**Práctica 4 - Conceptos Aplicados usando MySQL**

**CONSULTAS SQL**

Llevar a SQL las siguientes consultas antes realizadas en Álgebra Relacional:

**Dado el siguiente esquema:**

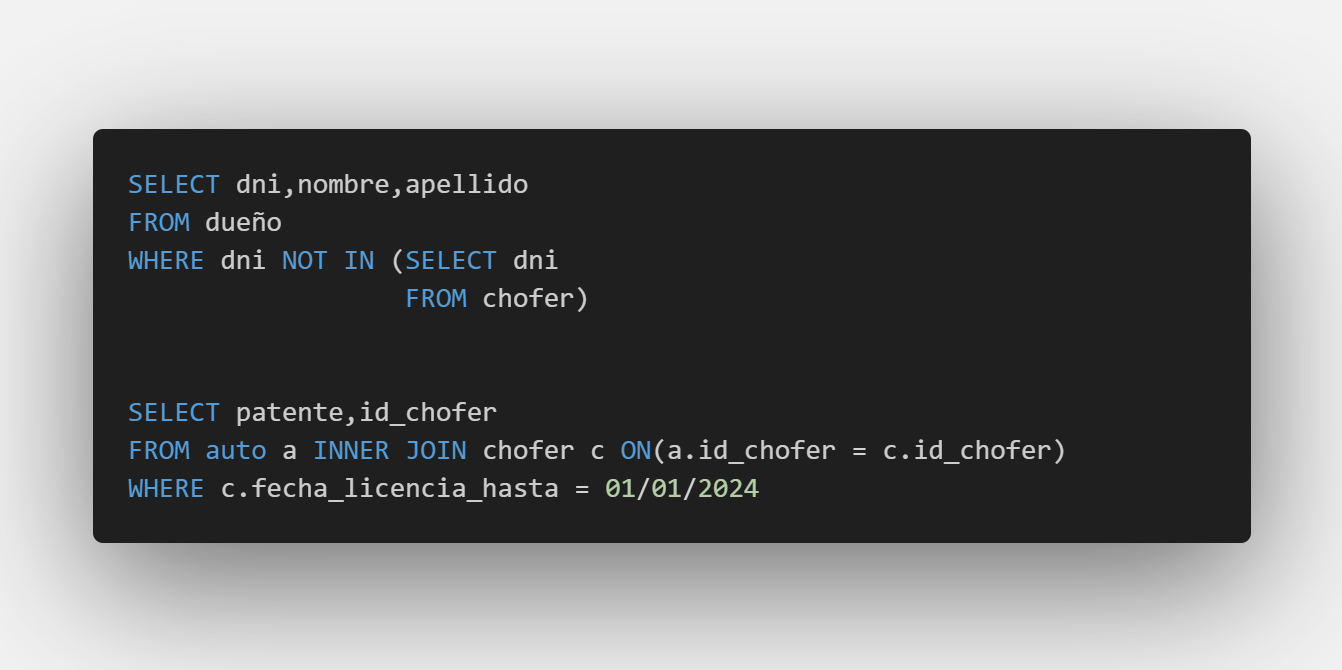
**DUEÑO** (id\_dueño, nombre, teléfono, dirección, dni)

**CHOFER** (id\_chofer, nombre, teléfono, dirección, fecha\_licencia\_desde, fecha\_licencia\_hasta, dni)

**AUTO** (patente, id\_dueño, id\_chofer, marca, modelo, año)

**VIAJE** (patente, hora\_desde, hora\_hasta, origen, destino, tarifa, metraje)

1. Listar el dni, nombre y teléfono de todos los dueños que NO son choferes.
2. Listar la patente y el id\_chofer de todos los autos a cuyos choferes les caduca la licencia el 01/01/2024.



