Fifth LACCEI International Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2007)
"Developing Entrepreneurial Engineers for the Sustainable Growth of Latin America and the Caribbean: Education, Innovation, Technology and Practice"
29 May – 1 June 2007, Tampico, México.

Base de reglas activas en un sistema bancario, un enfoque de red de Petri

Joselito Medina Marín
Centro de Investigación en Ingeniería Industrial, Universidad Autónoma del Estado de Hidalgo, Pachuca, Hidalgo, México, jmedina@uaeh.edu.mx

Aurora Pérez Rojas
Centro de Investigación en Ingeniería Industrial, Universidad Autónoma del Estado de Hidalgo, Pachuca, Hidalgo, México, auropr@uaeh.edu.mx

Oscar Montaño Arango
Centro de Investigación en Ingeniería Industrial, Universidad Autónoma del Estado de Hidalgo, Pachuca, Hidalgo, México, oscarma11@yahoo.com.mx

RESUMEN

Las bases de datos activas (BDA) son extensiones de las bases de datos (BD), las cuales, además de tener un comportamiento pasivo (modificar ú obtener información solicitada por el usuario), reaccionan ante la presencia de uno o más eventos en la BD. El comportamiento activo de una BD puede modelarse con las reglas evento-condición-acción (reglas ECA). En este trabajo se presenta un modelo de red de Petri que puede ser utilizada para representar reglas ECA, así como un software que se ha desarrollado, denominado ECAPNSim, donde una base de reglas ECA, puede ser convertida en un red de Petri extendida, con la finalidad de analizar el comportamiento de las reglas, y en su caso, poder ser conectadas a un sistema de base de datos y, que este sistema, reaccione automáticamente ante la ocurrencia de eventos de interés para el administrador del sistema. Además, se muestra un caso de estudio sobre un sistema bancario, donde se utilizan reglas ECA para lograr un estado consistente de la base de datos.

Palabras claves: Reglas activas, bases de datos activas, redes de Petri, reglas ECA

ABSTRACT

Active Databases (ADB) are database (DB) extensions, which in addition to have a pasive behavior (update or get information required from the user), they react when one or more events occur inside of DB. It can be modeled with the event-condition-action (ECA) rules. In this work a Petri net model is presented and the software ECAPNSim, where a ECA rule base can be converted into an extende Petri Net, with the aim of to analyze the behavior of the rules, and consequently, the extended Petri net can be connected to the database system, which can reacts automatically when certain interesting events occur. Furthermore, a study case about a banking system is showed, where ECA rules are used in orderto achieve a consistent state of the database.

Keywords: Active rules, active database, Petri nets, ECA rules

1. INTRODUCCION

Un sistema de Base de Datos (SBD) se forma a partir de la unión de una base de datos (BD) ó repositorio de datos, y de un conjunto de programas que se encargarán de manipular a los datos almacenados en la BD (Silberschatz et al, 1999). En la figura 1 se muestra el esquema de un SBD.

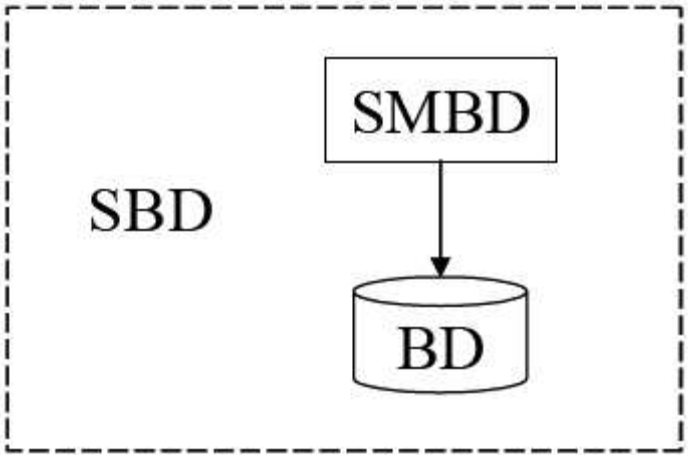

Figura 1. Esquema tradicional de un sistema de base de datos.

