Universidad Central “Marta Abreu” de Las Villas.
Facultad Matemática, Física y Computación
Licenciatura en Ciencia de la Computación

Tesis en opción del grado de Master en Ciencias de la Computación

“Reglas de Negocio en Bases de Datos Relacionales”

AUTOR: Lic. Alain Pérez Alonso.

TUTORES: MSc. Martha Beatriz Boggiano Castillo.
Dr. Ramiro Pérez Vázquez.

CONSULTANTE: Dra. Luisa González González.

"Año 52 de la Revolución"
19 Julio del 2010

Dedicatoria.

A mis abuelos,

A mi mamá y papá,

A mi futura esposa,

A mi tío y tías,

A mi hermano,

A mis sobrinos,

A mis primos y primas,

Y a todos mis amigos.

Agradecimientos.

A mis tutores Martha Beatriz Boggiano Castillo y
Ramiro Pérez Vázquez por su asesoría en este trabajo.
A mi profesora consultante Dra. Luisa González González.
Al Dr. Carlos García por compartir sus conocimientos.
A todo el grupo de Base de Datos.
A todos los que me formaron y enseñaron.
A todos los que de una u otra forma me han ayudado y
contribuido a la terminación de esta tesis.

Pensamiento.

Y dijo Dios a Salomón : Por cuanto hubo esto en tu corazón, y no pediste riquezas, bienes o gloria, ni la vida de los que te quieren mal, ni pediste muchos días, sino que has pedido para ti sabiduría y ciencia … sabiduría y ciencia te son dadas; y también te daré riquezas, bienes y gloria como nunca tuvieron los reyes que han sido antes de ti, ni tendrán los que vengan después de ti.

CRÓNICAS 1. 11,12

Resumen.

Múltiples autores sugieren considerar como factor de esencial importancia la forma en que se manejan los requisitos, enfocados actualmente como reglas de negocio. Estas por tanto han de tener un tratamiento especial desde tempranas fases de diseño e implementación en los sistemas de información.

Las bases de datos de estos sistemas son un lugar ideal para implementar dichos requerimientos debido a su cercanía con los datos del negocio. Por ello el presente trabajo fija su objetivo en obtener los beneficios de esta filosofía entre los que se pueden citar: agilidad para responder a requisitos dinámicos, transparencia en las auditorías que se lleven a cabo y la reducción de costos en crear o actualizar partes del sistema que implementan políticas del negocio.

Como resultado se obtiene la definición de cuatro patrones junto a un lenguaje que le brinda a las reglas un elevado nivel técnico, de este lenguaje se establecen semejanzas con uno reconocido mundialmente. Ambos son creados bajo la premisa de no alejarse de un lenguaje natural, pues este último es el inicio de toda regla. Se establecen además métodos de manipulación dentro del sistema para que las reglas cumplan su función correctamente.

Para lograr implementar reglas de negocio se recurren a numerosos recursos de bases de datos como funciones, vistas, tablas, transacciones, triggers y procedimientos los cuales son especificados siguiendo un estándar para los gestores de bases de datos.

Abstract.

A great assortment of authors suggest considering, as a paramount element, the ways in which the different requirement are managed. These requisites are currently acknowledged as stable business rules. Thus, these ways must be specially treated since the very first phases of the design, as well as since the very first phases of its implementation on the information systems.

These systems databases are an ideal place to foster the already mentioned requirement due to its very close relationship with the business' data. That is why this research's purpose is to obtain benefits from this philosophy. Some of these benefits are: quickness and skillfulness to respond to a dynamic requisite, openness and transparency on the diverse audits to come and costs reduction when creating or upgrading system's parts which advance business policies.

As a result, a definition of four patterns, along with a language which gives a high technical level to the rules was obtained. This language has similarities to one which is known worldwide. These were created with the principle of being as close to natural language as possible, for the last one is the foundation of every rule. Some contained by the system management methods are also established for the rules to accomplish its right function.

To advance business rules countless database resources are used. These resources can be functions, views, tables, transactions, triggers and several procedures which are specified according to a database management standard.

Contenido.

Ilustraciones.

Tablas.

Introducción.

Tradicionalmente en el desarrollo de sistemas de información se han manejado los requisitos con métodos rudimentarios y casuales. Este engorroso mecanismo deja de ser viable para un nuevo mundo de continuos cambios en las políticas del negocio. Un método que divide los sistemas de información en tres componentes (datos, procesos y reglas de negocio) es más adecuado para desarrollar modernos sistemas de información. De acuerdo con esta visión, las reglas de negocio son una parte independiente de los requerimientos del sistema y necesitan un especial manejo (Zimbrão et al., 2002).

Las reglas son ciudadanos de primera clase en el mundo de los requisitos (Business Rules Group, 2003). Estas pueden ser consideradas como sentencias que permiten a los usuarios expertos definir políticas, condiciones y conocimientos del negocio en unidades pequeñas y aisladas. Los requisitos, enfocados hoy como reglas de negocio (RN), habitualmente han sido realizados “a mano” por el programador en el código fuente del programa, en los objetos de bases de datos de la aplicación, o en ambos.

Adicionando la automatización a este proceso se evitan fallas y, de este modo, se logran importantes reducciones de tiempo y costo. Para automatizar estos “requisitos” es necesario primeramente darles una estructura inicial y mediante un lenguaje técnico acercarlos a una representación más matemática, para luego aplicar un mecanismo de implementación.

Estudios actuales tienden a considerar la base de datos del negocio como un lugar ideal para implementar las reglas, debido principalmente a la cercanía con los datos del mismo. Son múltiples los recursos que lo facilitan, ofrecidos por el estándar SQL (Melton and Simon, 2002).

Este trabajo continúa una investigación iniciada en (Alonso, 2008) y enriquecida en (Toledo, 2009) para implementar reglas de negocio en bases de datos relacionales. Para ello se formaliza un lenguaje propio, enriquecido por un lenguaje reconocido mundialmente y en muchos aspectos semejante a este. Con esta base se es capaz de aceptar varios tipos de reglas

Se definen cuatro patrones para reglas de negocio, uno de ellos retomado de previa investigación (Alonso, 2008). Se define detalladamente como se ha de implementar cada tipo de regla y el lenguaje creado para admitirla. También se mejoran los mecanismos de comprobación de las políticas del negocio y se precisan los eventos que pueden generar inconsistencias o alertas.

La presente investigación utiliza como caso de estudio el esquema lógico del departamento de transplante renal en el Hospital Universitario “Arnaldo Milián Castro” presentado en Anexo1. Las reglas relacionadas como ejemplo no constituyen políticas reales de este negocio, son el resultado de planteamientos sobre posibles requerimientos y muestran la esencia del presente trabajo, que trasciende cualquier negocio en particular.

Problema de Investigación:

Nuestro problema radica en:

¿Cómo implementar reglas de negocio en bases de datos relacionales?

Objetivo General:

Determinar las bases teórico/prácticas para implementar reglas de negocio en bases de datos relacionales.

Objetivos Específicos:

- Definir patrones de reglas para captar los requerimientos del negocio.
- Establecer un lenguaje técnico a utilizar por las reglas de los patrones definidos.
- Implementar mediante recursos estándares de bases de datos las reglas de estos patrones y su lenguaje técnico.
- Proponer un método de manejo para las reglas implementadas.

Preguntas de Investigación:

- ¿Cuáles patrones de reglas de negocio utilizar?
- ¿Qué lenguaje técnico establecer para las reglas de los patrones definidos?
- ¿Qué recursos en base de datos relacionales posibilitan la implementación de reglas de negocio?
- ¿Cómo deben ser manejadas las reglas implementadas?

Hipótesis de Investigación:

Se puede, mediante recursos estándares de bases de datos relacionales, implementar y manejar reglas de negocio.

Descripción de Capítulos:

Esta investigación parte de un conocimiento conceptual y lógico del negocio independientemente de su estructura y origen. Su principal aporte es la formalización de un lenguaje matemático sencillo y natural, no muy complejo de alcanzar por los requerimientos informales del negocio.

Para las reglas técnicas que utilizan este lenguaje son creados varios patrones con el fin de estructurar su contenido y obtener reglas seminaturales, simplificando el tratamiento de las mismas. Estas reglas tratadas involucran solo requerimientos sobre los datos y su último nivel de expresión será en la base de datos del negocio.

El Capítulo I recoge aspectos generales sobre las reglas de negocio, se exploran los diferentes tipos de reglas y sus disímiles formas de implementación. Es examinado un lenguaje reconocido para reglas y se analizan varios recursos de base de datos que posibiliten un nivel de expresión final para las mismas.

El Capítulo II recoge los puntos más relevantes sobre los patrones de reglas de negocio. Se detallan estos patrones y el lenguaje utilizado para representarlos. Por último, son analizados los sucesos que pueden activar estas reglas luego de implementadas y cómo manejarlas para garantizar su objetivo.

El Capítulo III trata sobre la implementación en bases de datos relacionales de todos los aspectos abordados en el capítulo anterior. Para ello se utilizan únicamente los recursos que ofrece el estándar SQL y sus dialectos. En el transcurso del mismo son analizadas y elegidas equivalentes formas de implementación.

Capítulo I:

CONSIDERACIONES GENERALES SOBRE REGLAS DE NEGOCIO.

I. CONSIDERACIONES GENERALES SOBRE REGLAS DE NEGOCIO.

Este capítulo recoge aspectos generales sobre las reglas, así como los elementos necesarios para su ajuste en determinado negocio. Se explora su tipología y disímiles formas de implementación, profundizando sobre los recursos brindados por Base de Datos. Se examina además un lenguaje reconocido para expresar reglas.

I.1. ¿Qué son las Reglas de Negocio?

Existen varias definiciones de reglas de negocio, algunas sencillas y compactas, otras más amplias y detalladas. Una regla de negocio según Ronald G. Ross (Ross, 2010) es simplemente una regla que está bajo jurisdicción del negocio, lo cual significa que estas pueden ser creadas, revisadas y eliminadas cuando el negocio lo estime conveniente. Tony Morgan (Morgan, 2002) por otra parte las define como una declaración compacta sobre un aspecto del negocio, usando un lenguaje simple, inequívoco, accesible a todas las partes interesadas: el dueño del negocio, el analista, el arquitecto técnico, y así sucesivamente. Este último autor enumera características universales deseables en las declaraciones de reglas, las cuales se aplican a cualquier lenguaje o dominio de aplicación:

- Atómico: no pueden ser descompuestas sin que se pierda información.
- No ambiguo: tienen solamente una obvia interpretación.
- Compacta: típicamente son frases cortas.
- Consistente: juntas, ellas proporcionan una única y coherente descripción.
- Compatible: usan las mismas condiciones en el resto del modelo de negocio.

Dos ejemplos se muestran a continuación:

Nombre	Regla
R1:	“El salario mínimo de un nefrólogo es $457”.
R2:	“Un cirujano no debe atender más de 3 donantes vivos”.

Tabla 1: Ejemplos de Reglas de Negocio.

I.2. Beneficios al utilizar Reglas de Negocio.

Varios son los beneficios que pueden derivarse del uso de reglas de negocio. De todos, según Lowenthal (Lowenthal, 2005) los tres más importantes son:

- Agilidad: respuesta simple y rápida a los requisitos dinámicos.
- Reducción del Costo: bajo costo para crear o actualizar las partes de aplicaciones que implementan las políticas del negocio.
- Transparencia: las reglas permiten fácilmente la auditoría que los servicios de software llevan a cabo en sus políticas de negocios correspondientes.

I.3. Formas de expresión de las Reglas de Negocio.

De acuerdo con Morgan (Morgan, 2002) las reglas de negocio pueden expresarse de diversas formas, principalmente, según su depuración a lo largo del sistema o incluso en la manera en que se introduzcan a este. Es necesario señalar que existen varios modos a considerar, pero fundamentalmente, se definen tres niveles:

- Informal: este nivel proporciona una sentencia en lenguaje natural, sin un rango limitado de parámetro, tal y como el cliente del negocio desee.

- Técnico: este nivel combina referencias a datos estructurados, operadores y restricciones con el lenguaje natural, nivel intermedio entre la entrada de la regla y su implementación.
- Formal: este nivel proporciona sentencias conforme a una sintaxis definida y proporciona la funcionalidad automática de la regla.

I.4. Tipos de Reglas de Negocio.

Existen múltiples tipos de reglas de negocio. Debido a su diversidad y complejidad los autores tienden a agruparlas y clasificarlas siguiendo diferentes puntos de vista, pero siempre con el objetivo común de recoger todas las reglas del análisis de requisitos del negocio; por tanto, un estudio superficial de estas clasificaciones muestra claramente sus similitudes y la complementación entre ellas.

I.4.1. Clasificación según Weiden.

Según Weiden (Weiden et al., 2002) se pueden utilizar las propiedades semánticas para clasificar reglas de negocio. Estas clasificaciones pueden ser agrupadas en tres grandes categorías: estructurales, de comportamiento y de administración.

Categoría	Tipo de Regla	Nro.
Estructura	Estructura de Conceptos	1
	Persistencia	2
	Historia	3
De Comportamiento	Flujo de Información	4
	Pre Condición	5
	Post Condición	6
	Frecuencia	7
	Duración	8
	Flujo de Control	9
	Conocimiento de la Tarea	10
Administrativas	Organización	11
	Objetivo	12
	Valor	13
	Aptitud del Actor	14
	Responsabilidad del Actor	15
	Recursos	16

Tabla 2: Esquema de Clasificación Semántica de Reglas de Negocio.

I.4.2. Clasificación según Morgan.

Por el pensamiento de Tony Morgan (Morgan, 2002), la forma más conveniente de crear sentencias de regla es seleccionar patrones adecuados desde una pequeña lista disponible. Por ejemplo: <sentencia>:= <sujeto> debe <restricción>.

Agrega que no existe un estándar de como hacer reglas de negocio pero sí algunas recomendaciones. Los patrones pueden reflejar las formas o tipos de problemas que un sistema automatizado pretende negociar. Relaciona varios patrones: los de Restricción Básica, Lista de Restricciones, Clasificación, Cómputo y Enumeración.

Los convenios utilizados para los patrones son los siguientes:

- Paréntesis: ( ) Encierran un grupo de ítems.
- Corchetes: [ ] Encierran elementos opcionales.
- Barra Vertical: | Separa términos alternativos.
- Corchetes Angulares: <> Encierran términos especiales como los mostrados en la siguiente tabla.

Elemento	Significado
<determinante>	Es el determinante para cada sujeto, por ejemplo: Una, Uno, El, La, Cada, Todos.
<sujeto>	Es una entidad reconocible del negocio, tal como objetos, un nombre de Rol o una propiedad de un objeto. La entidad puede ser cualificada por otros elementos descriptores, tales como la existencia en un estado particular, relacionada con una aplicabilidad específica de la regla.
<característica>	El comportamiento del negocio a tomar lugar o interrelación que debe ser establecida.
<hecho>	Interrelación entre términos identificables en el modelo de hechos. Esta puede ser cualificada por otro elemento descriptor relativo a identificar la aplicabilidad de la regla precisada.
<lista de hechos>	Una lista de <hechos> elementos.

