IBM Cúram Social Program Management

Servidor de búsqueda genérico de Cúram

Versión 6.0.5

IBM Cúram Social Program Management

Servidor de búsqueda genérico de Cúram

Versión 6.0.5

Nota

Antes de utilizar esta información y el producto al que hace referencia, lea la información que figura en el apartado "Avisos" en la página 53

Revisión: mayo de 2013

Esta edición se aplica a IBM Cúram Social Program Management v6.0 5 y a todos los releases posteriores hasta que se indique lo contrario en nuevas ediciones.

Contenido

Figuras

Tablas

Capítulo 1. Introducción

1.1 Guía del Servidor de búsqueda genérico de Cúram

El Servidor de búsqueda genérico de Cúram es una herramienta proporcionada por IBM Corporation que se puede utilizar para desarrollar búsquedas eficientes y escalables para la solución de la aplicación.

Este documento describe el Servidor de búsqueda genérico de Cúram y proporciona una visión general de su arquitectura. También es una referencia para la configuración del Servidor de búsqueda genérico y sus tablas de base de datos. Por último, proporciona un ejemplo completo de cómo implementar una búsqueda utilizando el Servidor de búsqueda genérico de Cúram.

1.2 Requisitos previos

Los lectores de la Guía del Servidor de búsqueda genérico de Cúram deben estar familiarizados con la arquitectura de Cúram, además de estar familiarizados con el modelado de Cúram y las construcciones y procesos de desarrollo.

1.3 Público al que va dirigida

Este documento está orientado a arquitectos, diseñadores y desarrolladores interesados en utilizar el Servidor de búsqueda genérico de Cúram para implementar páginas de búsqueda.

Capítulo 2. Conceptos y definiciones

2.1 Introducción

Este capítulo presenta varios conceptos de búsqueda e indexación importantes, además de definiciones relacionadas con el Servidor de búsqueda genérico de Cúram que se utilizan en todo este documento.

2.2 El Servidor de búsqueda genérico

El Servidor de búsqueda genérico de Cúram es una aplicación autónoma que permite realizar búsquedas eficientes de datos de aplicación por medio de diversas API. Detrás de los bastidores, el Servidor de búsqueda genérico se implementa utilizando la API Apache Lucene. Las personas que implementan búsquedas GSS deben utilizar solo las API expuestos por GSS.

El Servidor de búsqueda genérico se puede desplegar como una aplicación Java™ simple (para facilitar las pruebas durante el tiempo de desarrollo), y también como una aplicación J2EE.

2.3 Índices

En el corazón del Servidor de búsqueda genérico se encuentra el concepto de buscar en un índice, que es una representación eficiente y que no es de base de datos de un conjunto de datos en los que se pueden realizar búsquedas relacionadas. Un índice de Servidor de búsqueda genérico es un "índice invertido" que correlaciona palabras con registros de la base de datos en la que aparecen.

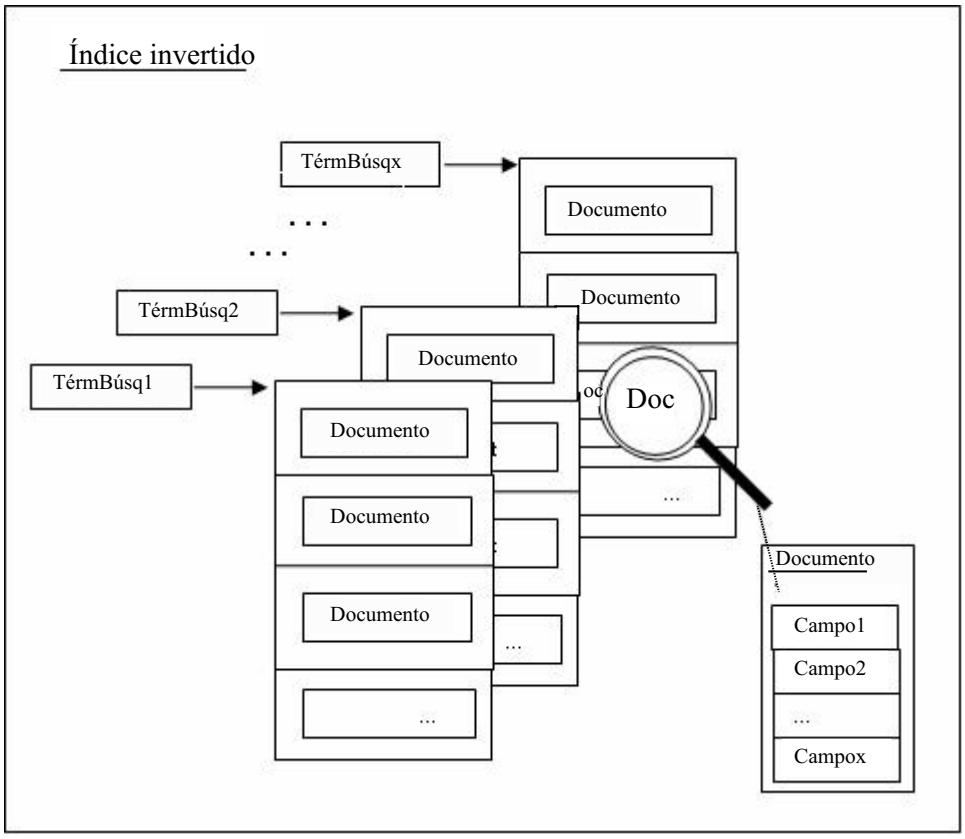

Figura 1. Descripción de índice invertido

Al buscar una palabra en un índice, se recuperan todos los registros coincidentes sin tener que buscar en grandes conjuntos de datos. Como resultado, estos índices se escalan bien, y para los sistemas grandes será posible ejecutar varios índices en paralelo, lo que permite un rendimiento de búsqueda excelente si se seleccionan la configuración de despliegue y los parámetros de ajuste de índice correctos.

Los desarrolladores que crean búsquedas de aplicación no manipulan ni mantienen los índices directamente, todo esto lo gestiona automáticamente el Servidor de búsqueda genérico tras bastidores.

2.4 Servicio de búsqueda

Un servicio de búsqueda describe:

1. Información relacionada con los campos en los que se está buscando
2. Los analizadores utilizados en cada campo, los tipos de datos de los campos
3. Información de entidades para llenar un índice de tiempo de ejecución
4. Estado del servicio de búsqueda (si está actualizado o si requiere sincronización)

Cuando se ve de este modo, un servicio de búsqueda son simplemente metadatos, no obstante, este documento también utiliza el término para describir el índice llenado en tiempo de ejecución.

Debería haber un servicio de búsqueda definido para cada conjunto de datos discreto en el que se debe buscar (por ejemplo, búsqueda de persona, búsqueda de pago, etc.). Cada búsqueda realizada debe especificar sobre qué servicio de búsqueda debe funcionar.

2.5 Campo

Tal como se ha mencionado anteriormente, los servicios de búsqueda constan de conjuntos de campos. Se pueden considerar algo análogo a las definiciones de columna de las tablas de base de datos. Un campo tiene un nombre y un tipo, y si se devuelve desde una búsqueda también tendrá un valor, que es el resultado.

Los campos se pueden marcar como almacenados. Los campos marcados de este modo harán que el índice contenga físicamente valores relevantes (consulte el apartado 2.13, “Extractor”, en la página 6) extraídos de la base de datos. Esto significa que sus valores se pueden recuperar directamente del índice después de una búsqueda y devolverse al remitente sin la necesidad de acceder al registro relacionado en la tabla de base de datos de la aplicación. No obstante, tenga en cuenta que esto no aumenta el tamaño de índice y puede afectar al rendimiento de la búsqueda.

Los campos también se pueden marcar como indexados o no. Se pueden realizar búsquedas en los campos que están marcados de este modo pero no en los que no lo están. Esta característica resulta útil para los campos como por ejemplo ID exclusivos que puede resultar deseable almacenar en el índice, pero en los que no se puede buscar.

Tenga en cuenta que los campos no tienen que estar marcados como almacenados para que se puedan realizar búsquedas en ellos.

2.6 Documento

Un documento es un registro en un índice. A su vez, un documento consta de un conjunto de campos. Los resultados de la búsqueda se devuelven del Servidor de búsqueda genérico como conjuntos de documentos que se pueden convertir luego en objetos de estructura de Cúram. Por ejemplo, un documento de búsqueda de persona puede constar de los campos Nombre, Apellido, Dirección, Sexo, etc., y la realización de una búsqueda/consulta de persona (consulte el apartado 2.9, “Consulta”, en la página 5) basada en diversos criterios de entrada devolverá cero o más documentos del tipo en cuestión.

2.7 Lucene

Lucene es un proyecto de código abierto creado por la Apache Software Foundation. Detrás de los bastidores, el Servidor de búsqueda genérico de Cúram utiliza Lucene para su funcionalidad de indexación y búsqueda.

Nota: Tenga en cuenta que la información sobre la indexación y Lucene se proporciona exclusivamente con fines de información de fondo; los desarrolladores que crean búsquedas utilizando el Servidor de búsqueda genérico no necesitan manipular índices ni objetos de Lucene directamente. Todos estos quedan envueltos por la API del Servidor de búsqueda genérico.

2.8 Base de datos de transferencia

La base de datos de transferencia del Servidor de búsqueda genérico consta de un conjunto de tablas de base de datos utilizadas para los fines siguientes:

- Para almacenar definiciones del servicio de búsqueda - información acerca de qué servicios de búsqueda están disponibles junto con su estructura
- Para almacenar valores extraídos de la base de datos operativa que se utilizarán para llenar índices correspondientes a las definiciones del servicio de búsqueda.

Los principios de diseño fundamentales para utilizar tablas de base de datos como intermediario son los siguientes:

- Descargan las búsquedas de la base de datos principal, lo que significa que las búsquedas no afectan al rendimiento del sistema activo
- Permanecen adecuadamente para el servicio de búsqueda - los datos persisten en un formato adecuado para el fin de la creación de los índices de búsqueda. Los datos de la aplicación se transforman, barren y consolidan antes de que se almacenen en la base de datos de transferencia. Por lo tanto, los trabajos por lotes no se tendrán que volver a ejecutar continuamente para volver a extraer los datos cada vez que se inicia una instancia del Servidor de búsqueda genérico.

2.9 Consulta

Una consulta es un objeto (una estructura, para ser exactos) que se pasa al Servidor de búsqueda genérico cuando se realiza una búsqueda.

2.10 Término

Un término es una parte de un objeto de consulta. En la actualidad, hay tres tipos distintos de términos: términos estándar (Standard) para la búsqueda en campos de texto normales, términos de fecha (Date) para la búsqueda en campos de fecha y términos de rango de fechas (DateRange) para especificar un rango de fechas en el que buscar.

2.11 Analizador

Un analizador es un concepto de Lucene, que representa una clase que implementa la clase abstracta `org.apache.lucene.analysis.Analyzer` de Lucene.

Los analizadores preparan el texto para la indexación y la búsqueda. Por ejemplo, no tiene sentido que cada palabra de un campo de texto se indexe: palabras de detención, como “y”, “de” y “un” pueden resultar irrelevantes durante una búsqueda. Si se deben ignorar durante una búsqueda en un campo, entonces el campo se simboliza, es decir, pasa a través de un analizador antes de escribir el campo en el índice y lo mismo sucede para un valor de término en el que se están buscando.

Los analizadores son específicos del idioma, lo que define que una palabra no es igual en todos los idiomas. Algunos pueden configurarse para ignorar palabras comunes a evitar (un, el, si, etc.), para ignorar números, etc. Los analizadores utilizados por el Servidor de búsqueda genérico se pueden configurar para cada servicio de búsqueda.

2.12 Correlacionador

Un correlacionador es una clase que tiene que ser escrita por los desarrolladores de búsquedas de aplicación para cada servicio de búsqueda. Su función consiste en transformar los datos de la aplicación en un formato que pueda escribirse en la base de datos de transferencia e importarse en un índice. La transformación implica identificar las propiedades de entidad relevantes de interés para el servicio de búsqueda, construir una lista de estos valores y correlacionarlos con un solo valor de texto consolidado. Este valor, almacenado en la base de datos de transferencia, se utiliza posteriormente en la construcción de un documento de índice de búsqueda único. Cada servicio de búsqueda que se escribe debe proporcionar su propia implementación de correlacionador.

