14 del 2 al 5 de Diciembre de 2008
CONVENCIÓN CIENTÍFICA
DE INGENIERÍA Y ARQUITECTURA

44 ANIVERSARIO cujae 2008

Bases de datos activas, un enfoque de red de Petri

Joselito Medina-Marín, Xiaoou Li, Aurora Pérez-Rojas, Oscar Montaño-Arango, y José Ramón Corona-Armenta

Resumen. Las bases de datos activas (BDA) son extensiones de las bases de datos (BD), las cuales, además de tener un comportamiento pasivo, 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 artículo se está proponiendo un modelo general para el desarrollo de reglas ECA, basado en teoría de red de Petri (RdP), el cual puede utilizarse como un motor independiente sobre cualquier sistema de base de datos (SBD). Un sistema de base de datos activa (SBDA) debe ofrecer un modelo de conocimiento y un modelo de ejecución. En el modelo de conocimiento se especifican los elementos que conforman a la regla ECA. En la conversión de una base de reglas ECA, se toman los elementos de las reglas y se convierten en elementos del modelo extendido de RdP que se propone.

Palabras Claves. Base de datos activa, red de Petri, regla ECA.

I. Introducción

En un SBD tradicional se realizan operaciones sobre los datos que se encuentran almacenados en él, a través de comandos o instrucciones definidas por el administrador de la BD o los usuarios de la misma, es decir, si un usuario de la BD desea agregar, modificar, eliminar o simplemente hacer una consulta a la BD, entonces debe ejecutar los comandos necesarios para llevar a cabo estas operaciones.

Sin embargo, existen sistemas en el mundo real, donde de acuerdo a lo que esté ocurriendo en el universo de discurso, es necesario realizar una acción o procedimiento dependiendo de los cambios que hayan ocurrido, si de antemano se conocen las acciones que se deben ejecutar y el momento en que éstas deben ejecutarse, entonces es factible definir una serie de procedimientos para incorporarle a la base de datos este comportamiento.

Los esquemas tradicionales de sistemas de bases de datos no tienen la capacidad de soportar a los sistemas descritos en el párrafo anterior, en los cuales es necesario especificarle a la BD un comportamiento de reacción automática ante los cambios que se susciten dentro del universo de discurso de la BD, ya que de hacerlo un usuario humano es posible que incurra en errores en la detección de eventos.

Por esta razón surgieron los sistemas activos de bases de datos, mejor conocidos como Sistemas de Bases de Datos Activas (SBDA), los cuales deben ser capaces de detectar los eventos que ocurren en el universo de discurso, verificar el estado de la base de datos durante la ocurrencia de eventos, y además, ejecutar acciones o procedimientos dentro de la BD sin la intervención directa de un usuario humano.

Para proporcionar el comportamiento activo a una BD es necesario definir reglas que describan la reacción automática que debe seguir el manejador de la BD, también conocidas como reglas activas. La forma más general para lograr la definición de las reglas activas es mediante la aplicación del modelo de regla Evento-Condición-Acción (ó regla ECA), en la cual se define el evento que debemos detectar, la condición que debemos evaluar contra el estado que guarde en esos momentos la BD, así como la acción o grupo de acciones que se ejecutarán si la evaluación de la condición es verdadera. La mayoría de los sistemas comerciales (SYBASE, ORACLE, SQL Server, etc. [1] – [6]) ofrecen funciones para darle comportamiento activo a una base de datos, por medio de la definición de "triggers" o "disparadores".

El área de aplicación de un SBDA es muy amplia, desde sistemas de bases de datos administrativos, hasta sistemas de bases de datos de control en aerolíneas y de control de hospitales, así como monitoreo de transacciones financieras, identificación de actividades inusuales en el sistema, cumplimiento de restricciones de integridad, mantenimiento de datos derivados, generación oportuna de reportes, realización de procesos periódicos, entre otros [2].

Además, se han utilizado SBDA para la detección de intrusos en redes, utilizando el sistema VenusIDS [7].

Actualmente existen sistemas que pretenden incorporar un comportamiento activo a una BD, tanto comerciales como proyectos de investigación, sin embargo, cada uno de ellos ofrecen su propia sintaxis y plataforma de trabajo, convirtiéndolos en sistemas dependientes de la BD, además de que no ofrecen en su totalidad el análisis de reglas. A continuación se describen algunos de los más difundidos SBDA.

A. *Estado del arte*

En esta sección se mencionan sistemas de BD que ofrecen un comportamiento activo, la cual está dividida en dos

secciones, considerando por un lado a los sistemas de BD comerciales que proveen la definición de un sistema de reglas activas. Posteriormente se describen los sistemas de BD que no son sistemas comerciales, pero que también ofrecen la definición de reglas activas.

1) Sistemas Manejadores de BD comerciales

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".

Por ejemplo, en SYBASE, un sistema manejador de bases de datos relacionales, la parte condicional de un "trigger" solo puede referirse a una sola tabla, específicamente a la tabla donde se define el evento a detectar; un "trigger" no puede crearse sobre una vista (una vista es una tabla lógica, que muestra una parte de la base de datos; las vistas permiten "almacenar" de manera lógica los resultados de los queries o consultas a la BD); solamente un "trigger" puede asociarse con una operación en una tabla; la acción de un "trigger" está limitada a una secuencia de expresiones escritas en SQL; además, el disparo de "triggers" está limitado a un solo nivel, donde las acciones disparadas no provocan la activación de otros "triggers" [3].

INFORMIX es un sistema manejador de base de datos basado en el modelo relacional el cual consiste en un conjunto de módulos que forman el sistema completo. De manera similar a SYBASE, en este sistema manejador de base de datos, un usuario puede crear un "trigger" en una tabla de la BD actual; un usuario no puede crear un "trigger" en una tabla temporal, una vista o en una tabla de catálogo del sistema. Además, en INFORMIX, si un usuario define en cada uno de los eventos de los "triggers" más de un elemento, los elementos de los eventos entre los diferentes "triggers" deben de ser mutualmente exclusivos, es decir, un mismo evento no puede disparar a dos o más "triggers" [4].

ORACLE es un sistema manejador de bases de datos relacional con características que permiten la implementación del paradigma orientado a objetos, en donde los usuarios primeramente deben definir que tipo de evento activará a los "triggers", además de escribir "triggers" con una especificación explícita para esos eventos. Los "triggers" se ejecutan implícitamente cuando un comando INSERT, UPDATE, DELETE o SELECT es utilizado sobre una tabla asociada, o en algunos casos sobre una vista, o cuando ocurren determinadas acciones en el sistema de BD. Un "trigger" en ORACLE es un bloque instrucciones en PL/SQL o Java y almacenado en la BD, o pueden ser escritos en subrutinas de lenguaje C, que se dispara cuando una determinada instrucción en SQL se va a ejecutar sobre dicha tabla. PL/SQL es un lenguaje de programación procedural utilizado en los sistemas manejadores de bases de datos ORACLE [5].

