Trabajo Final Integrador (TFI) - Bases de Datos I

Datos Generales

- **Alumnos:**

Bruno Ludueña – brunoluduena@abc.gob.ar

Pablo Mariasch – pablomariasch85@gmail.com

Pablo Grassi – pjgrassi@gmail.com

- **Carrera:** Tecnicatura Universitaria en Programación

- **Materia:** Bases de Datos I

- **Profesor/a:** Ana Eugenia Mutti – Alejandro Lasalas

- **Fecha de Entrega:** 23/10/2025

Introducción

En este trabajo práctico desarrollamos una base de datos relacional basada en un sistema de gestión de pacientes e historias clínicas.
El objetivo principal fue aplicar los conceptos aprendidos sobre modelado y administración de bases de datos, creando un sistema que permita organizar y almacenar información médica de forma segura y ordenada.

El trabajo está dividido en cinco etapas: modelado conceptual y definición de constraints, implementación y carga de datos, consultas avanzadas, seguridad e integridad, y finalmente concurrencia y transacciones.
En cada etapa fuimos incorporando nuevas herramientas y prácticas de SQL, con el propósito de construir una base de datos completa y funcional que refleje un caso real del ámbito sanitario.

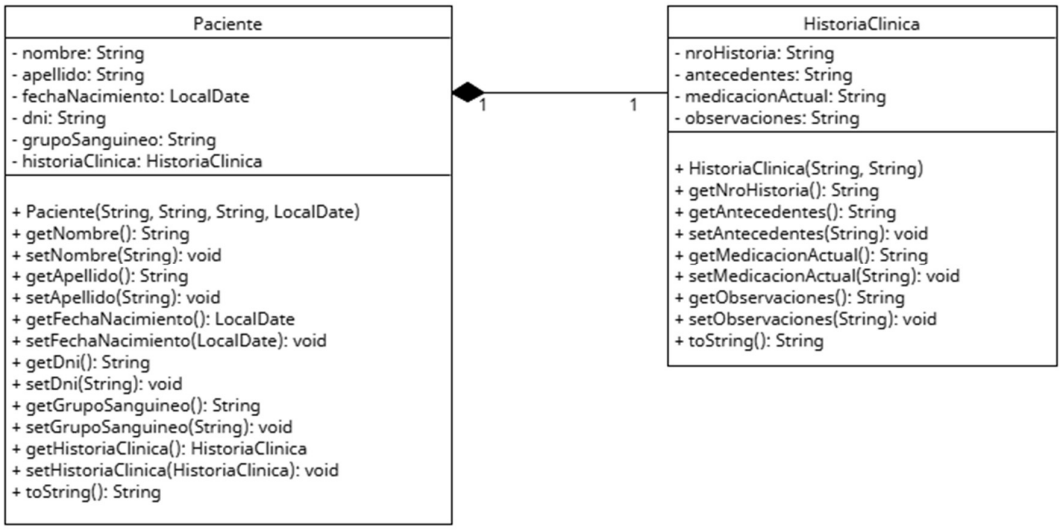
Además, promovemos el uso responsable de herramientas de Inteligencia Artificial (IA) como apoyo durante el proceso de aprendizaje. Su uso se enfoca en acompañar, revisar razonamientos y explorar distintas soluciones, sin reemplazar el trabajo crítico ni el análisis propio de los estudiantes.

Este trabajo cuenta con dos anexos y 2 videos:

- **Anexo I:** *Prompt completo para uso pedagógico de IA.*
- **Anexo II:** *Archivo con todas las consultas SQL utilizadas durante el desarrollo del proyecto.*
- **Video I:** *https://youtu.be/8EhKtRtbYh4*
- **Video II:** *https://youtu.be/it0qqw0g0fI*

Etapa 1: Modelado y Definición de Constraints.

Para el trabajo práctico integrador de Programación II se seleccionó la temática *paciente / historia clínica* como base del proyecto. A partir de ella se elaboró el siguiente modelo UML inicial, que podrá ser modificado y ampliado según los requerimientos de las etapas posteriores.



En el modelo UML inicial se definieron dos clases principales:

- Paciente, que almacena los datos personales y médicos básicos de cada persona.
- HistoriaClinica, que contiene la información relacionada con los antecedentes, la medicación actual y las observaciones médicas.

Entre ambas entidades existe una relación uno a uno (1:1), ya que cada paciente tiene una sola historia clínica asociada, y cada historia clínica pertenece a un único paciente.

Se crea la base de datos clinica53:

```
/* Se crea base de datos. */

CREATE DATABASE clinica53;
USE clinica53;
```

A partir de la relación uno a uno entre Paciente e HistoriaClinica, se arman las dos siguientes tablas en la base de datos.

Tabla: Paciente
Guarda los datos personales y médicos básicos de cada persona.

```sql
/* Definición de Esquema: Tabla Paciente */

USE clinica53;
CREATE TABLE Paciente (
    id_paciente INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NOT NULL,
    fecha_nacimiento DATE NOT NULL,
    dni VARCHAR(20) NOT NULL UNIQUE,
    grupo_sanguineo VARCHAR(5) NULL,

    CONSTRAINT chk_grupo_sanguineo CHECK (
        grupo_sanguineo IS NULL
        OR grupo_sanguineo IN ('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-')
    )
);
```

Detalles de esta tabla

- La PRIMARY KEY (id_paciente) garantiza que cada paciente tenga un identificador único.
- El campo dni tiene la restricción UNIQUE, para evitar duplicados.
- El CHECK (grupo_sanguineo) valida que solo se puedan ingresar valores válidos (por ejemplo, “A+”, “O-”, etc.) o quedar vacío si no se conoce.
- Los campos nombre, apellido y fecha_nacimiento están marcados como NOT NULL porque son datos obligatorios.

Tabla: HistoriaClinica
Almacena los antecedentes médicos, medicación y observaciones de cada paciente.

```sql
/* Definición de Esquema: Tabla HistoriaClinica */

CREATE TABLE HistoriaClinica (
    id_historia INT AUTO_INCREMENT PRIMARY KEY,
    nro_historia VARCHAR(20) NOT NULL UNIQUE,
    antecedentes TEXT NULL,
    medicacion_actual TEXT NULL,
    observaciones TEXT NULL,

    id_paciente INT NOT NULL UNIQUE,

    CONSTRAINT fk_historia_paciente
        FOREIGN KEY (id_paciente)
        REFERENCES Paciente (id_paciente)
        ON DELETE CASCADE
);
```

Detalles de esta tabla

- PRIMARY KEY (id_historia): identifica de forma única cada historia clínica.
- UNIQUE (nro_historia): evita duplicar números de historia.
- FOREIGN KEY (id_paciente): crea el enlace con la tabla *Paciente*.
- ON DELETE CASCADE: elimina automáticamente la historia clínica si se borra el paciente.
- UNIQUE (id_paciente): asegura que un paciente no pueda tener más de una historia clínica, manteniendo la relación uno a uno.

Para mantener la integridad de los datos en la tabla Paciente, creamos dos triggers que validan automáticamente la fecha de nacimiento cada vez que se agrega o modifica un registro.

- El trigger de inserción (validar_fecha_nacimiento) se ejecuta antes de guardar un nuevo paciente.
- El trigger de actualización (validar_fecha_nacimiento_update) se ejecuta antes de modificar un registro existente.

Trigger 1 – Validación antes de insertar

```
/* Triggers de Reglas de Negocio: Validación de Edad (Insert). */
DELIMITER $$

CREATE TRIGGER validar_fecha_nacimiento
BEFORE
INSERT ON Paciente
FOR EACH
ROW
BEGIN
    DECLARE edad INT;

    IF NEW.fecha_nacimiento > CURDATE() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: la
fecha de nacimiento no puede ser futura.';
    END IF;

    SET edad =
TIMESTAMPDIFF(YEAR, NEW.fecha_nacimiento, CURDATE());

    IF edad > 120 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: la
edad no puede superar los 120 años.';
    END IF;
END$$

DELIMITER ;
```

Trigger 2 – Validación antes de actualizar

```
/* Triggers de Reglas de Negocio: Validación de Edad (Update). */
DELIMITER $$

CREATE TRIGGER validar_fecha_nacimiento_update
BEFORE
UPDATE ON Paciente
FOR EACH
ROW
BEGIN
    DECLARE edad INT;

    IF NEW.fecha_nacimiento > CURDATE() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: la
fecha de nacimiento no puede ser futura.';
    END IF;

    SET edad = TIMESTAMPDIFF(YEAR,
NEW.fecha_nacimiento, CURDATE());

    IF edad > 120 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: la
edad no puede superar los 120 años.';
    END IF;
END$$

DELIMITER ;
```

Validación práctica:
Luego de crear los triggers, realizamos distintas pruebas de inserción y actualización sobre la tabla Paciente para comprobar su funcionamiento.

Casos válidos:

```
/* Scripts de Inserción: Datos de Validación */

USE clinica53;
INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dni, grupo_sanguineo)
VALUES ('Juana', 'Paz', '1988-12-15', '40123456', 'AB-');
```

```
/* Scripts de Inserción: Datos de Validación */

USE clinica53;
INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dni, grupo_sanguineo)
VALUES ('María', 'Gómez', '1985-04-12', '32145678', 'A+');
```

```
/* Scripts de Inserción: Datos de Validación */

USE clinica53;
UPDATE Paciente
SET grupo_sanguineo = 'O-'
WHERE id_paciente = 1;
```

Tipo	Texto	Duración	Filas	Resultado
SQL / User	UPDATE Paciente¶SET grupo_sanguineo = 'O-'¶WHERE id_paciente = 1	0.002s	1	Éxito
SQL / User	INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dni, grupo_sanguineo)¶VALUES ('María', 'Gómez', '1985-04-12', '32145678'...	0.002s	1	Éxito
SQL / User	INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dni, grupo_sanguineo)¶VALUES ('Juana', 'Paz', '1988-12-15', '40123456', 'A...	0.002s	1	Éxito

Casos inválidos:

```
/* Scripts de Inserción: Datos de Validación */

INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dni, grupo_sanguineo)
VALUES ('Carlos', 'Fernández', '2030-05-10', '45988777', 'B+');
```

Resultados 1

INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dn | Enter a SQL expression to filter results (use Ctrl+Space)

SQL Error [1644] [45000]: Error: la fecha de nacimiento no puede ser futura.

```
/* Scripts de Inserción: Datos de Validación */

INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dni, grupo_sanguineo)
VALUES ('Carlos', 'Fernández', '2030-05-10', '45988777', 'B+');
```

Resultados 1

INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dn | Enter a SQL expression to filter results (use Ctrl+Space)

SQL Error [1644] [45000]: Error: la fecha de nacimiento no puede ser futura.

```
/* Scripts de Inserción: Datos de Validación */

INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dni, grupo_sanguineo)
VALUES ('Roberto', 'Barto', '1980-12-15', '40123456', 'AB-');
```

Resultados 1

INSERT INTO Paciente (nombre, apellido, fecha_nacimiento, dn | Enter a SQL expression to filter results (use Ctrl+Space)

