Rev. Téc. Ing. Univ. Zulia. Vol. 36, Nº 3, 253 - 261, 2013

Automatic insertion of business rules in databases

Martha Beatriz Boggiano-Castillo, Alain Pereira Toledo, Alain Pérez-Alonso, Luisa-Manuela González-González, Ramiro Pérez-Vázquez

Universidad Central "Marta Abreu" de Las Villas. Centro de Estudios Informáticos. Carretera de Camajuaní Km 5 ½. Santa Clara, Villa Clara. Cuba. CP: 54830. Teléfono: 281515. mbeatriz@uclv.edu.cu

Abstract

One of the actual trends in the development of the information systems is the leading through the focus of the business rules. The business rules dealt with this work are the ones known as constraint rules and let to express constraints of integrity about data in a database. All these can involve several charts of a database and in order to create a rule, the more important item is to automatically find out which chart must be applied and which resources must be used to. In this article a method sort is proposed to sort out the problem related with the Constraint Rules when the database has been created. The examples used refer to just a part of an application to the renal transplantation.

Keyword: restriction business rules, business database, automatic implementation.

Inserción automática de reglas de negocio en bases de datos

Resumen

Una de las tendencias actuales en el desarrollo de los sistemas de información es su conducción mediante el enfoque de reglas de negocio. Las reglas de negocio que son tratadas en este trabajo son aquellas conocidas como reglas de restricción y permiten expresar restricciones de integridad sobre datos en una base de datos. Este tipo de regla puede ser implementada más cerca de los datos. Ellas pueden involucrar a varias tablas de una base de datos, y para generar la regla, lo más importante es descubrir automáticamente a qué tabla debe ser aplicada, y con qué recursos se puede aplicar. En este artículo se propone un método para resolver esta problemática para las reglas de restricción cuando la base de datos ya ha sido creada. Los ejemplos utilizados se refieren a parte de una aplicación para el control del trasplante renal.

Palabras clave: reglas de negocio, reglas de restricción, bases de datos de negocios, generación automática de reglas.

Introducción

La consideración del enfoque de reglas de negocio en el diseño e implementación de sistemas de información (SI) supone una relativa independencia entre los desarrolladores del SI y de las reglas de negocio.

Hay reglas de negocio que pueden ser implementadas en las bases de datos con mecanismos más potentes que las opciones *check, foreignkey, primarykey*, estos son los disparadores(*triggers*), funciones, procedimientos almacenados, vistas, que también son suministrados por la mayoría de los Sistemas de Gestión de Bases de Datos Relacionales (SGBDR), como fragmentos de código en lenguaje SQL. En este artículo se valora el uso de los disparadores como recursos para acercar ambos desarrollos e implementar las reglas de negocio en SI. El problema esencial radica en determinar cuáles reglas de

negocio pueden ser implementadas en la base de datos y crear mecanismos que logren esta implementación a la manera de generación automática para independizar las reglas de negocio del desarrollo de las aplicaciones.

Este trabajo se basa en la clasificación de reglas de negocio realizada por Morgan [1], se seleccionan las reglas de tipo restricción, se tratan las entidades del negocio como tablas de la base de datos y se muestra un tipo de mecanismo que puede generar de forma automática las implementaciones de estas reglas en la base de datos ya creada. Los mecanismos de los ejemplos pertenecen al lenguaje transact SQL, compatible con Microsoft SQL Server versión 2012.

Generalidades sobre las reglas de negocio

Según Ross [2] una "regla de negocio es una regla que está bajo la jurisdicción o gobierno del negocio". Una regla de negocio es una sentencia que define o restringe algunos aspectos del negocio; tiene la finalidad de establecer la estructura del mismo, controlar o influenciar su comportamiento [1, 3, 4]. En [4] se precisaque *"...está relacionado con los hechos que son grabados como datos y las restricciones sobre los cambios a los valores de tales hecho...."*.

