Facultad Matemática Física y Computación

Ingeniería Informática

Trabajo de Diploma

Título: Reglas de negocio desde la perspectiva de datos generadas automáticamente de modo diferido.

Autor: Lisset Torres Vazquez

Tutor: Dra. Martha Beatriz Boggiano Castill**o**

Curso 2014- 2015

Dictamen con derechos de autor para MFC

Hago constar que el presente trabajo fue realizado en la Universidad Central Marta Abreu de Las Villas como parte de la culminación de los estudios de la especialidad de Ingeniería Informática, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

Firma del autor _____

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del tutor _____ Firma del jefe del Seminario_____

Pensamiento

“Los grandes sueños solo se hacen realidad con inmensos sacrificios”

Ernesto Che Guevara

Dedicatoria

A:

Mi pequeña María Carla, mi nuevo motivo para vivir.
Mi abuelo Julio que vive en mí y ha sido todo mi apoyo espiritual.
Mis padres por la perseverancia y amor con que me han ayudado.
Mi esposo por su amor, comprensión y apoyo en todo momento.

Agradecimientos

Deseo ofrecer mi gratitud a:

- *Dios bendito por guiar mis pasos en la vida hasta el logro de mi preparación intelectual y personal.*
- *Mi tutora Martha Beatriz Boggiano Castillo, por la dedicación y paciencia con que me preparó para coronar con éxito mis estudios superiores.*
- *Mis profesores de la UCI y la UCLV que me pertrecharon de los conocimientos necesarios y las habilidades para desarrollar este trabajo final , en especial a estos últimos que han sabido comprender los momentos difíciles a los que he tenido que enfrentarme.*
- *A Ramiro y Beatriz por la ayuda que me ofrecieron en mis estudios y el apoyo para culminarlos.*
- *A mi hija, mi razón de ser, que con solo su presencia me ha dado las fuerzas necesarias para enfrentar cualquier reto de la vida.*
- *Mis padres por su incondicional comprensión y el cariño que siempre me han brindado.*
- *Mi esposo Carlos Pedro Soberats porque sin él sería imposible el logro de este sueño.*
- *Mi familia incluyendo a Juana y Pedro Luis por su apoyo en todo momento.*
- *A mis amigos por sus consejos, la confianza y el amor con que han estado a mi lado.*
- *A mis compañeros de aula que me han brindado su ayuda siempre que los he necesitado.*
- *A todos muchas gracias.*

Resumen

En la actualidad los Sistemas de Información constituyen una poderosa herramienta para la gestión de los datos a los distintos niveles de la sociedad. Tradicionalmente dichos SI se han desarrollado tratando los requisitos con técnicas que se basan en la codificación no automatizada.

El presente trabajo continúa con la investigación iniciada en (Boggiano, 2014) para la implementación de reglas de negocio desde la perspectiva de los datos en bases de datos relacionales. Los resultados teóricos obtenidos en dicha investigación se han implementado para la variante inmediata de todas las reglas desde la perspectiva de los datos. Para las reglas de restricción y notificación se propone una manera de implementación diferida, que no se había desarrollado.

Se realiza el diseño e implementación del subsistema modo diferido para la herramienta LPT-SQL, el cual permite realizar el chequeo a un conjunto de operaciones sobre los datos del negocio a través de transacciones. Se realiza además el diseño e implementación de un caso de estudio que pone a prueba la utilización de esta poderosa herramienta en la generación de reglas de negocio tipo restricción y notificación en un modo diferido.

Abstract

Today Information Systems are a powerful tool for data management at different levels of society. Traditionally these have been developed trying SI requirements with techniques based on automated no coding.

This thesis continues the investigation initiated (Boggiano, 2014) to implement business rules from the perspective of data in relational databases. The theoretical results obtained in this investigation have been implemented for immediate variant of all rules from the perspective of data. For Restriction and Notification rules proposes a way of deferred implementation, which had not been developed.

The design and implementation of the subsystem deferred mode for LPT-SQL tool, which allows checking a set of operations on business data through transactions performed. It also carries out the design and implementation of a case study that tests the use of this powerful tool in generating such restriction rules and reporting business in a delayed mode.

Tabla de contenido

Tabla de Ilustraciones

Tablas

INTRODUCCIÓN

En la actualidad los Sistemas de Información constituyen una poderosa herramienta para la gestión de los datos a los distintos niveles de la sociedad. Tradicionalmente dichos SI se han desarrollado tratando los requisitos con técnicas que se basan en la codificación no automatizada.

Un nuevo método que asume los sistemas de información compuestos por tres componentes (datos, procesos y reglas de negocios) se utiliza 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)

A partir de la década de 1980 se desarrollan los trabajos enfocados al tratamiento de las reglas de negocio de manera independiente a los SI, contribuyendo, en la implementación de las reglas en los SI, su efectivo cumplimiento.

Resulta común encontrar que dichas políticas o reglas de negocio son realizadas por los programadores e incluidas en el propio código fuente del sistema, conllevando a un engorroso proceso de control y mantenimiento de los cambios de dichas reglas. (Boggiano, 2014)

Existen disímiles tipos de herramientas de software dedicadas a la automatización de las reglas de negocio, dichas herramientas son llamadas Sistemas de Gestión de Reglas de Negocio (SGRN) o BRMS (de sus siglas en inglés Business Rules Manager System).

Los SGRN contienen como núcleo un motor de reglas de inferencia, ayudan a implementarlas desde un enfoque de reglas de negocio, permitiendo que los propios expertos del negocio participen en la escritura de las reglas para los SI. (Boggiano, 2014)

Las desventajas principales de utilizar los SGRN, reconocidas por (Hartmann, 2012) y apoyadas por (Boggiano, 2014) son las siguientes:

- Se requiere de alta experticia para desarrollar e implementar con un SGRN específico, además de habilidades avanzadas en software para diseñar y desarrollar aplicaciones, así como de comunicación efectiva usando las mejores prácticas; por todo lo anterior requiere un incremento en los costos de recursos humanos.
- Normalmente los ciclos de desarrollo son extensos.
- Un SGRN no es un ambiente de desarrollo totalmente adecuado, pues su eficiencia decrece ante tareas complejas.
- Los cambios de su modelo de objeto pueden tener implicaciones importantes para el sistema; si el modelo cambia, esto afectará a todas las reglas cuyas condiciones y acciones dependen de estas. Los cambios de requerimientos de software pueden llevar horas de trabajo para tratar errores de compilación. Esto sucede, sobre todo, porque los datos son enviados al motor de reglas a través de un modelo de objeto.
- El desarrollo con un SGRN conlleva a un fuerte acoplamiento con un vendedor específico, y una vez que se ha tomado la decisión de desarrollar un sistema usando un SGRN determinado, el éxito de la implementación está en la funcionalidad y atributos del SGRN particular. La opción de eliminar el uso del producto conduce a un serio costo de desarrollo, con implicaciones significativas de tiempo.

Una automatización adecuada de este proceso conlleva a una reducción considerable de tiempo y costo. Para automatizar estos “requisitos” es necesario primeramente darles una estructura inicial y mediante un lenguaje formal acercarlos a una representación más matemática, para luego aplicar un mecanismo de implementación (Morgan, 2002).

Un componente clave de la generalidad de los SI son las bases de datos (BD), cuyos esquemas conceptuales y lógicos permiten captar algunas reglas de negocio inherente a los mismos.

Se pueden considerar las reglas de negocio como las sentencias que permiten definir políticas de los negocios que se reflejan en los sistemas de información que ayudan a la gestión de los mismos. En la investigación (Boggiano, 2014) quedan demostradas las ventajas que brindan la aplicación de las reglas de negocio desde la perspectiva de los datos en bases de datos relacionales

El presente trabajo continúa con la investigación iniciada en (Boggiano, 2014) para la implementación de reglas de negocio desde la perspectiva de los datos en bases de datos relacionales. Los resultados teóricos obtenidos en dicha investigación se han implementado para la variante inmediata de todas las reglas desde la perspectiva de los datos. Para las reglas de restricción y notificación se propone una manera de implementación diferida, que no se había desarrollado.

Así, la evaluación y verificación de las reglas de negocio ya ha sido implementada para el modo inmediato, para las reglas de restricción y notificación lo que implica obligar a interrumpir la operación que desea realizar el usuario sobre los datos si la regla se incumple para el negocio cuando instancias de datos son insertadas, modificadas o eliminadas.

En los fundamentos teóricos del modo de chequeo diferido se describe como que en este modo de trabajo las acciones no son tomadas inmediatamente después de ser incumplida una regla, sino que son registradas como posibles reglas infringidas. El usuario tiene la opción de deshacer todas las operaciones sobre los datos o de evaluar por cada regla violada cual es el objeto causante de la infracción.

Se cuenta con la descripción de cómo se ha de implementar cada tipo de regla.

Problema científico:

¿Cómo desarrollar el subsistema del modo diferido para la implementación automática de la evaluación de las reglas de negocio?

Objetivo general:

Desarrollar el subsistema del modo diferido para la implementación automática de la evaluación de las reglas de negocio.

Objetivos específicos:

1. Valorar la utilización de un modo diferido para la implementación automática de las reglas de negocio desde la perspectiva de los datos.
2. Diseñar el subsistema para la herramienta LPT-SQL que adicione a la misma las funcionalidades de generación diferida.
3. Implementar el subsistema diseñado para la generación diferida de las RN.
4. Desarrollar una aplicación ejemplo que demuestre el uso de una base de datos con reglas de negocio desde la perspectiva de los datos (restricción y notificación) generadas automáticamente de modo diferido.

Preguntas de investigación:

1. ¿Qué consideraciones deben tenerse en cuenta para la implementación automática de modo diferido de las reglas de negocio desde la perspectiva de los datos?
2. ¿Cómo diseñar el subsistema para la herramienta LPT-SQL que adicione a la generación inmediata de las reglas ya existentes las funcionalidades de generación diferida?
3. ¿Cuáles herramientas serán adecuadas para implementar el módulo diseñado para la generación diferida de las RN?

4. ¿Qué estrategia emplear para el desarrollo de una aplicación ejemplo que demuestre el uso del módulo para la implementación diferida?

Justificación de la investigación:

Dentro del proceso de desarrollo de los sistemas de bases de datos, los cuales se han convertido en un producto estratégico de primer orden, se necesitan herramientas adecuadas que permiten deducir, inferir u obtener información nueva a partir de los datos almacenados o sucesos condicionados. En la revisión bibliográfica realizada existen antecedentes de estudios relacionados con aspectos medulares acerca de las reglas de negocio, partiendo de que los requisitos, enfocados hoy como reglas de negocio, 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, llegando hasta la automatización de este proceso para evitar fallas y, de este modo, se logran importantes reducciones de tiempo y costo.

La herramienta LPT-SQL es el resultado práctico de las investigaciones sobre reglas de negocio desde la perspectiva de los datos y ha sido demostrado su uso en estudio de casos para agilizar la implementación de reglas de negocio desde la perspectiva de los datos., sin embargo la versión actual no cubre las posibilidades de reflejar en el SI el posible incumplimiento temporal de las reglas de restricción e implementación, aquellas que después de un conjunto de operaciones sobre los datos del negocio quedan finalmente en estado e cumplimiento.

Es necesario permitir que la herramienta incorpore la posibilidad de tratar las reglas de restricción y notificación de manera diferida para hacer que situaciones de incumplimiento temporal que no afectan la estabilidad del negocio sean consideradas. Atendiendo a lo planteado podremos entonces lograr evaluar el cumplimiento de las reglas después de realizar un conjunto de operaciones con los datos.

El presente trabajo está estructurado de acuerdo a la siguiente secuencia lógica: introducción, cuatro capítulos, conclusiones, recomendaciones, bibliografía y anexos.

Capítulo 1: “Consideraciones sobre las reglas de negocio desde la perspectiva de los datos y los modos de implementación de las reglas de restricción y notificación”. Se abordan aspectos generales sobre Reglas de Negocio, características fundamentales que deben cumplir, modos de implementación de las reglas de negocio desde la perspectiva de los datos, así como las desventajas de la herramienta actual LPT-SQL.

Capítulo 2: “Diseño del subsistema Modo Diferido del LPT-SQL”. Se reflejan los artefactos del diseño del subsistema modo diferido, tales como: Diagrama de Clases, Diagrama de Paquetes, Diagrama de Componentes, Diagrama de Despliegue.

Capítulo 3: “Implementación y prueba del subsistema modo diferido del LPT-SQL”. Se reflejan los pasos a seguir para la implementación del subsistema dentro de la herramienta LPT-SQL, así como casos de prueba a realizar para la validación de su funcionamiento.

Capítulo 4: “Caso de Estudio para la utilización del LPT-SQL modo diferido, y prueba de la aplicación”.

CAPÍTULO I

En el presente capítulo se abordan aspectos generales de las reglas de negocio, sobre todo las reglas de negocio denominadas desde la perspectiva de los datos, reflejando los distintos modos de implementación y los disímiles recursos de bases de datos utilizados para ello.

Reglas de negocio

Una regla de negocio, es una regla que está bajo jurisdicción del negocio, lo cual significa que puede ser creada, revisada y eliminada cuando el negocio lo estime conveniente. (Ross, 2010)

Tony Morgan (Morgan, 2002) por su 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*. El citado autor enfatiza en un conjunto de características fundamentales que las reglas de negocio deben cumplir, y que la autora de este trabajo apoya a partir de (Boggiano, 2014), se consideran relevantes las siguientes:

- En términos generales, son restricciones que definen condiciones, las que deben ser verdaderas en situaciones especificadas, bajo las cuales un proceso es realizado, o que existirán después de completado un proceso.
- Las declaraciones de reglas de negocio en el modelo de negocio definen la lógica deseada del negocio, ellas describen una situación que el negocio exige; si se expresan como funciones lógicas, siempre devolverían el valor verdadero.
- El analista de negocio debe especificar una serie de declaraciones claras sobre la lógica base de un negocio.

Según los autores (Hay, 2000) una regla de negocio es *una sentencia que define o restringe algunos aspectos del negocio*. Tiene la finalidad de establecer la estructura del negocio, controlar o influenciar el comportamiento de negocio generalmente son estudiadas desde dos perspectivas: desde la perspectiva del negocio, que pertenece a cualquier restricción de la conducta de las personas en la empresa y desde la perspectiva de los SI, relacionadas con los hechos que son grabados como datos y las restricciones sobre los cambios a los valores de tales hechos, además de los procesos que hay que seguir.