Los SBD's fueron diseñados para manejar grandes volúmenes de información. El manejo de datos involucra la definición de las estructuras donde serán almacenados los datos, así como de proveer mecanismos que manipulen eficaz y adecuadamente a éstos datos. Una BD no es mas que el conjunto datos que tienen relación entre sí y mantienen un significado implícito. Como se mencionó anteriormente, en la BD se refleja una parte del mundo real, solo aquella parte que es de nuestro interés, llamada minimundo o universo de discurso. Si el estado del mundo real es modificado, tales modificaciones también son realizadas dentro de la BD, para mantener una coherencia de información entre el mundo real y el minimundo.

Los mecanismos de manipulación de datos, en lo que se refiere a la creación de la BD, las modificaciones, agregaciones, eliminaciones y acceso a los datos, se realizan mediante un programa o un conjunto de programas conocido como Sistema Manejador de Bases de Datos (SMBD). Las bases de datos activas (BDA) son extensiones de las bases de datos (BD), las cuales, además de tener un comportamiento pasivo (modificar ú obtener información solicitada por el usuario), reaccionan ante la presencia de uno o más eventos en la BD (Paton et al, 1999). El comportamiento activo de una BD puede modelarse con las reglas evento-condición-acción (reglas ECA). La mayoría de las BDA comerciales utilizan el esquema de reglas ECA y cada una de ellas proporciona al usuario una sintaxis de definición de reglas. Sin embargo, el administrador de la BDA no puede llevar a cabo una simulación del comportamiento de la base de reglas ECA antes de su implementación en la BDA. Una base de reglas ECA es considerada como un sistema basado en eventos y es posible representarla con una red de Petri (PN) extendida, así como los eventos que las disparan.

Los Sistemas Manejadores de BD comerciales soportan la implementación de "triggers" en varios niveles. Sin embargo, generalmente presentan ciertas limitaciones. Entre estas limitaciones pueden mencionarse las siguientes: el evento del "trigger" solamente pueden construirse a partir de expresiones de SQL (como update, insert, delete o select) en una sola tabla de la BD; además, los "triggers" no pueden anidarse, es decir, que dentro de un "trigger" no puede invocarse a otro "trigger". Entre los Sistemas Manejadores de BD que soportan la definición del comportamiento activo se encuentra SYBASE (McGoveran et al, 1992), INFORMIX (Lacy-Thompson, 1990), ORACLE (Hursh et al, 1991), Microsoft SQL Server (González-Pérez, 1999), entre otros. Dentro de los sistemas no comerciales que proporcionan la definición de reglas activas podemos mencionar a Ariel, HiPAC, Startburst y POSTGRES(Paton et al, 1999).

2. MODELO CONCEPTUAL

La definición de la base de reglas ECA es la que provee de la funcionalidad reactiva al SBDA y está estrechamente ligada a la sintaxis del lenguaje de reglas que tenga el SBDA donde se deseen implementar. (Dittrich 1995)

Una regla ECA está formada por tres elementos: el evento, la condición y la acción. La forma general para representar a una regla ECA es la siguiente:

on evento

if condición

then acción

Debido a que una regla ECA es un sistema manejado por eventos, por medio de una PN es factible llevar a cabo la modelación y simulación de un sistema de reglas activas.

Una PN es un tipo particular de grafo dirigido bipartito, compuesto por dos tipos de objetos. Estos objetos son lugares y transiciones, los arcos que los unen están dirigidos de lugares a transiciones o de transiciones a lugares. Gráficamente, los lugares son representados por círculos y las transiciones son representadas por barras o por rectángulos.

En su forma más simple, una PN se representa por una transición unida con su lugar de entrada y su lugar de salida. Esta red básica se utiliza para representar varios aspectos de un sistema que se pretende modelar. Con la finalidad de estudiar el comportamiento dinámico de los sistemas modelados, desde el punto de vista del estado que presentan en un momento dado y los cambios de estado que pueden ocurrir, cada lugar puede no tener tokens o tener un número positivo de ellos. Los tokens se representan gráficamente por pequeños círculos rellenos. La presencia o ausencia de un token dentro de un lugar indica si una condición asociada con este lugar es falsa o verdadera, ó también nos indica si un dispositivo que se está modelando se encuentra disponible o no (Murata 1989). En la figura 2 se muestra un lugar de entrada, que está conectado mediante un arco desde el lugar hacia la transición; la transición en el centro; y un lugar de salida, que está conectado por un arco dirigido desde la transición hacia el lugar. En el lugar de entrada se tiene a un token, lo que significa que en este momento el sistema cuenta con la presencia del elemento modelado por el lugar de entrada.