Formas de expresar las reglas de negocio

Algunos autores [1, 5] coinciden en que, independientemente de la clasificación de la regla de negocio: restricción, cómputo, enumeración, clasificación, etc.; las reglas de negocio se expresan de diferentes maneras, o con diferentes niveles de abstracción, y que su expresión comienza por el lenguaje natural. Según [5] son cuatro las formas de expresarlas, cada una para una audiencia diferente: conversación informal del negocio, versión en lenguaje natural, versión en lenguaje de especificación y versión en lenguaje de implementación de reglas. En [1] se distinguen sólo tres formas de expresión de reglas de negocio: informal, técnico y formal.

Se observan algunas similitudes entre los dos criterios expuestos; aquí se trabaja con los expuestos por Morgan porque es más preciso en su descripción y su nivel informal se sustituye por "*cercano al natural*", como un patrón que puede ser entendido por todos; el nivel técnico que presenta especificaciones de funciones, símbolos matemáticos, especifica la regla siguiendo el patrón de regla, pero aún no puede ser ejecutado por la computadora. El nivel formal se refiere a su implementación en un lenguaje de programación para ser ejecutado.

Reglas de restricción

Morgan en [1] propone categorías de reglas de negocio, las denomina como: restricción básica, lista de restricciones, clasificación, computación y enumeración. Para cada categoría de regla se propone un patrón de regla como una manera conveniente de expresar estas.

Las reglas de negocio de restricción están dirigidas a especificar qué restricciones o condiciones deben cumplirse para que un dato se considere válido. Estas reglas, pueden tener una implementación en la base de datos. De lo que se trata, entonces, es de encontrar un mecanismo que pueda insertar automáticamente la implementación correspondiente.

Lenguajes para especificar restricciones

La especificación de restricciones sobre los datos es un tema de interés en las bases de datos, y también en el diseño de clases.

El lenguaje OCL (Object Constraint Language), facilidad de UML (Unified Model Language) ha sido definido como un estándar por la OMG (Object Management Group) [6] para la representación de restricciones en los diagramas UML, útil para promulgar restricciones sobre objetos del negocio [7]. Ya existen varias herramientas que utilizan OCL para la representación de restricciones [8, 11], OCL Toolkit [10, 12, 13]. Sin embargo, Morgan en [1] no le da un peso especial al uso del OCL.

Algunos lenguajes utilizados en la descripción de reglas de negocio se basan en XML y responden a la necesidad de separar los datos y la estructura del modelo del negocio, de las reglas que los rigen, así pueden ser guardadas aparte de los datos. Cada sintaxis descrita por un esquema XML para tales lenguajes, implica que, si se cumplen ciertas condiciones se produce un evento en el sistema [14]. Esto es básicamente lo que en el

campo de las bases de datos activas se ha dado en llamar como *ECA* (Event Condition Action) *rules* [15]; o sea, eventos condición-acción.

En [16] se ha definido un lenguaje para especificar restricciones, nombrado en dicho trabajo lenguaje técnico de patrones (LPT). El mismo se basa en dos elementos fundamentales: la descripción de las reglas de negocio de tipo restricción a partir de un patrón y de la notación punto. Además considera que los términos de las reglas son elementos de la base de datos, tablas y atributos.

En este trabajo se utiliza un patrón, inspirado en la forma del patrón de restricción básico de Morgan [16]: <determinante><sujeto> (no puede tener <características>) |

(puede tener <características> sólo si <hechos>).

En la Tabla 1 aparecen los significados para los elementos que intervienen en el patrón, las entidades reconocibles del negocio para Morgan se sustituyen por elementos de la base de datos: tablas y atributos.

Por otro lado la notación punto es utilizada para acceder a los atributos de las tablas y navegar entre estas de manera que se pueda lograr: acceso simple a un atributo particular de una tabla, Tabla_1.Atributo y navegar entre entidades, Tabla_1.Tabla_2. ... .Tabla_n. [Atributo].