<m>,<n>	Son parámetros numéricos.
<resultado>	Cualquier valor, no necesariamente numérico, que tiene algún significado en el negocio. El resultado es usualmente el valor del atributo de un objeto del negocio.
<algoritmo>	Definición de una técnica usada para obtener el valor de un resultado; normalmente expresada utilizando combinaciones de términos del negocio junto a constantes disponibles.
<clasificación>	Definición de un término del negocio. Típicamente define el valor de un atributo, quizás llamado "estado" o algo similar, o un subconjunto de objetos en una clase existente.
<lista enumerada>	Lista de valores enumerados. La numeración abierta indica que la lista puede ser modificada por futuros requerimientos mientras que la cerrada no.

Tabla 3: Elementos en los patrones de Morgan.

Patrón de Restricción Básica: Es el más común entre los patrones para reglas de negocio, establece una restricción sobre el sujeto de una regla.

<determinante> <sujeto> [no] (debe | tiene) <característica>
[(si | a menos que) <hecho>].

Patrón de Lista de Restricción: Este patrón también restringe al sujeto, delimitando la cantidad de hechos verdaderos tomados de una lista.

<determinante> <sujeto> [no] (debe | tiene) <característica>
(si | a menos que) como mínimo <m> [no más de <n>] de las siguientes es verdadera: <lista de hechos>.

Clasificación: Establece la definición para un término del negocio.

<determinante> <sujeto> [no] es definido como <clasificación>
[(si | a menos que) <hecho>].

Enumeración: Este patrón establece un rango de valores que pueden ser legítimamente tomados.

<determinante> <resultado> debe ser elegido de la siguiente
[abierto | cerrado] enumeración: <lista enumerada>.

Cómputo: Establece relaciones entre términos del negocio suficientes para hallar o establecer un valor.

<determinante> <resultado> es definido como <algoritmo>
<determinante> <resultado> = <algoritmo>.

I.4.3. Clasificación según Ashwell.

Según Ashwell (Ashwell, 2006) se crea la regla de negocio usando una sentencia de declaración que al menos debe tener un sujeto, un verbo y un objeto; las palabras usadas deben tener un único significado dentro del dominio del negocio. Si son nombres, deben aparecer en el modelo de datos del negocio como entidades o atributos. Si hay verbos, éstos también deben aparecer en el modelo de datos como parte de las interrelaciones entre entidades. Si estas palabras no aparecen en el modelo, entonces deben presentarse en el glosario.

Los posibles tipos de reglas de negocio son:

- Restricciones: estas reglas previenen de posibles violaciones, principalmente en su forma o valor.
- Cómputo: reglas que calculan un valor aritmético o booleano basado en una fórmula.

- Acciones (al ocurrir un evento o satisfacer una condición) :
  - Habilitar o deshabilitar un proceso u otra regla.
  - Ejecutar un proceso.
  - Crear, modificar o eliminar datos.

I.4.4. Clasificación según Solivares.

Una buena clasificación de reglas de negocio es la que ofrece Solivares (P. A. Solivares; citados en (Besembel and Chacón)). De esta se muestran los cinco grupos más interesantes.

Reglas del Modelo de Datos:

El primer grupo de reglas de negocio, engloba todas aquellas encargadas de controlar que la información básica almacenada para cada atributo o propiedad de una entidad u objeto sea válida.

Reglas de Relación:

Otro grupo importante de reglas incluye todas aquellas que controlan las relaciones entre los datos. Estas reglas especifican, por ejemplo, que todo pedido debe ser realizado por un cliente, y el mismo debe ser atendido. Además, una vez que un cliente haya hecho algún pedido, se deberá garantizar no eliminarlo, a menos que previamente se eliminen todos sus pedidos.

Reglas de Restricción:

Otro grupo es el compuesto por las reglas que restringen los datos contenidos en el sistema. Nótese que este grupo se solapa en cierto modo con las reglas del modelo de datos, pues aquellas también impiden la introducción de datos erróneos. La diferencia estriba en que las reglas de restricción condicionan el valor de los atributos o propiedades de una entidad más allá de las restricciones básicas sobre las mismas existentes.

Intervalos Temporales:

Ocurren entre dos eventos específicos, por ejemplo: el cliente llama entre el envío de la oferta y 30 días después.

Periódicos:

Son aquellos que ocurren en intervalos fijos de tiempo, por ejemplo: cada quince días o cada cinco órdenes de compra.

I.5. Lenguaje de Especificación de Reglas de Negocio.

Diversos son los lenguajes declarativos creados para expresar reglas de negocio. Estos lenguajes recogen conceptos del negocio y los convierten en expresiones más o menos inteligibles para un especialista técnico. Por ello, su función se limita a crear un punto intermedio de expresión de las reglas; pues es el lenguaje natural, y no el técnico o el formal, el objetivo en la definición de las políticas. El propietario del negocio no puede lidiar con tal tipo de expresión, pero al diseñador se le humaniza el trabajo (Morgan, 2002).

I.5.1. Lenguaje de Restricción de Objetos.

Muchos autores exigen que las reglas de negocio deban declararse en algún lenguaje técnico (Bruegge and Dutoit, 2009, Demuth, 2004, Tedjasukmana, 2006, Zimbrão et al., 2002). Se han establecido diagramas y otras herramientas proporcionadas por el Lenguaje de Modelado Unificado (Unified Modeling Language, UML) como estándares para la modelación de sistemas de software. Una de las herramientas sugerida por UML para la especificación de restricciones es el Lenguaje de Restricción de Objetos (Object Constraint Language, OCL).

OCL es parte de UML, un estándar del Grupo de Manejo de Objetos (Object Management Group, OMG) para modelar aplicaciones orientadas a objetos. Este es un lenguaje técnico para definir restricciones en diferentes modelos UML de cualquier dominio. Muy poderoso al ser utilizado por diferentes capas de modelado. En los últimos años UML ha sido aceptado por la mayoría de los desarrolladores de software como herramienta de visualización, de documentación, de análisis y de diseño (Demuth, 2004).

Por ejemplo si en un modelo del hospital infantil se desea restringir la edad de los pacientes a menos de 8 años solo es necesario declarar en OCL la siguiente expresión:

```
context Paciente
inv:  self.edad  < 8
```

Según indica Bruegge (Bruegge and Dutoit, 2009) OCL puede ser usado con distintos propósitos:

- Para especificar características estáticas sobre clases y tipos en un modelo de clases.
- Para especificar características estáticas de tipo para Estereotipos.
- Para especificar pre y post-condiciones sobre Operaciones y Métodos.
- Como lenguaje de navegación.
- Para especificar restricciones sobre operaciones.

Existen varias características interesantes que presenta Verónica (Tedjasukmana, 2006):

- OCL es un lenguaje declarativo. En un lenguaje declarativo una expresión simplemente muestra que debe hacerse pero no cómo.
- Una expresión OCL garantiza ausencia de efectos colaterales. Cuando una expresión es evaluada simplemente retorna un valor. Ella no modificada nada en el modelo.

- OCL no es un lenguaje de programación y por tanto no es posible escribir lógica de programa o control de flujo. No se puede procesar o activar operaciones porque OCL en un lenguaje de modelación en primer lugar y sus expresiones por definición no son directamente ejecutables.

Según OMG (OMG, 2010) OCL puede ser utilizado para diferentes propósitos:

- Como un lenguaje de consulta.
- Especificar invariantes en clases y tipos del modelo de clases.
- Para describir pre y post operaciones en operaciones y métodos.
- Para especificar destinos para mensajes y acciones.
- Para especificar restricciones en operaciones.

I.5.2. Notación punto.

Según expone Morgan (Morgan, 2002) hay que decidir como referirse a los atributos de objetos. Por ejemplo, el objeto Paciente puede tener un atributo llamado Edad, un estilo es confiar en las estructuras normales disponible en español o cualquier idioma que se emplee: “la Edad del Paciente”, “del Paciente su Edad” o algo similar.

Otra opción a considerar es la utilización de la notación punto, produciendo expresiones como Paciente.Edad. Además se considera el uso de esta notación, para especificar las relaciones entre los objetos del negocio.

Esta alternativa es mucho más precisa y puede emplear los nombres del diagrama de clases, pero no es aceptable en el lenguaje informal del negocio. Se utiliza solo para expresiones en lenguaje técnico como OCL el cual lo emplea principalmente como lenguaje de navegación.

I.5.3. Implementaciones de Reglas de Negocio.

Las reglas se pueden representar formalmente de disímiles maneras e incluso pueden existir diferentes técnicas para una misma regla. Morgan (Morgan, 2002) muestra algunas de las posibles a utilizar.

Lenguajes de Programación:

Las reglas incorporadas en el código del programa, probablemente, son la vía de aplicación más común. Es posible utilizar sentencias normales de un lenguaje de programación para implementar una regla. El requisito mínimo es tener una forma de seleccionar, entre las ramas del código, una alternativa basada en una condición dada. Esto es bastante fácil de satisfacer en cualquier lenguaje de programación, aunque los detalles pueden variar de uno a otro.

Ejemplo en Object Pascal:

```
procedure R10 (paciente: TPaciente);
 begin
          ...
       if paciente.edad > 18 Then
//Código si la restricción es satisfecha
       else
//Código si la restricción no es satisfecha
          ...
 end.
```

Scripts:

La principal ventaja de los scripts es que al encontrarse en la aplicación cliente son muy veloces, aunque tienen algunos puntos negativos. De estos el fundamental es la dificultad en su manejo y modificación.

Ejemplo en Java Script:

```
<script language="JScript">
     function chequear(obj) {
          if (obj.Value > 18) {
          alert("la restricción es satisfecha ");}}
</script>
```

Bases de Datos:

Es probable que las reglas de negocio encajen más naturalmente dentro de la base de datos, donde pueden tener un contacto más directo con los datos según aclara Morgan (Morgan, 2002). Lo más habitual es utilizar las reglas de negocio respecto a palabras funcionales que estén alrededor de lo básico (CREATE, READ, UPDATE y DELETE). Estos mecanismos son comunes a todos los servicios de datos.

I.6. Recursos de Bases de Datos.

El lenguaje estructurado de consultas (Structured Query Language, SQL) estándar en sus múltiples revisiones y particularmente la implementación para SQL Server 2005 (MSDN, 2008) ofrece varios recursos de manejo de datos que se analizan en esta sección para seleccionar a la postre los más convenientes en la implementación de reglas de negocio.

Restricciones (Constraints) CHECK:

Las restricciones CHECK exigen la integridad del dominio mediante la limitación de los valores que puede aceptar una columna. Se puede crear una restricción CHECK con cualquier expresión lógica (booleana) que devuelva TRUE (verdadero) o FALSE (falso) basándose en operadores lógicos (Oppel and Sheldon, 2008).

Es posible aplicar varias restricciones CHECK a una sola columna y a varias columnas si se crea a nivel de la tabla. Así se pueden comprobar varias

condiciones en un mismo sitio. Una restricción CHECK devuelve TRUE cuando la condición que está comprobando no es FALSE para ninguna fila de la tabla. Si una tabla recién creada no tiene filas, cualquier restricción CHECK en esta tabla se considerará válida. Esta situación puede generar resultados inesperados al igual que en las instrucciones DELETE dado que las restricciones CHECK no se validan en ellas. (MSDN, 2008).

Desencadenadores (Triggers):

Los desencadenadores son una clase especial definida para la ejecución automática al emitirse una instrucción UPDATE, INSERT o DELETE en una tabla o una vista. Son una herramienta eficaz que pueden utilizar los sitios para exigir automáticamente las reglas comerciales cuando se modifican los datos. Amplían la lógica de comprobación de integridad, valores predeterminados y reglas del estándar SQL, aunque se deben utilizar las restricciones y los valores predeterminados siempre que estos aporten toda la funcionalidad necesaria (Melton and Simon, 2002).

Los desencadenadores pueden consultar otras tablas e incluir instrucciones SQL complejas. Son especialmente útiles para exigir reglas o requisitos complejos. También son útiles para exigir la integridad referencial, que conserva las relaciones definidas entre tablas.

CREATE TRIGGER debe ser la primera instrucción en el proceso por lotes y solo se puede aplicar a una tabla. Un desencadenador se crea solamente en la base de datos actual; sin embargo, un desencadenador puede hacer referencia a objetos que están fuera de la base de datos actual (MSDN, 2008).

Vistas:

Una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla real, una vista consta de un conjunto de columnas y filas de datos con un nombre. Suelen utilizarse para centrar, simplificar y

personalizar la percepción de la base de datos para cada usuario (Melton and Simon, 2002).

Las vistas se pueden utilizar para realizar particiones de datos y para mejorar el rendimiento cuando estos se copian. Además, permiten a los usuarios centrarse en datos de su interés y en tareas específicas de las que son responsables. Los datos innecesarios pueden quedar fuera de la vista; de ese modo, también es mayor su seguridad, dado que los usuarios solo pueden ver los definidos en la vista y no los que hay en la tabla subyacente (MSDN, 2008).

Funciones definidas por el usuario:

Al igual que las funciones en los lenguajes de programación, las funciones definidas por el usuario de Microsoft SQL Server 2005 (MSDN, 2008) son rutinas que aceptan parámetros, realizan una acción, como un cálculo complejo, y devuelven el resultado de esa acción como un valor. Son múltiples las ventajas de las funciones definidas por el usuario entre las cuales se tienen:

- Permiten una programación modular.
- Permiten una ejecución más rápida.
- Pueden reducir el tráfico de red.

Procedimientos Almacenados:

Los procedimientos almacenados pueden facilitar en gran medida la administración de la base de datos y la visualización de información sobre esta y sus usuarios. Son una colección precompilada de instrucciones SQL e instrucciones de control de flujo opcionales, almacenadas bajo un solo nombre y procesadas como una unidad. Estos se guardan en una base de datos, permitiendo ser ejecutados desde una aplicación.

Los procedimientos almacenados pueden contener flujo de programas, lógica y consultas a la base de datos; aceptar parámetros y proporcionar sus resultados, devolver conjuntos individuales o múltiples y retornar valores.

Los procedimientos almacenados de Microsoft SQL Server son similares a los procedimientos de otros lenguajes de programación en el sentido de que pueden:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al lote o al procedimiento que realiza la llamada.
- Contener instrucciones de programación que realicen operaciones en la base de datos, incluidas las llamadas a otros procedimientos.
- Devolver un valor de estado a un lote o a un procedimiento que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores (y el motivo de éstos).

Se pueden utilizar procedimientos almacenados para cualquier finalidad que requiera la utilización de instrucciones SQL, con estas ventajas:

- Ejecución de una serie de instrucciones SQL en un único procedimiento almacenado.
- Referenciar a otros procedimientos almacenados desde el propio procedimiento almacenado, con lo que se puede simplificar una serie de instrucciones complejas.
- El procedimiento almacenado se compila en el servidor cuando se crea, por tanto, se ejecuta con mayor rapidez que las instrucciones SQL individuales.

