ADMINISTRACIÓN DE BASE DE DATOS CON POSTGRESQL LABORATORIO 4. TRANSACCIONES Y PROCEDIMIENTOS ALMACENADOS

Luis Antonio Álvarez Oval
loval@unach.mx

Christian Mauricio Castillo Estrada
cmce@unach.mx

Aron De La Cruz Vázquez
aron.cruz@unach.mx

Facultad De Contaduría Pública C-IV Universidad Autónoma de Chiapas

Para citar este artículo:
Álvarez, L., Castillo, C. y De la Cruz, A. (2015) Administración de base de datos con postgreSQL laboratorio 4. Transacciones y procedimientos almacenados. *Espacio I+D Innovación más Desarrollo 4* (9) 172-199. Recuperado de http://espacioimasd.unach.mx/suplemento/espacioimasd_es_9.pdf

RESUMEN

La serie de laboratorios de Administración de Bases de Datos con PostgreSQL, muestra de forma práctica la administración de este tipo de sistemas, el cual tiene un amplio uso en la industria de desarrollo de software. Mientras que las bases de datos es la herramienta que requieren todas las empresas que necesitan almacenar la información que generan, y es en este tipo de sistemas donde se guarda ésta. De ahí la importancia de entender y aplicar los conceptos de administración estándares que se usan en la industria. Se usa el sistema PostgreSQL debido a que ofrece los mecanismos que tienen otros sistemas similares pero de carácter propietario. PostgreSQL se ofrece bajo una licencia PostgresSQL, lo que permite desde el punto de vista del propietario de un sistema de información evitar el pago de costosas licencias por el uso de una base de datos.

Palabras Clave.
Administración de Base de Datos, SQL, Programación de procedimientos almacenados, postgreSQL.

Esta cuarta entrega de una serie de seis laboratorios de Administración de Base de Datos (ABD) enseña el uso de los procedimientos almacenados asociados a las transacciones en la base de datos. Este trabajo no pretende enseñar a programar procedimientos almacenados en la base de datos, por lo que el lector debe de investigar el tema denominado Lenguaje de Procedimientos de SQL para PostgreSQL (SQL Procedural Language, mejor conocido como PLPGSQL). Con relación a este tema, debo de comentar que ya estamos trabajando con algunos docentes en el desarrollo de tres laboratorios en los que enseñaremos la programación de procedimientos almacenados, así que estén pendientes. Para implementar la experiencia de las transacciones se hace uso de programas desarrollados en lenguaje de programación Java, además se requiere, que los equipos usados para desarrollar este experimento se encuentren conectados en una red local. Sin embargo, no es del alcance de este trabajo el enseñar la conexión y configuración de una red, ni se pretende enseñar conceptos de programación orientada a objetos con Java.

Los laboratorios se han diseñado para proporcionar los conceptos y la experiencia necesarios para conocer detalladamente el sistema, se aprovecha la función de “copiar y pegar” que nos ofrece el sistema operativo para disminuir el esfuerzo del lector en la preparación del ambiente de trabajo y en la solución de los problemas. En la sección denominada “trabajo adicional” se requiere que el lector aplique la experiencia obtenida en la solución de problemas relacionados al tema central del laboratorio. La sección de conceptos básicos se muestra la sintaxis de los comandos y da algunas explicaciones del uso de los mismos, este material ha sido tomado del manual de usuario del sistema PostgreSQL el cual está disponible en la página oficial de la herramienta, en algunos casos se ha tomado del sitio oficial en Español. Los conceptos básicos se aplican en torno al mismo proyecto que usaremos en esta serie: “Universidad ACME”, el cual es producto de la imaginación del autor, así como la solución práctica de los problemas planteados. Los libros que se ofrecen en la sección de referencias, sirven como consulta para apoyar algunos de los conceptos que

se aplican en la solución práctica de problemas de administración de base de datos.

Estos laboratorios se han preparado para procurar experiencia práctica a los estudiantes de la materia Administración de Base de Datos de la Licenciatura en Sistemas Computacionales que se ofrece en la Facultad de Contaduría Pública (FCP) del Campus IV de la Universidad Autónoma de Chiapas (UNACH). En la FCP tenemos al menos 14 años de experiencia en el uso de PostgreSQL en las aulas, proyectos de investigación y en sistemas que se han implementado para la automatización de las actividades cotidianas de la FCP. Como producto de esa experiencia académica e industrial se han obtenido estos laboratorios que se usan en las aulas para capacitar a nuestros estudiantes. También se tiene noticia de que son una fuente de consulta para egresados que laboran en el sector empresarial.

Como se ha mencionado previamente la herramienta tiene características y lenguajes de programación estándar que ofrecen sistemas propietarios, por lo que los ejemplos fácilmente pueden ser aplicados en otros sistemas de bases de datos del mercado, o pueden ser referencia para aplicar los conceptos en proyectos industriales. Por lo que puedan servir como consulta a profesionales de las Ciencias de la Computación.