Para esto es necesario que entre las tablas se pueda establecer un acople natural.

Por ejemplo, si se desea acceder al nombre del paciente se puede hacer con *Paciente.Nombre*. También es posible navegar entre tablas, por ejemplo con *Cirujano.DonantesPotenciales.Evolución* se expresan las evoluciones de los donantes vivos atendidos por un cirujano; esto resulta una colección de datos y como puede observarse, el orden es importante. Véase Figura 1.

En LPT se considera al sujeto como inicio de la navegación; se ha incluido un grupo de operadores para manipular colecciones de datos, como se muestra en la Tabla 2.

Por ejemplo la regla "*Un Paciente no puede tener Donantes Potenciales con más de 3 Evoluciones*". Con el uso los operadores descritos del LPT se puede indicar:

Tabla 1. Especificaciones del patrón de reglas

Elemento	Significado
<determinante>	Es el determinante para cada sujeto, por ejemplo: Una, Uno, El, La, Cada, Todos. Según el mejor sentido en la redacción.
<sujeto>	Es un elemento de la base de datos del negocio, tal como una tabla. La tabla puede ser cualificada por otros atributos descriptores.
<características>	Describe las características del sujeto en el negocio, tanto internas como relacionadas con otras entidades.
<hechos>	Hechos relativos al estado o comportamiento de la base de Datos del negocio, incluyendo o no al sujeto.

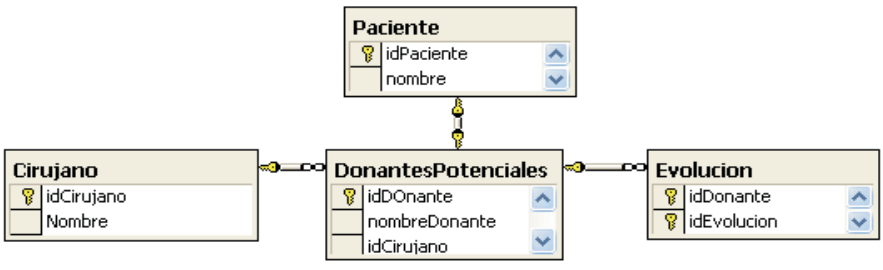

Figura 1. Parte de la base de datos para Trasplante Renal [16].

Tabla 2. Operaciones de conjuntos permitidas para elementos múltiples

Operador	Utilización
Sizeof <collection>	Define cuántos elementos contiene la colección de elementos.
empty <collection>	Retorna verdadero si la colección no contiene elementos.
Exists (<element>, <collection>)	Retorna verdadero si el elemento especificado existe al menos una vez en la colección.
max	Retorna el elemento máximo de la colección.
min	Retorna el elemento mínimo de la colección

Un Paciente no puede tener sizeof(DonantesPotenciales.Evolucion.idEvolucion) > 3

Donde: <sujeto>: Paciente,

<características>: sizeof(DonantesPotenciales.Evolucion.idEvolucion) > 3

Al escribir Donantes Potenciales.Evolucion.id Evolucion se está suponiendo que el sujeto es quien comienza la cadena, por lo tanto es equivalente a decir Paciente. Donantes Potenciales.Evolucion.id Evolucion.

Resultados y discusión

En este trabajo las reglas del negocio de tipo restricción se especifican usando el LPT, la forma de nombrar las tablas y los atributos de la base de datos coinciden con los términos utilizados en el modelo de hechos relacionado con el negocio. Esto no es un problema porque, como se plantea en Martínez del Busto [17] las reglas de negocio se construyen tomando como base un vocabulario que está formado por las palabras y frases establecidas por la comunidad de usuarios de un negocio específico. La modelación de las mismas supone un uso controlado y previo de dicho vocabulario.

