AUTOMATIZACIÓN DEL DESPLIEGUE DE RECURSOS EN BASE DE DATOS

Francisco Javier Blanco, Rubén Jiménez, Carlos A. Iglesias
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
28040
fcojavibr@dit.upm.es, jimmy399@hotmail.com, cif@gsi.dit.upm.es

***Resumen*—En una arquitectura SOA, los servicios presentan dependencias con otros servicios y recursos ya desplegados, necesitando comunicarse con la base de datos para que les provea de información y donde almacenen los datos obtenidos. Una arquitectura telemática que gestione la configuración y el despliegue de tales servicios se hace esencial y dentro de esta, un sistema que gestione los recursos de bases de datos. Las tareas relacionadas con base de datos suponen una actividad ardua y compleja que es realizada de forma manual por administradores especializados. Rompiendo con esta idea y siguiendo la línea de investigación actual, el sistema desarrollado pretende gestionar automáticamente los recursos en el proceso de despliegue dando un soporte integral al DBA, facilitando considerablemente su trabajo y colocándole a un nivel superior de validación y optimización. El sistema se puede separar en dos: una herramienta de generación de unidades de despliegue y otra para realizar el despliegue en base de datos.**

I. INTRODUCCIÓN

Una de las actividades a llevar a cabo durante el desarrollo de aplicaciones empresariales es el despliegue de unidades software funcionales denominados comúnmente servicios. Este despliegue consiste en realizar todas las acciones necesarias para poder poner dichos servicios en funcionamiento. Es habitual que los servicios cooperen entre sí y necesiten de una serie de recursos previamente instalados y disponibles, existiendo dependencias entre estos servicios y otras unidades software en función de los recursos ofertados y los requisitos demandados. Siguiendo esta línea, unos de los recursos más demandados son los relacionados con base de datos ya que pocos servicios ejecutan su función de forma ajena a la información almacenada en el sistema o sin necesitar hacer persistentes los datos computados u obtenidos.

Los recursos de base de datos son unidades software que no realizan ninguna función ni operativa visible al usuario pero son los encargados de manipular la base de datos en dos vertientes: aquellos que tocan la estructura de la base de datos y aquellos que contienen la información y los datos a ser almacenados. Más técnicamente, los primeros se implementan siguiendo el lenguaje DDL, del inglés *Data Definition Language*, permitiendo llevar a cabo tareas de definición de las estructuras que almacenarán los datos. Por lo tanto, estos recursos, llamados a partir de ahora recursos DDL, son desplegados para construir la estructura y el esquema de la base de datos. Los segundos son definidos usando al lenguaje DML, *Data Manipulation Language*, llevando a cabo tareas de manipulación de los datos, organizados por la estructura correspondiente. Estos recursos serán llamados recursos DML a partir de ahora. En este artículo, se va a tratar mayormente el manejo de los recursos DDL pues estos son los recursos más complejos y los más propensos a errores y conflictos sobre la base de datos. El despliegue de los recursos de base de datos sería el proceso íntegro para poner dichos recursos en funcionamiento, es decir, que estén disponibles para los servicios y el resto de unidades software en general. El despliegue supone un procedimiento para situar los recursos en las base de datos correspondientes.

Los recursos de base de datos y más concretamente su despliegue son piezas clave para el funcionamiento de los servicios ya que definen la estructura requerida para el modelado de datos y manipulan los datos que son requerridos por estos. El manejo y gestión de bases de datos, incluyendo el despliegue de recursos, en sistemas de cierto volumen es una tarea ardua y complicada, que requiere que en los proyectos existan personas especializadas únicamente en esta tarea conocidos como administradores de base de datos (DBA). Durante el despliegue, un mínimo fallo, una inserción de datos incoherentes, una ejecución inadecuada puede dejar parte de la base de datos, sino toda, en un estado de error haciendo que servicios dejen de funcionar e incluso provocando que el sistema entero falle. Debido a la importancia de las bases de datos, la cantidad de factores y riesgos a tener en cuenta y la cantidad de actividades primarias y secundarias involucradas hace que el despliegue de los recursos de bases de datos sea una actividad compleja, costosa y realizada desde siempre de forma manual por los DBAs presentando en total un alto coste asociado.

En los últimos años se está investigando en propuestas para que la gestión de la configuración y el despliegue de recursos en base de datos esté más automatizada [1]. El desarrollo de herramientas que faciliten la labor de gestión de base de datos y el manejo de modelos y datos, incluido migración, control, abstracción, transformación, extracción y carga que representan las actividades existentes en los procesos de despliegue, han crecido de gran forma en los últimos años, tal y como se expone en la siguiente sección. El objetivo principal del sistema desarrollado es seguir este nuevo punto de vista de automatización y dar soporte automático al administrador de una forma integral, implementando el proceso completo de despliegue de recursos en bases de datos facilitando el trabajo al DBA y colocándole en un nivel superior de validación y optimización donde sólo ejerza su participación en tareas muy específicas.

ITECBAN (Infraestructura Tecnológica y Metodológica de Soporte para un Core Bancario) es un proyecto de inves-

tigación del programa nacional I+D CENIT cuyo principal objetivo es el desarrollo de una plataforma que sirva como base para la creación de sistemas de gestión destinados al sector bancario, eliminándose las actuales limitaciones de los sistemas de información empleados en entornos financieros. Las aplicaciones empresariales empleadas en este área se caracterizan por presentar en la mayor parte de los casos una arquitectura orientada a servicios [2], lo que las permite adaptarse de manera continua y flexible a los cambios que se producen en su alrededor, en respuesta a la demanda del mercado. Esta arquitectura facilita la interoperabilidad entre sistemas, al definir una manera estándar de anunciar e invocar servicios que pueden actuar de manera independiente. La heterogeneidad y dinamismo de estos servicios hace esencial la creación de un sistema telemático que los gestione incluido su adecuación y estado. Así, surge la necesidad de una solución que facilite el despliegue de los servicios, adecuándolos a las necesidades y características de cada entorno en concreto, lo que ha llevado al desarrollo dentro de un sistema de gestión al objetivo de dar soporte a las operaciones de despliegue y configuración de unidades software sobre los distintos entornos gestionados. Es, dentro de este sistema, donde se incluye el gestor de despliegue específico de los recursos de base de datos.