OBJETIVO

El lector aprenderá a usar Procedimiento Almacenados y su relación con las transacciones en la Base de Datos.

PRERREQUISITOS

Se espera que el lector tenga experiencia previa en el uso y conversión de diagramas Entidad-Relación (E-R), los temas asociados al Diseño de Base de Datos no se cubren en este documento. También se espera que

el lector tenga conocimientos de programación en cualquier lenguaje de programación y si necesita información adicional del PLPGSQL, se sugiere que visite el sitio: http://www.postgresql.org/docs/9.3/static/plpgsql.html, o busque esta información en el libro "PostgreSQL" de los autores Susan y Korry Douglas (ISBN: 0672327562).

También se espera que el lector tenga experiencia en la conexión de redes locales y en la instalación y programación del lenguaje de programación Java. Oracle, la empresa que es propietaria de Java, ofrece una guía para instalar este lenguaje:

https://www.java.com/es/download/help/download_options.xml

Finalmente, es necesario instalar la base de datos PostgreSQL versión 9.3 sobre el sistema operativo Windows o Linux, verifique los requerimientos para instalación en la página oficial de la herramienta: www.postgresql.org. El sistema puede descargarse del sitio Web:

http://www.enterprisedb.com/products-services-training/pgdownload#windows

Si tiene alguna duda con respecto a PostgreSQL, se recomiendo visitar el sitio oficial con información publicada en idioma español: http://www.postgresql.org.es/primeros_pasos

PARTES QUE COMPONEN ESTE LABORATORIO

1. Proyecto a desarrollar
2. Conceptos básicos
3. Preparación del ambiente de trabajo
4. Problemática a resolver
5. Trabajo adicional
6. Referencias

1. PROYECTO A DESARROLLAR

El ejercicio que se va a realizar consiste en un proyecto que describe el problema de una empresa dedicada a la prestación de servicios educativos: después de leer el texto se genera el diagrama E-R con la solución a este problema, se continúa con la creación de las tablas y población de las tablas, para finalmente trabajar con los permisos de grupos y usuarios.

Proyecto Universidad ACME

En UACME, se ofrecen dos tipos de cursos en el periodo especial de verano, en el cual se imparten cursos de verano y cursos extracurriculares. Los primeros son materias que un alumno regular que estudia una carrera cursa en este periodo, se le permite adelantar hasta dos materias; mientras que los segundos son cursos especiales de capacitación que se ofrecen a alumnos regulares como estudiantes o profesionistas externos.

Los docentes de la UACME, son los únicos a los que se les permite impartir estos cursos, por los cuales recibe un pago adicional, se les paga de acuerdo a un tabulador que indica el costo de la hora de estos cursos de acuerdo al nivel académico del docente. El pago se genera a partir del alta del curso y solo se permite expedir un cheque por cada curso. Además los estudiantes deben acudir a pagar adicionalmente al costo del semestre por asistir a ellos.

UACME tiene dos departamentos que intervienen en la administración de los cursos:

A) Departamento de Administración (DA) y B) Departamento de Control Escolar (DCE). Corresponde al DA, efectuar el pago a los docentes y los cobros a los alumnos. El DA es dirigido por el C.P. Ávila y es auxiliado por el Sr. Cancino. Mientras que el DCE, es dirigido por el Lic. Barroso y auxiliado por los Sras. Tirado, Martínez, Aquino y Ramos y es en este donde se decide cuales cursos se imparten en el periodo, quién los imparte, y se aceptan las solicitudes de los alumnos. Un caso especial, es el de los Profesores, ya que el DA es quién les puede modificar el