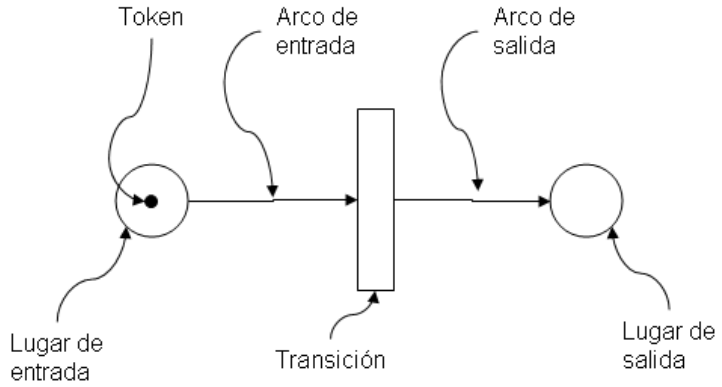

Figura 2. Elementos de una red de petri.

La ejecución de una PN es controlada por el número de tokens y su distribución en la red. Los tokens se alojan en los lugares y controlan el disparo de las transiciones que forman la red. Cambiando la distribución de los tokens en los lugares, podremos estudiar el comportamiento dinámico que puede alcanzar el sistema en diferentes estados. Una PN es ejecutada a través del disparo de transiciones, que se lleva a cabo con la aplicación de la regla de habilitación (enabling rule) y posteriormente se aplica la regla de disparo (firing rule), las cuales gobiernan el flujo de los tokens por la red.

1. Regla de habilitación de transiciones: Una transición t se dice que será habilitada si cada lugar de entrada p de t contiene al menos un número de tokens igual al peso k del arco dirigido que conecta a p con t.

2. Regla de disparo de transiciones:

a. Una transición habilitada t puede dispararse o no dependiendo de la interpretación adicional que tenga la transición, y

b. El disparo de una transición habilitada t elimina de cada lugar de entrada p el mismo número de tokens que el valor del peso k del arco dirigido que conecta a p con t. Además, agrega a cada lugar de salida p el número de tokens que sea igual al peso k del arco dirigido que conecta a la transición t con el lugar p.

Sin embargo, los atributos y características que tienen las reglas ECA no pueden ser representados fácilmente con el modelo de PN original definido en (Murata, 1989). Para lograr una representación adecuada de las reglas ECA se desarrolló un modelo de PN extendido, al que hemos denominado Red de Petri Coloreada Condicional (CCPN, Condicional Colored Petri Net) (Medina-Marín, 2005).

La Red de Petri Coloreada Condicional (Conditional Colored Petri Net, CCPN) es una extensión de PN, la cual hereda atributos y la regla de disparo de transiciones de PN. Además, en la CCPN se toman conceptos que están presentes en la definición de la red de Petri coloreada (CPN), tales como la definición de tipos de datos, asignación de colores (valores) a los tokens, y la asignación de tipos de datos a los lugares. En el caso de CPN's la asignación de tipos de dato se hace hacia todos los lugares de la CPN, en el caso de la CCPN, la asignación de tipos de datos a lugares no es general, ya que en la CCPN se manejan lugares (lugares virtuales) con la capacidad de alojar tokens de diferentes tipos de datos.

En una regla ECA se evalúa la condición de la regla. En la CCPN se utiliza una función que realiza la evaluación de la parte condicional de la regla ECA almacenada en una transición. Para el caso de los eventos compuestos donde se tiene que verificar un intervalo de tiempo, la CCPN ofrece una función para asignar los intervalos de tiempo a una transición, la cual verificará si determinados eventos ocurren dentro del intervalo, de manera similar a como realiza la evaluación de la condición de una regla. A este tipo de transiciones la denominamos transición compuesta.

Como se definió previamente, cada uno de los eventos ocurre en un punto del tiempo, por lo tanto, la CCPN proporciona un función que asigna a cada token, que representa la ocurrencia de un evento, una estampa de tiempo, el cual especifica el momento en que éste ocurrió, para efectos de evaluación de intervalos y de eventos compuestos.

Finalmente, cada vez que ocurre un evento, la CCPN contiene una función para inicializar los tokens, es decir, generar la estructura del token y la asignación de los valores correspondientes al evento detectado en la BD.

