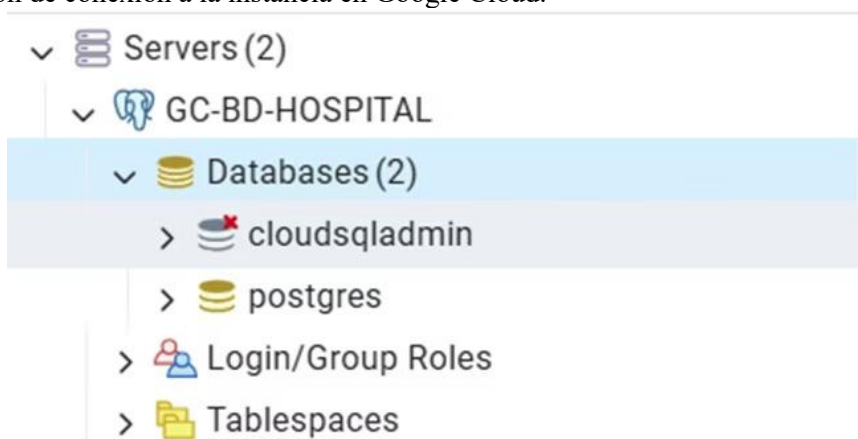


Ilustración 17. Verificación de instancia



### Ejercicio 1: Multi-AZ y Failover

Objetivo: Comprender el funcionamiento de la replicación síncrona y los mecanismos de conmutación por error (failover) en Cloud SQL.

Instrucciones:

- Acceder a la instancia previamente creada en Cloud SQL con configuración de alta disponibilidad (Multi-AZ).
- Abrir el cliente de administración ( pgAdmin ).

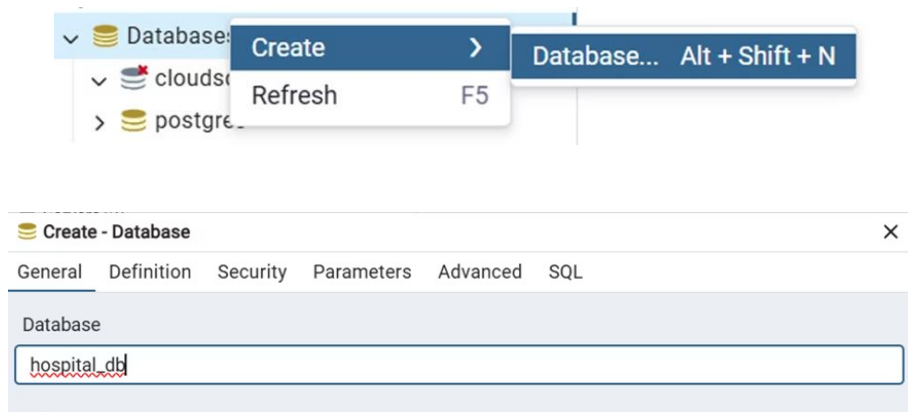


Ilustración 18. Creación de BD

### Modelo de Base de Datos

El modelo de datos está compuesto por tres entidades principales: medico, paciente y consulta. Estas tablas representan la estructura lógica de un sistema hospitalario básico, permitiendo registrar profesionales médicos, pacientes atendidos y las consultas realizadas.



```
hospital_db/postgres@GC-BD-HOSPITAL* ×
hospital_db/postgres@GC-BD-HOSPITAL
No limit
Query Query History
1  -- Tabla médicos
2  CREATE TABLE medico (
3      id SERIAL PRIMARY KEY,
4      nombre VARCHAR(100) NOT NULL,
5      especialidad VARCHAR(50),
6      telefono VARCHAR(20),
7      fecha_registro TIMESTAMP DEFAULT NOW()
8  );
9
10 -- Tabla pacientes
11 CREATE TABLE paciente (
12     id SERIAL PRIMARY KEY,
13     nombre VARCHAR(100) NOT NULL,
14     fecha_nacimiento DATE,
15     telefono VARCHAR(20),
16     direccion VARCHAR(200),
17     fecha_registro TIMESTAMP DEFAULT NOW()
18 );
19
20 -- Tabla consultas
21 CREATE TABLE consulta (
22     id SERIAL PRIMARY KEY,
23     medico_id INTEGER REFERENCES medico(id),
24     paciente_id INTEGER REFERENCES paciente(id),
25     fecha_consulta TIMESTAMP DEFAULT NOW(),
26     diagnostico TEXT,
27     tratamiento TEXT,
28     costo DECIMAL(10,2)
29 );
```

*Ilustración 19. Creación de tablas*

### *Sql*

-- Creación de tabla: médicos

```
CREATE TABLE medico (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    especialidad VARCHAR(50),
    telefono VARCHAR(20),
    fecha_registro TIMESTAMP DEFAULT NOW()
);
```

-- Creación de tabla: pacientes

```
CREATE TABLE paciente (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha_nacimiento DATE,
    telefono VARCHAR(20),
    direccion VARCHAR(200),
    fecha_registro TIMESTAMP DEFAULT NOW()
);
```



);

-- Creación de tabla: consultas médicas

```
CREATE TABLE consulta (  
    id SERIAL PRIMARY KEY,  
    medico_id INTEGER REFERENCES medico(id),  
    paciente_id INTEGER REFERENCES paciente(id),  
    fecha_consulta TIMESTAMP DEFAULT NOW(),  
    diagnostico TEXT,  
    tratamiento TEXT,  
    costo DECIMAL(10,2)  
);
```

### Verificación de configuración en modo de alta disponibilidad

La verificación de parámetros de replicación y conmutación por error confirma que la instancia opera bajo un esquema de alta disponibilidad. Se consulta el catálogo del sistema de PostgreSQL para validar los ajustes críticos relacionados con WAL, replicación y standby.

```
1  SELECT name, setting  
2  FROM pg_settings  
3  WHERE name IN (  
4      'synchronous_commit',  
5      'wal_level',  
6      'max_wal_senders',  
7      'hot_standby'  
8  );
```

	name text	setting text
1	hot_standby	on
2	max_wal_senders	10
3	synchronous_commit	on
4	wal_level	replica

*Ilustración 20. Modo alta disponibilidad*

*Sql*

```
SELECT name, setting  
FROM pg_settings  
WHERE name IN (  
    'synchronous_commit',  
    'wal_level',  
    'max_wal_senders',  
    'hot_standby'  
);
```

### Replicación Síncrona y Monitoreo en Cloud SQL (PostgreSQL)

La replicación síncrona es un componente esencial en arquitecturas de alta disponibilidad (HA), como las implementadas en entornos Multi-AZ. Este mecanismo asegura que cada transacción confirmada en la instancia primaria también se haya escrito en al menos una réplica sincronizada antes de finalizar