Para la implementación de las reglas de negocio se considera el lenguaje SQL y en particular los recursos que este brinda, usando el transact SQL. Se utilizan los disparadores y funciones de usuarios para implementar dichas reglas.

De acuerdo al patrón especificado antes, una regla puede ser violada sólo en la cláusula que corresponde a <características> que es el centro de interés porque permite identificar las tablas de la base de datos del negocio donde se deben chequear las reglas correspondientes.

Una regla de negocio tipo restricción, se escribe según el patrón descrito en lenguaje natural y formalizado en LPT, usando caminos de navegación, para generar un disparador y una función.

El disparador será generado en tabla(s) de las <características> donde puede ocurrir la violación de la regla. La función se genera de tal manera que se produce el valor lógico verdadero cuando las características que indica la regla se incumplen. Por ejemplo, para la regla vista anteriormente:

<sujeto>: Paciente; <características>: sizeof(DonantesPotenciales.Evolucion.IdEvolucion) > 3,

La especificación de las <características> se puede violar únicamente al insertar o modificar una nueva Evolución para un Donante Potencial, por lo tanto se genera el siguiente disparador en la Tabla 3.

En este disparador se declara @subject que es el <sujeto> condicionado de la regla (Paciente), el que se extrae a través de una consulta, donde el acople de las tablas se genera a partir de la expresión que brinda la notación punto. Después se cuestiona si la regla es quebrantada o no para la instancia del sujeto que se inserta o modifica, en caso afirmativo se toman las medidas correspondientes. Este cuestionamiento se hace a través de una función (FRN#3); encargada de implementar el cuerpo de la regla; se verifica si <sujeto> viola las condiciones impuestas por la restricción mediante un valor lógico, en caso que se violen, devuelve verdadero. La función que se genera, en este caso, puede verse en la Tabla 4.

Esta forma de implementar las reglas de negocio es válida siempre que se tengan relaciones 1:m entre las tablas que están asociadas a las en-

Tabla 3

```
CREATE Trigger TRN#30 ON Evolution
AFTER INSERT,UPDATE
AS
DECLARE @Subject int;
SET @Subject = (SELECT a.idPaciente
       FROM Pacientea, DonantesPotencialesb,Inserted c
    WHERE (a.idPaciente=b.idPaciente) AND
         (b.idDonantesPotenciales=c.idDonantesPotenciales));
IF dbo.FRN#3(@Subject) = 1
  BEGIN
    RAISERROR('RN#3',16,1);
ROLLBACK TRANSACTION;
END
```

Tabla 4

```
CREATE FUNCTION FRN#3(@id Sujeto Integer)
Returns BIT AS
BEGIN
  DECLARE @result BIT ;
  SET @result = 0 ;
  if (SELECT COUNT(c.idEvolucion)
     FROM Paciente a, DonantesPotencialesb, Evolucion c
     WHERE(@idSujeto=a.idPaciente)and (a.idPaciente=b.idPaciente)AND
  (b.idDonantesPotenciales=c.idDonantesPotenciales) )>3 ;
   set @result = 1 ;
  return @result;
end
```

tidades del negocio, que se expresan con la notación punto, a esta secuencia de tablas separadas unas de otras por un punto se le llamará de ahora en adelante *camino de navegación explícito*. En el ejemplo anterior, el camino sería Paciente.Donantes Potenciales. Evolucion.id Evolucion.

Sin embargo en la base de datos pueden existir tablas que no correspondan con las entidades fundamentales del problema. El caso más notorio es cuando existen, en el diagrama conceptual correspondiente, interrelaciones muchos - muchos entre las entidades; en estos casos pueden aparecer tablas auxiliares en el diseño lógico. No es común que en la escritura de las reglas se haga referencia explícita a estas tablas porque ellas son propias del diseño lógico y no conceptual. En estos casos es más complicado determinar dónde es necesario generar el disparador y cómo implementarlo.