2.13 Extractor

El extractor utiliza los metadatos del servicio de búsqueda para obtener los datos de aplicación relevantes necesarios para llenar los índices de búsqueda. El extractor interroga a las entidades de aplicación relevantes identificadas por medio de los metadatos y las propiedades de entidad necesarias se correlacionan (mediante el correlacionador) con la base de datos de transferencia para indexar al iniciarse el servicio de búsqueda.

Capítulo 3. Visión general del Servidor de búsqueda genérico

3.1 El Servidor de búsqueda genérico y Lucene

Los conceptos que hay tras la indexación y la API de Lucene ya se han presentado. Por lo tanto, ¿por qué no utilizar Lucene directamente en una aplicación Cúram?

Aunque Lucene es una API excelente para la indexación y las búsquedas, no hace frente a todos los requisitos de un producto de búsqueda de Cúram:

- No hace frente a las cuestiones de despliegue - cómo ejecutar varios servidores de búsqueda, cómo debe comunicarse la aplicación con los servidores de búsqueda, etc.
- No hace frente a la cuestión de cómo importar datos en índices
- No hace frente a la cuestión de cómo mantener los datos de índice sincronizados con datos de origen en la aplicación en ejecución.
- No hace frente a la cuestión de cómo interpretar los datos devueltos por una búsqueda de índice como tipos de datos y estructuras de Cúram.
- No hace frente a la necesidad de aplicación más general de proteger al desarrollador de aplicaciones frente a la necesidad de disponer de conocimientos profundos de productos específicos de terceros; puesto que Lucene solo es una solución de búsqueda potencial, parecería tener más sentido proporcionar una API de búsqueda más genérica.

El Servidor de búsqueda genérico de Cúram se ha desarrollado para hacer frente a estas necesidades.

3.2 Importación de datos de Cúram

Una consecuencia del uso de una tecnología de indexación es que, para poder buscar en un índice, en primer lugar se debe crear. Puesto que una gran cantidad del trabajo de búsqueda se realiza esencialmente por adelantado en la construcción del índice, las búsquedas en tiempo de ejecución resultan más rápidas; no obstante, cabe tener en cuenta que el propio proceso de indexación puede tardar algún tiempo, y este tiempo aumenta proporcionalmente con la cantidad de datos que se deben indexar.

La inicialización del Servidor de búsqueda genérico se realiza en dos fases.

En la primera fase, los datos de la aplicación existentes se exportan de la aplicación a un conjunto de tablas de base de datos utilizado por el Servidor de búsqueda genérico: las tablas base. Esta exportación se ha implementado como un proceso por lotes, que se conoce como el extractor de búsqueda de base de datos, y se proporciona como parte de la distribución del Servidor de búsqueda genérico. Solo es necesario realizar la exportación una vez, cuando el Servidor de búsqueda genérico se utiliza por primera vez. Se necesitan clases auxiliares especiales denominadas correlacionadores para cada servicio de búsqueda; estas ayudan al extractor a preparar los datos que deben importarse en las tablas base.

En la segunda fase, se construye un índice para cada servicio de búsqueda definido. Cuando se inicia el servidor de búsqueda genérico, se ejecuta un proceso para leer los datos adecuados de las tablas de la base de datos de transferencia y construir los índices y otras estructuras de datos que se utilizarán para realizar búsquedas. Una vez construidos los índices, el servidor estará en condiciones de responder a las solicitudes de búsqueda. Encontrará información sobre cómo optimizar este rendimiento en el apartado Capítulo 12, “Rendimiento”, en la página 43

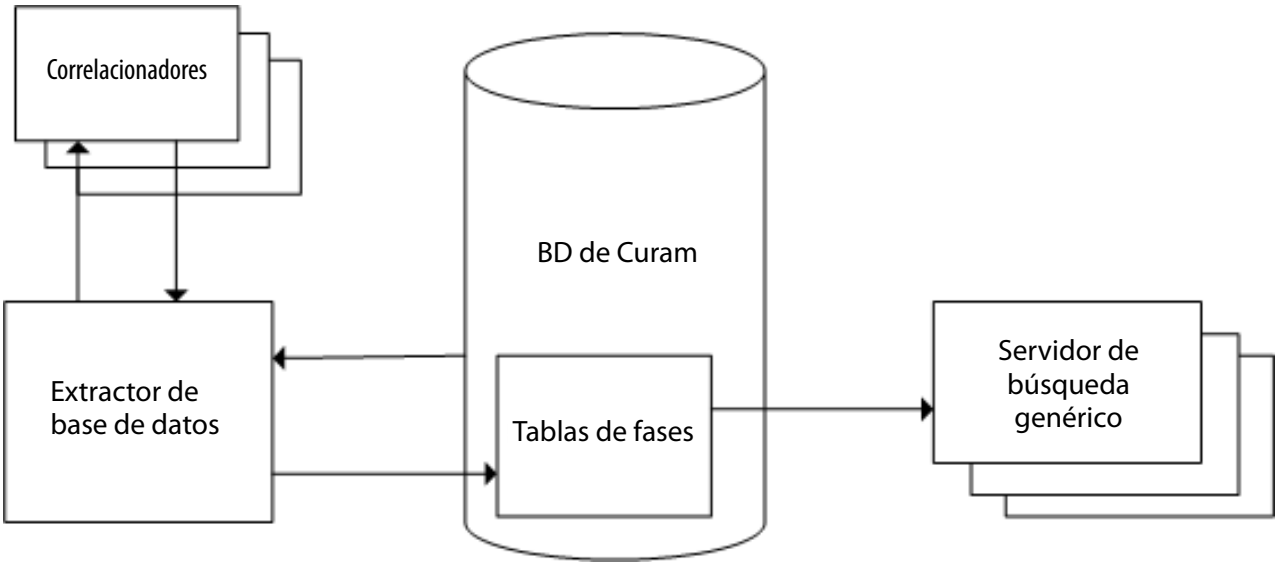

Figura 2. Proceso de inicio del extractor de base de datos y el Servidor de búsqueda genérico

3.3 Sincronización del Servidor de búsqueda

Debido a que el Servidor de búsqueda genérico busca no en los propios datos activos sino en un índice que se crea a partir de dichos datos, las actualizaciones de los datos de aplicación deben replicarse en el índice. En las implementaciones de Cúram, resulta esencial que las actualizaciones de los datos en los que se puede buscar se reflejen en los índices relevantes de forma oportuna y predecible. Con el Servidor de búsqueda genérico, el retardo de tiempo es breve (y configurables).

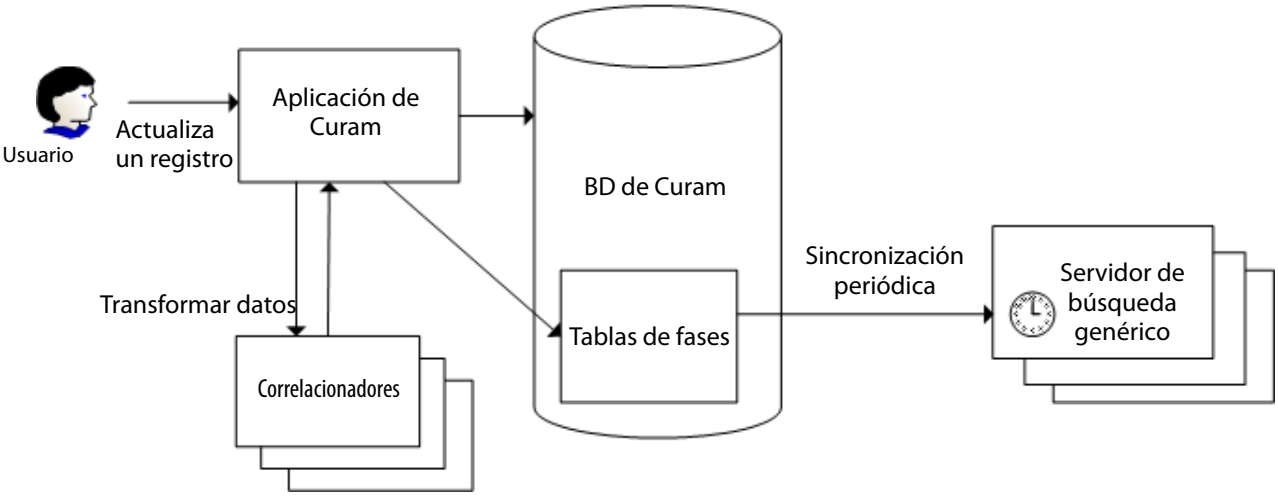

Figura 3. Sincronización de datos

De forma similar a la importación inicial de datos descrita anteriormente, hay dos pasos para el proceso de sincronización.

El primer paso del proceso se produce cuando los datos de la aplicación (que se utilizan en un índice) cambian, normalmente como resultado de una inserción, actualización o supresión lógica. Cuando esto

ocurre, la aplicación debe escribir información acerca de este cambio de datos en las tablas del Servidor de búsqueda genérico. Todos los elementos nuevos y actualizados están marcados con una indicación de fecha y hora.

En el segundo paso (que se produce periódicamente), el Servidor de búsqueda genérico sincroniza sus índices sobre el contenido actual de la base de datos de transferencia. Para hacerlo, lee todos los elementos cambiados recientemente modificados desde la última vez en que se sincronizó y los importa en los índices; concretamente, esto se lleva a cabo comparando las indicaciones de fecha y hora asociadas con cada elemento modificado con la última indicación de fecha y hora utilizada durante el paso de la última sincronización.

Nota: Al escribir pruebas de unidad que incluyan llamadas a búsquedas del Servidor de búsqueda genérico, es importante tener en cuenta el retardo en la sincronización de los datos. Además, como consecuencia del hecho de que la instancia del Servidor de búsqueda genérico se estará ejecutando en un proceso separado de las pruebas de unidad, no formará parte de la misma transacción. Por lo tanto, las sincronizaciones del Servidor de búsqueda genérico no recopilarán datos que hayan cambiado en la transacción de prueba, a menos que se comprometa explícitamente.

3.4 Controlador de búsqueda

El Controlador de búsqueda es un componente importante del mecanismo de sincronización. Mantiene una lista de todas las entidades asociadas con cada servicio de búsqueda.

Cuando una entidad cambia, puede comprobarse el Controlador de búsqueda para ver si esa entidad es utilizada por uno o más servicios de búsqueda. Si se utiliza, los datos de la base de datos de transferencia se deben actualizar en la misma transacción que la actualización de la entidad. El Controlador de búsqueda también proporciona una API para actualizar la base de datos de transferencia.

Nota: Diversas entidades de la plataforma Cúram (que aparecen en algunas búsquedas de la plataforma Cúram) se han modificado para permitir la implementación de actualizaciones de sincronización de este tipo en el release futuro. Estas modificaciones han tomado la forma de la creación de puntos de salida previos o posteriores a la operación que contienen implementaciones con apéndice; estos puntos de salida previos y posteriores están reservados para la implementación en el futuro y los clientes no deben cambiarlos directamente.

3.5 El proceso de búsqueda

El proceso de búsqueda puede dividirse en tres fases.

En la primera fase, la aplicación Cúram crea una consulta válida para presentarla al Servidor de búsqueda genérico. Llena esta consulta utilizando criterios de búsqueda especificados por el usuario.

En la segunda fase, la aplicación Cúram contacta con una instancia del Servidor de búsqueda genérico en ejecución y realiza la búsqueda tal como la define el objeto de consulta.

En la fase final, la aplicación Cúram interpreta los resultados que recibe del Servidor de búsqueda genérico como tipos de datos de Cúram, realiza sus comprobaciones de seguridad habituales con respecto a la confidencialidad de los datos y los muestra al usuario.

3.6 Referencias

Sitio web de Lucene: http://lucene.apache.org/.

Capítulo 4. Búsquedas habilitadas para el Servidor de búsqueda genérico

4.1 Introducción

IBM Corporation ha introducido el Servidor de búsqueda genérico como un mecanismo de búsqueda opcional para búsquedas de plataforma y módulo de solución. Se han implementado varias búsquedas utilizando tanto el Servidor de búsqueda genérico de Cúram como la búsqueda de base de datos, y algunas están disponibles sólo como búsquedas de GSS. Para las búsquedas que están disponibles como búsquedas de base de datos o GSS, los clientes pueden habilitar o inhabilitar la búsqueda eficiente búsqueda por búsqueda por medio de las propiedades de la aplicación.