La replicación síncrona en acción: cada INSERT tarda 5-10ms extra esperando confirmación del standby.

```
1  -- 2. Crear tabla de prueba con timestamps precisos
2  CREATE TABLE prueba_replicacion (
3      id SERIAL PRIMARY KEY,
4      accion VARCHAR(100),
5      timestamp_escritura TIMESTAMP(6) DEFAULT CLOCK_TIMESTAMP(),
6      zona_escritura VARCHAR(50)
7  );
8
9  -- 3. Insertar datos y medir el tiempo
10 -- Esto se replica SÍNCRONAMENTE a la standby
11 INSERT INTO prueba_replicacion (accion, zona_escritura)
12 VALUES ('Escritura en Primary', current_setting('cluster_name'));
13
14 -- 4. Ver el WAL (Write-Ahead Log) que se está replicando
15 SELECT
16     pg_current_wal_lsn() as posicion_actual,
17     pg_wal_lsn_diff(pg_current_wal_lsn(), '0/0') / 1024 / 1024 as mb_escritos;
18
19 -- 5. Ver lag de replicación (debe ser casi 0)
20 SELECT
21     CASE
22         WHEN pg_last_wal_receive_lsn() = pg_last_wal_replay_lsn()
23         THEN 'Replicación al día'
24         ELSE 'Replicación con retraso'
25     END as estado_replicacion,
26     pg_wal_lsn_diff(pg_last_wal_receive_lsn(), pg_last_wal_replay_lsn()) as bytes_pendientes;
```

Data Output Messages Notifications

	estado_replicacion	bytes_pendientes
	text	numeric
1	Replicación con retraso	[null]

*Ilustración 21. Replicación síncrona*

## Sql

```
CREATE TABLE prueba_replicacion (  
    id SERIAL PRIMARY KEY,  
    accion VARCHAR(100),  
    timestamp_escritura TIMESTAMP(6) DEFAULT CLOCK_TIMESTAMP(),  
    zona_escritura VARCHAR(50)  
);
```

```
INSERT INTO prueba_replicacion (accion, zona_escritura)  
VALUES ('Escritura en Primary', current_setting('cluster_name'));
```

```
SELECT  
    pg_current_wal_lsn() AS posicion_actual,  
    pg_wal_lsn_diff(pg_current_wal_lsn(), '0/0') / 1024 / 1024 AS  
mb_escritos;
```

```
SELECT  
    CASE  
        WHEN pg_last_wal_receive_lsn() = pg_last_wal_replay_lsn()  
        THEN 'Replicación al día'  
        ELSE 'Replicación con retraso'  
    END AS estado_replicacion,  
    pg_wal_lsn_diff(pg_last_wal_receive_lsn(),  
pg_last_wal_replay_lsn()) AS bytes_pendientes;
```





### Simulación de Carga y Comportamiento en Arquitectura Multi-AZ

La siguiente prueba permite observar el impacto de la replicación síncrona en una instancia de Cloud SQL configurada con alta disponibilidad (Multi-AZ). Se utiliza una función en PL/pgSQL que simula múltiples inserciones en la tabla, midiendo el tiempo de escritura por cada operación.

```
1  -- 6. Crear procedimiento que simula carga hospitalaria
2  CREATE OR REPLACE FUNCTION simular_consultas(num_consultas INT)
3  RETURNS TABLE(consulta_id INT, tiempo_ms NUMERIC) AS $$
4  DECLARE
5      inicio TIMESTAMP;
6      fin TIMESTAMP;
7      nuevo_id INT;
8  BEGIN
9      FOR i IN 1..num_consultas LOOP
10         inicio := CLOCK_TIMESTAMP();
11
12         INSERT INTO consulta (medico_id, paciente_id, diagnostico, tratamiento, costo)
13         VALUES (
14             (RANDOM() * 2 + 1)::INT, -- Médico aleatorio 1-3
15             (RANDOM() * 2 + 1)::INT, -- Paciente aleatorio 1-3
16             'Consulta de prueba ' || i,
17             'Tratamiento ' || i,
18             (RANDOM() * 100 + 20)::NUMERIC(10,2)
19         )
20         RETURNING id INTO nuevo_id;
21
22         fin := CLOCK_TIMESTAMP();
23
24         consulta_id := nuevo_id;
25         tiempo_ms := EXTRACT(EPOCH FROM (fin - inicio)) * 1000;
26         RETURN NEXT;
27     END LOOP;
28 END;
29 $$ LANGUAGE plpgsql;
30
31 -- 7. Ejecutar simulación (verás que cada escritura espera la confirmación del standby)
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 174 msec.

*Ilustración 22. Simulación de carga*

#### Sql

```
CREATE OR REPLACE FUNCTION simular_consultas(num_consultas INT)
RETURNS TABLE(consulta_id INT, tiempo_ms NUMERIC) AS $$
DECLARE
    inicio TIMESTAMP;
    fin TIMESTAMP;
    nuevo_id INT;
BEGIN
    FOR i IN 1..num_consultas LOOP
        inicio := CLOCK_TIMESTAMP();

        INSERT INTO consulta (medico_id, paciente_id, diagnostico,
tratamiento, costo)
        VALUES (
            (RANDOM() * 2 + 1)::INT, -- Médico aleatorio 1-3
            (RANDOM() * 2 + 1)::INT, -- Paciente aleatorio 1-3
            'Consulta de prueba ' || i,
            'Tratamiento ' || i,
            (RANDOM() * 100 + 20)::NUMERIC(10,2)
        )
        RETURNING id INTO nuevo_id;

        fin := CLOCK_TIMESTAMP();
```



```
consulta_id := nuevo_id;
tiempo_ms := EXTRACT(EPOCH FROM (fin - inicio)) * 1000;
RETURN NEXT;
END LOOP;
END;
$$ LANGUAGE plpgsql;
```

### Ejecución y Observación

Cada INSERT tarda un poco más (~5-10ms extra) porque espera confirmación del standby.

**Sql**

```
-- Ejecutar simulación con 10 consultas
SELECT * FROM simular_consultas(10);
```

	consulta_id integer	tiempo_ms numeric
1	4	1.603000
2	5	0.178000
3	6	0.125000
4	7	0.141000
5	8	0.105000
6	9	0.122000
7	10	0.032000
8	11	0.023000
9	12	0.022000
10	13	0.020000

*Ilustración 23. Confirmación de standby*

Cada fila retornada incluye el id de la consulta insertada y el tiempo en milisegundos que tomó la operación. En entornos Multi-AZ, se espera que cada INSERT incurra en una latencia adicional de 5 a 10 ms, debido a la espera de confirmación del standby

### Simulación de Failover en Arquitectura Multi-AZ

La conmutación por error (failover) es el proceso mediante el cual una instancia secundaria toma el control cuando la primaria deja de estar disponible. En Cloud SQL con alta disponibilidad, este proceso es automático y transparente, pero puede ser monitoreado y probado manualmente para fines académicos o de validación.



The screenshot shows a SQL query editor with the following code:

```
1 -- 8. ANTES del failover - Insertar registro marcado
2 INSERT INTO prueba_replicacion (accion, zona_escritura)
3 VALUES ('ANTES del Failover', inet_server_addr()::TEXT);
4
5 -- Anota la hora exacta
6 SELECT NOW(), pg_current_wal_lsn();
```

Below the query, the 'Data Output' tab is active, showing a single row of results:

	now timestamp with time zone	pg_current_wal_lsn pg_lsn
1	2025-11-04 20:57:47.15567+00	0/5104F50

*Ilustración 24. Simulación de failover*

Este registro sirve como punto de control para verificar que la réplica recibió y aplicó la transacción antes del evento de failover.

### Sql

```
-- Insertar registro marcado antes del failover
INSERT INTO prueba_replicacion (accion, zona_escritura)
VALUES ('ANTES del Failover', inet_server_addr()::TEXT);

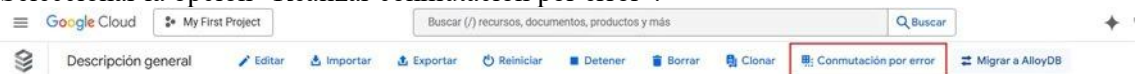
-- Registrar hora exacta y posición del WAL
SELECT NOW(), pg_current_wal_lsn();
```

### Conmutación por Error Manual (Failover) en Cloud SQL

La conmutación por error manual permite validar el comportamiento de la arquitectura Multi-AZ en condiciones controladas. Este procedimiento fuerza el traspaso

Acceder a la consola de Google Cloud Platform.

1. Ingresar al menú de la instancia configurada con HA.
2. Seleccionar la opción “Realizar conmutación por error”.



*Ilustración 25. Conmutación por error*

3. En la ventana emergente, ingresar el ID de la instancia (por ejemplo, postgres1).



4. Confirmar la operación haciendo clic en Activar conmutación por error.

#### Activar una conmutación por error de forma manual

⚠ Esta operación no se puede deshacer.

Mientras se realice la conmutación por error, tu instancia no estará disponible durante unos minutos.

Para confirmar la conmutación por error, ingresa el ID de la instancia **postgres1** a continuación:

postgres1 \*

Cancelar

Activar conmutación por error

*Ilustración 26. Activar comunicación por error*

Durante el proceso, la instancia no estará disponible por aproximadamente 60 segundos. Esta operación no se puede deshacer.

### Verificación Post-Failover en Arquitectura Multi-AZ

Tras ejecutar una conmutación por error manual en Cloud SQL, se realiza una verificación estructurada para validar la integridad de datos, continuidad operativa y comportamiento de la infraestructura.

#### Evidencia de Persistencia

- No hay pérdida de datos: Todos los registros insertados antes del failover, incluyendo los marcados como ANTES del Failover, están presentes en la instancia activa posterior al evento.
- Esto confirma que la replicación síncrona funcionó correctamente, asegurando que el standby recibió y aplicó los WAL antes de asumir el rol primario.

```
1  -- 9. DESPUÉS del failover - Reconecta y verifica
2  -- Deberías reconectarte automáticamente
3
4  INSERT INTO prueba_replicacion (accion, zona_escritura)
5  VALUES ('DESPUÉS del Failover', inet_server_addr()::TEXT);
6
7  -- 10. Ver la evidencia completa
8  SELECT
9      id,
10     accion,
11     timestamp_escritura,
12     zona_escritura,
13     timestamp_escritura - LAG(timestamp_escritura) OVER (ORDER BY id) as tiempo_entre_escrituras
14  FROM prueba_replicacion
15  ORDER BY id DESC
16  LIMIT 10;
```

	id [PK] integer	accion character varying (100)	timestamp_escritura timestamp without time zone (6)	zona_escritura character varying (50)	tiempo_entre_escrituras interval
1	35	DESPUÉS del Failover	2025-11-04 21:04:20.067301	136.119.137.39/32	00:06:32.910907
2	2	ANTES del Failover	2025-11-04 20:57:47.156394	136.119.137.39/32	00:04:21.10275
3	1	Escritura en Primary	2025-11-04 20:53:26.053644		[null]

*Ilustración 27. Evidencia de persistencia*



### **Cambio de Zona**

- IP o zona activa: Puede variar tras el failover, dependiendo de cómo Google Cloud redistribuye la instancia.
- El campo zona\_escritura muestra la IP activa en cada momento, permitiendo rastrear el nodo que ejecutó la escritura.

*sql*

```
SELECT inet_server_addr(); -- Verifica la IP activa post-failover
```

### **Tiempo de Interrupción**

- El análisis de timestamp\_escritura y la función LAG() permite calcular el tiempo entre escrituras.
- Se observa un gap de aproximadamente 60 segundos, correspondiente al tiempo de indisponibilidad durante el failover.

*sql*

```
SELECT
    id,
    accion,
    timestamp_escritura,
    zona_escritura,
    timestamp_escritura - LAG(timestamp_escritura) OVER (ORDER BY id)
AS tiempo_entre_escrituras
FROM prueba_replicacion
ORDER BY id DESC
LIMIT 10;
```

- Tras el failover, la instancia responde normalmente a nuevas operaciones INSERT, SELECT, etc.
- La función de simulación y los registros posteriores (DESPUÉS del Failover) confirman que el sistema recupera su estado operativo sin intervención adicional.

### **Ejercicio 2: Réplicas de Lectura para Escalado de Consultas**

Este ejercicio demuestra cómo utilizar read replicas en Cloud SQL para distribuir la carga de consultas y mejorar el rendimiento en entornos con alta demanda de lectura, como sistemas hospitalarios.





```
1  -- EN LA INSTANCIA PRINCIPAL (Primary):
2
3  -- 1. Crear función que simula escrituras constantes
4  CREATE OR REPLACE FUNCTION registro_consultas_continuo()
5  RETURNS VOID AS $$
6  BEGIN
7      -- Simular registro de consultas cada segundo
8      INSERT INTO consulta (medico_id, paciente_id, diagnostico, tratamiento, costo)
9      SELECT
10         (RANDOM() * 2 + 1)::INT,
11         (RANDOM() * 2 + 1)::INT,
12         'Consulta automatizada',
13         'Tratamiento estándar',
14         (RANDOM() * 100 + 20)::NUMERIC(10,2)
15     FROM generate_series(1, 5);
16 END;
17 $$ LANGUAGE plpgsql;
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 195 msec.