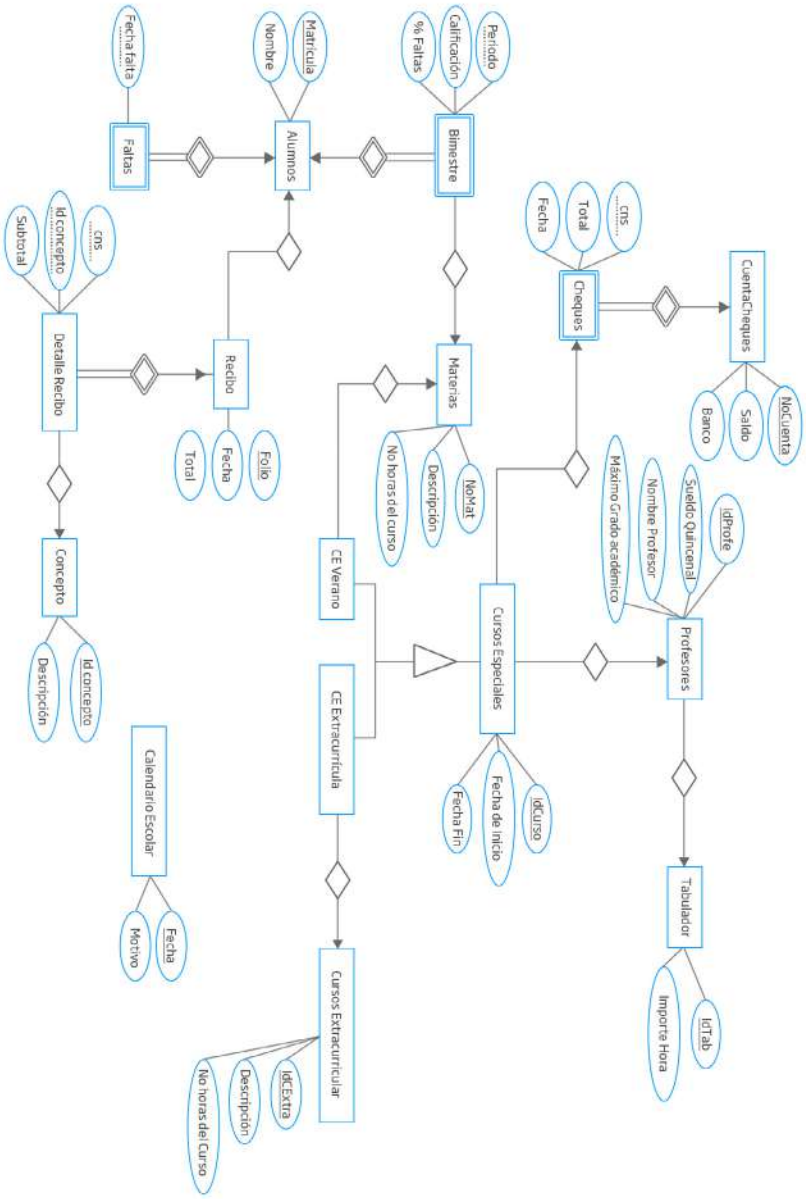
sueldo quincenal, mientras que el DCE ni siquiera puede visualizar éste. Lo curioso radica en que, es el DCE quién acepta los docentes y los registra en el sistema, pero es el DA donde se captura el sueldo. Importante es para la administración de la UACME que esta política se aplique al pie de la letra, y que sea implementado directamente sobre la DB. A continuación se describe detalladamente las tablas a las que se le permite el acceso al personal de la Secretaría Administrativa: CuentaCheques, Cheque, Tabulador, Profesores, Concepto, Recibo, y DetalleRecibo.

Como casos especiales, este departamento podrá acceder a consultar las tablas de Cursos Especiales, Cursos Especiales Verano, Cursos Especiales Extracurriculares, Cursos Extracurriculares y Materias. Explícitamente no se les permite modificar ningún campo o registro.

Tablas a las que se le permite el acceso al personal de la Secretaría Escolar:

CursosEspeciales, CursosExtracurricular, Materias, CEVerano, CEExtracurricula, Alumnos, Bimestre, Faltas, CalendarioEscolar.

Figura 1. Diagrama E/R que resuelve el problema anterior.



2.CONCEPTOS BÁSICOS

Transacción

Una transacción es una ejecución de un programa de usuario, visto por el DBMS como una serie de operaciones *lecturas y escrituras*, la cual accede a la base de datos que es compartida por varios usuarios en forma simultánea. Es una colección de acciones que hacen transformaciones de los estados de un sistema preservando la *consistencia* del mismo. Una base de datos está en un estado *consistente* si obedece todas las restricciones de integridad definidas sobre ella. Los cambios de estado ocurren debido a actualizaciones, inserciones, y supresiones de información. Por supuesto, se quiere asegurar que la base de datos nunca entra en un estado de inconsistencia. Sin embargo, durante la ejecución de una transacción, la base de datos puede estar temporalmente en un estado inconsistente. El punto importante aquí es asegurar que la base de datos regresa a un estado consistente al fin de la ejecución de una transacción.

Lo que se persigue con el uso de transacciones es por un lado contar con una transparencia adecuada de las acciones concurrentes a una base de datos y por el otro tener una transparencia adecuada en el manejo de las fallas que se pueden presentar en una base de datos.

Consistencia: una transacción es un programa correcto que lleva la base de datos de un estado consistente a otro con la misma característica. Gracias a esto, las transacciones no violan las reglas de integridad de una base de datos.

Aislamiento: una transacción en ejecución no puede revelar sus resultados a otras transacciones concurrentes antes de su compromiso (**commit**). Más aún, si varias transacciones se ejecutan concurrentemente, los resultados deben ser los mismos que si ellas se hubieran ejecutado de manera secuencial (seriabilidad). La seriabilidad consiste en asegurarse que los cambios siguen un orden adecuado.

