XI Congreso Internacional de la
Academia de Ciencias Administrativas A.C. (ACACIA).

Aplicación de Bases de Datos Activas en un sistema bancario

Mesa del trabajo en el que se registra (elegir sólo una mesa).
Ingeniería y Gestión de Sistemas

Nombre del autor o coautores.
Joselito Medina Marín
Oscar Montaño Arango
José Ramón Corona Armenta

Instituciones de adscripción.
Centro de Investigación Avanzada en Ingeniería Industrial
Universidad Autónoma del Estado de Hidalgo

Dirección completa incluyendo, teléfonos, fax y
correo-electrónico de quien será el responsable.
Ciudad Universitaria, Carr. Pachuca-Tulancingo Km. 4.5
Unidad Central de Laboratorios, Planta Alta
C.P. 42084
jmedina061074@yahoo.com.mx

Lugar y fecha del evento.
ITESO, Universidad Jesuita de Guadalajara.
Periférico Sur, Manuel Gómez Morín 8585
CP. 45090, Tlaquepaque, Jalisco México
22 al 25 de mayo de 2007

Aplicación de Bases de Datos Activas en un sistema bancario

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). 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. Existen herramientas, tales como las redes de Petri, con las cuales puede llevarse a cabo la representación de reglas ECA. Una base de reglas ECA es considerada como un sistema basado en eventos y es posible representarla con una red de Petri extendida, así como los eventos que las disparan.

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.

Aplicación de Bases de Datos Activas en un sistema bancario

Introducción

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 [1]. En la figura 1 se muestra el esquema de un SBD.

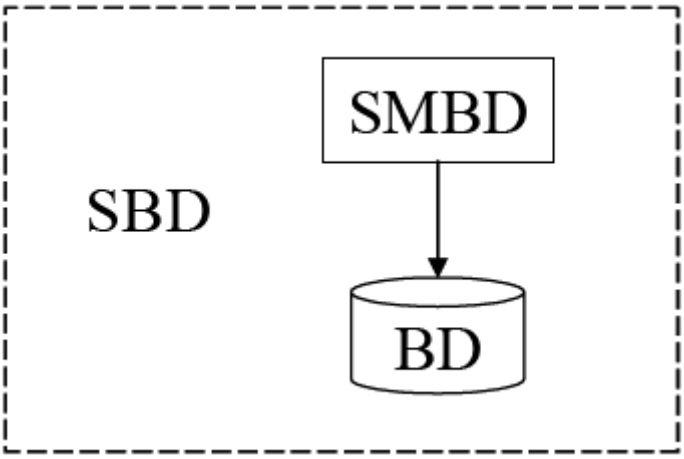

Figura 1. Esquema tradicional de un sistema de base de datos.

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 [2]. 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.

La estructura de la BD está definida por el esquema conceptual de la BD. Los esquemas son especificados utilizando un lenguaje de definición de datos (DDL) y el acceso a la BD es realizado a partir de un lenguaje de manipulación de datos (DML). La unión del DDL y del DML forman el modelo de datos. [1].

Los modelos de datos más ampliamente utilizados se clasifican en modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos, los cuales se describen a continuación:

Modelos lógicos basados en objetos

Los modelos lógicos basados en objetos son utilizados para describir datos en los niveles lógico y de visualización. Están caracterizados por el hecho de que proporcionan mucha flexibilidad en su estructura y permiten especificar restricciones de datos. Hay muchos modelos diferentes, entre los más conocidos se encuentran:

El modelo de entidad-relación. El modelo de datos de entidad-relación (E-R) está basado en una percepción del mundo real que consiste de una colección de objetos básicos, llamados entidades, y de relaciones entre estos objetos. Una entidad es una ”cosa” ó un ”objeto” en el mundo real que es distinguible de otros objetos. Una relación es una asociación entre varias entidades. El conjunto de todas las entidades del mismo tipo y el conjunto de todas las relaciones del mismo tipo forman un conjunto de entidades y un conjunto de relaciones, respectivamente.

El modelo orientado a objetos. Al igual que el modelo E-R, el modelo orientado a objetos está basado en una colección de objetos. Un objeto tiene valores almacenados en variables de instancia que se encuentran dentro de él. Además, contiene segmentos

de código que operan como parte del objeto, a estos segmentos de código se les llama métodos.