Cualquier base de reglas ECA puede ser representada por medio de la CCPN. El evento activador de la regla ECA se convierte en una estructura de CCPN capaz de realizar la detección y conformación del evento. Si se trata de un evento de tipo primitivo, éste es representado por un lugar de la CCPN. Sin embargo, si el evento de la regla es un evento compuesto, se genera la estructura de CCPN apropiada para definir al evento compuesto. Ambos tipos de eventos, primitivos y compuestos proporcionan un lugar, el cual será utilizado como lugar de entrada para una transición.

El siguiente elemento de la regla ECA, la condición, es almacenada dentro de una transición de la CCPN, la cual, además de evaluar la presencia de tokens en su lugar de entrada (la ocurrencia del evento) evalúa la condición de la regla que tiene almacenada. Incrementándole a la regla de disparo de transiciones de PN's, la evaluación de una expresión booleana.

Finalmente, el elemento de la regla acción, debido a que al ser ejecutado modifica el estado de la BD, se representa como un lugar de salida de la transición que tiene almacenada a la condición correspondiente.

3. IMPLEMENTACIÓN

Se desarrolló una interfaz utilizando el lenguaje de Programación Orientado a Objetos Java, para llevar a cabo la implementación de la CCPN, en el cual se realiza una conversión automática de una base de reglas ECA en una CCPN, al que hemos denominado ECAPNSim (ECA & PN Simulator). Esta interfaz funciona como una capa superior colocada sobre la BD, la cual escucha los eventos que genera el usuario al escribir instrucciones de SQL en una consola. Estos eventos son analizados y se determina si existe una regla en la CCPN que se va a disparar, y en consecuencia se modifica el estado de la BD. En la figura 3 se muestra la arquitectura del funcionamiento de ECAPNSim.

ECAPNSim ofrece un medio gráfico y visual para representar bases de reglas ECA, auxiliándose de la CCPN. Como cualquier editor de PN, es posible realizar una simulación del comportamiento del sistema, en este caso, la simulación de la base de reglas ECA. Durante la ejecución de la simulación pueden observarse problemas de

BDA, como la no-terminación y la confluencia, y por lo tanto, el desarrollador de la base de reglas puede modificar la base de reglas y pensar en otra alternativa de solución para evitar éstos problemas que conducen a estados inconsistentes del Sistema de BD.

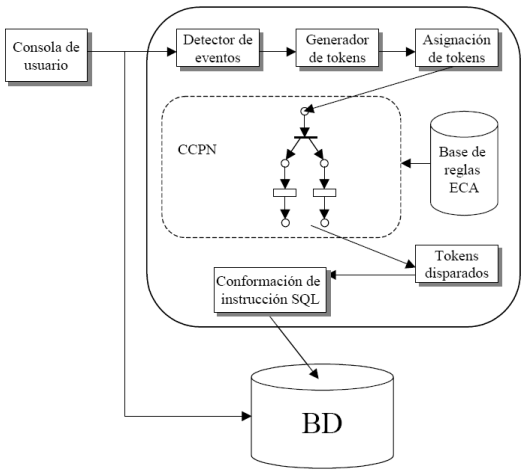

Figura 3. Arquitectura de funcionamiento de la interfaz ECAPNSim.

A diferencia de otros sistemas que soportan la definición de reglas ECA, ECAPNSim ofrece dos modalidades de uso. En la modalidad "Simulación", el usuario ejecuta la simulación del comportamiento de la base de reglas. En la modalidad "Real", la base de reglas analizadas previamente, dará comportamiento activo a una BD pasiva, utilizando la misma CCPN con que se ejecutó la simulación. En la figura 4 se muestra el ambiente con que trabaja ECAPNSim.

4. CASO DE ESTUDIO

Se desarrolló un caso de estudio para las reglas que existen en una institución bancaria, de acuerdo a los procesos, restricciones y movimientos que automáticamente pueden ser ejecutados, de acuerdo a ciertos parámetros que existan en el contenido de la BD. Se utilizó el modelo relacional para describir el universo de discurso de la BD para este caso de estudio. Las relaciones y sus atributos se describen a continuación:

CLIENTE (número, nombre, dirección, y la fecha de inicio).

CUENTA (número, número de cliente, saldo, límite, tipo {crédito, ahorro, nómina, cheques},estado (activada, bloqueada, cancelada)).

TRANSACCIONES PENDIENTES (clave, tipo {depósito, venta}, cuenta origen, cuenta destino, cantidad, periodo).