(MSDN, 2008).

Assertions:

Las assertions son un tipo de restricción especial que se puede especificar en SQL sin que deba estar asociada a una tabla en particular, como es el caso de los constraints (Mota, 2005). Generalmente son utilizados para describir restricciones que afectan a más de una tabla. Como los constraints

solo pueden establecerse sobre tuplas de una tabla, las assertions son útiles cuando es necesario especificar una condición general de la base de datos que no se puede asociar a una tabla específica. Por esta característica de poder especificar una assertion para toda la base de datos y no para una tabla en particular, se le llama restricción autónoma. Esto tiene grandes ventajas a nivel conceptual, pero las hace difíciles de implementar.

Transacciones:

Una transacción es una unidad única de trabajo. Si una transacción tiene éxito, todas las modificaciones de los datos realizadas durante la transacción se confirman y se convierten en una parte permanente de la base de datos. Si una transacción encuentra errores y debe cancelarse o revertirse, se borran todas las modificaciones de los datos (Gennick, 2006).

Una transacción es una secuencia de operaciones realizadas como una sola unidad lógica de trabajo. Una unidad lógica de trabajo debe exhibir cuatro propiedades, conocidas como propiedades de atomicidad, coherencia, aislamiento y durabilidad (ACID), para ser calificada como transacción.

Una transacción debe ser una unidad atómica de trabajo, tanto si se realizan todas sus modificaciones en los datos, como si no se realiza ninguna de ellas. Cuando finaliza, una transacción debe dejar todos los datos en un estado coherente. En una base de datos relacional, se deben aplicar todas las reglas a las modificaciones de la transacción para mantener la integridad de todos los datos. Todas las estructuras internas de datos, como índices de árbol B o listas doblemente enlazadas, deben estar correctas al final de la transacción.

Las modificaciones realizadas por transacciones simultáneas se deben aislar unas de otras. Una transacción reconoce los datos en el estado en que estaban antes de que otra transacción simultánea los modificara o después

de que la segunda transacción haya concluido, pero no reconoce un estado intermedio. Esto se conoce como seriabilidad, ya que deriva en la capacidad de volver a cargar los datos iniciales y reproducir una serie de transacciones para finalizar con los datos en el mismo estado en que estaban después de realizar las transacciones originales (MSDN, 2008).

Una vez concluida una transacción, sus efectos son permanentes en el sistema. Las modificaciones persisten aún en el caso de producirse un error del sistema.

I.7. Conclusiones Parciales.

En este capítulo se ha brindado un cuadro teórico sobre las reglas de negocio, de las cuales se ofrecieron patrones que las estructuran en un lenguaje seminatural para facilitar su manejo e implementación. Esto proporciona un conocimiento preliminar sobre los objetivos generales de este trabajo, y facilita la comprensión de las decisiones a tomar posteriormente. Se estudió un lenguaje para expresar reglas de negocio y los recursos en Base de Datos que pueden ser empleados para implementarlas.

Capítulo II:

PATRONES Y LENGUAJE PARA REGLAS DE NEGOCIO.

II. PATRONES Y LENGUAJE PARA REGLAS DE NEGOCIO.

En este capítulo se recogen los puntos más relevantes sobre la formalización de las reglas de negocio en patrones. Se detallan estos patrones de estructuración así como el lenguaje utilizado para conformarlos.

Algunas de estas reglas luego de implementadas necesitan ser invocadas por determinados sucesos del negocio. Estos aspectos son abordados en esta sección junto a su modo de empleo.

Como cierre se presenta la arquitectura de un sistema gerenciador de reglas de negocio creado en investigación previa (Alonso, 2008). Mediante esta arquitectura se muestra el ciclo de vida de las reglas y aunque el presente capítulo realiza profundas modificaciones teóricas en este sistema su arquitectura permanece inflexible.

II.1. Patrones para Reglas de Negocio.

La creación de reglas de negocio mediante un lenguaje natural completamente libre es un tema bien complejo. Por tanto, es deseable reducir esta complejidad sin afectar las opciones de aceptar requerimientos en un negocio. Aquí empiezan a aparecer las bondades de utilizar patrones en la captación de reglas, pues por una parte se restringe el lenguaje informal del usuario y por otra se brinda una estructura bien definida que ayuda al tránsito de la regla por los diferentes niveles de expresión, hasta llegar a su implementación.

Esta investigación, aunque parte de un lenguaje técnico, le es válida dicha idea. Al tomar reglas desde determinados patrones se establece un punto de contacto con el lenguaje seminatural y se estructura un método de transformación hacia el lenguaje formal en cada patrón, según sus particularidades.

De las diversas clasificaciones de reglas vistas en el Capítulo I se tomaron, como modelo, los patrones de reglas mencionados por Morgan (Morgan, 2002) . Estos patrones se encuentran bien definidos y son un importante punto de inicio y continuidad de la presente investigación.

II.2. Patrones definidos.

Los patrones a definir en el presente trabajo son los siguientes:

- Patrón de Clasificación.
- Patrón de Cómputo.
- Patrón de Notificación.
- Patrón de Restricción.

En investigación previa (Alonso, 2008) se aborda el patrón de restricción, los tres restantes son exigencias reales del negocio expresadas por múltiples autores (Ashwell, 2006, Morgan, 2002, Ross, 2010). Los patrones de Clasificación y Cómputo parten de la investigación de Morgan (Morgan, 2002) mientras el de Notificación es resultado de la presente tesis.

Los patrones, como formas de definir reglas de negocio, responden a la necesidad de especificar requisitos del negocio. Por lo tanto es acertado imaginar una relación directa entre el aumento de patrones y el aumento de requerimientos aceptados. Esto no quiere decir que sea correcto definir innumerables patrones, se deben tener argumentos de su necesidad o lo que es lo mismo comprobar la existencia de requerimientos que lo transformen en indispensable.

Comparar un patrón con otro en cuanto a su importancia es realmente difícil, todos los que se comprueben necesarios son igualmente significativos pues indistintamente satisfacen requerimientos del negocio y se complementan unos a otros. Únicamente negocios en particular donde se denoten los empleos de patrones o se establezcan niveles de importancia a los mismos se puede realizar este tipo de comparación.

Los convenios utilizados para precisar elementos variables de los patrones definidos en este trabajo son:

- Paréntesis: ( ) Encierran un grupo de ítems.
- Corchetes: [ ] Encierran elementos opcionales.
- Barra Vertical: | Separa términos alternativos.
- Corchetes Angulares: <> encierran términos especiales como los mostrados en la siguiente tabla.

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 una entidad reconocible del negocio o una clasificación de la misma.
<hecho>	Hechos relativos al estado o comportamiento del negocio.
<característica>	Describe las características del sujeto en el negocio, tanto internas como relacionadas con otras entidades. Pueden incluir hechos con el fin de caracterizar al sujeto.

<resultado>	Cualquier valor, no necesariamente numérico, que tiene algún significado en el negocio. El resultado es usualmente el valor del atributo de un objeto del negocio.
<algoritmo>	Definición de una expresión matemática para obtener el valor de un resultado; normalmente expresada utilizando combinaciones de términos del negocio junto a constantes disponibles y operadores.
<clasificación>	Definición de un término del negocio. Típicamente define el valor de un atributo o un subconjunto de objetos en una clase existente.
<mensaje>	Mensaje de información entre comillas para usuarios autorizados del negocio.

Tabla 4: Elementos de variables de los patrones.

Estos patrones para reglas de negocio son aplicados desde un nivel informal. No varían al transformarse las reglas a un nivel técnico y objetivamente la presente investigación se centra en su implementación. Los ejemplos de reglas mostrados parten de un lenguaje seminatural (producto de los patrones) y se observa su ciclo de vida completo.

II.2.1. Patrón de Clasificación.

Este patrón puede ser catalogado como una regla de definición (también llamadas reglas estructurales). Estas reglas organizan el conocimiento básico del negocio contribuyendo claramente al significado de conceptos, centrándose en la esencia del mismo (Ross, 2010). Este patrón toma la siguiente estructura:

<determinante> <sujeto> [no] es definido como <clasificación>
[(si | a menos que) <característica>]

Ejemplo de este patrón en lenguaje seminatural podría ser:

RN#1: Un cirujano es definido como excelente cirujano si los donantes vivos operados por él poseen evoluciones satisfactorias.

<sujeto> : cirujano
<clasificación>: excelente cirujano
<característica>: los donantes vivos operados por él poseen evoluciones satisfactorias

Nótese que se está introduciendo con esta regla un nuevo término en el negocio (excelente cirujano). Estos conceptos pueden ser utilizados en las <características> de nuevas reglas, lo que será tratado más adelante. Tal es el caso de “evoluciones satisfactorias”, el cual es un concepto reconocible del negocio y que se puede tener definido mediante la siguiente regla de clasificación en lenguaje seminatural:

RN#2: Una evolución es definido como evolución satisfactoria si su temperatura es menor de 37ºC.

<sujeto> : evolución
<clasificación>: evolución satisfactoria
<característica>: la temperatura es menor de 37ºC

Una regla con este patrón clasifica a un sujeto según determinadas características del mismo. En este aspecto existen ligeras modificaciones con el patrón mencionado por Morgan (Morgan, 2002), debido principalmente a las transformaciones en los elementos variables de los patrones. Este tipo de regla crea una <clasificación>, la cual se incorpora al vocabulario del negocio y posteriormente puede ser utilizada por otras reglas.

Este patrón deriva en la comprobación de si determinado <sujeto> puede o no ser clasificado como <clasificación>. Por tanto este tipo de regla no puede ser violada directamente, esta conclusión es generalizada para cualquier regla de definición, tal como lo ratifica Ronald G. Ross (Ross, 2010), considerado padre de las reglas de negocio.

Recordar que unas de las características necesarias en las reglas de negocio es que no se puedan descomponer en otras reglas sin pérdida de información lo cual es confirmado por el propio Garros (Ross and Lam); aunque este criterio está sujeto al análisis de los requisitos del negocio. Enúnciese la siguiente regla en lenguaje seminatural:

RN#3: Un cirujano es definido como excelente cirujano si los donantes vivos operados por él poseen evoluciones con temperaturas menores de 37ºC.

<sujeto> : cirujano
<clasificación>: excelente cirujano
<característica>: los donantes vivos operados por él poseen evoluciones con temperaturas menores de 37ºC

Esta última regla es el resultado de integrar las dos primeras en una sola. En principio esta regla es divisible sin perder información y por tanto incorrecta, pues no es atómica, pero, cuál sería el sentido de definir “evoluciones satisfactorias” si donde único se utiliza es en esta regla y no constituye una necesidad de concepto para el negocio. Con esta consideración no se estaría cometiendo una violación a las características deseables de las reglas. Otro caso es si las “evoluciones satisfactorias” fuesen necesarias como término del negocio.

Por tanto, el patrón de clasificación satisface la necesidad de crear conceptos complejos del análisis de requisitos del negocio y contribuye a la fácil modificación de los mismos; con más razón si existen en un negocio determinados términos que son dinámicos y sus variaciones frecuentes.

II.2.2. Patrón de Cómputo.

Este patrón igualmente pertenece al grupo de reglas de definición y como tal, comparte grandes semejanzas con el patrón de clasificación (Ross, 2010). Su objetivo es calcular un valor determinado en el negocio, asociado o no al <sujeto> y su resultado es numérico.

Este patrón toma la siguiente estructura:

<determinante> <resultado> [en <sujeto>] es calculado como <algoritmo>.

Ejemplo de este patrón es la siguiente regla de negocio en lenguaje seminatural:

RN#4: El porciento de temperatura alta en paciente es calculado como (todas sus evoluciones con temperatura mayor que 38ºC por 100) dividido por todas las evoluciones en pacientes con temperatura alta.

<resultado> : porciento de temperatura alta
<sujeto>: paciente
<algoritmo>: (todas sus evoluciones con temperatura mayor que 38ºC por 100) dividido por todas las evoluciones en pacientes con temperatura alta

Este patrón introduce nuevos términos en el negocio que igualmente deben ser insertados en el vocabulario del mismo y también puede utilizar términos ya existentes. En este caso por ejemplo, el término “evoluciones en pacientes con temperatura alta” puede ser planteado por la siguiente regla en lenguaje seminatural:

RN#5: La cantidad de temperaturas altas para pacientes es calculado como la cantidad de evoluciones de pacientes con temperatura mayor de 38ºC.

<resultado> : cantidad de temperaturas altas para pacientes
<algoritmo>: la cantidad de evoluciones de pacientes con temperatura mayor de 38ºC

Respecto al patrón original de Morgan (Morgan, 2002) la única diferencia radica en la inclusión opcional de un sujeto. Este cambio responde a la necesidad de definir cálculos solo con respecto a un <sujeto> específico del negocio y no con respecto a todos. Cuando se incluye un <sujeto> se está definiendo un concepto del mismo, como sucede con las reglas de clasificación, de lo contrario se está definiendo simplemente un término del negocio, en ambos casos términos numerables.

Este tipo de regla no puede ser violada directamente y simplifica la modificación de términos variables en el negocio, pues solo se necesita modificar esta para que tenga efecto inmediato en todas las que la utilizan. Cumple además las mismas propiedades de atomicidad citada por varios autores (Lowenthal, 2005, Ross and Lam) y explicadas en el patrón de clasificación, lo cual puede ser igualmente ilustrado con la siguiente regla en lenguaje seminatural:

RN#6: El porciento de temperatura alta en paciente es calculado como (todos sus exámenes con temperatura mayor que 38ºC por 100) dividido por la cantidad de exámenes de pacientes con temperatura mayor de 38ºC.

<resultado> : porciento de temperatura alta
<sujeto>: paciente
<algoritmo>: (todos sus exámenes con temperatura mayor que 38ºC por 100) dividido por la cantidad de exámenes de pacientes con temperatura mayor de 38ºC

II.2.3. Patrón de Notificación.

La finalidad de este patrón es tan sencilla como útil; consiste en informar a los usuarios autorizados del negocio sobre algún conocimiento básico en tiempo real. Este patrón se define como:

Notificar <mensaje> si <hecho>

Como comenta Ross (Ross, 2010) si se piensa solo en reglas de negocio simplemente como restricciones complejas y rápidas se pierde una parte importante del aspecto global de las mismas, las reglas en un amplio sentido tratan siempre de distribuir conocimientos básicos del negocio en tiempo real. Las reglas de notificación son una parte importante de este esquema.

Considérese el siguiente ejemplo de reglas de notificación, en lenguaje seminatural.

RN#7: Notificar “Alerta Roja de la Pandemia” si la cantidad de pacientes con exámenes cuyo resultado es el virus AH1N1 exceden los 800.