SQL Error [1062] [23000]: Duplicate entry '40123456' for key 'paciente.dni'

Etapa 2: Generación y carga de datos masivos con SQL puro

En esta etapa realizamos la carga masiva de datos utilizando SQL puro, con el objetivo de generar un conjunto grande y coherente que permita realizar pruebas de rendimiento en las siguientes etapas del trabajo.
El proceso se aplicó sobre las tablas Paciente y HistoriaClinica, manteniendo la relación uno a uno (1:1) entre ambas.

```sql
/* Carga Masiva de Datos: Población de Paciente en Dos Fases (2 x 100.000 Registros ) */
/* Primera fase */

USE clinica53;

INSERT INTO Paciente (nombre, apellido, dni, fecha_nacimiento, grupo_sanguineo)
SELECT
    CONCAT('Nombre', t1.i, t2.i, t3.i, t4.i, t5.i),
    CONCAT('Apellido', t1.i, t2.i, t3.i, t4.i, t5.i),
    LPAD(((t1.i*10000 + t2.i*1000 + t3.i*100 + t4.i*10 + t5.i)), 8, '0'),
    DATE_SUB(CURDATE(), INTERVAL FLOOR(RAND()*36500) DAY),
    ELT(FLOOR(RAND()*8)+1, 'A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-')
FROM
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t1,
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t2,
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t3,
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t4,
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t5
LIMIT 100000;
```

```sql
/* Carga Masiva de Datos: Población de Paciente en Dos Fases (2 x 100.000 Registros ) */
/* Segunda fase */

USE clinica53;

INSERT INTO Paciente (nombre, apellido, dni, fecha_nacimiento, grupo_sanguineo)
SELECT
    CONCAT('NombreB', t1.i, t2.i, t3.i, t4.i, t5.i),
    CONCAT('ApellidoB', t1.i, t2.i, t3.i, t4.i, t5.i),
    LPAD(((t1.i*10000 + t2.i*1000 + t3.i*100 + t4.i*10 + t5.i + 100000)), 8, '0'),
    DATE_SUB(CURDATE(), INTERVAL FLOOR(RAND()*36500) DAY),
    ELT(FLOOR(RAND()*8)+1, 'A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-')
FROM
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t1,
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t2,
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t3,
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t4,
    (SELECT 0 i UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
     UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) t5
LIMIT 100000;
```

Para la tabla Paciente se usaron tablas semilla creadas con subconsultas UNION ALL, que combinan los números del 0 al 9 mediante un producto cartesiano controlado.
Esto permitió generar de forma automática nombres, apellidos y DNIs distintos.
Se usaron funciones como CONCAT() para unir textos, LPAD() para completar DNIs, RAND() y DATE_SUB() para asignar fechas aleatorias, y ELT() para seleccionar grupos sanguíneos válidos.
El proceso se dividió en dos fases de 100.000 registros, alcanzando un total de 200.000 pacientes.

Luego se generaron 200.000 historias clínicas, vinculadas directamente con los pacientes mediante un INSERT … SELECT, garantizando así la integridad referencial.
Cada historia clínica quedó asociada a un paciente único gracias a la clave foránea id_paciente.

```sql
/* Carga Masiva de 200.000 Historias Clínicas */

USE clinica53;

INSERT INTO HistoriaClinica (nro_historia, antecedentes, medicacion_actual, observaciones, id_paciente)
SELECT
    CONCAT('HC', LPAD(id_paciente, 6, '0')),
    CONCAT('Antecedentes del paciente ', id_paciente),
    CONCAT('Medicación del paciente ', id_paciente),
    CONCAT('Observaciones del paciente ', id_paciente),
    id_paciente
FROM Paciente
ORDER BY id_paciente
LIMIT 200000;
```

Validación de Carga Masiva: Para validar la carga se realizaron consultas de verificación (COUNT, JOIN, DISTINCT) que confirmaron:

```sql
/* Validación de Carga Masiva: Conteo de Registros */

USE clinica53;
SELECT 'Pacientes' AS tabla, COUNT(*) AS cantidad FROM Paciente
UNION ALL
SELECT 'Historias Clínicas', COUNT(*) FROM HistoriaClinica;
```

Resultados 1

SELECT 'Pacientes' AS tabla, COUNT(*) AS cantidad FROM Paci — Enter a SQL expression to filter

Grilla

	tabla	cantidad
1	Pacientes	200.002
2	Historias Clí...	200.000

200.000 pacientes y 200.000 historias clínicas cargadas correctamente.

```sql
/* Integridad referencial */
SELECT COUNT(*) FROM HistoriaClinica WHERE id_paciente NOT IN (SELECT id_paciente FROM Paciente);
```

Resultados 1

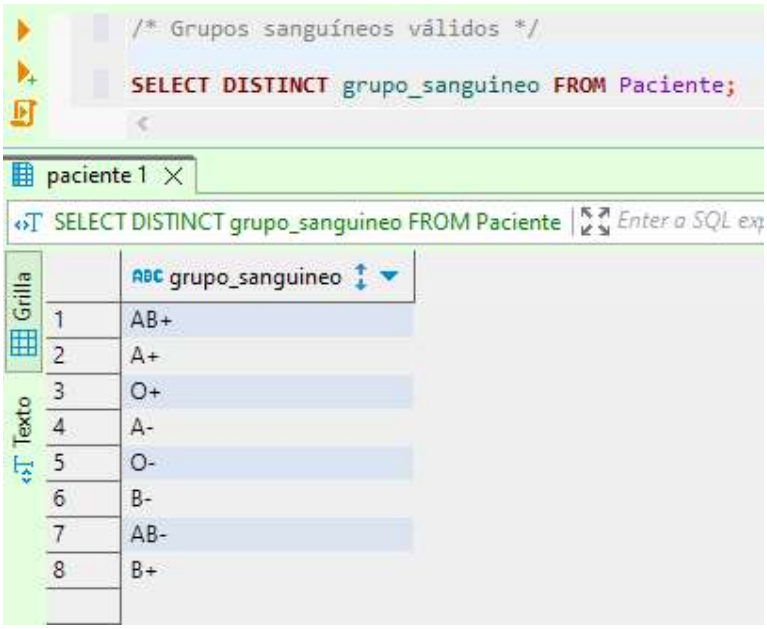
SELECT COUNT(*) FROM HistoriaClinica WHERE id_paciente N — Enter a SQL expression to filter results (use Ctrl+Space)

Grilla

	COUNT(*)
1	0

No existen registros huérfanos ni duplicados.

Cardinalidad 1:1 respetada



Los grupos sanguíneos cumplen con los valores válidos.

En conclusión, el proceso de carga masiva fue exitoso, logrando un conjunto de datos consistente, realista y listo para las pruebas de rendimiento de las próximas etapas.

Etapa 3: Consultas Avanzadas y Reportes

En esta etapa diseñamos consultas SQL más avanzadas que permiten extraer información útil del sistema y analizar el rendimiento con grandes volúmenes de datos.
El objetivo fue pasar del CRUD básico a consultas multi-tabla, agregadas y con subconsultas, aplicando los conceptos de integridad, eficiencia y utilidad práctica en un entorno realista.

Consulta 1 – Datos combinados de Pacientes e Historias Clínicas (JOIN)

```sql
USE clinica53;
SELECT
    p.id_paciente,
    CONCAT(p.nombre, ', ', p.apellido) AS nombre_completo,
    p.dni,
    p.grupo_sanguineo,
    h.nro_historia,
    h.medicacion_actual
FROM Paciente p
INNER JOIN HistoriaClinica h
ON p.id_paciente = h.id_paciente
ORDER BY p.id_paciente
LIMIT 20;
```

paciente(+) 1

SELECT p.id_paciente, CONCAT(p.nombre, ', ', p.apellido) AS nc — Enter a SQL expression to filter results (use Ctrl+Space)

Grilla | Texto

	id_paciente	nombre_completo	dni	grupo_sanguineo	nro_historia	medicacion_actual
1	1	Nombre96030, Apellido960...	00096030	A-	HC000001	Medicación del paciente 1
2	2	Nombre05030, Apellido050...	00005030	A-	HC000002	Medicación del paciente 2
3	3	Nombre15030, Apellido150...	00015030	O-	HC000003	Medicación del paciente 3
4	4	Nombre25030, Apellido250...	00025030	A+	HC000004	Medicación del paciente 4
5	5	Nombre35030, Apellido350...	00035030	A-	HC000005	Medicación del paciente 5
6	6	Nombre45030, Apellido450...	00045030	A+	HC000006	Medicación del paciente 6
7	7	Nombre55030, Apellido550...	00055030	B-	HC000007	Medicación del paciente 7
8	8	Nombre65030, Apellido650...	00065030	B+	HC000008	Medicación del paciente 8
9	9	Nombre75030, Apellido750...	00075030	O-	HC000009	Medicación del paciente 9
10	10	Nombre85030, Apellido850...	00085030	O+	HC000010	Medicación del paciente 10
11	11	Nombre95030, Apellido950...	00095030	A+	HC000011	Medicación del paciente 11
12	12	Nombre04030, Apellido040...	00004030	O-	HC000012	Medicación del paciente 12
13	13	Nombre14030, Apellido140...	00014030	AB+	HC000013	Medicación del paciente 13
14	14	Nombre24030, Apellido240...	00024030	O+	HC000014	Medicación del paciente 14
15	15	Nombre34030, Apellido340...	00034030	O-	HC000015	Medicación del paciente 15
16	16	Nombre44030, Apellido440...	00044030	B-	HC000016	Medicación del paciente 16
17	17	Nombre54030, Apellido540...	00054030	B-	HC000017	Medicación del paciente 17
18	18	Nombre64030, Apellido640...	00064030	O-	HC000018	Medicación del paciente 18
19	19	Nombre74030, Apellido740...	00074030	AB-	HC000019	Medicación del paciente 19
20	20	Nombre84030, Apellido840...	00084030	O-	HC000020	Medicación del paciente 20

Utilidad práctica:
Permite validar que la relación uno a uno entre paciente e historia clínica funcione correctamente y obtener una vista general de los datos combinados. Se usa para comprobaciones y reportes básicos del sistema.