*Ilustración 28. Replicas por escalado*

### Sql

```
-- 1. Crear función que simula escrituras constantes
CREATE OR REPLACE FUNCTION registro_consultas_continuo()
RETURNS VOID AS $$
BEGIN
    -- Simular registro de 5 consultas por ejecución
    INSERT INTO consulta (medico_id, paciente_id, diagnostico,
tratamiento, costo)
    SELECT
        (RANDOM() * 2 + 1)::INT,
        (RANDOM() * 2 + 1)::INT,
        'Consulta automatizada',
        'Tratamiento estándar',
        (RANDOM() * 100 + 20)::NUMERIC(10,2)
    FROM generate_series(1, 5);
END;
$$ LANGUAGE plpgsql;

-- 2. Ejecutar la función para insertar registros
SELECT registro_consultas_continuo();

-- 3. Verificar que hay datos en la tabla
SELECT COUNT(*) FROM consulta;
```

### Validación desde la Réplica de Lectura

Una vez que la réplica está sincronizada:

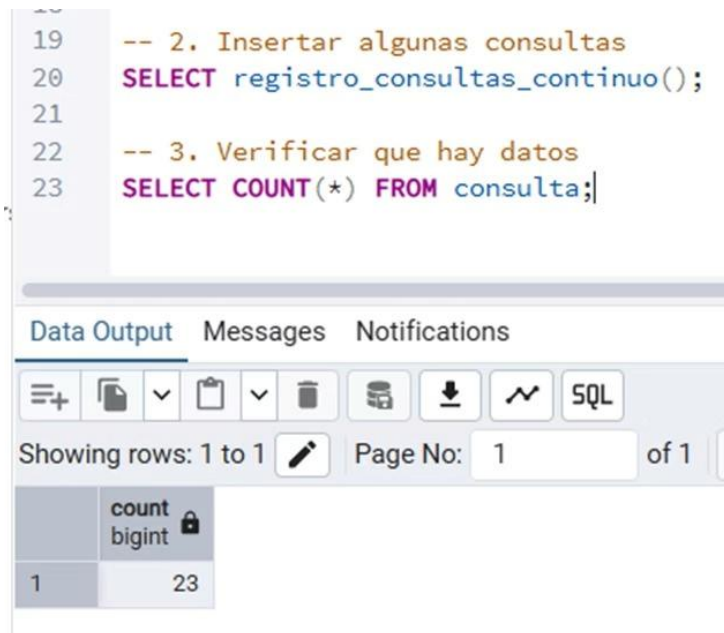


Ilustración 29. Validación desde la réplica de lectura

### Sql

**SELECT COUNT(\*) FROM consulta;**

- Si el conteo coincide con la instancia primaria, se confirma que la réplica recibe los datos correctamente.
- Las réplicas de lectura permiten ejecutar consultas intensivas (SELECT, JOIN, AGGREGATE) sin afectar el rendimiento de escritura de la instancia principal.

### Creación de Réplica de Lectura desde la Consola de Cloud SQL

La creación de réplicas de lectura permite escalar horizontalmente las operaciones de consulta, mejorar la disponibilidad y distribuir la carga entre múltiples nodos. En Cloud SQL, este proceso puede realizarse fácilmente desde la interfaz gráfica.

1. Acceder a la consola de **Google Cloud Platform**.
2. Seleccionar la instancia principal (por ejemplo, postgres1).
3. En el menú lateral, ingresar a la sección “**Réplicas**”.

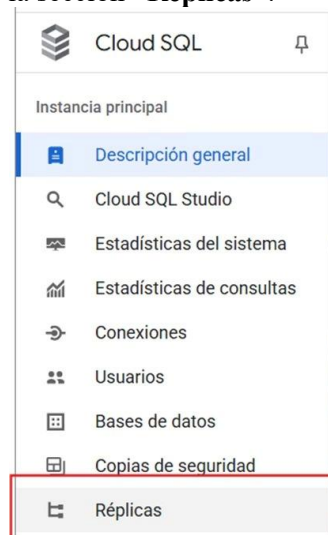


Ilustración 30. Sección réplicas



4. Hacer clic en el botón “Crear réplica”.

Todas las instancias > postgres1

✓ postgres1

PostgreSQL 17

Crea réplicas a partir de cualquier instancia principal para replicar tus datos. Para las instancias de Enterprise Plus, se puede designar una réplica de lectura entre regiones como réplica de recuperación ante desastres para minimizar el tiempo de inactividad debido a interrupciones regionales, mientras que los grupos de lectura ofrecen lecturas con alta disponibilidad y escalamiento horizontal de lectura.

También puedes crear réplicas en cascada (réplicas de réplicas) para crear jerarquías de replicación. [Más información](#)

+ Crear réplica

Nombre ↑	Tipo	Versión	Nodos de grupo de lectura
----------	------	---------	---------------------------

Esta instancia no tiene réplicas aún.

Ilustración 31. Crear replica

5. Configurar los parámetros de la réplica:

Parámetros de Configuración

Tipo de réplica

Seleccionar “Réplica de lectura” para mayor control sobre el nodo.

Alternativa: Grupo de lectura para balanceo automático y administración simplificada.

ID de instancia

Definir un nombre válido, por ejemplo: .

Requiere letras minúsculas, números y guiones, iniciando con una letra.

Información de la instancia

Versión de la base de datos  
PostgreSQL 17

ID de instancia \*  
postgres1-replica

Usa letras minúsculas, números y guiones. Comienza con una letra.

Elige el tipo de réplica

Un grupo de lectura es adecuado para simplificar las operaciones y la administración de una colección de réplicas de lectura. Una réplica de lectura es adecuada cuando quieres más control y también se usa para configurar flujos de trabajo de recuperación ante desastres. [Más información](#)

☒ Réplica de lectura

Es una copia de solo lectura de los datos de tu instancia principal que descarga el tráfico de lectura, lo que mejora el rendimiento y la disponibilidad de lectura.

☐ Grupo de lectura

Un grupo de lectura es un conjunto de uno o más nodos. Facilita las operaciones y la administración de los nodos, y balancea las cargas del tráfico.

Ilustración 32. Definición de nombre

Región

Elegir una región cercana a los servicios consumidores, por ejemplo:

Seleccionar “Varias zonas (con alta disponibilidad)” para habilitar conmutación por error automática.

Zona principal: .

Zona secundaria: cualquier otra dentro de la región.



**UNIVERSIDAD TÉCNICA DE AMBATO**  
**FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL**  
**CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN**  
**CICLO ACADÉMICO: AGOSTO – ENERO 2026**



### Elige la región y la disponibilidad zonal

Para obtener un mejor rendimiento, mantén tus datos cerca de los servicios que los necesitan. La región es permanente, mientras que la zona se puede cambiar en cualquier momento.

Región

southamerica-east1 (São Paulo)

Disponibilidad zonal

☐ Zona única

En caso de interrupción del servicio, no se aplica la conmutación por error. No se recomienda esta opción para la producción.

☒ Varias zonas (con alta disponibilidad)

La conmutación por error automática se aplica a otra zona en la región que seleccionaste. Esta opción se recomienda para las instancias de producción. Aumenta el costo.