Tradicionalmente los programadores han incluido las Reglas de Negocio en el código fuente de las aplicaciones como parte del flujo de ejecución de las mismas, o como parte de los objetos de las bases de datos, conllevando a un aumento del costo de codificación, validación y mantenimiento de ellas.

Una solución viable a estos problemas es el desarrollo de los Sistemas de Gestión de Reglas de Negocio (SGRN), centrados en facilitar la creación, implementación y mantenimiento de las reglas permitiendo su integración con el resto de Sistemas de Información. (Morgan, 2002)

Los Sistemas de Gestión de Reglas de Negocio (SGRN) usan una variedad de lenguajes propietarios para representar las reglas de negocio que gestionan. Actualmente se aprecian esfuerzos de estandarización para representar estas reglas de la forma if…then que garanticen el intercambio de reglas en la Web. (Boggiano, 2014)

1.1 Implementación de Reglas de Negocio

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 (Alonso, 2010). Entre las formas tradicionales de programación de reglas están:

- **Scripts**

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

- **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 enfatiza (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.

En el momento de implementar las RN es necesario considerar cada una de las alternativas para determinar cuál funciona mejor en una determinada situación, se deben tener en cuenta: la viabilidad a largo plazo de la estrategia, el rendimiento en tiempo de ejecución, el grado de cumplimiento de la regla, la flexibilidad, y la capacidad para mantener las operaciones del negocio.

Se evidencia una fuerte tendencia a implementar las reglas automáticamente usando los motores de reglas o los SGRN como una buena solución basada en BRA, sin embargo Morgan (Morgan, 2002) apoya la posibilidad de que algunos tipos de reglas se pueden implementar de manera más simple o mejor fuera de estos. A pesar de ser una herramienta muy utilizada en los últimos tiempos, presenta según (Boggiano, 2014), desventajas significativas, tales como:

- La adopción de uno de los motores de reglas de inferencia, comercializados actualmente, requiere una gran inversión de tiempo.

- Obliga en algunos casos a complejizar la implementación de reglas que pudieran programarse de manera más sencilla

1.2 Herramientas para la gestión de Reglas de Negocio.

En la actualidad se evidencia una fuerte tendencia al desarrollo de herramientas para la gestión de Reglas de Negocio. Para el manejo de las reglas de negocio, según el modo en que se representan, implementan y ejecutan estas herramientas se clasifican en:

- Herramientas independientes de los gestores de bases de datos: las reglas se implementan en la BD, pero los recursos son generados automáticamente y manejadas por herramientas que no son gestores de bases de datos (RuleBurst).
- Herramientas basadas en servicios: las reglas son creadas por herramientas de desarrollo que constituyen una capa intermedia de servicios de aplicación y residen en el servidor de aplicaciones.
- Sistemas basados en reglas: Se captura la lógica del negocio a un alto nivel y las reglas asociadas, se usan métodos orientados a la lógica. En tiempo de ejecución se usan motores especiales para procesar las reglas y generar respuestas apropiadas (Drools o JBoss Rules).

La tecnología a utilizar depende de varios factores, pero particularmente en el tipo de sistema en desarrollo. Por ejemplo, en una aplicación típica basada en la aplicación del conocimiento, las reglas se capturan y se almacenan en una base de reglas y se ejecutan por un motor de reglas. Sin embargo, en un sistema típico de flujo de trabajo, las reglas de negocio se integrarán en la definición del flujo de trabajo y será utilizado un sistema de flujo de trabajo para ejecutar el mismo.

En esta investigación se fundamenta a través de una aplicación el primer tipo **Herramientas independientes de los gestores de bases de datos**, pero que

las reglas se implementan dentro de las bases de datos. Con la herramienta LPT-SQL a partir de sus fundamentos teóricos se propone una manera de trabajar los sistemas de información a partir de reglas de negocio, originadas en el propio negocio vinculadas a los datos del mismo, de modo que necesita ser evaluada ante los cambios en el negocio al registrarse nuevos datos, modificarse o eliminarse.

Para trabajar la implementación de las reglas de negocio en cualquier tipo de herramienta se reconoce como un componente esencial el repositorio de reglas que se encarga de almacenar las declaraciones de las reglas y lograr algún grado de independencia de las mismas con los Sistemas de Información.

1.3 Repositorio de Reglas.

El repositorio de reglas es uno de los componentes fundamentales para trabajar con reglas de negocio y lograr independencia de las reglas y los SI. El repositorio almacena la declaración de las reglas y aspectos que permiten garantizar su control y mantenimiento (Morgan, 2002). Dicho investigador plantea que *cada regla de negocio debe describirse en el repositorio por un conjunto de atributos, entre los más importantes están: el identificador de la regla, número de la versión de la regla correspondiente, la fecha de creación o del último cambio, el tipo de regla, el estado, la sentencia de la regla en sí.*

Reglas de Negocio desde la perspectiva de los datos

Atendiendo al criterio de diferentes investigadores se trabaja el concepto de reglas de negocio desde la perspectiva de los datos, como aquellas reglas que están involucradas en las operaciones sobre la base de datos del negocio, a continuación se presenta la siguiente clasificación de los tipos de reglas desde la perspectiva de los datos, resultado obtenido en (Boggiano, 2014):

- **Restricción:** Obliga a que se cumplan los requisitos del negocio, contribuyendo a preservar la integridad del mismo.
- **Cómputo:** Su objetivo es calcular un valor determinado en el negocio, y su resultado es numérico. Este patrón comparte grandes semejanzas con el patrón de clasificación (Ross, 2009)
- **Clasificación:** Organiza el conocimiento básico del negocio contribuyendo claramente al significado de conceptos, este patrón es conocido también como una regla de definición.
- **Notificación:** Este patrón informa a los usuarios autorizados del negocio sobre algún conocimiento básico en tiempo real, no restringe estados del negocio, solo lo informa para que se tomen las medidas pertinentes.

1.4 Formas de expresar las reglas de negocio desde la perspectiva de los datos y lenguajes para escribirlas.

Los niveles para expresar las reglas son cuatro (Von Halle, 2002), cada uno para una audiencia diferente: conversación informal del negocio, versión en lenguaje natural, versión en lenguaje de especificación y versión en lenguaje de implementación. En última instancia las reglas se traducen del lenguaje de especificación al lenguaje de implementación, el cual tiene todo el potencial para ser ejecutado.

En (Morgan, 2002) se distinguen tres niveles de expresión de reglas de negocio: informal, con la regla de negocio expresada como sentencia en lenguaje natural, tal y como el cliente del negocio desee; técnico, en que se combina referencias a datos estructurados, operadores y restricciones con el lenguaje natural; y el formal, de implementación, para proporcionar sentencias conforme a una sintaxis definida y proporciona la funcionalidad completa de la regla.

En esta investigación se utilizan los niveles expuestos por (Morgan, 2002) y recreados en (Boggiano, 2014) donde se incorpora el patrón de notificación, que aporta al nivel informal el término lenguaje natural estructurado , definiendo para el lenguaje LPT (lenguaje de patrones técnico) el formal, utilizando recursos de bases de datos.

Los patrones utilizados en la presente investigación son el patrón de restricción y el patrón de notificación.

A continuación se presentan las reglas de notificación y restricción, con sus patrones correspondientes en el nivel de lenguaje natural estructurado y en lenguaje técnico LPT (lenguaje de patrones técnico), cuya definición se puede encontrar en (Boggiano, 2014) y (Alonso, 2010).

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. En (Boggiano, 2014) se define la estructura de dicho patrón:

Su expresión en lenguaje natural estructurado:

Notificar <mensaje> si <hecho>

Ejemplo tomado de (Alonso, 2010):

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 no pueden ser violadas directamente; responden a la necesidad, no de restringir algún estado del negocio sino de anunciarlo a las partes interesadas.

Su expresión en lenguaje técnico es:

RN#7: Notificar "Alerta Rija de la Pandemia" si sizeof(sujeto.Examen.resultado="AH1N1")>800

Patrón de Restricción

No son pocos los autores que consideran a este patrón de gran importancia para satisfacer los requisitos del negocio, siguiendo el criterio de que solo con reglas de este tipo puede hacerse cumplir políticas prohibitorias en el sistema del negocio.

Su expresión en lenguaje natural estructurado:

<determinante> <sujeto> (no puede tener <características>) |

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

Ejemplo tomado de (Alonso, 2010):

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

Su expresión en lenguaje de patrones técnico:

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

1.5 Modos de implementación de reglas de negocio de restricción y notificación.

Para la implementación de las reglas de negocio tipo restricción y notificación desde la perspectiva de los datos se tienen dos enfoques principales: inmediato y diferido. (Boggiano, 2014)

La implementación de las reglas bajo un enfoque inmediato provoca que inmediatamente después de finalizada una operación se chequee si se infringe la o las reglas de negocio asociadas a la misma usando un disparador y una vista para implementar la regla. Dicho disparador, según (Boggiano, 2014), tendrá la siguiente estructura:

CREATE TRIGGER <nombretrigger> AFTER <INSERT|UPDATE|DELETE >

ON <tablas implicadas según LE>

if /* consulta sobre la vista, si devuelve elementos se lanza el mensaje*/

Ilustración 1: Chequeo inmediato para proceso de integridad para regla de negocio tipo restricción

Aunque una regla del negocio implementada sobre la bases de datos tiene la función de garantizar la integridad de los datos gestionados, no es menos cierto que en una determinada ejecución del SI dichas reglas puedan incumplirse

temporalmente. En ocasiones se hace necesario que los registros en la base de datos no deban chequearse inmediatamente después de una operación evaluando que se infrinjan una o más reglas; así a la manera de hacer que una regla sea chequeada después de realizarse un conjunto de pasos u operaciones sobre los datos se le ha llamado modo diferido (Boggiano, 2014).

Según (Boggiano, 2014), para dar solución a este problema, para cada regla de negocio tipo restricción se tiene una vista de integridad que dice si esa regla ha sido infringida y qué <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).

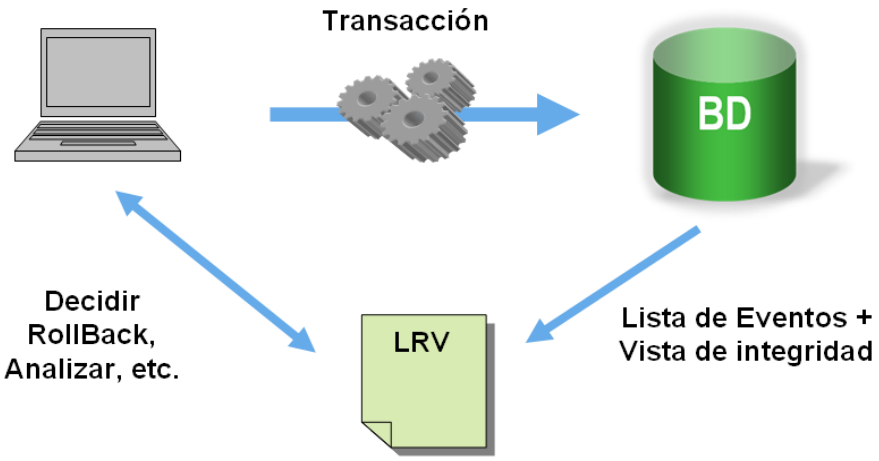

Ilustración 2: Chequeo diferido para proceso de integridad para regla de negocio tipo restricción

El sistema del negocio puede decidir qué hacer, si deshacer todas las operaciones o, 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.

Este método espera a que termine de ejecutarse la operación o conjunto de operaciones para realizar la validación de las reglas de negocio, las reglas que se hayan incumplido son adicionadas a una lista de posibles reglas infringidas, relacionando los sujetos que provocaron la infracción, para su posterior valoración por parte del usuario o de la aplicación.

En (Boggiano, 2014) se propone que 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.

Para garantizar el correcto inicio del proceso de integridad en la transacción se deben tener en cuenta los siguientes pasos:

1. Realizar las operaciones necesarias del usuario.
2. Verificar las violaciones de las operaciones previamente finalizadas.
3. Análisis y decisión final sobre las infracciones cometidas.
4. Mostrar opcionalmente la información almacenada de dicha transacción.

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

END;

Pasos previos:

1. Eliminar todas las instancias de la tabla *rn_restriccion* (tabla que contiene aquellas reglas que en una operación dada infringieron la regla de negocio asociada).
2. Comenzar la transacción de la que luego se toman determinaciones.

Para ello la siguiente sección de código en SQL estándar.

```
DELETE FROM rn_restriccion;

START TRANSACTION;
```

Primer segmento:

En el primer segmento, posterior a la inclusión de los pasos previos anteriores el diseñador de la base de datos es el responsable de codificar cada una de las operaciones a realizar por el usuario.

Segundo segmento:

Se realiza la verificación del cumplimiento o no de las reglas presentes en la lista de posibles reglas infractoras para adicionarlas a la lista de reglas violadas.

Tercer segmento:

Se realiza un análisis de cada una de las reglas presentes en la lista de reglas violadas, mostrando el nombre de la regla y la cantidad de sujetos causantes de la violación, para decidir los pasos a seguir: aceptar la violación o deshacer las operaciones involucradas.

Cuarto segmento:

Se muestra opcionalmente la información al usuario de las infracciones cometidas en las operaciones realizadas.

Los procedimientos almacenados existentes en una base de datos del negocio son los encargados de ejecutar la o las operaciones sobre los datos a través de las transacciones. Al realizar el análisis de las posibles reglas violadas, este mecanismo brinda la posibilidad al usuario de hacer un rollback de las operaciones involucradas con la regla o de analizar los sujetos implicados en la violación.

Después de evaluar los fundamentos teóricos propuestos por (Boggiano, 2014) se decide utilizarlos en este trabajo, para efectivamente llevar a cabo el diseño e implementación de una versión de la herramienta LPT-SQL que incluya el subsistema modo diferido, permitiendo que con la nueva versión de la aplicación las operaciones sobre la base de datos puedan evaluarse para las reglas de restricción y notificación de manera aisladas (modo inmediato) o en transacciones (modo diferido).

1.6 Recursos de Bases de Datos y su automatización en LPT-SQL para reglas de restricción y notificación para el modo inmediato

La herramienta LPT-SQL genera los recursos de bases de datos en la implementación automática de las reglas, para lo cual toma como entrada la regla en lenguaje LPT. Para la ejecución adecuada de la herramienta es necesario contar con una base de datos del negocio, cuyas tablas han sido creadas a partir de los términos del negocio y en función de estas han sido escritas las reglas usando LPT.