4.2 Propiedades relacionadas con el Servidor de búsqueda genérico en la aplicación Cúram

Estas propiedades son las propiedades del sistema de aplicaciones y se pueden administrar de la forma habitual a través de la administración de la propiedad en la aplicación. Todas las propiedades relevantes están disponibles en la categoría denominada “Solicitud: parámetro de búsqueda mejorada de Lucene”. Encontrará una lista completa de estas propiedades en el apartado A.1, “Propiedades de configuración”, en la página 47

Tabla 1. Propiedades relacionadas con el Servidor de búsqueda genérico de Cúram

Nombre de propiedad	Descripción
curam.lucene.luceneEnhancedSearchEnabled	Valor predeterminado: “NO”. De forma predeterminada, toda la funcionalidad del Servidor de búsqueda genérico está inhabilitada. Para habilitarla, debe establecer esta propiedad en “YES” para activar una búsqueda mejorada. A menos que esté establecida en “YES”, no habrá búsquedas mejoradas disponibles.
curam.lucene. luceneOnlineSynchronizationEnabled	Valor predeterminado: “NO”. Para habilitar el mecanismo de publicación de sucesos que realiza cambios en datos que se pueden buscar disponibles para el Servidor de búsqueda, debe establecer esta propiedad en “YES”. Si no lo hace, las inserciones y actualizaciones de los datos que se pueden buscar no se propagarán al Servidor de búsqueda.
curam.lucene.externalUpdateEventsEnabled	Valor predeterminado: “NO”. Para asegurarse de que si algún dato relacionado con el servicio de búsqueda se actualiza externamente, el sistema externo recibirá sucesos de sincronización de actualización relacionados con la sincronización de los datos que se pueden buscar, si la propiedad "curam.lucene.luceneOnlineSynchronizationEnabled" no está habilitada. La habilitación de esta propiedad tiene el mismo efecto que habilitar “curam.lucene.luceneOnlineSynchronizationEnabled” en la aplicación. Para habilitar la propiedad “curam.lucene.externalUpdateEventsEnabled” establezca esta propiedad en “YES”.

Por último, cada búsqueda que da soporte a la búsqueda mejorada tiene una propiedad que determina si utiliza el Servidor de búsqueda genérico o la base de datos. Esto permite que cada organización elija búsqueda por búsqueda qué búsquedas mejoradas utilizará.

4.3 Cómo mantener sincronizados los datos de Cúram y los datos de búsqueda

Es necesario mantener los datos de la aplicación activa y el índice de búsqueda sincronizados si los resultados de la búsqueda tienen que ser precisos. la infraestructura que el GSS proporciona para conseguir este objetivo se han descrito en otro lugar (consulte el apartado 6.3, “Controlador de búsqueda”, en la página 18).

No obstante, los desarrolladores de aplicaciones también tienen la responsabilidad de añadir llamadas al SearchController cuando cambian datos relevantes en la aplicación. En esta sección se describe con fines informativos el enfoque basado en sucesos utilizado y que recomendamos a los clientes que implementan sus propias búsquedas basadas en GSS.

Además del mecanismo de sucesos también proporcionamos la sincronización del Correlacionador de extracción, que se describe en su propio capítulo en esta guía, consulte el apartado Capítulo 8, “Correlacionador de extracción”, en la página 29.

4.3.1 Sincronización basada en sucesos

Cúram proporciona sucesos para permitir que los componentes débilmente acoplados de la aplicación se proporcionen información entre sí acerca de los cambios de estado. Se documentan en la publicación *Cúram Server Developer's Guide.*.

Para cada entidad que contribuye a un servicio de búsqueda se deben emitir sucesos cuando se crea, suprime o modifica. El manejador de sucesos llama luego a la clase `SearchController` para actualizar el servidor de búsqueda con el cambio.

Cualquier entidad que contribuye a un servicio de búsqueda debe tener operaciones de modificación posterior, inserción posterior y eliminación posterior añadidas que emitan los sucesos.

Capítulo 5. Tablas de la base de datos de transferencia

5.1 Introducción

Las tablas de base de datos de transferencia son tablas de base de datos de la base de datos operativa utilizadas por el Servidor de búsqueda genérico. Existen cuatro tablas de este tipo: SearchService, SearchServiceField, SearchServiceRow y SearchSrvcRowExt.

Este capítulo detalla el objetivo y la estructura de las tablas SearchService y SearchServiceField. Los desarrolladores que crean servicios de búsqueda no necesitan acceder a las tablas SearchServiceRow o SearchSrvcRowExt directamente, ni grabar archivos DMX para ellas.

La tabla SearchService define los servicios de búsqueda conocidos por el Servidor de búsqueda genérico (consulte el apartado 2.4, "Servicio de búsqueda", en la página 4 para ver una introducción a los servicios de búsqueda). Puesto que no se ha proporcionado una API de administración para gestionar los servicios de búsqueda, los registros del servicio de búsqueda actualmente se deben crear y mantener accediendo a la tabla de base de datos directamente o editando los archivos DMX y volviendo a crear la base de datos de la aplicación.

La tabla SearchServiceField define un campo único de un servicio de búsqueda: su nombre, su tipo de datos y varios otros atributos que se describen de forma completa a continuación. Cada fila de la base de datos SearchServiceField está asociada con una sola fila SearchService. Al igual que con los servicios de búsqueda, los registros del campo del servicio de búsqueda actualmente se deben crear y mantener accediendo a la tabla de base de datos directamente o editando los archivos DMX y volviendo a crear la base de datos de la aplicación.

SearchServiceRow es una tabla que se utiliza para almacenar datos que se pueden buscar de la aplicación para su uso en la construcción de índices. El Servidor de búsqueda genérico proporciona una API (consulte el apartado Capítulo 6, "Iniciación a la API Servidor de búsqueda genérico", en la página 17 y Capítulo 7, "Implementación de una búsqueda con el Servidor de búsqueda genérico", en la página 23) que se utiliza para manipular SearchServiceRows; los desarrolladores deben interactuar con esta tabla de base de datos solo a través de esta API en lugar de acceder directamente.

Hay dos otras tablas de base de datos de GSS: GSSMapperType y GSSEntity. Solo se utilizan con la característica del Correlacionador de extracción; de lo contrario se pueden ignorar. Estas tablas se describen en el apartado Capítulo 8, "Correlacionador de extracción", en la página 29.

5.2 Tabla SearchService

Cada servicio de búsqueda debe contener un registro en la tabla SearchService. Junto con sus filas hijas SearchServiceField, la tabla SearchService define el esquema para cada servicio de búsqueda. A continuación se proporciona una descripción de cada columna de la tabla SearchService:

5.2.1 searchServiceId

El identificador del servicio de búsqueda; una serie utilizada para identificar de forma exclusiva un servicio de búsqueda.

5.2.2 extKeyName

El nombre de un campo de servicio de búsqueda que identificará de forma exclusiva cada registro de un índice creado a partir de esta definición de servicio de búsqueda. Es esencial que los valores del índice correspondientes a este campo de servicio de búsqueda sean exclusivos, puesto que cuando los datos en

los que se pueden realizar búsquedas se actualicen en la aplicación, el valor de este campo se utilizará para identificar el documento adecuado que se debe actualizar en el índice.

5.2.3 analyzer

El analizador del servicio de búsqueda que debe utilizarse al convertir de los términos de texto de la base de datos de la aplicación a los términos del índice de texto. El contenido de esta columna debe indicar uno de los nombres de analizador predefinidos proporcionados por el Servidor de búsqueda genérico (consulte la lista siguiente) o un nombre de clase Java totalmente calificado de una clase que implementa la clase abstracta `org.apache.lucene.analysis.Analyzer`. Puede ser un analizador de Lucene estándar o una implementación de terceros o personalizada. Tenga en cuenta que la clase debe estar disponible en la ruta de clase del Servidor de búsqueda genérico si no es un analizador estándar Lucene.

Para obtener una lista de los analizadores proporcionados con GSS y una explicación más detallado sobre cómo elegir un analizador, consulte el apartado 9.6, "Los analizadores en profundidad", en la página 34.

5.2.4 frcdReidxTimeStmp

Lo utiliza el Extractor para forzar que el Servidor de búsqueda genérico reconstruya sus índices después de que se haya ejecutado una extracción. Al crear registros del servicio de búsqueda, se debe establecer inicialmente como nulo.

5.2.5 mapperName

El nombre de la implementación de correlacionador (consulte el apartado 7.4, "Implementación de operaciones del correlacionador", en la página 24). Una implementación de correlacionador es una clase que convierte un conjunto de datos de entidad de aplicación en un formato adecuado para la indexación. El valor de esta columna debe ser el nombre de clase totalmente calificado de la clase de correlacionador, y del mismo modo que la implementación del analizador, debe estar en la vía de acceso de clases de tiempo de ejecución del Servidor de búsqueda genérico (si el correlacionador se ha desarrollado como parte de la aplicación, estará en la vía de acceso de clases de forma predeterminada).

5.2.6 dbLastWritten

Se utiliza en la sincronización. No debe ser inicializado ni actualizado por código de aplicación ni por administradores.

5.2.7 prstBlobSize

Especifica el tamaño del objeto binario grande asociado con la tabla utilizada para hacer que persista este índice de servicio de búsqueda. Si no se especifica, el tamaño predeterminado del objeto binario grande es de 50M. El tipo de la propiedad es una serie y el valor debe ajustarse a la sintaxis del especificador de tamaño de la base de datos en cuestión.

5.3 Tabla SearchServiceField

Cada campo de un servicio de búsqueda debe contener un registro en la tabla SearchServiceField. Cada campo del servicio de búsqueda representa un elemento SearchService en el que se puede buscar, que se puede devolver de una búsqueda, o ambos. Los campos del servicio de búsqueda se utilizan en distintos lugares de todo el Servidor de búsqueda genérico: en términos, en consultas, en documentos. A continuación se proporciona una descripción de cada columna de la tabla SearchServiceField:

5.3.1 srchServiceFldId

El identificador exclusivo del campo del servicio de búsqueda.

5.3.2 searchServiceId

searchServiceId del registro del servicio de búsqueda padre.

5.3.3 name

El nombre asociado con el campo del servicio de búsqueda. Es el nombre que se utiliza para hacer referencia al campo al realizar búsquedas o recuperar resultados. No es necesario que corresponda exactamente con los nombres de campo en entidades y estructuras de Cúram, aunque si lo hace simplifica el desarrollo.

5.3.4 type

El tipo de datos de Cúram de este campo. El conjunto de valores aceptables se describe en la tabla siguiente.

El proceso de exportación y sincronización de datos para el servicio de búsqueda implica alguna conversión de los datos operativos en series y viceversa, por lo tanto, es importante que se defina un tipo de datos preciso para cada campo. Consulte la tabla siguiente para obtener información al respecto. Si se presentan valores incorrectos al Servidor de búsqueda genérico, se emitirá una excepción.

Tabla 2. Correlaciones de definiciones de dominio de Cúram básicas con tipos de datos de campos de GSS

Definición del dominio	Tipo de datos del campo de GSS
SRV_BOOLEAN	boolean
SRV_DATE	Date
SRV_DATETIME	DateTime
SRV_INT8	byte
SRV_INT16	short
SRV_INT32	int
SRV_INT64	long
SRV_FLOAT	float
SRV_DOUBLE	double
SRV_MONEY	Money
SRV_CHAR	char
SRV_STRING	String
SRV_UNBOUNDED_STRING	String

Nota: El campo de tipo es sensible a las mayúsculas y minúsculas, por lo que debe asegurarse de utilizar el nombre del tipo exactamente tal como se ha especificado anteriormente.

5.3.5 indexed

Indica si se pueden realizar búsquedas en este campo. A veces puede ser conveniente almacenar un valor para un registro en el servicio de búsqueda, pero no realizar búsquedas en él (un ejemplo sería el ID exclusivo de un registro o quizás su nivel de confidencialidad). Los valores que no son de índice que no se deben indexar minimizarán el tamaño del índice y contribuirán al rendimiento, por lo que es una buena práctica indexar solo los campos que sus búsquedas utilizarán.

5.3.6 stored

Indica si este campo se puede devolver en un resultado de búsqueda o no, es decir, si el propio valor se almacena en el índice o no. Tenga en cuenta que los campos almacenados igualmente se devolverán únicamente si el objeto de consulta pasado al Servidor de búsqueda genérico indica que se deben devolver. Cada campo debe ser indexado o almacenar o de ambos tipos, si no es de ninguno de los dos tipos, entonces no es relevante para el servicio de búsqueda. Una vez más, si no se almacenan valores que sus búsquedas no utilizarán se minimizará el tamaño de índice y contribuirá al rendimiento, por lo que solo debe almacenar los campos que sus búsquedas utilizarán.

