TABLAS:

PATIENT (patient\_id , patient\_name, patient\_address, patient\_city, primary\_phone,

secondary\_phone)

DOCTOR (doctor\_id, doctor\_name, doctor\_address, doctor\_city, doctor\_speciality)

APPOINTMENT (patient\_id, appointment\_date, appointment\_duration, contact\_phone,

observations, payment\_card)

MEDICAL\_REVIEW (patient\_id, appointment\_date, doctor\_id)

PRESCRIBED\_MEDICATION (patient\_id, appointment\_date, medication\_name)

USE bd1;

1) Paso 1: Crear el usuario appointments\_user con todos los permisos

CREATE USER 'appointments\_user'@'localhost';

GRANT ALL PRIVILEGES ON bd1.\* TO 'appointments\_user'@'localhost';

FLUSH PRIVILEGES;

1. Paso 2: Crear el usuario appointments\_select con permisos solo de selección

CREATE USER 'appointments\_select'@'localhost';

GRANT SELECT ON bd1.\* TO 'appointments\_select'@'localhost';

FLUSH PRIVILEGES;

1. Paso 3: Crear el usuario appointments\_update con permisos de selección, inserción, actualización y eliminación a nivel de filas.

CREATE USER 'appointments\_update'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE ON bd1.\* TO 'appointments\_update'@'localhost';

FLUSH PRIVILEGES;

1. Paso 4: Crear el usuario appointments\_schema con permisos de selección, inserción, actualización, eliminación y modificación del esquema.

CREATE USER 'appointments\_schema'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE, ALTER, CREATE, DROP ON bd1.\* TO 'appointments\_schema'@'localhost';

FLUSH PRIVILEGES;

**2)** USE bd1;

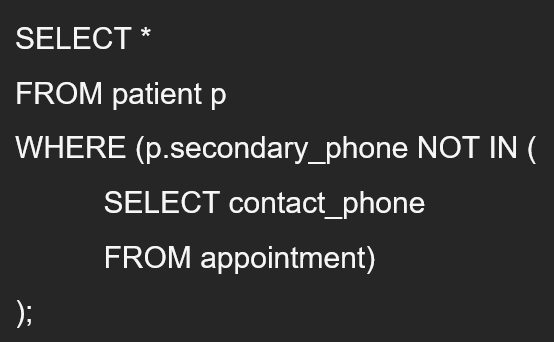
SELECT \*

FROM PATIENT p

WHERE NOT EXISTS (

SELECT **1**

FROM APPOINTMENT a

 WHERE a.patient\_id = p.patient\_id

AND a.contact\_phone = p.secondary\_phone

);

**3)** **VISTA**:

USE bd1;

DROP VIEW doctors\_per\_patients;

CREATE VIEW doctors\_per\_patients AS

SELECT p.patient\_id, d.doctor\_id

FROM patient p

INNER JOIN doctor d ON p.patient\_city = d.doctor\_city;

**4) a-** USE bd1;

SELECT patient\_id

FROM patient p

WHERE (

SELECT COUNT(doctor\_id)

FROM doctor d

WHERE p.patient\_city = d.doctor\_city)

= (

SELECT COUNT(DISTINCT d.doctor\_id)

FROM medical\_review mw

INNER JOIN doctor d ON mw.doctor\_id = d.doctor\_id

WHERE p.patient\_id = mw.patient\_id

AND p.patient\_city = d.doctor\_city

)

ORDER BY patient\_id

LIMIT 100;

**b-** USE bd1;

SELECT patient\_id

FROM patient p

WHERE (

SELECT COUNT(doctor\_id)

FROM doctors\_per\_patients dp -- ESTO CAMBIA

WHERE p.patient\_id = dp.patient\_id) -- ESTO CAMBIA

= (

SELECT COUNT(DISTINCT d.doctor\_id)

FROM medical\_review mw

INNER JOIN doctor d ON mw.doctor\_id = d.doctor\_id

WHERE p.patient\_id = mw.patient\_id

AND p.patient\_city = d.doctor\_city

)

ORDER BY patient\_id

LIMIT 100;