Cuando se realiza una acción sobre los datos del negocio (insert, update, delete) es realizado el chequeo del cumplimiento de las reglas asociadas a los

mismos, ejecutando los disparadores en caso de la violación de una de las vistas de integridad presentes en la base de datos.

1.7 Utilización de transacciones para la implementación de reglas de negocio tipo restricción y notificación con el modo diferido.

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. 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)

Cuando una transacción finaliza 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 o listas doblemente vinculadas, deben estar correctas al final de la transacción.

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.

La propuesta del modo diferido para la evaluación de las reglas de restricción y notificación mejora considerablemente la usabilidad de la herramienta, ya que permite la implementación de un modo de chequeo diferido de las reglas de negocio desde la perspectiva de los datos en bases de datos relacionales. Permite además la realización de varias operaciones sobre los datos del negocio, chequeando al culminar las mismas la integridad de las vistas y a su vez la integridad de los datos almacenados.

La herramienta está desarrollada bajo los estándares de la metodología de desarrollo RUP, pasando por cada una de las etapas de la misma y obteniéndose los artefactos correspondientes. LPT-SQL utiliza JAVA como plataforma para su ejecución sin depender del Sistema Operativo utilizado. Cuenta además con la capacidad de operar con los Gestores de Bases de Datos PostgreSQL y Microsoft SQLServer, este último hasta la versión Microsoft SQLServer 2005.

CAPÍTULO 2

En el presente capítulo se realiza el diseño del Subsistema modo diferido para la herramienta LPT-SQL, se confeccionan los artefactos de ingeniería correspondientes y necesarios para su posterior implementación.

2.1 Modelo del Sistema Actual

Actualmente el traductor de reglas de negocio LPT-SQL v1.4 cuenta solamente con el subsistema de chequeo inmediato, el cual brinda la posibilidad de verificar cada una de las vistas de integridad implicadas en una operación sobre la base de datos del negocio.

El método de chequeo inmediato viabiliza a corto y mediano plazo las operaciones aisladas (insert, update, delete) para el Sistema de Información, no siendo así cuando se requiere la realización de varias operaciones sobre la base de datos del negocio cumpliendo, evaluando posteriormente las reglas de integridad predefinidas.

A través de la herramienta LPT-SQL el especialista puede generar o crear los repositorios de reglas, los cuales almacenan las reglas para la base de datos del negocio en cuestión. Además permite que cada regla creada sea almacenada en el repositorio de generación, para posteriormente, activar dicha regla en los datos.

La herramienta permite extraer la información de catálogos de la base de datos del negocio y los recursos de la base de datos que pudieran constituir la implementación de las reglas.

2.2 Descripción de la herramienta LPT-SQL v1.4

La herramienta LPT-SQL v1.4 está diseñada para ser utilizada por los especialistas o técnicos en la captura y análisis de los requerimientos funcionales de un Sistema de Información, viabilizando su traducción a un

lenguaje más entendible por los diseñadores y programadores de las bases de datos del negocio.

2.2.1 Vista arquitectónica del proceso de traducción de las reglas de LPT a SQL

Como resultado de la investigación (Boggiano, 2014) se obtiene la arquitectura base para la implementación de una herramienta para la gestión de reglas de negocio en bases de datos relacionales (Ilustración 3).

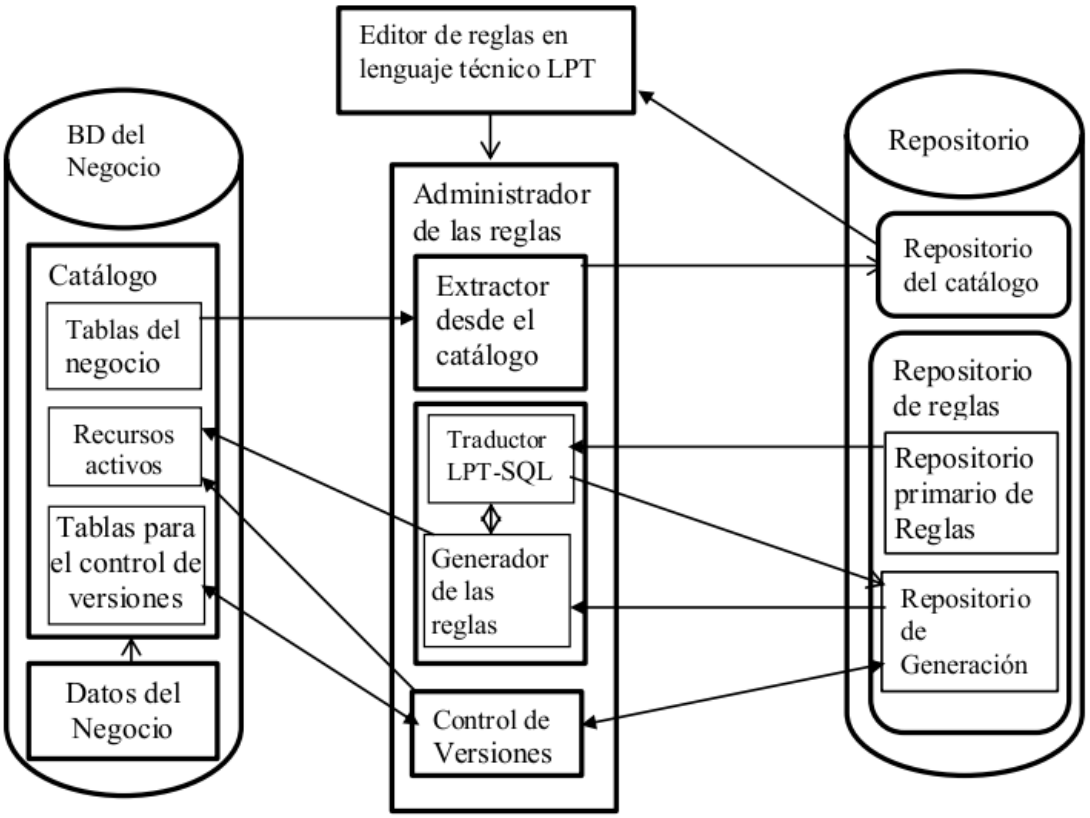

Ilustración 3: Arquitectura de la administración de reglas de negocio en BD relacionales.

Según (Boggiano, 2014) los componentes principales de la arquitectura y sus subcomponentes son los siguientes:

- Repositorios.

  - Repositorio de reglas de negocio: Conformado por dos secciones, el repositorio primario de reglas que contiene cada regla en sus tres niveles: natural, técnico y formal. La segunda sección, que contiene las reglas compiladas que se deben generar y almacena la consulta base para la generación de la regla.
  - Repositorio del catálogo: Contiene información sobre las tablas de las bases de datos del negocio, información sobre los nombres, atributos, interrelaciones, recursos activos, funciones, procedimientos almacenados y funciones. También almacena información sobre las tablas auxiliares para el control de la modificación.
- Base de datos relacional del negocio: Se distinguen tres grupos de componentes:
  - Tablas principales y sus atributos: Estos han sido nombrados con los términos adecuados del negocio en cada caso.
  - Recursos activos: Los disparadores, funciones, procedimientos almacenados, muchos de los cuales constituyen implementaciones de reglas de negocios de las categorías con perspectiva de los datos.
  - Tablas de apoyo al control de versiones: Son tablas que registran con cuáles reglas se inserta cada dato.
- Editor de reglas de negocio: Es un módulo que ha de permitir editar las reglas en lenguaje natural estructurado de patrones y lenguaje LPT, reconociendo los datos de los nombres de tablas y atributos.
- Administrador de reglas: Es el núcleo de esta arquitectura, está compuesto por :
  - Módulo extractor desde el catálogo: Extrae el catálogo de la base de datos y conforma el repositorio del catálogo.
  - Traductor LPT-SQL: Con los datos almacenados en el repositorio primario de reglas, traduce las reglas de LPT a consulta base para la regla y las almacena en el repositorio de generación.

- Generador de reglas: A partir de la información guardada en el repositorio de generación que constituye el núcleo de la regla en el lenguaje formal SQL produce los recursos correspondientes a la implementación de la regla y los recursos (disparadores) que implementa las listas de eventos.

En la herramienta que se diseña, LPT-SQL, ver 1.5 el generador de reglas, permitirá dos modos de implementación para las reglas de restricción y notificación; cuando se seleccione el modo diferido.

2.3 Diagrama de Casos de Uso de LPT-SQL v1.4

En la siguiente figura se pueden apreciar los principales casos de uso con que cuenta la herramienta. El técnico o especialista tiene la posibilidad de realizar las siguientes operaciones:

1. Crear un Repositorio de Reglas del Negocio.
   El especialista tiene la posibilidad de crear un nuevo Repositorio de Reglas del Negocio. En dicho repositorio es donde se almacenarán las nuevas reglas.
2. Cargar un Repositorio de Reglas del Negocio anteriormente creado.
   El especialista puede cargar un Repositorio de Reglas del Negocio creado con anterioridad, el cual tiene almacenadas reglas ya creadas y generadas, con el estado de la misma.
3. Crear una Regla del Negocio.
   El especialista tiene la posibilidad de crear una nueva Regla del Negocio, introduciendo el lenguaje formal de dicha regla y el lenguaje LPT.
4. Eliminar una Regla del Negocio.
   El especialista puede eliminar una regla deseada del Repositorio de Reglas.
5. Modificar una Regla del Negocio.
   El especialista puede realizar la modificación de una regla deseada, ya sea el estado de misma o en la escritura del lenguaje LPT o formal.

6. Activar una Regla del Negocio.

   El especialista puede activar una regla deseada. Cuando una regla es activada, son creados en la base de datos del negocio los recursos necesarios para la correcta ejecución de la misma.

7. Generar una Regla del Negocio.

   Cuando es generada una regla del negocio el lenguaje LPT es traducido a lenguaje SQL, teniendo en cuenta las características del Gestor de Bases de Datos seleccionado.

8. Extraer información de la Base de Datos física.

   Brinda la posibilidad de extraer la información referente a los catálogos, funciones, vistas, procedimientos y tablas existentes en la base de datos física.

9. Seleccionar Modo de Interfaz de Usuario.

   Permite seleccionar el modo de la interfaz de usuario de la aplicación (Simple o Avanzada).

10. Realizar conexión a la base de datos del Negocio.

    Permite realizar la conexión con la base de datos física para realizar las operaciones sobre los recursos de la misma.

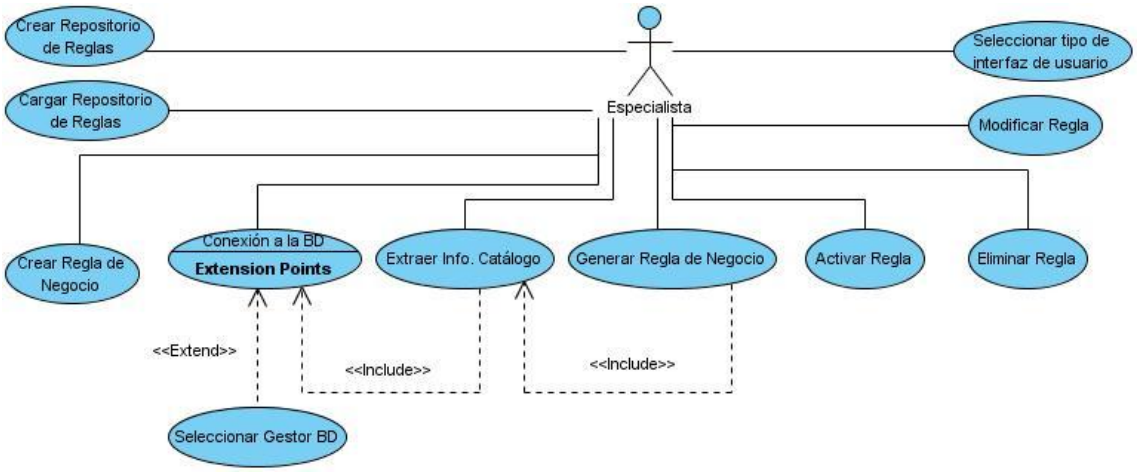

Ilustración 4: Diagrama de Casos de Uso del LPT-SQL v1.4

2.3.1 Diagrama de Paquetes y Diagrama de Clases de LPT-SQL v1.4

Uno de los paquetes principales con que cuenta la herramienta en su versión actual es el paquete *Generation*. Dicho paquete agrupa al paquete *Collect*: encargado de extraer toda la información de catálogos presentes en la base de datos del negocio, el paquete *Store*: encargado de controlar la lógica para almacenar o activar la regla de negocio en la base de datos y el paquete *Assembly*: encargado de controlar la lógica de negocio para la generación de la regla bajo en enfoque inmediato, dependiendo del Gestor de Bases de Datos seleccionado.