Microsoft SQL Server, uno de los productos que componen a la plataforma BackOffice de Microsoft, es un servidor para BD's relacionales que utiliza como lenguaje de manipulación de datos una extensión del lenguaje SQL (Structured Query Language ó Lenguaje Estructurado de Consulta) llamado Transact SQL. Microsoft SQL Server también utiliza "triggers" para definir un comportamiento activo en su BD. Los eventos que se consideran para el disparo de los "triggers" en este manejador de BD son las operaciones de inserción, borrado o actualización de datos. Los "triggers" basados en instrucciones creadas enteramente en Transact SQL tienen una serie de limitaciones que vienen impuestas por el diseño de SQL Server: para evitar el disparo infinito o un ciclo infinito de disparo de "triggers", un "trigger" no puede nunca disparar a otro "trigger" en su interior, además por estas mismas razones, un "trigger" no puede ejecutar una instrucción DDL (definición, modificación o eliminación de estructuras de tablas, índices, BD's, etc.) [6].

2) Sistemas Manejadores de BD no comerciales

Dentro de los sistemas no comerciales que proporcionan la definición de reglas activas podemos mencionar a Ariel, HiPAC, Startburst y POSTGRES.

El sistema Ariel es la implementación de una base de datos relacional con un sistema de reglas tipo "built-in". El enfoque tomado en el diseño de Ariel adopta el modelo de sistemas de producción, tal como el OPS5 (Official Production System 5). OPS5 es un lenguaje para ingeniería cognoscitiva que soporta el método de representación del conocimiento en forma de reglas, el cual incorpora un módulo unificador, un intérprete que incluye un mecanismo de encadenamiento progresivo, y herramientas para edición y depuración de los programas [8].

HiPAC es una BDA orientada a objetos. El proyecto HiPAC es el pionero de muchas de las más importantes ideas en BDA's, tales como los modos de acoplamiento de reglas y la composición de eventos, aunque el diseño final de HiPAC no fue implementado en su totalidad. HiPAC utiliza el modelo relacional como marco de trabajo y el modelo de transacciones anidadas como el marco para la ejecución de reglas [13]. Cada regla en HiPAC está estructurada de acuerdo al paradigma evento-condición-acción. Los eventos pueden ser operaciones de la BD genéricos. La acción de la regla ECA, dentro de HiPAC, pueden contener operaciones de la BD, operaciones de transacción, operaciones de reglas, señales para identificar que un evento definido por el usuario a ocurrido o llamadas a los procedimientos de la aplicación. HiPAC estuvo asociado con la Base de Datos Orientada a Objetos (BDOO) pasiva PROBE, donde los objetos de este modelo fueron usados para almacenar a las reglas ECA de la parte extendida correspondiente a la BDA. HiPAC tiene la capacidad de ejecutar de manera simultánea ó paralela el disparo de reglas como subtransacciones [2].

Starburst es un sistema de BD relacional desarrollado en el Centro de Investigaciones de Almaden - IBM, el cual fue complementado con el sistema de reglas Starburst. El lenguaje de reglas de Starburst difiere de la mayoría de los lenguajes de reglas de BDA en que está basado en transiciones de estado de BD arbitrarias en lugar de tuplas. La característica del sistema de reglas de Starburst que más llama la atención es su modelo de ejecución, en el cual las reglas se disparan por el efecto final de un conjunto de cambios o modificaciones realizadas

sobre los datos almacenados en la BD [5]. Sin embargo, la semántica en la ejecución de las reglas es demasiado compleja, probablemente esto se deba a que al proveer más facilidades conduzca al desarrollo de conjuntos de reglas difíciles de entender y mantener [9].

El sistema manejador de BD POSTGRES utiliza un subsistema de reglas para ofrecer una reacción automática a la ocurrencia de eventos en el espacio de estados de la BD [10]. POSTGRES, a pesar de ser un sistema manejador de BD relacional, ofrece facilidades para el manejo de objetos y de operaciones definidas por el usuario. El sistema de reglas POSTGRES está orientado a tuplas, además, la ejecución de las reglas definidas en él se efectúan inmediatamente después de que alguna modificación a una tupla dispara y satisface la condición de una o más reglas [1].

B. Motivación

Debido a la necesidad de utilizar sistemas con capacidad de reacción a la ocurrencia de eventos, también conocidos como sistemas reactivos, en el área de BD se hizo necesaria la implementación de las BDA, sin embargo, cada una de las propuestas y sistemas manejadores de BD activas que se han analizado tienen una definición de reglas ECA exclusiva de cada sistema. En el caso de los sistemas de BDA comerciales podemos mencionar que solo manejan la definición de eventos que modifican, agregan, eliminan o seleccionan datos (por medio de los comandos update, insert, delete y select, respectivamente) y de los eventos compuestos (eventos que ocurren a partir de la ocurrencia de dos o más eventos utilizando un álgebra de eventos: conjunción, disyunción, negación, etc.) solo soporta la definición de la disyunción de eventos, además de que no debe existir una relación directa entre la acción de una regla y el disparo de otra [3] - [5]. Los sistemas de BDA que son aún prototipos de investigación ofrecen una mayor cantidad de propiedades de BDA que los sistemas comerciales, sin embargo, al igual que los anteriores, la sintaxis que utilizan en la definición de reglas es exclusiva para tales sistemas. Por otro lado, en el diseño y desarrollo de una base de reglas activas pueden presentarse problemas como el de No-terminación (disparo infinito de reglas al caer en un ciclo de disparo de reglas) y el de confluencia (cuando en el disparo simultáneo de dos reglas, el estado final de la BD depende del orden en que éstas se disparen).

Cada sistema de BDA ofrece su propia sintaxis de especificación de reglas ECA existiendo una incompatibilidad entre los diferentes manejadores de SBDA.

La fortaleza de un modelo depende de la capacidad que ofrezca para ser utilizado, sin ningún problema, en el área para la cual fue diseñado y en diferentes situaciones de aplicación donde sea requerido.

En el área de BDA se han propuesto diferentes plataformas, prototipos y sintaxis para la especificación de las reglas ECA, sin embargo hay características importantes, que están inherentes dentro del desarrollo de reglas ECA, que en algunas propuestas no son consideradas, en algunas otras sí y viceversa. Además, cada una de las propuestas está pensada en el diseño de un SBDA, no en un modelo que cubra las características propias de una base de reglas ECA y que pueda ser aplicado a cualquier diseño de SBDA.

Los SBDA se utilizan para sistemas de control de misiones de aeronaves, sistemas de navegación de a bordo, en la industria manufacturera de automóviles, en aplicaciones de telecomunicaciones, sistemas de simulación, subastas por comercio electrónica y ciertos aspectos de workflows.

En vista de lo anterior, se está proponiendo un modelo basado en Teoría de RdP, en el cual se ofrece un modelo formal con características de una RdP Coloreada, al que se ha denominado Red de Petri Coloreada Condicional (CCPN, Conditional Colored Petri Net).

En el modelo propuesto en este marco es posible almacenar suficiente información de la base de reglas ECA para proporcionar el comportamiento activo adecuado a una BD. Además, el modelo de conocimiento de una regla ECA se puede incorporar dentro de la CCPN, y el modelo de ejecución de la regla ECA se ajusta al disparo de transiciones de la teoría de RdP. Finalmente, dentro de la CCPN es posible simular, ejecutar y analizar una base de reglas ECA. Aunado a lo anterior, dentro de la propuesta se realiza el análisis de las reglas, ofreciendo una herramienta de análisis de reglas basado en la matriz de incidencia de RdP.

