Revista Cubana de Ciencias Informáticas
Vol. 7, No. 4, octubre-diciembre, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-15
http://rcci.uci.cu

Tipo de artículo: Artículo de revisión
Temática: Tecnologías de Bases de datos
Recibido: 23/09/2013 | Aceptado: 31/10/2013 | Publicado: 10/12/2013

Análisis de la capacidad de OCL para generar restricciones de integridad de negocio

Analisys of OCL capability to generate business integrity constraints

José Enrique Saura Guerra[1*], Luisa González González[2]

[1] Departamento de Programación, Facultad 1. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½, Torrens, Boyeros, La Habana, Cuba. CP.: 19370
[2] Universidad Central Martha Abreu de las Villas, Santa Clara, Villa Clara, Cuba. Correo-e: luisagon@uclv.edu.cu

*Autor para la correspondencia: jsaura@uci.cu

Resumen

Toda implementación de un sistema de información debe asegurar no solamente que una operación se aplicó sino que al hacerlo no derive en una violación de alguna restricción de integridad del sistema de información a modelar, definidas en el modelo conceptual. La violación de alguna de estas implicaría obtener un modelo de datos impreciso o incompleto. Para garantizar una correcta definición de las restricciones que modela cada Universo del Discurso, es preciso conocer las principales caracteristicas y clasificaciones de las restricciones de integridad y las herramientas que faciliten la definición de estas en el modelo conceptual. En el presente trabajo se desglosan varias clasificaciones de restricciones de integridad, los tipos que existen y las causas que producen las violaciones. Se enuncia el concepto de lenguajes de especificación y se caracteriza a, *Object Constraint Language* al ser este un lenguaje para la definición de restricciones ampliamente utilizado. Obteniéndose al final herramientas que permiten identificar y definir adecuadamente las restricciones de integridad en dependencia del Universo del discurso a modelar.

Palabras clave: Modelo conceptual; OCL, restricciones de integridad, sistemas de información.

Revista Cubana de Ciencias Informáticas
Vol. 7, No. 4, octubre-diciembre, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-15
http://rcci.uci.cu

Abstract

Every system implementation most ensure that not only an operation has been applied but also that doing that does not break any information system's integrity constraints defined in the conceptual model. To guarantee a correct definition of the constraints that each information system carries out is necessary to know the main characteristics and classifications of the integrity constraints and the tools that ensures their definition in the conceptual model. This paper describes how restrictions are classified, different types of constraints and causes of constraints violations. Also, Object Constraint Language is characterized, as an language for constraints definition, integrated with UML and widely used to this end. The main objective is to obtain the tools that ensures the correct indentification of restrictions and it integration with a modeling language such as UML.

Keywords: *Conceptual model, integrity constraints, information systems, OCL.*

Introducción

En una sociedad moderna, la dinámica de la cotidianidad trae consigo, que las aplicaciones con las que se interactua a diario, deben encontrarse en constante cambio y actualización. Traduciéndose estas actualizaciones en modificación de los datos que conforman el universo del discurso (UD) del sistema de información (SI) al que sirven de interfaz. El enorme volumen de información que se maneja no está ajeno a que aparezcan errores u omisiones, dado que existen hechos o mensajes erróneos que tienden a introducirse en nuestros sistemas (Pérez, 2013). Según plantea Cabot (2008), la generalidad de estos problemas son producidos por factores humanos.

El prevenir la ocurrencia de estos problemas está acotado dentro del campo de Verificación de Integridad. El chequeo y validación de la integridad en un sistema de información, contribuye a contar con un universo del discurso semánticamente conciso ya que la integridad de un sistema es igual a la validez + completitud, por lo que restricción de integridad (RI) es una condición que no podría satisfacerse en algunos estados de la base de información o por algunos eventos y que el sistema (interfaz + BD) debe resolver (Cabot, 2008).

Las RI son reglas definidas para garantizar la exactitud y consistencia de los datos almacenados en una base de datos relacional. Codd (1990) define cuatro tipos de restricciones en el modelo relacional estas son: Integridad de Entidad, Integridad Referencial, Integridad de Dominio e Integridad definida por el usuario.