Consulta 2 – Identificación de Pacientes Menores de 18 Años (JOIN + filtro)

```sql
USE clinica53;
SELECT
    p.id_paciente,
    CONCAT(p.nombre, ' ', p.apellido) AS nombre_completo,
    p.dni,
    TIMESTAMPDIFF(YEAR, p.fecha_nacimiento, CURDATE()) AS edad,
    p.grupo_sanguineo,
    h.nro_historia,
    h.antecedentes,
    h.medicacion_actual
FROM Paciente p
JOIN HistoriaClinica h
ON p.id_paciente = h.id_paciente
WHERE TIMESTAMPDIFF(YEAR, p.fecha_nacimiento, CURDATE()) < 18
ORDER BY edad ASC;
```

paciente(+) 1

SELECT p.id_paciente, CONCAT(p.nombre, ' ', p.apellido) AS no | Enter a SQL expression to filter results (use Ctrl+Space)

	id_paciente	nombre_completo	dni	edad	grupo_sanguineo	nro_historia	antecedentes	medicacion_actual
1	175.585	NombreB39479 ApellidoB394...	00139479	0	O+	HC175585	Antecedentes del paciente 1755...	Medicación del paciente 1755...
2	175.820	NombreB86679 ApellidoB866...	00186679	0	B+	HC175820	Antecedentes del paciente 1758...	Medicación del paciente 1758...
3	175.973	NombreB10779 ApellidoB107...	00110779	0	B-	HC175973	Antecedentes del paciente 1759...	Medicación del paciente 1759...
4	176.019	NombreB76879 ApellidoB768...	00176879	0	O-	HC176019	Antecedentes del paciente 1760...	Medicación del paciente 1760...
5	176.109	NombreB77979 ApellidoB779...	00177979	0	A+	HC176109	Antecedentes del paciente 1761...	Medicación del paciente 1761...
6	176.133	NombreB14979 ApellidoB149...	00114979	0	O-	HC176133	Antecedentes del paciente 1761...	Medicación del paciente 1761...
7	176.141	NombreB94979 ApellidoB949...	00194979	0	O+	HC176141	Antecedentes del paciente 1761...	Medicación del paciente 1761...
8	176.209	NombreB77069 ApellidoB770...	00177069	0	AB-	HC176209	Antecedentes del paciente 1762...	Medicación del paciente 1762...
9	176.392	NombreB08269 ApellidoB082...	00108269	0	O+	HC176392	Antecedentes del paciente 1763...	Medicación del paciente 1763...
10	176.393	NombreB18269 ApellidoB182...	00118269	0	AB+	HC176393	Antecedentes del paciente 1763...	Medicación del paciente 1763...
11	176.649	NombreB73469 ApellidoB734...	00173469	0	O-	HC176649	Antecedentes del paciente 1766...	Medicación del paciente 1766...
12	176.740	NombreB84569 ApellidoB845...	00184569	0	A+	HC176740	Antecedentes del paciente 1767...	Medicación del paciente 1767...
13	176.801	NombreB98669 ApellidoB986...	00198669	0	B-	HC176801	Antecedentes del paciente 1768...	Medicación del paciente 1768...
14	176.909	NombreB77769 ApellidoB777...	00177769	0	B-	HC176909	Antecedentes del paciente 1769...	Medicación del paciente 1769...
15	176.977	NombreB50769 ApellidoB507...	00150769	0	B-	HC176977	Antecedentes del paciente 1769...	Medicación del paciente 1769...
16	177.056	NombreB42869 ApellidoB428...	00142869	0	AB-	HC177056	Antecedentes del paciente 1770...	Medicación del paciente 1770...
17	177.093	NombreB18969 ApellidoB189...	00118969	0	A-	HC177093	Antecedentes del paciente 1770...	Medicación del paciente 1770...
18	177.160	NombreB82969 ApellidoB829...	00182969	0	B-	HC177160	Antecedentes del paciente 1771...	Medicación del paciente 1771...
19	177.194	NombreB28059 ApellidoB280...	00128059	0	B+	HC177194	Antecedentes del paciente 1771...	Medicación del paciente 1771...
20	177.418	NombreB66259 ApellidoB662...	00166259	0	AB+	HC177418	Antecedentes del paciente 1774...	Medicación del paciente 1774...
21	177.457	NombreB52259 ApellidoB522...	00152259	0	B+	HC177457	Antecedentes del paciente 1774...	Medicación del paciente 1774...
22	177.512	NombreB06359 ApellidoB063...	00106359	0	A-	HC177512	Antecedentes del paciente 1775...	Medicación del paciente 1775...
23	177.553	NombreB12359 ApellidoB123...	00112359	0	O+	HC177553	Antecedentes del paciente 1775...	Medicación del paciente 1775...
24	177.780	NombreB80559 ApellidoB805...	00180559	0	O-	HC177780	Antecedentes del paciente 1777...	Medicación del paciente 1777...
25	177.946	NombreB43759 ApellidoB437...	00143759	0	O-	HC177946	Antecedentes del paciente 1779...	Medicación del paciente 1779...
26	177.985	NombreB39859 ApellidoB398...	00139859	0	O-	HC177985	Antecedentes del paciente 1779...	Medicación del paciente 1779...
27	178.028	NombreB65859 ApellidoB658...	00165859	0	AB-	HC178028	Antecedentes del paciente 1780...	Medicación del paciente 1780...
28	178.044	NombreB23859 ApellidoB238...	00123859	0	A+	HC178044	Antecedentes del paciente 1780...	Medicación del paciente 1780...

Renovar | Save | Cancel | Exportar datos ... | 10000 | 35.748 | 10000 row(s) fetched - 0,866s (0,011s fetch), on 2025-10-22 at 14:

Utilidad práctica:
Identifica pacientes menores de edad para generar alertas o aplicar tratamientos diferenciados según el protocolo médico. Es una consulta típica de control en sistemas de salud. En este caso la consulta arroja que hay 35.748 pacientes menores de 18 años.

Consulta 3 – Conteo de Pacientes por Grupo Sanguíneo (GROUP BY / HAVING)

```sql
USE clinica53;
SELECT
    grupo_sanguineo,
    COUNT(*) AS cantidad_pacientes
FROM Paciente
WHERE grupo_sanguineo IS NOT NULL
GROUP BY grupo_sanguineo
HAVING COUNT(*) > 25000
ORDER BY cantidad_pacientes DESC;
```

paciente 1

SELECT grupo_sanguineo, COUNT(*) AS cantidad_pacientes FR

Grilla | Texto

	grupo_sanguineo	cantidad_pacientes
1	B+	25.265
2	O-	25.200
3	A-	25.079
4	B-	25.046
5	AB+	25.027
6	AB-	25.019

Utilidad práctica:
Permite conocer la distribución de grupos sanguíneos dentro de la base de datos, y detectar los tipos más comunes o escasos. Es útil para planificación hospitalaria o gestión de bancos de sangre.

Consulta 4 – Pacientes de 80 años o más (Subconsulta)

```sql
USE clinica53;
SELECT
    id_paciente,
    nombre,
    apellido,
    dni,
    TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE()) AS edad
FROM Paciente
WHERE id_paciente IN (
    SELECT id_paciente
    FROM Paciente
    WHERE TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE()) >= 80
)
ORDER BY edad ASC;
```

paciente 1

SELECT id_paciente, nombre, apellido, dni, TIMESTAMPDIFF(YE... Enter a SQL expression to filter results (use Ctrl+Space)

	id_paciente	nombre	apellido	dni	edad
1	33.494	Nombre45828	Apellido45828	00045828	80
2	33.963	Nombre38318	Apellido38318	00038318	80
3	34.029	Nombre92318	Apellido92318	00092318	80
4	34.525	Nombre52818	Apellido52818	00052818	80
5	34.639	Nombre91918	Apellido91918	00091918	80
6	34.685	Nombre56008	Apellido56008	00056008	80
7	34.689	Nombre96008	Apellido96008	00096008	80
8	34.719	Nombre93008	Apellido93008	00093008	80
9	34.885	Nombre56208	Apellido56208	00056208	80
10	34.946	Nombre60208	Apellido60208	00060208	80
11	34.995	Nombre55308	Apellido55308	00055308	80
12	35.071	Nombre17408	Apellido17408	00017408	80
13	35.088	Nombre86408	Apellido86408	00086408	80
14	35.129	Nombre92408	Apellido92408	00092408	80
15	35.644	Nombre40908	Apellido40908	00040908	80
16	36.876	Nombre53229	Apellido53229	00053229	80
17	37.174	Nombre33529	Apellido33529	00033529	80
18	37.194	Nombre31529	Apellido31529	00031529	80
19	37.220	Nombre99629	Apellido99629	00099629	80
20	37.362	Nombre14729	Apellido14729	00014729	80
21	37.497	Nombre61829	Apellido61829	00061829	80
22	37.512	Nombre19929	Apellido19929	00019929	80
23	37.575	Nombre43929	Apellido43929	00043929	80
24	37.634	Nombre37019	Apellido37019	00037019	80
25	37.653	Nombre25019	Apellido25019	00025019	80
26	37.672	Nombre13019	Apellido13019	00013019	80
27	37.673	Nombre23019	Apellido23019	00023019	80
28	37.875	Nombre43219	Apellido43219	00043219	80
29	37.877	Nombre63219	Apellido63219	00063219	80
30	37.996	Nombre51319	Apellido51319	00051319	80

Grilla | Texto | Record

Renovar | Save | Cancel | Exportar datos ... | 10000 | 39.741

Utilidad práctica:
Esta consulta permite identificar rápidamente a los pacientes de 80 años o más, considerados de riesgo en la mayoría de los sistemas médicos.
Es útil para generar listados de seguimiento, priorizar controles médicos o aplicar programas específicos de atención geriátrica.

Creación de Vista – Identificación de Donantes Universales Elegibles

```sql
/* Creación de Vista: Identificación de Donantes Universales Elegibles */

CREATE OR REPLACE VIEW Vista_Donantes_Universales AS
SELECT
    p.id_paciente,
    CONCAT(p.nombre, ' ', p.apellido) AS nombre_completo,
    p.dni,
    p.grupo_sanguineo,
    TIMESTAMPDIFF(YEAR, p.fecha_nacimiento, CURDATE()) AS edad
FROM Paciente p
JOIN HistoriaClinica h
    ON p.id_paciente = h.id_paciente
WHERE
    p.grupo_sanguineo = 'O-'
    AND TIMESTAMPDIFF(YEAR, p.fecha_nacimiento, CURDATE()) BETWEEN 18 AND 50;
```

```sql
/* Consulta DML: Visualización de Donantes Universales Elegibles */

USE clinica53;
SELECT * FROM vista_donantes_universales;
```

vista_donantes_universales 1

SELECT * FROM vista_donantes_universales | Enter a SQL expression to filter results (use Ctrl+Space)

	id_paciente	nombre_completo	dni	grupo_sanguineo	edad
1	12	Nombre04030 Apellido040...	00004030	O-	43
2	25	Nombre33030 Apellido330...	00033030	O-	48
3	31	Nombre93030 Apellido930...	00093030	O-	20
4	85	Nombre37130 Apellido371...	00037130	O-	49
5	134	Nombre22130 Apellido221...	00022130	O-	23
6	161	Nombre90130 Apellido901...	00090130	O-	40
7	171	Nombre99230 Apellido992...	00099230	O-	34
8	175	Nombre38230 Apellido382...	00038230	O-	28
9	216	Nombre44230 Apellido442...	00044230	O-	45
10	221	Nombre94230 Apellido942...	00094230	O-	39
11	240	Nombre82230 Apellido822...	00082230	O-	31
12	252	Nombre00230 Apellido002...	00000230	O-	43
13	283	Nombre17330 Apellido173...	00017330	O-	33
14	292	Nombre06330 Apellido063...	00006330	O-	36
15	295	Nombre36330 Apellido363...	00036330	O-	24
16	327	Nombre53330 Apellido533...	00053330	O-	50
17	329	Nombre73330 Apellido733...	00073330	O-	48
18	339	Nombre72330 Apellido723...	00072330	O-	26
19	365	Nombre39430 Apellido394...	00039430	O-	38
20	410	Nombre85430 Apellido854...	00085430	O-	48
21	424	Nombre23430 Apellido234...	00023430	O-	45
22	442	Nombre01430 Apellido014...	00001430	O-	37
23	451	Nombre91430 Apellido914...	00091430	O-	24
24	469	Nombre79530 Apellido795...	00079530	O-	28
25	482	Nombre07530 Apellido075...	00007530	O-	24
26	485	Nombre37530 Apellido375...	00037530	O-	39
27	490	Nombre87530 Apellido875...	00087530	O-	44
28	507	Nombre55530 Apellido555...	00055530	O-	39
29	519	Nombre74530 Apellido745...	00074530	O-	35
30	599	Nombre76630 Apellido766...	00076630	O-	31
31	646	Nombre41630 Apellido416...	00041630	O-	36
32	647	Nombre51630 Apellido516...	00051630	O-	30
33	655	Nombre30630 Apellido306...	00030630	O-	20
34	657	Nombre50630 Apellido506...	00050630	O-	22
35	795	Nombre36830 Apellido368...	00036830	O-	38
36	816	Nombre44830 Apellido448...	00044830	O-	33
37	859	Nombre70830 Apellido708...	00070830	O-	47