La notación punto no especifica el tipo de interrelación entre las tablas que se relacionan. Así, si Tabla1.Tabla2. ... .Tablan.[Atributo], conforma el camino de navegación explícito de las <características>; existe la incertidumbre sobre en cuál tabla deberá almacenarse la regla en forma de disparador: la situación está en ¿cómo se descubre esta tabla para luego posicionar el disparador en el lugar correcto?

Supóngase la declaración de una regla cuya característica se expresa con el camino de navegación: *...E_i.E_j... Atributo*. Pero las tablas E_i y E_j no acoplan naturalmente, sino a través de la tabla auxiliar, E_{res}, que no está especificada en el camino de navegación explícito de las <*características*>. El problema, entonces, es ¿cómo encontrar esta tabla? La respuesta viene dada por encontrar un conjunto de tablas que, por medio de las llaves foráneas, apuntan a las tablas E_i o E_j. Así,

para averiguar por las tablas intermedias solo es necesario contar con:

1. Las tablas principales que componen el <sujeto> y el camino de navegación explícito de las <características> de una regla, estas tablas se corresponden con los términos principales del negocio con los que es enunciada la regla.

2. Un método de búsqueda a través del esquema relacional que dada una tabla sea capaz de devolver todas las tablas que la referencien.

Se propone representar el esquema relacional como un grafo dirigido, en el cual los vértices constituyen las tablas y las aristas vienen dadas por las interrelaciones que se establecen entre las tablas, de manera que un vértice E_{res}se conecta a un vértice E_j adyacente si en E_{res}está presente una llave foránea que referencia a E_j. A las tablas que contienen más de 2 arcos de salidase les llama en esta investigación *tablas de resolución*. En este caso E_{res}es tabla de resolución. Ver Figura 2.

En esta investigación se trabaja con un modo de búsqueda, que dado un vértice, obtiene todos aquellos vértices que lo apuntan, para esto se utiliza la información que se encuentra en el catálogo de la base de datos.

Cada proveedor de SGBDR ha implementado el catálogo, de las bases de datos, según le ha parecido conveniente [18] por lo que es necesario acceder a los objetos de la base de datos, con características particulares en cada gestor. En este trabajo se utiliza el SGBDR SQL Server de Microsoft, el cual brinda en su documentación la forma de acceder al catálogo, de manera que se puede recuperar toda la información necesaria.

Como resultado, se obtiene un conjunto de pasos que permite la generación automática de la implementación de cadaregla de negocio:

1. Distinguir los términos del negocio y las *tablas de resolución*.

2. Reconstruir la regla por medio de la inserción de las tablas de resolución.

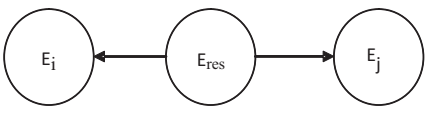

Figura 2. Grafo dirigido que representa la estructura de un diagrama relacional.

3. Determinar en cuál tabla insertar el disparador.

El problema de la posible existencia de tablas de resolución en el camino de navegación en una regla obliga a la formulación de dos preguntas:

- ¿Cómo se distingue la existencia de tablas de resolución entre, al menos, dos de los términos constituyentes de una regla?
- ¿Dónde ocurre el evento desencadenador de la violación de la regla?

Identificación de la tabla de resolución.

Para la identificación de la(s) *tabla(s) de resolución*, se chequea cada regla de negocio. Por cada regla se tiene una lista ordenada de sus términos, que comienza en el sujeto y continúa con los términos del camino de navegación explícito de la cláusula <características>. Se escogen dos a dos los términos consecutivos del camino de navegación explícito, y se busca la existencia de una tabla de resolución. Una *tabla de resolución* tiene determinadas propiedades:

- Las llaves foráneas son las llaves primarias de las tablas que corresponden a términos del negocio que intervienen en la regla.
- Las llaves foráneas en cuestión están contenidas en la llave primaria de la tabla de resolución.

