Proyecto de Sistemas de Bases de Datos I

Plataforma de crowdfunding para proyectos sociales

Sistemas de Bases de Datos I
Primer Término 2022-2023
ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

Índice

Tabla de contenido

Integrantes

- CAMPODONICO ROMERO JESSE AARON
- MOREIRA MERINO MARCOS ALEJANDRO
- MURILLO TORRES JOSE RICARDO

1. Descripción de los Triggers

Se han creado dos triggers para el sistema:

1. Trigger para Donación_Pago: Este trigger se activa automáticamente al insertar una nueva donación en la tabla Donación_Pago. Su propósito es registrar automáticamente la fecha actual cuando se realiza una donación.

```
-- Trigger para Donación_Pago table: Setear automaticamente la fecha
DELIMITER //
CREATE TRIGGER tr_set_fecha_donacion
BEFORE INSERT ON Donación_Pago
FOR EACH ROW
BEGIN
    SET NEW.fecha = NOW();
END;
//
```

2. Trigger para Comentario: Este trigger se activa al insertar un nuevo comentario en la tabla Comentario. Al igual que el trigger anterior, su función es registrar automáticamente la fecha actual cuando se añade un comentario.

```
-- Trigger para Comentario table: Setear automaticamente la fecha
DELIMITER //
CREATE TRIGGER tr_set_fecha_comentario
BEFORE INSERT ON Comentario
FOR EACH ROW
BEGIN
    SET NEW.fecha = NOW();
END;
//
DELIMITER ;
```

2. Uso de Reportes

Se han creado cuatro vistas para generar reportes en el sistema. Estas vistas combinan información de diferentes tablas para proporcionar reportes descriptivos que faciliten la toma de decisiones y el análisis de datos.

```
-- 1. Vista de Usuarios y sus Campañas:
CREATE VIEW Usuarios_Campañas AS
SELECT
    Usuario.ID AS ID_Usuario,
    Usuario.nombre AS Nombre_Usuario,
    Campaña.ID AS ID_Campaña,
    Campaña.nombre AS Nombre_Campaña
FROM
    Usuario JOIN Campaña ON Usuario.ID_campaña = Campaña.ID;
-- 2. Vista de Comentarios por Campaña:
CREATE VIEW Comentarios_Por_Campaña AS
SELECT
    Comentario.ID AS ID_Comentario,
    Comentario.contenido AS Contenido_Comentario,
    Campaña.ID AS ID_Campaña,
    Campaña.nombre AS Nombre_Campaña
FROM
    Comentario JOIN Campaña ON Comentario.ID_campaña = Campaña.ID;
-- 3. Vista de Donaciones por Campaña:
CREATE VIEW Donaciones_Por_Campaña AS
SELECT
    Donación_Pago.ID AS ID_Donación,
    Donación_Pago.monto AS Monto_Donación,
    Campaña.ID AS ID_Campaña,
    Campaña.nombre AS Nombre_Campaña
FROM
    Donación_Pago JOIN Campaña ON Donación_Pago.ID_campaña = Campaña.ID;


-- 4. Vista de Usuarios y sus Donaciones:
CREATE VIEW Usuarios_Donaciones AS
SELECT
    Usuario.ID AS ID_Usuario,
    Usuario.nombre AS Nombre_Usuario,
    Donación_Pago.ID AS ID_Donación,
    Donación_Pago.monto AS Monto_Donación
FROM
    Usuario JOIN Donación_Pago ON Usuario.ID = Donación_Pago.ID_usuario;
```

3. Procesos almacenados

Se han creado procesos almacenados para las operaciones básicas de inserción, actualización y eliminación de registros en las tablas principales del sistema. Estos procesos almacenados no solo hacen que las operaciones sean más eficientes, sino que también proporcionan una capa adicional de seguridad al evitar la inyección de SQL.

SP 1

Este proceso almacecnado (SP) llamado “InsertUsuario” se diseñó para insertar nuevos registros en la tabla “Usuario”.

Este SP proporciona una forma segura y eficiente de insertar nuevos usuarios en la base de datos, garantizando que el email proporcionado sea válido y manejando errores de forma adecuada. Además, al utilizar transacciones, se asegura de que la base de datos se mantenga en un estado consistente incluso si ocurren errores durante la inserción.

Aquí está su funcionamiento desglosado:

1. Parámetros de Entrada:

- “nombre_param”, “email_param”, “contraseña_param”, “tarjetaDebito_param”, “ID_tipo_usuario_param”, “ID_campaña_param”: Estos son los parámetros que se esperan recibir cuando se llama al SP. Corresponden a los campos que se desean insertar en la tabla “Usuario”.