Renovar | Save | Cancel | Exportar datos ... | 10000 | 8.323

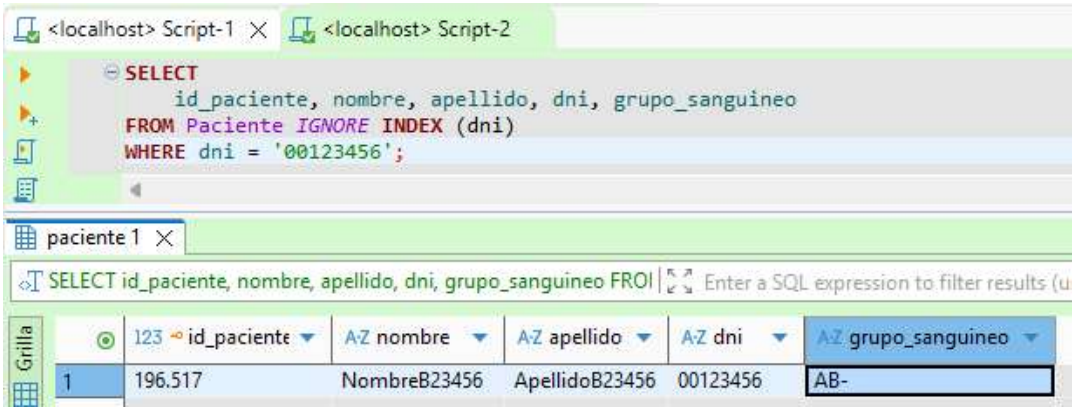
Utilidad práctica:
Crea una vista permanente para consultar rápidamente todos los donantes universales (grupo O-) en edad apta para donar sangre.

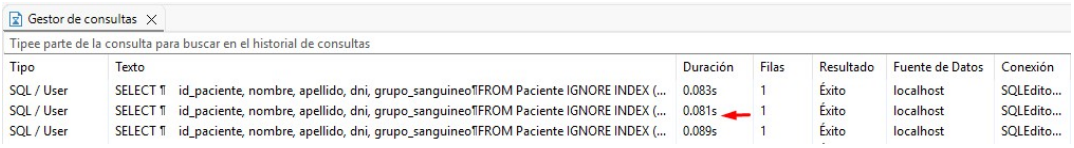
Medición comparativa con/sin índice

Para ello, se ejecutaron tres consultas representativas (igualdad, rango y JOIN) primero sin índices y luego con ellos, registrando los tiempos de respuesta y analizando las mejoras obtenidas.

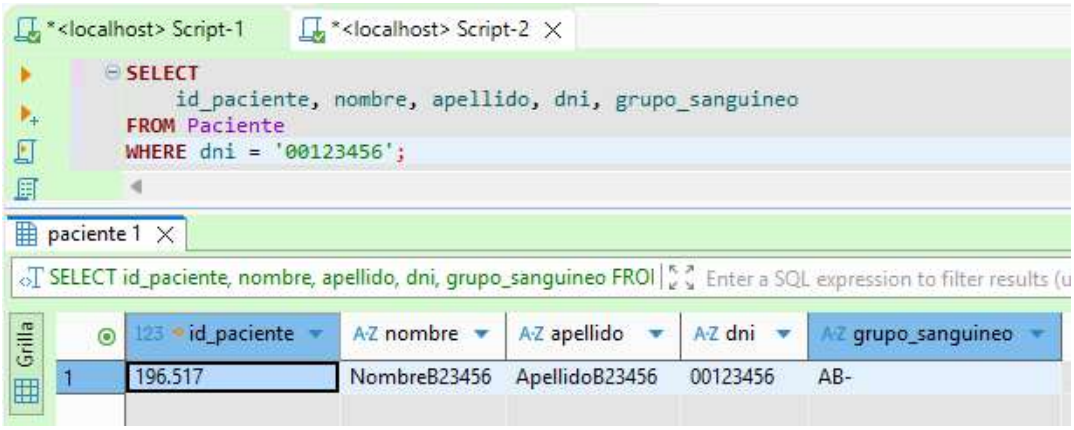
Igualdad: Busca un paciente exacto por su DNI.

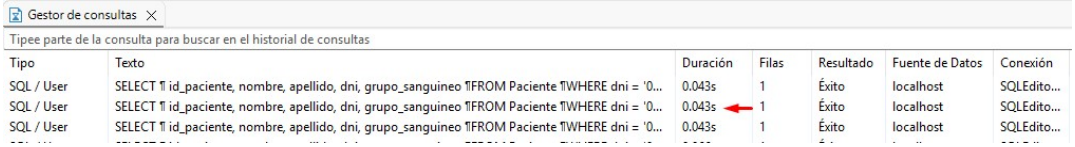
Sin índice: (dni → ya tiene índice UNIQUE, así que se ignora.)





Con índice:





Rango: Filtra pacientes mayores o iguales a 80 años.

Rango sin índice:

```sql
SELECT
    id_paciente,
    nombre,
    apellido,
    dni,
    TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE()) AS edad
FROM Paciente
WHERE id_paciente IN (
    SELECT id_paciente
    FROM Paciente
    WHERE TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE()) >= 80
)
ORDER BY edad ASC;
```

paciente 1

SELECT id_paciente, nombre, apellido, dni, TIMESTAMPDIFF(YEAR — Enter a SQL expression to filter results (use Ctrl+Space)

	id_paciente	nombre	apellido	dni	edad
1	223.396	NombreB11329	ApellidoB11329	00111329	80
2	223.508	NombreB32429	ApellidoB32429	00132429	80
3	223.708	NombreB32629	ApellidoB32629	00132629	80
4	223.780	NombreB59629	ApellidoB59629	00159629	80
5	223.968	NombreB38829	ApellidoB38829	00138829	80
6	224.263	NombreB87139	ApellidoB87139	00187139	80
7	224.415	NombreB03339	ApellidoB03339	00103339	80
8	224.463	NombreB87339	ApellidoB87339	00187339	80
9	224.487	NombreB20439	ApellidoB20439	00120439	80
10	224.583	NombreB89439	ApellidoB89439	00189439	80
11	224.622	NombreB73539	ApellidoB73539	00173539	80
12	224.702	NombreB71639	ApellidoB71639	00171639	80
13	224.825	NombreB04739	ApellidoB04739	00104739	80
14	224.838	NombreB35739	ApellidoB35739	00135739	80

Renovar | Save | Cancel | Exportar datos ... | 200000 | 39.984

Gestor de consultas

Tipee parte de la consulta para buscar en el historial de consultas

Tipo	Texto	Duración	Filas	Resultado	Fuente de Datos	Conexión
SQL / User	SELECT ¶ id_paciente,¶ nombre,¶ apellido,¶ dni,¶ TIMESTAMPDIFF(YEAR, fecha_nacimien...	3.321s	39984	Éxito	localhost	SQLEdito...
SQL / User	SELECT ¶ id_paciente,¶ nombre,¶ apellido,¶ dni,¶ TIMESTAMPDIFF(YEAR, fecha_nacimien...	3.302s	39984	Éxito	localhost	SQLEdito...
SQL / User	SELECT ¶ id_paciente,¶ nombre,¶ apellido,¶ dni,¶ TIMESTAMPDIFF(YEAR, fecha_nacimien...	3.275s	39984	Éxito	localhost	SQLEdito...

Se crea el índice para rango

*<localhost> Script-1

```sql
/* Indice para rango */
USE clinica53;
CREATE INDEX idx_paciente_fecha_nac ON Paciente(fecha_nacimiento);
```

Rango con índice:

```
SELECT
    id_paciente,
    nombre,
    apellido,
    dni,
    TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE()) AS edad
FROM Paciente
WHERE fecha_nacimiento <= DATE_SUB(CURDATE(), INTERVAL 80 YEAR)
ORDER BY edad ASC;
```

paciente 1 | paciente 2 | paciente 3

SELECT id_paciente, nombre, apellido, dni, TIMESTAMPDIFF(YEAR — Enter a SQL expression to filter results (use Ctrl+Space)

	id_paciente	nombre	apellido	dni	edad
1	323	Nombre80300	Apellido80300	00080300	80
2	328	Nombre31300	Apellido31300	00031300	80
3	452	Nombre73400	Apellido73400	00073400	80
4	754	Nombre93700	Apellido93700	00093700	80
5	1.084	Nombre96010	Apellido96010	00096010	80
6	1.217	Nombre20210	Apellido20210	00020210	80
7	1.335	Nombre02310	Apellido02310	00002310	80
8	1.472	Nombre75410	Apellido75410	00075410	80
9	1.492	Nombre77410	Apellido77410	00077410	80
10	1.812	Nombre79710	Apellido79710	00079710	80
11	1.828	Nombre31810	Apellido31810	00031810	80
12	1.935	Nombre02910	Apellido02910	00002910	80
13	2.036	Nombre12020	Apellido12020	00012020	80
14	2.124	Nombre90120	Apellido90120	00090120	80
15	2.224	Nombre90220	Apellido90220	00090220	80
16	2.226	Nombre11220	Apellido11220	00011220	80
17	2.264	Nombre94220	Apellido94220	00094220	80

Renovar | Save | Cancel | Exportar datos ... | 200000 | 39.984

Gestor de consultas

Tipee parte de la consulta para buscar en el historial de consultas

Tipo	Texto	Duración	Filas	Resultado	Fuente de Datos	Conexión
SQL / User	/* Rando con indice */¶SELECT ¶ id_paciente,¶ nombre,¶ apellido,¶ dni,¶ TIMESTAMPDIF...	0.134s	39984	Éxito	localhost	SQLEdito...
SQL / User	/* Rando con indice */¶SELECT ¶ id_paciente,¶ nombre,¶ apellido,¶ dni,¶ TIMESTAMPDIF...	0.135s	39984	Éxito	localhost	SQLEdito...
SQL / User	/* Rando con indice */¶SELECT ¶ id_paciente,¶ nombre,¶ apellido,¶ dni,¶ TIMESTAMPDIF...	0.133s	39984	Éxito	localhost	SQLEdito...