Los objetos que contienen el mismo tipo de valores y los mismo métodos están agrupados en clases. Una clase puede ser vista como la definición de un tipo de dato para objetos. La combinación de datos y métodos formando una definición de tipo de dato es semejante a un tipo de dato abstracto en lenguajes de programación. El único medio por el cual un objeto puede acceder a los datos de otro objeto es a través de la invocación de un método del otro objeto. Esta acción se le conoce como el envío de mensajes de un objeto a otro.

Modelos lógicos basados en registros

A los modelos basados en registros se les conoce así porque la base de datos está estructurada en registros de formato fijo de varios tipos. Cada tipo de registro define un número fijo de campos, ó atributos, y cada campo generalmente tiene una longitud fija. Los tres modelos de datos basados en registros más aceptados son el modelo relacional, el modelo de red y el modelo jerárquico.

Modelo relacional. El modelo relacional usa una colección de tablas para representar datos y las relaciones que existen entre ellos. Cada tabla tiene múltiples columnas, y cada columna tiene un nombre único.

Modelo de red. En el modelo de red, los datos son representados por colecciones de registros, y las relaciones entre los datos son representados por ligas, las cuales pueden considerarse como punteros. Los registros en la base de datos están organizados como colecciones de gráficas arbitrarias.

Modelo jerárquico. El modelo jerárquico es parecido al modelo de red en el sentido de que los datos y las relaciones entre ellos están representados por registros y ligas, respectivamente. Pero difiere del modelo de red en que los registros están organizados como colecciones de árboles, en lugar de gráficas arbitrarias.

Modelos físicos

Los modelos de datos físicos son usados para describir datos en el nivel más bajo. A diferencia de los modelos lógicos de datos, son pocos los modelos físicos de datos que se utilizan. Dos de los más conocidos son el modelo unificado y el modelo de estructura de memoria.

Sin embargo, la definición del modelo de datos con que se va diseñar un SBD así como el conjunto de programas generados para la manipulación de los datos no son suficientes para representar universos de discurso donde se requiere de una reacción automática por parte del SBD. Para tal efecto, se introdujeron los Sistemas de Bases de Datos Activas.

Estado del arte

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 [3], INFORMIX [4], ORACLE [5], Microsoft SQL Server [6], entre otros. Dentro de los sistemas no comerciales que proporcionan la definición de reglas activas podemos mencionar a Ariel, HiPAC, Startburst y POSTGRES[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. [10]

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 [8]. En la figura 1 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.

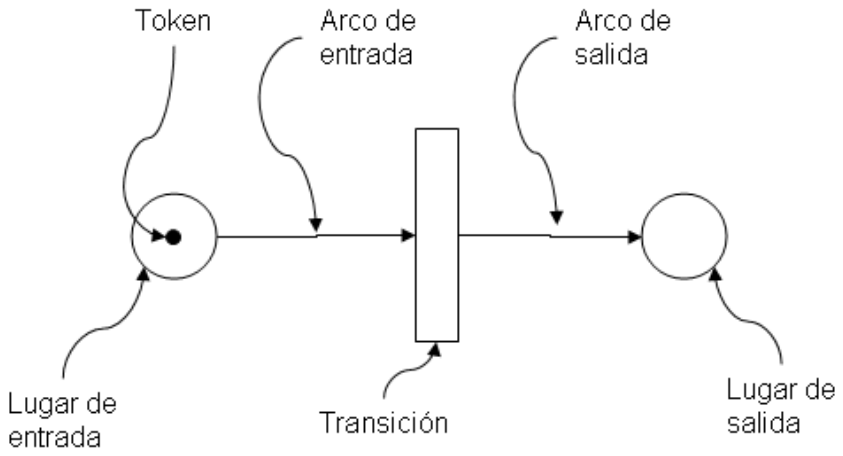

Figura 1. Elementos de una red de petri.

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 [8]. 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) [9].

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.

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 2 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.