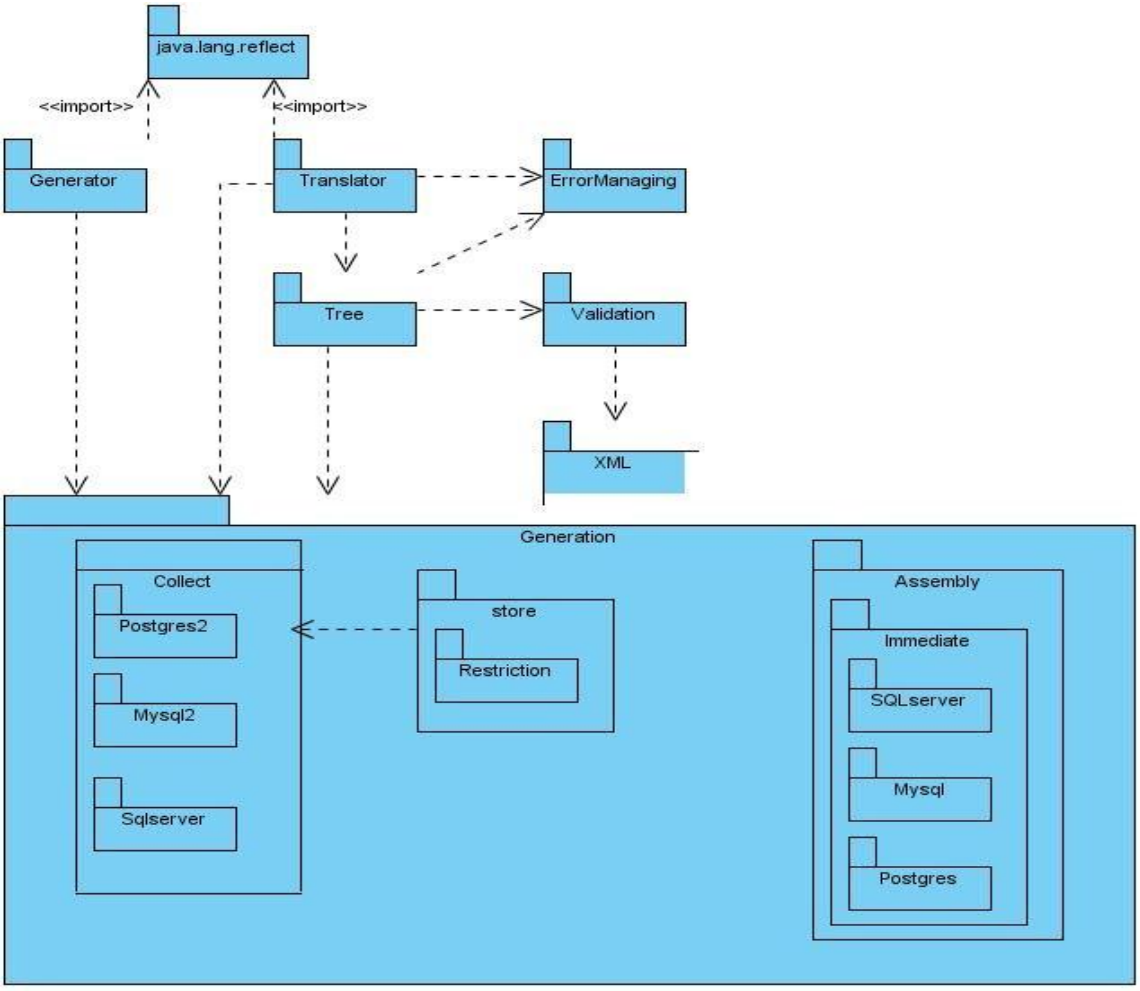

Ilustración 5: Diagrama de Paquetes LPT-SQL v1.4

A continuación se muestra el diagrama de clases de la versión 1.4 del LPT-SQL, como se puede apreciar el mismo solo contempla las implementaciones para el modo inmediato de las reglas de negocio tipos notificación y restricción.

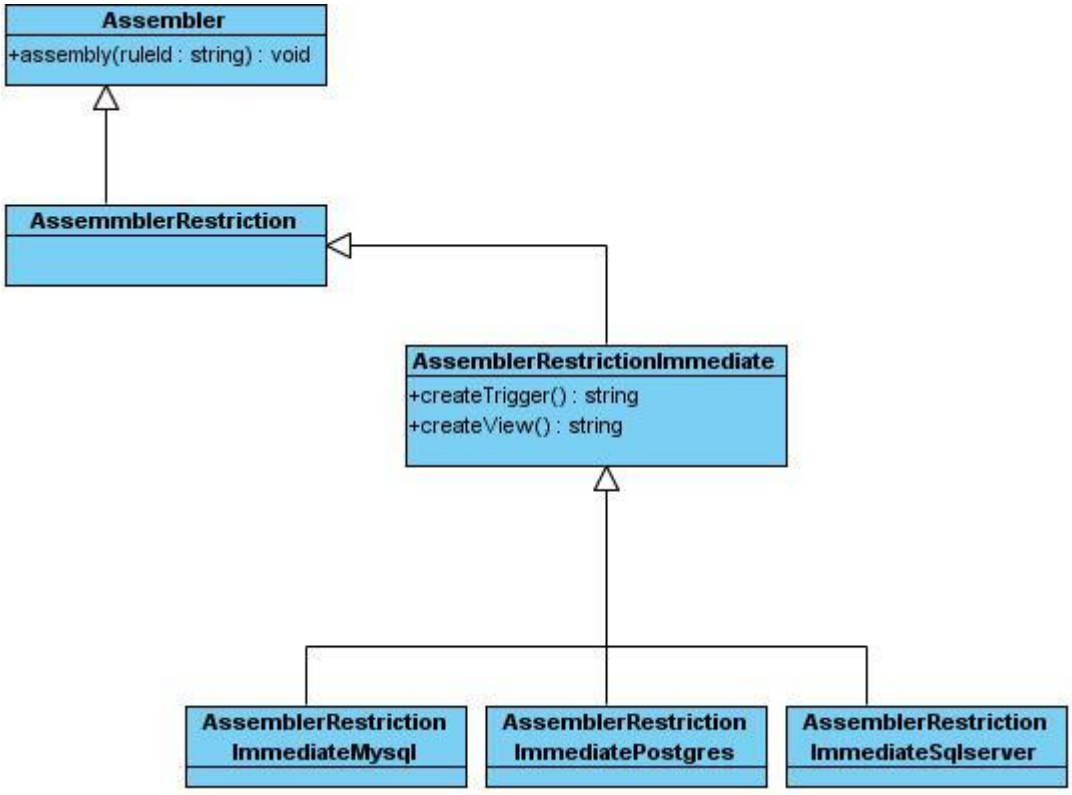

Ilustración 6: Diagrama de Clases LPT-SQL v1.4

2.4 Diseño del subsistema diferido del LPT-SQL

La utilización de un modo de chequeo diferido para la validación de las reglas de negocio, presentes en un SI, permite realizar un conjunto de operaciones sobre los datos del negocio para, posteriormente a la culminación de dichas operaciones revisar la integridad de los datos asociados a las reglas y en caso de existir una violación el usuario tendrá la posibilidad de deshacer las operaciones anteriores.

A raíz de que en la actualidad la herramienta no cuenta con la posibilidad de implementar chequeo de reglas sobre un conjunto de operaciones sobre los datos del negocio se decidió realizar el estudio y análisis para la implementación de un nuevo subsistema que permita la generación de reglas de negocio con una secuencia distinta de chequeo, garantizando la verificación de las vistas de integridad posterior a la ejecución de un conjunto de operaciones sobre la base de datos.

Se hizo necesario la inclusión de un nuevo caso de uso o requerimiento funcional: Seleccionar Modo (Diferido o Inmediato), el cual permite seleccionar el tipo de enfoque bajo el cual serán generadas las reglas.

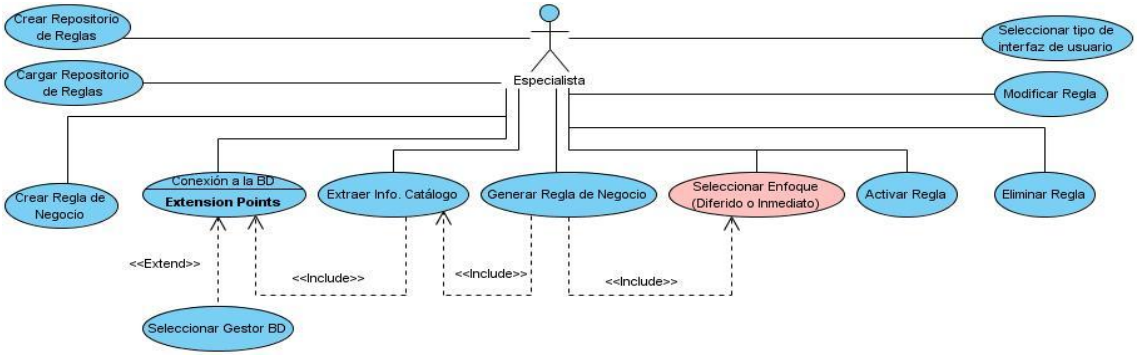

Ilustración 7: Diagrama de Casos de Uso LPT-SQL (Subsistema Diferido)

2.4.1 Diagrama de Paquetes

Para la implementación del subsistema diferido se adicionaron 5 paquetes de clases al conjunto de paquetes existentes:

1. *Notificación*: Se encuentra dentro del Paquete *Store* (procesa el tipo de regla a generar). se encarga de almacenar las características de las reglas tipo notificación que serán generadas en el sistema.
2. *Mediate*: Se encuentra dentro del Paquete *Assembly* (genera las reglas en lenguaje SQL), encargado de contener los Paquetes de clases necesarios para la generación de las reglas en modo diferido.

3. *SQLServer*: Se encuentra dentro del Paquete *Mediate*. Contiene la clase encargada de proporcionar las características de dicho Gestor para la generación de la regla.
4. *MySQL*: Se encuentra dentro del Paquete *Mediate*. Contiene la clase encargada de proporcionar las características de dicho Gestor para la generación de la regla.
5. *Postgres*: Se encuentra dentro del Paquete *Mediate*. Contiene la clase encargada de proporcionar las características de dicho Gestor para la generación de la regla.

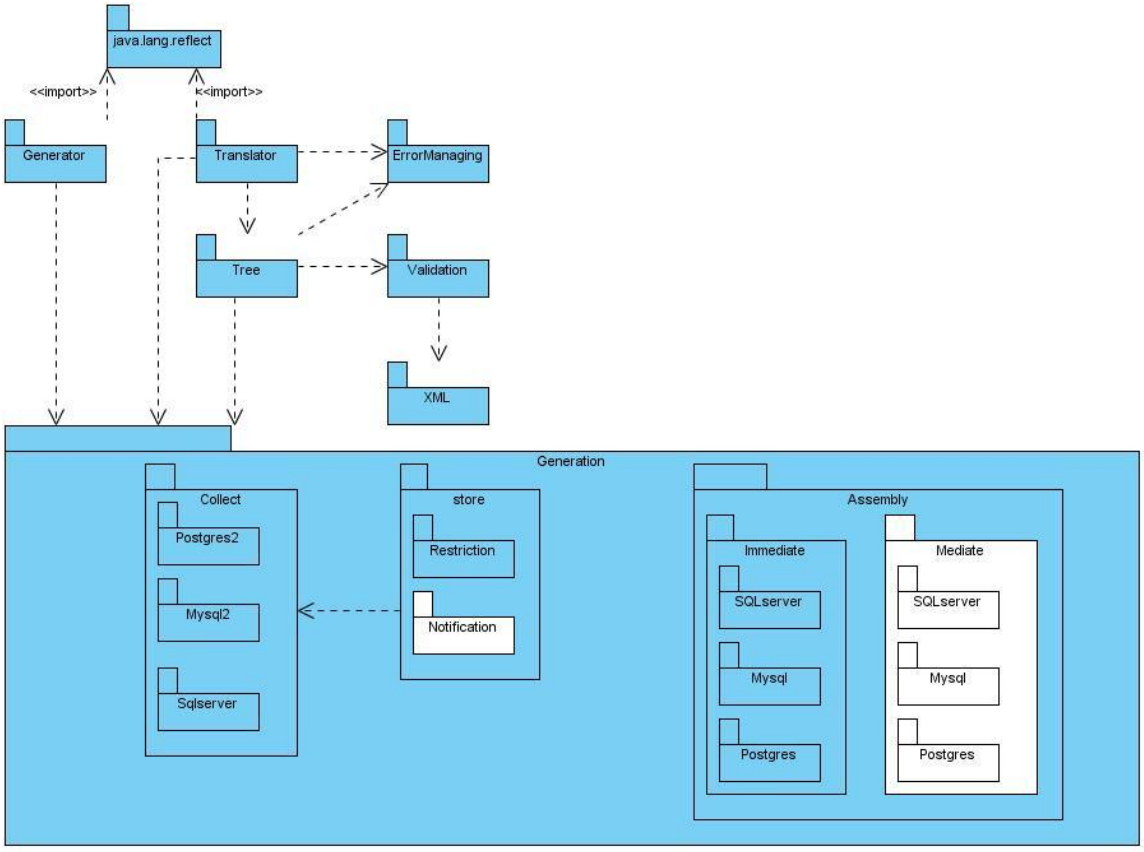

Ilustración 8: Diagrama de Paquetes

2.4.2 Diagrama de Clases

Para garantizar el correcto funcionamiento del subsistema diferido del LPT-SQL se hizo necesario la modificación de la estructura de entidades existentes en la herramienta, adicionándose 13 nuevas clases como se muestra a continuación:

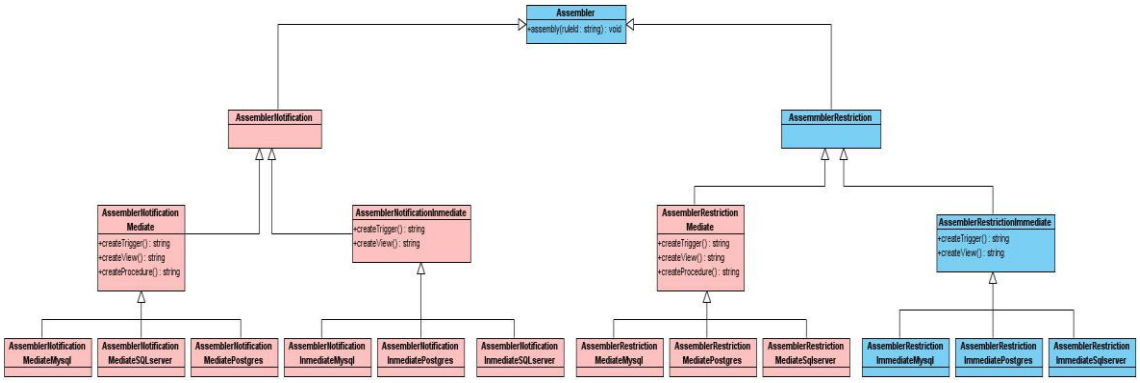

Ilustración 9: Diagrama de Clases

AssemblerNotification: Heredando de la Interfaz Assembler es la encargada de permitir al sistema acceder a las operaciones de las reglas tipo notificación utilizando el modo inmediato o diferido.

AssemblerNotificationMediate: Extendiendo de *AssemblerNotification* es la encargada de generar la regla en lenguaje sql, utilizando los recursos necesarios, dependiendo del Gestor de Bases de Datos seleccionado bajo un modo diferido.

Las clases *AssemblerNotificationMediateMySQL*, *AssemblerNotificationMediateSQLServer* y *AssemblerNotificationMediatePostgres*, contienen las características específicas de cada Gestor de Base de Datos para la generación de los recursos relacionados con la regla de negocio en cuestión.

AssemblerNotificationInmediate: Extendiendo de *AssemblerNotification* es la encargada de generar la regla en lenguaje sql, utilizando los recursos necesarios, dependiendo del Gestor de Bases de Datos seleccionado bajo un modo inmediato.