Zona principal

southamerica-east1-a

Zona secundaria

Cualquiera (diferente ...)

[^ Ocultar zonas](#)

*Ilustración 33. Elección de región*

6. Confirmar la creación y esperar el aprovisionamiento.

La réplica puede tardar varios minutos en estar disponible. Una vez activa, permite ejecutar consultas de solo lectura.

### Verificación de Réplica de Lectura en Cloud SQL

Una vez creada la réplica de lectura, se realiza una verificación técnica para confirmar su disponibilidad, conectividad y sincronización con la instancia principal.

Nombre ↑	Tipo	Versión	Nodos de grupo de lectura	Ubicación	Alta disponibilidad
✓ postgres1-replica	Réplica de lectura de PostgreSQL	PostgreSQL 17	—	southamerica-east1-a	Habilitada

### Cadena de conexión para conexión a réplica

Conectarse a esta instancia

Nombre de la conexión	project-6126b735-487b-4654-87f:southamerica-east1:postgres1-replica
Conectividad de IP privada ?	Inhabilitado
Conectividad de IP pública ?	Habilitado
Dirección IP pública	34.39.243.88
Número de puerto predeterminado de la base de datos de TCP	5432

*Ilustración 34. Cadena de conexión*

Verifica conectividad desde pgAdmin con cadena de conexión específica (IP pública 34.39.243.88:5432).



*Ilustración 35. Registro de servidor*

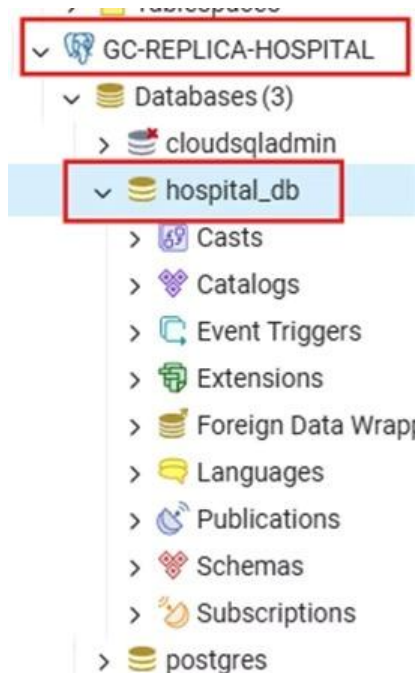


Ilustración 36. Verificación de conectividad

### Verificación de Réplica de Lectura en Cloud SQL

Una vez creada la réplica de lectura, es fundamental validar su comportamiento operativo y confirmar que cumple con las restricciones propias de una instancia de solo lectura.

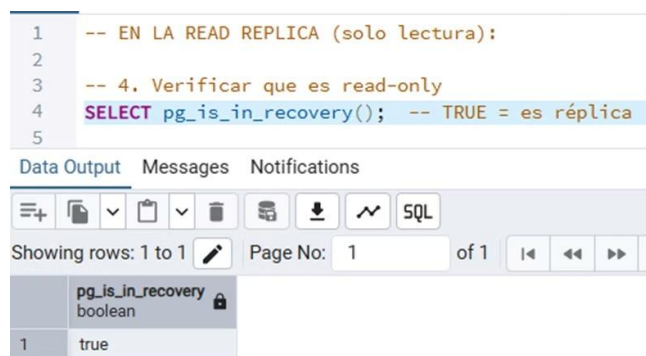


Ilustración 37. Verificación de replica

### Verificación de Disponibilidad de Datos en Réplica de Lectura

Para confirmar que la réplica de lectura está sincronizada y disponible, se ejecutan consultas SELECT sobre tablas replicadas desde la instancia principal. Esto valida que los datos se replican automáticamente y que la réplica responde correctamente a operaciones de lectura.





## Consultas de Verificación

```
11 -- 6. Queries de lectura pesadas (aquí se descarga el primary)
12 -- Reporte: Consultas por médico
13 SELECT
14     m.nombre as medico,
15     m.especialidad,
16     COUNT(c.id) as total_consultas,
17     AVG(c.costos) as costo_promedio,
18     SUM(c.costos) as ingreso_total
19 FROM medico m
20 LEFT JOIN consulta c ON m.id = c.medico_id
21 GROUP BY m.id, m.nombre, m.especialidad
22 ORDER BY total_consultas DESC;
23
```

Data Output Messages Notifications

	medico character varying (100)	especialidad character varying (50)	total_consultas bigint	costo_promedio numeric	ingreso_total numeric
1	Dr. Luis Rojas	Traumatología	9	68.9800000000000000	620.82
2	Dra. Ana Flores	Pediatría	9	58.2088888888888889	523.88
3	Dr. Carlos Méndez	Cardiología	5	45.5460000000000000	227.73

```
24 -- Reporte: Pacientes frecuentes
25 SELECT
26     p.nombre,
27     p.telefono,
28     COUNT(c.id) as num_consultas,
29     MAX(c.fecha_consulta) as ultima_consulta,
30     SUM(c.costos) as gasto_total
31 FROM paciente p
32 LEFT JOIN consulta c ON p.id = c.paciente_id
33 GROUP BY p.id, p.nombre, p.telefono
34 HAVING COUNT(c.id) > 0
35 ORDER BY num_consultas DESC;
36
```

Data Output Messages Notifications

	nombre character varying (100)	telefono character varying (20)	num_consultas bigint	ultima_consulta timestamp without time zone	gasto_total numeric
1	José Pérez	0992-333-444	13	2025-11-04 21:26:50.064806	835.27
2	Laura Sánchez	0993-555-666	5	2025-11-04 21:26:50.064806	337.05
3	María González	0991-111-222	5	2025-11-04 21:26:50.064806	200.11

Ilustración 38. Verificación de disponibilidad

### 1. Consultas por médico

**Sql**

**SELECT**

    m.nombre AS medico,  
    m.especialidad,  
    COUNT(c.id) AS total\_consultas,  
    AVG(c.costos) AS costo\_promedio,  
    SUM(c.costos) AS ingreso\_total

**FROM** medico m

**LEFT JOIN** consulta c **ON** m.id = c.m

**SELECT**

    p.nombre,  
    p.telefono,  
    COUNT(c.id) AS num\_consultas,  
    MAX(c.fecha\_consulta) AS ultima\_consulta,  
    SUM(c.costos) AS gasto\_total

**FROM** paciente p

**LEFT JOIN** consulta c **ON** p.id = c.paciente\_id



GROUP BY p.id, p.nombre, p.telefono  
HAVING COUNT(c.id) > 0  
ORDER BY num\_consultas DESC;

### Ejercicio 3: MERGE para Actualización Masiva

Escenario: Sistema externo envía actualizaciones de pacientes y nuevas consultas

Se está realizando un proceso de sincronización de datos entre una tabla temporal (staging) y la tabla principal de pacientes en una base de datos hospitalaria.