C. Organización del trabajo

El contenido del presente trabajo en las secciones subsecuentes está desarrollado de la siguiente manera: en la sección 2 se da una breve descripción sobre los conceptos básicos que engloba la teoría de RdP; en la sección 3 se describe en qué consiste el modelo que se propone para la modelación de las reglas ECA, utilizando los conceptos de la teoría de RdP; la conversión del conjunto de reglas ECA hacia una estructura de red de Petri se describe en la sección IV; en la sección V se explica la forma en que la red de Petri se dispara de acuerdo al modelo de ejecución que presenta un conjunto de reglas ECA; finalmente, en la sección VI se muestran algunas de las conclusiones obtenidas en el desarrollo del presente trabajo.

II. CONCEPTOS BÁSICOS DE REDES DE PETRI

Una RdP 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 [11]. Gráficamente, los lugares son representados por círculos y las transiciones son representadas por barras o por rectángulos. En su forma más simple, una RdP 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 tokens. 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.

Formalmente, una RdP puede definirse como sigue: [12]

Definición 1. *Una RdP es una 5-tupla, RdP = {P, T, F, W, M_0}, donde*

$P = \{p_1, p_2, ..., p_m\}$ es un conjunto finito de lugares.

$T = \{t_1, t_2, ..., t_n\}$ es un conjunto finito de transiciones.

$F \subseteq (P \times T) \cup (T \times P)$ es un conjunto de arcos (relación de flujo)

$W : F \rightarrow \{1, 2, 3, ...\}$ es una función de peso.

$M0 : P \rightarrow \{0, 1, 2, 3, ...\}$ es el marcado inicial.

$$P \cap T = \varnothing \text{ y } P \cup T \neq \varnothing.$$

El marcado es la asignación de tokens a los lugares de una RdP. Un token es un concepto primitivo que forma parte de las RdP's (como los lugares y las transiciones), el cual puede ser asignado a un lugar de la RdP para denotar el estado del sistema modelado. Los tokens son usados para definir la ejecución de una RdP, por lo tanto, la cantidad y la posición de los tokens puede cambiar durante la ejecución.

Las RdP's pueden realizar la descripción y el análisis de sistemas que procesan información, los cuales se caracterizan por ser concurrentes, asíncronos, distribuidos, paralelos, indeterministas, y/o estocásticos [12]. Para la modelación de éstos sistemas, podemos utilizar las RdP de dos maneras: gráficamente y matemáticamente.

Gráficamente las RdP pueden utilizarse como una ayuda de comunicación visual parecidas a los diagramas de flujo, diagramas de bloques y redes. Además, los tokens son utilizados en este tipo de redes para simular las actividades dinámicas y concurrentes de un sistema. Como una herramienta matemática, es posible obtener ecuaciones de estado, ecuaciones algebraicas, y otros modelos matemáticos que gobiernen el comportamiento de los sistemas.

III. Modelo propuesto

Existen muchas propuestas para soportar comportamientos y mecanismos reactivos dentro de un SBD para generar lo que se conoce como SBDA. Sin embargo estas propuestas están diseñadas para un sistema en particular, sin permitir su migración a otros ambientes, además de que no existe una propuesta de SBDA formalizada.

En este trabajo se está proponiendo un modelo general para el desarrollo de reglas ECA, basado en teoría de RdP, el cual puede utilizarse como un motor independiente sobre cualquier SBD.

Como se mencionó anteriormente, un SBDA debe ofrecer un modelo de conocimiento y un modelo de ejecución. En el modelo de conocimiento se especifican cada uno de los elementos que conforman a la regla ECA, es decir, el evento, la condición y la acción. En la conversión de una regla ECA a una CCPN, sus elementos son convertidos en elementos de 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 RdP's, la evaluación de una expresión lógica.

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.

El modelo de ejecución para la CCPN se basa en la regla de disparo de transiciones de la teoría de RdP, proporcionando mecanismos para la generación de tokens con información, o color, correspondiente a los eventos que estén ocurriendo en la BD. Los tokens generados son colocados en los lugares que representen a esos eventos. De esta manera se procesan todas las reglas definidas y se detectan los eventos tanto primitivos como compuestos.

A. Definición formal de la CCPN

La Red de Petri Coloreada Condicional (Conditional Colored Petri Net, CCPN) es una extensión de RdP, la cual hereda atributos y la regla de disparo de transiciones de RdP. Además, en la CCPN se toman conceptos que están presentes en la definición de la RdP coloreada, 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 RdP's coloreadas la asignación de tipos de dato se hace hacia todos los lugares de la red, 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 una 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.

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.

Formalmente, y siguiendo la misma estructura en la definición de una RdP, la CCPN puede ser definida como:

Definición 2. *Una red de Petri coloreada condicional (conditional colored Petri net, CCPN) es una 11-tupla*

CCP N = (Σ, P, T, A, N, C, Con, Acción, D, τ , I)

Donde:

(i) Σ es un conjunto finito de tipos de datos, también llamados conjuntos de colores.

(ii) P es un conjunto finito de lugares. Para una representación gráfica más adecuada, P está dividido en los siguientes subconjuntos,

$$P = P_{prim} \cup P_{comp} \cup P_{virtual} \cup P_{copy}$$

Donde P_{prim}, P_{comp}, $P_{virtual}$ y P_{copy} representan a lugares primitivos, compuestos, virtuales y copia, respectivamente.

(iii) T es un conjunto finito de transiciones. Existen tres tipos de transiciones,

$$T = T_{regla} \cup T_{comp} \cup T_{copy}$$

Donde T_{regla}, T_{comp} y T_{copy} representan a transiciones tipo regla, compuestas y copia, respectivamente

(iv) A es un conjunto finito de arcos, tales que

$$P \cap T = P \cap A = T \cap A = \varnothing.$$

Donde A está dividido en dos subconjuntos, A_{inh} y A_{nor}, los cuales representan a los conjuntos de arcos inhibidores y arcos normales, respectivamente.

(v) $N : A \rightarrow P \times T \cup T \times P$ es una función nodo. Está definida desde A hacia $P \times T \cup T \times P$.

(vi) $C : P \rightarrow 2^{\Sigma}$ es una función color. Está definida desde el conjunto P hacia 2^{Σ}.

(vii) *Con* es una función para evaluar condiciones. Está definida desde una transición $t \in T_{regla} \cup T_{comp}$ hacia expresiones tales que $\forall t \in T_{regla}$: [Type(Con(t)) = (Falso | Verdadero)],

Donde la función *Con* evalúa la condición de la regla ECA. $\forall t \in$ Tcomp : [T ype(Con(t)) = (Falso | Verdadero)],

Donde la función Con evalúa el intervalo de tiempo, almacenado en t, contra la estampa de tiempo de los tokens.