El resto del artículo se estructura de la siguiente manera. En el capítulo II se realiza un breve estado del arte sobre la gestión automática de los recursos de base de datos incluyendo un análisis de las herramientas existentes de despliegue y migración sobre base de datos exponiendo antes los requisitos del sistema implementado. A continuación, en III se describe en detalle el sistema implementado, centrándose en las dos herramientas que lo componen. En el capítulo IV se explica brevemente implementación final de las dos herramientas en la arquitectura global con sus pruebas finales. Por último, en V se muestran las conclusiones.

II. Gestión automática del despliegue en base de datos

En los últimos años han surgido muchas propuestas y un número considerable de herramientas que conllevan una automatización, o al menos la inclusión de tareas bastante más automatizadas, de gestión y configuración de las bases de datos y del despliegue de los recursos en estas. Según Mateen [3] la escasez de DBA cualificados ha motivado que la industria de las bases de datos desarrolle sistemas de gestión de base de datos automáticos, SGBDA o en inglés ADBMS.

Ahora que la complejidad de los sistemas está alcanzando un nivel que supera las capacidades humanas, con el desarrollo de las tecnologías las personas pueden gestionar tales sistemas complejos de una manera más fiable y eficiente. Con la inclusión de capacidades automáticas, incluso con ideas que parten de la *autonomic computing*, los sistemas crecen en rapidez, eficacia, fiabilidad y precisión con menos o sin interacción humana. Tales capacidades automáticas se pueden implementar en los sistemas de gestión, configuración y despliegue sobre base de datos [4]. La importancia de tales sistemas ha ido creciendo sustancialmente y actualmente existen bastantes herramientas que contemplan actividades sobre base de datos de forma automática o con un nivel de autonomía elevado.

Nuevas capacidades automáticas se están investigando en la gestión de base de datos que la dotarían, en el mayor de los casos, con suficiente inteligencia para ser autosuficientes [5] [6]. Tales características se pueden agrupar en seis áreas: auto-optimización (mejoras del rendimiento, manejo eficiente de recursos, configuración y carga de trabajo), auto-configuración (reconocer los cambios del entorno y reconfigurarse dinámicamente ante ellos), auto-reparación (mantenerse en un estado consistente todo el tiempo), auto-protección (seguridad, mecanismo de auditoría, capacidades de encriptamiento), auto-inspección (realizar decisiones inteligentes basadas en la consciencia que tiene la base de datos de ella misma: recursos, limitaciones, estado, entorno...) y auto-organización (reorganizar y reestructurar dinámicamente el modelo de datos e incluso los índices). Sin embargo, estás capacidades todavia distan de llevarse a cabo en plenitud y se encuentran pocas herramientas que integren de forma completa alguna de estas tareas.

Desde otro punto de vista más real, desde los últimos años existe un amplio abanico de herramientas, que aún siguiendo parte de esas capacidades automáticas, plantean actividades más claras y objetivas. La gran mayoría de las herramientas basan sus tareas automáticas en la gestión, el tratamiento y manejo de los recursos de bases de datos más que en la propia base de datos, refiriéndose a procesos de extracción, transformación y carga (ETL) como herramientas de ayuda a la migración [7] [8] [9]. Las herramientas abarcan áreas como el despliegue [10], la configuración [11], la recuperación [12], la monitorización [13] o el diagnosis de problemas de rendimiento [14]. Estas herramientas desarrollan la perspectiva de las capacidades automáticas pero sobre tareas concretas realizando actividades específicas de valor, dando soporte y facilitando de gran forma el trabajo de los DBA. Es bajo este punto de vista menos teórico donde la industria de las bases de datos ha decidido revolucionar el mercado, sacando una variedad de herramientas que llevan la gestión automática a muchas de las tareas de base de datos.

A. Marco impuesto para el sistema

La arquitectura de configuración y despliegue de ITECBAN debe gestionar correctamente los servicios, en el sentido en el que éstos necesitan recursos de base de datos, estos recursos deben ser gestionados también. Si la arquitectura pretende dar gestión de la forma más automática posible, el despliegue de los recursos debe realizarse siguiendo esta línea y, por lo tanto, siguiendo la línea actual de investigación. Esta actividad de despliegue contiene características automáticas de auto-configuración y de auto-organización, el resto de capacidades se alejan más del sistema implementado.

Según el modelo de configuración y depliegue [15], los recursos de base de datos se tratan como unidades software. Cada unidad software representa una entidad que exporta por sí sola uno o varios recursos, esta unidad es posible que necesite de los recursos exportados por otras unidades como ya se ha dicho. Además, una unidad software es desplegada, o replegada, de una única vez y de forma independiente a las otras, aunque se sigue para salvaguardar las posibles dependencias. Para que las unidades software sean manejadas por la arquitectura global y puedan desplegarse según un mismo patrón, estas deben estar contenidas en una estructura

estandarizada. Esta estructura estandarizada, definida en el modelo citado, se le aplica a una unidad software para ser desplegada pasando a llamarse el conjunto unidad de despliegue. Así, los recursos de bases de datos son paquetizados en una unidad de despliegue, conteniendo los archivos organizados, que será utilizada para desplegar la unidad software, en este caso el modelo de datos o los propios datos, sobre un contenedor software, en este caso una base de datos.