Las clases *AssemblerNotificationInmediateMySQL*, *AssemblerNotificationInmediateSQLServer* y *AssemblerNotificationInmediatePostgres*, contienen las características específicas de cada Gestor de Base de Datos para la generación de los recursos relacionados con la regla de negocio en cuestión.

La clase *AssemblerRestrictionMediate* es la clase controladora de las operaciones de generación de las reglas de negocio tipo restricción bajo un modo diferido de chequeo. Las clases *AssemblerRestrictionMediateMySQL*, *AssemblerRestrictionMediateSQLServer* y *AssemblerRestrictionMediatePostgres*, contienen las características específicas de los gestores de bases de datos MySQL, Microsoft SQLServer y PostgreSQL respectivamente, para la generación de los recursos relacionados con la regla de negocio en cuestión.

2.4.3 Diagrama de Componentes

El traductor LPT-SQL v1.4 cuenta con una estructura de componentes bien definida, permitiendo la comunicación entre ellos a través de interfaces. El componente *Compiler* provee una interfaz para compilación, mientras que el componente *Generator* proporciona la interfaz para la generación de la regla. Con dichas interfaces va a interactuar la aplicación, contando con una estructura semi_desacoplada, permitiendo la reutilización de los mismos por otra aplicación.

Para la implementación del subsistema diferido no fue necesario modificar la relación entre los componentes del traductor (Solís, 2011), solamente la estructura interna de los componentes **StoreKeeper** y **Assembler**, debido a que las clases y paquetes creados se encuentran embebidos dentro de dichos componentes como se muestra en la figura **Ilustración 11: Parte del Diagrama de Componentes**. En la figura se muestran solamente los paquetes y entidades adicionadas a la aplicación.

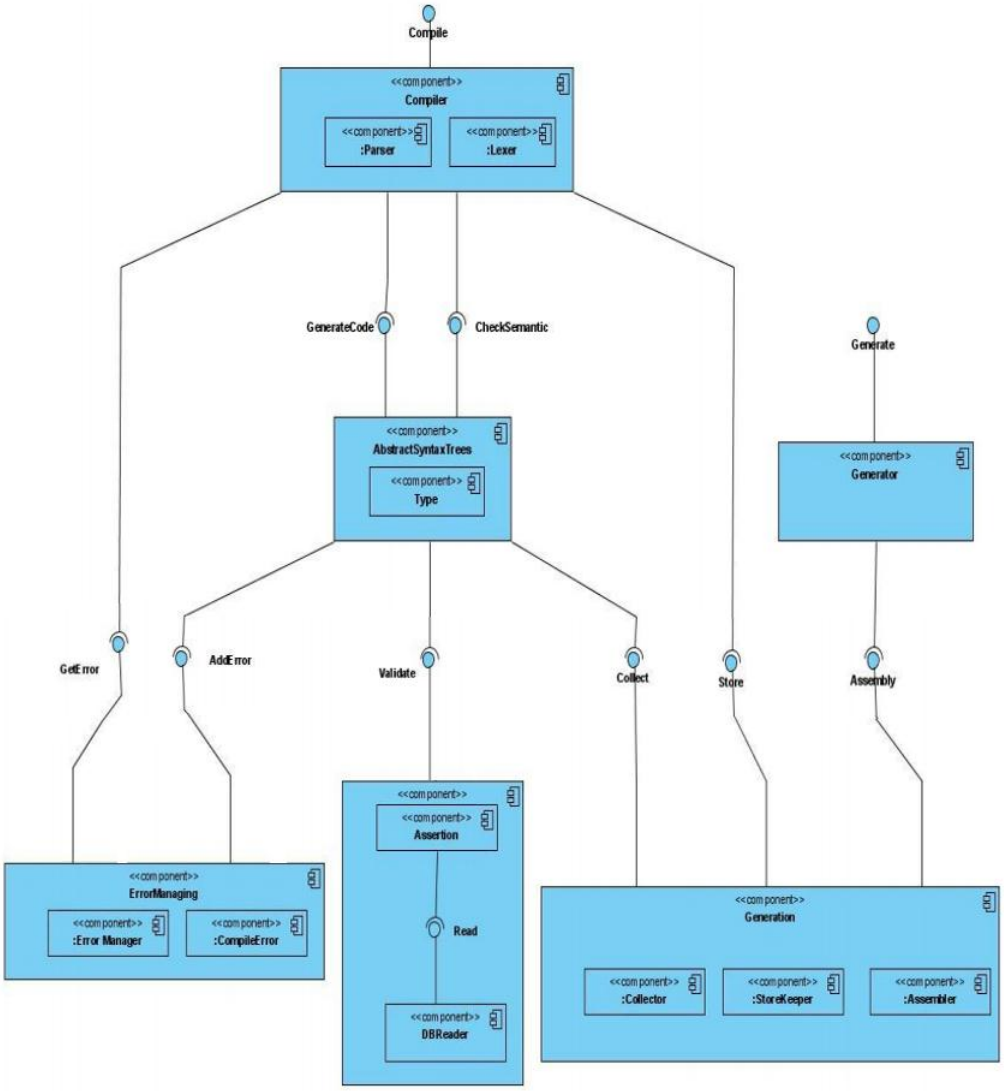

Ilustración 10: Diagrama de Componentes

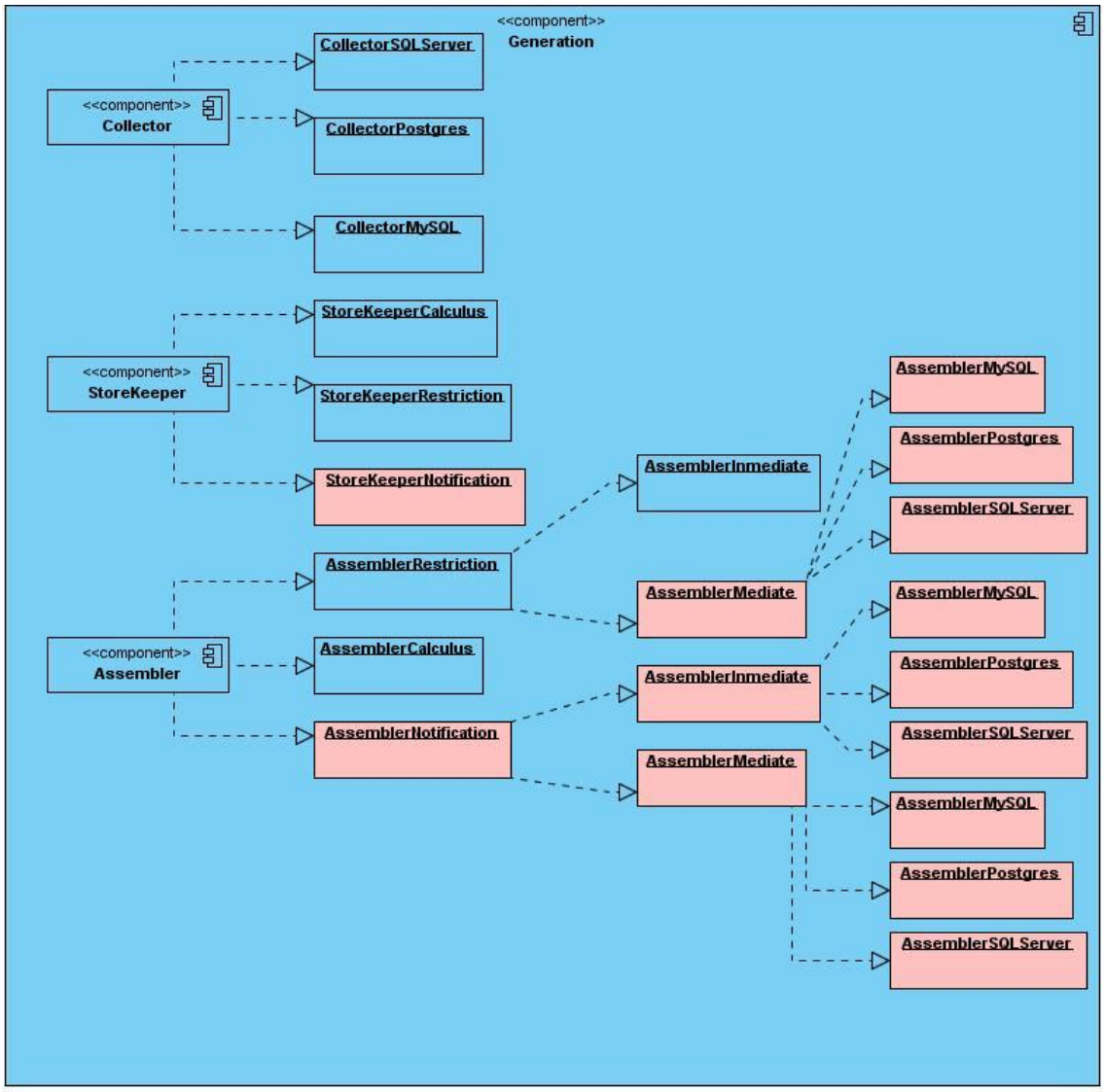

Ilustración 11: Sección del Diagrama de Componentes LPT v1.5

CAPÍTULO 3

En el presente capítulo se recogen los principales aspectos para la implementación del subsistema modo diferido del LPT-SQL, así como el diseño e implementación del caso de estudio "Sistema de Inventarios" para la puesta a prueba de la herramienta en cuestión.

3.1 Modificaciones en la Base de Datos LPT-SQL

Al implementar el subsistema diferido para reglas de tipo notificación se hace necesario la creación de la tabla *Notificaciones*, la cual almacenará las notificaciones generadas al ejecutarse un conjunto de operaciones sobre los datos. Además, para las reglas de negocio tipo restricción, se adicionará la tabla *rn_restriccion*, encargada de almacenar las posibles reglas que se han infringidos en un conjunto de operaciones para su posterior análisis de integridad. La tabla *LRV* contiene las reglas que son infringidas al concluir la transacción, pues la vista de integridad contiene un conjunto de filas. A continuación se muestran las sentencias, en SQL estándar, a ejecutar para la creación de las tablas mencionadas anteriormente:

```
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));

CREATE TABLE RN_RESTRICCION (NOMBRE_RN CHARACTER (100) NOT NULL, MENSAJE CHARACTER (100) NOT NULL, PRIMARY KEY (NOMBRE_RN));

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

Bajo un modo diferido las operaciones sobre los datos deben realizarse a través de las transacciones, para que cuando se hagan realidad las violaciones de integridad se puedan deshacer las mismas.

Cuando en la base de datos del negocio es activada por primera vez una regla en modo diferido, a través del LPT-SQL, se generan automáticamente los

recursos necesarios para el correcto funcionamiento de la misma, entre los que se encuentran: *pmediaterestriccion* (procedimiento almacenado responsable de la verificación de la vista de integridad de cada una de las posibles reglas violadas, almacenadas en la tabla *rn_restriccion*), *transaccionrestriccion* (estructura que deberá tener un procedimiento para la correcta ejecución del SI con reglas tipo restricción), *transaccionnotificacion* (estructura que deberá tener un procedimiento para la ejecución de las operaciones con reglas de tipo notificación), así como, las tablas *rn_restriccion*, *lrv* y *notificaciones*.

A continuación se muestra el código del procedimiento almacenado **pmediaterestriccion()** en lenguaje plpgsql, utilizado por las transacciones para realizar el chequeo en modo diferido de las reglas de negocio:

```
create or replace function pmediaterestriccion()
returns void as
$BODY$
DECLARE SQLString VARCHAR(500);
DECLARE Nombre_RN character varying (500);
DECLARE Rules_cursor CURSOR FOR SELECT * FROM rn_restriccion;
DECLARE sujetos bigint default 0;
BEGIN
delete from lrv;
for vare in Rules_cursor loop
        SQLString:=concat('v',lower(vare.nombre_rn));
        execute concat('select count(*) from ',SQLString) into sujetos;
        if(sujetos<>0) then
                INSERT INTO lrv VALUES (vare.nombre_rn ,sujetos) ;
        end if;
end loop;
return ;
END;
$BODY$ LANGUAGE plpgsql;
```

En este procedimiento se consulta la tabla *rn_restriccion* para obtener la lista de los identificadores de aquellas reglas que, en un momento de dado de la

transacción, incumplieron con la integridad de sus datos. Luego son verificadas cada una de las vistas de integridad asociadas a las reglas obtenidas para verificar si esas reglas aún están siendo violadas, en caso de ser cierto se almacenan en la tabla *lrv* el identificador de dicha regla y la cantidad de sujetos que la infringen.

El programador del sistema de información deberá modificar la estructura de las transacciones generadas automáticamente, adicionando las operaciones a realizar a través del SI. La estructura de dichas transacciones, para su utilización en reglas de negocio tipo restricción, utilizando el lenguaje plpgsql, será la siguiente:

```
CREATE OR REPLACE FUNCTION transaccionrestriccion()
RETURNS void AS
$$
DELETE FROM RN_RESTRICCION;
BEGIN
/*Conjunto de operaciones a realizar sobre los datos del negocio*/
EXECUTE pmediaterestriction();
IF(EXISTS (SELECT * FROM lrv)) THEN
        RAISE ERROR 'Error en las operaciones';
END IF;
END;
$$ LANGUAGE plpgsql;
```

Para las reglas de negocio tipo notificación, la estructura de las transacciones quedaría de la siguiente forma:

```
CREATE OR REPLACE FUNCTION transaccionnotificacion()
RETURNS void AS
$$
BEGIN
/*Conjunto de operaciones a realizar sobre los datos del negocio*/
IF(EXISTS (SELECT * FROM NOTIFICACIONES)) THEN
        Raise exception 'Error';
```