(viii) *Acción* es una función sobre la acción de la regla ECA. Está definida desde el conjunto T_{regla} hacia expresiones tales que:

$$\forall t_n \in T_{regla}, p \in t_n : [Type(Acción(t_n)) = C(p)]$$

(ix) $D : T_{comp} \rightarrow R \times R$ es una función de intervalos de tiempo. Está definida desde el conjunto T_{comp} hacia intervalos de tiempo $[d_1(t), d_2(t)]$, donde $t \in T_{comp}$ y $d_1(t)$, $d_2(t)$ son los instantes inicial y final, respectivamente, del intervalo de tiempo.

(x) $\tau : M(p) \rightarrow R^+$ es una función de estampas de tiempo. Está definida desde la marca M(p) de un lugar p, hacia $\{0\} \cup R^+$, la cual asigna a cada token del lugar p una estampa de tiempo correspondiente a un instante del reloj en la forma año : mes : día − hora : minuto : segundo, por ejemplo, un token puede tener una estampa de tiempo como 2008 : 10 : 06 − 16 : 30 : 00.

(xi) $I : P \rightarrow C(p)$ es una función de inicialización de valores en la CCPN. Está definida a partir del conjunto P hacia el conjunto de colores C(p) .

B. Elementos de la CCPN

El conjunto de tipos de datos Σ, o conjuntos de colores, determinan los tipos de datos que serán utilizados dentro de la CCPN para efectos de manipular la información concerniente a eventos de la BD, evaluación de expresiones lógicas en la parte condicional y los datos necesarios para llevar a cabo la acción de las reglas ECA.

El conjunto de lugares P está dividido en los conjuntos P_{prim}, P_{comp}, $P_{virtual}$ y P_{copy}. P_{prim} es un conjunto finito de lugares, los cuales representan a los eventos primitivos de las reglas ECA, gráficamente son representados mediante un círculo de un sola línea. Fig. 1 (a). Este tipo de lugares pueden ser lugares de entrada para cualquier tipo de transición $p \in P_{prim}$, $p \in \bullet t$, $t \in T$, pero solo pueden ser lugares de salida de las transiciones T_{regla}, debido a que representan operaciones primitivas y éstas son utilizadas para describir la acción de la regla ECA, $p \in P_{prim}$, $p \in t\bullet$, $t \in T_{regla}$.

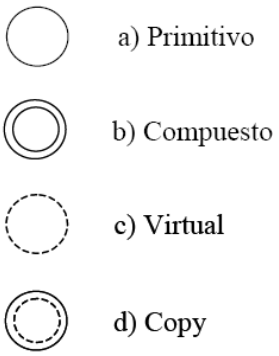

Figura 1. Clasificación de los lugares en la CCPN.

P_{comp} es un conjunto finito de lugares, mediante los cuales se representan a los eventos compuestos de las reglas ECA, donde se manejan estampas de tiempo. Estos lugares son representados gráficamente con un círculo de doble línea. Figura 1 (b). Este tipo de lugares son lugares de salida para las transiciones de tipo T_{comp} que se utilizan para representar a eventos compuestos, es decir, $\forall p \in P_{comp}[p \in t\bullet, t \in T_{comp}]$. Pero son lugares de entrada para cualquier tipo de transición, $\forall p \in P_{comp}[\ p \in \bullet t, t \in T\ ]$. Es decir, un lugar $p \in P_{comp}$ representa a un evento compuesto que dispara a una regla ECA ($p \in \bullet t$, $t \in T_{regla}$); o a un evento compuesto que se utiliza para disparar a dos o más reglas ECA ($p \in \bullet t$, $t \in T_{copy}$); o bien, a un evento compuesto que forma parte de otro evento compuesto ($p \in \bullet t$, $t \in T_{comp}$).

$P_{virtual}$ es un conjunto de lugares virtuales. Éstos lugares solamente se utilizan para almacenar tokens en la formación de los eventos compuestos conjunción y disyunción. Gráficamente se representan por un círculo punteado. Figura 1 (c). Los lugares virtuales son lugares de salida de transiciones T_{comp} para el caso del evento compuesto conjunción o bien, son lugares de salida de transiciones tipo T_{copy} para el caso del evento compuesto disyunción. Por otro lado, éste tipo de lugares puede ser lugar de entrada para cualquier transición,

$\forall p \in P_{comp}[\ p \in \bullet t, t \in T\ ]$. Como en el caso de P_{comp}, un lugar $p \in P_{virtual}$ representa a un evento compuesto que dispara a una regla ECA, a un evento compuesto que se utiliza para disparar a dos o más reglas ECA, o a un evento compuesto que forma parte de otro.

P_{copy} es un conjunto de lugares que son utilizados para replicar eventos en el disparo de dos o más reglas. Cuando un mismo evento participa en el disparo de dos o más reglas, el procesamiento de las reglas se realiza de manera separada, por lo que es necesario generar réplicas del evento para el proceso de evaluación. Las réplicas de los eventos que se generan, son representadas por lugares tipo P_{copy}. Gráficamente, los lugares tipo P_{copy} son representados mediante un circulo dibujado con doble línea, cuya línea interior es una línea punteada, tal y como se muestra en la figura 1 (d). Los lugares tipo P_{copy} solamente son lugares de salida para transiciones de tipo T_{copy}, $\forall p \in P_{copy}[\ p \in t\bullet, t \in T_{copy}\ ]$; y son lugares de entrada para transiciones de tipo T_{regla} y T_{comp}, $\forall p \in P_{copy}[p \in \bullet t, t \in T_{regla} \mid t \in T_{comp}]$.

El conjunto de transiciones T está dividido en los conjuntos T_{regla}, T_{comp} y T_{copy}.

Una transición $t \in T_{regla}$ representa a una regla ECA, en la cual se almacena la parte condicional de la regla y está conectada con un lugar de entrada $p_1 \in P$, donde p_1 es el evento activador de la regla, el cual puede ser un lugar de tipo P_{prim}, P_{comp}, $P_{virtual}$ o P_{copy}; y con uno o más lugares de salida $p_2 \in P_{prim}$, dependiendo del número de acciones de la regla ECA. Gráficamente, una transición $t \in T_{regla}$ es representada mediante un rectángulo. Figura 2 (a).

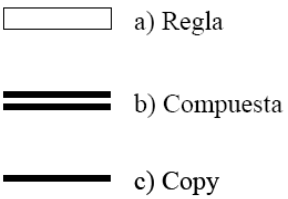

Figura 2. Clasificación de las transiciones en la CCPN.

Una transición $t \in T_{comp}$ es utilizada para formar las estructuras de los eventos compuestos, teniendo como lugares de entrada el o los eventos que conforman al evento compuesto. Todos los tipos de lugares son válidos como lugares de entrada para este tipo de transición, es decir, $\bullet t = \{p \mid p \in P\}$; y los lugares de salida para t son lugares virtuales o compuestos, $t\bullet = \{p \mid p \in P_{virtual} \cup P_{comp}\}$. Las transiciones $t \in T_{comp}$ se representan gráficamente mediante una doble barra. Figura 2 (b).