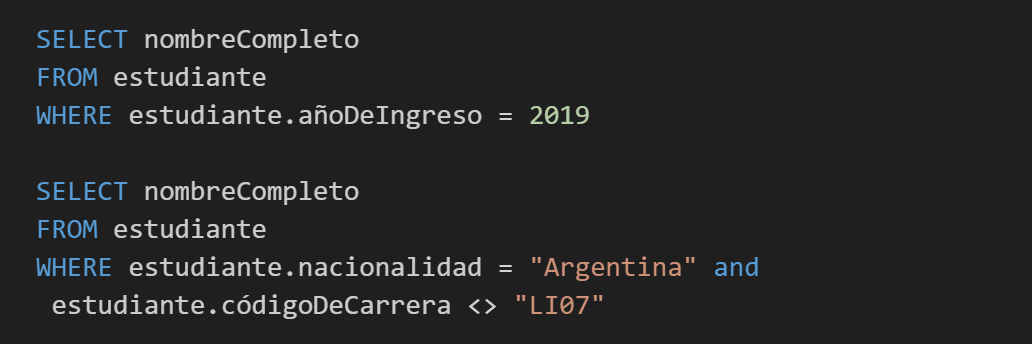
**ESTUDIANTE** (#legajo, nombreCompleto, nacionalidad, añoDeIngreso, códigoDeCarrera)

**CARRERA** (códigoDeCarrera, nombre)

**INSCRIPCIONAMATERIA** (#legajo, códigoDeMateria)

**MATERIA** (códigoDeMateria, nombre)

1. Obtener el nombre de los estudiantes que ingresaron en 2019.
2. Obtener el nombre de los estudiantes con nacionalidad “Argentina” que NO estén en la carrera con código “LI07”.



**PREPARACIÓN BD**

Un hospital posee una base de datos para almacenar información sobre las atenciones que se realizan para sus pacientes, además de los doctores que los atendieron en cada atención y los medicamentos que le fueron recetados. El esquema con el que cuentan es el siguiente:

**APPOINTMENTS**(patient\_id, patient\_name, patient\_address, patient\_city, primary\_phone, secondary\_phone, doctor\_id, doctor\_name, doctor\_address, doctor\_city, doctor\_speciality, appointment\_date, appointment\_duration, observations, payment\_card, contact\_phone, medication\_name)

Clave candidata del esquema **APPOINTMENTS**:

**CC**: (patient\_id, doctor\_id, appointment\_date, medication\_name)

Dependencias funcionales válidas en el esquema **APPOINTMENTS**:

**DF1**: patient\_id -> patient\_name, patient\_address, patient\_city, primary\_phone, secondary\_phone

**DF2**: doctor\_id-> doctor\_name, doctor\_address, doctor\_city, doctor\_speciality

**DF3**: patient\_id, appointment\_date -> appointment\_duration, contact\_phone, observations, payment\_card

Dependencias multivaluadas válidas en el esquema **APPOINTMENTS**:

**DM1**: patient\_id, appointment\_date ->> doctor\_id

**DM2**: patient\_id, appointment\_date ->> medication\_name

Luego de haber aplicado el proceso de normalización quedan las siguientes particiones en 4FN:

**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)

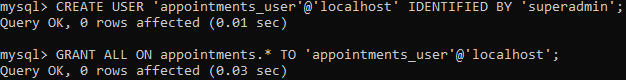
**PRESCRIBED\_MEDICATION** (patient\_id, appointment\_date, medication\_name)

Y la siguiente Clave Primaria:

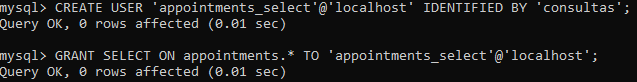
**CP** = (patient\_id, doctor\_id, appointment\_date, medication\_name)

**EJERCICIOS**

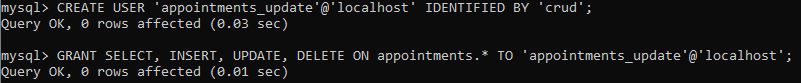
1. Crea un usuario para las bases de datos usando el nombre ‘appointments\_user’. Asigne a estos todos los permisos sobre sus respectivas tablas. Habiendo creado este usuario evitaremos el uso de ‘root’ para el resto del trabajo práctico. Adicionalmente, con respecto a esta base de datos:
   1. Cree un usuario sólo con permisos para realizar consultas de selección, es decir que no puedan realizar cambios en la base. Use el nombre ‘appointments\_select’.
   2. Cree un usuario que pueda realizar consultas de selección, inserción, actualización y eliminación a nivel de filas, pero que no puedan modificar el esquema. Use el nombre ‘appointments\_update’.
   3. Cree un usuario que tenga los permisos de los anteriores, pero que además pueda modificar el esquema de la base de datos. Use el nombre 'appointments\_schema’.