Los componentes desarrollados para la arquitectura siguen un diseño multicapa, con el objetivo de separar en la medida de lo posible las distintas responsabilidades. En concreto, cuenta con un diseño típico de tres capas: capa de presentación que utiliza la capa de lógica de negocio que a su vez se apoya en la capa de persistencia. Para esta interacción se emplea el patrón DAO (Data Access Object), que permite independizar las capas de la tecnología de implementación de la persistencia. Siguiendo este patron, la arquitectura impone la solución ORM (Object-Relational Mapping) que realiza la persistencia apoyándose en una base de datos relacional. De las soluciones ORM, la arquitectura permite utilizar dos tecnologías: JPA, anotaciones directamente en las clases Java, o Hibernate con el uso de archivos de mapeo específicos.

La arquitectura implementa el tratamiento de versiones sobre los servicios. El uso de versiones en los recursos de base de datos se hace necesaria ya que diferentes versiones de aplicaciones pueden necesitar diferentes modelos o datos de la base de datos. Así, aparecen dos conceptos asociados a las versiones que son únicos de base de datos: la desinstalación o vuelta atrás de una versión y la actualización entre versiones. La vuelta atrás, o rollback, necesita del conjunto de sentencias inversas al proceso de instalación para llevar a la base de datos al estado previo antes de la instalación. Es decir, para desinstalar un recurso de base de datos se necesita explícitamente las sentencias inversas a las sentencias que realizan la instalación. La actualización representa el conjunto de sentencias diferencia, es decir, las que llevan de un modelo de datos existente en la base de datos a otro de una versión posterior a ser desplegado con los menores cambios posibles. Lógicamente el conjunto de cambios de la actualización debe ser más reducido que si se eliminara completamente la versión anterior y se instalara de cero la nueva. A nivel DDL, esto también contribuye a que los datos existentes en la base de datos permanezcan tras la actualización de la versión ya que el proceso de destrucción de la versión anterior también elimina todos los datos existentes, si bien es cierto que la existencia de datos puede generar ciertos errores en el proceso de actualización. A través de la Fig. 1 se puede ver de forma fácil el uso de versiones en los recursos.

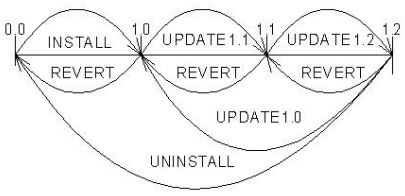

Figura 1: Uso de versiones en los recursos de base de datos

El sistema debe operar sobre un amplio espectro de bases de datos, así como el uso de tecnologías que permitan abstraerse del tipo de base de datos al definir el modelo de esquema. Es decir, el sistema debe dar soporte a múltiples SGBD. Algunas de las razones para tal motivación son que las empresas constantemente están buscando formas de mejorar y de obtener ventajas competitivas. A menudo, estas nuevas oportunidades de mejora pueden repercutir potencialmente en la infraestructura de TI básica de la empresa y requerir la migración de aplicaciones y bases de datos vitales a un entorno nuevo. La capacidad de desplegar un entorno nuevo con rapidez puede reportar considerables beneficios y ventajas a la empresa. La motivación de estas acciones puede ser debida a factores como la pérdida de fe en el proveedor actual, el bajo rendimiento de la plataforma o los elevados gastos de soporte. Ya que la arquitectura implementa el uso de versiones, otro de los objetivos es dotar al sistema de capacidades de control de versiones para poder realizar una navegación sobre ellas, incluido la posibilidad de regresar a una versión anterior conocida en caso de error.

Las actividades que demanda el marco impuesto para realizar el proceso íntegro de despliegue automático de recursos de base de datos encajan en un primer proceso que genere la unidad de despliegue y otro que realice el despliegue. Estos dos procesos se adecuan a las herramientas de ETL, extracción-carga-transformación, como procesos responsables del transporte, control e integración de datos de uno o más sistemas fuentes a uno o más sistemas destino, que contienen actividades de migración y de ejecución sobre bases de datos. La idea es que una aplicación ETL lea los datos primarios, realice su transformación o migración sobre un procesamiento cualitativo, abstrayéndose lo máximo posible de las bases de datos específicas, y al final cargue estos datos tratados en el almacén para que estén disponibles por los servicios.

B. Análisis de herramientas de despliegue y migración

Se ha hecho un análisis de las herramientas existentes que ofrecen procesos ETL y de migración de base de datos. Estas son las herramientas más importantes:

- Ruby on Rails[1]: para facilitar la administración de la DB, Rails pone a disposición las migraciones, que se pueden definir como clases de Ruby destinadas a la modificación del esquema de la base de datos. Rails contiene clases de migración que realizan operaciones de creación o de destrucción, permitiendo además mantener un control de versiones.
- CloverETL[2]: framework Java que puede ser utilizado para la transformación de datos estructurados. Las transformaciones están definidas en forma de gráfico que contiene descripciones metadata, secuencias, conexiones, componentes de transformación... Los metadatos pueden ser definidos, entre otras, automáticamente basándose en la estructura de una base de datos o con consultas SQL. El usuario selecciona de una paleta de componentes existentes y los coloca en una hoja de trabajo, luego los conecta a través de un gráfico de transformación. La representación gráfica es entonces traducida automáticamente por cloverGUI en código

[1] http://www.rubyonrails.org.es/
[2] http://www.cloveretl.com/

XML que posteriormente es ejecutado por el motor cloverETL.