```
END IF;
END;
$$ LANGUAGE plpgsql;
```

3.2 Recursos y métodos implementados

Los recursos implementados en el subsistema para lograr la implementación diferida son:

- Para las reglas de negocio tipo restricción se utiliza el método **createFunction():** Método encargado de crear el mecanismo para lograr, que cuando sea lanzado algún evento de la lista de eventos no se ejecuten los disparadores de los mismos, sino que sea adicionada a la tabla *rn_restriccion* la regla vinculada a dicho evento para su posterior análisis de integridad.
- Para las reglas de negocio tipo notificación uno de los principales métodos utilizados es el **createFunction()**, logrando que cuando un evento de la lista de eventos lance una notificación, esta sea insertada en la tabla *notificaciones*.

3.3 Características generales del traductor LPT-SQL v1.5

Como se referenció en epígrafes anteriores la presente investigación constituye una continuación de investigaciones anteriores. Una de las principales características adicionadas a la versión actual de la herramienta es la posibilidad de seleccionar el enfoque bajo el cual serán generadas las reglas de negocio, permitiendo la generación de sentencias SQL teniendo en cuenta el Gestor de Bases de Datos seleccionado.

Al iniciar la aplicación, el usuario tiene la posibilidad de cargar un repositorio anteriormente creado o adicionar uno nuevo. Se muestra además las características de cada una de las reglas almacenadas en el repositorio (estado, enfoque, etc.).

Cabe destacar que para la generación de una regla, ya sea bajo el modo inmediato o el diferido, se debe de realizar la conexión con la base de datos y extraer la información de los recursos existentes en la misma por parte del usuario al iniciar la aplicación. Posterior a la compilación y generación de una regla, esta cuenta con todas las características necesarias para ser activada para uso en la base de datos del negocio, en caso de error se notifica al usuario y se aborta la generación o activación.

Cuando una regla está activa, significa que sus recursos están insertados en a base de datos y ante cualquier operación sobre los datos las reglas serán tomadas en cuenta.

Cuando una regla es desactivada, los recursos vinculados a ella son eliminados de la base de datos, para evitar la presencia innecesaria de los mismos y la incongruencia en caso de que se desee activar nuevamente la regla.

Cuando se va a realizar la conexión con la base de datos física, la aplicación brinda, en el caso de utilizarse el Gestor de Base de Datos SQLServer, un vínculo directo a la creación y configuración de un origen de datos ODBC.

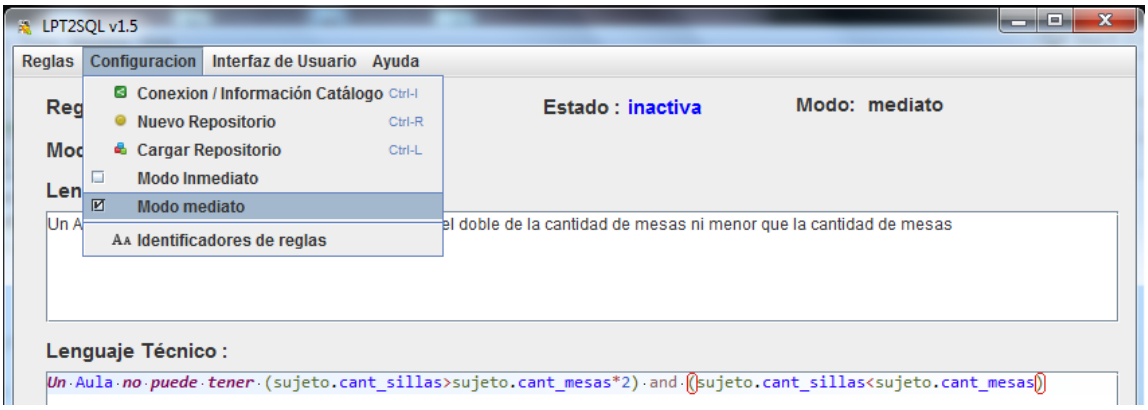

Ilustración 12: Interfaz general de la aplicación con la selección del enfoque

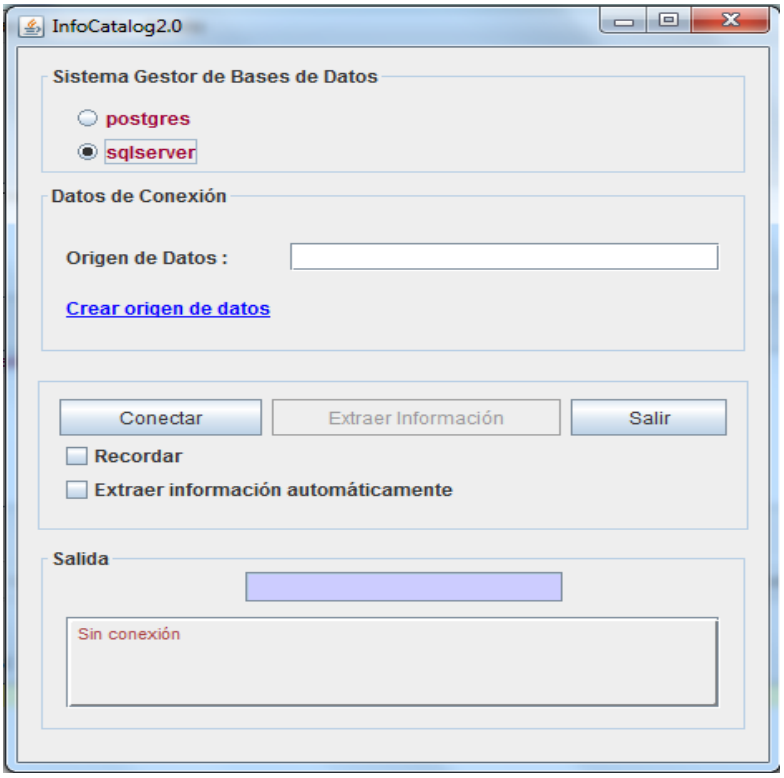

Ilustración 13: Interfaz de conexión y extracción de información de la base de datos física

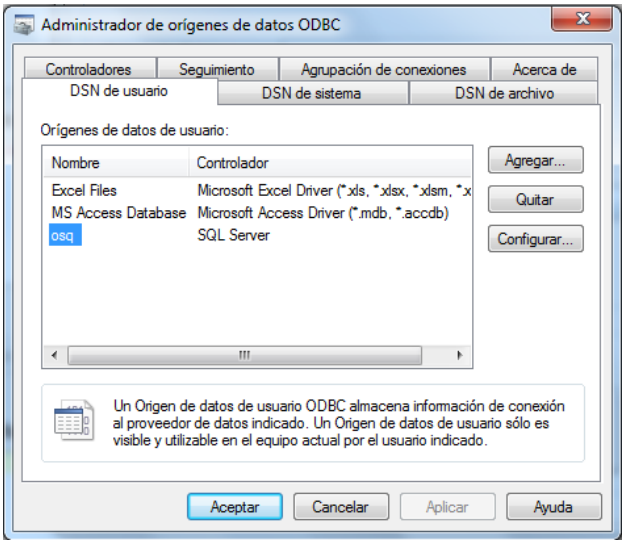

Ilustración 14: Creación del origen de datos ODBC para la conexión con SQLServer

3.4 Diseño e implementación del caso de estudio "Sistema de Inventarios"

Para la puesta a prueba del subsistema modo diferido del LPT-SQL v1.5 se diseña e implementa un caso de estudio, como herramienta de análisis del funcionamiento de la implementación de las reglas del negocio tipo restricción y notificación en bases de datos relacionales.

La descripción del negocio del caso de estudio es la siguiente:

Se desea implementar un Sistema de Información para el área de mantenimiento de la UCLV, con el propósito de procesar y almacenar toda la información referente a los inventarios de cada una de las facultades de la Universidad. El sistema deberá permitir adicionar una Facultad, aula, departamento y laboratorio.

De la facultad se recogerá el nombre de la misma y el nombre del Decano. Para el caso del aula se tendrá en cuenta la facultad a la que pertenece, además de la cantidad de mesas/pupitres y sillas existentes en las mismas. Se debe tener en cuenta que en el caso de que en el aula existan mesas, la cantidad de sillas no debe ser menor que la cantidad de mesas y no deben ser mayor que el doble de la cantidad de dichas mesas.

Para los departamentos, la facultad y el jefe de departamento. La cantidad de sillas no debe ser menor que la cantidad de mesas existentes en ellos. En los laboratorios se debe garantizar que exista una relación de 1 silla por mesa, permitiendo el uso total de los medios por parte de los estudiantes y profesores, registrando además, la facultad y el responsable del laboratorio.

De cada silla, mesa o pupitre se recogerá el número de inventario del mismo, así como el aula, laboratorio o departamento al que pertenece.

El sistema deberá permitir adicionar un área, eliminar un área y actualizar los medios básicos de cada una de las áreas registradas, así como realizar comprobaciones de cuadre de inventarios en un momento dado.

3.4.1 Reglas del Negocio presentes

A partir de la problemática anterior se derivan las siguientes reglas de negocio:

- RN#1: Un Aula no puede tener cantidad de sillas mayor que el doble de la cantidad de mesas ni menor que la cantidad de mesas.
- RN#2: Un Laboratorio no puede tener la cantidad de mesas mayor que la cantidad de sillas.
- RN#3: Un Departamento no puede tener la cantidad de sillas menor que la cantidad de mesas.
- RN#4: Notificar "Departamento duplicado" si se adicionan dos departamentos de Matemática.

Mediante la utilización de la herramienta LPT-SQL v1.5 se obtuvieron las siguientes reglas de negocio en lenguaje LPT:

- RN#1: Un Aula no puede tener (sujeto.cant_sillas>sujeto.cant_mesas*2) OR (sujeto.cant_sillas<sujeto.cant_mesas).
- RN#2: Un Laboratorio no puede tener sujeto.cant_mesas>sujeto.cant_sillas.
- RN#3: Un Departamento no puede tener sujeto.cant_sillas<sujeto.cant_mesas.
- RN#4: Notificar 'Departamento duplicado' si sizeof(Departamento.nombre_departamento='Matemática')

3.4.2 Diagrama de Casos de Uso

El caso de estudio en análisis presenta entre sus principales casos de uso:

- Adicionar Facultad: El usuario tiene la posibilidad de adicionar una facultad al sistema. La facultad es considerada el área principal para el control de inventarios.
- Adicionar Aula: El usuario tiene la posibilidad de adicionar un aula a una facultad dada, introduciendo la cantidad de inventarios presentes en la misma.
- Editar Inventario Aula: El usuario puede realizar modificaciones al inventario existente un en aula determinada.
- Adicionar Laboratorio: El usuario puede adicionar un laboratorio a una facultad dada, introduciendo las existencias físicas de cada medio básico del área.
- Actualizar Inventario Laboratorio: El usuario tiene la posibilidad de modificar las existencias físicas de medios en el área.
- Adicionar departamento: El usuario puede adicionar un departamento a una facultad dada, introduciendo las cantidades de cada uno de los medios.
- Actualizar Inventarios Departamento: El usuario puede modificar las cantidades de los medios existentes en el área.
- Chequear integridad de Inventarios: Caso de uso de mayor peso dentro del sistema, pues permite realizar el chequeo de modo diferido de la integridad de las reglas de negocio asociadas a los datos.

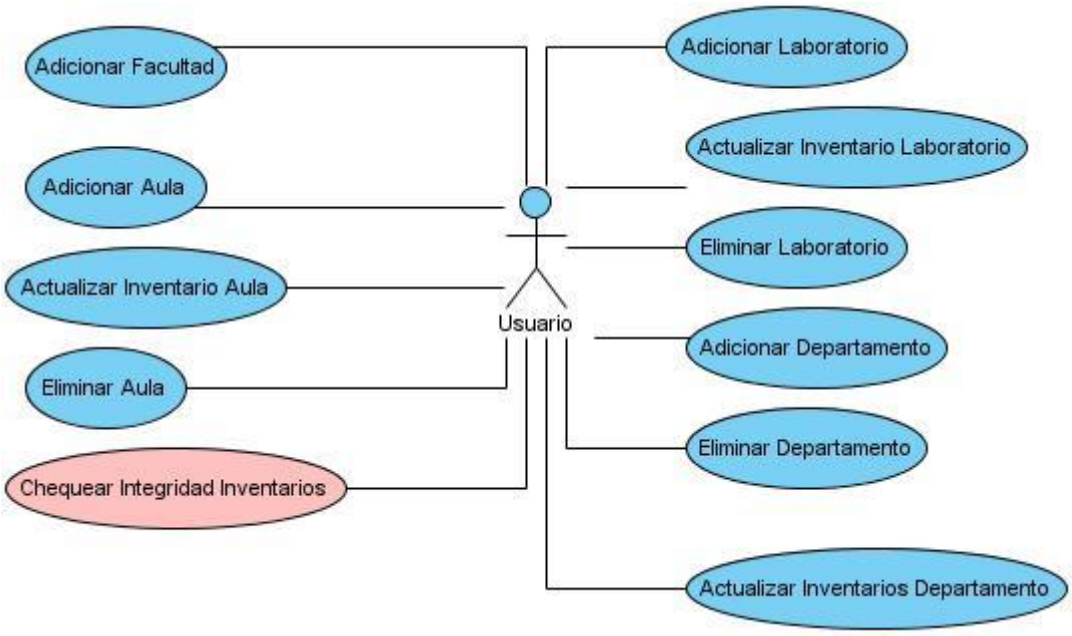

Ilustración 15: Diagrama de Casos de Uso Sistema Inventarios

3.4.2.1 Diagrama de Clases

Para la implementación de dicho sistema se hace necesario la creación de 4 clases principales (Facultad, Aula, Departamento, Laboratorio), encargadas de la gestión y control de los inventarios asociados a las áreas asociadas a las mismas.

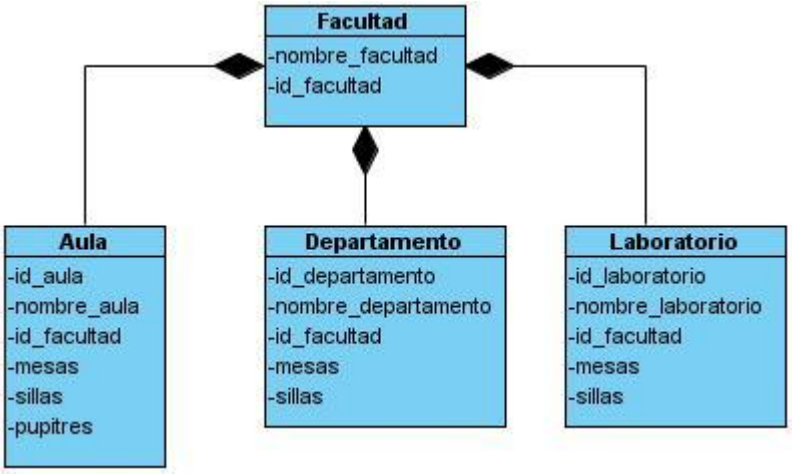

Ilustración 16: Diagrama de Clases Sistema Inventarios

CAPÍTULO 4

4.1 Planificación basada en un método de estimación

El método de casos de uso permite documentar los requisitos de un sistema en términos de actores y casos de uso, proporcionando uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

La estimación mediante el análisis de Puntos de Casos de Uso, consiste en un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para que luego a partir de esos factores se pueda contabilizar el tiempo total estimado para el proyecto en cuestión.

El método de Casos de Uso permite documentar los requerimientos de un sistema en términos de Actores y Casos de Uso. Un Actor típicamente representa un rol que puede ejercer un usuario humano u otro sistema externo que interactúa con el sistema bajo análisis. Un Caso de Uso representa una funcionalidad que modela una interacción Actor-Sistema y visto como una secuencia de acciones que uno o más actores llevan a cabo en el sistema para obtener un resultado de valor observable.

4.1.1 Cálculo de los Puntos de Casos de Uso sin Ajustar

El primer paso para la estimación consiste en el cálculo de los Puntos de Casos de Uso sin ajustar. Este valor, se calcula a partir de la siguiente ecuación:

PCU = FPA + FPCU donde,

- PCU: Puntos de Casos de Uso sin ajustar.
- FPA: Factor de Peso de los Actores sin ajustar.
- FPCU: Factor de Peso de los Casos de Uso sin ajustar.

Factor de Peso de los Actores sin ajustar (FPA)

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Actores se establece teniendo en cuenta en primer lugar si se trata de una persona o de otro sistema, y en segundo lugar, la forma en la que el actor interactúa con el sistema. Los criterios se muestran en la siguiente tabla:

Tipo de Actor	**Descripción**	**Factor de peso**
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación *ApplicationProgrammin Interface* (API).	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3

En nuestro caso el usuario constituye el tipo de actor, este es de tipo complejo, ya que se trata de una persona utilizando el sistema mediante una interfaz gráfica, al cual se le asigna un peso 3.

Luego, el factor de peso de los actores sin ajustar resulta:

FPA = 1 x 3 = 3.

Factor de Peso de los Casos de Uso sin ajustar (FPCU)

Este valor se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo. Los criterios se muestran en la siguiente tabla:

Tipo de Caso de Uso.	**Descripción**	**Factor de peso**
Simple	El Caso de uso tiene de 1 a 3 transacciones.	5
Medio	El Caso de uso tiene de 4 a 7 transacciones.	10
Complejo	El Caso de uso tiene más de 8 transacciones.	15

Cada uno de los casos de uso:

- Seleccionar Modo.
- Generar Regla de Negocio.

El primer caso de uso posee una cantidad máxima de 1 transacción por lo que el factor de peso es de 5. Para el caso de uso Generar Regla de Negocio posee un máximo de 7 transacciones por lo que posee un factor de peso de 10

Se tiene entonces 1 caso de uso tipo simple (peso 5) y un caso de uso tipo medio (peso 10) con lo cual el factor de peso de los casos de uso sin ajustar resulta:

FPCU1 = 1x 5 = 5

FPCU2 = 1x 10 = 10

Finalmente, los Puntos de Casos de Uso sin ajustar resultan

PCU = FPA + FPCU = 3 + 15 = 18

4.1.2 Cálculo de Puntos de Casos de Uso ajustados

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe ajustar este valor mediante la siguiente ecuación: **PCUA = PCU x FCT x FA** donde:

- **PCUA**: Puntos de Casos de Uso ajustados
- **PCU**: Puntos de Casos de Uso sin ajustar
- **FCT**: Factor de complejidad técnica
- **FA**: Factor de ambiente

Factor de complejidad técnica (FCT)

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante. En la siguiente tabla se muestra el significado y el peso de cada uno de estos factores con el valor que representa el sistema:

Factor	**Descripción**	**Peso**	**Valor asignado**	**Comentario.**
T1	Sistema distribuido.	2	0	El sistema es centralizado.
T2	Objetivos de *performance* o	1	1	La velocidad depende de

	tiempo de respuesta.			la cantidad de usuarios conectados.
T3	Eficiencia del usuario final.	1	1	No es necesario un usuario con eficiencia.
T4	Procesamiento interno complejo.	1	1	No hay cálculos complejos.
T5	El código debe ser reutilizable.	1	0	No tiene que ser reutilizable.
T6	Facilidad de instalación.	0.5	1	Es de fácil instalación.
T7	Facilidad de uso.	0.5	3	Normal.
T8	Portabilidad	2	0	El sistema no necesita instalación.
T9	Facilidad de cambio.	1	3	Se necesita un costo moderado de mantenimiento.
T10	Concurrencia	1	0	No hay concurrencia.
T11	Incluye objetivos especiales de seguridad.	1	3	Seguridad normal.
T12	Provee acceso directo a	1	0	No posee acceso directo a

	terceras partes.			terceras partes.
T13	Se necesitan facilidades especiales de entrenamiento a usuarios.	1	4	Pocos usuarios internos, sistema fácil de usar.

El Factor de complejidad técnica se calcula mediante la siguiente ecuación:

FCT = 0.6 + 0.01 x Σ (Peso$_i$ x Valor asignado$_i$)

El Factor de complejidad técnica resulta:

FCT = 0.6 + 0.01 x 15 = 0.75

Factor de ambiente (FA)

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del Factor de ambiente. El cálculo del mismo es similar al cálculo del Factor de complejidad técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5.

En la siguiente tabla se muestra el significado y el peso de cada uno de estos factores y el valor asignado con respecto al sistema.

Factor	**Descripción**	**Peso**	**Valor asignado**	**Comentario**
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	2	No se está bastante familiarizado.

E2	Experiencia con la aplicación.	0.5	2	Poca.
E3	Experiencia en orientación a objetos.	1	4	Suficiente.
E4	Capacidad del analista líder.	0.5	3	Capacidad moderada.
E5	Motivación.	1	5	Alta.
E6	Estabilidad de los requerimientos.	2	2	No se esperan cambios.
E7	Personal *part-time.*	-1	0	El personal es de tiempo completo.
E8	Dificultad del lenguaje de programación.	-1	3	Se utiliza JavaScript como lenguaje.

El Factor de ambiente se calcula mediante la siguiente ecuación:

$FA = 1.4 - 0.03 \times \Sigma (Peso_i \times Valor\ asignado_i)$

$FA = 1.4 - 0.03 \times 15.5$

$FA = 1.4 - 0.465$

$FA = 0.93$

Finalmente, los Puntos de Casos de Uso ajustados resultan:

PCUA = 18 * 0.75 * 0.93= 12.55

4.1.3 De los Puntos de Casos de Uso a la estimación del esfuerzo.

Karner originalmente sugirió que cada Punto de Casos de Uso requiere 20 horas-hombre. Posteriormente, surgieron otros refinamientos que proponen valores según los siguientes criterios:

Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6, y se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.

- Si el total es 2 o menos, se utiliza el **factor de conversión** 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre.
- Si el total es 3 o 4, se utiliza el **factor de conversión** 28 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 28 horas-hombre.
- Si el total es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto. El esfuerzo en horas-hombre viene dado por:

E = PCUA x FC donde,

- **E**: esfuerzo estimado en horas-hombre
- **PCUA**: Puntos de Casos de Uso ajustados
- **FC**: factor de conversión

Por lo que la estimación será.

E= 12.55 * 20 = 251 Horas-Hombre

Este método proporciona una estimación del esfuerzo en horas-hombre contemplando sólo el desarrollo de la funcionalidad especificada en los casos de uso (Programación / Implementación).Para una estimación más completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software.

4.1.4 Estimación del tiempo de desarrollo del proyecto.

El Esfuerzo total estimado en el epígrafe anterior es entonces E (total)= 251 HH. A partir de este esfuerzo se pueden realizar otras estimaciones como por ejemplo el tiempo de desarrollo aproximado del proyecto que se calcularía de la siguiente manera:

TDes = E(total)**/CH** donde,

- **TDes**: tiempo de desarrollado del proyecto
- **E**(total)**:** esfuerzo total
- **CH:** es la cantidad de hombres que desarrollan el proyecto.

Por lo tanto para un desarrollador la estimación del tiempo de desarrollo este proyecto es:

TDes= 251 HH / 1 Hombre

TDes= 251 Horas

El resultado de esta estimación se puede brindar también en días o meses solo haciendo la conversión correspondiente.

Hombres – Mes x 160	Hombres – Hora

Hombres – Mes x 20	Hombres – Día
Hombres – Mes / 12	Hombres – Año

4.2 Prueba del Subsistema modo diferido del LPT-SQL v1.5

Para la puesta a prueba del subsistema modo diferido del LPT-SQL v1.5 se generaron reglas de negocio basadas en un caso de estudio (Capítulo 4) creado por la autora de la presente investigación, el cual permitió la utilización de ambos enfoques de chequeo (Inmediato y Diferido).

4.2.1 Reglas generadas

Lenguaje Natural:

- RN#1: Un Aula no puede tener cantidad de sillas mayor que el doble de la cantidad de mesas ni menor que la cantidad de mesas.
- RN#2: Un Laboratorio no puede tener la cantidad de mesas mayor que la cantidad de sillas.
- RN#3: Un Departamento no puede tener la cantidad de sillas menor que la cantidad de mesas.
- RN#4: Notificar "Departamento duplicado" si se adicionan dos departamentos de Matemática.

Lenguaje LPT:

- RN#1: Un Aula no puede tener (sujeto.cant_sillas>sujeto.cant_mesas*2) OR (sujeto.cant_sillas<sujeto.cant_mesas).
- RN#2: Un Laboratorio no puede tener sujeto.cant_mesas>sujeto.cant_sillas.

- RN#3: Un Departamento no puede tener sujeto.cant_sillas<sujeto.cant_mesas.
- RN#4: Notificar 'Departamento duplicado' si sizeof(Departamento.nombre_departamento='Matemática')

4.2.2 Recursos Generados por LPT-SQL v1.5

Al utilizar la herramienta para generar las reglas de negocio asociadas al caso de estudio se generaron diversos recursos en la base de datos:

- Tablas
    1. *LRV*: tabla para almacenar las reglas, que al culminar las operaciones violan sus vistas de integridad correspondientes.
    2. *RN_RESTRICCION*: tabla que almacena las reglas que en algún momento dado infringieron su vista de integridad.
    3. *NOTIFICACIONES*: tabla que almacena las reglas de tipo notificación que incumplieron con la lógica del negocio.
- Funciones disparadoras
    1. Frn1_1: función encargada de realizar el chequeo del cumplimiento de la regla de negocio #1, en caso de violación adiciona la regla a *rn_restriccion* como posible regla violada. **(Anexo 1)**
    2. Frn2_1: función encargada de realizar el chequeo del cumplimiento de la regla de negocio #2, en caso de violación adiciona la regla a *rn_restriccion* como posible regla violada. **(Anexo 2)**
    3. Frn3_1: función encargada de realizar el chequeo del cumplimiento de la regla de negocio #3, en caso de violación adiciona la regla a *rn_restriccion* como posible regla violada. **(Anexo 3)**
    4. Frn4_1: función encargada de realizar el chequeo del cumplimiento de la regla de negocio #4, en caso de violación

adiciona la regla a la tabla *notificaciones* como posible regla violada. **(Anexo 4)**

- Vistas de Integridad
    1. Vrn1: vista de integridad asociada a la regla de negocio #1. **(Anexo 5)**
    2. Vrn2: vista de integridad asociada a la regla de negocio #2. **(Anexo 6)**
    3. Vrn3: vista de integridad asociada a la regla de negocio #3. **(Anexo 7)**
    4. Vrn4: vista de integridad asociada a la regla de negocio #4. **(Anexo 8)**

Cada tabla existente en la base de datos que se encuentre asociada a una o varias reglas de negocio contarán con un disparador, encargado de ejecutar la función disparadora correspondiente. El código de los recursos generado por la herramienta se encuentra en los Anexos del trabajo.

4.2 Pruebas de aceptación o caso de prueba

Para la validación de las principales funcionalidades del sistema se hace uso de la técnica de pruebas de caja negra. Dicha técnica no hace uso de la estructura del código fuente de la aplicación, sino del funcionamiento lógico de la misma. Las pruebas de caja negra se centran principalmente en lo que se quiere de un módulo o sección específica de un software, es decir, es una manera de encontrar casos específicos en ese modulo que atiendan a su especificación.

Ilustración 17: Esquema de las pruebas de caja negra

Según (Rogers, 2005) las pruebas de caja negra permiten identificar problemas tales como:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Los casos de prueba son diseñados a través de distintos escenarios para comprobar que los casos de uso del sistema han sido correctamente implementados. Un caso de uso no se considera terminado hasta que las pruebas no sean satisfactorias, en caso de que fallen varias pruebas, se deben indicar el orden de prioridad de las fallas encontradas e intentar corregir las mismas.

A continuación se muestran algunas de las pruebas de aceptación realizadas para la validación de la aplicación:

Caso de prueba de aceptación	
No. de la tarea: A_1	**Nombre del caso de uso:** Adicionar facultad
Nombre de la persona que realiza la tarea: Lisset Torres Vazquez	
Descripción de la prueba: Se ejecuta la prueba y se verifica que se muestre toda la información ordenada correctamente así como el muestreo de los nuevos datos insertados desde la aplicación	

Condiciones de ejecución: -
Entrada / Pasos de ejecución: Cuando el sistema muestre la interfaz para adicionar una facultad el usuario deberá introducir el nombre de la misma y deberá aceptar la operación para persistir los datos en la base de datos.
Resultado esperado: Que se muestren las facultades adicionadas en el sistema.
Evaluación de la prueba: Satisfactoria

Tabla 1: Caso de prueba para Adicionar Facultad

Caso de prueba de aceptación	
No. de la tarea: A_2	**Nombre del caso de uso:** Adicionar Aula
Nombre de la persona que realiza la tarea: Lisset Torres Vazquez	
Descripción de la prueba: Se ejecuta la prueba y se verifica que se muestre toda la información ordenada correctamente así como el muestreo de los nuevos datos insertados desde la aplicación	
Condiciones de ejecución: Que se hayan adicionado las facultades a las cuales pertenecen las aulas.	

Entrada / Pasos de ejecución: Cuando el sistema muestre la interfaz para adicionar un aula el usuario deberá, seleccionar la facultad a la cual pertenece el aula. Deberá además introducir la cantidad de pupitres en caso de existir, la cantidad de mesas y de sillas por mesas.
Resultado esperado: Que se muestren las aulas adicionadas en el sistema con el inventario actualizado.
Evaluación de la prueba: Satisfactoria

Tabla 2: Caso de prueba para Adicionar Aula

Caso de prueba de aceptación	
No. de la tarea: A_3	**Nombre del caso de uso:** Adicionar Laboratorio
Nombre de la persona que realiza la tarea: Lisset Torres Vazquez	
Descripción de la prueba: Se ejecuta la prueba y se verifica que se muestre toda la información ordenada correctamente así como el muestreo de los nuevos datos insertados desde la aplicación	
Condiciones de ejecución: Que se hayan adicionado las facultades a las cuales pertenecen los laboratorios.	
Entrada / Pasos de ejecución: Cuando el sistema muestre la interfaz para adicionar un laboratorio el usuario deberá, seleccionar la facultad a la cual	

pertenece el mismo. Deberá además introducir la cantidad de mesas y de sillas por mesas.
Resultado esperado: Que se muestren los laboratorios adicionadas en el sistema con el inventario actualizado.
Evaluación de la prueba: Satisfactoria

Tabla 3: Caso de prueba para Adicionar Laboratorio

Caso de prueba de aceptación	
No. de la tarea: A_4	**Nombre del caso de uso:** Adicionar Departamento
Nombre de la persona que realiza la tarea: Lisset Torres Vazquez	
Descripción de la prueba: Se ejecuta la prueba y se verifica que se muestre toda la información ordenada correctamente así como el muestreo de los nuevos datos insertados desde la aplicación	
Condiciones de ejecución: Que se hayan adicionado las facultades a las cuales pertenecen los departamentos.	
Entrada / Pasos de ejecución: Cuando el sistema muestre la interfaz para adicionar un departamento el usuario deberá, seleccionar la facultad a la cual pertenece el mismo. Deberá además introducir la cantidad de mesas y de sillas por mesas.	

Resultado esperado: Que se muestren los departamentos adicionadas en el sistema con el inventario actualizado.
Evaluación de la prueba: Satisfactoria

Tabla 4: Caso de prueba para Adicionar Departamento

Caso de prueba de aceptación	
No. de la tarea: A_5	**Nombre del caso de uso:** Verificar integridad inventario
Nombre de la persona que realiza la tarea: Lisset Torres Vazquez	
Descripción de la prueba: Se ejecuta la prueba y se verifica que se muestre toda la información ordenada correctamente así como el muestreo de los nuevos datos insertados desde la aplicación	
Condiciones de ejecución: Que se hayan realizado operaciones sobre los inventarios de las áreas de una facultad	
Entrada / Pasos de ejecución: El usuario accederá a través del menú Facultad a la opción Verificar Integridad Inventario para realizar el chequeo de las operaciones realizadas. En caso de operaciones incorrectas el sistema mostrará al usuario un mensaje de error con las violaciones realizadas en las operaciones sobre la base de datos del SI.	
Resultado esperado: En caso de violación mostrar un mensaje de error con	

las violaciones realizadas.
Evaluación de la prueba: Satisfactoria

Tabla 5: Caso de prueba para Chequear Integridad Inventario

CONCLUSIONES

1. La valoración y análisis de la implementación de las reglas de restricción y notificación de modo diferido permite utilizar estos fundamentos teóricos para extenderlos, de modo que constituyan las bases para el diseño del subsistema.
2. El diseño del subsistema para la herramienta LPT-SQL permite que se adicione a ella las funcionalidades de la generación de modo diferido para las reglas de restricción y notificación.
3. Se implementó la generación diferida de estas reglas de negocio, así se adiciona este modo de trabajo a la herramienta LPT-SQL.
4. El desarrollo de una aplicación de ejemplo demuestra el uso del subsistema para la implementación diferida.

RECOMENDACIONES

1. Extender la librería InfoCatalogo v2.0 para que permita la extracción de los procedimientos almacenados del Gestor de Bases de Datos Microsoft SQLServer como recursos presentes en la base de datos del negocio. Actualmente solo brinda la posibilidad de extraer los recursos: FUNCTION, VIEW y TRIGGER.
2. Realizar el diseño e implementación del subsistema modo diferido e inmediato en la herramienta LPT-SQL para el Gestor de Bases de Datos MySQL, ya que actualmente solo permite su ejecución con los gestores Microsoft SQLServer y PostgreSQL.

REFERENCIAS

Andreescu, A. a. M. M., 2012. *Perspectives on the role of business rules in dabase.* s.l.:s.n.

Ashwell, R., 2006. *Define Business Rules, Editor^Editors, CRaG Systems..* s.l.:s.n.

Baisley, D. S., 2008. *What Are the Possibilities?.* s.l.:Business Rules Journal..

Boggiano, M., 2014. *REGLAS DE NEGOCIO DESDE LA PERSPECTIVA DE LOS DATOS EN BASES DE.* Santa Clara, Cuba: s.n.

Boley, H. A. P. a. O. S., 2010. *RuleML 1.0: the overarching specification of web rules.* s.l.:Lecture Notes in Computer Science.

Calderón Solis, A., 2011. *Traductor LPT-SQL para reglas de negocio en Bases de Datos Relacionales,* s.l.: s.n.

Cover, R., 2002. *Business Rules Markup Language.* [En línea] Available at: http://xml.coverpages.org/brml.html

Date, C., 2001. *Constraints & Predicates: A Brief Tutorial (Part 2). Business Rules Journal. 2.* [En línea] Available at: http://www.BRCommunity.com

Demuth, B., 2004. *THE DRESDEN OCL TOOLKIT AND ITS ROLE IN INFORMATION SYSTEMS DEVELOPMENT,* s.l.: s.n.

Gennick, J., 2006. *SQL pocket guide.* s.l.:s.n.

Hay, D. a. K. H., 2000. *Defining businessrules- what are they really? Technical Report.* [En línea] Available at: http://businessrulesgroup.org/first_paper/br01c0.htm

Lowenthal, B., 2005. *Rule Enabling Applications with Oracle Business Rules..* [En línea] Available at: http://webdelprofesor.ula.ve/ingenieria/ibc/cleiOnRn.pdf

Martínez Hernández, J., 2010. *Introduciendo semántica en un proceso de desarrollo software a través de reglas de negocio.* s.l.:Escuela técnica superior de ingenieros de telecomunicación Universidad politécnica de Madrid, Madrid.

Matei, I., 2006. *Implementing Business Rules with Software Agents..* s.l.:s.n.

Melton, J. a. S. A. R., 2002. *SQL1999: underestanding relational languaje components, Morgab Kaufmann..* s.l.:s.n.

MES, 2007. *Plan de Estudio D INGENIERÍA INFORMÁTICA PRESENCIAL.* s.l.:s.n.

Morgan, T., 2002. *Defining Business Rules in Business Rules and Information Systems: Aligning IT with Business Goals, Sección del Libro Addison Wesley.* s.l.:s.n.

Mota, S., 2005. *Bases de Datos Activas.* s.l.:s.n.

MSDN, 2008. *MSDN LIBRARY VisualStudio 2008.* s.l.:Microsoft Corporation.

Oppel A., S. R., 2008. *SQL: a biginner's guide.* s.l.:McGraw-Hill Profesional.

Paschke, A. P. V. a. F. S., 2011. *Standards for complex event processing and reaction rules, in Rule-Based Modeling and Computing on the Semantic Web.* s.l.:s.n.

Pérez Alonso, A., 2010. *Reglas de Negocio en Bases de Datos Relacionales.* s.l.:s.n.

Rogers, P., 2005. *Ingeniería de Software un Enfoque Práctico,* s.l.: Editorial McGraw-Hill, Madrid..

Ross, R., 1997. *The Business Rule Book: Classifying, Defining and Modeling Rules.* s.l.:s.n.

Ross, R. G., 2009. *Business Rule Concepts ~ Getting to the Point of Knowledge.* s.l.:s.n.

Ross, R. G., 2010. *What Is a Business Rule?.* s.l.:s.n.

Von Halle, B. a. R. R., 2002. *Business rules applied: building better systems using the business rules approac.* s.l.:s.n.

ANEXOS

Anexo 1: Código generado por el LPT-SQL para la función frn1_1()