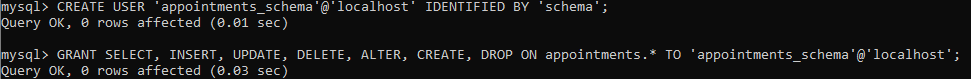
a)



b)



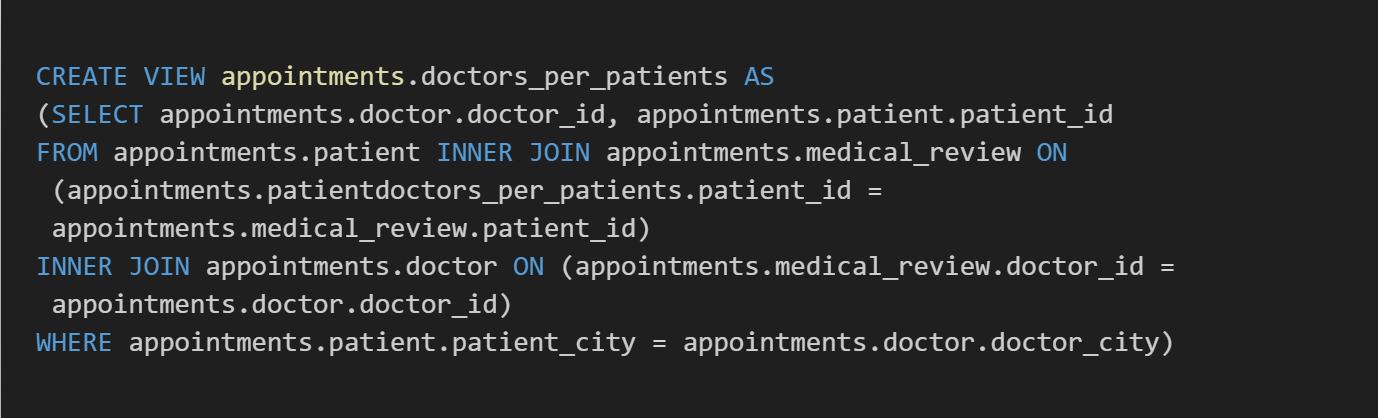
c)



1. Hallar aquellos pacientes que para todas sus consultas médicas siempre hayan dejado su número de teléfono primario (nunca el teléfono secundario).

**Todos los que vi tienen primario y secundario**

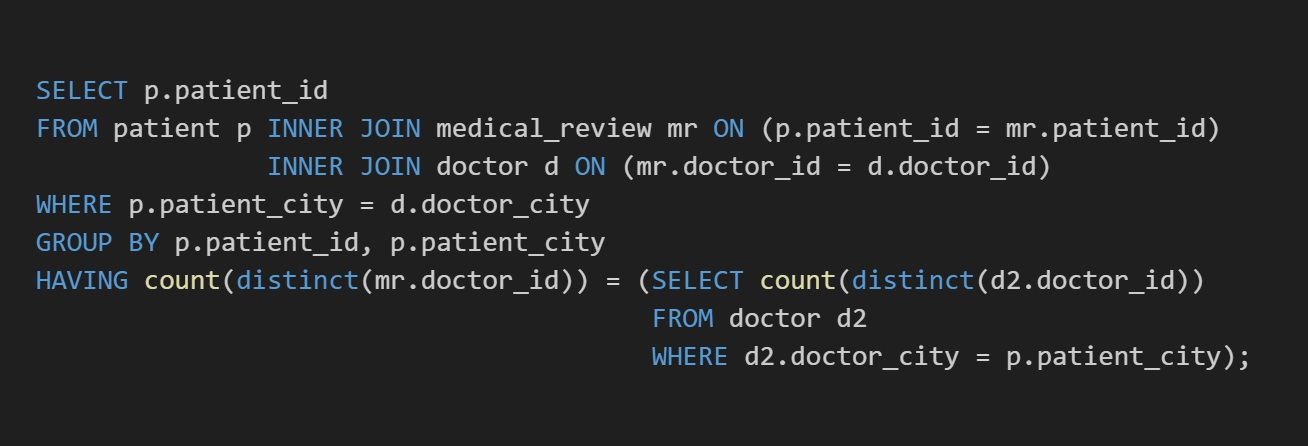
1. Crear una vista llamada ‘doctors\_per\_patients’ que muestre los id de los pacientes y los id de doctores de la ciudad donde vive el cliente.