- Enhydra Octopus[3]: herramienta basada en Java. Se puede conectar a cualquier fuente de datos JDBC y realizar transformaciones que se definen en un archivo XML. El modelo de datos DODS (Data Object Design Studio) es soportado generando identificadores para los nuevos objetos. Este modelo de datos es utilizado por Enhydra DODS, herramienta ligera de mapeo que soporta objetos relacionales persistentes para diferentes bases de datos. Se proporciona un generador de esquemas, e incluso archivos DODS, de una base de datos existente.
- Liquibase[4]: herramienta de software libre que se puede emplear para refactorizar bases de datos, es decir, resuelve el problema habitual de cómo sincronizar la información que se encuentra en diferentes bases de datos. Su funcionamiento se basa en crear un archivo XML que define los cambios que se han producido en la estructura y en la información de una base de datos respecto de otra. Liquibase proporciona una serie de funcionalidades: actualización de la base de datos, generación de rollback automático, validación base de datos, generación de documentación de BD...
- Pentaho Data Integration[5]: componente responsable de los procesos de ETL. Es la herramienta ETL más popular de código abierto disponible. PDI soporta una amplia gama de entrada y salida de formatos, incluyendo archivos de texto, hojas de datos y motores de bases de datos comerciales y libres. La capacidad de transformación de PDI le permite manipular los datos con muy pocas limitaciones. PDI es fácil de usar, cada proceso es creado con una herramienta gráfica donde se especifica qué hacer sin tener que escribir código. Posee una arquitectura escalable basada en plugins. PDI realiza además migración de datos entre bases de datos, exportación de datos a archivos planos, carga de datos de forma masiva, limpieza de datos, integración de aplicaciones y permite la definición de conjuntos de procesos referidos como ETTL.
- Talend Open Studio[6]: una potente solución que proporciona las capacidades de integración de datos avanzados. Su interfaz gráfica cuenta con numerosos componentes para el modelado del proceso de negocio, así como implementaciones técnicas para extraer, transformar y mapear flujos de datos. Posee una amplia escalabilidad al poderse realizar nuevos componentes para integrarlos con el sistema. Incluye funcionalidades interesantes como estar totalmente integrado con Eclipse, hacer transformaciones y mappings complejos, tener mecanismos de debugging y control sobre los procesos realizados, varios wizards para ayudar en todo el proceso, etc.
- Glashfish ESB: servidor de aplicaciones que implementa las tecnologías definidas en la plataforma Java EE. Posee dos subproyectos con capacidades de ETL, el más importante es:
  - OpenESB (ETLSE)[7]: dentro de esta plataforma existe una herramienta de integración de datos (ETLSE) que pueden utilizarse en los procesos ETL ya sea para construir almacenes de datos o migrarlos. ETLSE está diseñado para organizar y gestionar grandes volúmenes de datos, con transformaciones de alto rendimiento dentro de los niveles de SOA. Es un módulo empresarial optimizado para los procesos de ETL entre archivos y bases de datos. Soporta operadores de limpieza de datos para mantener la integridad de los datos. Para garantizar la calidad de los datos, ETL Integrator permite al usuario configurar la lógica de validación de datos. Su capacidad de procesamiento concurrente/paralelo permite manejar diferentes flujos de datos.
- Otras herramientas analizadas, que se omiten en la comparativa posterior debido a que sus características son englobadas en las anteriores, son Migrate4j (herramienta flexible de migración Java que pueden aplicarse a diferentes motores a la vez), Scriptella (herramienta ligera ETL con ejecución de scripts que se utilizan para actualizarse entre versiones), OpenDBCopy (utilidad de base de datos universal destinada a la migración basado en plugins que pueden encadenarse), KETL (plataforma de integración en Java como servidor multi-threaded que gestiona diversas tareas), Jitterbit (poderosa herramienta de ETL basado en operaciones sobre una GUI, incluido *drag and drop*), JasperETL (plataforma Open Source de Business Intelligence que incluye herramienta ETL con interfaz), Pentaho Data Integration (herramienta ETL más popular de código abierto disponible, cada proceso es creado con una herramienta gráfica sin tener que escribir código, poseyendo una arquitectura escalable basada en plugins), Apatar (innovadora y potente suite de herramientas de software diseñado para proporcionar beneficios de productividad a las organizaciones que necesitan mover datos de diferentes fuentes) y Mural (comunidad de código abierto con el fin de desarrollar un "ecosistema" de productos que solucionen los problemas de *Master Data Management*).

C. Comparativa

En la Fig. 2 se representa una tabla en la que se exponen las características más esenciales de las herramientas ETL. Se puede observar en las tablas características tales como los diversos modos de ejecución, control de versiones, el lenguaje de transformación, seguridad, si poseen interfaz gráfico, capacidad de generar documentación y monitorización. Se omiten datos referentes a la conectividad ofrecida pues todas las herramientas, aun con alguna diferencia, soportan las principales bases de datos.

D. Selección

La selección se realizó pensando en una herramienta de migración de bases de datos que debía integrarse en la arquitectura de configuración y despliegue. Además se deseaba que las migraciones o cambios realizados estuvieran gestionados

[3]http://www.enhydra.org/tech/octopus/index.html
[4]http://www.liquibase.org/
[5]http://kettle.pentaho.org/
[6]http://es.talend.com/index.php
[7]http://wiki.open-esb.java.net/Wiki.jsp?page=ETLSE

	Herramientas				Suites		
Nombre del producto	*Ruby*	*Octopus*	*CloverETL*	*Liquibase*	*TOS*	*KETTLE*	*Open ESB*
Control de Versiones							
• Por TAG	No	No	No	Si	No	No	No
• Por hora/fecha	No	No	No	Si	No	No	No
• Por últimos x cambios	No	No	No	Si	No	No	No
• Por nº de migración	Si	No	No	No	No	No	No
Lenguaje de Transformación							
• Java	No	No	Si	No	Si	No	No
• JavaScript	No	Si	No	No	No	No	No
• Ruby	Si	No	No	No	No	No	No
• XML	No	Si	Si	Si	No	No	No
• SQL	No	No	No	Si	No	No	No
• Perl	No	No	No	No	Si	No	No
• Gráfico. Librería de transformaciones	No	No	No	No	Si	Si	Si
Seguridad							
• Apache Commons logging	No	No	Si	No	No	No	No
GUI	No	No	Si. Clover.GUI	Si	Si	Si. Spoon	Si
Genera Documentación	No	No	No	Si	No	No	No
Validación de BD	No	No	No	No	Si. Mediante operaciones	Si. Mediante operaciones	Si. Mediante operaciones
Monitorización	No	No	No	No	No	Si. Herramienta Carte, remoto.	Si. ETL monitor