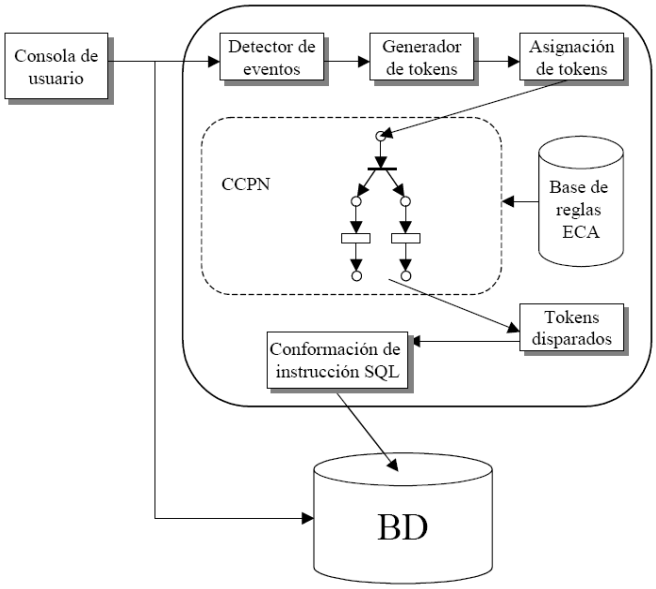

Figura 2. Arquitectura de funcionamiento de la interfaz ECAPNSim.

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 3 se muestra el ambiente con que trabaja ECAPNSim.

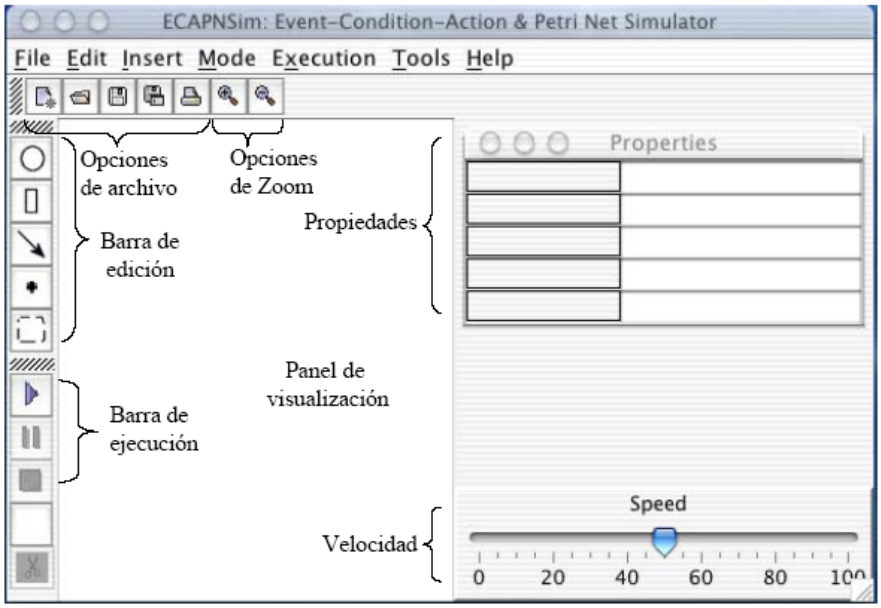

Figura 3. Ambiente de ejecución de ECAPNSim.

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). Almacena información acerca de los clientes del banco, donde se considera el número de cliente, el nombre, su domicilio, y la fecha cuando comenzó como cliente del banco.

CUENTA (número, número de cliente, saldo, límite, tipo {crédito, ahorro, nómina,cheques},estado (activada, bloqueada, cancelada)). Contiene datos sobre las cuentas que maneja el banco, tales como los números de cuenta; el número del cliente propietario de la cuenta; el saldo actual; el total de retiros realizados desde cajero automático; el tipo de la cuenta, la cualpuede ser una cuenta de crédito, una cuenta de ahorros, una cuenta de nómina o una cuenta de cheques; además, es necesario conocer si una cuenta está activada, bloqueda o cancelada.

TRANSACCIONES PENDIENTES (clave, tipo {depósito, venta}, cuenta origen, cuenta destino, cantidad, periodo). Esta relación se utiliza para almacenar aquellas transacciones que fueron realizadas fuera del horario normal de atención del banco.