Atomicidad: una transacción es tratada como una unidad de operación. Por lo tanto todas las acciones de la transacción se llevan a cabo o ninguna de ellas se realiza .La atomicidad requiere que si una transacción se interrumpe por una falla, sus resultados parciales deben ser deshechos. Se efectúan todas las transacciones, pero en caso de fallas no se realiza ninguna. Una transacción debe concluir comprometida o abortada. En el caso del compromiso se instalan todas las actualizaciones y en el aborto se descartan todas las actualizaciones.

Durabilidad: es la propiedad de las transacciones que asegura que una vez que una transacción realiza su compromiso, sus resultados son permanentes y no pueden ser borrados de la base de datos, se asegura que los resultados de una transacción sobrevivirán a fallas del sistema. Una transacción siempre termina, aun en la presencia de fallas. Si una transacción termina de manera exitosa se dice que la transacción hace un compromiso. Si la transacción se detiene sin terminar su tarea, se dice que la transacción aborta. Cuando la transacción es abortada, puede ser por distintas razones relacionadas con la naturaleza de la transacción misma, o por conflicto con otras transacciones o por fallo de un proceso o computador, entonces su ejecución es detenida y todas las acciones ejecutadas hasta el momento son deshechas regresando a la base de datos al estado antes de su ejecución. A esta operación también se la conoce como restauración (*rollback*).

Un candado es un mecanismo de control de acceso concurrente a un dato. Los datos pueden tener candados en dos modos:

- Modo exclusivo (X). El dato puede ser leído y escrito. Un candado en este modo se solicita con la instrucción lock-X
- Modo compartido (S). El dato solo puede ser leído. Un candado en este modo se solicita con la instrucción lock-S.

Las peticiones por candados se hacen al administrador de control de concurrencia. La transacción puede proceder solo después de que una solicitud es otorgada.

- Un candado es un mecanismo de control de acceso concurrente a un dato
- Los datos pueden tener candados en dos modos:
    - ◇ Modo exclusivo (X). El dato puede ser leído y escrito. Un candado en este modo se solicita con la instrucción lock-X
    - ◇ Modo compartido (S). El dato solo puede ser leído. Un candado en este modo se solicita con la instrucción lock-S.
- Las peticiones por candados se hacen al administrador de control de concurrencia. La transacción puede proceder solo después de que una solicitud es otorgada.
- Un candado es otorgado si el candado solicitado es compatible con otros candados previamente otorgados.
- Se pueden tener varios candados compartidos sobre un dato, pero sólo un candado exclusivo
- Si un candado no puede ser concedido, la transacción que lo solicita debe esperar a que todos los candados incompatibles sean liberados.
- El poner candados no es suficiente para garantizar seriabilidad.
- Ejemplo: si A se actualiza después de la lectura de B, la suma será incorrecta
- Un protocolo basado en candados es un conjunto de reglas seguidas por todas las transacciones para solicitar y liberar candados.

Protocolo de candado en dos fases (2PL): La importancia de los candados de dos fases es que se ha demostrado de manera teórica que

todos los planificadores generados por algoritmos de control de concurrencia que obedecen a los candados de dos fases son serializables.

Abortos en Cascada: Puede suceder si una transacción aborta después de liberar un candado, otras transacciones que hayan accedido el mismo elemento de datos aborten también. Para evitar lo anterior, los despachadores para candados de dos fases implementan lo que se conoce como los **candados estrictos de dos fases** en los cuales se liberan todos los candados cuando la transacción termina.

Comandos del SQL que se usan en este laboratorio

BEGIN —Comienza una transacción en modo encadenado
Sintaxis
BEGIN [WORK | TRANSACTION]

Entradas
WORK
TRANSACTION

Palabras clave opcionales. No tienen efecto.

Salidas
BEGIN - Esto significa que una nueva transacción ha sido comenzada.
NOTICE: BEGIN: “already a transaction in progress” - Esto indica que una transacción ya está en progreso. La transacción en curso no se ve afectada.

Descripción
Por omisión, PostgreSQL ejecuta las transacciones en modo no encadenado (también conocido como “autocommit” en otros sistemas de base de datos). En otras palabras, cada estado de usuario es ejecutado en su propia transacción y un compromiso se ejecuta implícitamente al

final de cada comando (si la ejecución fue exitosa, de otro modo se ejecuta una restauración). BEGIN inicia una transacción de usuario en modo encadenado, i.e. Todos los estados de usuarios después de un comando BEGIN se ejecutaran en una transacción única hasta un explícito COMMIT, ROLLBACK, o aborte la ejecución. Los estados en modo encadenado se ejecutan mucho más rápido, porque la transacción arranque/compromiso (start/commit) requiere una actividad significativa de CPU y de disco. La ejecución de múltiples estados dentro de una transacción también es requerida para la consistencia cuando se cambian muchas tablas relacionadas. El nivel de aislamiento por omisión de las transacciones en PostgreSQL es READ COMMITTED, donde las consultas dentro de la transacción solo tienen en cuenta los cambios consolidados antes de la ejecución de la consulta. Así pues, debe utilizar SET TRANSACTION ISOLATION LEVEL SERIALIZABLE justo después de BEGIN si necesita aislamiento de transacciones más riguroso. Las consultas del tipo SERIALIZABLE solo tendrán en cuenta los cambios consolidados antes de que la transacción entera comience (realmente, antes de la ejecución del primer estado DML en una transacción serializable).