Paso 1. Preparación

Se crearon tablas temporales (staging) para pacientes y consultas

Se insertaron datos de prueba en las tablas temporales.

```
hospital_db/postgres@GC-BD-HOSPITAL
Query Query History
1  -- 1. Crear tabla staging para datos externos
2  CREATE TABLE paciente_staging (
3      id INTEGER,
4      nombre VARCHAR(100),
5      fecha_nacimiento DATE,
6      telefono VARCHAR(20),
7      direccion VARCHAR(200),
8      fecha_actualizacion TIMESTAMP DEFAULT NOW()
9  );
10
11 CREATE TABLE consulta_staging (
12     id INTEGER,
13     medico_id INTEGER,
14     paciente_id INTEGER,
15     fecha_consulta TIMESTAMP,
16     diagnostico TEXT,
17     tratamiento TEXT,
18     costo DECIMAL(10,2)
19 );
20
21 -- 2. Simular datos del sistema externo
22 INSERT INTO paciente_staging (id, nombre, telefono, direccion) VALUES
23 (1, 'María González Pérez', '0991-111-999', 'Ambato, Av. Cevallos 123'), -- Actualización
24 (2, 'José Pérez', '0992-333-444', 'Ambato, Ficoa'), -- Sin cambios
25 (4, 'Pedro Morales', '0994-777-888', 'Ambato, Huachi'); -- Nuevo paciente
26
27 INSERT INTO consulta_staging (medico_id, paciente_id, diagnostico, tratamiento, costo) VALUES
28 (1, 4, 'Chequeo general', 'Sin tratamiento', 30.00), -- Nueva consulta
29 (2, 1, 'Control pediátrico', 'Vacunas', 40.00); -- Nueva consulta
30
```

*Ilustración 39. Merge para actualización*

**Sql**

```
CREATE TABLE paciente_staging (
    id INTEGER,
    nombre VARCHAR(100),
    fecha_nacimiento DATE,
    telefono VARCHAR(20),
    direccion VARCHAR(200),
    fecha_actualizacion TIMESTAMP DEFAULT NOW()
);
```

```
CREATE TABLE consulta_staging (
    id INTEGER,
    medico_id INTEGER,
    paciente_id INTEGER,
    fecha_consulta TIMESTAMP,
    diagnostico TEXT,
    tratamiento TEXT,
    costo DECIMAL(10,2)
```



);

### Paso 2: Situación Actual (Antes del MERGE)

Antes de ejecutar el proceso de sincronización entre las tablas temporales (staging) y las tablas principales del sistema hospitalario, se realiza una verificación del estado actual de los datos en la tabla paciente. Esta inspección permite identificar qué registros serán actualizados, cuáles permanecerán sin cambios y cuáles serán insertados como nuevos.

```
--
31 -- 3. Ver estado ANTES del MERGE
32 SELECT 'ANTES DEL MERGE - Pacientes:' as estado;
33 SELECT * FROM paciente ORDER BY id;
34
```

id	nombre	fecha_nacimiento	telefono	direccion	fecha_registro
1	María González	1985-03-15	0991-111-222	Ambato, Av. Cevallos	2025-11-04 20:44:20.527109
2	José Pérez	1990-07-22	0992-333-444	Ambato, Ficoa	2025-11-04 20:44:20.527109
3	Laura Sánchez	2015-11-10	0993-555-666	Ambato, La Merced	2025-11-04 20:44:20.527109

Ilustración 40. Antes de ejecutar el proceso de sincronización

sql

```
-- Ver estado actual de la tabla paciente antes del MERGE
SELECT 'ANTES DEL MERGE - Pacientes:' as estado;
SELECT * FROM paciente ORDER BY id;
```

```
-- Ver estado actual de la tabla consulta antes del MERGE
SELECT COUNT(*) AS total_consultas FROM consulta;
```

```
35 SELECT 'ANTES DEL MERGE - Consultas:' as estado;
36 SELECT COUNT(*) as total_consultas FROM consulta;
```

total_consultas
23

Ilustración 41. Ver estado actual de la tabla consulta antes del merge

### Resultados del MERGE en Pacientes

Tras ejecutar la instrucción MERGE entre las tablas paciente\_staging y paciente, se obtienen los siguientes resultados que reflejan el impacto del proceso de sincronización:

Resumen del MERGE

- 1 actualización
- ID 1: Se actualizó el nombre, teléfono y dirección de María González, ahora registrada como María González Pérez.
- También se actualizó el campo fecha\_registro por cambio en el nombre.
- 1 inserción
- ID 4: Se insertó el nuevo paciente Pedro Morales, proveniente del sistema externo.
- 2 sin cambios
- ID 2: José Pérez no presentó diferencias; se mantuvo sin modificación.
- ID 3: Laura Sánchez no fue incluida en paciente\_staging, por lo tanto no fue afectada.



```
1  -- 4. Ejecutar MERGE para sincronizar pacientes
2  MERGE INTO paciente AS target
3  USING paciente_staging AS source
4  ON target.id = source.id
5  WHEN MATCHED THEN
6      UPDATE SET
7          nombre = source.nombre,
8          telefono = source.telefono,
9          direccion = source.direccion,
10         fecha_registro = CASE
11             WHEN target.nombre != source.nombre THEN NOW()
12             ELSE target.fecha_registro
13         END
14 WHEN NOT MATCHED THEN
15     INSERT (id, nombre, fecha_nacimiento, telefono, direccion, fecha_registro)
16     VALUES (source.id, source.nombre, source.fecha_nacimiento, source.telefono, source.direccion, NOW());
17
```

Data Output Messages Notifications

MERGE 3

Query returned successfully in 157 msec.

*Ilustración 42. Resultados del MERGE*

### Sql

```
MERGE INTO paciente AS target
USING paciente_staging AS source
ON target.id = source.id
```

-- Si el paciente ya existe, actualiza sus datos

```
WHEN MATCHED THEN
    UPDATE SET
        nombre = source.nombre,
        telefono = source.telefono,
        direccion = source.direccion,
        fecha_registro = CASE
            WHEN target.nombre != source.nombre THEN NOW()
            ELSE target.fecha_registro
        END
```

-- Si el paciente no existe, insértalo como nuevo

```
WHEN NOT MATCHED THEN
    INSERT (id, nombre, fecha_nacimiento, telefono, direccion,
    fecha_registro)
    VALUES (
        source.id,
        source.nombre,
        source.fecha_nacimiento,
        source.telefono,
        source.direccion,
        NOW());
```

### Verificación de Cambios Post-MERGE

Tras ejecutar el proceso de sincronización entre y , se realiza una verificación detallada para identificar qué registros fueron modificados, insertados o permanecieron sin cambios.