**5)** CREATE TABLE APPOINTMENTS\_PER\_PATIENT (

idApP INT(11) PRIMARY KEY AUTO\_INCREMENT, -- Clave primaria con auto-incremento

id\_patient INT(11) NOT NULL, -- Identificador del paciente, obligatorio

count\_appointments INT(11), -- Cantidad de citas

last\_update DATETIME, -- Fecha y hora de la última actualización

user VARCHAR(16), -- Usuario que realizó la última modificación

);

No hace falta:

FOREIGN KEY (id\_patient) REFERENCES PATIENT(patient\_id) -- FK a la tabla PATIENT

**6) STORED PROCEDURE**:

USE bd1;

DELIMITER //

CREATE PROCEDURE storedProcedure() BEGIN

DECLARE fin INT DEFAULT 0;

DECLARE id\_paciente INT(11);

DECLARE cant INT(11);

-- Definir el cursor para calcular el número de citas para cada paciente

DECLARE contar CURSOR FOR

SELECT a.patient\_id, COUNT(a.patient\_id) -- AS cantAppointments -- var contadora

FROM appointment a

GROUP BY a.patient\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = 1; -- Manejar fin del cursor

START TRANSACTION; -- Iniciar la transacción

OPEN contar; -- Abrir el cursor

loop\_cursor: LOOP -- Iterar sobre cada fila en el cursor

FETCH NEXT FROM contar INTO id\_paciente, cant;

IF fin = 1 THEN

LEAVE loop\_cursor;

END IF;

INSERT INTO appointments\_per\_patient(id\_patient, count\_appointments, last\_update, user) -- columna del appointments\_per\_patient

VALUES (id\_paciente, cant, NOW(), CURRENT\_USER());

END LOOP;

CLOSE contar; -- Cerrar el cursor

COMMIT; -- Confirmar la transacción

END;

// DELIMITER ;

**7)** **TRIGGER**: -- no se puede modificar, si o si borrar y desp volver a crear

USE bd1;

DROP TRIGGER IF EXISTS addAppointmentPorCliente;

DELIMITER //

CREATE TRIGGER addAppointmentPorCliente

AFTER INSERT ON appointment FOR EACH ROW

BEGIN

UPDATE appointments\_per\_patient

SET count\_appointments = (count\_appointments + 1),

last\_update = NOW(),

user = CURRENT\_USER()

WHERE id\_patient = NEW.patient\_id;

END;

// DELIMITER;

**VERIFICAR**: DROP TRIGGER IF EXISTS addAppointmentPorCliente;

DELIMITER //

CREATE TRIGGER addAppointmentPorCliente

AFTER INSERT ON appointment FOR EACH ROW

BEGIN

DECLARE existe INT;

SELECT 1 INTO existe

FROM appointments\_per\_patient

WHERE id\_patient = NEW.patient\_id

LIMIT 1;

IF existe = 1 THEN

UPDATE appointments\_per\_patient

SET count\_appointments = count\_appointments + 1,

last\_update = NOW(),

user = CURRENT\_USER()

WHERE id\_patient = NEW.patient\_id;

ELSE

INSERT INTO appointments\_per\_patient (id\_patient, count\_appointments, last\_update, user)

VALUES (NEW.patient\_id, 1, NOW(),CURRENT\_USER());

END IF;

END;

// DELIMITER ;

**8)** USE bd1;

DROP PROCEDURE IF EXISTS agregarAppointment;

DELIMITER //

CREATE PROCEDURE agregarAppointment (IN idPaciente INT, IN idDoctor INT, IN durCita INT, IN telContacto VARCHAR(45), IN nombreMedicacion VARCHAR(30))

BEGIN

DECLARE diaCita DATETIME; -- Declarar variable local

SET diaCita = NOW();

START TRANSACTION;

INSERT INTO appointment (patient\_id, appointment\_date, appointment\_duration,

contact\_phone, observations, payment\_card)

VALUES (idPaciente, diaCita, durCita, telContacto, NULL, NULL);

INSERT INTO medical\_review (patient\_id, appointment\_date, doctor\_id)

VALUES (idPaciente, diaCita, idDoctor);

INSERT INTO prescribed\_medication (patient\_id, appointment\_date, medication\_name)

VALUES (idPaciente, diaCita, nombreMedicacion);

COMMIT;