Las RI son parte incidente de modelos conceptuales y son utilizadas extensivamente durante el análisis y diseño de un SI. Estas son esenciales para asegurar la exactitud del modelo de información dado su capacidad para representar una semántica adecuada del dominio del problema. Según (Elita y Miliauskaite, 2005) pueden llevarse a cabo

restricciones de integridad en sistemas de almacenamiento de datos (la base de datos) o en el de código del software (aplicación que maneja los datos), para proteger contra cambios no admisibles en la información.

En gran parte de los sistemas, la integridad total puede ser lograda sólo por la intervención humana. Para asegurar la integridad, debemos verificar los hechos sistemáticamente en la base de información contra el dominio (Olivé, 2007). Sin embargo, es posible construir los mecanismos en un sistema de información que automáticamente garantice algún nivel de integridad. Se puede definir las condiciones en la base de información (BI) y los eventos la alteran tal que, si es satisfecho, se contará con algún nivel de confianza en la integridad de la BI. Se entiende que el sistema de información incluirá los mecanismos para garantizar la satisfacción de las RI en cualquier momento. (Morgan, 2002)
La gestión de las RI en la actualidad cae principalmente dentro de las responsabilidades del administrador de la Base de Datos (BBDD) en el propio gestor luego de pasar por todo el proceso de diseño o al desarrollador del software garantizarlas al implementar.

Según (Richters, 2000), debe quedar clara la idea de que el modelo y sus restricciones deben ser validadas antes de comenzar su implementación, porque de esa forma se pueden evitar muchos errores de diseño e implementación.
Como parte de los mecanismos para garantizar un nivel semántico adecuado al UD es aconsejable que una RI sea definida desde el mismo modelado conceptual, para ello el diseñador cuenta con diversas herramientas que lo asistan en ese proceso, por ejemplo los lenguajes de especificaciones de RI.

Un lenguaje de especificación o lenguaje de descripción es un lenguaje formal o semiformal cuya función es construir modelos de los sistemas que se desea elaborar (Elsmari y Navathe, 2012). A diferencia de los lenguajes de programación, que son lenguajes interpretables o traducibles por una computadora hacia una representación ejecutable, los lenguajes de especificación no son utilizados para implementar el sistema, sino para especificarlo, conceptualizarlo o incluso validarlo, aunque suelen ser legibles para un programa de computadora, que puede asistir en el proceso de validación (Cabot, 2008). Existen una gran variedad de lenguajes de especificación, algunos de los más representativos son:

- OCL el cual es un lenguaje para la descripción formal de expresiones en los modelos UML.
- Alloy, lenguaje de especificaciones que utiliza la lógica de primer orden y se basa en el uso de relaciones.
- Autómatas es un formalismo utilizado para modelar sistemas discretos en general.
- B, lenguaje de descripción formal basado en la lógica de predicados.

- Cálculo Pi, lenguaje de especificación para sistemas distribuidos y paralelos.
- CSP, lenguaje formal basado en el álgebra de procesos.
- Estelle, lenguaje formal basado en autómatas de estado finito para la especificación de sistemas distribuidos.

Todos estos lenguajes son aplicables a problemas especifícos dentro de distintas ramas de la informática, ingenieria eléctrica y ramas afines.

Partiendo de la importancia de asegurar la integridad de un sistema de información expresada anteriormente, en el presente trabajo se caracterizan las RI y posteriormente se analiza OCL (del inglés, *Object Constraints Language*), un lenguaje de especificación ampliamente utilizado para especificar restricciones de integridad con una fuerte integración con el lenguaje de modelado unificado (UML, por sus siglas en inglés). Enfocándose la segunda parte de esta investigación en este lenguaje para la especificación de RI y la descripción formal de expresiones. Además se definen en la investigación algunas de las características del lenguaje OCL que permiten la gestión de las restricciones y como quedan definidas en su sintaxis.

Desarrollo

La validez y totalidad de los datos son componentes primordiales en la integridad de un sistema de información. La integridad es garantizada cuando todos los hechos pertenecientes al sistema son válidos y este contiene todos los hechos relevantes (Olivé, 2007).

