Universidad Nacional de Quilmes

BASES DE DATOS
TRABAJO PRÁCTICO - SQL - 2018S2

Pautas

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

(a) Este TP **debe resolverse individualemente**.

(b) Genere el archivo `apellido_nombre_unqfaces.sql`, reemplazando las palabras *apellido_nombre* por los correspondientes al alumno. Ejemplo: `arevalo_gabriela_unqfaces.sql`.

(c) Guarde todas las consultas pedidas en ese archivo.

(d) Guarde las descripciones o aclaraciones en el mismo archivo 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) Envíe el archivo `apellido_nombre_unqfaces.sql` con la resolución del trabajo a `tpi-doc-bd@listas.unq.edu.ar` escribiendo en el Asunto del email *[UNQ - BD] <apellido_nombre> - Entrega TP SQL*, por ejemplo *[UNQ - BD] arevalo_gabriela - Entrega TP SQL*.

(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.

(j) La fecha límite de entrega es el **DIA 22/11/2018** inclusive. Si termina el TP antes de la fecha límite, puede enviarlo también.

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

3. Enunciado

Vista la popularidad que lograron las redes sociales y la invasión a la privacidad a la que han llegado las tradicionales, un grupo de alumnos de la UNQ decidió crear una red social inclusiva y no invasiva, que no haga uso de lo publicado para fines comerciales o similares.

Para ello este grupo comenzó con el diseño de la base de datos.

El primer paso fue el relevamiento de requerimientos. Se pudo identificar que es necesario registrar **usuarios** y **grupos**. Los usuarios pueden o no formar parte de los grupos.

Cada usuario puede hacer **publicaciones** y/o **comentarlas**, como también dar *likes/dislikes* a publicaciones o comentarios (es decir votar positiva o negativamente).

Cada una de estas actividades estará permitida o restringida en función de los grupos. Es decir, si alguien hace una publicación destinada al grupo "DocentesBBDD" solo podrán verla, comentarla o *likearla* miembros de dicho grupo.

Existen grupos abiertos y grupos cerrados. En los grupos abiertos cualquier usuario puede sumarse y participar. Para los grupos cerrados es necesario recibir una invitación previamente.

En resumen, se trata de una mini red social en donde los usuarios pertenecen a diferentes grupos; los usuarios pueden hacer publicaciones pero **sólo lo dentro de los grupos**. No hay publicaciones abiertas (o públicas), la ven aquellos miembros que pertenezcan al grupo en donde el usuario generó la publicación. Luego cualquier miembro del grupo puede comentar y dar *likes* a las publicaciones.

Por supuesto, el usuario es único en el sistema y se identifica con un *username* (texto variable de hasta 30 caracteres) y *password* (texto variable de hasta 30 caracteres). Internamente será indentificado con un *id* auto-numérico. Se deberá registrar con su nombre y apellido (ambos textos variable de hasta 100 caracteres), fecha de nacimiento (formato de fecha) y la carrera que está cursando en la universidad (texto variable de hasta 200 caracteres).

3.1. Relaciones

```
usuario<id SERIAL, nombre VARCHAR(100), apellido VARCHAR(100),
        username VARCHAR(30), contrasenia VARCHAR(30),
        fecha_nacimiento DATE, email, id_carrera INT>

grupo<id SERIAL, nombre_grupo VARCHAR(100), requiere_invitacion BOOL>

grupo_usuario<id_grupo INT, id_user INT>

publicacion<id SERIAL, id_user, id_grupo, titulo, contenido, fecha_publicacion TIMESTAMP>

comentario<id SERIAL, id_public INT, id_user INT, contenido, fecha_comentario TIMESTAMP>

like_publicacion<id_public INT, id_user INT, positivo, fecha TIMESTAMP>

like_comentario<id_coment INT, id_user INT, positivo BOOL, fecha TIMESTAMP>

carrera<id SERIAL, nombre VARCHAR(200)>
```

3.2. Preguntas

1. Generar las sentencias de creación de cada una de las tablas. Identificar en cada caso las restricciones correspondientes de manera que queden establecidas en las sentencias de creación. Algunos campos no muestran el tipo de valor que requieren. **En esos casos es necesario deducir cuál es el tipo adecuado**, la elección podra validarla con el test que será provisto.

2. Haga los cambios necesarios en la estructura para poder registrar alumnos que cursen más de una carrera simultaneamente. Es importante registrar las modificaciones en la estructura.