2. Manejo de Errores:

- Se declara un manejador de errores (EXIT HANDLER) para las excepciones SQL. Si ocurre algún error durante la ejecución del SP, este manejador se activará.

- En caso de error, se realiza un “ROLLBACK”, lo que deshace cualquier cambio hecho en la base de datos durante la transacción.

- Luego, se lanza una señal (“SIGNAL”) con un mensaje de error específico.

3. Transacción:

- Se inicia una transacción con “START TRANSACTION”. Esto significa que cualquier cambio hecho en la base de datos no se confirmará hasta que se ejecute el comando “COMMIT”.

4. Validación del Email:

- Se verifica si el parámetro “email_param” contiene el carácter '@' usando la función “LOCATE”. Si no lo contiene, se considera que el email no es válido.

- Si el email no es válido, se lanza una señal con un mensaje de error específico.

5. Inserción en la Base de Datos:

- Si el email es válido, se procede a insertar el nuevo registro en la tabla "Usuario" usando el comando "INSERT INTO".

- Una vez que la inserción ha sido exitosa, se confirman todos los cambios en la base de datos con el comando "COMMIT".

```
-- 1. SP para insertar datos en la tabla Usuario:
DELIMITER //
CREATE PROCEDURE InsertUsuario(IN nombre_param VARCHAR(100), IN email_param
VARCHAR(100), IN contraseña_param VARCHAR(100),
                                IN tarjetaDebito_param VARCHAR(100), IN
ID_tipo_usuario_param INT, IN ID_campaña_param INT)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error al insertar en Usuario';
    END;

    START TRANSACTION;
    IF LOCATE('@', email_param) = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Email no válido';
    ELSE
        INSERT INTO Usuario(nombre, email, contraseña, tarjetaDebito, ID_tipo_usuario,
ID_campaña)
        VALUES (nombre_param, email_param, contraseña_param, tarjetaDebito_param,
ID_tipo_usuario_param, ID_campaña_param);
        COMMIT;
    END IF;
END;
//DELIMITER;
```

SP 2

Este Proceso almacenado(SP) llamado "UpdateUsuario" está diseñado para actualizar registros existentes en la tabla "Usuario". Permite actualizar registros de usuario de manera eficiente, asegurando que el email proporcionado sea válido y manejando errores de forma adecuada. Al igual que con el SP de inserción, la transacción garantiza que la base de datos se mantenga en un estado consistente, incluso si se encuentran errores durante la actualización.

Vamos a desglosar su funcionamiento:

1. Parámetros de Entrada:

- "ID_param", "nombre_param", "email_param", "contraseña_param", "tarjetaDebito_param", "ID_tipo_usuario_param", "ID_campaña_param": Estos son los parámetros que se espera recibir cuando se llama al SP. El parámetro "ID_param" determinará qué registro de usuario se actualizará, mientras que los otros parámetros representan los nuevos valores para ese registro.

2. Manejo de Errores:

- Similar al primer SP, se declara un manejador de errores para las excepciones SQL.

- En caso de error, se realiza un "ROLLBACK" para revertir cualquier cambio hecho en la base de datos durante la transacción.

- Luego, se lanza una señal ("SIGNAL") con un mensaje de error específico.

3. Transacción:

- Se inicia una transacción con "START TRANSACTION", lo que asegura que todos los cambios en la base de datos como parte de esta operación sean tratados como una única unidad.

4. Validación del Email:

- Se verifica si el parámetro "email_param" contiene el carácter '@' usando la función "LOCATE". Si no lo contiene, se considera que el email no es válido y se lanza una señal con un mensaje de error.

5. Actualización en la Base de Datos:

- Si el email es válido, se procede a actualizar el registro de usuario especificado por "ID_param" en la tabla "Usuario" usando el comando "UPDATE".

- Una vez que la actualización ha sido exitosa, se confirman todos los cambios en la base de datos con el comando "COMMIT".

```
-- 2. SP para actualizar datos en la tabla Usuario:
DELIMITER //
CREATE PROCEDURE UpdateUsuario(IN ID_param INT, IN nombre_param VARCHAR(100), IN
email_param VARCHAR(100), IN contraseña_param VARCHAR(100),
                               IN tarjetaDebito_param VARCHAR(100), IN
ID_tipo_usuario_param INT, IN ID_campaña_param INT)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error al actualizar Usuario';
    END;
    START TRANSACTION;
    IF LOCATE('@', email_param) = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Email no válido';
    ELSE
        UPDATE Usuario
        SET nombre = nombre_param, email = email_param, contraseña =
contraseña_param, tarjetaDebito = tarjetaDebito_param,
            ID_tipo_usuario = ID_tipo_usuario_param, ID_campaña = ID_campaña_param
        WHERE ID = ID_param;
        COMMIT;
    END IF;
END;
DELIMITER;
```

SP 3

Este Proceso almacenado (SP) llamado "DeleteUsuario" está diseñado para eliminar registros específicos de la tabla "Usuario".