Validez: SI - > s implica Realidad ->s

Totalidad: Realidad ->s implica SI ->s

SI: Sistema de información s: sentencia

Según (Pérez, 2013) la falta de integridad trae múltiples problemas que pueden indistintamente presentarse a corto, mediano o largo plazo.

En el trabajo de Pakalnickiene (2007) se definen las ventajas de un modelo conceptual organizado. Al extenderse estos modelos con chequeo de restricciones de integridad, permite obtener modelos más precisos y capaces de asegurar que la semántica del dominio está expresada adecuadamente (Halping, 2003).

Clasificaciones de RI

Las restricciones de integridad nos permiten obtener un modelo, semánticamente preciso atendiendo al dominio del sistema de información a modelar, por ende optimizando el proceso de diseño. El chequeo de las RI está orientado a

determinar, de forma eficiente, cuándo el estado de la base de información es consistente después de un conjunto de eventos estructurales.

Existen diversas clasificaciones para las restricciones, atendiendo a ciertos puntos de vista. Estos grupos se solapan generalmente y son escasos los estudios que estructuran las RI a partir de una misma raíz, uno de ellos lo podemos encontrar en (Elita y Miliauskaite, 2005). Otras clasificaciones de RI son presentadas en (Olivé, 2007; Chomicki, 2005; Halping, 2006).

Las RI se pueden clasificar según:

- La razón que la restricción debe sostener (la fuente);
- Los hechos involucrados por la restricción (el alcance);
- La causa de la violación de la restricción.

Fuente

Las restricciones de integridad pueden ser clasificadas según su fuente en analítico, deóntico o empírico (Olivé, 2007).

- Las violaciones de *restricciones analíticas* son debidas a los errores en la representación de hechos. Por ejemplo: "una puerta no puede ser abierta y cerrada al mismo tiempo" es analítico.
- Las violaciones de *restricciones deónticas* pueden ser causadas por los errores en la representación de hechos o porque la conducta del dominio se desvía de la condición declarada. Por ejemplo: "el salario de un empleado no puede disminuir" es deóntico.
- Las violaciones de *restricciones empíricas* pueden ser causadas por los errores en la representación de hechos o porque alguna excepción se ha levantado en el dominio. Por ejemplo: "Un cliente no compra más de 999 unidades de cualquier artículo" sería empírico.

Alcance

Las restricciones son condiciones que deben satisfacerse por la base de información y los eventos. Normalmente, una restricción involucra sólo un juego limitado de hechos en la base de información y/o un juego limitado de eventos, y esto permite clasificarlo según los hechos que involucra o su alcance (Alonso, 2010).

- Una *restricción estática* abarca los hechos de un solo estado de la base de información, y debe satisfacerse en cada estado. Todos los lenguajes de modelado conceptual permiten definir las restricciones estáticas.
- Una *restricción de transición* comprende los hechos de dos o más estados de la base de información. Normalmente, una restricción incluye hechos de sólo dos estados consecutivos, reprimiendo la transición entre ellos, pero en general la restricción puede referirse a cualquier número de estados.
- Una *restricción de evento* implica sólo un evento.
- Una *restricción de historia* de evento define dos o más eventos que ocurren en los mismos o diferentes momentos. Las restricciones de este tipo son usadas a menudo para determinar las clasificaciones temporales permitidas, de ocurrencias de evento.
- Una *restricción global* involucra los hechos de uno o más estados de la base de información y uno o más eventos.
- Una *restricción de condición previa de evento* (es un tipo particular de restricción global), atañe sólo a un evento y el estado de la base de información cuando este ocurre. Los lenguajes de modelado conceptuales permiten su definición.

Los siguientes son ejemplos de restricciones, con su clasificación según su alcance:

- Todos los empleados siempre se asignan a algún proyecto (estática).
- El sueldo de un empleado no puede disminuir (transición).
- El depósito inicial de una nueva cuenta bancaria debe ser por lo menos un euro (evento).
- Un cliente no puede abrir dos cuentas en el mismo día (historia de evento).
- Un cliente no puede abrir una nueva cuenta si él es un poseedor de alguna cuenta que se ha sobregirado para más de 30 días durante el último año (global).
- Un cliente no puede abrir una nueva cuenta si el equilibrio total de las cuentas que él ya tiene es negativo (condición previa de evento).