5.3.7 entityName

El nombre de la entidad de la aplicación asociada con este campo o, para ser más específico, el nombre de la entidad de aplicación que contiene un atributo correspondiente a este campo que se utilizará para llenar el índice basándose en la definición del servicio de búsqueda padre. Esta información es necesaria para la sincronización de los datos de la aplicación con el Servidor de búsqueda genérico; todas las entidades que aparecen como relacionadas con los campos del servicio de búsqueda se registrarán con el SearchController (consulte el apartado 3.4, "Controlador de búsqueda", en la página 9) y se supervisarán para identificar inserciones, actualizaciones y supresiones. Es de vital importancia que el atributo entityName se llene con los valores adecuados; la omisión o invalidez de atributos entityName pueden dar como resultado actualizaciones de índice no válidas a lo largo del tiempo.

5.3.8 untokenized

Esta propiedad indica si un campo se debe simbolizar y si debe pasar a través del analizador o no. Se trata de un valor booleano. Si se establece en true, no se realizará ninguna simbolización y el análisis no se realizará en este campo antes de la indexación o mientras se realiza la búsqueda.

5.3.9 analyzerName

Esta propiedad especifica el analizador que se debe utilizar al simbolizar este campo. El contenido de este campo se puede establecer en LUCENESTANDARD, STANDARD, SIMPLE, STOP, WHITESPACE, KEYBOARD. (consulte el analizador en el apartado 5.2, "Tabla SearchService", en la página 13) Si este campo no está establecido, el analizador predeterminado utilizado será el que se toma del campo del analizador del SearchService asociado.

Capítulo 6. Iniciación a la API Servidor de búsqueda genérico

6.1 Introducción

Este capítulo no pretende ser una descripción exhaustiva de toda la API del Servidor de búsqueda genérico; hay disponible un conjunto completo de Javadoc como parte de la instalación. La finalidad de este capítulo es ofrecer una breve introducción a las clases y operaciones más importantes de la API para permitir que las búsquedas en el Servidor de búsqueda genérico se desarrollen rápidamente.

6.2 Correlacionadores

Los correlacionadores son clases que definen cómo se correlacionan los datos del servicio de búsqueda de las tablas de la base de datos de aplicaciones a las tablas de la base de datos de transferencia. Cada servicio de búsqueda tiene su propio correlacionador; el correlacionador que se debe utilizar se especifica en la tabla de base de datos SearchService. Para ver más detalles, consulte 5.2.5, “mapperName”, en la página 14.

Esta funcionalidad del correlacionador se utiliza en dos procesos:

1. Cuando se ejecuta el extractor de la base de datos, cada campo del servicio de búsqueda se repite sobre un servicio de búsqueda determinado. Para cada campo, los datos del atributo de entidad correspondientes se recuperan de la base de datos de la aplicación y se llenan en la tabla de la base de datos de transferencia SearchServiceRow
2. Cuando se llama a una operación de creación, actualización o eliminación para una entidad que se utiliza en un servicio de búsqueda, las filas SearchServiceRow relevantes se actualizan con las modificaciones de la entidad relacionada

En ambos procesos, el correlacionador relevante se invoca para cada servicio de búsqueda para correlacionar datos de las tablas de base de datos de la aplicación con las tablas de la base de datos de transferencia.

Al inicializarse el Servidor de búsqueda genérico, la información de la base de datos de transferencia se lee y se utiliza para construir los índices a partir de los metadatos del servicio de búsqueda. El Servidor de búsqueda comprobará periódicamente la base de datos de transición para ver si hay actualizaciones y mantener los datos de servicio actualizados.

Los siguientes métodos de la API de correlacionador deben ser implementados por los desarrolladores de búsquedas para cada servicio de búsqueda:

```
SearchServiceRowDtlsList mapToStagingDb(
  final SearchServiceKey id) throws AppException,
    InformationalException;

List getObjectList(final SearchServiceKey serviceId,
  final Object obj) throws AppException, InformationalException;

String getExtKey(final SearchServiceKey serviceId, List objList);

void remove(final SearchServiceKey serviceId, final Object objKey)
    throws AppException, InformationalException;

Object getFieldValue(final SearchServiceKey serviceId,
    final List objList, final SearchServiceFieldDtls field);
```

Consulte 7.4, “Implementación de operaciones del correlacionador”, en la página 24 si desea obtener información detallada

6.3 Controlador de búsqueda

El Controlador de búsqueda es un objeto singleton que se puede utilizar en la aplicación. Se encarga de mantener un seguimiento de las entidades a las que se hace referencia y en qué Servicios de búsqueda se hace referencia a ellas. Además, proporciona una API para sincronizar los cambios realizados en datos de aplicación con los índices relevantes sobre el Servidor de búsqueda genérico. Tenga en cuenta que desde una perspectiva cliente-servidor, el Controlador de búsqueda se encuentra en el "Cliente" (en este caso, el servidor de aplicaciones de Cúram) y no en el "Servidor" (en este caso, el Servidor de búsqueda genérico).

La API SearchController consta de tres métodos que pueden invocarse si cualquier entidad implicada en llenar un índice se ha modificado. El desarrollador de la búsqueda debe ser consciente de qué operaciones de entidad de aplicación tendrán como resultado dichas modificaciones e invocar los métodos adecuados en el SearchController. Los métodos expuestos en esta API son:

```
void SearchController.insert(final Object objectDtls,
  String entityName);
void SearchController.modify(final Object objectDtls,
  String entityName)
void SearchController.remove(final Object objKey, final String entityName);
```

Consulte 7.6, “Adición de sincronización a cada entidad de búsqueda”, en la página 27 si desea obtener información detallada

6.4 Conector del servicio de búsqueda

El SearchServiceConnector es una clase de programa de utilidad que permite realizar búsquedas. La operación de búsqueda en esta clase es la única forma soportada para que los desarrolladores de búsqueda invoquen una búsqueda en un índice del Servidor de búsqueda genérico.

Entre bastidores, esta clase maneja los detalles de conexión de la aplicación en ejecución a una instancia del Servidor de búsqueda genérico, independientemente de dónde se despliegue.

Se pueden realizar búsquedas con el SearchServiceConnector utilizando el método:

```
static SearchServerResults search(CuramQuery query)
```

Nota: Si el índice de búsqueda no contiene datos generará una IndexEmptyException. Los desarrolladores que implementan búsquedas deben manejar esta excepción correctamente.

Se necesitan credenciales de usuario para conectarse al Servidor de búsqueda genérico. El conector recopila los detalles del usuario actual y los utiliza para comunicarse con el Servidor de búsqueda genérico.

Nota: No intente utilizar el método DoSearch (o cualquier método del Servidor de búsqueda genérico) directamente. No funcionará porque se está ejecutando en el contexto de la aplicación Cúram y no en el contexto de una aplicación del Servidor de búsqueda genérico en ejecución

6.5 Consultas

Para realizar una búsqueda, se debe construir un objeto CuramQuery. La clase CuramQuery consta de:

- El searchServiceId del SearchService en cuyo índice desea buscar. Consulte el apartado 2.4, “Servicio de búsqueda”, en la página 4 para obtener más información sobre el concepto de servicios de búsqueda y 5.2.1, “searchServiceId”, en la página 13 para obtener información detallada acerca de cómo se define el searchServiceId
- Una lista de objetos CuramTerm o un atributo de texto que representa una serie de consulta de Lucene; estos representan los criterios de búsqueda. A continuación encontrará más información sobre los términos Cúram y el atributo de texto

- Una lista de objetos CuramField; se devolverán valores para estos campos como parte de los resultados de la búsqueda, pero sólo si los campos se han marcado como "Stored" en la definición de SearchServiceField (consulte el apartado 5.3.6, “stored”, en la página 15)
- Un atributo entero maxHits que indica el número máximo de coincidencias que se devolverán para esta consulta.
- Un distintivo booleano maxHitsUnbounded que indica que el número máximo de coincidencias no está limitado. Si este distintivo se establece el valor del atributo maxHits se ignora.

6.6 CuramTerm

Los CuramTerms forman parte de la estructura de CuramQuery que representa los criterios de búsqueda.

Existen tres tipos de términos: StandardTerm, DateTerm o un término DateRange. El objeto CuramTerm contiene uno de cada uno de estos tipos y tiene un atributo termType que especifica cuáles de los subtipos de término deben utilizarse. Solo de uno de los subtipos de término agregados es válido para cada objeto CuramTerm.

Para todos los tipos de término, el atributo "field" especifica el nombre del campo en el servicio de búsqueda en el que se debe buscar (consulte el apartado 2.5, “Campo”, en la página 4 y 5.3.3, “name”, en la página 15). El atributo "value" es el criterio de búsqueda que se debe utilizar, y su significado varía para los distintos tipos de términos y se describe a continuación.

6.6.1 Estructura de consulta

Cada término tiene un campo denominado *occurs*. El modo de establecerlo determina la estructura de la consulta: si todos los términos de búsqueda deben existir, si debe existir solo uno o cualquier otra combinación. Los valores posibles para *occurs* son `MUST`, `SHOULD`, `MUST_NOT` y `MUST_FIELD`.

Si se especifica `MUST` para el atributo *occurs* para un conjunto de términos entonces se devuelve un resultado sólo si se cumplen todas las condiciones. Si se especifica `SHOULD` para un conjunto de términos entonces se devolverá un resultado si se encuentran uno o varios de los términos. No obstante, si se combinan en una sola consulta se producirá un resultado indefinido y debe evitarse. Si es necesario construir consultas complejas con subconsultas AND y OR, entonces debe utilizar el atributo de consulta *text* descrito en el apartado 6.6.4, “Texto”, en la página 20.

Si se especifica `MUST_NOT` para el atributo *occurs*, entonces solo se devolverán los documentos que no coincidan con el término. Los términos que especifican este pueden combinarse con términos que especifiquen otros valores para el atributo *occurs*.

El uso de la opción `MUST_FIELD` le permite construir una subconsulta que pruebe un campo de índice determinado para un valor de un conjunto de valores, es decir, una subconsulta OR dentro de la consulta principal. Debe establecerlo como valor de *occurs* para todos los términos relacionados con ese campo y añadir un término para cada valor aceptable. Los términos que utilizan `MUST_FIELD` pueden formar parte de una consulta general que utilice las opciones de términos `MUST` o `SHOULD` término de opciones.

6.6.2 Términos estándar

Un término estándar se utiliza para todas las búsquedas que no implican fechas, por lo que este es el tipo de término que utilizará con más frecuencia.

La forma más básica de utilizar un término estándar consiste simplemente en especificar el nombre del campo y un solo símbolo como valor. El servidor de búsqueda devolverá resultados donde el valor del campo coincide con el término de búsqueda exactamente.

Otra forma de utilizar un término estándar consiste en especificar un valor que contenga varios símbolos como, por ejemplo, en la dirección. Una vez más, el servidor de búsqueda devolverá resultados donde el valor del campo coincide con el término de búsqueda exactamente.

Si el término de búsqueda especificado es un solo símbolo que contiene un carácter comodín, el servidor de búsqueda devolverá todos los resultados coincidentes. Los caracteres comodín soportados son "*", que coincide con cualquier serie de caracteres, y "?", que coincide con un solo carácter. Ejemplo: term = "Dub*"

Una StandardTerm se puede tratar como un prefijo de búsqueda. Esto significa que estamos buscando resultados de búsqueda que contengan los criterios de búsqueda al principio. Debe especificar un prefijo de búsqueda estableciendo el atributo isPrefixSearch del StandardTerm. Tiene el mismo efecto que especificar un comodín multicarácter "*" al final del valor de búsqueda. Un término de búsqueda de prefijo no puede contener ningún otro comodín.

Ejemplo 1: Para un término de prefijo simbolizado estándar "abc", la búsqueda subyacente es term = "abc*", para realizar búsquedas de varios términos simbolizados y de prefijo, por ejemplo, un término de búsqueda con prefijo "abc def", la búsqueda subyacente es term = "abc* def*"

Ejemplo 2: Para un término sin prefijo simbolizado con abc, se debe especificar el valor del término = "abc*. Para búsquedas de varios términos sin prefijo simbolizados que empiecen por "abc" y "def", debe especificarse el valor "abc* def*".

6.6.3 Términos de fecha y de rango de fechas

Un término de fecha es similar a un término estándar, salvo que se utiliza para buscar en campos de tipo Date o DateTime.