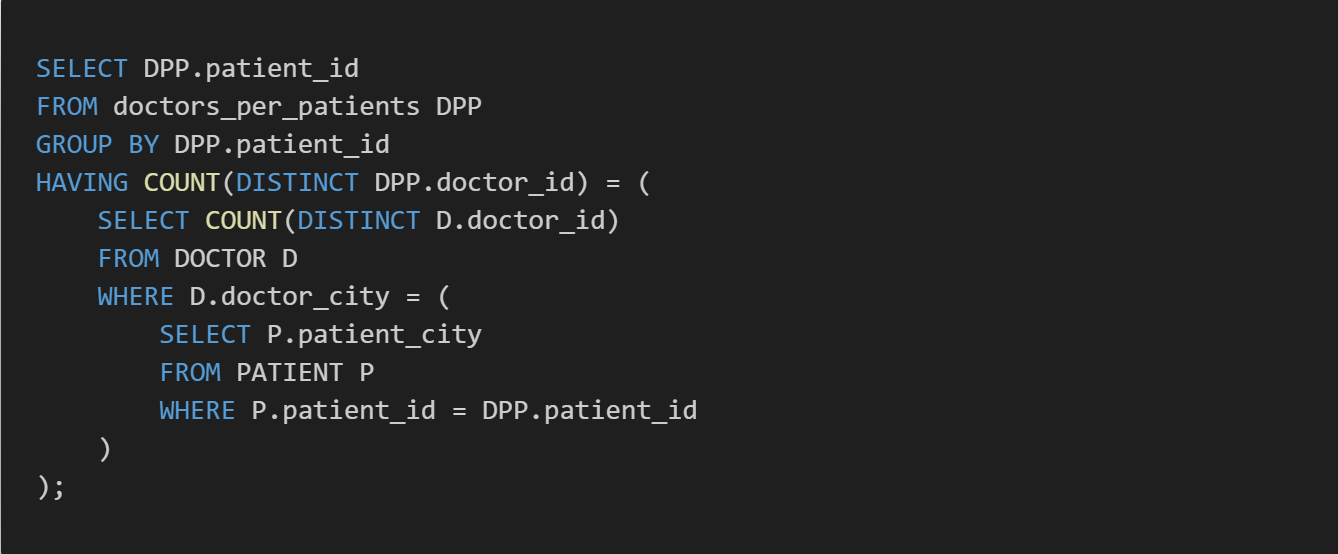
1. Hallar los pacientes (únicamente es necesario su id) que se atendieron con todos los doctores de la ciudad en la que viven.
   1. Realice la consulta sin utilizar la vista creada anteriormente.
   2. Realice la consulta utilizando la vista creada anteriormente.

Restricción: resolver este ejercicio sin usar la cláusula “NOT EXIST”.

a)



b)