En base a estas propiedades se buscan la(s) *tabla(s) de resolución*.

Reconstrucción de la regla.

Con la tabla o tablas de resolución correspondientes y las tablas del camino de navegación explícitose construye el camino de navegación con todas las tablas implicadas, llamado camino de navegación completo.

En el proceso de transformación de la regla a una expresión válida SQL para la base de datos relacional, se sustituye el camino de navegación completo, por un acople entre las tablas homólogas a los términos de la regla, contando con la información contenida en las tablas de resolución. A este proceso se le llama reconstrucción de la regla.

La inserción de la tabla de resolución en el camino de navegación puede verse también como una *reducción* del problema al transformar *todos los caminos de navegación explícitos a caminos de*

navegación completos, incluyendo las tablas de resolución. Nótese que una interrelación m:m(muchos-muchos) en un esquema conceptual es transformada a una secuencia de interrelaciones [1:m, 1:m, ...].

Construcción de la lista de eventos

Luego de reconstruir la regla, para responder: ¿a dónde ocurre el evento desencadenador de la violación?, se pudiera considerar, a priori, que el evento siempre ocurre en la tabla donde se realiza alguna de las operaciones sobre base de datos: insertar, actualizar y eliminar. Sin embargo, esta respuesta no es totalmente correcta.

Supóngase la regla: *E1 no puede tener E2.atributo > 5.*

Y supóngase que los términos constituyentes *E* y *E* se interrelacionan a través de una tabla de resolución, que no se conoce (similar a la Figura 2, donde E_i es *E1* y E_j es *E2*); al insertar una nueva fila en la Tabla *E2*, en la cual el atributo toma un valor mayor que 5, aparentemente se realiza una violación de la regla, pero no es así, porque entonces la regla se podría expresar como *E2 no puede tener E2.atributo > 5*, que es una regla más sencilla que la presentada anteriormente.

Hay que tener presente que violación es chequeada con respecto al camino de navegación entre todas las tablas constituyentes de la regla, que se le ha llamado camino de *navegación completo.*

En la Tabla E2 se pueden insertar artículos que no tengan relación con la tabla E1 (ver Figura 3). Sin embargo al insertar una fila en la Tabla E_{12}, tiene que tener un artículo relacionado en la Tabla E1 y un artículo relacionado en la Tabla E2,con atributo >5.

Por tanto, la violación nunca se originará en *E2*, como se puede pensar con premura, sino en la Tabla E_{12}, es decir, en la tabla de resolución. Pero no siempre ocurre del mismo modo. Si así fuera, ¿qué quedaría para aquellas reglas que no poseen interrelaciones con tablas auxiliares involucradas, es decir cuando el camino de navegación completo se corresponde con el camino de navegación explícito?

El evento no depende exactamente de una *tabla de resolución*, sino de una característica de este tipo de tablas. La tabla de resolución representa un vértice del cual "salen" aristas que lo conectan con los vértices adyacentes. A las tablas que solamente tienen aristas salientes se convierten en candidatas para lanzar el evento, y se les llama *tabla común* (ver Figura 4).