Un término de rango de fechas se puede utilizar para buscar valores que queden entre una fecha mínima (beginDate) y una fecha máxima (endDate). El atributo booleano "isExclusive" determina si las fechas de inicio y finalización se incluyen en los criterios de búsqueda. Si "isExclusive" se establece en true, la búsqueda se realiza sin incluir las fechas de inicio y finalización. Si "isExclusive" se establece en false, la búsqueda se realiza incluyendo las fechas de inicio y finalización.

Nota: Cuando una consulta contiene más de un término, los resultados devueltos son los que coinciden con todos los términos de búsqueda; actualmente no existe ningún concepto de OR o NOT en la API del Servidor de búsqueda genérico

Nota: Tenga en cuenta al utilizar fechas para buscar que es responsabilidad suya asegurarse de que la fecha del término de búsqueda hace referencia al mismo huso horario que se ha utilizado al exportar los datos al servicio de búsqueda

6.6.4 Texto

El atributo de texto de la clase CuramQuery es una alternativa a un conjunto de términos y permite más flexibilidad al especificar los criterios de búsqueda. Se debe utilizar solo si es necesario, puesto que también es más fácil que introduzca errores en las búsquedas con este enfoque. El formato para especificar criterios de búsqueda utilizando este atributo se describe en la documentación de Lucene. Está disponible en http://lucene.apache.org/java/2_2_0/queryparsersyntax.html.

No puede combinar los términos y el uso de la serie de consulta de texto. Si la serie de consulta de texto está presente, cualquier CuramTerms presente en la consulta se ignorará.

6.7 Generación de consultas

La API Servidor de búsqueda genérico contiene una clase de programa de utilidad diseñada para permitir construir objetos CuramQuery fácilmente. Esta clase es: `curam.core.impl.util.QueryBuilder`.

6.7.1 Construcción de un creador de consultas

`QueryBuilder` no es una clase estática, debe construir una nueva instancia de `QueryBuilder` para cada consulta que produzca.

Utilice los métodos `setUnbounded(boolean unbounded)` y `setMaxHits(long maxHits)` para especificar el número de coincidencias que debe devolver la consulta generada.

6.7.2 Adición de criterios de búsqueda

`QueryBuilder` proporciona una selección de métodos con el formato `addXXTerm(...parámetros...)` para añadir fácilmente distintos tipos de términos de búsqueda a su consulta generada. Estos términos se juntan mediante un operador AND para formar una consulta compleja. Estos métodos no se describirán por completo aquí, pero encontrará los detalles completos en el javadoc de GSS.

6.7.3 Generación de consultas a partir de una estructura

Si tiene una estructura Cúram que desee utilizar para generar una consulta, puede hacerlo utilizando este método: `setTerms(clave de objeto final)`.

Espera una estructura donde cada atributo *XX* tiene un atributo booleano correspondiente llamado *searchByXX* que especifica si ese atributo debe utilizarse para buscar. Se asumirá que cada atributo *XX* corresponde a un SearchServiceField de su SearchService.

Si los nombres de los atributos de su estructura no se corresponden con los nombres de los campos que ha definido para el servicio de búsqueda (consulte el apartado 2.5, "Campo", en la página 4 y 5.3.3, "name", en la página 15), puede definir una correlación entre ellos utilizando una correlación hash de diccionario. La correlación se realiza de los nombres de atributo de la estructura a los nombres de SearchServiceField. Solo tiene que añadir los pares de series a la correlación hash, con el nombre del atributo de estructura como clave y el nombre del campo como valor. El diccionario se puede especificar en el constructor al crear el objeto QueryBuilder o más adelante utilizando el método `setDictionary(HashMap<String, String>)`.

6.7.4 Especificación de los campos del servicio de búsqueda que se deben devolver

En la consulta, puede especificar qué subconjunto de los campos del servicio de búsqueda desea que se devuelvan como resultados. A menudo querrá que se devuelvan todos, por lo que puede utilizar los siguientes métodos de conveniencia:

- `includeAllFieldsInService()`
- `excludeField(String fieldName)`
- `excludeFields(String[] fieldNames)`

6.7.5 Obtención del objeto de consulta

Utilice el método `getQuery()` para obtener el objeto CuramQuery generado.

6.8 Cómo trabajar con resultados de búsqueda

De forma similar al requisito de convertir estructuras clave de Cúram en objetos `CuramQuery`, los `CuramDocuments` devueltos por las búsquedas también deben convertirse en estructuras de Cúram para ser utilizados en la aplicación.

El método de búsqueda `SearchServiceConnector` devuelve resultados en forma de objeto `SearchServerResults`. Consta de una lista de `CuramDocuments` y cada `CuramDocument` consta de una lista de

CuramFields. Se proporciona una clase de programa de utilidad denominada curam.core.impl.util.CuramDocToResultStruct para convertir entre CuramDocuments y estructuras de Cúram.

```
static java.lang.Object convert(CuramDocument document,
     java.lang.Object structObj,
     java.util.HashMap dictionary)
```

Este método toma un CuramDocument y una instancia de estructura (por medio del parámetro structObj). Para cada campo del CuramDocument, el método intenta encontrar un atributo en la estructura con el mismo nombre y tipo de datos. Se devuelve una estructura que contiene todos los valores correlacionados y se debe convertir en una estructura del tipo correcto.

Si los nombres de los atributos de su estructura no se corresponden con los nombres de los campos que ha definido para el servicio de búsqueda (consulte el apartado 2.5, “Campo”, en la página 4 y 5.3.3, “name”, en la página 15), puede definir una correlación entre ellos utilizando el parámetro dictionary. La correlación es de los nombres de campo del servicio de búsqueda con los nombres de atributos de la estructura; solo tiene que añadir los pares de series a la correlación hash, con el nombre del campo como clave y el nombre del campo atributo de estructura como valor. La función de conversión hará coincidir después los nombres de los campos con los nombres de los atributo utilizando esta correlación hash

Nota: Tenga en cuenta que los atributos de la estructura de resultados cuyos nombres corresponden a los campos del documento deben tener tipos de Cúram simples y no deben ser estructuras agregadas.

6.9 Conversión de tipos de datos y series

El Servidor de búsqueda genérico contiene una API para convertir en series tipos de datos de Cúram en los que se pueden realizar búsquedas y viceversa. Es posible que sea necesario utilizarlos ocasionalmente en correlacionadores personalizados o, si se obtienen los resultados del análisis directamente en lugar de utilizar la clase de programa de proporcionada curam.core.impl.util.CuramDocToResultStruct.

La clase de convertidor es curam.core.impl.search.datatypes.DataTypeConverter. Esta clase contiene métodos para convertir tipos de datos de Cúram en series y volver a convertir series en tipos de datos de Cúram (pasando estructura y especificando qué atributo de la estructura se debe establecer).

Capítulo 7. Implementación de una búsqueda con el Servidor de búsqueda genérico

7.1 Visión general

En este capítulo se proporciona un ejemplo de la implementación de una búsqueda basada en un Servidor de búsqueda genérico dentro de la aplicación Cúram. El ejemplo desarrollado aquí es una búsqueda de persona.

Los pasos de implementación son los siguientes:

- Escriba los archivos dmx SearchService y SearchServiceField
- Implemente la interfaz Mapper
- Implementar el direccionamiento de búsqueda y la funcionalidad de invocación
- Añada la sincronización de las operaciones de la aplicación a las entidades de búsqueda (o utilice el enfoque del Correlacionador de extracción, consulte el apartado Capítulo 8, “Correlacionador de extracción”, en la página 29
- Cree una interfaz de usuario y una fachada para la búsqueda; este es el desarrollo normal de la aplicación.

7.2 Ejemplo de búsqueda de personas: Visión general

Es importante tener en cuenta que los usuarios del Servidor de búsqueda genérico de Cúram no deben notar ninguna diferencia funcional entre sus búsquedas y las búsquedas de servidor implementadas utilizando SQL; además, las pantallas y la experiencia general del usuario pueden seguir siendo iguales. Como tal, el ejemplo siguiente presupone que los lectores desarrollarán esta funcionalidad de la aplicación (junto con las clases Facade apropiadas, etc.) del modo normal.

En nuestro ejemplo de búsqueda de personas, los usuarios navegarán a la página de UIM relevante para realizar una búsqueda de persona. En esta página, rellenarán uno o varios criterios de búsqueda. Cuando pulsen el botón "Buscar", se realizará la búsqueda. Los resultados constarán de una lista de registros que coinciden con los criterios de búsqueda.

En las búsquedas de aplicaciones, es común que los criterios de búsqueda y los detalles devueltos en la lista de resultados se clasifiquen a partir de varias entidades relacionadas. Para la búsqueda de personas, se utilizan las entidades siguientes y sus atributos como criterios de búsqueda o se devuelven como campos de resultados:

- Person - primaryAlternateID, personBirthName, motherBirthSurname, dateOfBirth, gender
- ConcernRole - sensitivity, concernRoleID
- AlternateName - firstForeName, surname
- AddressElement - city, address.

Cada una de estas entidades se relaciona mediante una asociación de clave foránea; concernRoleID es, por lo tanto, la clave externa del atributo SearchService para el servicio de búsqueda PersonSearch (consulte el apartado 5.2, “Tabla SearchService”, en la página 13)

Por lo tanto, los atributos siguientes se utilizarán en la búsqueda, ya sea como parte de los criterios de búsqueda o como parte visualizable de la lista de resultados:

- referenceNumber
- forename
- surname

- address
- city
- dateOfBirth
- sex
- birthSurname
- motherSurname

Como tales, serán los campos almacenados en la tabla SearchServiceField del servicio de búsqueda PersonSearch.

7.3 Desarrollo de archivos DMX de SearchService

7.3.1 Configuración del registro de SearchService

Consulte los apartados B.1, “Registro del servicio de búsqueda”, en la página 49 y 5.2, “Tabla SearchService”, en la página 13

7.3.2 Configuración del registro de SearchServiceField

Consulte los apartados B.2, “Registro de campo del servicio de búsqueda”, en la página 50 y 5.3, “Tabla SearchServiceField”, en la página 14

7.4 Implementación de operaciones del correlacionador

Consulte el apartado 2.12, “Correlacionador”, en la página 6 y 6.2, “Correlacionadores”, en la página 17 para obtener una introducción a los correlacionadores.

En los apartados siguientes se describe la implementación de los métodos de la interfaz Mapper para cada servicio de búsqueda. Se proporciona un ejemplo para el servicio de búsqueda PersonSearch para cada método de la interfaz. También hay disponible Javadoc completo para la interfaz Mapper que todos los desarrolladores que implementan un servicio de búsqueda deben leer.

7.4.1 Interfaz Mapper.mapToStagingDb

```
/**
 * Correlaciona información de la base de datos de la aplicación con la
 * base de datos de transferencia del servicio de búsqueda para el ID del
 * servicio de búsqueda especificado.
 *
 * @param id el identificador del servicio de búsqueda.
 * @return la lista de todas las filas correlacionadas con
 *   el servicio de búsqueda especificado.
 * @throws AppException excepción de aplicación
 * @throws InformationalException excepción de información.
 */
SearchServiceRowDtlsList mapToStagingDb(
  final SearchServiceKey id) throws AppException,
    InformationalException;
```

Este método se invoca durante el proceso por lotes de extracción de la base de datos; para cada servicio de búsqueda, mapToStagingDb se invoca para recuperar información de las entidades de origen y devolverla al proceso por lotes.

Debe grabarse una ReadmultiOperation de Cúram para procesar todos los registros que deben almacenarse en la base de datos de transferencia para cada servicio de búsqueda. Debe invocarse una operación del Servidor de búsqueda genérico denominada ExtractReadMultiOperation en cada uno de estos registros. Internamente, esta operación averigua qué otras entidades son necesarias para llenar una SearchServiceRow completa basándose en estos datos, y también construye un objeto SearchServiceRow.

El resultado de todo este proceso es simplemente una lista de SearchServiceRows, que constituyen todos los datos iniciales que se deben llenar en la base de datos de transferencia. El proceso por lotes de extracción de la base de datos se encarga entonces de insertar estas filas en la base de datos de transferencia.