3. Insertar el usuario `canapedemondongo` con nombre y apellido "Matias Silvestre", password "`Dal41lama`", nacido el 9/11/2000 y que cursa la carrera "Tecnicatura en Programación". Identificar si la carrera se encuentra o no en la base de datos y decidir en tal caso si es necesario insertarla primero o no.

4. **Una vez terminado el punto anterior**, insertar el set de datos provisto en el archivo `datos_unqfaces.sql`

5. Listar la cantidad de alumnos por carrera que hay registrados en unqfaces. Las columnas a mostrar carrera (nombre), cant_alu. El listado debe estar ordenado por el largo del nombre de la carrera de forma descendente

6. Se desea identificar todos los *influencers*. Un *influencer* es un usuario que cuenta con alguna publicación que ha recibido por lo menos 10 *likes* positivos (Sólo es necesario tener en cuenta los *likes/dislikes* de la publicación, no de sus comentarios) y ningún *dislike* en comentarios.

   Deben mostrarse solamente el nombre de usuario, nombre y apellido.

7. Se quiere premiar a los *influencers premium*, los cuales son *influencers* que además no tienen ninguna publicación con más *dislikes* que *likes*. A los campos mostrados en el punto anterior debe agregarle el mail de cada usuario.

8. Listar los primeros 6 grupos con mayor cantidad de *likes* en sus publicaciones. Estos grupos no deben tener registrados a menores de 16 a;os de edad. El listado deberá retornar el nombre del grupo y la cantidad de *likes*.

9. Listar el nombre de usuario y la cantidad de grupos a los que pertenece, ordenado por la cantidad y el nombre de usuario, de todos los usuarios del sistema.

10. Listar todos aquellos comentarios que tengan mas de 3 dislikes. El listado debe proveer contenido del comentario, el contenido de la publicacion , la cantidad de likes y la cantidad de dislikes, ordenado por mayor cantidad de dislikes y por la fecha del comentario.

11. Listar todas las publicaciones que tengan dislikes. El reporte debe tener el nombre y apellido del autor y el título, el contenido y la fecha de la publicación.

12. Hacer una lista de todos los grupos donde se puede visualizar el promedio de edad de sus miembros, asi como la edad del miembro mas joven y el del mayor.

13. Se requiere identificar a los usuarios que hayan hecho like unicamente a usuarios que no cursen su carrera.

14. Listar el último comentario hecho por cada usuario (en cualquier publicación de cualquier grupo).

15. Listar a todos los usuarios sin actividad, es decir, aquellos que no hayan hecho publicaciones, comentarios ni likes de ningun tipo. El listado debe mostrar la o las carreras que cursa, nombre, apellido, username y mail. Si cursa mas de una carrera estas deben mostrarse en UNA sola fila

16. Generar un listado de todos los usuarios (solo dos 2 campos): nombre y apellido y edad.

17. Crear una vista que muestre la última publicación de cada usuario, mostrando username, contenido y fecha de la publicación.

18. De la última publicación debe mostrarse el contenido, username del autor, fecha y hora, con la cantidad de comentarios hechos, la cantidad de likes y cantidad de dislikes hasta el momento.

19. Listar a los usuarios (username) con la cantidad de likes y la cantidad de dislikes que haya recibido tanto en publicaciones como en comentarios. Además, agregar una columna con la diferencia entre likes y dislikes (si da positivo es porque tiene más likes que dislikes), ordenada de menor a mayor diferencia.

20. Para evitar cualquier caso de acoso en esta red, se considera necesario generar un sistema de registro de logs mediante *triggers* que deben alimentar las tablas `log_publicacion` que registre `id_log`, `fecha_log`, `accion_log`, `id`, `id_user`, `id_grupo`, `titulo`, `contenido`, `fecha_publicacion` y otra tabla `log_comentario` con `id_log`, `fecha_log`, `accion_log`, `id`, `id_public`, `id_user`, `contenido`, `fecha_comentario`, donde en cambos casos `id_log` es un número incremental y PK de las tablas, `accion_log` debe registrar `I` en caso de inserts, `U` en caso de updates y `D` en caso de deletes.

21. Insertar al menos 2 comentarios y al menos 2 publicaciones para comprobar el funcionamiento del sistema de logs creado en el punto anterior.