Figura 2: Tabla características herramientas ETL

de forma eficiente. Los motivos de ello eran mantener el control sobre las acciones llevadas a cabo y para poder deshacer de una forma sencilla las acciones realizadas. Teniendo en cuenta la tabla anterior y los requisitos brevemente descritos se seleccionó la herramienta Liquibase por sus capacidades de rastreo, gestión y aplicación de cambios en la base de datos. Otros motivos de la elección fueron principalmente:

- Su gran capacidad de integración con otros frameworks y entornos de desarrollo.
- Una amplia capacidad de gestión de versiones, que supera a la de Rails, con funciones adicionales.
- Es capaz de generar documentación asociada a los cambios a realizar, teniendo control sobre las acciones que se han tomado.
- Conexión con una amplia variedad de fuentes.
- Utiliza las capacidades de Hibernate para validar bases de datos, así como recrear el esquema de una base de datos.
- Retroceso de los cambios realizados, automáticamente genera las sentencias inversas necesarias.
- Realizar cambios que no han sido realizados con Liquibase. Genera automáticamente un informe para actualizar la base de datos, comparando una base de datos existente con archivos de mapeo Hibernate.
- Proporciona gran capacidad de acoplamiento en sistemas de despliegue. Con un único fichero de cambio puede generar los cambios en múltiples bases de datos.

III. Sistema implementado

El sistema implementado gestiona todo el proceso de manipulación y despliegue de los recursos de base de datos en la arquitectura. Los desarrolladores basan el funcionamiento de los servicios a desarrollar en la existencia de información almacenada en la base de datos. Dicha información está estructurada según un modelo de datos que es definido por el propio desarrollador o un analista siguiendo una serie de requisitos impuestos. Este modelo de datos representa un recurso DDL que será desplegado en las bases de datos para que los servicios correspondientes funcionen. Los servicios frecuentemente necesitan de datos existentes en la base de datos por lo que el despliegue de recursos DML es también parte del sistema.

El despliegue de estos recursos de base de datos en toda la arquitectura ITECBAN sigue siempre el mismo procedimiento. Este procedimiento es definido con una metología sencilla de dos pasos que se ejecutan de forma consecuente. Cada uno de los pasos define una tarea a realizar, que de forma general son explicadas en esta misma sección. Es decir, que estas dos tareas componen la metodología para desplegar los recursos de base de datos en la arquitectura de una forma automatizada.

Los recursos DDL suelen definirse, para mayor facilidad para el desarrollador, en un nivel más abstracto del que entiende el manejador de base de datos que, como se ha dicho, por requisitos de la arquitectura son las tecnologías JPA e Hibernate. Así, la primera tarea del sistema es la de traducir ese lenguaje abstracto en el que el desarrollador ha definido el modelo a un lenguaje entendible por la base de datos; además, toda la información, tanto la traducida como la previa, es encapsulada en una unidad de despliegue, paquete con una estructura estandarizada, para que sea entendida por la arquitectura global.

La siguiente tarea es, de forma consecuente, el despliegue de los recursos en la base de datos. Se ha implementado una herramienta encargada de coger la unidad de despliegue adecuada, analizarla y coger las sentencias en un lenguaje entendible por la base de datos y ejecutarlas sobre esta en función de una serie de propiedades o parámetros dados. Debido al dinamismo existente de los servicios Web, la arquitectura permite el desarrollo de los recursos en versiones asociadas a las unidades aumentando con ello la complejidad de los sucesivos despliegues.

El sistema se compone, por lo tanto, de dos herramientas que realizan respectivamente cada una de las dos tareas secuenciales. La primera consiste en la generación de la información sobre un paquete estructurado y la segunda es la que se encarga de desplegar los paquetes en la base de datos. Estas herramientas están implementadas sobre scripts Ant como consecución de múltiples tareas, entre ellas cabe

destacar las tareas que utilizan la herramienta Liquibase.

A. Herramienta de creación de unidades de despliegue

Liquibase no soporta directamente la transformación de clases anotadas por lo que se debe incluir para este caso una tarea intermedia proporcionada por Hibernate para traducir el conjunto de clases anotadas a un conjunto de archivos de mapeo. Sobre estos archivos de mapeo, Liquibase los traduce a su lenguaje específico, independiente del SGBD.

Debido a la importancia de las bases de datos es necesario tener asegurado que todo lo que se ejecute contra una base de datos esté validado. Además puede ser necesaria la modificación de algunos atributos del modelo de datos obtenido del desarrollo de la aplicación. Esto también puede venir impuesto por las limitaciones del SGBD concreto, por ejemplo, existen diferencia en la longitud máxima del nombre de las tablas entre diferentes SGBD y algunos nombres pueden resultar excesivamente largos. Por ello es necesario que el DBA se encargue de validar las unidades de despliegue. Además, las sentencias generadas pueden ser optimizadas debido a la experiencia del propio DBA y a su conocimiento específico de toda la arquitectura y de la base de datos en concreto, siendo este un proceso clave cuando el volumen de información sea muy elevado.