Causa de violación

Se ha mencionado que una restricción puede violarse por la llegada de un mensaje en la entrada o por la ausencia de uno o más mensajes durante un intervalo de tiempo. En el primer caso, se dice que la causa de la violación es el evento informado por el mensaje, y que la restricción es evento-violable. En el segundo caso, que la causa de la violación es el paso de tiempo, y entonces la restricción es tiempo-violable (Olivé, 2007).

Unos ejemplos de cada clase son:

- El sueldo de un empleado no puede disminuir (evento – violable).
- El depósito inicial en una cuenta debe ser por lo menos 50 pesos (evento – violable).
- Una cuenta no puede sobregirarse para más de 30 días (tiempo – violable).
- Todos los empleados deben reportar sus actividades por lo menos una vez al mes (tiempo – violable).

De acuerdo al modelo

Todos los modelos de datos conceptuales permiten definir una serie de Restricciones de Integridad en sus diagramas. El soporte de estas restricciones varía según el modelo utilizado (Elita y Miliauskaite, 2005; Olivé, 2003). Vale destacar que un esquema conceptual = diagrama + Restricciones de Integridad (Elsmari y Navathe, 2012). Por lo cual se puede clasificar una RI en explícita y en implícita o estructural, para un modelo determinado.

En un ejemplo asociado al Modelo Entidad Relación, la siguiente restricción es estructural: Un cliente se identifica unívocamente por su número. El ME/R(Modelo Entidad Relación) permite en sus diagramas especificar el o los atributos identificadores (Date, 2006), por lo que esta restricción se representa estructuralmente. Sin embargo para el diagrama de clases de UML esta misma restricción se clasifica en explícita, pues este modelo no soporta los atributos identificadores (Gomaa, 2011). Esta restricción debe ser definida en otro lenguaje de manera explícita.

RI en el diseño de una Base de Datos

Lo más sensato es pensar que el lugar ideal para implementar Restricciones de Integridad, sea la base de datos del negocio. Esto es debido a un contacto más directo con los datos a través de gestores especializados para su manejo (Morgan, 2002).

En el diseño de bases de datos relacionales (Elsmari y Navathe, 2012) se observan varios diseños asociados a los datos: el diseño conceptual, lógico y físico. En los dos primeros se ubican los tres niveles de expresión de las RI. Conceptualmente se definen RI naturales o técnicas y lógicamente restricciones en un nivel formal, tal como muestra el trabajo (Pérez, 2013) presente en la Figura 1.

Revista Cubana de Ciencias Informáticas
Vol. 7, No. 4, octubre-diciembre, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-15
http://rcci.uci.cu

Figura 1. Sección del diseño de BD Relacionales.

Resultados y discusión

Partiendo de la premisa de captar Restricciones de Integridad al modelar para obtener un sistema conciso y preciso atendiendo al negocio, es importante contar con herramientas que agilicen y conduzcan el trabajo del diseñador. Existen diferentes lenguajes de modelación que simplifican el trabajo. Los más extendidos para diseñar conceptualmente Bases de Datos son UML y el ER. Pero estos lenguajes por si solos carecen de la capacidad para modelar las restricciones y necesitan otros que complementen sus funcionalidades. Para suplir estas deficiencias surgen los llamados lenguajes de especificación.

Lenguajes de especificación

En el contexto de la computación y ciencias afines, se puede definir un lenguaje de especificación o descripción, como el lenguaje formal o semi formal cuya función es construir modelos de los sistemas que se desean obtener.

A diferencia de los lenguajes de programación que son interpretables o traducibles por un dispositivo hacia una representación ejecutable, los lenguajes de especificación según su definición en (OMG, 2006), no son utilizados para implementar el sistema sino para especificarlo, conceptualizarlo o validarlo. Dentro de esos lenguajes de especificación tenemos a OCL como uno de los más extendidos. El lenguaje de restricción OCL se encuentra definido según los conceptos de UML al ser un sub-lenguaje de este.