Una transición $t \in T_{copy}$ es utilizada para replicar el token de su lugar de entrada y se aplica en dos casos, i) cuando el lugar de entrada $p \in \bullet t$ representa a un evento, sea primitivo o compuesto, y es utilizado en dos o más ocasiones, ya sea como evento activador de una regla o para conformar eventos compuestos, teniendo como lugar de salida a un lugar $p_2 \in P_{copy}$, $t\bullet = \{p_2\}$; y ii) para formar a los eventos compuestos *disyunción* y *alguno*, teniendo como lugar de salida a un lugar $p_i \in P_{virtual}$, es decir, $t\bullet = \{p \mid p \in P_{virtual}\}$. Los lugares de entrada válidos para t son $\bullet t = \{p \mid p \in P_{prim} \cup P_{comp} \cup P_{virtual}\}$. Las transiciones $t \in T_{copy}$ se representan gráficamente mediante una barra. Figura 2 (c).

El conjunto de arcos A de la CCPN está dividido en los conjuntos A_{inh} y A_{nor}. El conjunto de arcos $a \in A_{nor}$ está formado por los arcos normales de teoría de RdP y se utilizan para conectar lugares y transiciones donde no se especifique el evento compuesto negación. Los arcos A_{nor} se representan gráficamente con una flecha indicando los nodos que conecta. Figura 3 (a).

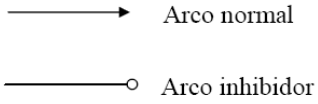

Figura 3. Clasificación de los arcos en la CCPN.

El conjunto de arcos $a' \in A_{inh}$ está formado por arcos inhibidores y se utiliza para formarla estructura del evento compuesto negación, el cual conecta a un lugar de entrada $p \in P$, de cualquier tipo, hacia una transición $t \in T_{comp}$. En teoría de RdP, un arco inhibidor tiene la función de habilitar una transición cuyos lugares de entrada no cumplan con la regla de disparo de transiciones y viceversa. Un evento compuesto de negación es, precisamente, la no ocurrencia de un evento en determinado intervalo, de tal manera que el arco inhibidor es útil para este fin, ya que un evento lo estamos representando con un lugar de entrada para una transición. La representación gráfica de este tipo de arcos es una línea que conecta a un lugar de entrada con una transición $t \in T_{comp}$, la cual tiene un círculo pequeño en el extremo que conecta a t. Figura 3 (b).

La función nodo *N* mapea cada arco de la CCPN en pares ordenados de números enteros, los cuales indican los índices de las transiciones y los lugares que están conectadas mediante un arco $a \in A$.

La función *Color* mapea cada lugar $p \in P$ con un tipo de color C(p), es decir, los datos almacenados dentro de un lugar, pertenecen a un tipo de color

Cuando la función de condición *Con* evalúa la condición de la regla ECA, mapea cada transición $t \in T_{rule}$ a una expresión lógica donde todas las variables tienen un tipo de dato que pertenece a Σ. Si la evaluación de *Con* resulta verdadera, entonces significa que la regla se dispara, produciendo el disparo de t. Por otro lado, la función Con verifica si la estampa de tiempo de los tokens provenientes de los lugares de entrada, hacia una transición $t \in T_{comp}$, cumplen con el intervalo de tiempo de t.

La función *Acción* mapea cada transición $t \in T_{rule}$, hacia un conjunto de valores de algún tipo de datos C(p), los cuales serán depositados en su correspondiente lugar de salida.

El intervalo de tiempo D es utilizado por la función *Con* para evaluar transiciones $t \in T_{comp}$.

Los tokens alojados en un lugar p pueden tener diferentes estampas de tiempo, así, utilizamos la función τ para cada token $M(p_i)$.

La función de inicialización *I* mapea cada lugar, p, hacia expresiones que deben de ser de tipo C(p).

Una parte importante dentro de una regla ECA es el evento que se desea detectar. La detección de eventos primitivos es una tarea sencilla, ya que solamente se estaría monitoreando un cambio específico de la BD. Sin embargo, el proceso de detección de eventos se complica a medida de que se desee incrementar la detección de la combinación de eventos primitivos, y en su caso de eventos formados a partir de la combinación de otros. La combinación de eventos genera un tipo de eventos denominados eventos compuestos.

C. *Modelación de reglas ECA con CCPN*

En la definición de una base de reglas ECA, utilizando la CCPN, se toma a cada una de las reglas para convertir sus elementos en elementos de la CCPN. Dado un conjunto de reglas ECA $R = \{r_1, r_2, \ldots, r_n\}$, para cada regla $r_i(e_i, c_i, a_i)$, $i = 1, 2, \ldots, n$, el evento e_i es modelado con una estructura de CCPN, en el caso de eventos primitivos se utilizan lugares $p \in P_{prim}$, y para el caso de eventos compuestos se generan estructuras correspondientes a cada evento compuesto. La condición de la regla c_i se almacena dentro de una transición $t \in T_{regla}$, la cual será evaluada durante el proceso de disparo de la transición t. La acción de la regla ECA es una operación que modifica al estado de la BD; esta operación puede ser cualquiera que se encuentra dentro de la fuente de eventos, y es representada por un lugar $p \in P_{prim}$. En la figura 4, se ilustra la migración del evento, la condición y la acción de una regla ECA hacia elementos de la CCPN. El lugar p_1 que representa al evento e_i de la regla puede ser de cualquier tipo ($p_1 \in P_{prim} \cup P_{comp} \cup P_{virtual} \cup P_{copy}$), por simplicidad se dibuja a un lugar P_{prim}. La condición c_i se almacena dentro de la transición $t_1 \in T_{regla}$. Finalmente, la acción de la regla ECA a_i se traduce en un lugar de salida $p_2 \in P_{prim}$ de t, tal que $t\bullet = \{p_2\}$. Podemos decir que la regla ECA está siendo representada por t, porque en t se conocen los elementos de la regla: el evento es su lugar de entrada, la condición está almacenada en ella y la acción es su lugar de salida.

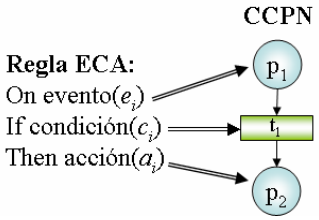

Figura 4. Migración de los elementos de la regla ECA (evento-condición-acción) a elementos de una CCPN.

Sin embargo, existen bases de reglas donde un mismo evento (primitivo o compuesto) disparan a dos o mas reglas. Utilizando el modelo de CCPN, el evento que tiene que participar más de una ocasión en la conformación de eventos compuestos o como evento de una regla ECA, se replica para ser considerado en los lugares donde sea solicitado. En la figura 5 se muestra la participación de un evento primitivo p, el cual dispara a dos reglas distintas. Dado un conjunto de reglas R, dos reglas $r_1(e_1, c_1, a_1)$, $r_2(e_2, c_2, a_3) \in R$ y $e_1 = e_2$, el evento de ambas reglas se mapea a un lugar p_1, pero como p_1 se utiliza en la evaluación de dos reglas, la información de p_1 se replica a los lugares p_2 y p_3, utilizando una transición $t_1 \in T_{copy}$ y los lugares $p_1, p_2 \in P_{copy}$, es decir, $\bullet t_1 = \{p_1\}$ y $t\bullet_1 = \{p_2, p_3\}$. De esta manera, ambas reglas, representadas mediante t_2 y t_3, utilizarán la información del mismo evento para evaluar sus condiciones, c_1 y c_2, almacenadas en las transiciones $t_2, t_3 \in T_{regla}$, respectivamente, $\bullet t_2 = \{p_2\}$ y $\bullet t_3 = \{p_3\}$. Por último, la acción de ambas reglas se representan por los lugares p_4 y p_5, $t\bullet_2 = \{p_4\}$ y $t\bullet_3 = \{p_5\}$.