DEPOSITO (fecha de depósito, número de cuenta, cantidad, lugar {sucursal, cajero automático, internet, teléfono}, tipo de moneda).

RETIRO (fecha de retiro, número de cuenta, cantidad, lugar {sucursal, cajero automático, internet, teléfono}, ubicación).

PAGO POR SERVICIOS (fecha, cuenta origen, cuenta destino, modo de acceso, cantidad, concepto del pago, ubicación).

IMPRESION DE COMPROBANTE (datos de la transacción).

REPORTE (tipo de reporte, descripción, cliente, lugar, estado).

SOLICITUD DE TARJETA DE CREDITO (número de tarjeta de crédito, cuenta, con fotografía, tarjeta de crédito adicional).

Para controlar algunas de las tareas realizadas por el banco, se desarrolló un conjunto de reglas ECA. Estas reglas pueden ser ejecutadas automáticamente en la BD, proporcionándole un comportamiento activo a la BD. Se tiene un total de diecisiete reglas, las cuales se describen a continuación:

Regla 0: Cuando un depósito, un retiro o un pago por servicio es realizado, se imprime un comprobante para el cliente.

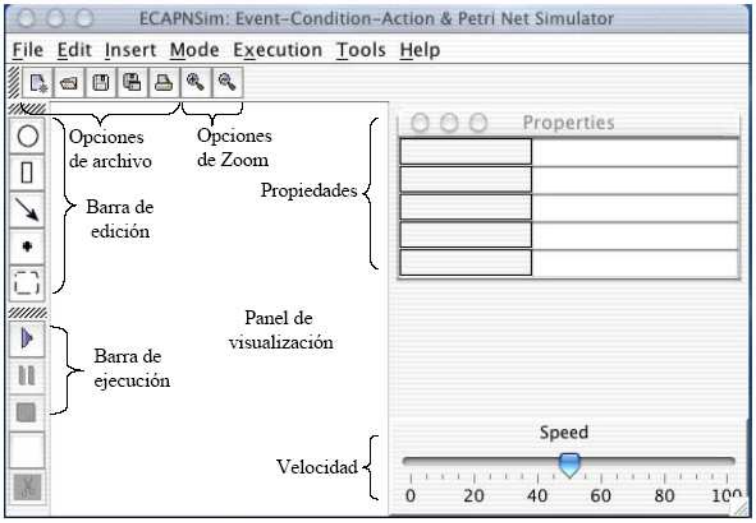

Figura 4. Ambiente de ejecución de ECAPNSim.

Para cada una de estas reglas, es necesario convertirlas a una forma de regla ECA, de acuerdo al modelo general para reglas ECA, tal y como se muestra en la tabla 1.:

Tabla 1: Descripción de reglas

Regla	Evento	Condición	Acción
000	or(insert_Deposito: insert_Retiro: insert_Pago PorServicio)	True	Then insert into ExpideComprobante values (’expide comprobante’) ;
001	insert_Retiro	Cuenta.saldo > insert.cantidad	Then update Cuenta set value saldo = saldo - Retiro.cantidad where numero = insert.cuenta;
002	insert_ PagoPorServicio	Cuenta.saldo > insert.cantidad	Then insert into Retiro values (today, insert.cuentaOrigen, insert.cantidad, ’Sucursal’, insert.ubicacion);
003	insert_Deposito	insert.fecha < ’16:00:00’	Then update Cuenta set value saldo = old.saldo + insert.cantidad where numero = insert.cuenta;
004	insert_Deposito	Deposito.fecha >= ’16:00:00’	Then insert into Pendientes values (, ’deposito’, 9999, insert.cuenta, insert.cantidad, 2) ;
005	insert_ PagoPorServicio	insert.modoAcceso = ’Internet’	Then insert into Retiro values (today, insert.cuentaOrigen, 25.00, ’Internet’, insert.ubicacion) ;
006	insert_ PagoPorServicio	insert.modoAcceso = ’Telefono’	Then insert into Retiro values (today, Cuenta.numero, 20.00, ’Telefono’, insert.ubicacion) ;
007	insert_ PagoPorServicio	True	Then insert into Retiro values (today, insert.cuentaOrigen, insert.amount, ’Sucursal’, insert.ubicacion) ;
008	insert_Retiro	insert.lugar = ’Cajero	Then insert into PagoPorServicio values (Today ,