DEPOSITO (fecha de depósito, número de cuenta, cantidad, lugar {sucursal, cajero automático, internet, teléfono}, tipo de moneda). Relación utilizada para almacenar todos los depósitos realizados en el banco, considerando la fecha y hora cuando el depósito fué realizado, el número de cuenta donde se abonó el depósito, el lugar en donde se llevó a cabo la operación y el tipo de moneda en que se realizó.

RETIRO (fecha de retiro, número de cuenta, cantidad, lugar {sucursal, cajero automático, internet, teléfono}, ubicación). Contiene atributos relacionados a transacciones de retiro de dinero de las cuentas del banco. Sus atributos son la fecha de la transacción, el número de cuenta donde se aplica el retiro, la cantidad de dinero retirado, el lugar donde la transacción fue hecha, y la ubicación, es decir, si la transacción fue realizada en México o en el extranjero.

PAGO POR SERVICIOS (fecha, cuenta origen, cuenta destino, modo de acceso, cantidad, concepto del pago, ubicación). Esta relación es utilizada para realizar pagos por servicios como el teléfono, internet, gas, entre otros. Los atributos para esta relación son la cuenta origen de donde se descontará el pago; la cuenta a donde se abonará el pago; el modo de acceso, es decir, la manera en que el cliente solicita este servicio, el cual puede ser realizado por internet o por teléfono; la cantidad de dinero que será pagada; el concepto del pago; y la ubicación de donde el cliente solicita el servicio.

IMPRESION DE COMPROBANTE (datos de la transacción). Esta relación se utiliza para generar los comprobantes sobre las transacciones que realiza un cliente, y solamente contiene el mensaje que será impreso en el comprobante. Los comprobantes son impresos cuando un cliente realiza un depósito, un retito, o un pago por servicio.

REPORTE (tipo de reporte, descripción, cliente, lugar, estado). El atributo tipo de reporte almacena información sobre el reporte que se sometió al banco, es decir, si la tarjeta (ya sea de débito o crédito) de un cliente fue robada, está dañada, o la extravió; o si el reporte trata sobre una queja o reclamación al banco. En el atributo descripción se describe la razón del reporte. Además, en un reporte debe considerarse el nombre del cliente que hace el reporte; el lugar donde el cliente hizo el reporte, el cual puede

ser sometido por internet, teléfono o en alguna sucursal del banco; y el estado del reporte, donde se especifica si el reporte no es válido o no procede, si está en espera para ser revisado, si está en proceso de revisión, o si ya ha sido resuelto.

SOLICITUD DE TARJETA DE CREDITO (número de tarjeta de crédito, cuenta, con fotografía, tarjeta de crédito adicional). Esta relación tiene como atributos el número de la tarjeta de crédito, el número de cuenta para esta tarjeta de crédito, si la tarjeta contendrá la fotografía del cliente, y si la tarjeta de crédito es una tarjeta 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. El conjunto de reglas contiene 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.

Regla 1: Cuando se realiza un retiro de alguna cuenta, si el saldo de la cuenta es mayor al monto solicitado para ser retirado, entonces se realiza la transacción y se sustrae la cantidad retirada del saldo de la cuenta.

Regla 2: Cuando un cliente realiza un pago por cierto servicio, si el saldo de la cuenta del cliente es suficiente para llevar a cabo el pago, entonces se retira la cantidad del pago de la cuenta del cliente.

Regla 3: Cuando un cliente realiza un depósito dentro del horario normal de atención al público, la cantidad por la que se hace el depósito se agrega al saldo de la cuenta del cliente.

Regla 4: Cuando un cliente realiza un depósito fuera del horario normal de atención al público, la transacción de pospone y será realizada al día siguiente.

Regla 5: Cuando un cliente realiza un pago por servicio, si está accesando por Internet, entonces el banco descuenta $25 de la cuenta del cliente como pago por el acceso al servicio de Internet.

Regla 6: El mismo caso que la regla anterior, pero ahora si el cliente realiza la transacción por teléfono el descuento es de $20.

Regla 7: Cuando el cliente realiza un pago por servicio, la cantidad del pago se descuenta de la cuenta del cliente.

