Universidad Nacional de Quilmes

BASES DE DATOS
TRABAJO PRÃĄCTICO - SQL-
02/06/2022

# Pautas

Lea **atentamente** las pautas para la resolución de los ejercicios planteados y la entrega del trabajo práctico.

(a) Este TP **debe resolverse en grupos de solamente 2 estudiantes**.

(b) Genere los archivos `apellido1_apellido2_gitunq_DDL.sql` y `apellido1_apellido2_gitunq_DML.sql`, reemplazando las palabras *apellido1_apellido2* por los correspondientes a los miembros del grupo.
Ejemplo: `palazzo_sandoval_gitunq_DDL.sql`.

(c) Guarde las consultas pedidas en el archivo correspondiente.

(d) Guarde las descripciones o aclaraciones en el archivo que corresponda utilizando comentarios de sql. Parte del proceso de corrección se realiza mediante parseo automático, es importante el uso de comentarios en todo lo que no sean sentencias sql.

(e) Escriba todas las consultas de forma prolija. Es importante que entre cada consulta haya espacios de líneas suficientes. Termine cada consulta con `;` (punto y coma).

(f) Es importante que cada consulta comience en una línea aparte. Esto disminuye la cantidad de errores y facilita la corrección.

(g) Esta es una instancia de evaluación, aclare todo lo que considere importante, aún lo que considere trivial.

(h) Suba al aula virtual ambos archivos comprimidos en un archivo `apellido1_apellido2_gitunq.zip` con la resolución del trabajo.

(i) Puede enviar consultas al `tpi-doc-bd@listas.unq.edu.ar` escribiendo en el Asunto del email *[UNQ - BD] Consulta TP Ejercicio(s) XX* identificando XX con el/los número(s) de ejercicio(s) sobre el/los cuales quiere hacer preguntas, o bien puede usar el canal de Discord pero recuerde no enviar soluciones de ejercicios.

(j) La fecha límite de entrega es el **15 de Junio de 2022** inclusive. Si termina el TP antes de la fecha límite, puede subirlo al aula antes.

# 1. Resumen de SQL

La sintaxis del DML de SQL puede resumirse de la siguiente manera:

```
SELECT [ALL | DISTINCT] <atributos>
FROM <tablas>
[WHERE <expresion condicional>]
[GROUP BY <atributo(s)>]
[HAVING <expresion condicional>]
[ORDER BY <columna(s)>]
```

donde:

- `<atributos>` es la información a obtener de la base de datos.
- `FROM <tablas>` especifica de qué tablas se obtiene la información buscada.
- `WHERE <expresion condicional>` expresa una condición que deben cumplir las filas de la consulta resultante.
- `GROUP BY <atributo(s)>` permite formar consultas agrupadas para extraer información global sobre los grupos formados.
- `HAVING <expresion condicional>` es condición sobre los grupos formados.
- `ORDER BY <columna(s)>` ordena por una o varias columnas.
- `DISTINCT`: No permite la aparición de filas idénticas.
- `<expresion condicional>`: formada por un conjunto de predicados combinados mediante los operadores lógicos `AND`, `OR` y `NOT`.
- Los predicados utilizados permiten comparar columnas:
  - predicados de comparación: =, <>, >, <, >=, <=.
  - predicado `BETWEEN`: permite comprobar si un escalar está en un rango.
  - predicado `IN`: permite comprobar si el valor está dentro de un conjunto.
  - predicado `IS NULL`: permite comprobar si el valor es nulo.

# 2. Estilo requerido para el código SQL

1. Uso de mayúsculas y minúsculas
   *a*) Palabras reservadas del lenguaje (select, on , where, etc.): MAYÚSCULAS
   *b*) Nombres de tablas: minúsculas y singular
   *c*) Nombres de atributos: minúsculas
2. Organización del código
   *a*) Un renglón para todo lo relativo al `SELECT`
   *b*) Un renglón para el `FROM`
   *c*) Un renglón para cada tabla joineada
   *d*) Un renglón para el `WHERE`
   *e*) Un renglón para cada `<expresion condicional>` del WHERE

# GITUNQ

Una empresa nos pide desarrollar una base de datos para un nuevo sistema de control de versiones. Estos sistemas son utilizados para centralizar el desarrollo de un producto de software. Son herramientas de software que ayudan a los equipos de desarrollo a gestionar los cambios en el código fuente a lo largo del tiempo.

El software de control de versiones realiza un seguimiento de todas las modificaciones en el código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden ir hacia atrás en el tiempo y

comparar las versiones anteriores del código para ayudar a resolver el error, al tiempo que se minimizan las interrupciones para todos los miembros del equipo.

