**Escuela Politécnica Superior**

Noticias

# Lectura de la tesis de Xabier de Carlos Garcia

16/06/2016

El 14 de junio, a las 12:00 h, el Doctorando Xabier de Carlos Garcia de la Escuela Politécnica Superior de Mondragon Unibertsitatea presentó su tesis doctoral en el Aula Magna del Colegio Mayor de Mondragon Unibertsitatea. El título de la tesis: *Model Query Tranformation Framework – MQT: from EMF-Based Model Query Languages to Persistence-Specific Query Languages*, y sus directores: Goiuria Sagardui y Salvador Trujillo. Además, obtuvo la calificación de Sobresaliente mención Cum Laude y la mención de Doctor Internacional.

**En el tribunal de la tesis participaron:**

- **Presidente**: Dr.D. Oscar Díaz García (EHU-UPV)
- **Vocal**: Dr. D. Daniel Varró (Budapest University of Technology and Economics)
- **Vocal**: Dr. D. Oeystein Haugen (SINTEF)
- **Vocal**: Dr. D. Juan De Lara Jaramillo (Universidad Autónoma de Madrid)
- **Secretario**: Dra. Dña. Leire Etxeberria Elorza (Mondragon Unibertsitatea)

**Resumen de tesis:**

Los problemas de memoria de XMI (mecanismo de persistencia por defecto en Eclipse Modelling Framework) al trabajar sobre modelos grandes, han motivado la aparición de mecanismos de persistencia alternativos para los modelos EMF. Los enfoques más recientes proponen el uso de bases de datos, y soportan la ejecución de operaciones usando lenguajes de consulta de modelos basados en EMF (EMF API, OCL, EMF Query, EOL, etc.). Sin embargo, este tipo de lenguajes necesitan almacenar en memoria todos los elementos implicados en la consulta. Todos los elementos del modelo en el caso de las consultas que recorren completamente los modelos. Esta estrategia de carga provoca problemas de memoria cuando los modelos son de gran tamaño.
La mayoría de las bases de datos tienen lenguajes específicos que aprovechan las capacidades del motor de la base de datos (mayor rapidez) y sin la necesidad de cargar en memoria los modelos (menor uso de memoria). Por ejemplo, SQL es el lenguaje específico para las bases de datos relacionales y Cypher para las bases de datos basadas en Neo4J.

Este trabajo propone MQT-Engine, un framework que permite ejecutar lenguajes de consulta para modelos con tiempos de ejecución y uso de memoria similares al de un lenguaje especifico de base de datos. MQT-Engine realiza una transformación a dos pasos de las consultas: primero transforma las consultas que han sido escritas con un lenguaje de consulta para modelos en un modelo que es independiente del lenguaje (QLI Model); después, el modelo generado se transforma en una consulta equivalente, pero escrita con un lenguaje especifico de base de datos. La transformación a dos pasos proporciona extensibilidad y reusabilidad ya que facilita la inclusión de nuevos lenguajes.

Se ha implementado un prototipo de MQT-Engine que transforma consultas EOL en SQL y las ejecuta directamente sobre un repositorio CDO. El prototipo se ha evaluado con dos casos de uso. El primero está basado en el dominio de la ingeniería inversa. Se han comparado los tiempos de ejecución y el uso de memoria que necesitan MQT-Engine y otros lenguajes de consulta (EMF API, OCL y SQL) para ejecutar una serie de consultas sobre modelos persistidos en CDO. El segundo caso de uso está basado en el dominio de los ferrocarriles y

compara los tiempos de ejecución que necesitan MQT-Engine y otros lenguajes (EMF API, OCL, IncQuery, etc.) para ejecutar varias consultas.

Los resultados obtenidos muestran que MQT-Engine es capaz de ejecutar correctamente todos los experimentos y además es una de las soluciones con mejores tiempos para la primera ejecución de las consultas de modelos. MQT-Engine es la opción más rápida y que necesita menos memoria entre los lenguajes ejecutados sobre repositorios CDO. Por ejemplo, en el caso del modelo más grande de ingeniería inversa, MQT-Engine es 162 veces más rápido y necesita 23 veces menos memoria que los lenguajes de consulta de modelos ejecutados al lado del cliente. Además, la sobrecarga de la transformación es pequeña y constante (menos de 2 segundos).

Estos resultados prueban el objetivo principal de esta tesis: proporcionar un framework que permite a los ingenieros de modelos definir las consultas con un lenguaje de consulta de modelos y además ejecutarlas con una con tiempos de ejecución y uso de memoria similares a los de un lenguaje especifico de bases de datos.

Sin embargo, la solución tiene una serie de limitaciones: solo soporta consultas que recorren el modelo completamente y sin modificarlo; el prototipo es específico para consultas en EOL y sobre repositorios CDO (relacionales); y habría que optimizar la ejecución de las consultas cuando estas se ejecutan más de una vez. Se ha planeado resolver estas limitaciones en versiones futuras del trabajo.