Rango con índice y EXPLAIN:

Resultados 1

EXPLAIN SELECT id_paciente, nombre, apellido, dni, TIMESTAMPC — Enter a SQL expression to filter results (use Ctrl+Space)

	id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	1	SIMPLE	Paciente	ALL	idx_paciente_fecha_nac	[NULL]	[NULL]	[NULL]	199339	Using where; Using filesort

JOIN: Evalúa la eficiencia de una unión (JOIN) entre tablas con relación 1:1.

Se crea el índice para JOIN.

```sql
/* Indice para JOIN */
USE clinica53;
CREATE INDEX idx_historia_fk ON HistoriaClinica(id_paciente);
```

JOIN sin índice:

```sql
/* JOIN sin índice*/
USE clinica53;
SELECT
    p.id_paciente,
    CONCAT(p.nombre, ', ', p.apellido) AS nombre_completo,
    p.dni,
    p.grupo_sanguineo,
    h.nro_historia,
    h.medicacion_actual
FROM Paciente p
INNER JOIN HistoriaClinica h IGNORE INDEX (idx_historia_fk)
    ON p.id_paciente = h.id_paciente
ORDER BY p.id_paciente
```

id_paciente	nombre_completo	dni	grupo_sanguineo	nro_historia	medicacion_actual
15	Nombre00000, Apellido00000	00000000	B-	HC000015	Medicación del paciente 15
16	Nombre10000, Apellido10000	00010000	B+	HC000016	Medicación del paciente 16
17	Nombre20000, Apellido20000	00020000	B+	HC000017	Medicación del paciente 17
18	Nombre30000, Apellido30000	00030000	B-	HC000018	Medicación del paciente 18
19	Nombre40000, Apellido40000	00040000	B+	HC000019	Medicación del paciente 19
20	Nombre50000, Apellido50000	00050000	A-	HC000020	Medicación del paciente 20
21	Nombre60000, Apellido60000	00060000	B+	HC000021	Medicación del paciente 21
22	Nombre70000, Apellido70000	00070000	B+	HC000022	Medicación del paciente 22
23	Nombre80000, Apellido80000	00080000	A+	HC000023	Medicación del paciente 23
24	Nombre90000, Apellido90000	00090000	B+	HC000024	Medicación del paciente 24
25	Nombre01000, Apellido01000	00001000	A+	HC000025	Medicación del paciente 25
26	Nombre11000, Apellido11000	00011000	AB-	HC000026	Medicación del paciente 26
27	Nombre21000, Apellido21000	00021000	B+	HC000027	Medicación del paciente 27
28	Nombre31000, Apellido31000	00031000	A-	HC000028	Medicación del paciente 28

Gestor de consultas

Tipee parte de la consulta para buscar en el historial de consultas

Tipo	Texto	Duración	Filas	Resultado	Fuente de Datos	Conexión
SQL / User	SELECT¶ p.id_paciente,¶ CONCAT(p.nombre, ', ', p.apellido) AS nombre_completo,¶ p.dni,¶ ...	0.596s	200000	Éxito	localhost	SQLEdito...
SQL / User	SELECT¶ p.id_paciente,¶ CONCAT(p.nombre, ', ', p.apellido) AS nombre_completo,¶ p.dni,¶ ...	0.640s	200000	Éxito	localhost	SQLEdito...
SQL / User	SELECT¶ p.id_paciente,¶ CONCAT(p.nombre, ', ', p.apellido) AS nombre_completo,¶ p.dni,¶ ...	0.605s	200000	Éxito	localhost	SQLEdito...

JOIN con índice:

```sql
/* Rando con indice */
SELECT
    id_paciente,
    nombre,
    apellido,
    dni,
    TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURDATE()) AS edad
FROM Paciente
WHERE fecha_nacimiento <= DATE_SUB(CURDATE(), INTERVAL 80 YEAR)
ORDER BY edad ASC;
```

id_paciente	nombre	apellido	dni	edad
323	Nombre80300	Apellido80300	00080300	80
328	Nombre31300	Apellido31300	00031300	80
452	Nombre73400	Apellido73400	00073400	80
754	Nombre93700	Apellido93700	00093700	80
1.084	Nombre96010	Apellido96010	00096010	80
1.217	Nombre20210	Apellido20210	00020210	80
1.335	Nombre02310	Apellido02310	00002310	80
1.472	Nombre75410	Apellido75410	00075410	80
1.492	Nombre77410	Apellido77410	00077410	80
1.812	Nombre79710	Apellido79710	00079710	80
1.828	Nombre31810	Apellido31810	00031810	80
1.935	Nombre02910	Apellido02910	00002910	80
2.036	Nombre12020	Apellido12020	00012020	80
2.124	Nombre90120	Apellido90120	00090120	80
2.224	Nombre90220	Apellido90220	00090220	80
2.226	Nombre11220	Apellido11220	00011220	80
2.264	Nombre94220	Apellido94220	00094220	80

Gestor de consultas

Tipee parte de la consulta para buscar en el historial de consultas

Tipo	Texto	Duración	Filas	Resultado	Fuente de Datos	Conexión
SQL / User	SELECT ¶ id_paciente,¶ nombre,¶ apellido,¶ dni,¶ TIMESTAMPDIFF(YEAR, fecha_nacimien...	0.128s	39984	Éxito	localhost	SQLEdito...
SQL / User	SELECT ¶ id_paciente,¶ nombre,¶ apellido,¶ dni,¶ TIMESTAMPDIFF(YEAR, fecha_nacimien...	0.132s	39984	Éxito	localhost	SQLEdito...
SQL / User	SELECT ¶ id_paciente,¶ nombre,¶ apellido,¶ dni,¶ TIMESTAMPDIFF(YEAR, fecha_nacimien...	0.132s	39984	Éxito	localhost	SQLEdito...

Conclusión de las mediciones:

Consulta por igualdad (búsqueda por DNI)
Esta consulta permite localizar rápidamente un paciente específico a partir de su número de documento.
El tiempo de ejecución se redujo de 0.085 s a 0.043 s, lo que representa una mejora del 49 %, siendo aproximadamente 2 veces más rápida gracias al índice único en el campo dni.

Consulta por rango (pacientes de 80 años o más)
Permite obtener listados de pacientes mayores o iguales a 80 años, útiles para campañas de salud o seguimiento de población de riesgo.
El índice sobre fecha_nacimiento optimizó el filtrado por edad, reduciendo el tiempo de 3.300 s a 0.135 s.
Esto equivale a una mejora del 96 %, haciendo la consulta alrededor de 24 veces más rápida.

Consulta JOIN (Pacientes ↔ Historias Clínicas)
Integra información básica del paciente con su historia clínica, generando reportes combinados con datos médicos relevantes.
La relación 1:1 entre ambas tablas se resolvió mucho más rápido al crear el índice idx_historia_fk, pasando de 0.640 s a 0.130 s.
Esto representa una mejora del 80 %, es decir, la consulta se ejecuta aproximadamente 5 veces más rápido.

Etapa 4: Seguridad e Integridad

En esta etapa se implementaron distintos mecanismos para garantizar la seguridad, integridad y confidencialidad de los datos dentro de la base de datos.

1. Creación de usuario con privilegios mínimos

```
USE clinica53;
CREATE USER IF NOT EXISTS 'app_user'@'localhost' IDENTIFIED BY '123456';

GRANT SELECT, INSERT, UPDATE
ON clinica53.paciente TO 'app_user'@'localhost';
GRANT SELECT, INSERT, UPDATE
ON clinica53.historiaclinica TO 'app_user'@'localhost';
```

Para verificar la correcta asignación de privilegios se utilizó el comando:

```
USE clinica53;
SHOW GRANTS FOR 'app_user'@'localhost';
```

Resultados 1

SHOW GRANTS FOR 'app_user'@'localhost' | Enter a SQL expression to filter results (use Ctrl+Space)

	Grants for app_user@localhost
1	GRANT USAGE ON *.* TO `app_user`@`localhost` IDENTIFIED BY PASSWORD '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9'
2	GRANT SELECT, INSERT, UPDATE ON `clinica53`.`paciente` TO `app_user`@`localhost`
3	GRANT SELECT, INSERT, UPDATE ON `clinica53`.`historiaclinica` TO `app_user`@`localhost`

Prueba de seguridad (restricción de DELETE)
Se intentó realizar una operación de eliminación con el usuario app_user:

```
/* Prueba de Seguridad y Acceso Restringido (Fallo de DELETE) */
USE clinica53;
DELETE FROM historiaclinica WHERE id = 2;
```

Resultados 1

DELETE FROM historiaclinica WHERE id = 2 | Enter a SQL expression to filter results (use Ctrl+Space)

```
SQL Error [1142] [42000]: DELETE command denied to user 'app_user'@'localhost' for
table `clinica53`.`historiaclinica`
```

Conclusión:
El intento de eliminación fue correctamente bloqueado, confirmando que el usuario tiene un acceso restringido y que la política de mínimo privilegio funciona de manera adecuada.

2. Creación de vistas para ocultar datos sensibles

Con el objetivo de proteger la información personal y cumplir con los principios de confidencialidad, se crearon dos vistas que permiten a ciertos usuarios acceder solo a datos no sensibles del sistema.