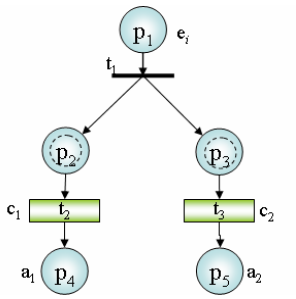

Figura 5. Estructura de una CCPN cuando un mismo evento e_1 dispara a dos reglas distintas.

En una base de reglas ECA existen relaciones entre reglas. Además de que dos reglas compartan el mismo evento, dos reglas se relacionan si el evento de una regla es a la vez la acción de la otra. Dado un conjunto de reglas ECA R, dos reglas $r_1(e_1, c_1, a_1)$, $r_2(e_2, c_2, a_3) \in R$ y $a_1 = e_2$, entonces la estructura a que da lugar esta relación es como se muestra en la figura 6. La regla r_1 está representada por la transición t_1 y la regla r_2 está representada por la transición t_2; $t_1, t_2 \in T_{regla}$.

El evento e_1 está representado por el lugar p_1, lugar de entrada para t_1 ($\bullet t_1 = \{p_1\}$) y la acción a_1 está representada por el lugar p_2, lugar de salida de t_1 ($t\bullet_1 = \{p_2\}$). Dado que $a_1 = e_2$ entonces e_2 está representado también por p_2, entonces $\bullet t_2 = \{p_2\}$ y $t\bullet_2 = \{p_3\}$. Por lo tanto $t\bullet_1 = \bullet t_2$.

IV. Algoritmo de conversión de reglas ECA a CCPN

Para llevar a cabo la conversión de una base de reglas ECA a una CCPN se utiliza un algoritmo que toma la definición de la base de reglas ECA en su forma general *on - if - then* y genera los lugares y transiciones correspondientes.

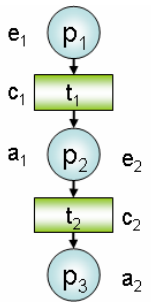

Figura 6. Relación existente entre dos reglas cuando la acción de una de ellas provoca el disparo de la otra.

A. Archivos ECA

Los archivos ECA donde se define la base de reglas ECA contiene la definición del esquema conceptual de la BD sobre la que se definirán las reglas. En el esquema conceptual se definen las estructuras de las tablas que participan en la BD, así como las relaciones existentes entre las tablas.

Además se describen los eventos que activan a las reglas ECA; finalmente se especifican las reglas en el formato *on evento*, *if condición*, *then acción*.

Los eventos primitivos se denotan utilizando los comandos básicos que se manejan en SQL para la modificación y el acceso a la información de la BD (*insert*, *update*, *delete*, *select*). Además, el nombre del comando se complementa con la tabla y, en su caso, el campo que afectan.

En el caso del comando *insert*, se utiliza el nombre del comando y el nombre de la tabla, el campo no es necesario porque en este caso no estamos insertando solo el campo sino todo el registro completo: *insert_tabla*.

Para nombrar a un evento primitivo que modifica a la BD con el comando *update*, si debemos especificar el nombre de la tabla y el nombre del campo, porque en este caso no necesariamente se modifica todo el registro sino que solamente puede modificarse un solo campo y para ubicar al evento debemos conocer la tabla y el campo que está siendo modificado: *update_tabla_campo*.

El comando que elimina un registro (*delete*), al igual que en el caso del comando *insert*, no necesita la especificación del campo porque se elimina el registro completo y no solamente el campo: *delete_tabla*.

El comando *select* no modifica el estado de la BD, sin embargo en ocasiones es importante conocer quién, cuando o como están accesando a la BD, por lo tanto es necesario incluirlo como evento primitivo. En este caso el nombre para identificar a éste evento se forma, al igual que el comando *update*, con el nombre del comando, la tabla y el campo al que se está accesando: *select_tabla_campo*.

La condición de la regla se especifica como una expresión lógica donde participan valores de las tablas de la BD o valores que se encuentran dentro del mismo evento activador de la regla. Si no se quiere especificar la condición simplemente se establece un valor *true* en la parte del *if*.

La acción de la regla se define mediante instrucciones de SQL, especificando la acción o acciones que se realizarán en el dado caso de que se dispare la regla y la condición sea evaluada a verdadera.

La definición de la base de reglas ECA observando estos lineamientos se almacenan en un archivo de texto, asignándole la extensión .eca, para identificar los archivos que contienen definiciones de reglas ECA. Este archivo es utilizado por el algoritmo de conversión de reglas ECA a una CCPN.

B. Descripción del algoritmo

El algoritmo inicia con la definición de las estructuras que se utilizan para almacenar los datos de las reglas ECA. En primer lugar se tiene el conjunto BD para almacenar el esquema conceptual de la base de datos, es decir, las tablas definidas en la base de datos (o relaciones si utilizamos los conceptos del modelo relacional de BD), sobre la que se aplicarán las reglas, son los elementos de éste conjunto. Además, se tienen los conjuntos de lugares (P), transiciones (T) y arcos (A).

Se recibe como entrada un archivo de texto (Reglas.eca) que contiene la definición del esquema conceptual de la BD y las reglas ECA en la forma *on − if − then*. Primero se lee el esquema conceptual BD, el cual obtiene la definición de las tablas descritas en el archivo de texto leído. Posteriormente se generan las estructuras correspondientes a los eventos a detectar por las reglas ECA.

El algoritmo comienza a leer cada una de las reglas ECA definidas hasta que ya no encuentra y termina su ejecución. Para cada regla ECA leída se crea una transición $t \in T_{regla}$, a la cual se le asigna la condición de la regla y es agregada al conjunto T. Se busca en P el lugar p_1 que represente al evento de la regla. Si p_1 no es lugar de entrada de ninguna transición ($p^{\bullet}_1 = \varnothing$), se crea un arco $a_1(p_1, t)$ y se agrega al conjunto de arcos A. En caso de que p_1 sea el lugar de entrada de una transición $t_1 \in T$ se debe verificar que tipo de transición es t_1. Si $t_1 \in T_{copy}$ se crea un lugar $p_3 \in P_{copy}$, el cual se agrega al conjunto P; se crean los arcos para conectar t_1 con p_3 ($a_1(t_1, p_3)$) y para conectar p_3 con t ($a_2(p_3, t)$) y se agregan al conjunto de arcos ($A \leftarrow A \cup \{a_1, a_2\}$). Este proceso se muestra en la figura 8 (a). En caso contrario, t1 $\notin$ Tcopy , entonces t1 $\in$ $T_{regla} \cup T_{comp}$, y se hace lo siguiente: Crear $t_2 \in T_{copy}$, agregar t_2 a T, crear lugares p_2, $p_3 \in P_{copy}$ y agregarlos al conjunto P, eliminar el arco que une a p_1 con t_1, crear los arcos $a_1(p_1, t_2)$, $a_2(t_2, p_2)$, $a_3(t_2, p_3)$, $a_4(p_2, t_1)$, $a_5(p_3, t)$ y agregarlos al conjunto de arcos $A \leftarrow A \cup \{a_1, a_2, a_3, a_4, a_5\}$. Este proceso se muestra en la figura 8 (b), donde t_1 puede ser de tipo regla o de tipo compuesto, en este caso se muestra para $t_1 \in T_{regla}$.