Si la transacción está consolidada, PostgreSQL asegurará que todas las actualizaciones sean hechas o si no que ninguna de ellas lo sea. Las transacciones tienen la propiedad estándar ACID (atómica, consistente, aislada y durable).

SELECT — Recupera registros desde una tabla o vista. Este comando es el mismo que usa para efectuar consultas.

Sintaxis

```
SELECT [ALL | DISTINCT [ON (expression [, ...] ) ] ]
expression [ AS name ] [, ...]
[INTO [TEMPORARY | TEMP ] [ TABLE ] new_table ]
[FROM table [alias ] [, ...] ]
[WHERE condition ]
```

[GROUP BY column [, ...]]
HAVING condition [, ...]]
[{ UNION [ALL] | INTERSECT | EXCEPT } select]
[ORDER BY column [ASC | DESC | USING operator] [, ...]]
[FOR UPDATE [OF class_name [, ...]]]
LIMIT { count | ALL } [{ OFFSET | , } start]

La cláusula FOR UPDATE permite a SELECT realizar un bloqueo exclusivo de los registros seleccionados.

COMMIT — Compromete la transacción actual
Sintaxis
COMMIT [WORK | TRANSACTION]

Entrada
WORK
TRANSACTION

Salida
COMMIT - Mensaje devuelto si la transacción se realiza con éxito.
NOTICE: COMMIT: “no transaction in progress” - Si no hay transacciones en progreso.

Descripción
COMMIT realiza la transacción actual. Todos los cambios realizados por la transacción son visibles a las otras transacciones, y se garantiza que se conservan si se produce una falla en la máquina.
Notas
Las palabras clave WORK y TRANSACTION son demasiado informativas, y pueden ser omitidas. Use *ROLLBACK* para abortar una transacción.

Uso
Para hacer todos los cambios permanentes:

COMMIT WORK;

ROLLBACK —Interrumpe la transacción en curso
Sintaxis
ROLLBACK [WORK | TRANSACTION]

Salida
ABORT - **Mensaje devuelto si la operación es exitosa.**
NOTICE: ROLLBACK: “no transaction in progress” - Si no hay transacciones en progreso actualmente.

Descripción
ROLLBACK deshace la transacción actual y provoca que todas las modificaciones originadas por la misma sean descartadas, es decir, restaura el estado anterior a la modificación.

Notas
Utilice COMMIT para terminar una transacción de forma exitosa. ABORT es un sinónimo de ROLLBACK.

Uso
Para cancelar todos los cambios:
ROLLBACK WORK;

Inicialización de parámetros usando el archivo de configuración.

Una manera de inicializar estos parámetros es editar el archivo *postgresql.conf*, el cual normalmente se encuentra en el directorio de datos (una copia se instala por omisión cuando el directorio de la base de

datos es inicializado). Un ejemplo de como se ve este archivo es el siguiente:

```
# This is a comment
log_connections = yes
log_destination = 'syslog'
search_path = '"$user", public'
shared_buffers = 128MB
```

Un parámetro es especificado por cada línea. El signo igual entre el nombre y el valor es opcional. El espacio en blanco es insignificante y las líneas en blanco son ignoradas. Un símbolo #, indica que el resto de la línea es un comentario. Los valores de parámetros que no son identificadores simples o nombres deben estar entre apostrofes.

El archivo de configuración es re-leído cuando el proceso servidor principal recibe una señal SIGHUB; esto se hace ejecutando el comando pg_ctl reload desde la línea de comando o invocando la función de SQL *pg_reload_conf()*. El servidor principal también propaga esta señal a todos los procesos servidores en ejecución por lo que las sesiones existentes también obtienen el nuevo valor. Alternativamente, se puede enviar la señal a un solo proceso servidor directamente. Algunos parámetros solo pueden ser configurados en el arranque del servidor; cualquier cambio a esas entradas en el archivo de configuración serán ignoradas hasta que este sea reiniciado. Las configuraciones inválidas en este archivo de son ignoradas durante el proceso de SIGHUB.

Autenticación de Clientes de PostgreSQL

La autenticación de los clientes se controla por medio de un archivo de configuración, al cual se le denomina *pg_hba.conf* y se almacena en el directorio donde se instala la base de datos (HBA quiere decir Host-based Authentication). El archivo *pg_hba.conf* se instala por omisión cuando el directorio de datos es inicializado por initdb.