Regla 8: Cuando un cliente hace un retiro en un cajero automático y si el cliente ya hizo más de cinco retiros de cajeros automáticos, el banco cobra al cliente $5 por cada retiro adicional que haga, descontándolos directamente de la cuenta del cliente.
Regla 9: Cuando un cliente realiza un depósito, y si el tipo de moneda es diferente del peso mexicano y el depósito se realiza dentro del horario de atención al público, se convierte el tipo de moneda y se agrega a la cuenta del cliente.
Regla 10: Similar a la regla anterior, pero si la transacción es hecha fuera del horario laboral, entonces la transacción es pospuesta para el siguiente día hábil.
Regla 11: Cuando un cliente somete una reclamación, pero la queja es rechazada por falta de elementos, el banco descuenta $200 por concepto de multa al cliente.
Regla 12: Cuando un cliente realiza un retiro en un cajero automático que no pertenece al banco donde el cliente tiene su cuenta, y si el retiro se realiza dentro del pais, la comisión cobrada por el banco propietario del cajero automático es de $10.
Regla 13: Similar a la regla anterior, pero en este caso si el retiro se realiza en el extranjero entonces la comisión que se cobra, por parte del banco propietario del cajero automático, es de $80.
Regla 14: Cuando una tarjeta de crédito es solicitada por un cliente, si el cliente requiere su tarjeta de crédito con su fotografía incluida, entonces el costo de la tarjeta de crédito se incrementa en $200.
Regla 15: Cuando una tarjeta de crédito es solicitada por un cliente, si el cliente requiere una tarjeta de crédito adicional, entonces el costo de la tarjeta de crédito se incrementa en $100.
Regla 16: Cuando se registra un nuevo número de cuenta en el banco, si se trata de una cuenta de crédito, el cliente debe pagar $350 por el servicio de crédito.

Para cada una de estas reglas, es necesario convertirlas a una forma de regla ECA, de acuerdo al modelo general para reglas ECA, quedando de la siguiente manera:
//Definicion de tablas
CLIENTE (numCliente integer, nombre CHAR(length), domicilio CHAR(length), fechaAlta date) ;

CUENTA (numero integer, numCliente integer, saldo float, limite integer, tipo CHAR(length), estado CHAR(length)) ;

PENDIENTES (clave integer, tipo CHAR(length), cuentaOrigen integer, cuentaDestino integer, cantidad float, periodo integer) ;

DEPOSITO (fecha date, cuenta integer, cantidad float, lugar CHAR(length), moneda CHAR(length)) ;

RETIRO (fecha date, cuenta integer, cantidad float, lugar CHAR(length) , ubicacion CHAR(length)) ;

PAGOPORSERVICIO (fecha date, cuentaOrigen integer, cuentaDestino integer, modoAcceso CHAR(length) , cantidad float, conceptoPago CHAR(length) , ubicacion CHAR(length)) ;

EXPIDECOMPROBANTE (descripcion CHAR(length)) ;

REPORTE (tipo CHAR(length) , descripcion CHAR(length) , cliente integer, lugar CHAR(length) ,estado CHAR(length)) ;

EXPEDICIONTC (numero integer, cuenta integer, conFotografia integer, adicional integer);

//Definicion de eventos

e0 : insert_DEPOSITO ;

e1 : insert_RETIRO ;

e2 : insert_PAGOPORSERVICIO ;

e3 : insert_EXPIDECOMPROBANTE ;

e4 : insert_CUENTA;

e5 : or(e0:e1:e2) ;

e6 : update_REPORTE_estado;

e7 : insert_EXPEDICIONTC;