El problema surge en que, en un principio, Liquibase ha creado un fichero con su lenguaje específico, este lenguaje no supone ningún estándar y no suele ser entendido por los DBA, además de que se queda a un nivel de abstracción superior al que entiende el SGBD y por lo tanto si un DBA validara el archivo en ese lenguaje no estaría 100 % seguro de que se fuera a ejecutar todo correctamente ni optimizado. Así, Liquibase presenta la funcionalidad de traducir las sentencias a lenguaje SQL, que es el lenguaje por antonomasia que domina las bases de datos. El problema de SQL es que tiene sentencias que son específicas de cada SGBD. En el presente sistema, se almacenarán las sentencias en lenguaje Liquibase y en los scripts SQL para los SGBD que se necesiten (generalmente sólo uno), si más adelante se requieren otros SGBD se traducirán desde el lenguaje Liquibase. El DBA necesita validar cada script SQL creado para cada SGBD.

La herramienta genera de forma automática desde los archivos de persistencia una unidad de despliegue que contiene una estructura formalizada con el archivo en lenguaje Liquibase y los scripts SQL para cada uno de los SGBD. Liquibase generara no sólo el script SQL con las sentencias a ejecutar en la base de datos, script de instalación, sino el script SQL de rollback que contiene las sentencias inversas para eliminar esas acciones una vez ejecutadas en la base de datos.

Liquibase es capaz de generar el script SQL de actualización que supone la diferencia entre los archivos de persistencia de la nueva versión (del que surge el script de instalación) y la estructura de datos existente en la base de datos que supone una versión anterior. También es capaz de generar el script SQL de rollback de dicha actualización conteniendo las sentencias inversas que llevarán desde la nueva versión a la anterior.

Además, la herramienta es capaz de generar la documentación asociada a dichos scripts conteniendo información referente al estado actual de la base de datos y a los cambios que realizarán estos scripts. En la Fig. 3 muestra el funcionamiento más específico de esta herramienta con los archivos que genera.

La herramienta también debe generar un descriptor de la unidad de despliegue que corresponde con un archivo XML que sigue un formato entendible por la arquitectura de configuración y despliegue. Este descriptor define un modelo estándar con información relativa a la propia unidad: tipo de unidad, nombre, versión...

Por último, la herramienta presenta un proceso de modificación de los archivos generados por Liquibase debido a que éste algunas veces genera errores fácilmente detectables y modificables por un patrón. Así, este proceso permite definir un patrón de coincidencia para borrar o reemplazar dichos errores de una forma automática y cómoda. El proceso es representado en la Fig. 3 por el nombre Sql transform.

B. Herramienta de despliegue en base de datos

La arquitectura de configuración y despliegue ha de presentar las herramientas suficientes para desplegar cualquier tipo de unidad sobre cualquier tipo de contenedor. Específicamente, la herramienta implementada ejecuta las instrucciones necesarias para realizar las operaciones demandadas sobre los recursos DDL o DML en un contenedor del tipo base de datos. La arquitectura define para realizar el despliegue de unidades un plan de despliegue [16] que contiene una serie de actividades. Cada actividad representa una operación a realizar sobre una unidad de despliegue. Las operaciones permitidas sobre los recursos de bases de datos son: instalación (se instala un recurso con la versión indicada en la base de datos), desinstalación (se elimina completamente un recurso de la base de datos), actualización (se despliega un recurso con cierta versión basándose en una versión anterior ya desplegada) y desactualización (se lleva un recurso con cierta versión ya desplegado a una versión anterior).

Cada una de estas cuatro operaciones presentan scripts diferentes creados a lo largo del proceso de generación de la unidad de despliegue. La arquitectura mira cada actividad del plan de despliegue y coge la unidad de despliegue específica. Si la unidad es de base de datos, entonces llama a esta herramienta pasándole el contenido de la unidad, la operación a realizar y el contenedor de base de datos donde se ha de desplegar. Esta herramienta, de forma básica, establece la comunicación con la base de datos y ejecuta los scripts SQL contenidos en la unidad que representan la operación indicada.

La herramienta apoyándose en Liquibase es capaz de llevar un control de versiones de unidades desplegadas en base de datos. Cada vez que realiza una operación en base de datos añade una línea indicando la unidad desplegada, la operación, la versión en la que queda dicho recurso, el cheksum (hashing de los scripts ejecutados) y la fecha en la que se ejecutó. Si una operación se vuelve a repetir, la fecha se actualizará en la tabla de control.

La herramienta necesita de una serie de parámetros para realizar el despliegue en la base de datos: usuario y contraseña, url, tipo de SGDB... Estos parámetros son recogidos bien del archivo de propiedades que es definido en el modelo asociado al contenedor de base de datos o del descriptor existente en la unidad de despliegue. Como parámetros específicos

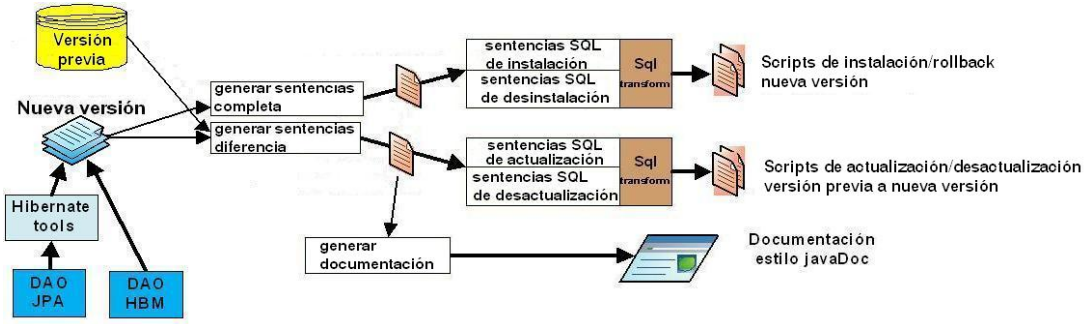

Figura 3: Funcionamiento de la herramienta de generación de unidades de despliegue

de la herramienta que configuran nuevas funcionalidades se encuentran dos. La primera, es la posibilidad de no ejecución de scripts modificados, si los scripts de una unidad de despliegue son modificados y cierta operación sobre esa unidad de despliegue ha sido realizada con anterioridad, al querer realizar el despliegue se comprueba la tabla de control de versiones, y al comparar los cheksums podrá impedir la ejecución de la operación. La segunda funcionalidad, es la elección del despliegue en modo transaccional o no, entendiendo una transacción como la capacidad que tiene la base de datos de hacer rollback automático si ocurre un error. A veces, las sentencias que se han ejecutado no tiene sentido darles vuelta atrás simplemente porque uno de las sentencias falle ya que la base de datos permanece en un estado correcto y es una pérdida de tiempo tener que hacer rollback automático de las sentencias ya ejecutadas para volverlos a ejecutar.