El formato general del archivo *pg_hba.conf* es un conjunto de registros, uno por línea. Las líneas en blanco son ignoradas, como lo es cualquier texto después del carácter de comentarios #, Los registros no pueden continuar en múltiples líneas. Un registro se compone de un cierto número de campos los cuales se separan por espacios y/o tabuladores. Los campos pueden contener espacios en blanco si el valor del campo es entrecomillado.

Cada registro especifica un tipo de conexión, un rango de direcciones IP de clientes (si es relevante para el tipo de conexión), un nombre de base de datos, un nombre de usuario, y el método de autenticación a ser usado para las conexiones que coincidan con estos parámetros. El primer registro que coincida con: un tipo conexión, dirección de cliente, base de datos solicitada, y el nombre del usuario es usado para efectuar la autenticación. No hay una lectura adicional o respaldo: si un registro es elegido y la autenticación falla, los registros subsecuentes no se consideran. Si ningún registro coincide, se niega el acceso.

Un registro de este archivo puede tener uno de estos siete formatos:

```
local       database  user  auth-method  [auth-options]
host        database  user  address  auth-method  [auth-options]
hostssl     database  user  address  auth-method  [auth-options]
hostnossl   database  user  address  auth-method  [auth-options]
host        database  user  IP-address  IP-mask  auth-method  [auth-options]
hostssl     database  user  IP-address  IP-mask  auth-method  [auth-options]
hostnossl   database  user  IP-address  IP-mask  auth-method  [auth-options]
```

3. PREPARACIÓN DEL AMBIENTE DE TRABAJO

Configuración de la red. Instale la red de área local usando el mecanismo de su elección. Asigne a cada equipo una dirección IP estática.

Configuración de PostgreSQL para aceptar conexiones remotas. En cualquier sistema operativo que esté usando busque los archivos ***Postgresql.conf*** y ***Pg_hba.conf*** y efectúe en ellos los siguientes cambios.

Postgresql.conf
Busque el renglón con contiene el siguiente comando:
listen_address = ´localhost'

Y modifique con la siguiente configuración
listen_address = ´*'

Pg_hba.conf
Busque el renglón donde se encuentra la configuración de IPv4:
IPv4 local connections:

Elimine el renglón de abajo y ponga la siguiente configuración:
IPv4 local connections:
host all all 127.0.0.1/32 md5

Es importante reiniciar el equipo que ejecuta el proceso servidor de PostgreSQL, para que desde el arranque este proceso adquiera los valores recién configurados.

Nota: Para usuarios de sistemas operativos Windows y Linux
Windows. Los archivos de configuración se encuentran en la ruta "C:\Program Files\PostgreSQL\9.3\data".

Linux
Regularmente se encuentra en el directorio ***/usr/local/pgsql/data/***, pero seguramente cada sabor tiene su propia ruta de instalación.

Programas en Java
Para este laboratorio se requiere que se conecten al menos dos computadoras en red, con direcciones IP fijas, y que solo una de ellas esté ejecutando la base de datos. **Descargue el proyecto en Java que se proporciona** en la página de la revista e instale en cada una de las computadoras que forman parte de la red, abrir en cada caso con

el IDE Netbeans o Eclipse. La clase conexión (donde se configura el JDBC) debe ser modificada en cada equipo. El comando **localhost** deberá ser cambiado por la **dirección IP** del equipo con ejecuta la base de datos, tal y como se muestra en el programa de abajo.

Clase CONEXIÓN.java.

```
public class Conexion {
   Connection conex;
   Statement stmt = null;
   boolean b;

   public Conexion(){
      try {
         // Creando un objeto para el driver JDBC
         Class.forName("org.postgresql.Driver");
         // Efectuando la conexion:
         IPAddr del BD Server/NombreBD, Usuario, Password
         conex = DriverManager.getConnection("jdbc:postgresql:
         //localhost/uacme","postgres","clavesecreta");
         System.out.println("conexion establecida");
         stmt = conex.createStatement();
         b = true;
      } catch (Exception e ){
         System.out.println
         ("Error al conectarse " + e.getMessage() );
         b = false;
      }finally{
         System.out.println("Base de datos conectada");
      }
   }

   public Statement getConexion(){
      return stmt;
   }
}
```

Preparando los datos y procedimientos almacenados para el ejercicio

Debido a que las operaciones de alta de Cursos Especiales y la Expedición de Cheques son dos actividades separadas y que cuando se efectúa la primera seguramente no se ha efectuado la segunda, es necesario volver a crear la tabla de Cursos Especiales, eliminando la referencia a Cheques. Esta integridad deberá efectuarse manualmente. También se crea el procedimiento almacenado que va a controlar la expedición de cheques.