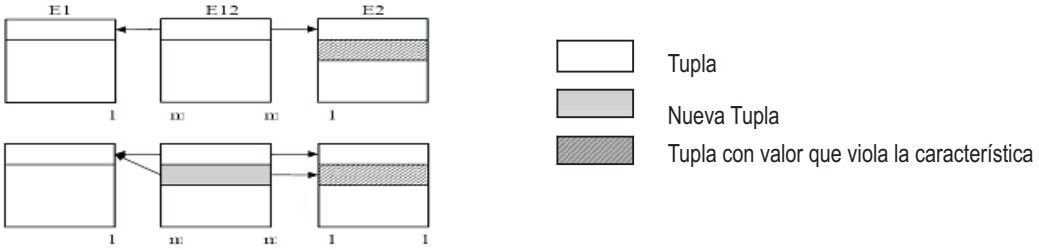

Figura 3. Secuencia de inserción en una interrelación m:m

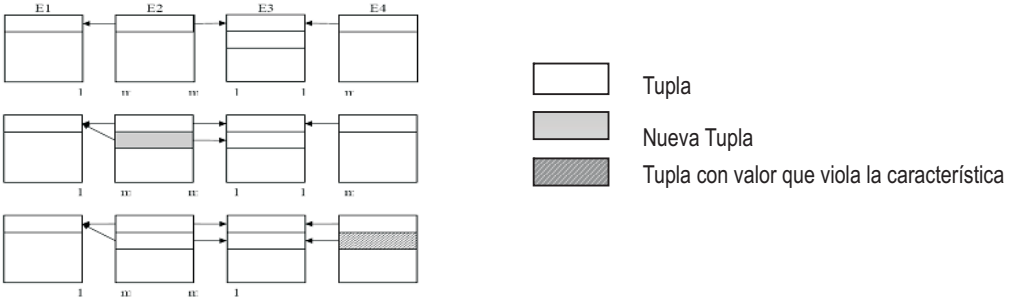

Figura 4. Secuencia de inserción y existencia de tablas de resolución.

Se tienen dos tablas comunes, la E_2 y la E_4, pues $gr_+(E_2) = 0$ y $gr_+(E_4) = 0$; donde gr_+ se refiere a cantidad de aristas entrantes.

En base a esto es posible construir una lista de eventos, denominada LE (lista de eventos).

Una LE es una lista donde, para cada regla se relaciona el *evento de violación* con la tabla donde ocurre el evento. Una estructura sencilla que materializa la LE, es una lista de variables binarias. El orden de la lista corresponde al orden de las tablas del camino si se parte del sujeto. Entonces se presenta la LE, como *Tablas{V1, V2, …, Vn}, con Vi ∈{0, 1}*.

Para el caso de la Figura 4 la LE quedaría como sigue: *Tablas{0, 1, 0, 1}*.

Luego de conformada la LE, ya se está listo para generar de manera automática el chequeo de restricciones en los lugares adecuados a través de los *disparadores*; en el ejemplo anterior se generarán los disparadores en E_2 y E_4.

Conclusiones

Este artículo muestra el conocimiento esencial para determinar cuáles tablas de una base de datos deben ser chequeadas para el cumplimiento de una regla de restricción, y así generar automáticamente las implementaciones de las reglas y el modo de hacerlas cumplir.

Se obtiene un conjunto de pasos para resolver el problema de dónde chequear las reglas, estos son:

1. Distinguir los términos del negocio y las tablas de resolución.

2. Reconstruir la regla por medio de la inserción de las tablas de resolución.

3. Determinar en cuál tabla insertar el disparador.

Con el paso tres (3) se construye la lista de eventos, que permitirá generar un disparador en cada tabla cuyo valor correspondiente en la lista de eventos es uno (1). Con este disparador se chequea el cumplimiento de la regla.

Se presupone la correspondencia entre los nombres de las tablas y atributos con los términos de las entidades principales del negocio y sus características; para la tablas que se forman de las interrelaciones muchos- muchos del modelo conceptual no hay convenio de nombre, si estas no corresponden a entidades reconocibles del negocio.