Lenguaje de especificación OCL

OCL es un lenguaje notacional, subconjunto del UML estándar, industrial, que permite a los desarrolladores de software escribir restricciones sobre modelos de objetos (pre y pos condiciones, invariantes, valores derivados y restricciones sobre operaciones). Estas restricciones son particularmente útiles, en la medida en que permiten a los desarrolladores crear un amplio conjunto de reglas que rigen el aspecto de un objeto individual (Cabot y Gogolla, 2012).

Es un lenguaje formal para expresar restricciones libres de efectos colaterales. Los usuarios del Lenguaje Unificado para Modelado (UML) y de otros lenguajes, pueden usar el OCL para especificar restricciones y otras expresiones incluidas en sus modelos. Según Pandey (2011) el OCL tiene características de un lenguaje de expresión, de un lenguaje de modelado y de un lenguaje formal.

En el modelado orientado a objetos, un modelo gráfico como el de clases no es suficiente para lograr una especificación precisa y no ambigua. Existe la necesidad de describir características adicionales sobre los objetos del modelo. Muchas veces estas características se describen en lenguaje natural. La práctica ha revelado que muy frecuentemente esto produce ambigüedades. Para escribir características no ambiguas se han desarrollado los lenguajes formales (Romero, 2010).

OCL tiene las características de un lenguaje de expresiones, un lenguaje de modelos y un lenguaje formal:

- OCL es un lenguaje de expresiones puro. Esto significa que un estado del sistema nunca cambiará debido a una expresión OCL, incluso una expresión OCL podría usarse para describir tal cambio de estado (p.e. en una postcondición). Los valores de todos los objetos, incluidos los enlaces, no cambiarán. En cualquier momento en que se evalúa una expresión OCL, simplemente devuelve un valor (Olivé y Cabot, 2007).
- OCL es un lenguaje de modelos y no un lenguaje de programación. No se puede escribir un programa lógico o un flujo de control en OCL. Especialmente, no se puede invocar procesos o activar operaciones de consulta en OCL. Debido a que OCL es en primer lugar un lenguaje de modelos, no se puede asegurar que todo sea directamente ejecutable. Como lenguaje de modelos, todos los aspectos de implementación están fuera de alcance y no pueden expresarse en OCL. Cada expresión OCL es conceptualmente atómica. El estado de los objetos en el sistema no puede cambiar durante la evaluación.
- OCL es un lenguaje formal donde todos los constructores tienen un significado formal definido. La especificación de OCL es parte de la especificación de UML. OCL no tiene la intención de reemplazar los lenguajes formales existentes. Los lenguajes formales tradicionales se usaban por personas con conocimientos matemáticos, pero dificulta su uso para la mitad de empresas y modeladores de sistemas. OCL ha sido desarrollado para llenar este hueco (Casas y Arenas, 2010).

Puesto que en un proyecto hay muchas personas involucradas (usuario, expertos, encargados del mantenimiento y demás.) los modelos deben ser entendidos por una amplia y variada audiencia. OCL es fácil de aprender y usar por los desarrolladores sin amplios conocimientos matemáticos. OCL tiene ciertas características que permiten a los

desarrolladores adoptarlo a su ritmo y sólo donde lo necesiten. Hace accesibles las especificaciones formales con un trasfondo matemático limitado (Cabot y Gogolla, 2012).

Según OMG (del inglés, *Object Managment Group*) (OMG, 2006), OCL puede ser usado con distintos propósitos:

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

Dentro de la semántica del UML, el OCL es usado en la sección reglas, como constantes estáticas sobre las meta-clases en la sintaxis abstracta. Varios diseñadores también lo utilizan para definir operaciones 'adicionales', que son tomadas en cuenta en la formación de reglas.

Definición de restricciones con OCL

Usualmente se debe definir otras restricciones a instancias, independientemente de las ya existentes en UML y otros lenguajes de modelado conceptual. Para ello OMG propone el uso de OCL (Cabot y Teniente, 2005).