		automatico' & Cuenta.limite > 10	insert.cuenta, Null, Null, 5.00, 'Servicios_Varios' , 'Nacional');
009	insert_Deposito	insert.moneda <> 'Peso' & insert.fecha < '16:00:00'	Then update Cuenta set cantidad = old.cantidad - insert.cantidad + insert.cantidad * tipomoneda where numero = insert.cuenta;
010	insert_Deposito	Deposito.moneda <> 'Peso' & Deposito.fecha >= '16:00:00'	Then insert into Pendientes values (1, 'deposito', null, insert.cuenta, insert.cantidad, 2));
011	update_Reporte_estado	Reporte.tipo = 'Reclamacion' & update.estado = 'No procede'	then insert into PagoPorServicio values (today, Null, Cuenta.numero, REPORTE.lugar, 200, 'Servicios Varios', 'Nacional');
012	insert_Retiro	insert.lugar = 'Cajero automatico externo' & Retiro.ubicacion = 'Nacional'	Then insert into PAGOPORSERVICIO values (today, NULL, insert.cuenta, 'Automatico', 10, 'Servicios varios', 'Nacional');
013	insert_Retiro	insert.lugar = 'Cajero automatico externo' & insert.ubicacion = 'Extranjero'	Then insert into PAGOPORSERVICIO values (today, NULL, insert.cuenta, 'Automatico', 80, 'Servicios varios', 'Extranjero');
014	insert_ExpedicionTC	insert.conFotografia = 1	Then insert into PAGOPORSERVICIO values (today, NULL, insert.cuenta, 'Automatico', 200, 'Servicios varios', 'Nacional');
015	insert_ ExpedicionTC	insert.adicional = 1	Then insert into PAGOPORSERVICIO values (today, NULL, insert.cuenta, 'Automatico', 100, 'Servicios varios', 'Nacional');
016	insert_Cuenta	insert.tipo = 'Credito'	Then insert into PAGOPORSERVICIO values (today, insert.numero, NULL, 'Automatico' , 350.00, 'Servicios varios' , 'Sucursal') ;

La CCPN solo para la regla 001 se muestra en la figura 5. El lugar p_0 representa al evento de la regla ECA, es decir, la inserción de elementos en la relación Retiro está siendo monitoreado por p_0. La transición t_0 representa a la regla 001, y en ella se almacena la parte condicional de la regla ECA. Cuando el evento representado por p_0 ocurre, un token con información sobre el evento es generado y colocado en el lugar p_0. La transición t_0 se habilita, recibe el token de p_0 y analiza la información contenida en él, si la condición de t_0 se satisface de acuerdo a la información del token entonces se genera un nuevo token con información sobre la acción de la regla, y se envía hacia el lugar de salida p_1, el cual representa a la acción de la regla ECA.

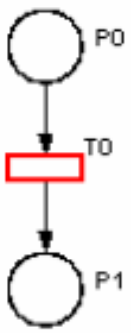

Figura 5.- CCPN para la regla 001.

La estructura de CCPN para la regla 000 se muestra en la figura 6. Los lugares p_0, p_1 y p_2 representan a los eventos primitivos agregar una tupla en la relación Deposito, agregar una tupla a la relación Retiro y agregar una tupla a la relación PagoPorServicio, respectivamente. Las transiciones t_0, t_1 y t_2 son transiciones de tipo copy, utilizadas para formar la estructura en CCPN para el evento compuesto disyunción. El lugar p_3 es un lugar virtual que representa al evento compuesto disyunción. La transición t_3 y el lugar p_4 representan a la regla y a la acción de la regla, respectivamente.

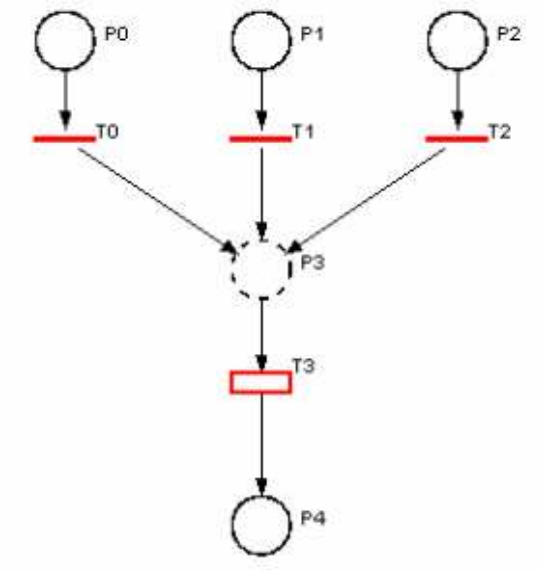

Figura 6.- CCPN para la regla 000.