La estructura de la base de datos se especifica a continuación:

```
USUARIO < usuario VARCHAR(16) PRIMARY KEY,
    correo VARCHAR(50),
    fecha_nacimiento DATE,
    ciudad VARCHAR(40),
    nyap VARCHAR(30),
    contrasenia VARCHAR(200),
    cantidad_pull_request INT >
REPOSITORIO < nombre VARCHAR(12),
    usuario VARCHAR(16),
    tipo_repositorio VARCHAR(12),
    cantidad_favoritos INT,
    cantidad_pull_request INT >
ARCHIVO < id SERIAL PRIMARY KEY,
    nombre_repositorio VARCHAR(12),
    usuario_repositorio VARCHAR(16),
    contenido VARCHAR(255) >
COMMIT < hash VARCHAR(40) PRIMARY KEY,
    id_archivo INT,
    titulo VARCHAR(30),
    descripcion VARCHAR(200),
    fecha_cambio DATE >
CONTRIBUCION  < hash VARCHAR(40),
    usuario VARCHAR(16),
    cantidad_cambios INT >
```

Dada esta BD, se necesitan realizar las siguientes tareas usando SQL, resolver las consultas planteadas en las siguientes secciones.

## Data Definition Language

1. Genere una base de datos en el motor PostgreSQL cuyo nombre sea tp_su_apellido. No desapruebe por literalidad. Describa los pasos que tuvo que llevar a cabo para lograrlo. Guarde las sentencias que usó para la creación de las tablas en el archivo sql.

2. Escriba las queries para crear las tablas y estructuras de acuerdo a lo descripto más arriba. La fecha de cambio por defecto de un commit debe ser la del momento en la que se inserta, y la cantidad de favoritos y pull requests de un repositorio debe ser 0 así como también la cantidad de pull requests de un usuario.

3. Identifique todas las claves foráneas que correspondan y escriba las queries para crearlas.

4. Limite el atributo fecha_nacimiento a un máximo del 12/12/2015 y el atributo tipo_repositorio a: informativo, empresarial, educacional o comercial.

5. Ejecute las queries de modo tal que todas estas estructuras sean creadas en la base de datos creada en el punto a.

6. Inserte en la base los datos brindados en el archivo githunq.sql. Describa los pasos que tuvo que llevar a cabo para lograrlo, qué metodo usó.

7. Describa los pasos necesarios para que exista la siguiente información en la base de datos: *Queremos un usuario 'pepim' nacido en Solano en el año 1997, que tenga un repositorio informativo con treinta favoritos y cinco pull requests. Luego queremos un usuario 'solan' que haya realizado tres commits en distintos archivos del repositorio de 'pepim'. Uno el 19/10/2015, otro el 28/02/2015 y el último el 29/10/2022*.
   **Aclaración:** los datos no especificados deben ser completados por ustedes mismos para realizar la inserción de datos de forma correcta.

## Data Manipulation Language

1. Obtener nombre, usuario y tipo de los repositorios con más de seis commits de usuarios de la ciudad de Quilmes, ordenados ascendentemente por cantidad de pull requests.
2. Obtener los commits con fecha de cambio luego del 01/10/2021 ordenados descendentemente por hash, ascendentemente por archivo y descendentemente por la fecha de cambio.
3. Obtener los repositorios que solo poseen contribuciones de usuarios de Solano y con edad entre 18 y 21 (deben cumplirse ambas condiciones).
4. Obtener un listado que muestre de cada repositorio, el promedio de contribuciones de usuarios de Quilmes y de Varela.
5. Obtener un listado que muestre de cada archivo, el usuario que más commiteó en el mismo junto con la cantidad de commits.
6. Obtener un listado de las últimas actividades. Que son, por usuario, la suma de contribuciones que hizo y el promedio de la cantidad de cambios, ordenados descendentemente por estas últimas dos.
7. Obtener el nombre de usuario y el repositorio en donde el usuario sea el creador del repositorio pero no haya hecho contribuciones, o haya hecho al menos tres.
8. Obtener la cantidad de repositorios superseguros por ciudad. Que son los que pertenecen a usuarios con contraseña que poseen al menos un '#' numeral, más de 32 caracteres y el usuario hizo más de diez contribuciones, ordenados descendentemente por cantidad de favoritos.
9. Obtener los tres archivos más modificados del 2021 por usuarios que hayan hecho más de 6 pull requests o que posean menos de tres repositorios (puede cumplirse una o ambas condiciones).
10. Obtener de los repositorios *'family friendly'*, el repositorio y la cantidad de contribuidores por edad. Son los repositorios en los que la edad de todos los usuarios que contribuyeron en el mismo es menor a 21.
11. Obtener los más activos. Que son los usuarios que realizaron más commits que el promedio, ordenados ascendentemente por nyap y ciudad, y descendentemente por la cantidad de commits.
12. Obtener de cada repositorio su contribuidor insignia. Que es aquel que más cambios realizó durante sus distintas contribuciones.
13. Tenemos una alta demanda de commits por archivo y fecha de cambio. Aplicar una estrategia para compensar esta alta demanda.
14. Cree una vista(view) de usuarios cuyo promedio histórico de contribuciones en repositorios de Quilmes sea mayor a cinco, commitearon al menos cinco veces en al menos tres repositorios distintos, tienen menos de tres repositorios con con más de cien favoritos y menos de veinte pull requests, y nacieron en la década de los noventa.
15. En la tabla usuario conocemos su usuario, pero tenemos un requerimiento que exige al sistema que la combinación de correos y ciudad sea única. Aplique una estrategia para resolverlo.