En la Figura 2 se define el diagrama de clases para un caso de estudio de una familia y posteriormente se define cómo quedaría una restricción de dominio descrita en OCL (Ebert y Walter, 2011).

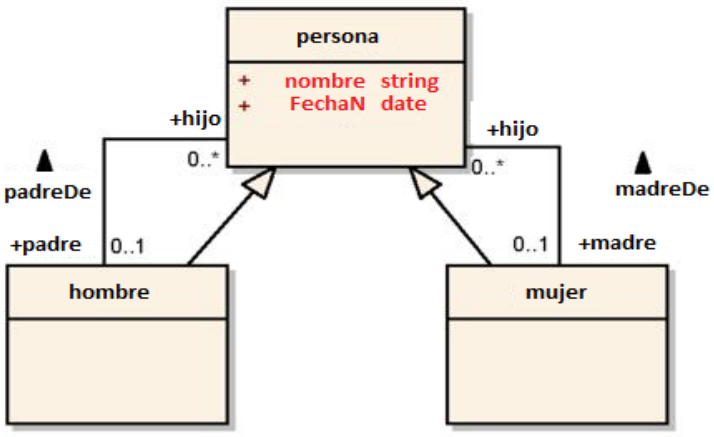

Figura 2. Diagrama de clases UML.

Revista Cubana de Ciencias Informáticas
Vol. 7, No. 4, octubre-diciembre, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-15
http://rcci.uci.cu

Un ejemplo sería que: se requiere describir en la restricción el hecho de que los padres nacen antes que sus hijos.

En OCL quedaría de la siguiente manera:

context Persona

inv: padre.FechaN < self.FechaN

and mother.FechaN < self.FechaN

Según las definiciones de RI vistas anteriormente esta se podría clasificar como; de fuente empírica, alcance estático y la causa de la violación es evento-violable.

Esta restricción posteriormente debería ser portada a SQL estándar al pasar al esquema físico, mediante el uso de la cláusula CHECK o TRIGGERS, quedando esto en manos del administrador de la BD.

En la figura 3 se presenta otro modelo de clases en este caso el correspondiente con un caso de uso de gestión de una flota de vehículos, el cual debe restringir; la cantidad de dueños por vehículos, la edad requerida para los dueños de auto y que toda persona debe tener al menos un auto negro.

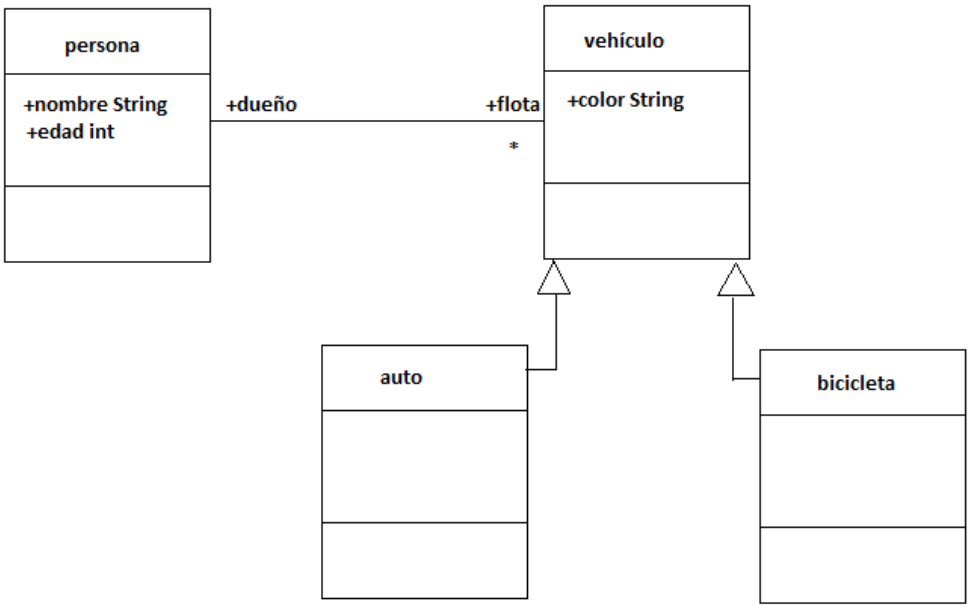

Figura 3. Diagrama de Clases Flota de Vehículos.