<mensaje> Alerta Roja de la Pandemia
<hecho> la cantidad de pacientes con exámenes cuyo resultado es el virus AH1N1 exceden los 800.

Estas reglas pueden ser violadas directamente aunque no se aplica ninguna acción. Responden a la necesidad, no de restringir algún estado del negocio sino de anunciarlo a las partes interesadas, esta información es simplemente publicada y según el interés del usuario es o no consultada. Un posible propósito para este patrón es la toma de decisiones en el negocio o simplemente brindar soluciones para el mismo.

II.2.4. Patrón de Restricción.

Este patrón ha sido abordado ampliamente en investigaciones previas (Alonso, 2008, Toledo, 2009) y no se ha modificado sustancialmente. Su utilidad para satisfacer requisitos del negocio es evidente, solo con reglas de este tipo puede hacerse cumplir políticas prohibitorias en el sistema del negocio.

Su patrón, respetando los convenios definidos toma la siguiente forma:

<determinante> <sujeto> (no puede tener <características>) |
(puede tener <características> solo si <hechos>).

En su estructura no posee ningún cambio pero existe una ligera modificación que lo hace aún más potente. En (Alonso, 2008) se encuentra el siguiente significado de las <características> “Describe las características del sujeto en el negocio, tanto internas como relacionadas con otras entidades.” el cual difiere a las <características> definidas para los patrones de la presente tesis.

En el primer significado se exige utilizar siempre al <sujeto> en la especificación de sus características, pero, ¿qué sucedería si se necesitara utilizar hechos de la base de datos del negocio? Con motivo de valorar esta cuestión se examina un ejemplo de regla de negocio, tipo restricción en lenguaje seminatural.

RN#8: Un cirujano no puede tener más donantes vivos que la sexta parte de todos los donantes vivos.

<sujeto>: cirujano
<características>: más donantes vivos que la sexta parte de todos los donantes vivos

Como se puede apreciar, se está restringiendo la cantidad de donantes vivos de un cirujano utilizando un hecho, el cual no está asociado al cirujano de la regla sino a la base de datos del negocio. Este hecho es la cantidad total de donantes vivos y la incapacidad de representarlo en las <características> conlleva a la ampliación de las mismas.

II.3. Lenguaje para Reglas Técnicas.

Los patrones para reglas de negocio surgen como necesidad de captar reglas en un lenguaje seminatural y sus diferentes tipos cubren los requerimientos del negocio, pero ¿qué sentido tendrían estos patrones si las reglas no pudieran ser expresadas técnicamente para su posterior y final implementación?

Aquí se evidencia la importancia de contar con un potente lenguaje, capaz de asimilar una regla casi natural en un lenguaje más matemático y sencillo de implementar. Esta idea es confirmada por varios autores como (Demuth, 2005, Heidenreich et al., 2005, Zimbrão et al., 2002).

El Lenguaje para Reglas Técnicas (LRT) no es más que una expresión matemática de los elementos que conforman los patrones de regla vistos anteriormente. Para este fin el lenguaje posee una notación específica, operadores lógicos, operadores aritméticos y funciones, que juntos conforman una potente herramienta para aceptar múltiples reglas seminaturales.

La presente investigación no estudia su alcance y es natural que múltiples expresiones no puedan ser asimiladas por el mismo. Este lenguaje fue propuesto en tesis previa (Alonso, 2008) donde se define esencialmente su gramática, en el presente trabajo es formalizado, reconsiderado y ampliado a partir de nuevos análisis, preservado siempre una estructura sencilla. Esta estructura le brinda dos ventajas fundamentales:

- La simple transformación desde un lenguaje seminatural.
- Creación relativamente fácil por un especialista técnico del negocio.

Mantener u obtener un lenguaje técnico simple es todo un reto debido a que la complejidad del mismo aumenta cuando se expande su soporte. Es este el caso de lenguajes como OCL el cual es muy difícil para los modeladores del negocio como afirma Demuth (Demuth, 2004). El lenguaje presentado busca un equilibrio entre el lenguaje OCL y la simplicidad deseada, teniendo en cuenta solo variantes indispensables, implícitas en su definición.

II.3.1. La Notación Punto, elementos individuales y múltiples.

La notación punto del LRT más que una notación es un estilo. Su existencia es imprescindible, pues establece el medio de acceso a los atributos de entidades y posibilita la navegación, lo cual refiere Zimbrão (Zimbrão et al., 2002). Esta notación le brinda consistencia al lenguaje y sus dos usos fundamentales se relatan a continuación:

Acceso simple a un atributo:

Entidad $_1$ [. Atributo]

Camino de navegación entre entidades:

Entidad $_1$. Entidad $_2$. (…) . Entidad $_N$ [. Atributo]

En ambos casos es necesario destacar la opción de terminar o no con un atributo en la última entidad. Si se finaliza con un atributo se estaría en un caso ya visto en estudio previo a este trabajo (Alonso, 2008), pero si concluyera solamente con el nombre de una entidad entonces se estaría haciendo referencia al objeto como tal, específicamente a su(s) atributo(s) identificador(es).

En esta notación es admisible emplear como entidad la palabra reservada *sujeto*, la cual no pertenece realmente a ninguna entidad específica, su utilización referencia al <sujeto> de la regla. Esta constante de la notación punto es similar al *this* de los diferentes lenguajes de programación (Horstmann, 2009). En el patrón de restricción visto por la precedente investigación (Alonso, 2008) existe una variante similar utilizada en sus <hechos>, aunque bastante limitada: *@idSujeto*. Esta constante (*sujeto*) es aplicada igualmente a los patrones de Clasificación y Cómputo, siempre y cuando este último contenga <sujeto>. Ejemplos de esta palabra reservada serán tratados más adelante aunque su utilización fundamental toma la forma de:

sujeto. Entidad **1** . Entidad **2** . (…) . Entidad **N** [. Atributo]

Al utilizar la notación punto se tiene como resultado elementos individuales o elementos múltiples. Los primeros son útiles para el acceso a atributos específicos de una entidad. En el siguiente diagrama:

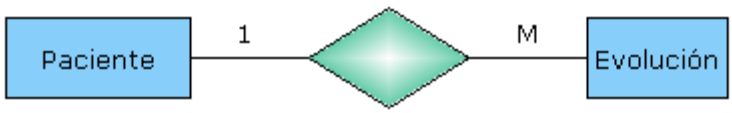

Ilustración 1: Porción del Esquema E/R para Transplante Renal.

Al expresar en lenguaje natural: “El paciente de la evolución…” (Evolución.Paciente) se está refiriendo a un único paciente, pues al ser la relación uno a muchos, se tiene que una evolución solo posee un paciente.

La notación punto no es el único modo de obtener elementos individuales ya que estos pueden ser el resultado de operadores de conjuntos, los cuales se verán más adelante, o simplemente constantes.

Los elementos múltiples surgen cuando a partir de la notación punto se obtienen como resultado varios elementos, por ejemplo, utilizando la Ilustración 1, al indicar “Las Evoluciones del Paciente…” (Paciente.Evolución) se conciben varias evoluciones relacionadas con un único paciente, como muestra la cardinalidad del diagrama. Nótese que el orden de la notación es muy importante.

En estudio previo (Toledo, 2009) se extiende la posibilidad de navegar por relaciones entre entidades de cualquier cardinalidad. Por tanto un elemento múltiple es únicamente resultado de una navegación en la cual al menos se tenga una interrelación (M:M) o (1:M). Igualmente se distingue la obtención elementos simples, aunque un elemento es simple cuando no es múltiple.

II.3.2. Operadores lógicos, aritméticos y de relación del LRT.

Los operadores lógicos utilizados por el LRT son los básicos para construir sentencias con un alto grado de complejidad. A continuación se muestran los mismos.

Operador	Significado	Argumentos/Resultado
O	Verdadero si alguna expresión lógica es verdadera.	Lógico
Y	Verdadero si ambas expresiones lógicas son verdaderas.	Lógico
NEG	Negación de cualquier valor lógico devuelto por otro operador lógico.	Un único argumento Lógico

Tabla 5: Operadores lógicos del LRT.

Estos operadores lógicos son utilizados igualmente por OCL como tratan varios autores (OMG, 2010, Tedjasukmana, 2006). Algunos rechazan el uso de estos por considerar que las reglas deben ser siempre extremadamente sencillas (Morgan, 2002), pero la realidad demanda en no pocos casos reglas con una lógica compleja.

Operador	Significado	Argumentos/Resultado
+	Adición	Numéricos
-	Resta	Numéricos
*	Multiplicación	Numéricos
/	División	Numéricos

Tabla 6: Operadores aritméticos del LRT.

Estos operadores aritméticos son muy útiles para las reglas de tipo cómputo, aunque pueden ser ampliamente utilizados en el contexto de otras reglas.

Operador	Significado	Argumentos	Resultado
=	Igual a	Numéricos/ Lógicos /Cadenas	Lógico
>	Mayor que	Numéricos	Lógico
<	Menor que	Numéricos	Lógico
>=	Mayor igual a	Numéricos	Lógico
<=	Menor igual a	Numéricos	Lógico
<>	Diferente a	Numéricos/ Lógicos/Cadenas	Lógico

Tabla 7: Operadores de relación del LRT.

Tanto los operadores aritméticos como los de relación son exactamente los mismos del OCL, especificados por su versión 2.2 (OMG, 2010). Estos últimos pueden utilizarse indistintamente para comparar elementos individuales y múltiples, obteniéndose cuatro combinaciones diferentes las cuales se muestran a continuación.

(Elemento Individual) operador de relación (Elemento Individual): Sentencia común en el LRT para establecer comparaciones simples. Un ejemplo se puede encontrar en la RN#7.

(Elemento Múltiple) operador de relación (Elemento Individual): Sentencia en LRT para restringir la colección de elementos mediante el elemento individual. Un ejemplo se puede encontrar en la RN#5. Un caso equivalente es **(Elemento Individual) operador de relación (Elemento Múltiple)**.

(Elemento Múltiple 1) operador de relación (Elemento Múltiple 2): Sentencia en LRT para restringir todos los elementos de la primera colección según los elementos de la segunda colección.

II.3.3. Operadores de conjunto del LRT.

Al utilizar la notación punto es posible obtener múltiples elementos. Estas colecciones necesitan ser manipuladas para expresar reglas de negocio, aquí nacen los operadores de conjuntos del lenguaje.

Operadores	Significado	OCL
CANTIDAD(<colección>)	Retorna la cardinalidad de la colección de elementos.	collection ->size()
VACÍO (<colección>)	Retorna verdadero si la colección no contiene elementos.	collection ->isEmpty()
EXISTE(<elemento>, <colección>)	Retorna verdadero si al menos un elemento existe en la colección.	collection-> exists(boolean expr)
PROMEDIO(<colección>)	Retorna el promedio de la colección numérica.	collection ->avg()
SUMA(<colección>)	Retorna la suma de la colección numérica.	collection ->sum()
MÍNIMO(<colección>)	Retorna el mínimo de la colección numérica.	collection ->min()
MÁXIMO(<colección>)	Retorna el elemento máximo de la colección.	collection ->max()

PROMDIF(<colección>)	Retorna el promedio de los elementos diferentes de la colección numérica.	-
CANTDIF(<colección>)	Retorna cuántos elementos diferentes contiene la colección de elementos.	-
SUMADIF(<colección>)	Retorna la suma de los elementos diferentes de la colección numérica.	-

Tabla 8: Operadores de conjunto del LRT.

II.4. Interdependencia de las Reglas de Negocio.

En investigación previa (Alonso, 2008) el único tipo de regla estudiado fue la restricción y no existían interdependencia entre ellas. Con los nuevos patrones para reglas de negocio no sucede de igual forma, ya se observó cuando se analizaron los patrones de Clasificación y Cómputo que entre ellos mismos podían existir relaciones.

Suponer los siguientes ejemplos de reglas de negocio en sus dos niveles de expresión inicial:

Lenguaje Seminatural:

RN#9: Un Riesgo Potencial no puede tener menos de 10 Exámenes ni ser un Donante Potencial.

RN#10: Un Paciente es definido como Riesgo Potencial si sus Síntomas Históricos Negativos son mayores que 80 o el promedio de la tensión arterial en sus evoluciones es mayor de 160.

RN#11: Los Síntomas_Históricos_Negativos de un Paciente es calculado como la suma de la cantidad de Evoluciones con temperaturas mayores de 38ºC.

Lenguaje Técnico en LRT:

RN#9: Un Riesgo Potencial no puede tener CANTIDAD(sujeto.Exámenes) < 10 Y NEG (VACÍO(sujeto.DonantePotencial))

RN#10: Un Paciente es definido como Riesgo Potencial si Síntomas_Históricos_Negativos (sujeto) > 80 O PROMEDIO(Paciente (sujeto).Evolución. tensión _ arterial) > 160

RN#11: Los Síntomas_Históricos_Negativos Paciente es calculado como CANTIDAD (Paciente(sujeto). Evolución.temperatura > 38)

Obsérvese que la regla RN#9 utiliza en el <sujeto> una clasificación de un paciente (Riesgo Potencial), por tanto depende de la regla que lo especifica RN#10. A su vez esta última maneja un cálculo de los pacientes (Síntomas Históricos Negativos) y por tanto depende de RN#11 la cual es totalmente independiente.

Para los patrones definidos en esta investigación se muestran sus posibles interdependencias mediante la siguiente tabla:

Tipo de Regla	Posibles Reglas a depender
CLASIFICACIÓN	CLASIFICACIÓN CÓMPUTO
CÓMPUTO	CÓMPUTO
NOTIFICACIÓN	CLASIFICACIÓN CÓMPUTO
RESTRICCIÓN	CLASIFICACIÓN CÓMPUTO

Tabla 9: Interdependencia de los Patrones para Reglas de Negocio

Este aspecto interesante de las reglas conlleva a problemas adicionales. Una posible interdependencia entre las reglas se puede mostrar mediante el siguiente grafo dirigido:

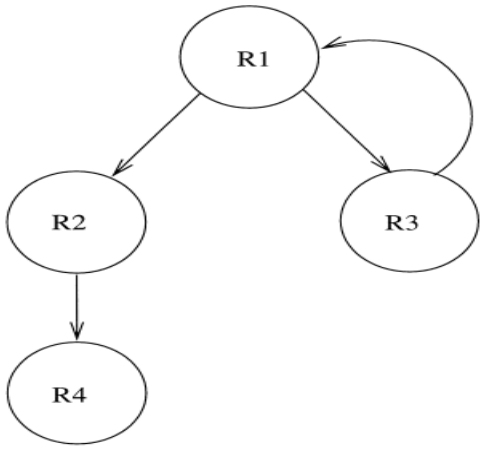

Ilustración 2: Ciclo entre Reglas de Negocio.

Obviamente se tiene un ciclo en el que nunca se podrá comprobar la regla uno (R_1) pues depende de la regla tres y esta a su vez de la R_1. Estos casos de interdependencia cíclica deben ser evitados (Paton and Díaz, 1999) para lo cual se puede utilizar teoría de grafos.