7.4.2 Interfaz Mapper.getObjectList

```
/**
 * Llena la lista con todos los objetos de entidad para el servicio
 * de búsqueda dado cualquiera de los objetos de entidad utilizados.
 * @param searchServiceId. el identificador del servicio de búsqueda
 * @param obj. El objeto de entidad del que se recuperan todos los
 *   demás
 * @return la lista de todos los objetos de entidad para este servicio
 *   de búsqueda dado un parámetro de objeto especificado.
 */
List getObjectList(final SearchServiceKey serviceId,
  final Object obj) throws AppException,
    InformationalException;
```

Tal como se ha mencionado anteriormente, es posible que los datos de un servicio de búsqueda se recopilen de diversas entidades diferentes. También es posible que estas entidades estén relacionadas mediante complejas relaciones de clave foránea (por ejemplo, un registro de dirección podría estar relacionado con un registro de persona mediante un addressID que esté enlazado a través de un concernRoleAddressID, que está a su vez enlazado a través de un concernRoleID).

Las cosas se complican más cuando una de estas entidades se actualiza por medio de la aplicación. Cuando esto sucede, el Servidor de búsqueda genérico debe ser capaz de averiguar qué entidad ha sido afectada, en qué búsquedas está implicada y cómo está relacionada con todas las demás entidades incluidas en cada servicio de búsqueda.

Por último, uno o más documentos de uno o varios índices de búsqueda se tendrán que actualizar y la información de estos documentos se puede recopilar de diversas entidades, no solo de la que se acaba de modificar. No obstante, puesto que los servicios de búsqueda tienen un solo correlacionador, cada implementación del correlacionador solo tiene que preocuparse de ensamblar la información de su propio servicio de búsqueda.

El método de interfaz getObjectList aborda este problema. Dado un solo registro de entidad actualizado, getObjectList recopila todos los demás registros Dtls de entidad que se necesitarán para actualizar el documento correspondiente en el índice del servicio de búsqueda actual. El método getObjectList debe estar codificado de tal modo que ninguna de las entidades implicadas en un servicio de búsqueda se pueda utilizar como punto de partida de este proceso. getObjectList se encarga de:

- Averiguar qué entidad se le ha pasado
- Averiguar todas las entidades relacionadas para el servicio de búsqueda en cuestión
- Leer y ensamblar todos los registros de entidad relacionados basándose en los datos de la entidad de parámetro

Se llama al método mapper.getobjectList () en los siguientes procesos:

- Inserción de sincronización de base de datos
- Modificaciones de sincronización de base de datos
- Extracción de base de datos inicial

Tenga en cuenta que, para la extracción de base de datos inicial, el método de interfaz getObjectList se invoca para cada elemento captado de ReadmultiOperation; normalmente, esta será la entidad de nivel superior en este caso (por ejemplo, para una extracción de búsqueda de persona, todos los registros de persona se leerían en un readmulti; luego se llamará a getObjectList para que cada uno recupere toda la demás información necesaria para crear una SearchServiceRow).

Si se llama a este método para una entrada que no es relevante para este servicio de búsqueda, entonces la aplicación debe devolver simplemente una lista vacía.

7.4.3 Interfaz Mapper.getExtKey

```
/**
 * Obtiene el valor externo de la fila para la lista de objetos especificada.
 * @param searchServiceId. el identificador del servicio de búsqueda
 * @param objList la lista de objetos de entidad relacionados con el
 *   servicio de búsqueda.
 * @devolver la externalKey.
 */
String getExtKey(final SearchServiceKey serviceId, List objList) ;
```

El método de interfaz `getExtKey` devuelve un identificador exclusivo para el servicio de búsqueda especificado. Esta clave se utiliza como clave para cada fila de la tabla `SearchServiceRow` en la base de datos de transferencia. Tenga en cuenta que el parámetro objList es la salida del método de interfaz getObjectList descrito anteriormente. Por ejemplo, `llamar a getExtKey` para el servicio de búsqueda PersonSearch debe devolver el concernRoleID del registro en cuestión.

Si se llama a este método para datos que no son relevantes para el servicio de búsqueda, debe devolver un valor nulo.

7.4.4 Interface Mapper.remove

```
/**
 * Suprime la fila identificada por la clave especificada de la
 *   base de datos de
 * transferencia.
 * @param serviceId identificador del servicio
 * @param objKey la clave.
 * @throws AppException
 * @throws InformationalException
*/
void remove(SearchServiceKey serviceId, Object objKey)
  throws AppException, InformationalException;
```

Suprime el objeto de fila especificado de la base de datos de transferencia.

7.4.5 Interfaz Mapper.getFieldValue

```
/**
 * Si un valor de campo especializado no puede ser cubierto por la
 * funcionalidad <code>SearchServiceMapper.getValue()
 * <code> este método
 * se deberá anular en el correlacionador para el servicio de búsqueda
 *   específico.
 * @param objList lista de objetos de entidad para este ID de servicio
 *   de correlacionadores.
 * @param field el campo cuyo valor es necesario.
 */
Object getFieldValue(final SearchServiceKey serviceId,
   final List objList, final SearchServiceFieldDtls fieldDtls);
```

La infraestructura del Servidor de búsqueda genérico intentará recuperar un valor de atributo de entidad a partir de una lista de objetos utilizando metadatos de campo recuperados de la tabla de campos del de búsqueda. Normalmente, las objectLists contendrán estructuras dtls de entidad y en esos casos resulta trivial para el Servidor de búsqueda genérico utilizar el reflejo para identificar el atributo correcto y obtener su valor; esto es exactamente lo que se realiza entre bastidores.

No obstante, si la objectList contiene algo distinto de una estructura dtls de entidad (como en el caso de Person Search, donde hay un AddressElementDtlsList presente, que contiene una sola estructura AddressElement) y, a continuación, el método de interfaz Mapper.getFieldValue debe ser implementado por desarrolladores de búsqueda.

El método de interfaz Mapper.getFieldValue debe implementarse si un correlacionador no puede correlacionar automáticamente un valor de atributo específico. La entidad relevante correspondiente y el nombre de campo se pasan a través del parámetro de estructura fieldDtls, y el valor del atributo puede recuperarse del objList mediante un reflejo. Corresponde al desarrollador de búsquedas implementar esta interfaz del método para el tipo o tipos a los que se debe prestar servicio.

No se deben devolver series vacías de este método, siempre se deben devolver valores nulos.

7.4.6 Mapper newInstance()

Si el correlacionador está modelado, entonces debe especificarse la clase de fábrica para la propiedad mapperName de SearchService. Si el correlacionador NO está modelado, entonces la implementación del correlacionador debe implementar una interfaz

```
public static Mapper newInstance();
```

que devuelva una nueva instancia del correlacionador de este servicio de búsqueda. En este caso, la propiedad mapperName de SearchService será el nombre de clase de esta clase de implementación.

7.5 Direccionador de búsqueda e implementación

Tal como se ha mencionado anteriormente, la búsqueda utiliza actualmente SQL. En versiones futuras, es probable que las búsquedas de plataforma y solución comiencen a utilizar el Servidor de búsqueda genérico como método de búsqueda de su elección. Sin embargo, es probable que la búsqueda SQL también siga estando soportada como lo está actualmente, tanto desde una perspectiva de protección de actualización como desde una perspectiva de opción de recuperación / migración tras error u otros problemas de despliegue.

Para facilitarlo, debe implementarse una clase de fábrica de direccionador de búsqueda que debe devolver una referencia a la implementación de búsqueda de la base de datos o la implementación basada en el Servidor de búsqueda genérico basada en un valor de propiedad.

7.6 Adición de sincronización a cada entidad de búsqueda

Tal como se ha indicado anteriormente, la base de datos de transferencia del Servidor de búsqueda genérico debe actualizarse de forma oportuna cuando se realicen modificaciones en las entidades relacionadas con el servicio de búsqueda. Una sola entidad se puede utilizar en más de un servicio de búsqueda y cada uno de estos servicios de búsqueda debe reflejar los cambios en dicha entidad.

La clase *SearchController* se encarga de garantizar que toda la información de la base de datos de transferencia está actualizada. Los métodos *SearchController insert*, *modify* y *remove* se deben llamar desde la aplicación cuando se ejecuta la operación de la entidad del servicio de búsqueda correspondiente. Las operaciones *insert* y *modify* de SearchController modifican la información de la tabla SearchServiceRow con los datos de la estructura de detalles de la entidad especificada. La interfaz remove requiere una clave que identifica el objeto de entidad que se elimina y el nombre de la entidad.

```
/**
 * Inserción genérica de actualizaciones de entidades en la base de datos.
 *
 * @param details los detalles del objeto.
 * @param entityName el nombre de la entidad
 * @throws AppException excepción de aplicación que recuperar el
 *   encargado de admisiones
 * o durante una inserción del correlacionador.
```

```
 * @throws InformationalException excepción de información.
 */

public final void insert(final Object details,
  final String entityName)
    throws AppException, InformationalException
/**
 * Modificación genérica de actualizaciones de entidades en la base de datos.
 *
 * @param details los detalles del objeto.
 * @param entityName el nombre de la entidad
 * @throws AppException excepción de aplicación que recuperar el
 *   encargado de admisiones
 * o durante una modificación del correlacionador.
 * @throws InformationalException excepción de información.
 */
public final void modify(final Object details,
  final String entityName)
    throws AppException, InformationalException
/**
 * Eliminación genérica de entidad de la base de datos.
 *
 * @param key la clave de objeto.
 * @param entityName el nombre de la entidad
 * @throws AppException excepción de aplicación.
 * @throws InformationalException excepción de información.
 */
public final void remove(final Object key,
  final String entityName) throws AppException,
    InformationalException
```

Capítulo 8. Correlacionador de extracción

8.1 Introducción

En el capítulo anterior describimos el mecanismo de sucesos y cómo se puede utilizar para mantener sus datos sincronizados con su servicio de búsqueda. El Servidor de búsqueda genérico ahora proporciona otra manera de mantener su servicio de búsqueda actualizado, que se conoce como el Correlacionador de extracción. En este capítulo se describe cómo funciona el Correlacionador de extracción y cómo se puede utilizar con las nuevas búsquedas que desarrolle.

8.2 Visión general del Correlacionador de extracción

El mecanismo de sucesos es de lejos el método más eficiente para mantener los servicios de búsqueda actualizados. Sin embargo, si las búsquedas son complejas, desarrollar y probar totalmente el servicio de búsqueda puede resultar engorroso. Este es el problema que el Correlacionador de extracción está orientado a resolver.

El correlacionador de extracción utiliza indicaciones de fecha y hora en registros de aplicación para buscar los registros que se han creado o actualizado desde que el correlacionador de extracción o el extractor se ejecutó por última vez. Cuando encuentra estos registros, los proporciona al Controlador de búsqueda para actualizar los servicios de búsqueda, y a partir de aquí el proceso es exactamente el mismo que el mecanismo de sucesos estándar. Este proceso requiere que todas las tablas de bases de datos implicadas en un servicio de búsqueda se exploren, lo que obviamente exige recursos de la base de datos. En esencia el Correlacionador de extracción sacrifica rendimiento de tiempo de ejecución para proporcionar un modo más rápido y fácil de desarrollar búsquedas.

8.3 Desarrollo con el Correlacionador de extracción

En esta sección se le guiará a través del proceso de desarrollo de un servicio de búsqueda utilizando el Correlacionador de extracción.

8.3.1 Habilitación del campo Última actualización en las entidades en las que se pueden realizar búsqueda

Se necesitan indicaciones de fecha y hora en todas las entidades de la base de datos implicadas en los servicios de búsqueda y que utilizan un Correlacionador de extracción. Estas columnas de indicación de fecha y hora se añaden automáticamente y se mantienen actualizadas mediante la infraestructura cuando se habilita la característica del campo Última actualización para la entidad en el modelo. El proceso para habilitar esta característica se documenta en la publicación `Server Modelling Guide`.

8.3.2 Modelado de la exploración de tablas

Otro requisito de modelado impuestas por el Correlacionador de extracción para modelar una operación denominada `searchByLastwritten` (debe escribirlo exactamente así, respetando las mayúsculas y las minúsculas).

Esta operación debe ser nsmulti. El valor para en caso de que no se genere SQL debe ser no. La operación debe tomar una estructura denominada *key*. Debe modelar su propia estructura como un parámetro, pero debe tener un atributo denominado *datetime*, que debe ser un `DateTime`. Posteriormente, especificará el nombre de clase de esta estructura en la tabla GSSEntity, tal como se describe a continuación.

Debe proporcionar SQL para la operación. A continuación se muestra un ejemplo sencillo para una entidad simple denominada Customer:

```
Select Customer.customer_id, Customer.name,
      recordStatus from Customer
 WHERE Customer.lastwritten >= :datetime
 INTO :customer_id :name :recordStatus
```

Debe asegurarse de seleccionar todas las columnas utilizadas por el servicio de búsqueda.