Usando la cuenta del **usuario postgres** y desde la **aplicación psql del PostgreSQL** ejecute los siguientes comandos:

```
--Conectando a la BD uacme
\c uacme

-- Eliminando la tabla CursosEspeciales
Drop table CursosEspeciales cascade;

-- Creando la tabla Cursos Especiales
create table CursosEspeciales(
idcurso int,
idprofe int,
fini varchar,
ffin varchar,
ncuenta int,
cns int,
foreign key(idprofe) references Profesores,
primary key (idcurso)
);

-- Creando los cursos que van a impartir los
docentes Julio y Samuel
```

```
insert into CursosEspeciales val-
ues(10,5,20150204,20150204, 0, 0);
insert into CursosEspeciales val-
ues(20,6,20150204,20150204, 0, 0);

-- Insertando datos en la especialización de
Cursos Especiales Extracurriculares
insert into CEExtracurricula values(10, 3);
insert into CEExtracurricula values(20, 4);

-- Eliminando la funcion, por si acaso lo
necesitan
DROP FUNCTION AltaDeCheque(int, int, numeric,
int);

-- Función que inserta el registro del cheque,
actualiza el saldo de la cuenta de cheques
-- y referencia el cheque contra la tabla
CursosEspeciales.
-- Solo se permite pagar un curso con un
cheque.
-- Parámetros: Número de Cuenta de Cheques,
Número de cheque, Importe del Cheque,
-- Curso que se está pagando.
```

```
CREATE OR REPLACE FUNCTION AltaDeCheque(int, int, numeric, int )
RETURNS int AS '
DECLARE
vcta alias for $1;
vnchq alias for $2;
vcant alias for $3;
vcur alias for $4;
vreg record;
vnuevo numeric(10,2);
vban int;

BEGIN
   vban = 0;
   -- Selecciona la cuenta y decrementa el saldo
   Select into vreg * from CuentaCheques where ncuenta = vcta;
   vnuevo := vreg.saldo – vcant;
   update CuentaCheques set saldo = vnuevo where ncuenta = vcta;

   -- Inserta el nuevo cheque
   Insert into Cheque values (vcta, vnchq, vcant, now() );

   -- Relaciona el cheque con el curso
   Update CursosEspeciales set ncuenta=vcta, cns = vnchq where
idcurso = vcur;
   vban = 1;
   return vban;
END;
' LANGUAGE 'plpgsql';
```

```
-- Consulta de saldos en las Cuentas de
Cheques, saldo actual de 9000 pesos
Select * from CuentaCheques;

-- Consulta a los Cursos Especiales
Select * from CursosEspeciales;

--Probando que la función funcione
adecuadamente
-- El curso 10 (diseño) lo imparte el profesor
Julio y el pago es de 3000 pesos
```

```
-- El curso 20 (java) lo imparte el profesor
Samuel y el pago es de 1000 pesos
Select AltaDeCheque( 2, 21, 3000, 10);
Select AltaDeCheque( 2, 22, 1000, 20);

-- Consultando el saldo de la cuenta de
cheques, verifique que todo está correcto
-- El nuevo saldo de la cuenta 2 debe ser 5000
pesos. ¿Funciona adecuadamente?
Select * from CuentaCheques;

-- Regresando los datos a los valores
originales
-- El curso 10 y 20 aún no están pagados
Update CursosEspeciales set ncuenta=0, cns = 0
where idcurso = 10;
Update CursosEspeciales set ncuenta=0, cns = 0
where idcurso = 20;

-- Eliminando todos los cheques de la Cuenta
de Cheques 2
Delete from Cheques where ncuenta = 2;

-- Devolviendo el saldo a 9000 pesos de la
Cuenta de Cheques 2
Update CuentaCheques set saldo = 9000 where
ncuenta = 2;
```

Una vez configurado PostgreSQL para recibir conexiones remotas y nuestro entorno en la base de datos, estamos listos para probar las transacciones y resolver el problema que nos plantea este laboratorio.

4. PROBLEMÁTICA A RESOLVER

Para nuestro ejercicio debemos hacer algunas suposiciones. Resulta que los dos empleados del DA están generando cheques para pagar cursos de la cuenta 2 (con un saldo de 9000 pesos), uno para el profesor Julio por 3000 pesos y otro para el profesor Samuel por 1000 pesos, lo curioso es que al momento de generarlos y debido a que el sistema está funcionando en red con una base de datos centralizada, lo hacen al mismo tiempo, como consecuencia el saldo quedó en 6000 pesos (o podría quedar en 8000 dependiendo de cuál cheque afecta el saldo primero). Así que efectuaremos primero las transacciones con un procedimiento almacenado pero sin usar transacciones y después incorporamos su uso para demostrar la utilidad de las mismas.