1. Agregar la siguiente tabla:

**APPOINTMENTS\_PER\_PATIENT**

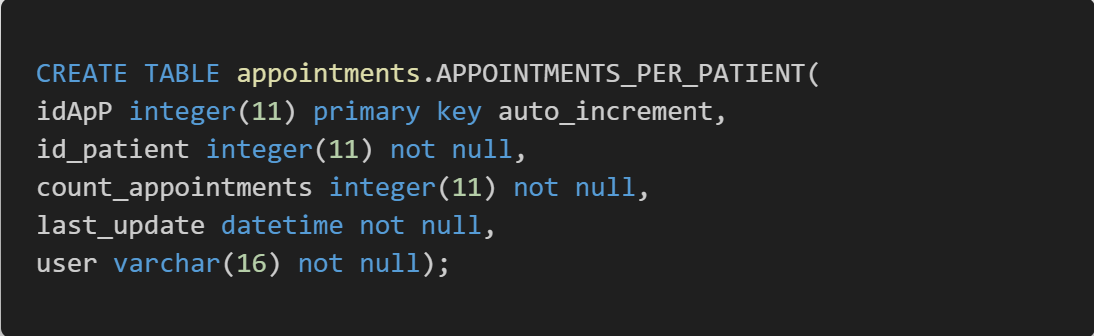
**idApP**: int(11) PK AI

**id\_patient**: int(11)

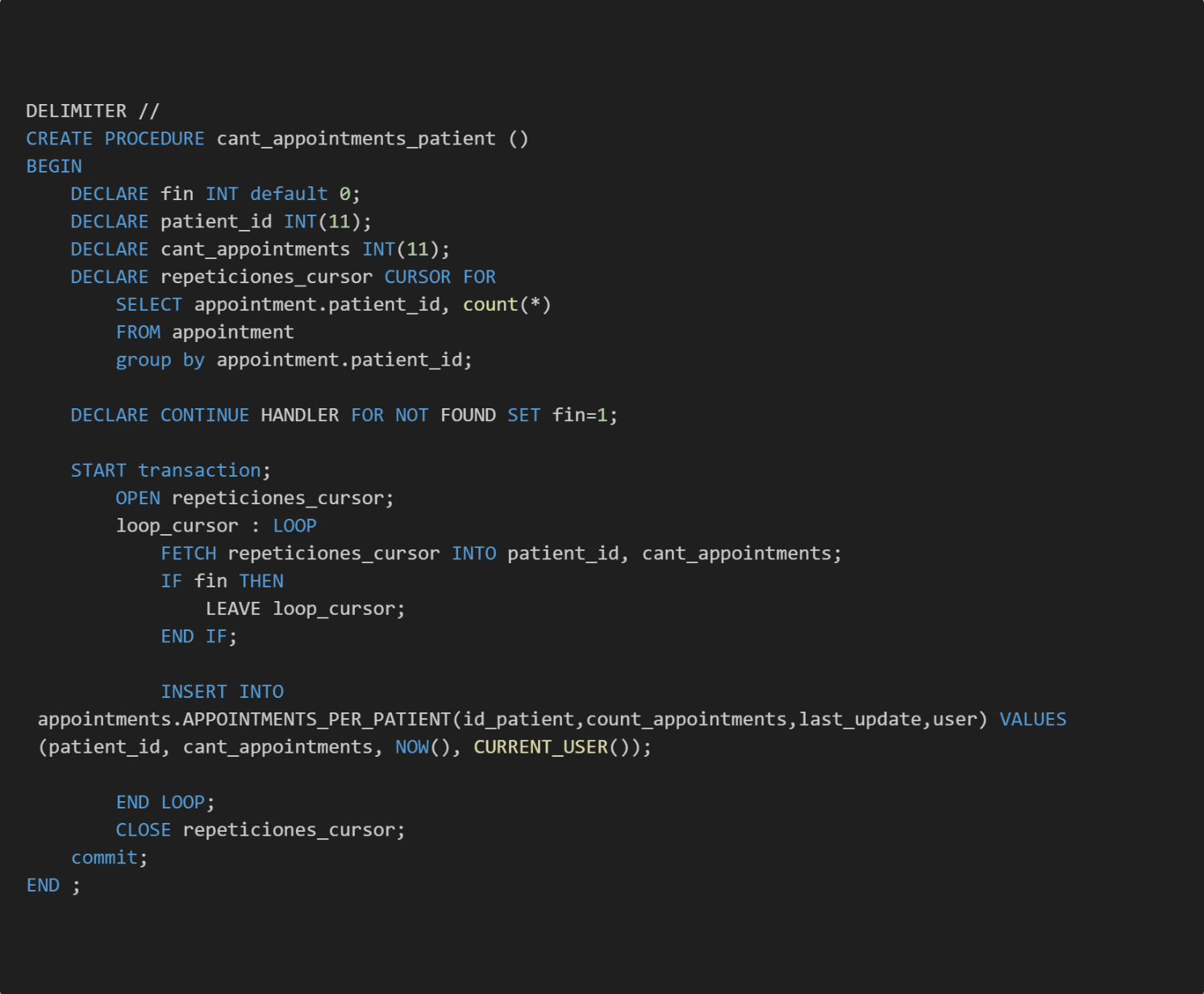
**count\_appointments**: int(11)