//Definicion de reglas

rule 000,

on e5,

if true,

Then insert into EXPIDECOMPROBANTE values (’expide comprobante’) ;

rule 001,

on e1,
if CUENTA.saldo > insert.cantidad,
Then update CUENTA set value saldo = saldo - RETIRO.cantidad where numero = insert.cuenta;
rule 002,
on e2,
200 Caso de estudio
if CUENTA.saldo > insert.cantidad,
Then insert into RETIRO values (today, insert.cuentaOrigen, insert.cantidad, 'Sucursal', insert.ubicacion);
rule 003,
on e0,
if insert.fecha < '16:00:00',
Then update CUENTA set value saldo = old.saldo + insert.cantidad where numero = insert.cuenta;
rule 004,
on e0,
if DEPOSITO.fecha >= '16:00:00',
Then insert into PENDIENTES values (, 'deposito', 9999, insert.cuenta, insert.cantidad, 2) ;
rule 005,
on e2,
if insert.modoAcceso = 'Internet',
Then insert into RETIRO values (today, insert.cuentaOrigen, 25.00, 'INTERNET', insert.ubicacion) ;
rule 006,
on e2,
if insert.modoAcceso = 'Telefono',
Then insert into RETIRO values (today, CUENTA.numero, 20.00, 'TELEFONO', insert.ubicacion) ;
rule 007,

on e2,
if true,
Then insert into RETIRO values (today, insert.cuentaOrigen, insert.amount, 'SUCURSAL', insert.ubicacion) ;
rule 008,
on e1,
if insert.lugar = 'Cajero automatico' & CUENTA.limite > 10,
Then insert into PAGOPORSERVICIO values (Today , insert.cuenta, NULL, NULL, 5.00, 'SERVICIOS_VARIOS' , 'Nacional');
rule 009,
on e0,
if insert.moneda <> 'PESO' & insert.fecha < '16:00:00',
Then update CUENTA set cantidad = old.cantidad - insert.cantidad + insert.cantidad * tipomoneda where numero = insert.cuenta;
rule 010,
on e0,
if DEPOSITO.moneda <> 'PESO' & DEPOSITO.fecha >= '16:00:00',
Then insert into PENDIENTES values (1, 'deposito', null, insert.cuenta, insert.cantidad, 2));
rule 011,
on e6,
if REPORTE.tipo = 'Reclamacion' & update.estado = 'No procede',
then insert into PAGOPORSERVICIO values (today, NULL, CUENTA.numero, REPORTE.lugar, 200, 'Servicios Varios', 'Nacional');
rule 012,
on e1,
if insert.lugar = 'Cajero automatico externo' & RETIRO.ubicacion = 'Nacional',
Then insert into PAGOPORSERVICIO values (today, NULL, insert.cuenta, 'Automatico', 10, 'Servicios varios', 'Nacional');
rule 013,
on e1,

if insert.lugar = ’Cajero automatico externo’ & insert.ubicacion = ’Extranjero’,
Then insert into PAGOPORSERVICIO values (today, NULL, insert.cuenta, ’Automatico’, 80, ’Servicios varios’, ’Extranjero’);
rule 014,
on e7,
if insert.conFotografia = 1,
Then insert into PAGOPORSERVICIO values (today, NULL, insert.cuenta, ’Automatico’, 200, ’Servicios varios’, ’Nacional’);
rule 015,
on e7,
if insert.adicional = 1,
Then insert into PAGOPORSERVICIO values (today, NULL, insert.cuenta, ’Automatico’, 100, ’Servicios varios’, ’Nacional’);
rule 016,
on e4,
if 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 4. 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.

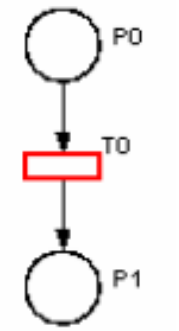

Figura 4.- CCPN para la regla 001.

La estructura de CCPN para la regla 000 se muestra en la figura 5. 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, respectívamente. 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.

La CCPN que representa a todo el conjunto de reglas ECA para el banco se muestra en la figura 6.