**UNIVERSIDAD TÉCNICA DE AMBATO**  
**FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL**  
**CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN**  
**CICLO ACADÉMICO: AGOSTO – ENERO 2026**



```
18 -- 5. Ver cambios
19 SELECT
20     id,
21     nombre,
22     telefono,
23     direccion,
24     fecha_registro,
25     CASE
26         WHEN fecha_registro > NOW() - INTERVAL '1 minute' THEN '✓ ACTUALIZADO'
27         ELSE '✗ Sin cambios'
28     END as estado
29 FROM paciente
30 ORDER BY id;
```

Data Output Messages Notifications

	id [PK] integer	nombre character varying (100)	telefono character varying (20)	direccion character varying (200)	fecha_registro timestamp without time zone	estado text
1	1	María González Pérez	0991-111-999	Ambato, Av. Cevallos 123	2025-11-04 22:11:09.031755	✓ ACTUALIZADO
2	2	José Pérez	0992-333-444	Ambato, Ficoa	2025-11-04 20:44:20.527109	Sin cambios
3	3	Laura Sánchez	0993-555-666	Ambato, La Merced	2025-11-04 20:44:20.527109	Sin cambios
4	4	Pedro Morales	0994-777-888	Ambato, Huachi	2025-11-04 22:11:09.031755	✓ ACTUALIZADO

Consulta de Verificación

*sql*

```
SELECT
    id,
    nombre,
    telefono,
    direccion,
    fecha_registro,
    CASE
        WHEN fecha_registro > NOW() - INTERVAL '1 minute' THEN '✓
ACTUALIZADO'
        ELSE '✗ Sin cambios'
    END AS estado
FROM paciente
ORDER BY id;
```

La verificación post-MERGE confirma que:

- Los cambios se aplicaron correctamente y de forma selectiva.
- La lógica condicional para actualizar fecha\_registro funcionó según lo esperado.
- El sistema preserva la integridad de registros no afectados.

Este paso cierra el ciclo de sincronización con trazabilidad completa, validando la eficacia del proceso en entornos hospitalarios que integran datos externos

### Resultados del MERGE en Consultas

Tras ejecutar el proceso de sincronización entre consulta\_staging y la tabla principal consulta, se aplicó una lógica de negocio que valida la existencia de médicos y pacientes antes de actualizar o insertar registros. El resultado fue exitoso y se procesaron 2 registros.

#### Actualizaciones

- Se actualizó al menos 1 consulta existente, modificando campos como diagnóstico, tratamiento y costo.

#### Inserciones

- Se insertó al menos 1 nueva consulta, correspondiente a un paciente y médico válidos que no tenían registros previos con ese id.

#### Filtrado de registros inválidos

- La cláusula WHERE p.id IS NOT NULL AND m.id IS NOT NULL garantiza que solo se procesen consultas con médicos y pacientes existentes en el sistema.





**UNIVERSIDAD TÉCNICA DE AMBATO**  
**FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL**  
**CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN**  
**CICLO ACADÉMICO: AGOSTO – ENERO 2026**



```
1  -- 6. MERGE más complejo con lógica de negocio
2  MERGE INTO consulta AS target
3  USING (
4      SELECT
5          cs.*,
6          p.id AS paciente_existe,
7          m.id AS medico_existe
8      FROM consulta_staging cs
9      LEFT JOIN paciente p ON cs.paciente_id = p.id
10     LEFT JOIN medico m ON cs.medico_id = m.id
11     WHERE p.id IS NOT NULL AND m.id IS NOT NULL -- Solo registros válidos
12 ) AS source
13 ON target.id = source.id
14 WHEN MATCHED THEN
15     UPDATE SET
16         diagnostico = source.diagnostico,
17         tratamiento = source.tratamiento,
18         costo = source.costo
19 WHEN NOT MATCHED THEN
20     INSERT (medico_id, paciente_id, fecha_consulta, diagnostico, tratamiento, costo)
21     VALUES (source.medico_id, source.paciente_id, source.fecha_consulta, source.diagnostico, source.tratamiento, source.costo);
22
```

Data Output Messages Notifications

MERGE 2

Query returned successfully in 242 msec.

### Sql

```
MERGE INTO consulta AS target
USING (
    SELECT cs.*, p.id AS paciente_existe, m.id AS medico_existe
    FROM consulta_staging cs
    LEFT JOIN paciente p ON cs.paciente_id = p.id
    LEFT JOIN medico m ON cs.medico_id = m.id
    WHERE p.id IS NOT NULL AND m.id IS NOT NULL
) AS source
ON target.id = source.id
WHEN MATCHED THEN
    UPDATE SET
        diagnostico = source.diagnostico,
        tratamiento = source.tratamiento,
        costo = source.costo
WHEN NOT MATCHED THEN
    INSERT (medico_id, paciente_id, fecha_consulta, diagnostico,
tratamiento, costo)
    VALUES (
        source.medico_id,
        source.paciente_id,
        source.fecha_consulta,
        source.diagnostico,
        source.tratamiento,
        source.costo
    );
```

Este MERGE avanzado permite:

- Sincronización segura de datos clínicos desde sistemas externos.
- Validación de integridad referencial, evitando registros huérfanos.
- Actualización eficiente de consultas existentes y inserción controlada de nuevas atenciones.



## Visualización de Merge

Visualización del resultado del MERGE aplicado a la tabla consulta, mostrando los 10 registros más recientes. Esta vista permite verificar que las nuevas consultas fueron insertadas correctamente y que los datos están sincronizados con médicos y pacientes válidos.

```
23 -- 7. Verificar resultado final
24 SELECT
25     c.id,
26     m.nombre as medico,
27     p.nombre as paciente,
28     c.diagnostico,
29     c.costo,
30     c.fecha_consulta
31 FROM consulta c
32 JOIN medico m ON c.medico_id = m.id
33 JOIN paciente p ON c.paciente_id = p.id
34 ORDER BY c.fecha_consulta DESC
35 LIMIT 10;
```

Data Output Messages Notifications

	id integer	medico character varying (100)	paciente character varying (100)	diagnostico text	costo numeric (10,2)	fecha_consulta timestamp without time zone
1	25	Dra. Ana Flores	María González Pérez	Control pediátrico	40.00	[null]
2	24	Dr. Carlos Méndez	Pedro Morales	Chequeo general	30.00	[null]
3	22	Dra. Ana Flores	José Pérez	Consulta automatizada	30.59	2025-11-04 21:26:50.064806
4	19	Dr. Luis Rojas	María González Pérez	Consulta automatizada	63.87	2025-11-04 21:26:50.064806
5	20	Dr. Carlos Méndez	Laura Sánchez	Consulta automatizada	66.33	2025-11-04 21:26:50.064806
6	21	Dr. Luis Rojas	Laura Sánchez	Consulta automatizada	86.65	2025-11-04 21:26:50.064806
7	23	Dr. Carlos Méndez	María González Pérez	Consulta automatizada	30.98	2025-11-04 21:26:50.064806
8	17	Dr. Luis Rojas	José Pérez	Consulta automatizada	61.31	2025-11-04 21:26:10.285233
9	18	Dra. Ana Flores	José Pérez	Consulta automatizada	28.81	2025-11-04 21:26:10.285233
10	15	Dr. Luis Rojas	María González Pérez	Consulta automatizada	45.21	2025-11-04 21:26:10.285233