Este SP proporciona una manera eficiente y segura de eliminar registros de usuario basados en su ID. La utilización de transacciones garantiza que, en caso de un error, cualquier cambio hecho como parte de la operación sea revertido, asegurando la integridad de los datos en la base de datos.

Vamos a analizar su funcionamiento:

1. Parámetros de Entrada:

- "ID_param": Este es el único parámetro que se espera recibir cuando se llama al SP. Determinará qué registro de usuario se eliminará basado en el ID del usuario.

2. Manejo de Errores:

- Se declara un manejador de errores para las excepciones SQL.

- En caso de error durante la eliminación, se realiza un "ROLLBACK", lo que revierte cualquier cambio hecho en la base de datos durante la transacción.

- Luego, se lanza una señal ("SIGNAL") con un mensaje de error específico, indicando que hubo un error al intentar eliminar un registro de la tabla "Usuario".

3. Transacción:

- Se inicia una transacción con "START TRANSACTION". Las transacciones aseguran que todas las operaciones de base de datos como parte de esta operación sean tratadas como una única unidad y que la base de datos se mantenga en un estado consistente.

4. Eliminación en la Base de Datos:

- Se utiliza el comando "DELETE FROM" para eliminar el registro de usuario especificado por "ID_param" en la tabla "Usuario".

- Una vez que la eliminación ha sido exitosa, se confirman todos los cambios en la base de datos con el comando "COMMIT".

```
-- 3. SP para eliminar datos de la tabla Usuario:
DELIMITER //
CREATE PROCEDURE DeleteUsuario(IN ID_param INT)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error al eliminar
de Usuario';
    END;

    START TRANSACTION;
    DELETE FROM Usuario WHERE ID = ID_param;
    COMMIT;
END;
DELIMITER;
```

4. Índices

Se han añadido varios índices en la base de datos para optimizar las operaciones de consulta. Los índices se han añadido en campos que se utilizan frecuentemente para buscar registros, lo que acelera estas operaciones. Además, los índices también facilitan la organización de los datos en la base de datos y mejoran la eficiencia general del sistema.

Al tener índices en columnas que se utilizan con frecuencia en consultas o condiciones WHERE, se reduce significativamente el tiempo que toma a la base de datos buscar y recuperar registros relevantes.

1. "CREATE INDEX idx_usuario_email ON Usuario(email);"

- Este índice se crea en la columna "email" de la tabla "Usuario".

- Propósito: Facilitar y acelerar las búsquedas y operaciones que involucren el email del usuario. Es común que las operaciones de autenticación o validación utilicen el email, por lo que tener un índice aquí es beneficioso.

2. "CREATE INDEX idx_campaña_id ON Campaña(ID);"

- Este índice se crea en la columna "ID" de la tabla "Campaña".

- Propósito: Mejorar la eficiencia de las operaciones que involucren la identificación de una campaña específica. Aunque el ID suele ser una clave primaria y ya tiene un índice por defecto, asegurarse de que hay un índice explícito puede ser útil en ciertos sistemas de bases de datos.

3. "CREATE INDEX idx_comentario_id_campaña ON Comentario(ID_campaña);"

- Este índice se crea en la columna "ID_campaña" de la tabla "Comentario".

- Propósito: Optimizar las consultas que buscan comentarios asociados con una campaña específica. Es probable que se quieran ver todos los comentarios de una campaña en particular, por lo que tener un índice en esta columna acelerará estas consultas.

4. "CREATE INDEX idx_donacion_id_campaña ON Donación_Pago(ID_campaña);"

- Este índice se crea en la columna "ID_campaña" de la tabla "Donación_Pago".

- Propósito: Mejorar la eficiencia de las consultas que buscan donaciones asociadas con una campaña específica. Similar al índice anterior, pero para donaciones en lugar de comentarios.

5. "CREATE INDEX idx_donacion_id_usuario ON Donación_Pago(ID_usuario);"

- Este índice se crea en la columna "ID_usuario" de la tabla "Donación_Pago".