END;

// DELIMITER ;

**9)** USE bd1;

CALL agregarAppointment (10004427, 1003, 30, "+54 15 2913 9963", "Paracetamol" );

//

SELECT \* FROM appointment WHERE patient\_id = 10004427;

SELECT \* FROM medical\_review WHERE patient\_id = 10004427;

SELECT \* FROM prescribed\_medication WHERE patient\_id = 10004427;

**10)** EXPLAIN

SELECT count(a.patient\_id)

FROM appointment a, patient p, doctor d, medical\_review mr

WHERE a.patient\_id = p.patient\_id

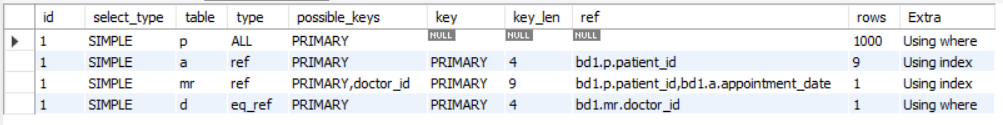
AND a.patient\_id = mr.patient\_id

AND a.appointment\_date = mr.appointment\_date

AND mr.doctor\_id = d.doctor\_id

AND d.doctor\_specialty = "Cardiology"

AND p.patient\_city = "Rosario"



a- ¿Qué atributos del plan de ejecución encuentra relevantes para evaluar la performance de la consulta?

* type: tipo de acceso realizado sobre las filas.
* key: índice seleccionado para realizar la operación. La elección correcta de un índice indica una mejor performance.
* rows: cantidad de filas sobre las que opera la consulta. Cuantas + filas haya, + lenta será.

ADICIONAL -> filtered: indica el porcentaje de filas que se espera que se filtren. Cuanto > sea el porcentaje, se considera que la consulta es + eficiente.

b- Observe en particular el atributo type ¿cómo se están aplicando los JOIN entre las tablas involucradas?

ALL (p): indica que las tablas se escanean por completo. Es el menos eficiente.

EQ\_REF (d): indica que se está usando un índice que coincide entre ambas tablas para unirlas.

REF (a, mr): indica que se está usando un índice para encontrar una fila específica.

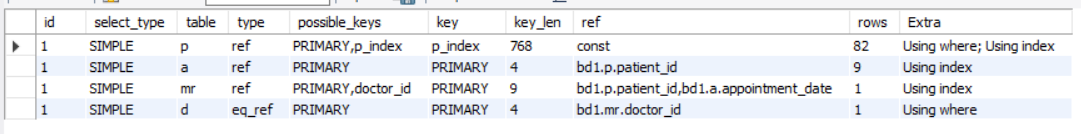
Comparando los 3 tipos, eq\_ref realiza el JOIN de manera más eficiente puesto que no evalúa las tablas completas.

c- Según lo que observó en los puntos anteriores, ¿qué mejoras se pueden realizar para optimizar la consulta?

Se pueden agregar índices en la tabla p, que es la que se une con el tipo ALL. De esta manera va a hacer una consulta + eficiente.

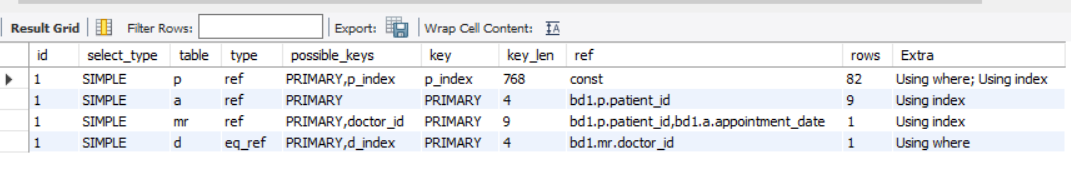
También podría agregar índice en la tabla d, para usarlo en vez del where.

d- Aplique las mejoras propuestas y vuelva a analizar el plan de ejecución. ¿Qué cambios observa?

CREATE INDEX p\_index ON patient(patient\_city);

DROP INDEX p\_index ON patient;

**POR QUE ACÁ NO MEJORA?????** Si además le agrego el índice al doctor:

CREATE INDEX d\_index ON doctor(doctor\_specialty);