## Sql

MERGE INTO consulta AS target

USING (

SELECT

cs.\*,

p.id AS paciente\_existe,

m.id AS medico\_existe

FROM consulta\_staging cs

LEFT JOIN paciente p ON cs.paciente\_id = p.id

LEFT JOIN medico m ON cs.medico\_id = m.id

WHERE p.id IS NOT NULL AND m.id IS NOT NULL -- Solo registros válidos

) AS source

ON target.id = source.id

-- Si la consulta ya existe, actualiza diagnóstico, tratamiento y costo  
WHEN MATCHED THEN

UPDATE SET

diagnostico = source.diagnostico,

tratamiento = source.tratamiento,

costo = source.costo

-- Si la consulta no existe, insértala como nueva

WHEN NOT MATCHED THEN

INSERT (medico\_id, paciente\_id, fecha\_consulta, diagnostico, tratamiento, costo)



```
VALUES (  
    source.medico_id,  
    source.paciente_id,  
    source.fecha_consulta,  
    source.diagnostico,  
    source.tratamiento,  
    source.costo  
);
```

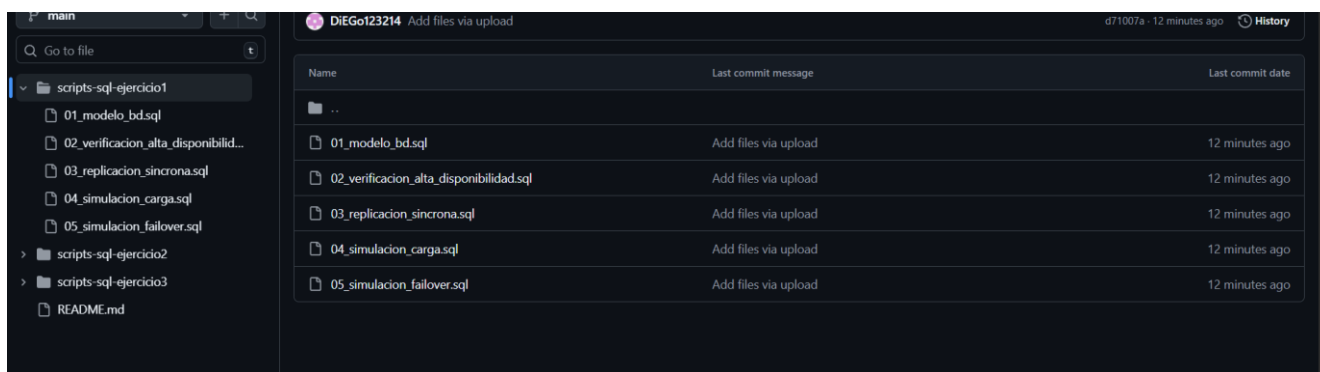
Médicos y pacientes están correctamente vinculados, lo que confirma que la lógica de negocio del MERGE filtró registros inválidos

## 2.8 Habilidades blandas empleadas en la práctica

- ☒ Liderazgo
- ☒ Trabajo en equipo
- ☐ Comunicación asertiva
- ☐ La empatía
- ☐ Pensamiento crítico
- ☐ Flexibilidad
- ☐ La resolución de conflictos
- ☐ Adaptabilidad
- ☐ Responsabilidad

- El trabajo en equipo se realizó de forma colaborativa utilizando un repositorio de GitHub

<https://github.com/DiEGo123214/APE4.-REPLICACION-DE-DATOS>



## 2.9 Conclusiones

- Cloud SQL con réplicas de lectura es una solución eficaz para escalar operaciones de consulta en entornos hospitalarios.
- El uso de MERGE con staging tables asegura una integración segura y controlada de datos externos.
- La arquitectura propuesta combina rendimiento, disponibilidad y consistencia eventual, cumpliendo con los requisitos de sistemas críticos de salud.
- El enfoque metodológico (scripts, evidencias, validaciones) garantiza reproducibilidad académica y aporta un marco sólido para futuros proyectos de integración de datos.



### **2.10 Recomendaciones**

Las réplicas de lectura deben ser monitoreadas de forma constante para asegurar que la sincronización con la instancia primaria se mantenga estable. Es recomendable ejecutar las consultas pesadas únicamente en estas réplicas, descargando así la carga de la base principal y mejorando el rendimiento general del sistema.

En los procesos de MERGE es importante mantener reglas de negocio claras que validen la existencia de médicos y pacientes antes de insertar nuevas consultas. Esto garantiza la integridad referencial y evita registros huérfanos, además de asegurar que las actualizaciones se apliquen solo cuando existan cambios reales en los datos.

Finalmente, se recomienda reforzar la seguridad mediante el uso de IP privada, cifrado en tránsito y en reposo, y asignación de permisos mínimos necesarios a los usuarios. Estas medidas, junto con pruebas de carga y documentación clara de evidencias, consolidan un sistema hospitalario robusto, escalable y confiable

### **2.11 Referencias bibliográficas**

- [1] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 7th ed. New York, NY, USA: McGraw-Hill, 2020.
- [2] J. R. Caiza, "Grupo 3 Google Cloud," Materiales de las Clases, Universidad Técnica de Ambato, Ambato, Ecuador. [Online]. Available: [https://utaedu-my.sharepoint.com/personal/jr\\_caiza\\_uta\\_edu\\_ec/\\_layouts/15/Doc.aspx?sourcedoc={94232b21-cace-40b7-a07b-33131d8940b1}&action=edit&wd=target%28Materiales%20de%20las%20Clases.one%7Cc5436bf-9456-424d-bfc4-f167013c92d3%2FGrupo%203%20Google%20Cloud%7C0ebc3462-fb9f-4c57-9f40-ae4d9247b261%2F%29&wdorigin=NavigationUrl](https://utaedu-my.sharepoint.com/personal/jr_caiza_uta_edu_ec/_layouts/15/Doc.aspx?sourcedoc={94232b21-cace-40b7-a07b-33131d8940b1}&action=edit&wd=target%28Materiales%20de%20las%20Clases.one%7Cc5436bf-9456-424d-bfc4-f167013c92d3%2FGrupo%203%20Google%20Cloud%7C0ebc3462-fb9f-4c57-9f40-ae4d9247b261%2F%29&wdorigin=NavigationUrl). [Accessed: Dec. 2, 2025].

### **2.12 Anexos**