Vista 1 – Información general de pacientes (sin datos sensibles)

```sql
/* Creación Vista 1: Información general de pacientes (sin datos sensibles) */
USE clinica53;
CREATE OR REPLACE VIEW vista_pacientes_publica AS
SELECT
    id_paciente,
    CONCAT(nombre, ' ', apellido) AS nombre_completo,
    grupo_sanguineo
FROM Paciente;
```

Vista 2 – Resumen clínico sin observaciones ni medicación

```sql
/* Creación Vista 2 – Resumen clínico sin observaciones ni medicación */
CREATE OR REPLACE VIEW vista_historial_resumido AS
SELECT
    h.id_historia,
    p.id_paciente,
    CONCAT(p.nombre, ' ', p.apellido) AS nombre_completo,
    h.nro_historia,
    h.antecedentes
FROM HistoriaClinica h
JOIN Paciente p ON h.id_paciente = p.id_paciente;
```

Otorgamiento de permisos de lectura al usuario app_user

*<localhost> Script-1

```sql
-- Otorgamiento de Permisos de Lectura
USE clinica53;
GRANT SELECT ON clinica53.vista_pacientes_publica TO 'app_user'@'localhost';
GRANT SELECT ON clinica53.vista_historial_resumido TO 'app_user'@'localhost';
```

Verificar los permisos asignados

```sql
-- Verificá los permisos asignados
USE clinica53;
SHOW GRANTS FOR 'app_user'@'localhost';
```

Resultados 1

SHOW GRANTS FOR 'app_user'@'localhost' | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla | Texto

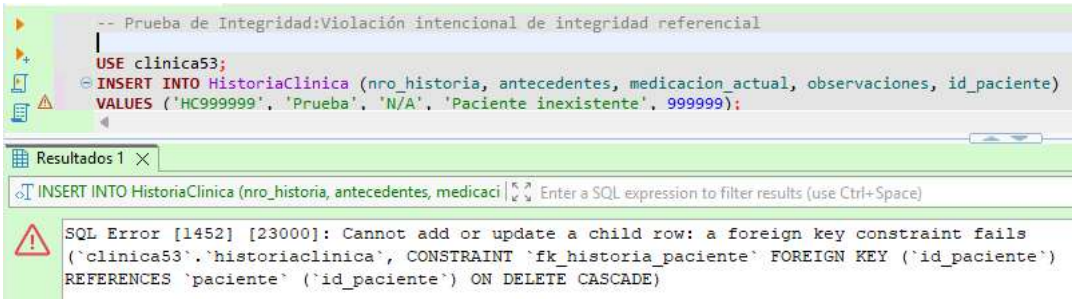
	Grants for app_user@localhost
1	GRANT USAGE ON *.* TO `app_user`@`localhost` IDENTIFIED BY PASSWORD '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9'
2	GRANT SELECT, INSERT, UPDATE ON `clinica53`.`historiaclinica` TO `app_user`@`localhost`
3	GRANT SELECT, INSERT, UPDATE ON `clinica53`.`paciente` TO `app_user`@`localhost`
4	GRANT SELECT ON `clinica53`.`vista_historial_resumido` TO `app_user`@`localhost`
5	GRANT SELECT ON `clinica53`.`vista_pacientes_publica` TO `app_user`@`localhost`

3. Pruebas de integridad documentadas

El objetivo es demostrar que las constraints y relaciones definidas en la base de datos funcionan correctamente y evitan inconsistencias.

Prueba 1 – Integridad referencial (FK válida entre Paciente e HistoriaClinica)

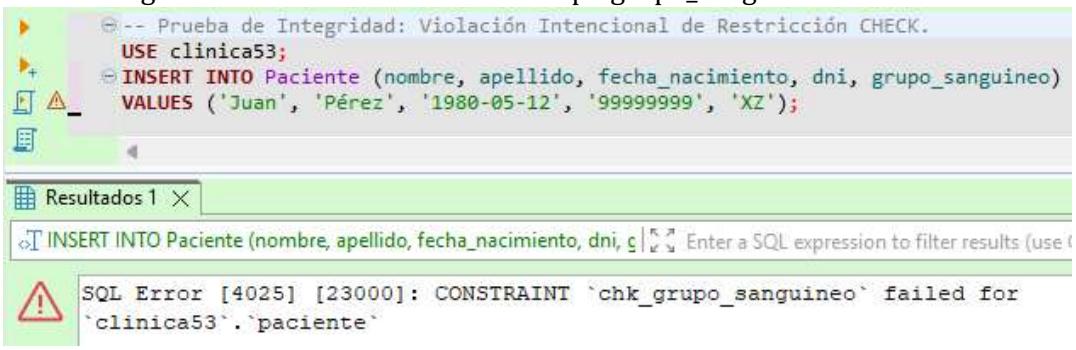
Intentar insertar una historia clínica con un id_paciente que no existe en la tabla Paciente.



Conclusión:
El sistema impide la creación de historias clínicas huérfanas.
Esto garantiza la integridad referencial, ya que cada historia clínica pertenece obligatoriamente a un paciente existente.

Prueba 2 – Integridad de dominio (CHECK de grupo sanguíneo)

Intentar ingresar un valor inválido en el campo grupo_sanguineo.



Conclusión:
La restricción CHECK impide el ingreso de valores fuera del dominio permitido, manteniendo la coherencia y validez de los datos médicos.

4. Implementación de consulta parametrizada segura

Para evitar vulnerabilidades de inyección SQL, se implementó un procedimiento almacenado que utiliza parámetros de entrada de forma segura, sin SQL dinámico. Este procedimiento devuelve los datos de un paciente a partir de su número de documento (dni).

Procedimiento almacenado

```
-- Implementación de Seguridad: Procedimiento Almacenado Anti-Inyección SQL.
DELIMITER $$

CREATE PROCEDURE sp_buscar_paciente_por_dni(IN dni_busqueda VARCHAR(20))
BEGIN
    SELECT
        id_paciente,
        CONCAT(nombre, ' ', apellido) AS nombre_completo,
        fecha_nacimiento,
        grupo_sanguineo
    FROM Paciente
    WHERE dni = dni_busqueda;
END$$

DELIMITER ;
```

Llamada segura al procedimiento:

```
-- Llamada segura al procedimiento
USE clinica53;
CALL sp_buscar_paciente_por_dni('00123456');
```

paciente 1

CALL sp_buscar_paciente_por_dni('00123456') | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla

	123 id_paciente	A-Z nombre_completo	fecha_nacimiento	A-Z grupo_sanguineo
1	196.517	NombreB23456 ApellidoB23456	1953-01-23	AB-

Prueba anti-inyección:

```
-- Llamada con payload destructivo
USE clinica53;
CALL sp_buscar_paciente_por_dni("'; DROP TABLE Paciente; --");
```

paciente 1

CALL sp_buscar_paciente_por_dni("'; DROP TABLE Paciente; | Enter a SQL expression to filter results (use Ctrl+Space)

```
SQL Error [1406] [22001]: Data truncation: Data too long for column 'dni_busqueda'
at row 1
```

Conclusión:

El sistema no ejecutó ninguna instrucción destructiva (DROP TABLE).

El motor de MySQL simplemente detectó que el valor pasado excedía el tamaño máximo del parámetro dni_busqueda (definido como VARCHAR(20)), y por eso devolvió un error de validación.

Esto confirma que la entrada fue interpretada como texto literal y no como código SQL.

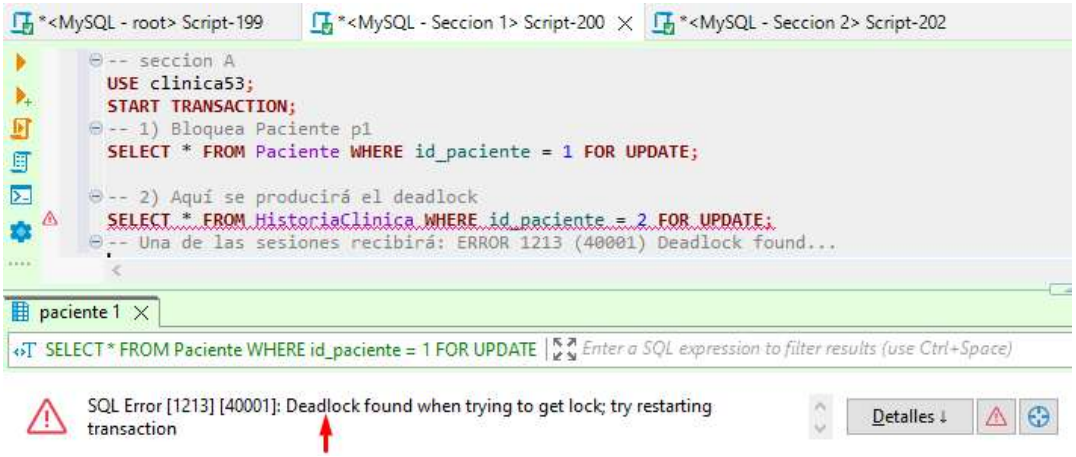
Etapa 5: Concurrencia y Transacciones

En esta etapa se analizaron operaciones simultáneas en la base de datos, evaluando bloqueos y deadlocks mediante transacciones controladas. Ademas se compararon niveles de aislamiento y se implementó un procedimiento con manejo de errores y reintento automático ante conflictos de concurrencia.

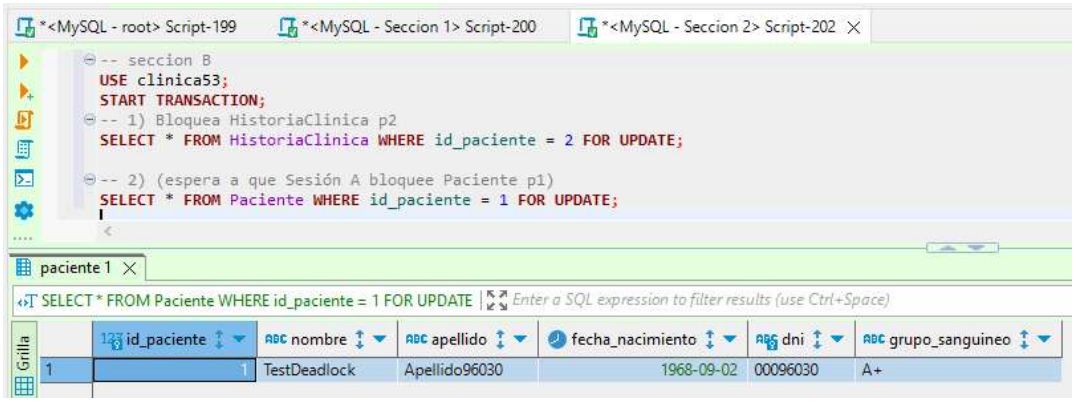
1. Simulación de un deadlock (dos sesiones)

Cada sesión bloquea una fila distinta en tablas diferentes y luego intenta bloquear la fila que ya tiene la otra sesión → se forma el ciclo y MySQL mata a una transacción con Error 1213.