La CCPN que representa a todo el conjunto de reglas ECA para el banco se muestra en la figura 7.

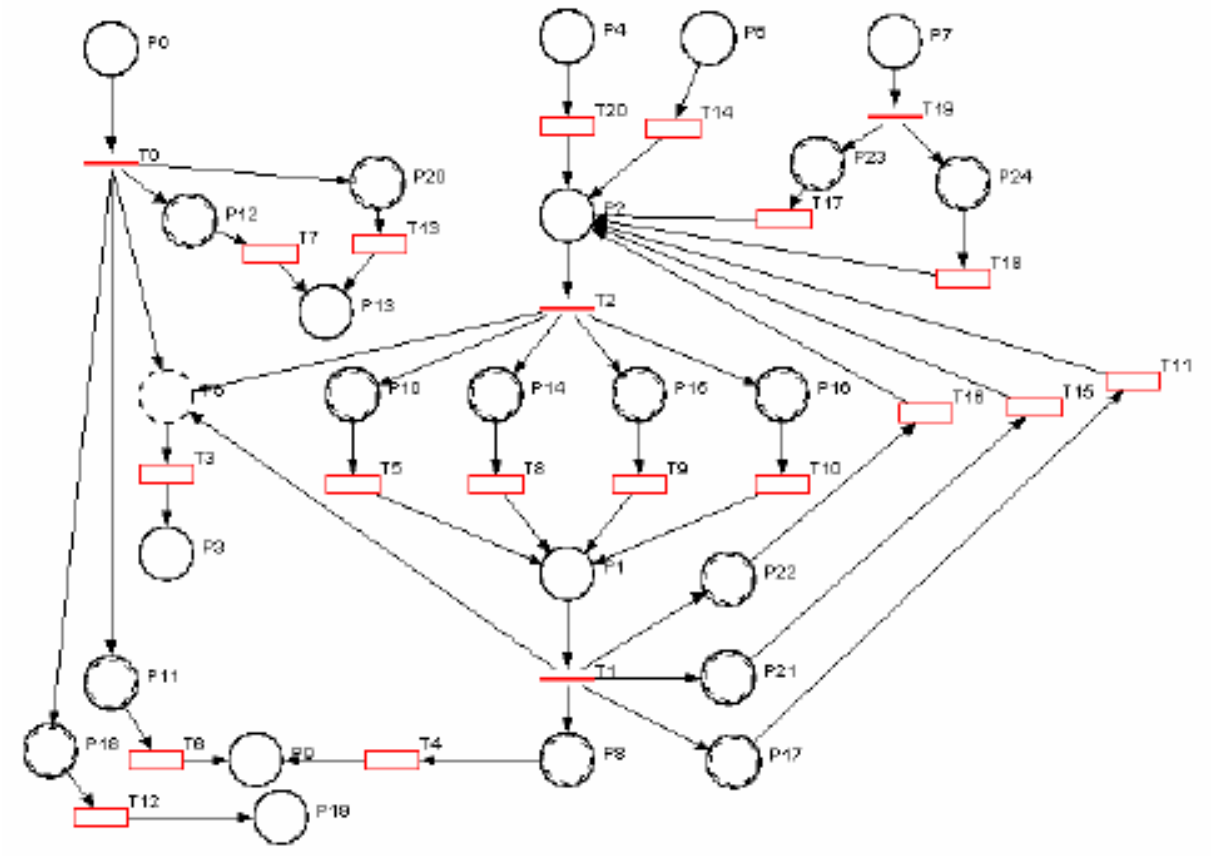

Figura 7.- CCPN para todo el conjunto de reglas ECA.

Tabla 2: Conjunto de acciones generadas.

Acción	Lugar
insert en reporte	P3
update cuenta.saldo	p9
insert en pendientes	p13
update cuenta.cantidad	p19

Tabla 3: Conjunto de reglas descritas como CCPN.