Además del método de exploración de tabla, debe tener un método de lectura estándar en todas las entidades que se pueden buscar.

8.3.3 Definición del servicio de búsqueda

El servicio de búsqueda se debe definir del modo habitual (consulte el apartado Capítulo 7, “Implementación de una búsqueda con el Servidor de búsqueda genérico”, en la página 23

Además de las tablas SearchService y SearchServiceField, debe añadir definiciones a las tablas GSSMapperType y GSSEntity.

8.3.3.1 GSSMapperType

Esta tabla simplemente correlaciona el nombre del servicio de búsqueda con una serie que define el tipo de correlacionador. El valor predeterminado es el correlacionador de sucesos estándar, que no es necesario especificar. Para utilizar el correlacionador de extracción con un servicio de búsqueda determinado, debe añadirse una fila a esta tabla que correlacione el nombre del servicio de búsqueda con el tipo de correlacionador “PULL”.

searchServiceId

El identificador del servicio de búsqueda; una serie utilizada para identificar de forma exclusiva un servicio de búsqueda. Es una clave foránea de la tabla SearchService.

mapperType

Establezca este valor en "PULL" (debe estar en mayúsculas) para habilitar el Correlacionador de extracción para el servicio de búsqueda.

8.3.3.2 GSSEntity

Cuando se utiliza el correlacionador de extracción, GSS requiere más información sobre las entidades que se utilizan en los servicios de búsqueda. Para cada entidad exclusiva listada en los registros searchServiceField hijos pertenecientes a cada SearchService que utiliza el Correlacionador de extracción, se debe añadir un registro GSSEntity (no obstante, si varios campos pertenecen a la misma entidad, no es necesario que repita la información).

searchServiceId

El identificador del servicio de búsqueda; una serie utilizada para identificar de forma exclusiva un servicio de búsqueda. Es una clave foránea de la tabla SearchService.

tblScanKeyStruct

Es el nombre de clase completo de la estructura que es el parámetro para el método `searchByLastwritten` modelado descrito aquí: 8.3.2, “Modelado de la exploración de tablas”, en la página 29.

entityKeyStruct

Es el nombre de clase completo de la estructura de parámetro para el método de lectura de la entidad.

EntityFactClass

Es el nombre de clase completo de la clase de fábrica generada para la entidad.

8.3.4 Escritura de la clase de correlacionador

Una implementación de SearchServiceMapper con el correlacionador de búsqueda es muy similar a una implementación de SearchServiceMapper estándar, tal como se describe en el capítulo Implementación de una búsqueda con GSS de esta guía. Sin embargo, existen algunas consideraciones adicionales.

Cuando se utiliza el Correlacionador de extracción con un servicio de búsqueda complejo que conste de varias entidades relacionadas, asegúrese de que la implementación de SearchServiceMapper se comporte adecuadamente cuando tenga que hacer frente a conjuntos de entidades incompletos, es decir, si las entidades A, B y C juntas forman un servicio de búsqueda se puede llamar al correlacionador cuando solo existen A y C. Según el servicio de búsqueda, el comportamiento correcto puede consistir en añadir el conjunto incompleto de datos al servicio de búsqueda, o en no hacer nada hasta que el conjunto se haya completado.

8.4 Operaciones de supresión

El Correlacionador de extracción no puede realizar operaciones de supresión estándar. Si tiene una entidad en la que se puedan realizar búsquedas que pueda suprimirse, deberá utilizar otro mecanismo para realizar esta operación (por ejemplo, el mecanismo basado en sucesos descrito en esta guía).

No obstante, el Correlacionador de extracción puede realizar operaciones de supresión lógica estándar, es decir, cuando se establece una columna recordStatus utilizando los valores de la tabla de códigos `RecordStatus` .

Capítulo 9. Búsquedas y consultas en detalle

9.1 Introducción

Al igual que cualquier otro software, sus búsquedas habilitadas para GSS deben ajustarse a determinadas restricciones de diseño para ofrecer un rendimiento aceptable y funcionar tal como los usuarios esperan. En este capítulo se describe de forma detallada el proceso de diseño de una búsqueda GSS y el uso adecuado de consultas GSS.

9.2 El servicio de búsqueda: directrices generales

La primera tarea de diseño consiste en decidir qué datos desea poder buscar. ¿En qué campos desea poder realizar búsquedas? ¿Qué datos desea que devuelva la búsqueda? Existen varios inconvenientes, por lo que vale la pena pensar en estas cuestiones detenidamente.

En primer lugar, el índice debe contener tan pocos campos como sea posible. Menos campos implican un índice menor en el momento de la ejecución y menos uso de los recursos del sistema. No los ponga en su servicio de búsqueda a menos que lo necesite.

Cada campo del índice se puede indexar (es decir, se pueden realizar búsquedas en él), almacenar (es decir, se puede recuperar su valor), o las dos cosas. Las razones por las que puede querer indexar un campo son evidentes: poder realizar búsquedas basadas en el campo en cuestión. Sin embargo, es posible que no desee realizar búsquedas en algunos campos, como por ejemplo ID no legibles. También es posible que desee añadirlos a su servicio de búsqueda como campos almacenados pero no indexados, de modo que pueda realizar búsquedas de base de datos basadas en los resultados de las búsquedas. Si no es necesario indexar un campo no lo haga; los procesos de extracción se ejecutarán más rápidamente y el índice consumirá menos recursos del sistema.

Del mismo modo, puede optar por almacenar los valores de los campos o no. En general, el índice no almacena el valor original de un campo, sino que mantiene solo una representación en la que se pueden realizar búsquedas. En general, para que sea útil, una búsqueda debe almacenar al menos un campo (la clave primaria correspondiente del registro de base de datos).

Después de esto, almacenar campos o no tiene sus ventajas y sus inconvenientes. Puede almacenar todos los campos que necesita para mostrar los resultados de la búsqueda o puede almacenar sólo los ID de base de datos y utilizarlos para recuperar los datos de la base de datos para visualizarlos. La primera opción tendrá como resultado un índice mucho mayor, pero una visualización más rápida de los resultados de la búsqueda porque la base de datos no es necesaria.

9.3 Correlación de la estructura de la base de datos con un índice: desnormalización

Puede que desee incluir datos de varias entidades diferentes en la búsqueda. A diferencia de las búsquedas de base de datos, las búsquedas con índices no se realizan utilizando uniones. Recuerde que la ventaja principal de utilizar un índice es permitir que la labor de búsqueda se realice esencialmente desde el primer momento, cuando el índice se crea en lugar de cuando se invoca la búsqueda. Por lo tanto, todas las tablas de bases de datos se deben desnormalizar para la indexación. La alternativa, que consiste en crear índices separados, buscar en ellos por separado y luego intentar fusionar los resultados, es mucho más compleja e ineficiente.

Por ejemplo, digamos que tiene las siguientes entidades: la entidad Persona con atributos de nombre, fecha de nacimiento y una clave foránea que apunta a una entidad Dirección con los atributos de

dirección, ciudad y país. Desea crear una búsqueda que le permita buscar personas por su nombre, fecha de nacimiento, dirección, ciudad y país. Crearía un índice de búsqueda que contiene todos los datos de ambas tablas.

Cuando tenga varias entidades que contribuyan a un solo índice de búsqueda, tenga en cuenta que las actualizaciones de cualquiera de las tablas implicadas puede provocar que el índice de búsqueda requiera una actualización.

9.4 Campos simbolizados y no simbolizados

Ya hemos mencionado brevemente la cuestión de la simbolización de los campos de búsqueda. Lo que la simbolización implica es esencialmente la división de los datos indexados en unidades denominadas símbolos. Esto se lleva a cabo mediante el uso de un analizador. Los distintos analizadores se comportan de forma diferente, algunos pueden dividir los símbolos en los espacios en blanco, algunos en la puntuación, etc. Los símbolos resultantes también se transforman normalmente en minúsculas. Para los campos simbolizados, las series de consulta se simbolizan de la misma manera y, por lo tanto, las búsquedas no son sensibles a mayúsculas y minúsculas, entre otras ventajas.

En el caso de algunos campos no tiene sentido simbolizarlos. Buenos ejemplos de esto son los valores generados por el sistema, como códigos de tabla de códigos. En general, sin embargo, la mayoría de los campos se deben simbolizar. En concreto, el comportamiento de los campos sin simbolizar de varias palabras y las búsquedas es contrario a lo que se podría intuir. Si ve que las búsquedas no devuelven los datos que espera, considere si este puede ser el caso.

Ejemplo: Tomamos como ejemplo un campo de dirección, con un documento que contiene "Joyce Way Parkwest Dublín". Si fuese un campo simbolizado utilizando el analizador estándar, el índice contendría cuatro términos: joyce, way, parkwest y dublín. Cualquier serie de consulta que contenga términos que coincidan con estos términos (exactamente o por medio de un comodín) encontrará este documento. Por ejemplo: "Dublín", "Joyce Way", "park*", etc.

No obstante, si este campo no está simbolizado y se añada el mismo documento, el índice contendrá un solo término: "Joyce Way Parkwest Dublín". Muchas menos series de consulta coincidirán con esto, esencialmente solo la misma serie o la primera parte de la serie como una búsqueda de prefijo. La búsqueda también distinguirá entre mayúsculas y minúsculas.

9.5 Comodines

GSS da soporte a comodines de un solo carácter y de varios caracteres. El símbolo de interrogación, “?” coincide con cualquier carácter individual. El símbolo de asterisco, “*” coincide con cualquier secuencia de caracteres. Ninguno de estos puede utilizarse como primer carácter en un término de búsqueda porque el resultado es un rendimiento deficiente. Al implementar una búsqueda, los desarrolladores deben tener en cuenta si se debe permitir a los usuarios especificar estos caracteres en las búsquedas y, de ser así, proporcionar ayuda en línea útil. De lo contrario, se pueden escapar con un carácter de escape: “\”. También puede ser útil comprobar que estos caracteres no se producen al principio de los términos de búsqueda y devolver al usuario un mensaje de error más específico del que la infraestructura de GSS puede proporcionar (se devolverá una excepción genérica para indicar que la consulta no es válida, pero el desarrollador que implementa la búsqueda podrá añadir más información sobre qué campo no es válido).

9.6 Los analizadores en profundidad

Tal como se ha explicado anteriormente, los analizadores preparan el texto en el que se pueden hacer búsquedas para la indexación y la búsqueda.

La elección de los analizadores es muy importante. Los analizadores son clases concretas que amplían la clase org.apache.lucene.analysis.Analyzer. El GSS incluye varios analizadores, pero se pueden crear y

utilizar analizadores propios. A veces, cuando esté tentado de definir un campo como no simbolizado, es recomendable que en lugar de hacerlo considera más detenidamente la elección de su analizador.

Cada servicio de búsqueda tiene un analizador predeterminado y cualquier campo del servicio de búsqueda puede alterar temporalmente dicho analizador para definir un analizador específico para utilizarlo con este campo (consulte el apartado 5.3.9, “analyzerName”, en la página 16). GSS utilizará el mismo analizador tanto para la indexación como para la búsqueda.

El Servidor de búsqueda genérico proporciona los siguientes analizadores predefinidos.

LUCENESTANDARD
Divide el texto en los caracteres de puntuación, eliminando la puntuación. No obstante, un punto que no vaya seguido de un espacio en blanco se considera parte de un símbolo. Divide las palabras en los guiones, a menos que haya un número en el símbolo, en cuyo caso el símbolo completo se interpreta como un número de producto y no se divide. Reconoce las direcciones de correo electrónico y los nombres de host Internet como un símbolo. Normaliza el texto del símbolo a minúsculas y elimina las palabras frecuentes comunes en inglés.

STANDARD
Es parecido al analizador LUCENESTANDARD, pero las palabras frecuentes comunes se eliminan de los términos simbolizados y si el contenido que se debe simbolizar es un número único no se alterará (lo que hace que resulte adecuado para procesar ID de infraestructura generados que pueden ser números negativos).

SIMPLE
Divide el texto en los caracteres que no son letras y normaliza el texto de símbolo en minúsculas.

STOP Divide el texto en los caracteres que no son letras, normaliza el texto del símbolo a minúsculas y elimina las palabras frecuentes comunes en inglés.

WHITESPACE
Divide el texto en los espacios en blanco. Las secuencias adyacentes de caracteres que no están en blanco forman símbolos.

KEYWORD
"Simboliza" la secuencia completa como un solo símbolo. Esto resulta útil para datos como códigos postales, ID y algunos nombres de producto.