Después de conectar el evento de la regla (lugar) con la condición (almacenada en la transición) ahora se prosigue con la acción de la regla. Primero se verifica si ya existe un lugar $p_j \in P$ que represente a la acción de la regla. Si p_j no existe aún entonces se crea un $p_j \in P_{prim}$, se le asigna la acción de la regla y se agrega al conjunto de lugares. Posteriormente se crea un arco para conectar t (que tiene almacenada la condición de la regla) con el lugar p_j. Si ya existiese el lugar p_j en el conjunto P entonces simplemente se crea el arco $a_1(t, p_j)$ para conectarlos, y se agrega a_1 al conjunto de arcos.

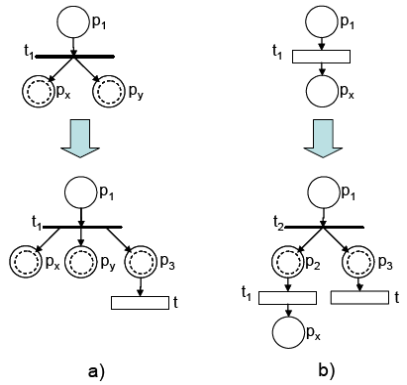

Figura 8. Proceso de para agregar transiciones tipo *copy*.

V. Ejecución de la CCPN

La ejecución de la CCPN difiere un poco de la ejecución de la RdP, está basada en la regla de disparo de teoría de RdP, sin embargo, ésta es insuficiente para modelar el disparo de reglas ECA y la formación de eventos compuestos.

Primero se verifica si la transición $t \in T$ cumple con la regla de disparo de RdP normal. Posteriormente se evalúa la transición, dependiendo el tipo de transición de que se trate. El disparo de las transiciones se maneja de diferente manera, la cual depende del tipo de transición.

Un elemento importante dentro de la ejecución de RdP son los tokens, los cuales rigen el comportamiento de la RdP. En CCPN se almacenan datos dentro de los tokens, concepto tomado de la RdP coloreada. Además, para la conformación de los eventos compuestos, debemos conocer el tiempo específico en que los eventos ocurrieron, así como el lugar al que pertenecen en algún estado que guarde la BD en determinado momento. Por lo tanto, definimos a un elemento token de la CCPN así:

Definición 3. *En CCPN, un elemento token es una 3-tupla (p, c, stamp) donde $p \in P$, $c \in C(p)$ y stamp indica el punto en el tiempo en que ocurre el evento correspondiente y el token es depositado en p. El conjunto de todos los elementos token está denotado por TE. Una marca M es un multiconjunto sobre TE. La marca inicial M_0 es obtenida al evaluar las expresiones de inicialización*:

$$\forall (p, c, stamp) \in TE : M_0(p, c, stamp) = I(p).$$

El conjunto de todas las marcas es expresado por R(M).

Introducimos una notación nueva, $N_{color}(p)$, la cual expresa el número de diferentes tipos de colores que está presentes en un lugar p. Esta función tendrá valores mayores a 1 en los lugares $p \in P_{virtual}$, debido a los tokens que son suministrados por los lugares que le preceden. Si $p \in P_{prim} \cup P_{comp} \cup P_{copy}$ entonces los tokens alojados en p tendrán el mismo tipo de color.

Cuando una transición t es habilitada, la función de habilitación $C_{enabled}$ es definida para especificar los tokens que provocaron la habilitación de t. En CCPN, una transición $t \in T_{copy}$ se dispara siempre y cuando esté habilitada. Pero, una transición $t \in T_{comp} \cup T_{regla}$ dispara de manera condicionada. Una transición $t_i \in T_{comp}$ dispara si está habilitada y si la condición del intervalo de tiempo se cumple. Y una transición $t_j \in T_{regla}$ dispara si está habilitada y si la evaluación de la condición de la regla ECA almacenada en t_j, contra el estado que guarde la BD, resulta verdadera.

Cuando una transición se habilita, no significa que además dispare. El disparo de una transición depende de la evaluación adicional que se tenga que hacer. Si una transición t se habilita, pero no es disparada, entonces el token que la activó es eliminado de la CCPN, con el fin de desechar información no necesaria y evitar la acumulación masiva de datos. Las transiciones que pueden habilitarse y no ser disparadas son transiciones $t \in T_{regla} \cup T_{comp}$, porque la evaluación de la condición de la regla ECA almacenada en una $t \in T_{regla}$ puede resultar falsa, entonces la transición no se dispara. Por otro lado, una transición $t \in T_{comp}$ puede habilitarse ante la presencia de los eventos que conforman al evento compuesto, pero si no cumplen con el intervalo de tiempo D especificado y la condición adicional que estipula el propio evento compuesto, entonces t no se dispara. La única excepción de las transiciones $t \in T_{comp}$ es cuando Type(t) = Negación, porque cuando t se habilita significa que el intervalo D ha terminado y no ocurrió el evento representado por p, $\{p\} = \bullet t$, entonces se procede a disparar a t. Por lo tanto, el desplazamiento de tokens en caso de que una transición $t \in T_{regla} \cup (T_{comp} - \{t \mid \text{Type}(t) = \text{Negación}\})$ no se dispare es el siguiente:

VI. Conclusiones

La red de Petri coloreada condicional (CCPN, Conditional Colored Petri Net) es una extensión de RdP a la que se le han agregado los elementos necesarios para la modelación de reglas activas.

La CCPN es un modelo formal de representación de reglas ECA donde, a partir de la misma estructura de CCPN obtenida para un conjunto de reglas ECA, puede llevarse a cabo el análisis de las reglas, la simulación del comportamiento de las reglas sobre una BD y la ejecución de las reglas sobre un SBD real.

Además ofrece una definición formal, donde se especifican los elementos de la regla ECA y sus elementos correspondientes en la CCPN. El evento de la regla es mapeado hacia un lugar de la CCPN (evento primitivo) ó a una estructura de CCPN (evento compuesto); la condición de la regla se almacena dentro de una transición; y la acción de la regla, debido a que modifica el estado de la BD se considera como evento, también se representa por un lugar de la CCPN.

Este modelo cuenta con un conjunto P de lugares, el cual se divide en cuatro subconjuntos: el conjunto P_{prim}, para representar eventos primitivos; P_{comp}, para representar eventos compuestos; P_{copy}, para representar eventos que son requeridos en al menos dos ocasiones como evento de regla o como elemento constituyente de un evento compuesto; y $P_{virtual}$, cuyos elementos son utilizados como almacenes de tokens.

Incluye un conjunto de transiciones T, dividida en los conjuntos T_{regla}, donde se almacena la parte condicional de la regla, el evento que debe detectarse y la acción a ejecutarse; T_{comp}, cuyos elementos son utilizados para la especificación de los eventos compuestos; y T_{copy}, cuyos elementos se utilizan para replicar tokens con información sobre algún evento, hacia los lugares que pertenecen a P_{copy}.