Otro aspecto a lo cual la presente investigación no dedica recursos es al hecho de la eliminación de las reglas. Una ventaja que ofrece la interdependencia es que la modificación de una regla no afecta directamente a quienes dependen de ella, otro asunto es cuando se decide eliminarla. En este caso habría que lidiar con todas las reglas a la cual se encuentra asociada.

II.5. Creación de Lista de Eventos.

La Lista de Eventos es relevante en el proceso de integridad de las reglas, específicamente las dependientes a los cambios de estado del negocio. De los patrones examinados solo tienen dicha propiedad los de restricción y notificación, debido en los primeros, a posibles violaciones, y en los segundos a potenciales estados informativos. Aunque ha de tenerse en cuenta que estos patrones pueden depender de otros tipos de patrones.

Su objetivo por tanto es discernir cuales reglas deben ser evaluadas tras algún evento ocurrido. Para ello primeramente se debe buscar, en cada regla, qué eventos posibilitan un cambio de estado relacionado a ella.

Una regla tipo restricción solo puede ser violada en sus <características>, para las de notificación sus <hechos> se consideran el elemento detonante. De ambos elementos se conoce que están formalizados en lenguaje técnico (LRT) cuyo núcleo consiste en la notación punto. El resultado del camino de navegación es el único elemento que puede variar finalizada una operación en la base de datos del negocio, por tanto su análisis se hace imprescindible.

La notación punto se compone por caminos de navegación entre entidades. Estos caminos, expresados en un lenguaje matemático, son creados con información conceptual del negocio, mientras la lista de eventos utiliza información relacional o lógica, por lo que se detalla esta última.

Un esquema relacional como lo define Ullman (Ullman, 1982) puede ser representado mediante un digrafo como en investigación previa (Toledo, 2009), donde los nodos representan relaciones y las aristas interrelaciones que tienen lugar en el esquema conceptual, expresadas en el enfoque relacional mediante llaves foráneas y orientadas según el sentido de estas.

Imagínese un camino de navegación entre entidades:
Entidad **1** . Entidad **2** . Entidad **3** . Atributo

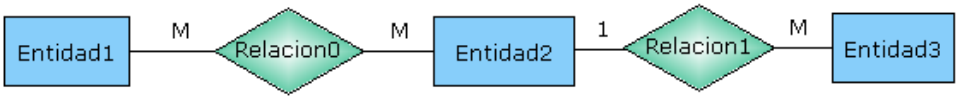

Ilustración 3: Ejemplo de un camino de navegación entre entidades.

Para este diagrama E/R es posible tener, según el diseñador de la base de datos del negocio, el siguiente esquema relacional asociado al camino de navegación. En él surgen nuevas relaciones ($R_{1,2}$) debido a la transformación del modelo conceptual al relacional explicado por Lucina (Hernández and Richardson, 2005):

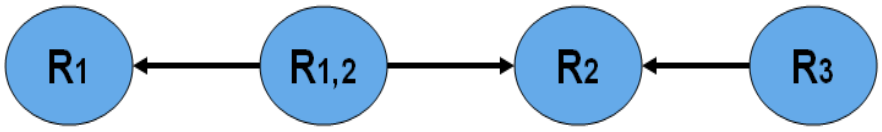

Ilustración 4: Digrafo relacional asociado a un camino de navegación.

En investigación previa (Alonso, 2008) se asociaban eventos de inserción, eliminación y actualización en un único nodo; este proceder era correcto para las limitadas posibilidades de navegación pero el desarrollo de estas hace necesario formalizar todas las variantes.

Pereira realiza un análisis sobre la inserción de nuevas tuplas (Toledo, 2009) y llega a la conclusión que deben ser colocados estos eventos en nodos donde solo se tengan aristas salientes, en nuestro ejemplo los nodos $R_{1,2}$ y R_3. Este resultado es debido a que nodos con estas características cumplen la propiedad que potencialmente cierran un camino de navegación, sin embargo debe asociarse este resultado a la garantía de integridad referencial en la base de datos del negocio. Pero este es un solo evento posible ¿qué sucede con los demás? Para ellos considérese que la base de datos del negocio fuerza la integridad referencial pues sin ella simplemente es necesario crear eventos de inserción, actualización y eliminación en todos los nodos del digrafo.

II.5.1. Evento de Actualización.

El tratamiento de este evento viene dado por la posibilidad de que determinada actualización en la base de datos del negocio modifique el resultado de un camino de navegación. Véase la siguiente regla en sus niveles iniciales de expresión y el esquema lógico para transplante renal en el Anexo1.

Lenguaje Seminatural:

RN#12: Un Cubículo no puede tener una cantidad de camas diferente a 8.

Lenguaje Técnico en LRT:

RN#12: Un Cubículo no puede tener CANTIDAD(sujeto.Objeto.tipo = ‘cama’) <> 8

Para esta regla, si se cambia el tipo de un objeto (teniendo sentido en el negocio) se tendría el riesgo de modificar la cantidad de camas en el cubículo, lo cual lleva a verificar el atributo final del último nodo de la navegación.

Algo similar se tiene al trasladar una cama de un cubículo a otro. Esta acción en el negocio lleva a realizar una modificación en la llave foránea del cubículo en el objeto y pone igualmente en riesgo la RN#12. Esto llama la atención sobre los vértices con aristas salientes.

Se tienen entonces dos casos a incluir en la lista de eventos:

- Evento de actualización en el último nodo con respecto al atributo del camino de navegación (si posee atributo).
- Evento de actualización en nodos con al menos una arista saliente.

Para los vértices con varias aristas salientes serían tantos eventos de actualización como aristas salientes posea. Para el ejemplo guía en el nodo $R_{1,2}$ dos eventos, asociados a R_1 y R_2 y dos eventos en R_3 uno asociado a R_2 y otro al atributo de la navegación.

II.5.2. Evento de Eliminación.

Igualmente el tratamiento de este evento viene dado por la posibilidad de que determinada eliminación modifique el resultado de un camino de navegación. Esto solo es posible cuando ocurre en nodos con todas las aristas salientes, excepto en el primero donde la relación asociada esté refiriéndose al *sujeto*. Por ejemplo de acuerdo con la RN#12 si se elimina un cubículo no se viola dicha regla pues la restricción es con respecto a ellos, otra cosa es si se elimina un objeto.

Se tiene entonces un caso para incluir eventos en la lista:

- Un evento de eliminación por cada nodo con solo aristas salientes excepto el nodo que esté asociado al *sujeto* de la regla.

En el ejemplo rector se incluirían dos eventos de eliminación en los nodos $R_{1,2}$ y R_3.

II.6. Proceso de integridad para R.N. tipo restricción.

Los tipos de regla no se consiguen comparar en cuanto a su importancia para el negocio, pero no cabe dudas que las reglas tipo restricción son las más complejas en cuanto a mantener su integridad en una base de datos relacional.

En el caso de las reglas tipo clasificación, cómputo y notificación es imposible su infracción directa, esto no sucede de igual modo con las restricciones. Se obliga por tanto a establecer un mecanismo para identificar y controlar determinadas acciones que incumplan restricciones establecidas en el negocio como muestran algunos autores (Heidenreich et al., 2005, Choi et al., 2006).

En estudios antecedentes (Alonso, 2008) se establece el siguiente proceso, el cual respondía a las exigencias hasta ese momento creadas.

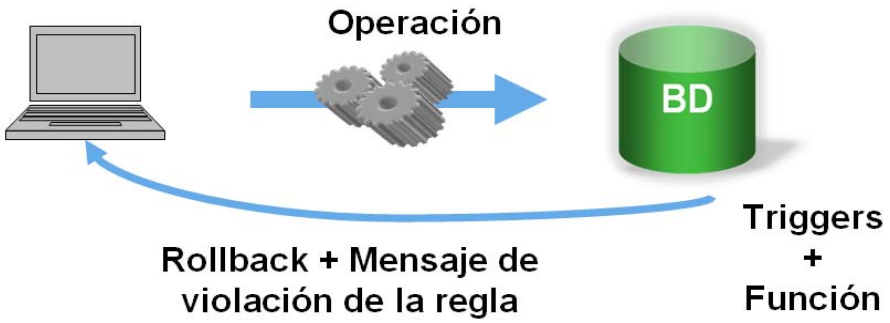

Ilustración 5: Previo proceso de integridad para R.N. tipo restricción.

En síntesis, una operación sobre la base de datos del negocio de actualización, inserción o eliminación la cual se considera que pudiese generar violaciones inicia una revisión de todas las reglas tipo restricción asociada a estos cambios. Esta revisión se realiza inmediatamente después de finalizada la operación; consiste en extraer las instancias de sujetos sospechosos mediante triggers y analizarlos utilizando una función específica

de la regla, si este <sujeto> viola la regla entonces se lanza un mensaje de error y se deshace la operación.

Este método de chequeo inmediato es brevemente abordado por Date (Date, 2001) al definir su Regla de Oro y en general pertenece a vías tradicionales de implementación de reglas de negocio (Bajec et al., 2000), esto trae consigo ciertos inconvenientes que deben ser superados.

Para la RN#12 descrita previamente simplemente no se puede quitar ni adicionar ninguna cama a un cubículo, dado que se desactualizarán la cantidad exacta de estas. Pero la realidad exige que estas operaciones sean permitidas bajo ciertas circunstancias, ¿que hacer si se necesitan reemplazar varias camas? En estos casos inmediatamente que se inserte o elimine, un mensaje de violación será levantado y la operación cancelada.

Cuando se buscan violaciones de reglas de negocio se necesita analizar más allá de una operación en particular y verlas como ilegalidades de un conjunto de operaciones (Date, 2001, Choi et al., 2006)

El nuevo proceso concebido para mantener la integridad de las reglas de negocio tiene este aspecto en cuenta. Su idea central es bien sencilla, para cada regla de negocio tipo restricción se tiene una vista de integridad que dice si esa regla ha sido infringida y que <sujetos> de la misma la han incumplido; para una serie de operaciones en la base de datos del negocio se extraen las reglas que han sido violadas de las posibles infractoras (Lista de Eventos) mediante las vistas de integridad y con estas se crea una Lista de Reglas Violadas (LRV). Este proceso es comentado por Zimbrão (Zimbrão et al., 2002) el cual lo aplica en su sistema Athena.

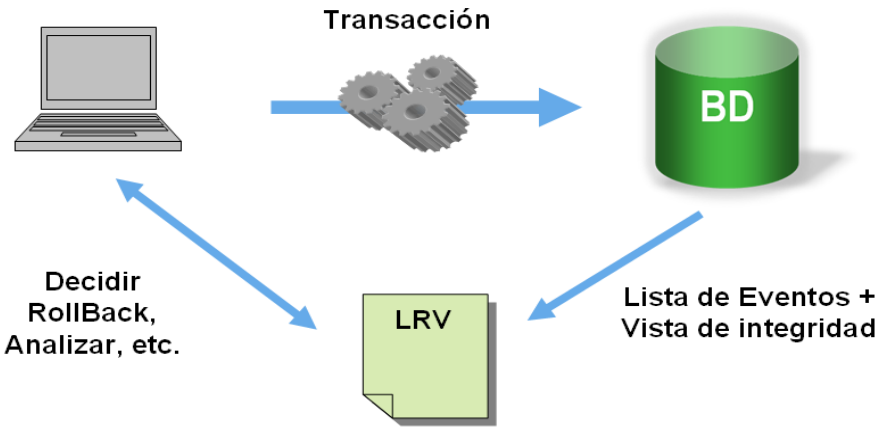

Ilustración 6: Proceso de integridad para R.N.

Con esta lista (LRV) el sistema del negocio puede decidir que hacer, si deshacer todas las operaciones, analizar las reglas que han sido violadas y de cada una cuales son los objetos causantes de la infracción. Con esta información se puede hacer un análisis profundo antes de tomar una medida, e incluso en casos extremos buscar más detalles. La decisión a tomar puede ser en caliente preguntándole directamente al cliente ¿qué desea hacer? o realizar acciones automáticas predefinidas.

En el ejemplo guía para remplazar las camas de un cubículo es permisible tener varias operaciones donde una elimine X número de objetos tipo cama (X <= 8) y otra inserte la misma cantidad X. Para el chequeo de estas operaciones la vista de integridad de la RN#12 no generará violación alguna de este cubículo, pues en el mismo seguirán existiendo 8 camas.

II.7. Arquitectura del SIGREN.

Primeramente el SIGREN es un sistema gerenciador de reglas de negocio implementado en investigación inicial (Alonso, 2008) y ampliado en posteriores (Lorenzo, 2009, Toledo, 2009) el cual se encuentra registrado apropiadamente (Tejada, 2009). La arquitectura de este sistema brinda una idea general sobre el ciclo de vida de las reglas de negocio en el presente trabajo.

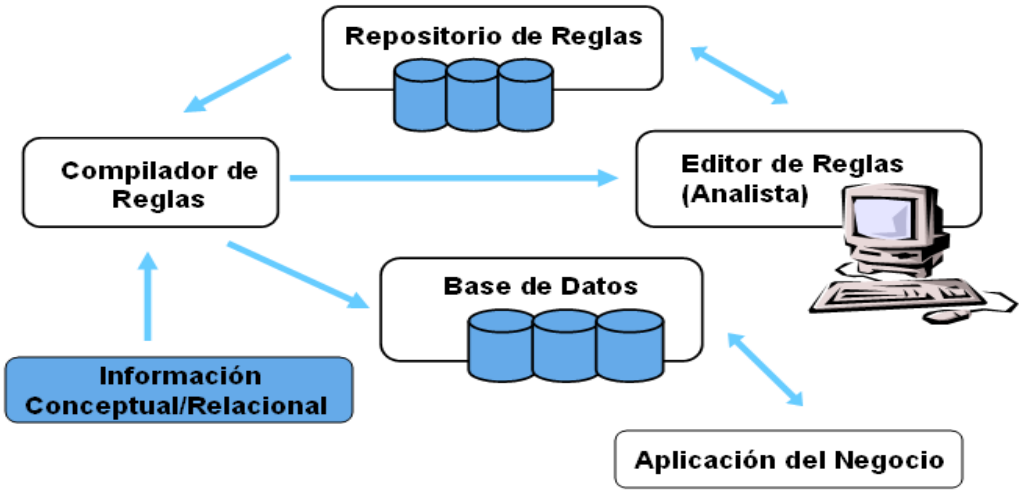

Ilustración 7: Arquitectura del SIGREN.

Las mismas comienzan su existencia en un editor de reglas el cual las capta en lenguaje técnico (LRT), por lo que solo pueden ser creadas por un analista del negocio. Para un sistema completo de reglas de negocio estas deben ser introducidas en lenguaje natural o seminatural (Business Rules Group, 2003) y luego traducidas al LRT.

Todo manejo de reglas de negocio debe ser almacenado en un repositorio, del cual son extraídas para su implementación en la base de datos. Esta transición es responsabilidad del compilador.