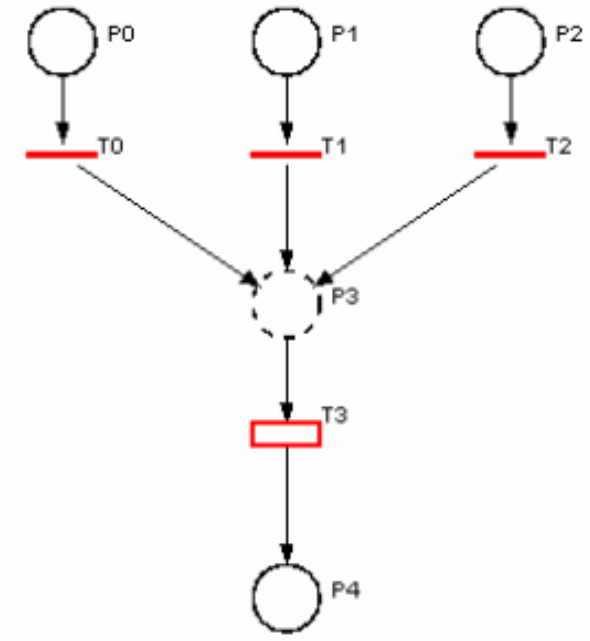

Figura 5.- CCPN para la regla 000.

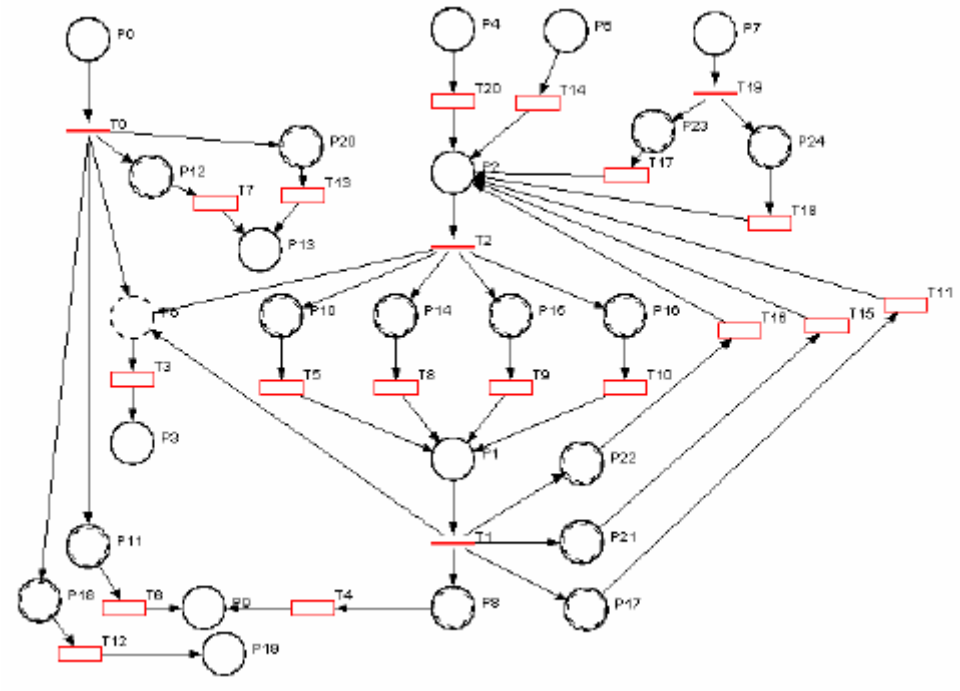

Figura 6.- CCPN para todo el conjunto de reglas ECA.

Las correspondencias entre el conjunto de reglas ECA y los elementos de la CCPN es como sigue:

Eventos:

Evento	Lugar	Copy Lugares
insert en depósito, retiro, or PagoPorServicio	p5	
insert en retiro	p1	p8, p17, p21, p22
insert en PagoPorServicio	p2	p10, p14, p15, p16
insert en depósito	p0	p11, p12, p18, p20
insert en expedición Tarjeta de Crédito	p7	p23, p24
insert en cuenta	p4	
update reporte.estado	p6	

Acciones:

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

Y finalmente, las *reglas ECA*:

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.

Conclusiones

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.

Las redes de Petri pueden ser utilizadas para definir reglas ECA, que modifiquen automáticamente el estado de una base de datos ante la ocurrencia de eventos que sean de interés alas políticas de un sistema. En este artículo se muestra una aplicación de un sistema de base de datos activa relacionado con un sistema bancario.

Referencias

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

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

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

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

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

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

(7) Hanson E.N.; (1996) “The Design and Implementación of the Ariel Active Database Rule System”, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 1.

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

(9) 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.

(10) Dittrich K., Gatziu S., Geppert A., ”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, 1995.