**last\_update**: datetime

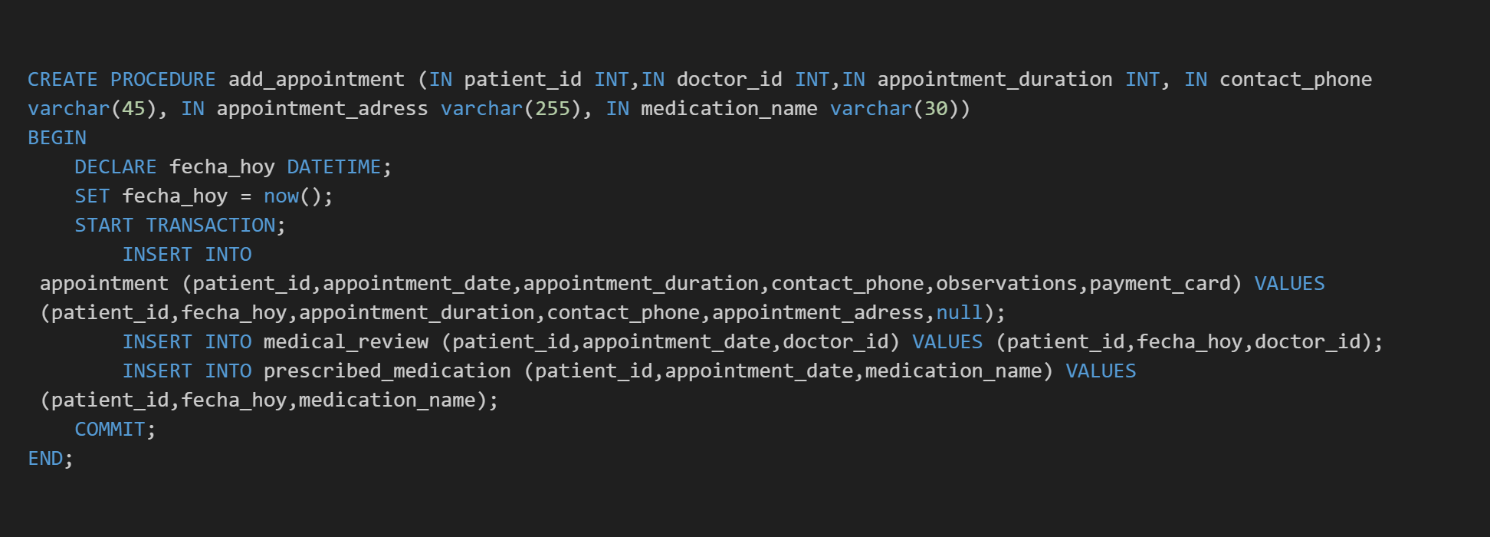
**user**: varchar(16)



1. Crear un Stored Procedure que realice los siguientes pasos dentro de una transacción:
   1. Realizar una consulta que para cada pacient (identificado por id\_patient), calcule la cantidad de appointments que tiene registradas. Registrar la fecha en la que se realiza esta carga y además del usuario con el se realiza.
   2. Guardar el resultado de la consulta en un cursor.
   3. Iterar el cursor e insertar los valores correspondientes en la tabla APPOINTMENTS PER PATIENT.



8 y 9. Crear un stored procedure que sirva para agregar un appointment, junto el registro de un doctor que lo atendió (medical\_review) y un medicamento que se le recetó (prescribed\_medication), dentro de una sola transacción. El stored procedure debe recibir los siguientes parámetros: patient\_id, doctor\_id, appointment\_duration, contact\_phone, appointment\_address, medication\_name. El appointment\_date será la fecha actual. Los atributos restantes deben ser obtenidos de la tabla Patient (o dejarse en NULL).



10. Considerando la siguiente consulta:

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\_speciality = ‘Cardiology’

and p.patient\_city = Rosario

Analice su plan de ejecución mediante el uso de la sentencia EXPLAIN.

1. ¿Qué atributos del plan de ejecución encuentra relevantes para evaluar la performance de la consulta?

**RTA:** Encuentro relevante los atributos possible\_keys, rows y extra.

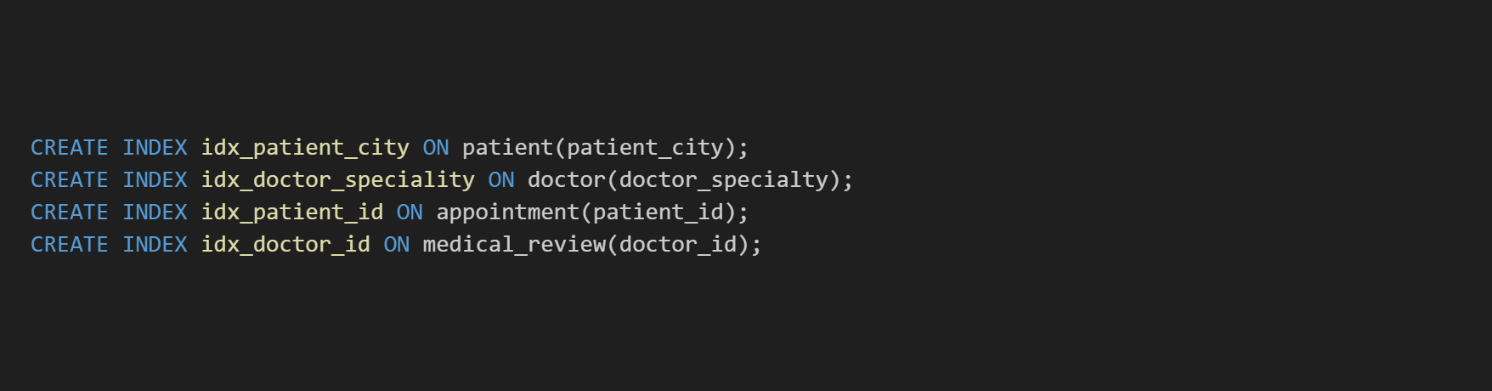
1. Observe en particular el atributo type ¿cómo se están aplicando los JOIN entre las tablas involucradas?