Para que este compilador sea capaz de generar las reglas en SQL para la base de datos del negocio, es necesario conocer la información relacional y conceptual de la misma lo cual será abordado en el próximo capítulo. El editor en este nivel técnico no forma parte de la aplicación del negocio dado principalmente por el abismo entre un analista (especializado con la herramienta, el negocio, el LRT, etc.) y un usuario del sistema. Para un editor desde un lenguaje natural este abismo no existe y la herramienta puede formar parte de la aplicación, siendo solo para personal autorizado con conocimientos del negocio.

II.8. Conclusiones Parciales.

En este capítulo se definen cuatro patrones los cuales permiten la creación de cuatro tipos diferentes de reglas de negocio, estas reglas son representadas utilizando un lenguaje propio formalizado en esta sección. Además se detallaron los eventos que activan la exploración de las mismas y se estableció un método para operarlas. Se presentaron posibles conflictos debido a la interdependencia de las reglas y por último se mostró la arquitectura de un sistema gerenciador de estas.

Se tiene con este capítulo la base teórica y los mecanismos necesarios para implementar y manejar reglas de negocio. Solo resta utilizar los múltiples recursos que ofrecen los gestores de bases de datos generalizados en el estándar SQL para obtener el último nivel de expresión.

Capítulo III:

IMPLEMENTACIÓN DE REGLAS DE NEGOCIO.

III. IMPLEMENTACIÓN DE REGLAS DE NEGOCIO.

El presente capítulo abordará todas las temáticas analizadas hasta el momento, con la intención de proponer una implementación de las mismas. Se examinará la información necesaria para este objetivo ya que es su punto de partida y consecuentemente utilizando el estándar SQL como lenguaje formal, se llevarán las reglas a su máxima expresión.

III.1. Información necesaria para implementar RN.

La implementación de reglas de negocio de esta investigación es realizada en el lenguaje relacional SQL con principios básicos de sus disímiles versiones. Es responsabilidad del motor de reglas de negocio compilar una regla en LRT la cual se conforma utilizando principalmente la notación punto.

Cuando mediante la notación punto se especifica Entidad1 **.** Entidad2 se está utilizando información conceptual del negocio y es necesario para lograr una implementación correcta tener a mano respuestas a varias interrogantes como:

- ¿A cuáles relaciones hacen referencia la Entidad1 y la Entidad2?
- ¿Qué cardinalidad posee la relación entre estas entidades?
- ¿Cuál regla de transformación se siguió para llegar al esquema relacional?

Por estas y otras cuestiones es necesario poseer información conceptual y relacional de la base de datos del negocio. Esta información puede obtenerse principalmente por dos vías: desde el catálogo de la base de datos o directamente de la herramienta de modelación.

III.1.1. Desde el catálogo de la base de datos.

Su necesidad es bien sencilla, para ello supóngase una empresa que posee de antemano un sistema de base de datos que satisface todos sus requerimientos, pero desea introducir las reglas de negocio al mismo conociendo sus beneficios.

En este caso es necesario considerar el riesgo de que no se cuente con el modelo con el cual fue diseñada la base de datos. Por tanto es preciso extraer esta información directamente de la base de datos mediante un único proceso de ingeniería inversa tal como muestra la siguiente figura:

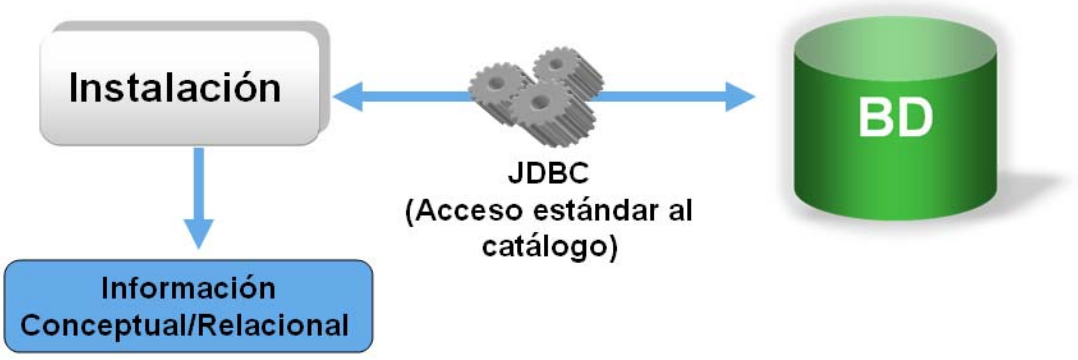

Ilustración 8: Información desde el catálogo de la base de datos.

Este proceso puede ser parte de la instalación del sistema de reglas pues ocurre en un único momento dentro del sistema del negocio. Una posible herramienta a utilizar, estable y sostenida, es Java Database Connectivity (JDBC) (Horstmann, 2009, Fisher et al., 2003) ya que brinda la posibilidad de aislarse del gestor de bases de datos. Dicho proceso debe ser interactivo con un usuario especializado del negocio pues se necesitan distinguir aspectos específicos del mismo.

III.1.2. Directamente de la herramienta de modelación.

Es el origen más utilizado en la literatura (Ross and Lam, OMG, 2010, Tedjasukmana, 2006),se puede destacar no la herramienta, sino el paquete de herramientas desarrollado por la Universidad de Dresden, denominado OCL Toolkit (Demuth, 2004) el cual utiliza un modelo en formato XMI generado por alguna herramienta UML como ArgoUML, Poseidon, Rational Rose entre otras (Demuth, 2005).

Este origen de hecho tiene en su concepción información conceptual y relacional, pues ya un diagrama ER es conceptual y para generar un diseño lógico y finalmente físico la herramienta posee sus propias reglas de transformación. En la siguiente figura se muestra el ciclo de vida de la modelación de datos propuesto por Umanath (Umanath and Scamell, 2007):

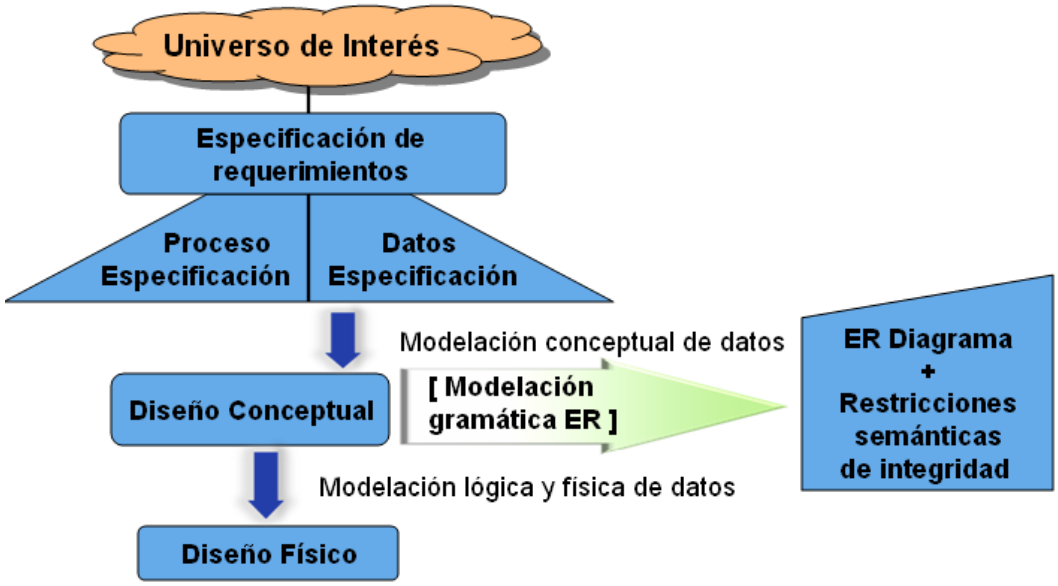

Ilustración 9: Modelación de bases de datos, ciclo de vida.

Es interesante retomar que las reglas de negocio tienen sentido no solo en la vida útil del sistema al cual se integran sino que desde la modelación de la base de datos comienzan a solucionar requerimientos del negocio. Son conocidas las limitantes de un diagrama ER para incluir restricciones

semánticas (Khan et al., 2002) por lo cual un modelo ER se conforma del diagrama ER más estas restricciones semánticas (Umanath and Scamell, 2007).

Véase las semejanzas entre la utilización de UML con OCL y una posible utilización de diagramas ER más reglas de negocio.

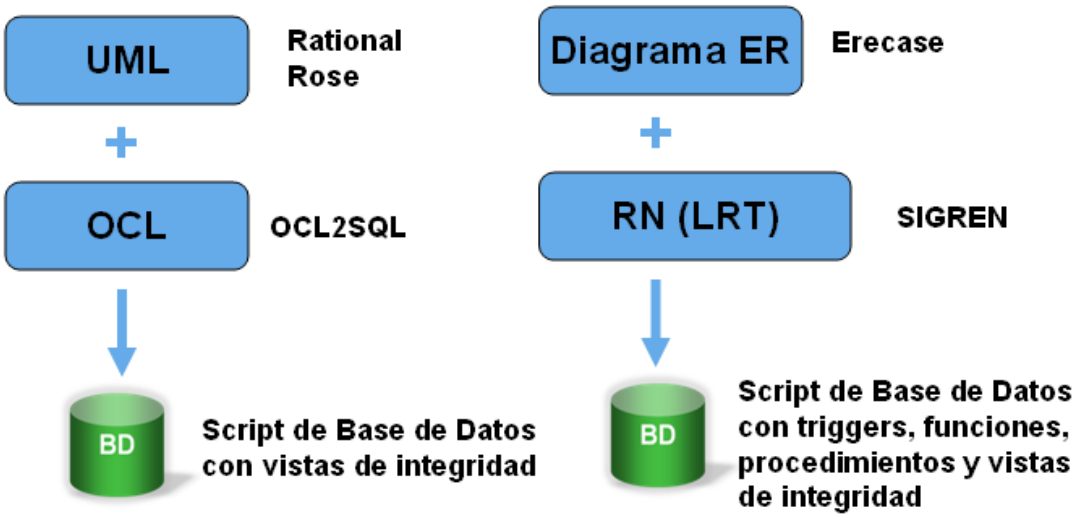

Ilustración 10: Semejanzas entre UML/OCL y diagramas ER/ Reglas de Negocio.

A la izquierda un ejemplo de UML/OCL con el Rational Rose y la herramienta ocl22sql del paquete de Dresden OCL Toolkit, a la derecha en paralelo el diagrama ER con el Erecase (Álvarez et al., 2006) y los patrones de reglas con el motor de reglas del SIREN. Ambos llegan a generar mediante un Lenguaje de Definición de Datos (LDD) (Ullman, 1982) el script de la base de datos del negocio y las vistas de integridad. En el caso particular de esta investigación se puede obtener además la lista de eventos y otros recursos utilizados en la implementación las reglas.

III.2. Implementación del LRT.

Un lenguaje técnico potente, sencillo y semejante al lenguaje natural es imprescindible para la presente investigación. Estas características están sujetas siempre a la posibilidad real que exista de implementarlas, pues no es objetivo llegar a un lenguaje consolidado como OCL complejo en su

estructura y totalmente separado de un lenguaje natural. Por tanto se mejora poco a poco el lenguaje LRT manteniendo un equilibrio entre la complejidad y la naturalidad del lenguaje, dos aspectos diametralmente opuestos.

III.2.1. La Notación Punto.

Como se ha dicho la notación punto es el núcleo del LRT, mediante este estilo es posible navegar entre entidades de forma sencilla y natural. Solo resta su implementación.

La implementación de la notación punto consiste en realizar asociaciones según se realice el camino de navegación. La consumación de cada camino debe considerarse como una serie de acoples entre las relaciones que estén en el mismo. Para el siguiente ejemplo obsérvese la solución según las notaciones de Lucina (Hernández and Richardson, 2005) utilizando el algebra relacional:

Paciente.DonantesPotenciales.ExamenFísico.Resultado

R1 = JOIN (Π idPaciente(Paciente), DonantesPotenciales)
R2 = JOIN (Π idDonantesPotenciales(R1), ExamenFísico)
R3 = Π Resultado(R2)

Implementar estos acoples en una consulta SQL estándar puede ser realizado de dos formas diferentes, en la cláusula WHERE o en el FROM de una consulta (Gennick, 2006, Oppel and Sheldon, 2008). El presente trabajo realiza las reuniones mediante el WHERE como consecuencia de la investigación inicial (Alonso, 2008) y del compilador creado por esta.

III.2.2. Operadores lógicos y aritméticos.

Para implementar los operadores del LRT se utilizan los principales operadores definidos para el estándar SQL (Melton and Simon, 2002, Oppel and Sheldon, 2008). En la siguiente tabla se muestran los operadores lógicos definidos en LRT, su formalización en el estándar SQL y sus homólogos en OCL.

LRT	estándar SQL	OCL
O	OR	OR
Y	AND	AND
NEG	NOT	NOT

Tabla 10: Implementación de operadores lógicos

La utilización de estos operadores se puede observar en parte de la implementación a las consultas de la RN#9 definida previamente:

Lenguaje Formal:

```
(SELECT COUNT (*)
FROM  Paciente a, ExamenFísico b   WHERE
/*Paciente=sujeto*/(a.idPaciente = b.idPaciente )) < 10
AND       NOT ( EXISTS( SELECT  *
          FROM Paciente a, DonantesPotenciales b
WHERE /*Paciente=sujeto*/(a.idPaciente = b.idPaciente )))
```

La utilización de operadores lógicos produce generalmente, como muestra el ejemplo, una consulta en parte izquierda y otra en parte derecha. Esto pudiera tener excepciones, al referenciar un mismo camino de navegación o comparaciones con constantes se pudiera realizar ambas partes del operador en una única consulta.

Lenguaje Seminatural:

RN#15: Un Donante Potencial no puede tener una Evolución con Temperatura mayor de 37ºC ni un pulso menor a 40.

Lenguaje Técnico en LRT:

RN#15: Un DonantesPotenciales no puede tener sujeto.Evolución.Temperatura > 37 ***Y*** sujeto.Evolución.Pulso < 40

Lenguaje Formal:

```
SELECT *
  FROM DonantesPotenciales a, Evolución b
  WHERE /* Donantes Potenciales = sujeto */
(a.idDonantesPotenciales = b. idDonantesPotenciales)  and
(b.Temperatura > 37) AND (b.Pulso < 40)
```

Siguiendo la variante original se obtendría la siguiente consulta:

```
SELECT *
   FROM DonantesPotenciales a, Evolución b
   WHERE /* Donantes Potenciales = sujeto */
     (a.idDonantesPotenciales=b.idDonantesPotenciales)
     and (b.Temperatura > 37)
AND
SELECT *
  FROM DonantesPotenciales a, Evolución b
  WHERE /* Donantes Potenciales = sujeto */
     (a.idDonantesPotenciales=b.idDonantesPotenciales)
     and (b.Pulso > 37)
```

Otros operadores son los aritméticos. Estos son los mismos en el lenguaje de reglas de negocio LRT, en el SQL estándar y en lenguaje de restricción de objetos OCL. Deben siempre aplicarse entre elementos individuales solo de tipos numéricos. Para el LRT se han definido cuatro tipos de datos fundamentales:

Tipo	Valores
Lógico	True, False
Cadena	“Esto es una cadena”
Entero	-2, -1, 0, 1, 2, 3…
Real	-0.5, 0.25, 0.5, 0.75, 1.25

Tabla 11: Tipos de datos del LRT.