C. Mejoras en Liquibase

Liquibase aporta una gran funcionalidad a la gestión automática de tareas en base de datos y es una herramienta excelente para manejar muchos de los procesos a llevar a cabo. Pero como cualquier herramienta que no tenga demasiado tiempo de vida presenta fallos y errores, y más si se trata de software libre, por lo que para dar pleno rendimiento, ajustarse a la arquitectura y realizar las tareas previstas ha sido necesario solucionar errores y solventar ciertas deficiencias que presentaba esta herramienta. Todos estos cambios están en proceso de análisis dentro de la comunidad Liquibase para ser implementados de forma oficial. Siendo una herramienta de software libre, la obtención de todo su código es inmediata y siendo una comunidad muy abierta cualquier duda o problema que surgió era contestada de una forma eficaz por los desarrolladores más expertos.

Tres son las aportaciones principales al código:

- Mejora de los tipos de datos predeterminados desde Hibernate. Los archivos de persistencia presentan el modelo de datos siendo para ello necesario especificar el tipo de datos de cada elemento a representar. Para que quede bien definido el modelo, el correcto manejo de los tipos de datos es un punto clave. Uno de los problemas que tiene Liquibase es que la conversión de ciertos tipos de datos expuestos en los archivos de persistencia eran transformados incorrectamente a formato SQL y otros que aún transformándose correctamente a formato SQL entendible por el SGBD lo hacía con una precisión y escala en desacorde a lo que realmente representaba ese tipo de datos. Por este motivo, se ha decidido modificar el código para solucionar estos problemas de conversión.
- Cambios inversos a operaciones eliminatorias. Liquibase no es capaz de generar las sentencias SQL a cambios destructivos en la base de datos (acciones drop, alter drop/modify) en los scripts. Se añade esta funcionalidad, sacando la información del modelo de datos de la foto actual de la base de datos con lo que se pueden llenar los campos necesarios para generar las operaciones aditivas, es decir, las inversas a las destructivas.
- Soporte a modificación en los tipos de datos. La funcionalidad de modificar las columnas de las tablas se incluye en el cambio anterior pero hay que hacer más cambios para soportar las modificaciones en los tipos de datos. Primero se ha de definir la comparación entre los datos obtenidos desde el controlador de la base de datos y los datos generados desde los archivos de persistencia. Una vez se crea una tabla de sinónimos se extiende la comparación a la escala, longitud y precisión.

IV. Implementación y pruebas finales

La herramienta de generación de unidades de despliegue queda con un script Ant al que se le pasa los archivos de persistencia y crea la unidad de despliegue en la ruta elegida. Dicha unidad de despliegue es subida al repositorio de unidades.

La herramienta de despliegue en base de datos ha sido empaquetada en un bundle de OSGi. En el archivo de configuración se indica que exporta un servicio OSGi para ser utilizado por cualquier otro módulo de la arquitectura ITECBAN. El servicio exportado lanza las tareas Ant de la herramienta tras pasarle la operación a realizar, la unidad de despliegue y las propiedades de la base de datos. Para la integración en la arquitectura de despliegue, se añade un servlet configurado en Spring MVC que contiene el controlador que, tras mirar el plan de despliegue seleccionado y para las actividades de despliegue en base de datos, enlaza con el servicio implementado para que se encargue del despliegue una vez le pase los datos necesarios y permitiendo la visualización del proceso en la consecución de sucesivas páginas web. La Fig. 4 visualiza el resultado del despliegue de uno de los escenarios.

Las pruebas en esta parte han sido las correspondientes a cinco escenarios. La idea era crear un recurso DDL con dos versiones consecutivas y un recurso DML. Los recursos DDL fueron creados a partir de archivos de persistencia necesarios para llevar a cabo una aplicación financiera relacionada con

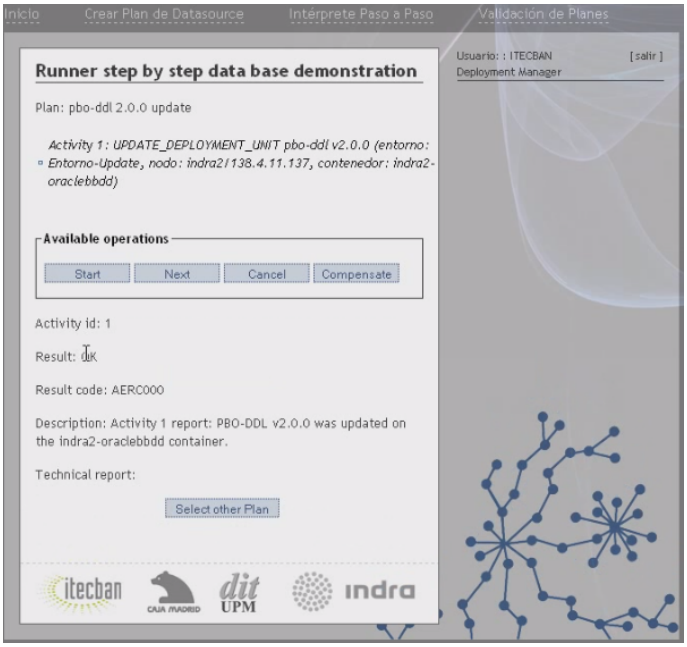

Figura 4: resultado del despliegue de un recurso sobre un contenedor de base de datos

la gestión de hipotecas. Las unidades de despliegue generadas fueron almacenadas en el repositorio real utilizado en la arquitectura. Las operaciones de despliegue, una por escenario, fueron instalación de un recurso DDL, carga de datos (DML), actualización de un recurso DDL, desactualización de un recurso DDL y desinstalación de un recurso DDL.