Regla	Evento (lugar entrada)	Transición	Acción (lugar salida)
000	p5	t3	p3
001	p8	t4	p9
002	p10	t5	p1
003	p11	t6	p9
004	p12	t7	p13
005	p14	t8	p1
006	p15	t9	p1
007	p16	t10	p1
008	p17	t11	p2
009	p18	t12	p19
010	p20	t13	p13
011	p6	t14	p2
012	p21	t15	p2
013	p22	t16	p2
014	p23	t17	p2
015	p24	t18	p2
016	p4	t20	p2

La regla 000 tiene como su evento activador al evento compuesto disyunción, el cual está formado por los eventos primitivos insert_depósito, insert_retiro, insert_PagoPorServicio, representados por los lugares p_0, p_1 y p_2, respectivamente.

Los lugares p_0, p_1 y p_2 son utilizados más de una vez, por lo que a estos lugares se les crea una copia para duplicar los tokens alusivos a éstos eventos.

5. CONCLUSIONES

Se desarrolló un caso de estudio para las reglas que existen en una institución bancaria, de acuerdo a los procesos, restricciones y movimientos que automáticamente pueden ser ejecutados, de acuerdo a ciertos parámetros que

Actualmente existen Sistemas Manejadores de Bases de Datos que soportan la definición de reglas ECA mediante "triggers"; sin embargo, los "triggers" presentan muchas restricciones que limitan la potencia que un sistema de BDA debe ofrecer.

Por otro lado, se tienen prototipos de investigación que también soportan la definición de reglas ECA, y ofrecen mayor potencia que el manejo de "triggers". Además, soportan la definición de eventos compuestos como la disyunción, conjunción, secuencia, entre otros. Sin embargo, a semejanza de los anteriores sistemas, la definición de las reglas ECA se lleva a cabo en base a una sintaxis propia del programa y solo trabajan sobre un Sistema Manejador de Base deDatos.

En este trabajo se puede mostrar que las redes de Petri pueden ser utilizadas para definir reglas ECA, las cuales modifican automáticamente el estado de una base de datos ante la ocurrencia de eventos que sean de interés a las

políticas de un sistema. Además de que la estructura de la red de Petri nos permite realizar análisis sobre la base de reglas aprovechando las herramientas que la teoría de redes de Petri nos ofrece.

Con el ejemplo desarrollado, se muestra la forma en que un conjunto de reglas de reglas ECA pueden ser convertidas en una red de Petri, en una red de Petri Coloreada Condicional, la cual tiene la capacidad de poder almacenar cada uno de los elementos de la regla: el evento (como un lugar de entrada), la condición (almacenada en una transición), y la acción o acciones de la regla ECA (denotadas como lugares de salida de las transiciones que almacenan la parte condicional.

En este ejemplo ejemplo se muestra la aplicación de una base de reglas ECA en un sistema bancario, en la cual e definen reglas que necesitan realizar ciertas acciones de manera automática sin necesidad de que el administrador o administradores de la base de datos intervengan directamente.

REFERENCIAS

Silberschatz A., Korth H.F., Sudarshan S.,(1999) ”Database System Concepts”, Third Edition, McGraw-Hill.

Paton N.W.; Diaz O.; (1999) “Active Database Systems”, ACM Computing Surveys, Vol. 31, No. 1, pp. 64-103.

McGoveran D.; Date. C.J.; (1992) “A guide to SYBASE and SQL Server : a user’s guide to the SYBASE product”, Sybase, Inc.

Lacy-Thompson T.; (1990) “INFORMIX-SQL, A tutorial and reference”, ISBN-0-13-465121-9, Ed. Prentice Hall.

Hursh C.J., Hursch J.L.; (1991) “Oracle SQL Developer’s Guide”, ISBN-0-8306-2529-1, Ed. McGraw- Hill.

González-Pérez A.; (1999) “SQL Server, Programación y administración”, ISBN 970-15-0376-7, Ed. Alfaomega ra-ma.

Dittrich K., Gatziu S., Geppert A., (1995) ”The Active Database Management System Manifesto: A Rulebase of ADBMS Features”. A Join Report by the ACT-NET Consortium. Proceedings of the 2nd International Workshop on rules in database systems”, pages 3—20.

Murata T.; (1989) ”Petri Nets: Properties, analysis, and applications”, Proceedings of the IEEE, 77(4):541-580.

Medina-Marín, Joselito; (2005) Tesis de Doctorado “Desarrollo de reglas ECA en Base de Datos Activas, Un enfoque de Red de Petri”, CINVESTAV-IPN, México.