- Propósito: Facilitar y acelerar las consultas que buscan todas las donaciones hechas por un usuario específico. Esto puede ser útil para generar historiales de donaciones por usuario o para análisis de comportamiento de donantes.

5. Permisos por Usuario

Estos comandos SQL están relacionados con la gestión de usuarios y permisos en una base de datos llamada "Crowfounding". Veamos qué hace cada uno:

1. Usuario "admin":

- Se crea un usuario llamado "admin" con la contraseña "admin123".

- Se le otorgan todos los privilegios sobre la base de datos "Crowfounding". Esto significa que "admin" puede hacer cualquier operación en esta base de datos, incluyendo crear, leer, actualizar, eliminar registros, gestionar otros usuarios, etc.

2. Usuario "gestor_campañas":

- Se crea un usuario llamado "gestor_campañas" con la contraseña "gestor123".

- Se le otorgan permisos para insertar, actualizar y eliminar registros en la tabla "Campaña" de la base de datos "Crowfounding".

- También tiene permiso para seleccionar registros en la vista o tabla "Usuarios_Campañas" (esto supone que hay una vista o tabla con este nombre que relaciona usuarios con campañas).

3. Usuario "lector":

- Se crea un usuario llamado "lector" con la contraseña "lector123".

- Se le otorga permiso para seleccionar (leer) registros en todas las tablas y vistas de la base de datos "Crowfounding".

4. Usuario "gestor_donaciones":

- Se crea un usuario llamado "gestor_donaciones" con la contraseña "donaciones123".

- Se le otorgan permisos para insertar y actualizar registros en la tabla "Donación_Pago" de la base de datos "Crowfounding".

- También tiene permiso para ejecutar el Proceso almacenado"InsertUsuario" de la base de datos "Crowfounding".

5. Usuario "gestor_comentarios":

- Se crea un usuario llamado "gestor_comentarios" con la contraseña "comentarios123".

- Se le otorga permiso para insertar registros en la tabla "Comentario" de la base de datos "Crowfounding".

- También tiene permiso para seleccionar registros en la vista o tabla "Comentarios_Por_Campaña" de la base de datos "Crowfounding".

```
-- User admin
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin123';
GRANT ALL PRIVILEGES ON Crowfounding.* TO 'admin'@'localhost';

-- User gestor_campañas
CREATE USER 'gestor_campañas'@'localhost' IDENTIFIED BY 'gestor123';
GRANT INSERT, UPDATE, DELETE ON Crowfounding.Campaña TO
'gestor_campañas'@'localhost';
GRANT SELECT ON Crowfounding.Usuarios_Campañas TO
'gestor_campañas'@'localhost';

-- User lector
CREATE USER 'lector'@'localhost' IDENTIFIED BY 'lector123';
```

```
GRANT SELECT ON Crowfounding.* TO 'lector'@'localhost';

-- User gestor_donaciones
CREATE USER 'gestor_donaciones'@'localhost' IDENTIFIED BY
'donaciones123';
GRANT INSERT, UPDATE ON Crowfounding.Donación_Pago TO
'gestor_donaciones'@'localhost';
GRANT EXECUTE ON PROCEDURE Crowfounding.InsertUsuario TO
'gestor_donaciones'@'localhost';

-- User gestor_comentarios
CREATE USER 'gestor_comentarios'@'localhost' IDENTIFIED BY
'comentarios123';
GRANT INSERT ON Crowfounding.Comentario TO
'gestor_comentarios'@'localhost';
GRANT SELECT ON Crowfounding.Comentarios_Por_Campaña TO
'gestor_comentarios'@'localhost';
```

A continuación, se presenta una tabla que especifica los permisos otorgados a cada usuario en el sistema.

Usuario	Contraseña	Permiso	Descripción
admin	admin123	ALL PRIVILEGES	Tiene todos los privilegios sobre la base de datos Crowfounding.
gestor_campañas	gestor123	INSERT, UPDATE, DELETE, SELECT	Tiene todos los privilegios sobre la base de datos Crowfounding.
lector	lector123	SELECT	Puede leer registros en todas las tablas y vistas de Crowfounding.
gestor_donaciones	donaciones123	INSERT, UPDATE, EXECUTE	Puede insertar y actualizar registros en la tabla Donación_Pago. Puede ejecutar el Proceso almacenadoInsertUsuario.
gestor_comentarios	comentarios123	INSERT, SELECT	Puede insertar registros en la tabla Comentario. Puede leer registros en la vista o tabla Comentarios_Por_Campaña.