III.2.3. Operadores de relación y conjunto.

En el capítulo anterior se plantearon, con los operadores de relación, cuatro posibles variantes de utilización. Para cada una de ellas se señala a continuación, su implementación.

(Elemento Individual) operador de relación (Elemento Individual):
En la RN#7 se encuentra la comparación de cantidad de pacientes con la constante 800. Véase su lenguaje técnico.

Lenguaje Técnico en LRT:

RN#7: Notificar “Alerta Roja de la Pandemia” si
CANTIDAD(Pacientes.ExamenesFísicos.Resultado = ‘AH1N1’) **>** 800

Los operadores de conjunto devuelven elementos individuales por lo que una sección de la implementación de esta regla será:

```
(SELECT COUNT(*) ...) > 800
```

Como se observa en la implementación es colocado el operador entre el miembro izquierdo y el derecho.

(Elemento Múltiple) operador de relación (Elemento Individual):
Previamente se refería a la RN#5 para ejemplificar esta segunda variante. Véase su lenguaje técnico.

Lenguaje Técnico en LRT:

RN#5: La Cantidad_Temperatura_Alta_Paciente es calculado como CANTIDAD(Paciente.Evoluciones.Temperatura **>** 38)

Al comparar Paciente.Evoluciones.Temperatura con la constante 38 se está restringiendo la colección de elementos a la izquierda del operador. Su implementación conlleva a introducir esta comparación dentro de la consulta generada por el camino de navegación, a continuación una sección de su implementación.

```
SELECT  COUNT(*)
FROM Paciente a, Evolución b
WHERE /* Paciente = sujeto */
(a.idPaciente = b. idPaciente) and (b.Temperatura > 38);
```

(Elemento Múltiple 1) operador de relación (Elemento Múltiple 2):

Este es un caso interesante, una posible variante sería utilizar el operador ALL del SQL estándar en una subconsulta pero se puede utilizar un artificio para caer en la segunda variante de los operadores de relación. La solución la provee Umanath (Umanath and Scamell, 2007) donde aclara que es posible reemplazar el >ALL(colección) por >MAX(colección) y el <ALL(colección) por <MIN(colección), las restantes variantes son obtenidas

bajo igual idea excepto el =. Al utilizar operadores de conjuntos se obtienen como resultado elementos individuales, posteriormente se mostrará un ejemplo de su manejo.

Operadores LRT	SQL estándar
CANTIDAD	COUNT
VACÍO	EXISTS
EXISTE(<elemento>, <colección>)	IN
PROMEDIO	AVG
SUMA	SUM
MÍNIMO	MIN
MÁXIMO	MAX
PROMDIF	AVG DISTINCT
CANTDIF	COUNT DISTINCT
SUMADIF	SUM DISTINCT

Tabla 12: Implementación para operadores de conjuntos

III.3. Implementación de Patrones.

Con los operadores definidos en el LRT solo resta confrontar la implementación de los patrones para reglas de negocio.

III.3.1. Patrón de Clasificación.

El comportamiento de este patrón es sencillo, como se ha abordado previamente su objetivo es, dado un <sujeto>, decir si pertenece a no a determinada clasificación. Para implementar este patrón es posible utilizar un recurso otorgado por el estándar SQL, el cual se encuentra en la mayoría de gestores de base de datos: las funciones. Estas cumplen con las características ideales para este patrón pues el mismo exige retornar

verdadero en caso que el <sujeto> encuestado pertenezca a la clasificación definida y falso en caso contrario. Para la RN#2:

Lenguaje Técnico en LRT:

RN#2: Una Evolución es definido como Evolución_Satisfactoria si sujeto.Temperatura < 37

Lenguaje Formal:

```
CREATE FUNCTION Evolución_Satisfactoria (sujeto1:INTEGER)
RETURNS BOOLEAN
RETURN ( SELECT Temperatura
          FROM Evolución a
          WHERE (sujeto1 = a.idEvolución )) < 37);
```

Esta función recibe un parámetro, el cual es interpretado como el <sujeto> específico que se quiere clasificar y utilizado para realizar el acople en la cláusula WHERE. Para este ejemplo en particular solo se necesita un único parámetro, pero su cantidad depende de los atributos que formen parte de la llave primaria y pueden seguirse definiendo como sujeto2, sujeto3,…, sujeto(n).

En la implementación de la RN#3 se puede encontrar un ejemplo de la utilización del operador MAX() sustituyendo al ALL().

Lenguaje Técnico en LRT:

RN#3: Un Cirujano es definido como Cirujano_Maestro si MÁXIMO (sujeto.DonanteVivo.Evolución.Temperatura) < 37

Lenguaje Formal:

```
CREATE FUNCTION Cirujano_Maestro (sujeto1: INTEGER)
RETURNS BOOLEAN
RETURN ((SELECT MAX(Temperatura)
                    FROM DonanteVivo a, Evolución b
                     WHERE (sujeto1 = a.idCirujano) and
(a.idDonantesPotenciales=b.idDonantesPotenciales)) < 37);
```

III.3.2. Patrón de Cómputo.

Las reglas que define este patrón tienen mucho en común con las reglas de clasificación, de hecho ambas son reglas estructurales. Esta afinidad conlleva a implementaciones similares.

El patrón de cómputo puede ser implementado utilizando igualmente las funciones que define el estándar SQL. Esta siempre devuelve un valor numérico que puede ser entero o real. Véase para la RN#5 su realización:

Lenguaje Formal:

```
CREATE FUNCTION Cantidad_Temperatura_Alta_Paciente
RETURNS INTEGER
RETURN (SELECT COUNT(*)
        FROM Paciente a, Evolución b  WHERE
(a.idPaciente=b.idPaciente) and (b.Temperatura < 37));
```

Es posible emplear los conceptos definidos mediante reglas de negocio en otras reglas. El único caso de dependencia para los patrones de cómputo mostrado en la Tabla 9 son ellos mismos. La RN#4 proporciona este ejemplo.

Lenguaje Técnico en LRT:

RN#4: El Porciento_Temperatura_Alta en Paciente es calculado como CANTIDAD(sujeto.Evolución.Temperatura > 38) * 100 / Cantidad_Temperatura_Alta_Paciente

Lenguaje Formal:

```
CREATE FUNCTION
Porciento_Temperatura_Alta(sujeto1:INTEGER)
RETURNS FLOAT
RETURN (SELECT COUNT(*)
         FROM Paciente a, Evolución b
          WHERE (sujeto1 = a.idPaciente) and
(a.idPaciente = b.idPaciente) and (b.Temperatura > 38)) *
100 / Cantidad_Temperatura_Alta_Paciente);
```

III.3.3. Patrón de Notificación.

El objetivo de este patrón es simple pues se basa en informar a los usuarios autorizados del negocio sobre algún comportamiento ocurrido. Para ello es necesario definir cómo se pretende alertar al usuario cuando determinados <hechos> son satisfechos. A tratar en esta investigación, dos vías de notificación, la primera un mensaje directo al sistema y la segunda un almacenamiento de todas las informaciones brindadas por este tipo de regla.

Ambas soluciones a este patrón tienen en común que se debe verificar si los <hechos> son satisfechos, lo cual puede ser realizado tomando un recurso brindado por el estándar SQL: los procedimientos. Véase la implementación para la RN#7.

```
CREATE PROCEDURE PRN#7
BEGIN
IF (SELECT COUNT(*)
      FROM Paciente a, ExamenFísico b
      WHERE (a.idPaciente = b. idPaciente) and
(b.Resultado = ´AH1N1´) > 800)
THEN  /* NOTIFICACIÓN  */
END;
```

La utilización de la cláusula IF es necesaria para evaluar si los <hechos> se cumplen o no. En caso afirmativo la notificación es realizada. Estos procedimientos son invocados por los eventos, los cuales se relatarán posteriormente.

Para notificar directamente al usuario solo es necesario definir un mensaje y levantarlo como una excepción.

```
DECLARE messg CONDITION FOR SQLSTATE
 'Alerta Roja de la Pandemia';
Signal messg;
```

Un efecto directo de esta variante son los constantes mensajes de notificación que pueden llegar a ser molestos y difíciles de manejar, suponga que se tuvieran más de 800 casos de AH1N1 y continuamente se necesitase lidiar con la alerta de la pandemia.

Por ello se busca almacenar estas informaciones para su posterior tratamiento. En este caso es necesario no solo guardar el mensaje sino además el nombre de la regla y el momento en que se notificó, para que luego el sistema pueda realizar un análisis correcto.

Para recopilar esta información es necesario tener en la base de datos del negocio una tabla que pueda ser posteriormente consultada. Esta se encontrará constantemente recibiendo las notificaciones de las políticas del negocio por lo que su cardinalidad puede aumentar vertiginosamente, para que esto no ocurra debe ser limpiada cada determinado período de tiempo, definido este por los requisitos del negocio. Para una idea clara de la misma se presenta su definición utilizando el LDD (Oppel and Sheldon, 2008).

```
CREATE TABLE NOTIFICACIONES(
          NOMBRE_RN CHAR (20) NOT NULL,
          FECHA DATETIME NOT NULL DEFAULT CURRENT_DATE,
          MENSAJE CHAR (100) NOT NULL,
          PRIMARY KEY ( NOMBRE_RN ,FECHA ) );
```

Cada vez que se produzca un estado en el negocio que deba ser notificado simplemente se inserta en la base de datos mediante la cláusula INSERT del estándar SQL. Para la sección de código restante de la RN#7 siguiendo esta filosofía se tiene:

```
INSERT INTO NOTIFICACIONES (NOMBRE_RN, MENSAJE)
VALUES ('RN#7','Alerta Roja de la Pandemia');
```

III.3.4. Patrón de Restricción.

Para este patrón es necesario como se ha expresado previamente implementar vistas de integridad que muestren todos los <sujetos> que han violado determinada restricción. En propuestas anteriores estas reglas se implementaban mediante una función (Alonso, 2008) la cual recibía como parámetro un único <sujeto> extraído desde el evento en tiempo real. Esta aunque insuficiente muestra un punto interesante al extraer los posibles sujetos infractores y por tanto ahorrarse la encuesta a todos los existentes.

En este patrón todo gira alrededor del <sujeto> de la regla, a este se le restringen ciertas <características> a excepción de determinados <hechos> y en los caminos de navegación que se definen es muy utilizado mediante la palabra reservada *sujeto*. Es por ello que su implementación emplea un valioso recurso del estándar SQL: las vistas. Un ejemplo mediante la RN#8:

Lenguaje Técnico en LRT:

RN#8: Un Cirujano no puede tener CANTIDAD(sujeto.DonanteVivo) > CANTIDAD(DonanteVivo)/6.

Lenguaje Formal:

```
CREATE VIEW VRN#8 AS
(SELECT * FROM Cirujano sujeto WHERE
(SELECT COUNT(*) FROM Cirujano a, DonanteVivo b
WHERE (sujeto.idCirujano = a.idCirujano) and
(a.idCirujano = b.idCirujano)) >
((SELECT COUNT(*) FROM DonanteVivo) / 6));
```

Estas vistas de integridad son realizadas sobre el <sujeto> mediante una consulta externa que contiene subconsultas generadas por la notación punto, junto a la implementación de los operadores. La utilización de vistas para realizar reglas de negocio es defendida por varios autores (Zimbrão et al., 2002, Tedjasukmana, 2006) y específicamente utilizada por la herramienta OCL22SQL de Dresden OCL toolkit (Heidenreich et al., 2005).

En el caso que la regla incluya <hechos> se tiene un caso particular que puede, bajo una mirada lógica, implementarse mediante un simple WHERE. Este patrón se definió como:

<determinante> <sujeto> puede tener <características> solo si <hechos>.

Aquí se emplea un elemento de la lógica booleana muy importante: la implicación. Algunos lenguajes naturales para describir P → Q son:

- Si P entonces Q.
- P es suficiente para Q.
- Q es necesario para P.
- P solo si Q.

Como dice Haaparanta (Haaparanta, 2009) P→ Q es equivalente a expresar ¬ P v Q por tanto la vista tendría la siguiente estructura:

```
CREATE VIEW VRN#? AS
SELECT * FROM TablaX sujeto WHERE
NOT ( /* Implementación de las <características> */) OR
(( /* Implementación de los <hechos> */);
```

III.4. Implementación de la Lista de Eventos.

La necesidad de esta lista responde básicamente a la búsqueda de estados específicos del negocio. Estos estados varían en la base de datos únicamente cuando ocurren operaciones básicas de inserción, actualización o modificación en las tablas asociadas al negocio. Existe un recurso disponible en la mayoría de los gestores de bases de datos y especificado en el estándar SQL que puede monitorear los cambios en el negocio desde cada una de sus tablas, este recurso son los triggers (Choi et al., 2006).

En el Capítulo II se analizó según un camino de navegación en cuales tablas deben ser creados estos triggers. Solo queda por definir la estructura de los mismos, para ello se muestran los triggers asociados a los eventos de inserción y eliminación, los cuales solo difieren en la cláusula AFTER.

```
CREATE TRIGGER T[I | D]RN
AFTER [INSERT | DELETE] ON Tabla
BEGIN ATOMIC
    /* Acción */
END;
```

Es de destacar el nombre del trigger, el cual para cada tabla puede tomar el valor de TIRN (trigger de inserción para reglas de negocio) o TDRN (trigger de eliminación para reglas de negocio). No existe la necesidad de crear más triggers para un evento sobre la misma tabla debido a que varias reglas que utilicen este evento pueden ser agrupadas. Así se evitaría la generación innecesaria de código, apostando por el rendimiento del sistema, aún cuando el mantenimiento de las reglas gane en complejidad.