Referencias bibliográficas

1. Morgan, T.: “Business Rules and Information Systems: Aligning IT with Business Goals”, Addison Wesley, Indianapolis, USA, 2002.
2. Ross, R.G.: "What Is a Business Rule?". Business Rules Journal, Vol. 11 No. 3 (2010) 1-4. Disponible en: http://www.BRCommunity.com/a2010/b525.html
3. Daum, B. and U. Merten: “System Architecture with XML, 1st Edition”, Morgan Kaufmann San Francisco, USA, 2002.
4. Hay, D. and K.A. Healy: "Defining businessrules- what are they really? Technical Report 1.3" (2000). Disponible en: http://businessrulesgroup.org/first_paper/br01c0.htm
5. Von Halle, B. and G. Ronald: “Business rules applied: building better systems using the business rules approach”, John Wiley & Sons, New York, 2002.
6. OMG: "Object Constraint Language". Object Management Group, Inc. (2010). Disponible en: http://www.omg.org/spec/OCL/2.2/PDF/.
7. Ivanov, A.N.: "Graphic Language for Describing Constraints on Diagrams of UML Classes". Programming and Computer Software, Vol. 30, No. 4 (2004) 204-208.
8. Santos, R. and H. Invernizzi: "OCL – Estado del Arte. “Facultad de Ciencias y Tecnología, Universidad Nova de Lisboa (2004). Disponible en: http://pwp.netcabo.pt/rmss/ficheiros/textos/OCL%20-%20Paper%20Draft.pdf.
9. Zimbrão, G. *et al.*: "Enforcement of Business Rules in Relational Databases Using Constraints" (2002). Disponible en: http://www.mii.lt/informatica/pdf/INFO764.pdf
10. Demuth, B.: “The Dresden OCL Toolkit and its Role in Information Systems Development”. Memorias de: 13th International Conference on Information Systems Devel-

opment: Methods and Tools, Theory and Practice, Vilnius, Lithuania, 2004. Disponible en: svn-st.inf.tu-dresden.de/svn/dresdenocl/branches/legacy/www/downloads/pdfs/BirgitDemuth_The DresdenOCL Toolkit.pdf

11. Tedjasukmana, V.N., *et al.*: "Translation of OCL Invariants into SQL: 99 Integrity Constraints". Tutor: N. refiere. Tesis de Maestría. Technical University of Hamburg, Alemania, 2006.
12. 12. Heidenreich, F., C. Wende, and B. Demuth: "A Framework for Generating Query Language Code from OCL Invariants". Vol. 9, No. (2008) 4-10. Disponible en: http://journal.ub.tu-berlin.de/eceasst/article/download/108/103.
13. Konermann, A.: "The Parser Subsystem of the Dresden OCL2 Toolkit. Design and Implementation".GNU Free Documentation License (2005). Disponible en: http://svn-st.inf.tu-dresden.de/svn/dresdenocl/branches/legacy/www/papers/ParserDesign.pdf.
14. Martínez Hernández, J.L.: "Introduciendo semántica en un proceso de desarrollo software a través de reglas de negocio". Tutor: P.M.F. José Carlos González Cristóbal. Tesis de Doctoral Escuela Técnica Superior de Ingenieros de Telecomunicación Universidad Politécnica de Madrid, Madrid, 2010.
15. Paton, N.W. and O. Díaz:"Active Database Systems". ACM Computing Surveys, Vol. 31 No. 1 (1999) 63-103. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.8207&rep=rep1&type=pdf
16. Pérez Alonso, A.: "Aplicación para reglas de restricción en negocios". Tutor: M.B. Boggiano-Castillo, MSc. and R. Pérez- Vásquez, Dr. Tesis de Licenciatura. Departamento de Bases de Datos, Universidad Central "Marta Abreu" de Las Villas, Santa Clara, 2008.
17. Martínez del Busto, M.E., *et al.*: "Business vocabulary of kidney transplant with ontological approach for a generic fact model". Revista Facultad de Ingenieríaa Universidad de Antioquia, Vol. No. 53 (2010) 155-162. Disponible en: //www.scielo. org.co/scielo.php?script=sci_arttext&pid=S0120-62302010000300014.
18. Kriegel, A. and B.M. Trukhnov: "SQL Bible", Wiley Publishing, Inc., Indianapolis, Indiana, 2003.

Recibido el 26 de Noviembre de 2012
En forma revisada el 16 de Septiembre de 2013