La CCPN utiliza dos tipos de arcos; el arco normal que conecta a un lugar con una transición o viceversa, trasladando tokens entre estos elementos. Y el arco inhibidor, el cual envía tokens hacia una transición siempre y cuando el lugar de entrada contenga un número de tokens menor al peso del arco inhibidor. Este último arco es útil en la definición del evento compuesto negación.

Al igual que en un SBDA, la CCPN cuenta con reglas de disparo, considerando que, además de la regla de disparo para una RdP, en las transiciones de tipo T_{regla} la evaluación de la condición debe resultar verdadera y en las transiciones de tipo

T_{comp} deben evaluarse las estampas de tiempo correspondientes a la ocurrencia de eventos así como los intervalos de tiempo en que los eventos deben de ocurrir en la formación de eventos compuestos.

Con este modelo, es posible llevar a cabo el desarrollo de bases de reglas ECA, de tal manera que el desarrollo de las reglas puede hacerse en un ambiente gráfico, aprovechando las ventajas de análisis que están inmersas en la teoría de RdP. A partir de la misma estructura de la RdP, el análisis y validación de la reglas puede llevarse a cabo, antes de montarlas sobre un sistema de BD en funcionamiento.

REFERENCES

[1] Y. I. Chan, F. L. Chen, "RBE: a rule-by-example active database system", *Software -Practice and experience*, Vol. 27, No. 4, pp. 365-394, April 1997;.

[2] N. W. Paton, O. Diaz, "Active Database Systems", *ACM Computing Surveys*, Vol. 31, No. 1, pp. 64-103, 1999.

[3] D. McGoveran, C. J. Date., *A guide to SYBASE and SQL Server : a user's guide to the SYBASE product*, Sybase, Inc, 1992.

[4] T. Lacy-Thompson, *INFORMIX-SQL, A tutorial and reference*, ISBN-0-13-465121-9, Ed. Prentice Hall, 1990.

[5] C.J. Hursh, J.L. Hursch, *Oracle SQL Developer's Guide*, ISBN-0-8306-2529-1, Ed. McGraw-Hill, 1991.

[6] A. González-Pérez, *SQL Server, Programación y administración*, ISBN 970-15-0376-7, Ed.Alfaomega ra-ma, 1999.

[7] L. Warshaw and L. Obermeyer and D. Miranker and S. Matzner, "VenusIDS: An Active Database Component for Intrusion Detection", *Applied Research Laboratories*, University of Texas, Austin, 1999, url = citeseer.ist.psu.edu/warshaw99venusids.html.

[8] E.N. Hanson, "The Design and Implementation of the Ariel Active Database Rule System", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 1, february 1996.

[9] J. Widom, "The Starburst Active Database Rule System", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 4, August 1996.

[10] M. Stonebraker, G. Kemmintz, "The POSTGRES Next-Generation Database Management System", *Communications of the ACM*, Vol. 34, No. 10, October 1991.

[11] J.L. Peterson, *Petri Net Theory and The Modeling of Systems*, Prentice-Hall, Inc., 1981. ISBN 0-13-661983-5.

[12] T. Murata, "Petri Nets: Properties, analysis, and applications", *Proceedings of the IEEE*, 77(4):541-580, 1989.

Joselito Medina-Marín. Recibió el título de Ingeniero en Computación por la Facultad de Ingeniería de la Universidad Autónoma de Guerrero en 1997. Obtuvo el grado de Maestro en Ciencias y Doctor en Ciencias especialidad Ingeniería Eléctrica opción Computación por parte del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional en 2002 y 2005, respectivamente. Actualmente se desempeña como Profesor Investigador en el Centro de Investigación Avanzada en Ingeniería Industrial de la Universidad Autónoma del Estado de Hidalgo. Es miembro del Sistema Nacional de Investigadores en México, y cuenta con el perfil PROMEP otorgado por la Secretaría de Educación Pública. Sus líneas de investigación están orientadas hacia simulación de sistemas, bases de datos activas y redes de Petri.

Xiaoou Li. Recibió los títulos de Licenciatura en Matemáticas Aplicados y Doctorado en Ciencias en Ingeniería Eléctrica por parte de la Universidad de Northeastern, en Shenyang, China, en 1991 y 1995, respectivamente. De 1995 a 1997, trabajó como Lecturer en Ingeniería Eléctrica en el Departamento de Control Automático, en la Universidad Northeastern. De 1998 a 1999, trabajó como Profesor Asociado de Ciencias de la Computación en el Centro de Instrumentos, Universidad Nacional Autónoma de México, México. Desde el año 2000, ha sido Profesor de Ciencias de la Computación en el Departamento de Computación, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV-IPN), Ciudad de México. Sus intereses de investigación incluyen teoría y aplicación de red de Petri, redes neuronales, sistemas de información, minería de datos y modelación y simulación de sistemas.

Aurora Pérez-Rojas. Recibió el grado de Ingeniería Industrial por prte del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE) en 1971, el grado de Maestría en Ciencias en Sistemas Automáticos por parte del ISPJAE en 1978, y el grado de Doctorado en Ciencias Técnicas por parte del ISPJAE en 1987. Ha trabajado en diversas universidades entre las que se encuentran la ISPJAE, Cuba, UAS y UDO México, UNI Nicaragua, etc. Ha participado en tres proyectos internacionales de intercambio académico, ha dictado conferencias, publicado artículos ha sido ponente en mas de 20 eventos nacionales e internacionales. Desde el 2001 trabaja como profesor-Investigador en el Centro de Investigación en Tecnologías de Información y Sistemas, en el 2006 pasa al Centro de Investigación Avanzada en Ingeniería Industrial de la Universidad Autónoma del Estado de Hidalgo en Pachuca, Hidalgo, México. Sus intereses de investigación están relacionados con las Bases de Datos, Ingeniería de Software y Tecnologías de Información.

José Ramón Corona-Armenta. Recibió la licenciatura en Ingeniería Civil por parte del Instituto Tecnológico de Pachuca en 1993, la Maestría en Ingeniería en Investigación de Operaciones por parte de la Universidad Nacional Autónoma de México en 1996, y el Doctorado en Ingeniería en Sistemas Industriales por parte del Institut National Polytechnique de Lorraine, Francia en 2005. Desde 2005 se desempeña como Profesor Investigador del Centro de Investigación Avanzada en Ingeniería Industrial de la Universidad Autónoma del Estado de Hidalgo en Pachuca, Hidalgo, México.

Oscar Montaño-Arango. Se recibió como Ingeniero Metalúrgico en la ESIQIE del Instituto Politécnico Nacional, el grado de Maestro en Ingeniería en Planeación por parte de la Universidad Nacional Autónoma de México en el año 2000 y el grado de Doctor en Ingeniería en Sistemas de Planeación por parte de la Universidad Nacional Autónoma de México en el año 2007. Desde 2006 se desempeña como Profesor Investigador del Centro de Investigación Avanzada en Ingeniería Industrial de la Universidad Autónoma del Estado de Hidalgo en Pachuca, Hidalgo, México.