A continuación se obtienen los siguientes triggers para inserción y eliminación desde un camino de navegación (donde el sujeto hace referencia a un cirujano) de una regla hipotética (RN#?), dependiente a los cambios en la base de datos del negocio.

sujeto.DonanteVivo.Evolución.Temperatura

```
CREATE TRIGGER TIRN
AFTER INSERT ON Evolución
BEGIN ATOMIC
    /* Acción */
END;

CREATE TRIGGER TDRN
AFTER DELETE ON Evolución
BEGIN ATOMIC
    /* Acción */
END;
```

En el caso de los eventos de actualización estos no tienen por qué estar incorporados a la tabla completamente. Su objetivo es chequear dos casos fundamentales: la actualización de las llaves foráneas entre las tablas asociadas al camino de navegación y la actualización del atributo o atributos de esta última. La estructura de los mismos:

```
CREATE TRIGGER TURN
AFTER UPDATE [ OF <lista de columnas del trigger > ]
               ON Tabla
BEGIN ATOMIC
  /* Acción */
END;
```

En el ejemplo tratado de la RN#3 se tiene:

```
CREATE TRIGGER TURN
AFTER UPDATE OF idCirujano ON DonanteVivo
BEGIN ATOMIC
  /* Acción */
END;

CREATE TRIGGER TURN
AFTER UPDATE OF idDonantesPotenciales, Temperatura ON
DonanteVivo
BEGIN ATOMIC
  /* Acción */
END;
```

Hasta este punto ambas reglas, las de tipo notificación y restricción comparten igual comportamiento y código, pero la implementación de sus

patrones rompe con esta semejanza. Esta diferencia queda registrada en las acciones a tomar por los eventos que genéricamente han sido tratadas como `/* Acción */`.

Para las reglas tipo notificación, mediante un procedimiento se define como transmitir el mensaje deseado al cliente, por lo tanto todos los eventos asociados a este patrón deben invocarlo. Para ello se emplea la rutina de invocación SQL EXEC tratada por Melton (Melton and Simon, 2002), en nuestro ejemplo hipotético se obtiene:

```
EXEC SQL CALL PRN#?();
```

Para las reglas de negocio tipo restricción se asumen consideraciones especiales. Las mismas son chequeadas luego de ocurrir varias operaciones lo que las aleja del tratamiento dado a las de notificación. Cada evento de la lista almacena en un único lugar las reglas que han de ser examinadas para posteriormente inspeccionarlas.

Para ello es necesario definir un objeto persistente donde se almacene los nombres de las reglas a ser exploradas:

```
CREATE TABLE RN_RESTRICCIÓN(
        NOMBRE_RN VARCHAR NOT NULL,
        PRIMARY KEY ( NOMBRE_RN ));
```

Solo se debe tener cuidado al insertar reglas en dicha tabla pues una misma regla podría tener más de un evento que solicitara su comprobación, lo cual obliga a verificar su existencia en la misma. Para el ejemplo tratado se tiene:

```
IF NOT EXISTS(SELECT *
FROM RN_RESTRICCIÓN
WHERE NOMBRE_RN = 'RN#?')
THEN INSERT INTO NOTIFICACIONES (NOMBRE_RN)
VALUES ('RN#?');
```

III.5. Proceso de integridad para R.N. tipo restricción.

La integridad en una base de datos relacional en cuanto a las reglas de negocio no es más que el cumplimiento total de las disposiciones que estas exijan. No todos los patrones especificados exigen este cumplimiento en el negocio, solo el de tipo restricción puede dejar el estado del negocio inconsistente respecto a determinada política prohibitoria.

Estas políticas prohibitorias deben ser asociadas a un conjunto de operaciones y no a una única operación como tradicionalmente se llevaba a cabo (Choi et al., 2006). La base esencial para implementar este proceso ha de ser otro de los recursos que brindan la mayoría de los gestores, definido en el estándar SQL: las transacciones (Date, 2001).

Por tanto cada operación del sistema en la base de datos del negocio debe realizarse mediante un procedimiento, el cual se encarga de ejecutar esta transacción y determinar si puede ser aceptada o no. Esto trae consigo que el programador del sistema solo utilice estos procedimientos para realizar todas las operaciones sobre la base de datos del negocio. Este procedimiento consta de cinco segmentos fundamentales:

- Garantizar el correcto inicio del proceso de integridad en la transacción.
- Realizar las operaciones necesarias del usuario.
- Verificar las violaciones de las operaciones previamente finalizadas.

- ❖ Análisis y decisión final sobre las infracciones cometidas.
- ❖ Mostrar opcionalmente la información almacenada de dicha transacción.

```
CREATE PROCEDURE PTRANSAC#X ()
BEGIN
     /* Primer segmento */
     /* Segundo segmento */
     /* Tercer segmento */
     /* Cuarto segmento */
     /* Quinto segmento */
END;
```

Para garantizar el correcto funcionamiento del proceso de integridad de la transacción es necesario primeramente eliminar todas las instancias previas de la tabla RN_RESTRICCIÓN. Aquí se almacena todas las reglas que necesitan ser exploradas luego de realizadas las operaciones. También es imprescindible comenzar la transacción de la que luego se toman determinaciones (Gennick, 2006). Para ello la siguiente sección de código en SQL estándar.

```
DELETE FROM RN_RESTRICCIÓN;
START TRANSACTION;
```

Luego de definir el primer segmento del procedimiento se pasa a ejecutar la acción o el conjunto de acciones deseados por el usuario. Estas acciones pueden ser de inserción, eliminación o actualización y conllevar a que ciertos eventos cuestionen varias reglas del negocio. Esto trae como consecuencia que se deba verificar posibles violaciones cometidas por las operaciones previamente finalizadas.

Para el caso de las reglas tipo restricción las posibles infractoras asociadas a las operaciones del procedimiento quedan almacenadas en la tabla RN_RESTRICCIÓN. Debido a cierto grado de complejidad que se reduce a consultas dinámicas y uso de cursores se presenta el siguiente segmento en un dialecto del SQL estándar (Transac-SQL) el cual puede ser ejecutado en el Query Analyzer de SQL SERVER 5.0 (MSDN, 2008). Para todas las reglas que se encuentran en RN_RESTRICCIÓN debe chequearse su vista de integridad y en caso de violación debe almacenarse y guardarse la cantidad de <sujetos> que la infringen. Para ello se define la siguiente tabla:

```
CREATE TABLE LRV(
      NOMBRE_RN VARCHAR NOT NULL,
           CANT_SUJETOS INTEGER,
      PRIMARY KEY ( NOMBRE_RN ));
```

Para cada regla a chequear se evalúa su vista de integridad correspondiente, el nombre de esta última puede ser construido con el nombre de la regla, lo cual es realizado en tiempo de ejecución mediante consultas dinámicas. Luego para cada vista solo es necesario verificar la cantidad de <sujetos> que retornan, y con el nombre de la regla van poblando la Lista de Reglas Violadas.

```
DECLARE @Nombre_RN NVARCHAR (500)
DECLARE Rules_cursor CURSOR FOR SELECT NOMBRE_RN FROM
RN_RESTRICCIÓN
OPEN Rules_cursor
FETCH NEXT FROM Rules_cursor INTO @Nombre_RN
WHILE @@FETCH_STATUS = 0
BEGIN
DECLARE @SQLString NVARCHAR(500)
DECLARE @ParmDefinition NVARCHAR(500)
```

```
SET @SQLString =N'
          DECLARE @Cant_Sujetos INTEGER
          SET @Cant_Sujetos = (SELECT COUNT(*) FROM V'+
     @Nombre_RN +') IF (@Cant_Sujetos <> 0)
                         INSERT INTO LRV VALUES (''' +
          @Nombre_RN + ''', @Cant_Sujetos)'
EXECUTE sp_executesql @SQLString
FETCH NEXT FROM Rules_cursor INTO @Nombre_RN
END
CLOSE Rules_cursor
DEALLOCATE Rules_cursor
```

Como resultado de este segmento se obtiene qué reglas y cuantos <sujetos> de ella violan los requerimientos del negocio. Con estos datos junto a las vistas previamente creadas el sistema puede analizar con bastante profundidad las infracciones cometidas y tomar una decisión. Esta disposición podría incluso incluir al cliente, pues para ciertos casos complejos este puede tomar partido aunque lo usual debe ser establecer medidas automáticas.

La naturaleza de las reglas tipo restricción es prohibir estados en el negocio por lo que la generalidad de las decisiones han de impedir cualquier tipo de infracción y la información de estas ser utilizada como una ventaja del proceso. Esta mayoría ha de ser realizada mediante el siguiente código.

```
IF EXIST(SELECT * FROM LRV)
THEN ROLLBACK
COMMIT
```

Como colofón final se tiene la opción de mostrarle al cliente las notificaciones que dichas operaciones produjeron en el negocio o algún análisis de las

mismas. Esto es posible solo si se almacena su mensaje en la tabla NOTIFICACIONES, para ello se debe conservar el tiempo de inicio de la transacción y mostrar toda la información posterior a ese momento.

III.6. Conclusiones Parciales.

De este modo culmina el capítulo, definiendo la forma de implementar cada patrón de reglas de negocio y su lenguaje técnico. Se mostraron además una serie de ejemplos para la creación de la lista de eventos y para expresar el proceso de integridad de las reglas. Estos ejemplos en su gran mayoría fueron concebidos utilizando el estándar SQL, de esta forma su traducción a algún dialecto en específico resulta una tarea sencilla y viable. Este conocimiento es la base necesaria para desarrollar un motor de reglas a partir del lenguaje LRT.

Conclusiones.

Para implementar reglas de negocio en bases de datos relacionales:

- Se definieron cuatro patrones de reglas para captar los requerimientos del negocio.

- Se creó un lenguaje técnico para expresar las reglas de estos patrones.

- Se implementaron mediante recursos estándares de bases de datos las reglas de los patrones en su lenguaje técnico.

- Se propuso un método de manejo para las reglas implementadas.

Recomendaciones.

Para investigaciones y trabajos futuros se recomienda:

- Actualizar el motor de reglas del SIGREN a partir del Lenguaje para Reglas Técnicas.
- Implantar la nueva versión del SIGREN en varios negocios a fin de buscar insuficiencias.
- Ampliar los patrones de reglas de negocio bajo nuevos tipos de requerimientos.
- Investigar las carencias del LRT para expresar reglas en lenguaje natural.
- Investigar las eliminaciones de reglas de negocio interdependientes.

Bibliografía.

ALONSO, A. P. (2008) Aplicación para reglas de restricción en negocios. *Departamento de Bases de Datos.* Santa Clara, Universidad Central de Las Villas.

ÁLVAREZ, W., RODRÍGUEZ, A. & GARCÍA, C. (2006) ERECASE v.2.0 Una herramienta para el diseño conceptual de bases de datos con validación estructural. *Departamento de Computación.* Santa Clara, Universidad Central de Las Villas.

ASHWELL, R. (2006) Define Business Rules. CRaG Systems.

BAJEC, M., RUPNIK, R. & KRISPER, M. (2000) USING BUSINESS RULES TECHNOLOGIES TO BRIDGE THE GAP BETWEEN BUSINESS AND BUSINESS APPLICATIONS. Ljubljana, Slovenija.

BESEMBEL, I. M. & CHACÓN, E. (2001) Objetos y reglas de negocios en la integración y automatización de procesos de producción continua.

BRUEGGE, B. & DUTOIT, A. H. (2009) *Object Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall.

BUSINESS RULES GROUP (2003) Manifiesto de Reglas de Negocio. IN ROSS, R. G. (Ed., Business Rules Group.

CHOI, E.-H., TSUCHIYA, T. & KIKONU, T. (2006) Model Checking Active Database Rules Nakouji, Amagasaki, Hyogo, Research Center for Verification and Semantics (CVS) National Institute of Advanced Industrial Science and Technology (AIST)

DATE, C. J. (2001) Constraints & Predicates: A Brief Tutorial (Part 2). *Business Rules Journal,* V2.

DEMUTH, B. (2004) THE DRESDEN OCL TOOLKIT AND ITS ROLE IN INFORMATION SYSTEMS DEVELOPMENT.

DEMUTH, B. (2005) The Dresden OCL Toolkit and the Business Rules Approach. Technische Universität Dresden.

FISHER, M., ELLIS, J. & BRUCE, J. (2003) JDBC™ API Tutorial and Reference, Third Edition. Addison Wesley.

GENNICK, J. (2006) *SQL pocket guide*, O'Reilly Media.

HAAPARANTA, L. (2009) *The Develepment of Modern Logic*, Oxford, Oxford University Press US.

HEIDENREICH, F., WENDE, C. & DEMUTH, B. (2005) A Framework for Generating Query Language Code from OCL Invariants.

HERNÁNDEZ, L. G. & RICHARDSON, M. M. D. O. (2005) *Sistema de Base de Datos*, La Habana, Editorial Félix Varela.

HORSTMANN, C. S. (2009) *Big Java: Compatible with Java 5, 6 and 7*, John Wiley and Sons.

KHAN, K. M., KAPURUBANDARA, M. & CHADHA, U. (2002) Incorporating Business Requirements and Constraints in Database Conceptual Models. Sydney.

LORENZO, I. M. (2009) Modificación de las reglas de negocio tipo restricción y su implementación. *Departamento de Bases de Datos.* Santa Clara, Universidad Central "Marta Abreu" de Las Villas.

LOWENTHAL, B. (2005) Rule Enabling Applications with Oracle Business Rules. Oracle Corporation.

MELTON, J. & SIMON, A. R. (2002) *SQL1999: underestanding relational languaje components*, Morgab Kaufmann.

MORGAN, T. (2002) Business Rules and Information Systems: Aligning IT with Business Goals. Addison Wesley.

MOTA, S. A. (2005) Bases de Datos Activas.

MSDN (2008) MSDN LIBRARY VisualStudio 2008. Microsoft Corporation.

OMG (2010) Object Constraint Language. Object Management Group, Inc.

OPPEL, A. & SHELDON, R. (2008) *SQL: a biginner's guide*, McGraw-Hill Profesional.

PATON, N. W. & DÍAZ, O. (1999) Active Database Systems. *ACM Computing Surveys*, Vol. 31, No. 1**,** 41.

ROSS, R. G. (2010) What Is a Business Rule? *Business Rules Journal,* Vol. 11, No. 3.

ROSS, R. G. & LAM, G. S. W. Developing the Business Model, The Steps of Business Rule Methodology. *The BRS Business Rule Methodology.* Business Rules Journal.

TEDJASUKMANA, V. N. (2006) Translation of OCL Invariants into SQL:99 Integrity Constraints Hamburg, Germany

TEJADA, G. V. D. (2009) Sistema de Reglas de Negocio (SIREN). IN CENDA (Ed.

TOLEDO, A. P. (2009) Solución al problema de la cardinalidad en la generación automática de reglas de negocio en bases de datos relacionales. *Departamento de Bases de Datos.* Santa Clara, Universidad Central "Marta Abreu" de Las Villas.

ULLMAN, J. D. (1982) *Principles of DATABASE SYSTEMS*.

UMANATH, N. S. & SCAMELL, R. W. (2007) *Data Modeling and Database Design*, THOMPSON.

WEIDEN, M., HERMANS, L., SCHREIBER, G. & ZEE, S. V. D. (2002) Classification and Representation of Business Rules.

ZIMBRÃO, G., MIRANDA, R., SOUZA, J. M. D., ESTOLANO, M. H. & NETO, F. P. (2002) Enforcement of Business Rules in Relational Databases Using Constraints.

Anexos.

Anexo1: Porción del esquema lógico para transplante renal.