Cátedra de Bases de
Datos 1 Cursada 2024

Práctica Nro. 4

Conceptos Aplicados usando MySQL

Publicación: 09/10/2024
Finalización: 21/10/2024

Para la resolución de este TP se necesita tener instalado una instancia de MySQL, para instalarla, ingrese a este link y descargue el instalador correspondiente a su SO.

CONSULTAS SQL

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

1. 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)

   a. **Listar el dni, nombre y teléfono de todos los dueños que NO son choferes**

```
SELECT d.dni, d.nombre, d.teléfono
FROM dueño d
WHERE d.dni NOT IN (
        SELECT c.dni
        FROM chofer c
);
```

   b. **Listar la patente y el id_chofer de todos los autos a cuyos choferes les caduca la licencia el 01/01/2024**

```
SELECT a.patente, a.id_chofer
FROM auto a
WHERE a.id_chofer IN (
        SELECT c.id_chofer
        FROM chofer c
        WHERE fecha_licencia_hasta = '20240101'
);
```

ESTUDIANTE (#legajo, nombreCompleto, nacionalidad, añoDeIngreso, códigoDeCarrera)
CARRERA (códigoDeCarrera, nombre)
INSCRIPCIONAMATERIA (#legajo, códigoDeMateria)
MATERIA (códigoDeMateria, nombre)

a. **Obtener el nombre de los estudiantes que ingresaron en 2019.**
```
SELECT nombre
FROM estudiante
WHERE añoDeIngreso = 2019;
```

b. **Obtener el nombre de los estudiantes con nacionalidad "Argentina" que NO estén en la carrera con código "LI07"**
```
SELECT e.nombre
FROM estudiante e
WHERE e.codigoDeCarrera <> 'LI07';
```

Compare las resoluciones de estos ejercicios con las realizadas en álgebra relacional, que paralelismo encuentra entre las diferentes operaciones de AR en SQL y en las formas de las resoluciones?
Las operaciones de SQL están basadas en el álgebra relacional, por lo que se encuentran paralelismos bastante evidentes entre ambos lenguajes:

SELECT <...atributos> equivale a π ...atributos
WHERE <condición> equivale a σ condición
<tabla> AS <nombre_nuevo> equivale a ρ nombre_nuevo Tabla
INNER JOIN <condición> es un producto natural, con la diferencia de que permite que el desarrollador elija la condición por la que se unen ambas tablas.

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)

Se proveen dos archivos separados con lo necesario para la creación de las tablas e inserción de datos. Ambos archivos se encuentran en el archivo comprimido **apppointments.sql.zip** adjunto a esta práctica.

Para crear los esquemas y cargar los datos, hacerlo desde línea de comando. Para esto, descomprimir los archivos y ejecutar desde la terminal el siguiente comando para acceder a la terminal mysql:

mysql -h localhost -u root -p

dentro de la terminal mysql, crear ambos esquemas:

mysql> create database appointments;
mysql> exit;

nuevamente en la terminal, ejecutar los scripts que contienen ambos archivos: *'appointments.sql'* creará las tablas, mientras que *'insert_appointments.sql'* crea una serie de tuplas de ejemplo para poder realizar las consultas.

mysql appointments -h localhost -u root -p < ruta_del_archivo

donde ruta_del_archivo es el path al archivo provisto.

Nota: Debe ingresar la contraseña del usuario *root* por cada comando que ejecute en nombre de este.

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:**
    a. **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'.**
    b. **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'.**
    c. **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'.**

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

3. 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.

4. Hallar los pacientes (únicamente es necesario su id) que se atendieron con todos los doctores de la ciudad en la que viven
    a. Realice la consulta sin utilizar la vista creada anteriormente
    b. Realice la consulta utilizando la vista creada anteriormente Restricción: resolver este ejercicio sin usar la cláusula "NOT EXIST".

5. 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)

6. Crear un Stored Procedure que realice los siguientes pasos dentro de una transacción:
    a. 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.
   b. Guardar el resultado de la consulta en un cursor.
   c. Iterar el cursor e insertar los valores correspondientes en la tabla APPOINTMENTS PER PATIENT.

7. Crear un Trigger de modo que al insertar un dato en la tabla Appointment, se actualice la cantidad de appointments del paciente, la fecha de actualización y el usuario responsable de la misma (actualiza la tabla APPOINTMENTS PER PATIENT).

8. 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).

9. Ejecutar el stored procedure del punto 8 con los siguientes datos: patient_id: 10004427

   doctor_id: 1003
   appointment_duration: 30
   contact_phone: +54 15 2913 9963
   appointment_address: 'Hospital Italiano'
   medication_name: 'Paracetamol'

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.

   a. ¿Qué atributos del plan de ejecución encuentra relevantes para evaluar la performance de la consulta?
   b. Observe en particular el atributo type ¿cómo se están aplicando los JOIN entre las tablas involucradas?
   c. Según lo que observó en los puntos anteriores, ¿qué mejoras se pueden realizar para optimizar la consulta?
   d. Aplique las mejoras propuestas y vuelva a analizar el plan de ejecución. ¿Qué cambios observa?