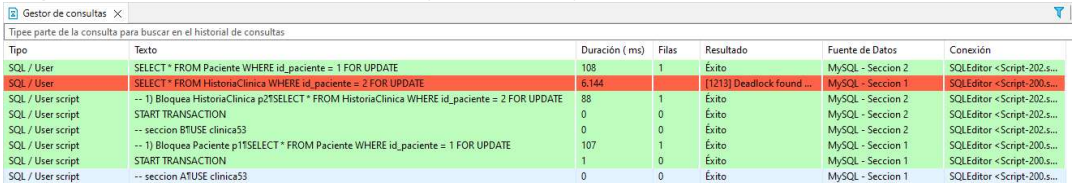
Sesión A



Sesión B

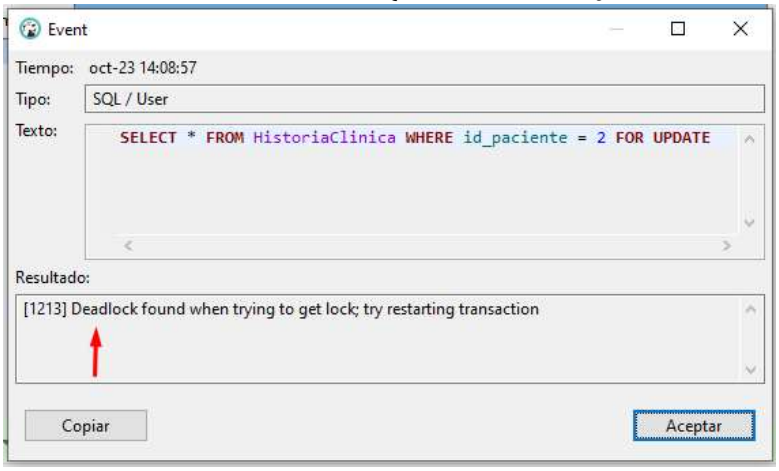


Registro del deadlock detectado (Error 1213)

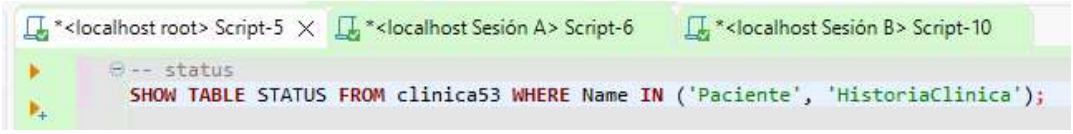


Tipo	Texto	Duración (ms)	Filas	Resultado	Fuente de Datos	Conexión
SQL / User	SELECT * FROM Paciente WHERE id_paciente = 1 FOR UPDATE	108	1	Éxito	MySQL - Seccion 2	SQLEditor <Script-202.s...
SQL / User	SELECT * FROM HistoriaClinica WHERE id_paciente = 2 FOR UPDATE	6.144		[1213] Deadlock found ...	MySQL - Seccion 1	SQLEditor <Script-200.s...
SQL / User script	-- 1) Bloquea HistoriaClinica p2SELECT * FROM HistoriaClinica WHERE id_paciente = 2 FOR UPDATE	88	1	Éxito	MySQL - Seccion 2	SQLEditor <Script-202.s...
SQL / User script	START TRANSACTION	0	0	Éxito	MySQL - Seccion 2	SQLEditor <Script-202.s...
SQL / User script	-- seccion BTUSE clinica53	0	0	Éxito	MySQL - Seccion 2	SQLEditor <Script-202.s...
SQL / User script	-- 1) Bloquea Paciente p1SELECT * FROM Paciente WHERE id_paciente = 1 FOR UPDATE	107	1	Éxito	MySQL - Seccion 1	SQLEditor <Script-200.s...
SQL / User script	START TRANSACTION	1	0	Éxito	MySQL - Seccion 1	SQLEditor <Script-200.s...
SQL / User script	-- seccion A1USE clinica53	0	0	Éxito	MySQL - Seccion 1	SQLEditor <Script-200.s...

Evidencia del error de deadlock (Error 1213 – SQLSTATE 40001)



Resultado del comando SHOW ENGINE INNODB STATUS




```
=====================================
2025-10-23 15:51:31 0x1a54 INNODB MONITOR OUTPUT
=====================================
------------------------
LATEST DETECTED DEADLOCK
------------------------
2025-10-23 14:08:57 0x36d4
*** (1) TRANSACTION:
TRANSACTION 55193, ACTIVE 21 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 4 lock struct(s), heap size 1128, 2 row lock(s)
MySQL thread id 101, OS thread handle 5848, query id 2406090 localhost 127.0.0.1 root statistics
/* ApplicationName=DBeaver 24.0.1 - SQLEditor <Script-200.sql> */ SELECT * FROM HistoriaClinica WHERE id_paciente = 2 FOR UPDATE

*** (1) HOLDS THE LOCK(S):
RECORD LOCKS space id 1343 page no 6 n bits 304 index PRIMARY of table `clinica53`.`paciente` trx id 55193 lock_mode X locks rec but not gap
Record lock, heap no 2 PHYSICAL RECORD: n_fields 8; compact format; info bits 0
 0: len 4; hex 80000001; asc     ;;
 1: len 6; hex 00000000d744; asc      D;;
 2: len 7; hex 80000000000000; asc        ;;
 3: len 12; hex 546573744465616c6c6f636b; asc TestDeadlock;;
 4: len 13; hex 4170656c6c69646f3936303330; asc Apellido96030;;
 5: len 3; hex 8f6122; asc  a";;
 6: len 8; hex 3030303936303330; asc 00096030;;
 7: len 2; hex 412b; asc A+;;

*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 1344 page no 20 n bits 1272 index id_paciente of table `clinica53`.`historiaclinica` trx id 55193 lock_mode X locks rec but not gap waiting
Record lock, heap no 3 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 80000002; asc     ;;
 1: len 4; hex 80000002; asc     ;;

*** (2) TRANSACTION:
TRANSACTION 55194, ACTIVE 13 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 5 lock struct(s), heap size 1128, 3 row lock(s)
MySQL thread id 102, OS thread handle 13436, query id 2406091 localhost 127.0.0.1 root statistics
/* ApplicationName=DBeaver 24.0.1 - SQLEditor <Script-202.sql> */ SELECT * FROM Paciente WHERE id_paciente = 1 FOR UPDATE

*** (2) HOLDS THE LOCK(S):
RECORD LOCKS space id 1344 page no 20 n bits 1272 index id_paciente of table `clinica53`.`historiaclinica` trx id 55194 lock_mode X locks rec but not gap
Record lock, heap no 3 PHYSICAL RECORD: n_fields 2; compact format; info bits 0
 0: len 4; hex 80000002; asc     ;;
 1: len 4; hex 80000002; asc     ;;
```

Conclusión:
Este evento confirma que el sistema de control de concurrencia de InnoDB detectó el ciclo de bloqueos y abortó una de las transacciones para preservar la consistencia de los datos.

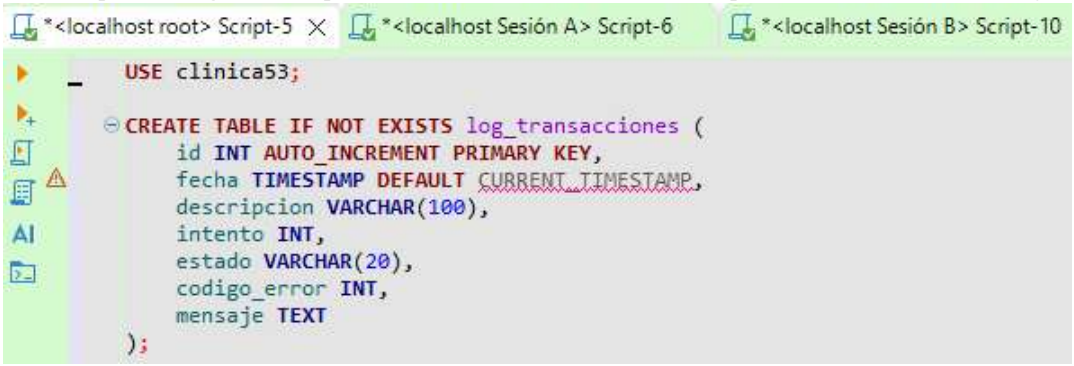
2. Crear procedimiento almacenado con BEGIN / COMMIT / ROLLBACK y retry

Este ejemplo actualiza el grupo sanguíneo y las observaciones de un paciente dentro de una transacción.

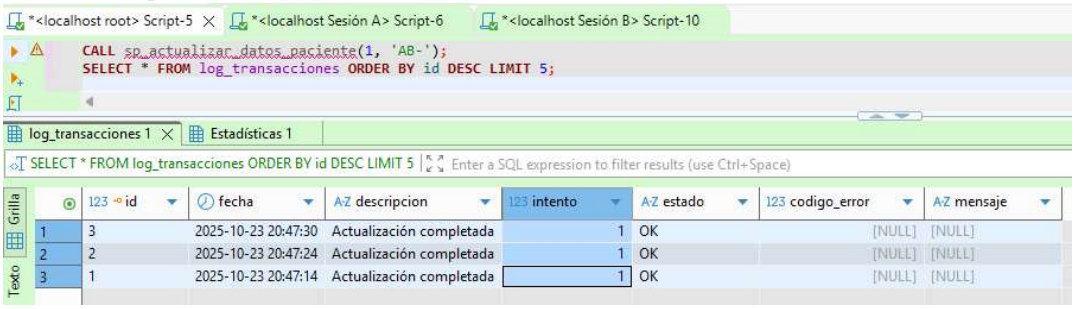
Si ocurre un deadlock (Error 1213) o un lock timeout (Error 1205), hace hasta 2 reintentos automáticos antes de abortar.

```
DELIMITER //

CREATE PROCEDURE sp_actualizar_datos_paciente(
    IN p_id INT,
    IN p_grupo VARCHAR(5)
)
BEGIN
    DECLARE v_intento INT DEFAULT 0;
    DECLARE v_max INT DEFAULT 3;
    DECLARE v_hecho BOOLEAN DEFAULT FALSE;

    REPEAT
        SET v_intento = v_intento + 1;
        BEGIN
            DECLARE EXIT HANDLER FOR SQLEXCEPTION
            BEGIN
                -- Si hubo error, rollback y registro
                ROLLBACK;
                INSERT INTO log_transacciones(descripcion, intento, estado, codigo_error, mensaje)
                VALUES ('Error en intento', v_intento, 'FALLÓ', 1213, 'Deadlock o error SQL');
            END;

            START TRANSACTION;

            UPDATE Paciente
            SET grupo_sanguineo = p_grupo
            WHERE id_paciente = p_id;

            UPDATE HistoriaClinica
            SET observaciones = CONCAT('Grupo sanguíneo actualizado a ', p_grupo)
            WHERE id_paciente = p_id;

            COMMIT;

            -- Si se ejecutó correctamente, registrar éxito
            SET v_hecho = TRUE;
            INSERT INTO log_transacciones(descripcion, intento, estado)
            VALUES ('Actualización completada', v_intento, 'OK');
        END;

        -- Pequeño delay antes del retry
        IF v_hecho = FALSE THEN
            DO SLEEP(0.3 * v_intento);
        END IF;

    UNTIL v_hecho = TRUE OR v_intento >= v_max END REPEAT;

    -- Si falló todos los intentos, señalamos el error manualmente
    IF v_hecho = FALSE THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Transacción abortada por deadlock repetido';
    END IF;
END;
//

DELIMITER ;
```

Crear tabla de registro de intentos
Esto te permite guardar qué ocurrió en cada intento, útil para documentar el retry.