1er. Caso:

Efectuaremos la expedición de dos cheques de manera simultánea desde los programas escritos en Java, uno para el profesor Julio y el otro para el profesor Samuel. Ejecute el programa index.java en el IDE de su elección, en cada equipo. Estos programas se deben ejecutar en equipos distintos para obligar a la base de datos a tratar datos en forma concurrente. La figura 2 muestra los datos que se deben capturar en el programa index.java, **desmarque el CheckBox de Transacciones** de la pantalla. Se intenta forzar al PostgreSQL a cometer un error, por lo que después de capturar los datos se debe dar clic sobre el botón Enviar de manera simultánea en los dos equipos que forman parte de la red. La figura 3 muestra el detalle del programa Index.Java que invoca la función AltaDeCheque cuando no se selecciona el CheckBox indicado, debe notar que no se ha incorporado el uso de transacciones.

Figura 2. Datos a capturar en cada equipo de la red

Figura 3. Detalle de la invocación del procedimiento almacenado para el caso 1.

```
Statement t = c.getConexion();
try {
   int x = t.executeUpdate("Select AltaDeCheque(" + cuenta
+ ", " + cheque + ", " + total + ", " + curso + ")");
}
   catch (Exception e){
   System.out.println(e.getMessage() );
}
```

Finalmente consulte el saldo de la Cuenta de Cheques 2, ¿Cuál es el saldo? ¿Es correcto? De estar correcto el saldo, devuelva los valores a su estado previo y vuelva a intentarlo hasta que encuentre un resultado erróneo.

Explique la razón por la cual fallan los procesos o explique la razón por la cual falla si la invocamos desde Java.

2do. Caso:

Ejecute la sección **Regresando los datos a los valores originales** de la sección 3. Preparación del Ambiente de Trabajo. Ejecute los programas de Java en cada equipo y capture los mismos datos del caso 1 (Figura 2), solo que ahora **seleccione el CheckBox de Transacciones** de la pantalla. Estos programas se deben ejecutar en equipos distintos para obligar a la base de datos a tratar datos en forma concurrente. La figura 4 muestra el detalle del programa *Index.Java* que invoca la función AltaDeCheque cuando es seleccionado el CheckBox indicado, debe notar que se ha incorporado el uso de transacciones. Se intenta forzar al PostgreSQL a cometer un error, por lo que después de capturar los datos se debe dar clic sobre el botón Aceptar de manera simultánea en los dos equipos que forman parte de la red. Note que se ha agregado el comando "for update" al cuerpo de la llamada a la AltaDeCheque, y que previamente se ha invocado el comando "Begin Transaction" posteriormente se ha agregado el comando "Commit Transaction".

Figura 4. Detalle de la invocación del procedimiento almacenado para el caso 2.

```
Statement t = c.getConexion();
try {
      int x = t.executeUpdate("Begin Transaction");
}
catch (Exception e){
   System.out.println(e.getMessage() );
}
try {
      int x = t.executeUpdate("Select AltaDeCheque(" + cuenta +
", " + cheque + ", " + total + ", " + curso + ")");
}
catch (Exception e){
      System.out.println(e.getMessage() );
}
try {
      int x = t.executeUpdate("Commit Transaction");
}
catch (Exception e){
      System.out.println(e.getMessage() );
}
```

Finalmente consulte el saldo de la Cuenta de Cheques 2, ¿Cuál es el saldo? ¿Es correcto? De estar correcto el saldo vuelva a intentarlo. Explique la razón por la cual ahora no falla.

5. TRABAJO ADICIONAL

- Modifique la función *AltaDeCheque* para que en caso de que no se tenga saldo en la cuenta de cheques no se permita ejecutar la inserción del cheque y devuelva un valor de 0. Además modifique la invocación desde Java para que en caso de que la función *AltaDeCheque* devuelva un cero se ejecute un *Rollback Transaction* y en caso de un 1 se ejecute un *Commit Transaction*;
- El mismo problema aparece con el total recabado por cada curso al momento de que dos cajeros cobren el mismo curso al mismo tiempo a dos alumnos distintos. Construya las funciones pertinentes en PL y adecúe el programa en Java que desarrolló en el laboratorio 3. Haga los cambios que considere necesarios.
- Para el sistema de inventarios, actualice el inventario en un procedimiento de transacciones.

6. REFERENCIAS

Douglas Korry y Susan Douglas. (2005) PostgreSQL A comprehensive guide to building, programming and administering Postgre SQL databases. (2nd. Edition). Sams

Elmasri, R.; Navathe, S.B. (2002). *Fundamentos de Sistemas de Bases de Datos*. 3ª Edición. Addison-Wesley

Garcia-Molina, Jeffrey Ullman y Jennifer Widom. (2008). *Database systems: the complete book*. Prentice-Hall.

Momjiam Bruce. (2001). *PostgreSQL Introduction and Concepts*. Boston: Addison-Wesley Logman Publishing Co. Inc.

Silberschatz Abraham, Henry Korth y S. Sudarshan. (2006). *Fundamentos de Base de Datos* (5a. Ed.). España: McGraw-Hill

The PostgreSQL Global Development Group. (2015). *Manual de postgreSQL*. Disponible enwww.postgresql.org