```
CREATE OR REPLACE FUNCTION frn1_1()
  RETURNS trigger AS
$BODY$
BEGIN
 if(EXISTS(SELECT * FROM public."vrn1" a WHERE (a."id_aula"=NEW."id_aula") AND (a."nombre_aula"=NEW."nombre_aula")) ) then
        if not (exists(select * from rn_restriccion where nombre_rn='RN1')) then
       insert into rn_restriccion(nombre_rn,mensaje) values ('RN1','Un Aula no puede tener cantidad de sillas mayor que el doble de la cantidad de mesas ni menor que la cantidad de mesas');
       END IF;
END IF;
RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION frn1_1()
  OWNER TO postgres;
```

Anexo 2: Código generado por el LPT-SQL para la función frn2_1()

```
CREATE OR REPLACE FUNCTION frn2_1()
  RETURNS trigger AS
$BODY$
 BEGIN
 if(EXISTS(SELECT * FROM public."vrn2" a WHERE (a."id_laboratorio"=NEW."id_laboratorio") AND (a."nombre_laboratorio"=NEW."nombre_laboratorio")) ) then
        if not (exists(select * from rn_restriccion where nombre_rn='RN2')) then
        insert into rn_restriccion(nombre_rn,mensaje) values ('RN2','Un laboratorio no puede tener cantidad de mesas mayor que la cantidad de sillas');
        END IF;
END IF;
RETURN NEW;
END;
 $BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION frn2_1()
  OWNER TO postgres;
```

Anexo 3: Código generado por el LPT-SQL para la función frn3_1()

```
CREATE OR REPLACE FUNCTION frn3_1()
  RETURNS trigger AS
$BODY$ BEGIN if(EXISTS(SELECT * FROM public."vrn3" a WHERE (a."id_departamento"=NEW."id_departamento") AND (a."nombre_departamento"=NEW."nombre_departamento")) ) then if not (exists(select * from rn_restriccion where nombre_rn='RN3')) then insert into rn_restriccion(nombre_rn,mensaje) values ('RN3','Un departamento no puede tener cantidad de sillas menor que la cantidad de mesas');END IF;END IF;RETURN NEW; END;  $BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION frn3_1()
  OWNER TO postgres;
```

Anexo 4: Código generado por el LPT-SQL para la función frn4_1()

```
CREATE OR REPLACE FUNCTION frn4_1()
  RETURNS trigger AS
$BODY$
 BEGIN
 if(EXISTS(SELECT * FROM public."vrn4" a WHERE (a."id_departamento"=NEW."id_departamento") AND (a."nombre_departamento"=NEW."nombre_departamento") ) ) then
        if not (exists(select * from notificaciones where nombre_rn='rn4')) then
        INSERT INTO notificaciones (nombre_rn,mensaje) values ('rn4','Departamento duplicado');
        END IF;
 END IF;
RETURN NEW;
END;
 $BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION frn4_1()
  OWNER TO postgres;
```

Anexo 5: Código generado por el LPT-SQL para la vista vrn1

```
CREATE OR REPLACE VIEW vrn1 AS
 SELECT sujeto.id_aula, sujeto.nombre_aula, sujeto.cant_sillas,
    sujeto.cant_mesas, sujeto.cant_pupitres, sujeto.id_facultad,
    sujeto.nombre_facultad FROM "Aula" sujeto  WHERE (( SELECT
a.cant_sillas FROM "Aula" a WHERE sujeto.id_aula = a.id_aula AND
sujeto.nombre_aula = a.nombre_aula)) > ((( SELECT a.cant_mesas
FROM "Aula" a WHERE sujeto.id_aula = a.id_aula AND sujeto.nombre_aula
= a.nombre_aula)) * 2) OR (( SELECT a.cant_sillas FROM "Aula" a
        WHERE sujeto.id_aula = a.id_aula AND sujeto.nombre_aula =
a.nombre_aula)) < (( SELECT a.cant_mesas FROM "Aula" a
        WHERE sujeto.id_aula = a.id_aula AND sujeto.nombre_aula =
a.nombre_aula));
ALTER TABLE vrn1
  OWNER TO postgres;
```

Anexo 6: Código generado por el LPT-SQL para la vista vrn2

```
CREATE OR REPLACE VIEW vrn2 AS
 SELECT sujeto.id_laboratorio, sujeto.nombre_laboratorio, sujeto.cant_mesas, sujeto.cant_sillas, sujeto.nombre_jefe, sujeto.id_facultad, sujeto.nombre_facultad FROM "Laboratorio" sujeto
  WHERE (( SELECT a.cant_mesas FROM "Laboratorio" a
        WHERE sujeto.id_laboratorio = a.id_laboratorio AND sujeto.nombre_laboratorio = a.nombre_laboratorio)) > (( SELECT a.cant_sillas FROM "Laboratorio" a WHERE sujeto.id_laboratorio = a.id_laboratorio AND sujeto.nombre_laboratorio = a.nombre_laboratorio));
ALTER TABLE vrn2
  OWNER TO postgres;
```

Anexo 7: Código generado por el LPT-SQL para la vista vrn3

```
CREATE OR REPLACE VIEW vrn3 AS
 SELECT sujeto.id_departamento, sujeto.nombre_departamento, sujeto.cant_mesas, sujeto.cant_sillas, sujeto.nombre_jefe, sujeto.id_facultad, sujeto.nombre_facultad FROM "Departamento" sujeto
  WHERE (( SELECT a.cant_sillas FROM "Departamento" a
        WHERE sujeto.id_departamento = a.id_departamento AND sujeto.nombre_departamento = a.nombre_departamento)) < (( SELECT a.cant_mesas FROM "Departamento" a
        WHERE sujeto.id_departamento = a.id_departamento AND sujeto.nombre_departamento = a.nombre_departamento));
ALTER TABLE vrn3
  OWNER TO postgres;
```

Anexo 8: Código generado por el LPT-SQL para la vista vrn4

```
CREATE OR REPLACE VIEW vrn4 AS
 SELECT sujeto.id_departamento, sujeto.nombre_departamento,
sujeto.cant_mesas, sujeto.cant_sillas, sujeto.nombre_jefe,
sujeto.id_facultad, sujeto.nombre_facultad
   FROM "Departamento" sujeto
  WHERE (( SELECT count(a.nombre_departamento) AS count
           FROM "Departamento" a
          WHERE a.nombre_departamento = 'Matemática'::bpchar)) > 1;
ALTER TABLE vrn4
  OWNER TO postgres;
```