Las restricciones de dominio definidas anteriormente quedarían de la siguiente forma:

Context Auto

Inv: self.dueño.edad >= 18 (el dueño de un auto debe ser mayor de 18)

Context Persona

Inv: self.flota->size() <= 3 (nadie tiene más de tres vehículos)

Context Persona

Inv: self.flota->forall(v| v.color = 'negro') (todos los vehículos de una persona son negros)

Nótese su lenguaje natural más afín a diseñadores sin amplios conocimientos matemáticos. OCL usa operaciones para describir las restricciones. Se define en una operación primeramente un contexto en el cual se especifica la RI, el atributo a restringir y por último el valor que acota la expresión.

Dichas operaciones presentan algunos problemas en OCL. Por ejemplo, una operación puede entrar en un ciclo infinito o puede ser indefinida y esto hace a OCL menos preciso. Otra deficiencia del lenguaje radica en las operaciones aplicadas a varias clases que tienen la relación de herencia, en donde las operaciones pueden ser redefinidas por los objetos de esas clases. Quedando ambiguo para decidir, el significado de la expresión que contiene las operaciones (He, 2011).

Otra de las deficiencias presentes en OCL es que este verifica todas las instancias de un tipo específico, sobre el que se definió la RI, no siendo esto óptimo pues las instancias no deberían verificarse teniéndose en cuenta, por ejemplo que si asumimos que usualmente la base de información es consistente antes de la actualización, solamente han sido modificados por un conjunto de eventos estructurales aplicados a la base de información estos pueden violar la RI asociada (Cabot y Teniente, 2005).

Partiendo de las deficiencias encontradas en OCL durante su revisión y su integración total con UML, sería interesante buscar alternativas a este lenguaje de especificación. Logrando que estas se integren con otros lenguajes o métodos de modelados más cercanos a diseñadores con perfil matemáticos o sin experiencia en programación como es el ER. Este modelo es muy utilizado a escala académica por su sencillez y expresividad y se encuentra en constante evolución existiendo múltiples extensiones al modelo original planteado por Chen (1976) como por ejemplo las planteadas por (Elsmari y Navathe, 2012).

Conclusiones

La definición correcta de las RI pasa por conocer las diversas clasificaciones de estas atendiendo fundamentalmente a la fuente, alcance y la causa de la violación.

El chequeo de RI desde el modelo conceptual optimiza el diseño, al ser este semánticamente más preciso que dejar la responsabilidad de chequear la integridad a la etapa del modelado físico, al introducir el modelo en el gestor de base de datos escogido, cambiando por tanto lo modelado en los pasos anteriores; conceptual y lógico.

La mayoría de los lenguajes de modelación no traen implícitos un chequeo de las RI sino que agregan otros lenguajes, extenciones o herramientas para lograr este propósito.

El lenguaje UML integra el sub lenguaje OCL para describir las restricciones. Es un lenguaje formal, fácil de leer y de escribir.

OCL no carece de deficiencias a la hora de su implementación y definición de RI y para solucionar estos problemas existen otras alternativas como las propuestas en (Miliauskaite, 2005) y otros trabajos que plantean el chequeo de RI en el modelo entidad relación (Saura, 2013), aun así OCL es una sólida opción, especialmente si se modela utilizando UML como lenguaje para el modelado conceptual.