V. Conclusiones

Una actividad fundamental en una arquitectura SOA es el despliegue de servicios siendo conscientes de que para su funcionamiento existen dependencias con otros servicios y una serie de recursos ya desplegados. En el sentido de que todos los servicios necesitan comunicarse con la base de datos para coger y hacer persistente información del entorno, los recursos más demandados son los de bases de datos. Estos recursos son, por tanto, una pieza clave y su despliegue es una actividad compleja, costosa y realizada de forma manual por gente especializada presentando un alto coste asociado.

Dentro del proyecto ITECBAN surge la necesidad de facilitar el despliegue de sistemas del sector bancario. El proyecto implementa una arquitectura de configuración y despliegue, dentro de la cual se encuentra nuestro gestor específico de base de datos cuyo objetivo es gestionar de una forma automatizada todo el proceso de manipulación de los recursos de base de datos.

Siguiendo las líneas actuales que intentan encontrar métodos automáticos para las tareas en bases de datos, el sistema se basa en dos herramientas principales que intentan automatizar lo máximo posible las operaciones a realizar sobre la arquitectura de configuración y despliegue, ayudando en gran medida al DBA. La primera herramienta consiste en la generación de unidades de despliegue de base de datos desde archivos de persistencia conteniendo los scripts SQL a ser validados por el DBA y que serán desplegados, según la operación definida en la actividad planificada, por la segunda herramienta, añadiendo ésta un control de versiones.

Realizando un extenso análisis de las herramientas existentes de migración y de ETL, el sistema desarrollado se ha basado en Liquibase, solucionando ciertos errores y añadiendo cierta funcionalidad de tal forma que se ha adaptado el software libre existente y los cambios han sido ofrecidos de vuelta a la comunidad.

El sistema, conformado por las dos herramientas, se adecua a los objetivos de la arquitectura de configuración y despliegue que, a través de una metodología sencilla de dos pasos, realiza las tareas secuenciales para desplegar los recursos de base de datos en un ambiente automático.

Agradecimientos

Nos gustaría mostrar nuestro agradecimiento al Gobierno de España, por la financiación del proyecto ITECBAN (MITYC CDTI-CENIT 2005) a través del Ministerio de Industria, Turismo y Comercio.

Referencias

[1] S. Elnaffar, W. Powley, D. G. Benoit, and T. P. Martin, "Today's DBMSs: How autonomic are they?" in *DEXA Workshops*. IEEE Computer Society, 2003, pp. 651–655.

[2] S. Hashimi, "Service-Oriented Architecture Explained," O'Reilly Media, Inc., 2003. [Online]. Available: {http://ondotnet.com/pub/a/dotnet/2003/08/18/soaexplained.html}

[3] A. Mateen, B. Raza, T. Hussain, and M. Awais, "Autonomic computing in sql server," in *ICIS '08*. Washington, DC: IEEE Computer Society, 2008, pp. 113–118.

[4] B. Raza, A. Mateen, T. Hussain, and M. Awais, "Autonomic success in database management systems," *Computer and Information Science, ACIS*, vol. 0, pp. 439–444, 2009.

[5] M. Parashar and S. Hariri, "Autonomic computing: An overview," in *Unconventional Programming Paradigms*. Springer Verlag, 2005, pp. 247–259.

[6] IBM White Paper, "Practical Autonomic Computing: Roadmap to Self Managing Technology," 2006.

[7] X. Zhang, W. Sun, W. Wang, Y. Feng, and B. Shi, "Generating Incremental ETL Processes Automatically," in *IMSCCS '06*, vol. 2, june 2006, pp. 516 –521.

[8] T. Jörg and S. Dessloch, "Towards generating etl processes for incremental loading," in *IDEAS '08*. New York, NY, USA: ACM, 2008, pp. 101–110.

[9] M. Hernandez, L. Popa, H. Ho, and F. Naumann, "Clio: A schema mapping tool for information integration," in *ISPAN '05*. Washington, DC, USA: IEEE Computer Society, 2005, p. 11.

[10] Y. Wang, "DB2 query parallelism: Staging and implementation," in *21st VLDB Conference*, 1995, pp. 686–691. [Online]. Available: http://www.vldb.org/dblp/db/conf/vldb/Wang95.html

[11] K. Eva, L. Sam, S. Berni, S. Adam, W. Leanne, "Automatic Database Configuration for DB2 Universal Database: Compressing Years of Performance Expertise into Seconds of Execution," IBM Journal Paper, 2002.

[12] S. S. Lightstone, G. Lohman, and D. Zilio, "Toward autonomic computing with DB2 universal database," *SIGMOD Rec.*, vol. 31, no. 3, pp. 55–61, 2002.

[13] Steven Warren, "SQL Server Performance Monitor," Database Journal, 2005.

[14] D. G. Benoit, "Automatic diagnosis of performance problems in database management systems," in *ICAC '05*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 326–327.

[15] F. Cuadrado and J.C. Dueñas and R. García and J.L. Ruiz, "A Model for Enabling Context-Adapted Deployment and Configuration Operations for the Banking Environment," *Networking and Services Conference*, vol. 0, pp. 13–18, 2009.

[16] F. J. Blanco, L. D. Casillas, and M. Garijo, "A Knowledge-based System for the Validation of the Deployment of Software Units," in *ICAART (1)*, 2010, pp. 305–310.