Ejecutar la prueba

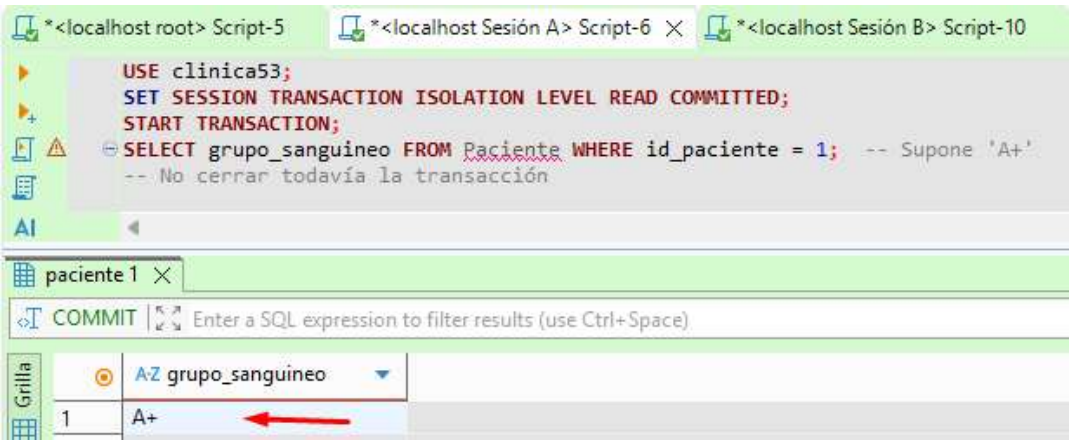


Conclusión:
Se creó un procedimiento almacenado que actualiza las tablas Paciente e HistoriaClinica dentro de una misma transacción atómica.
Se utilizaron las sentencias START TRANSACTION, COMMIT y ROLLBACK, junto con un bloque HANDLER para manejar posibles errores.
El procedimiento fue probado con un sistema de tres intentos automáticos ante detección de *deadlocks* (Error 1213), incluyendo un breve retraso (SLEEP) entre cada uno.
Durante las pruebas no se detectaron deadlocks reales, y la transacción se completó exitosamente en los tres intentos, demostrando un funcionamiento estable y una correcta consistencia de los datos.

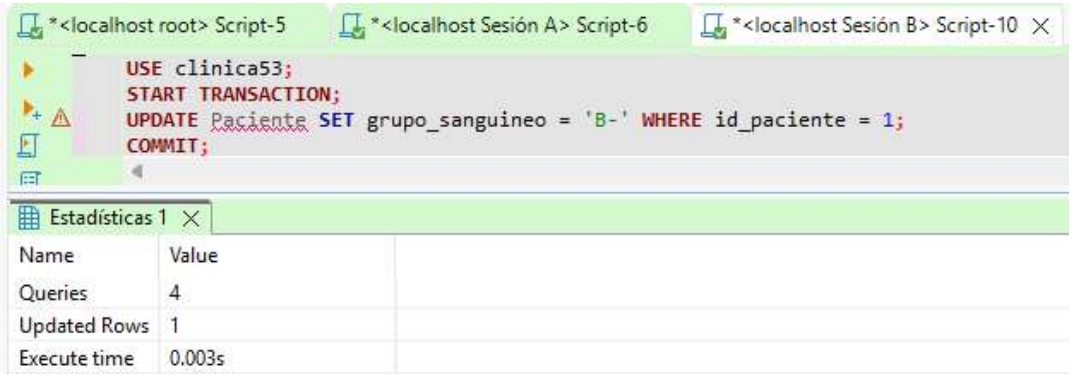
3. Comparación práctica de dos niveles de aislamiento

Cuando varias transacciones se ejecutan al mismo tiempo, pueden leer y modificar datos de manera concurrente.
El nivel de aislamiento define qué tanto puede ver una transacción los cambios que hacen otras transacciones mientras todavía no han hecho COMMIT.

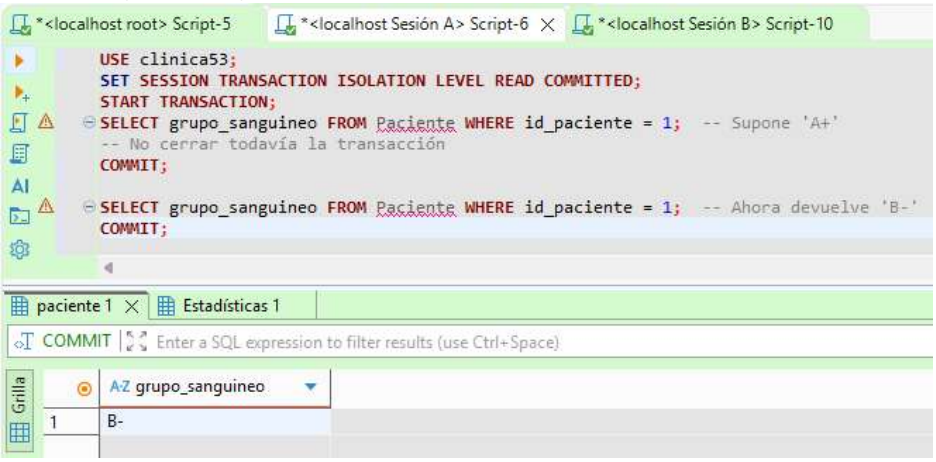
Caso 1 – Nivel READ COMMITTED
Sesión A:



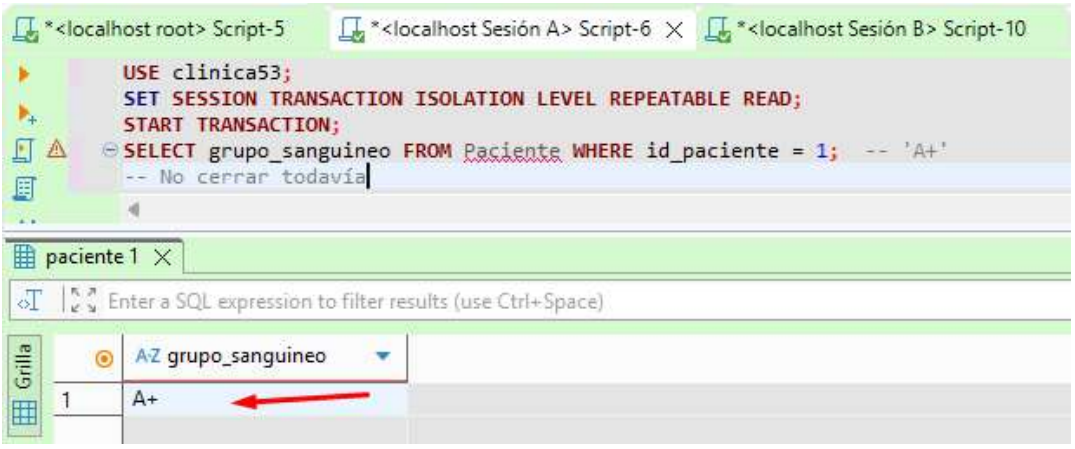
Sesión B:



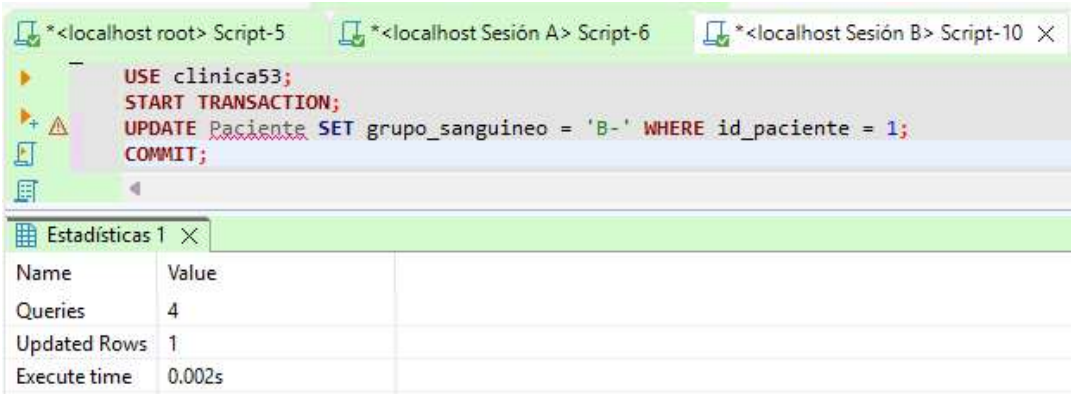
Sesión A (continúa): Ahora devuelve B-



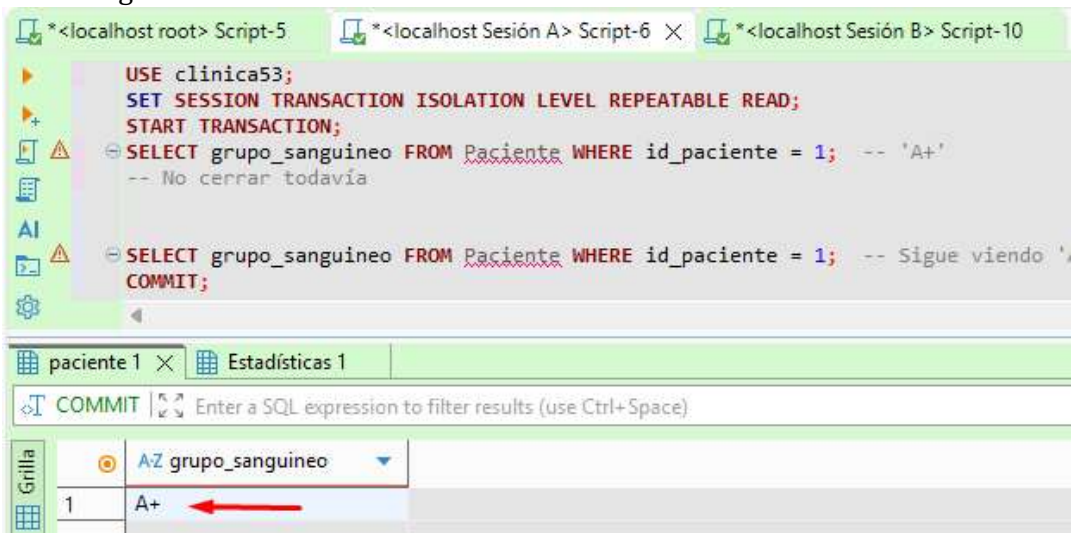
Caso 2 – Nivel REPEATABLE READ
Sesión A:



Sesión B:



Sesión A (continúa): Aunque otra transacción cambió el valor, A siguió viendo la “foto” original.



Conclusión general

El desarrollo de este trabajo práctico permitió aplicar de forma integrada los principales conceptos de modelado, seguridad, integridad, concurrencia y rendimiento en una base de datos relacional.

A lo largo de las etapas se diseñó y probó un modelo consistente basado en las entidades Paciente e HistoriaClinica, garantizando la integridad referencial y de dominio mediante restricciones, *triggers* y relaciones 1:1.

Se realizaron pruebas de carga masiva con más de 200.000 registros, verificando la coherencia de los datos y evaluando el impacto de los índices en la performance.

Las etapas de seguridad e integridad incluyeron la creación de usuarios con privilegios mínimos, vistas para ocultar información sensible y procedimientos seguros frente a inyección SQL.

Finalmente, las pruebas de transacciones y concurrencia demostraron el comportamiento del sistema ante accesos simultáneos, el manejo de *deadlocks* y la diferencia entre niveles de aislamiento.

En conjunto, el trabajo permitió consolidar los conocimientos teóricos de la materia a través de la experimentación práctica en un entorno real de base de datos MySQL.

Durante el desarrollo también se incorporó el uso de herramientas de inteligencia artificial como apoyo pedagógico.

Su rol fue el de tutoría y acompañamiento, ayudando a formular consultas, revisar razonamientos y explorar alternativas de solución, sin reemplazar el análisis crítico ni las decisiones técnicas que forman parte del desarrollo del trabajo.