Referencias

- ALONSO, A. P. “Reglas de Negocio En Bases de Datos Relacionales”. 2010.
- CABOT, J. “Verification of UML/OCL Class Diagrams Using Constraint Programming.” In IEEE International Conference. 2008.
- CABOT, J. and GOGOLLA, M. “Object Constraint Language (OCL): a Definitive Guide.” In *Proceedings of the 12th International Conference on Formal Methods for the Design of Computer, Communication, and Software Systems: Formal Methods for Model-driven Engineering*, 58–90. SFM’12. Berlin, Heidelberg: Springer-Verlag. doi: 10.1007/978-3-642-30982-3_3. 2012. Disponible en: [http://dx.doi.org/10.1007/978-3-642-30982-3_3].
- CABOT, J. and TENIENTE, E. *Computing Relevant Instances That May Violate an OCL Constraint*. Springer-Verlang. 2005.
- CASAS CUEVAS, J. and Arenas Fernández, M. “OCL (Object Constraint Language).” 2010.
- CHEN, PETER PIN-SHAN. “The Entity-relationship Model—toward a Unified View of Data.” *ACM Trans. Database Syst.* 1 (1) (March): 9–36. doi:10.1145/320434.320440. 1976.

- Chomicki, J. *Minimal Change Integrity Maintenance Using Tuple Deletions*. Vol. 197. Information and Computation. 2005.
- CODD, E. F. *The Relational Model for Database Management: Version 2*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. 1990.
- *Conceptual Modelling of Information Systems*. Berlín: Springer-Verlang. 2007.
- DATE, C J. "An Introduction to Database Systems". 2006.
- EBERT, J. and TOBIAS W. "Interoperability Services for Models and Ontologies" 2011.
- ELSMARI, R, and S NAVATHE. "Database Systems: Models, Languages, Design, and Application Programming". 2012.
- GOMAA, H. *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. 1st ed. New York, NY, USA: Cambridge University Press. 2011.
- HALPING, T. "UML Data Models from an ORM Perspective: Part 1 − 10." *Journal of Conceptual Modeling, 2003*. Disponible en: [http://www.orm.net/uml_orm.html].
- HALPING, TERRY. "Business Rule Modality". Neumot University. 2006.
- HE, YUJING. "Comparison of the Modeling Languages Alloy and UML." *Department of Computer Science. Portland State University*. 2011.
- MILIAUSKAITE, E. "Representation of Integrity Constraints in Conceptual Models." *Information Technology and Control, Kauno Technologijos Universitetas* 34-4: 335–365. 2005.
- MILIAUSKAITE, E. "Taxonomy of Integrity Constraints in Conceptual Models." In. 2005.
- MORGAN, T. *Business Rules and Information Systems: Aligning It with Business Goals*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. 2002.
- OLIVÉ, A. "Integrity Constraints Definition in Object-Oriented Conceptual Modeling Languages." In *Conceptual Modeling - ER 2003*, edited by Il-Yeol Song, StephenW. Liddle, Tok-Wang Ling, and Peter Scheuermann, 2813:349–362. Lecture Notes in Computer Science. Springer Berlin Heidelberg. 2003 Disponible en: [http://dx.doi.org/10.1007/978-3-540-39648-2_28].
- OLIVÉ, A. and CABOT, J. "A Research Agenda for Conceptual Schema-Centric Development." In *Conceptual Modelling in Information Systems Engineering*, edited by John Krogstie, AndreasLothe Opdahl, and Sjaak Brinkkemper, 319–334. Springer Berlin Heidelberg. Disponible en: [http://dx.doi.org/10.1007/978-3-540-72677-7_20. 2007].
- OMG. "Object Constraint Language V2.0. Technical Report." 2006.

- PAKALNICKIENE, E. "The Orderliness and Precision in Conceptual Modelling." In, 341–350. 2007.
- PANDEY, R. K. "Object Constraint Language (OCL): Past, Present and Future." *SIGSOFT Softw. Eng. Notes* 36 (1) (January): 1–4. doi:10.1145/1921532.1921543. 2011.
- PÉREZ ALONSO, A.; PEREIRA TOLEDO, A. and GONZÁLEZ GONZÁLEZ, L. "Generación de Restricciones de Integridad en Bases de Datos Relacionales." In. 2013.
- RICHTERS, M. "Validating UML Models and OCL Constraints." In. 2000.
- ROMERO GUILLÉN, P. "Análisis y Diseño Orientado a Objetos." 2010.
- SAURA GUERRA, J. E. "Captación de Restricciones de Integridad En El Modelo Entidad Relación." In. 2013.