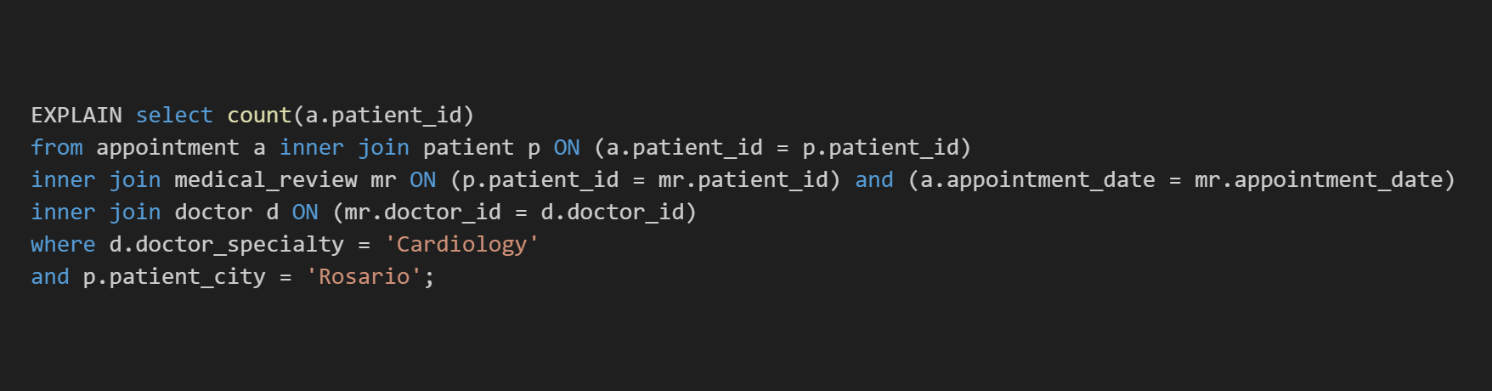
**RTA:** Aparecen algunos como **ALL** (Es un escaneo completo de la tabla (full table scan). MySQL lee todas las filas de la tabla sin usar ningún índice), **ref** (MySQL utiliza un índice para encontrar filas que coinciden con un valor no único. Es similar a eq\_ref, pero se puede encontrar más de una fila coincidente) y **eq\_ref** (Este es uno de los tipos más eficientes. Se utiliza cuando una consulta está haciendo una búsqueda de igualdad en una clave primaria o un índice único)

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

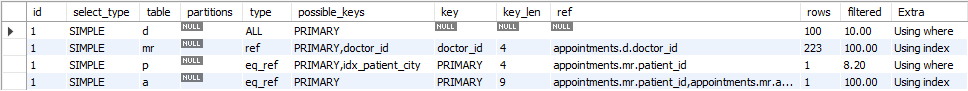
**RTA:** Tendríamos que llevar el type de las consultas a los mas eficientes posibles, podríamos hacer uso de indices y aplicar INNER JOINS.

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

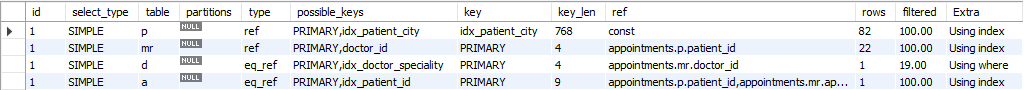




**Resultado viejo:**



**Resultado nuevo:**



Haciendo uso de los indices evitamos traernos toda la tabla con todos los atributos.