Tenga en cuenta que si utiliza un analizador que no sea un analizador de GSS predefinido o analizadores suministrados con Lucene, la clase debe estar disponible en la ruta de clase del Servidor de búsqueda genérico.

Capítulo 10. Ejecución del Servidor de búsqueda genérico en Eclipse

10.1 Introducción

En este capítulo se describe cómo configurar el entorno de desarrollo para ejecutar el Servidor de búsqueda genérico en el IDE de Eclipse con fines de desarrollo y prueba.

El Servidor de búsqueda genérico puede ejecutarse en modalidad RMI con fines de desarrollo, de forma similar a la propia aplicación Cúram. En este capítulo se detalla cómo configurarlo.

10.2 Bootstrap.properties

Antes de iniciar el desarrollo, los valores relevantes deben añadirse al archivo `bootstrap.properties`, donde sea necesario. Consulte el apartado A.1, "Propiedades de configuración", en la página 47 para obtener una descripción de las propiedades de configuración.

10.3 Inicio del Servidor de búsqueda genérico de Cúram desde Eclipse

Al igual que la aplicación Cúram, en la modalidad de desarrollo el Servidor de búsqueda genérico requiere que se ejecute un proceso tnameserv en la máquina.

En la instalación de desarrollo, encontrará los siguientes archivos de clase Java de implementación en el archivo `/CuramSDEJ/lib/gss.jar` en Eclipse:

- `curam.core.impl.DataBaseSearchExtractor.class`
- `curam.core.impl.admin.StartSearchServer.class`

Debería poder ejecutar los archivos de clase anteriores como una aplicación Java normal en el contexto del proyecto EJBServer de la forma habitual, es decir:

- Pulse el botón derecho del ratón en el archivo y seleccione **Ejecutar como aplicación Java**

Ejecute DataBaseSearchExtractor para crear la base de datos de transferencia antes de StartSearchServer. Y ejecute el proceso StartSearchServer siempre que necesite ejecutar una instancia del servidor de búsqueda para probar la funcionalidad de búsqueda. Debe volver a ejecutar DataBaseSearchExtractor antes de iniciar el SearchServer si ha reconstruido la base de datos de la aplicación.

Nota: Si alguno de los servicios de búsqueda utilizar analizadores de terceros o personalizados (es decir, los analizadores que no se proporcionan como parte de la distribución de Lucene), asegúrese de que se añadan a la ruta de clase del proyecto `EJBServer`.

Capítulo 11. Despliegue del Servidor de búsqueda genérico

11.1 Introducción

En este capítulo se describe el proceso de despliegue del Servidor de búsqueda de Cúram en el servidor de aplicaciones. Este capítulo está orientado a los administradores que desplegarán el servidor de búsqueda junto con la aplicación Cúram y que están familiarizados con la Guía de despliegue de Cúram relevante.

11.2 Opciones de despliegue

Puede desplegar GSS en su propio archivo `ear` o como parte del archivo `ear` de Cúram. El archivo `EJBServer/project/config/deployment_packaging.xml` contiene una opción para incluir GSS, denominada `requireSearchServer`. Si la establece y crea su `ear` de Cúram, no es necesario que despliegue GSS como un archivo `ear` separado (de hecho, el servidor de aplicaciones no lo permitirá). En general, esto no se recomienda porque no es una configuración de despliegue de alto rendimiento, pero puede ser útil para realizar pruebas o despliegues pequeños.

11.3 Proceso de despliegue

El proceso de despliegue consta de los pasos siguientes:

- Configure el archivo Bootstrap.properties con sus propiedades de configuración y cualquier propiedad relacionada con el servidor de búsqueda. Consulte el apartado A.1, “Propiedades de configuración”, en la página 47 para obtener una descripción de las propiedades de configuración.
- Cree su archivo ear de la aplicación Cúram como de costumbre (también se creará el archivo ear de GSS).
- Configure la base de datos de la forma habitual.
- Ejecute el extractor de base de datos de búsqueda del Servidor de búsqueda genérico de Cúram.
- Despliegue todos los archivos ear de aplicación, incluido SearchServer.ear
- Inicie sesión en la aplicación como administrador y configure las propiedades del sistema para habilitar las búsquedas soportadas por GSS que desee utilizar y para habilitar el mecanismo de sincronización. Consulte el apartado Capítulo 4, “Búsquedas habilitadas para el Servidor de búsqueda genérico”, en la página 11
- Ejecute el proceso de inicio del servidor de búsqueda genérico.

El Servidor de búsqueda genérico debe estar disponible entonces para responder a las consultas.

11.4 Agrupación en clúster

El despliegue de múltiples instancias de GSS está soportado en un entorno de clúster. La explicación detallada de topologías de clúster avanzadas queda fuera del alcance de esta guía. Consulte también 12.9, “Configuración recomendada para el entorno de producción”, en la página 46.

Nota: Se recomienda desplegar GSS en su propio clúster.

11.5 Destinos de creación

Los siguientes destinos de creación son específicos del Servidor de búsqueda genérico de Cúram.

11.5.1 weblogicEARGSS

Este destino crea el archivo SearchServer.ear y lo copia en el directorio EJBServer/build/ear/WLS/, junto con el archivo ear de Cúram. Se ejecuta automáticamente como parte del destino **weblogicEAR**. El archivo ear SearchServer se debe crear después del archivo ear de Cúram. Después de que se haya creado el archivo ear SearchServer, la aplicación está preparada para el despliegue en Oracle WebLogic Application Server utilizando los mismos destinos de creación o procesos manuales que el archivo ear de Cúram.

11.5.2 websphereEARGSS

Este destino crea el archivo SearchServer.ear y lo copia en el directorio EJBServer/build/ear/WLS/, junto con el archivo ear de Cúram. Se ejecuta automáticamente como parte del destino **websphereEAR**. El archivo ear SearchServer se debe crear después del archivo ear de Cúram. Después de que se haya creado el archivo ear SearchServer, la aplicación está preparada para el despliegue en IBM®WebSphere Application Server utilizando los mismos destinos de creación o procesos manuales que el archivo ear de Cúram.

11.5.3 runExtractor

Este destino debe ejecutarse después de que la base de datos de la aplicación se haya configurado. Extrae, de forma predeterminada, todos los datos relacionados con los servicios de búsqueda de CEF y cualquier otro servicio de búsqueda que haya definido fuera de la base de datos de la aplicación y los transforma en un formato adecuado para la indexación. La cantidad de tiempo que este proceso durará se incrementará con la cantidad de datos que tengan que extraerse. Este destino se puede volver a ejecutar varias veces si es necesario.

Este destino puede ejecutarse en un único servicio de búsqueda especificando la propiedad “SERVICE”. Por ejemplo: “build runExtractor -DSERVICE=PersonSearch”

11.5.4 runPersist

Si está utilizando un índice de base de datos permanente (consulte el apartado 12.3, “Persistencia del índice”, en la página 43, este destino de creación crea el índice a partir de las tablas de la base de datos de transferencia. Solo debe ejecutarse después de que la base de datos de la aplicación se haya configurado y el destino runExtract se haya ejecutado. El destino runExtract creará el índice persistente si la persistencia se ha configurado, por lo tanto, este destino sólo debe ejecutarse por separado si ha cambiado la configuración desde la ejecución del destino runExtractor.

11.5.5 startupSearchServer

Este destino es opcional. Si se va a ejecutar, debe ejecutarse después de que el Servidor de búsqueda genérico se haya desplegado en el servidor de aplicaciones. Activa el servidor de búsqueda para configurar sus índices de manera que estén disponibles para la búsqueda. La cantidad de tiempo que este proceso durará se incrementará con la cantidad de datos que tengan que indexarse. Si no ejecuta el destino de arranque explícitamente, el servidor de búsqueda inicializará sus índices en la primera solicitud de búsqueda. Esta función existe principalmente para facilitar la realización de pruebas con pequeños conjuntos de datos. Para los conjuntos de datos grandes la función de arranque automático no se debe utilizar. Puede inhabilitar el arranque automático estableciendo la propiedad “curam.searchserver.autostartup.disabled” en true en el archivo `bootstrap.properties` cuando configure el archivo ear; esto es recomendable.

11.6 Rendimiento de la base de datos

La aplicación Cúram y la aplicación del servidor de búsqueda comparten una base de datos común, pero imponen requisitos bastante diferentes sobre ella. La tabla SearchServiceRow experimentará la mayor parte de grabaciones y accesos y crecerá mucho, porque básicamente incluye una versión de todos los datos que se pueden buscar. La aplicación Cúram también grabará en esta tabla cuando se inserten o se

actualicen entidades que se pueden buscar. Periódicamente, si el servidor de búsqueda se reinicia o cuando se sincroniza, habrá un montón de lecturas de esta tabla. Puede tener sentido colocar la tabla SearchServiceRow en un espacio de tabla diferente al resto de las tablas de la aplicación, en función de los recursos y las necesidades de su organización.

11.7 Consideraciones sobre la hora

Si se utilizan máquinas diferentes para ejecutar las instancias de la aplicación Curam y el Servidor de búsqueda genérico, todos los sistemas deben tener sus relojes sincronizados y permanecer en el mismo huso horario. Le recomendamos emplear una solución de software como NTP (en función de su plataforma de despliegue) para garantizar que este siga siendo el caso. Si esto no se realiza, no puede haber ninguna garantía de que el Servidor de búsqueda genérico refleje con precisión todas las actualizaciones de los datos de aplicación.

Capítulo 12. Rendimiento

12.1 Introducción

Este capítulo describe el rendimiento del Servidor de búsqueda de Cúram y cómo pueden influir sobre él los distintos escenarios de despliegue y valores de configuración.

12.2 Tipos de índices

Tal como se describe en el apartado 2.3, "Índices", en la página 3, un índice es la estructura de datos en la que se basan las búsquedas de GSS. Puede ser una estructura de datos bastante considerable (consulte el apartado 12.7.1, "Cálculo del tamaño de índice", en la página 46 y esto plantea la pregunta siguiente: ¿dónde se debe almacenar? GSS proporciona dos opciones: memoria o archivo. Para obtener información sobre cómo configurar estas propiedades, consulte A.1, "Propiedades de configuración", en la página 47

Los directorios en RAM (en memoria) se deben reconstruir cada vez que un servidor de aplicaciones se inicia (a menos que se utilice la persistencia, consulte el apartado 12.3, "Persistencia del índice". Son de rápido acceso, pero sus requisitos de memoria pueden superar los recursos disponibles. No obstante, los directorios en RAM pueden ser muy útiles para realizar pruebas, porque no conservan el estado.

Los índices de archivos utiliza el sistema de archivos local para almacenar el índice. Aunque la especificación J2EE no cubre el acceso al sistema de archivos, en la práctica esto funciona con todas las versiones soportadas documentadas en un documento por separado, el documento *Curam Supported Prerequisites*. Naturalmente cuanto mejor sea el rendimiento del sistema de archivos subyacente utilizado mejor será el rendimiento de GSS.

12.3 Persistencia del índice

Cada servicio de búsqueda tiene un índice asociado que se consulta durante cada búsqueda. Este índice se genera a partir de las tablas de la base de datos de transferencia cuando el servidor de búsqueda se inicializa. Es posible que se necesite una cantidad considerable de tiempo para leer todos los datos del servicio de búsqueda de las tablas de la base de datos de transferencia y posteriormente para generar los índices relevantes para estos datos.

El Servidor de búsqueda genérico proporciona los medios para hacer persistir el índice actual en la base de datos, con el fin de mejorar el tiempo de inicio. Cuando la persistencia del índice está habilitada, y antes de que se interroguen las tablas base, el índice persistente se carga si está disponible. Si no está disponible, todos los datos se leen de las tablas base y el inicio será más lento.

El índice persistente tiene una indicación de fecha y hora asociada que se almacena en la tabla del servicio de búsqueda apropiado correspondiente a ese índice. Esta indicación de fecha y hora indica la hora a la que se guardó el índice RAM de forma persistente en el disco. El hecho de conocer esta hora permite al Servidor de búsqueda genérico recuperar datos del servicio de búsqueda nuevos o modificados de las tablas base. El índice persistente y los datos nuevos/modificados de las tablas base proporcionan un índice en memoria completo listo para la búsqueda. Se ahorra tiempo mediante la reducción del acceso a las tablas base y el proceso asociado durante la construcción del índice.

Los datos del índice persistente se almacenan en formato BLOB; por lo tanto, el rendimiento de lectura y escritura de un índice de gran tamaño desde y hacia la base de datos es óptimo.

12.3.1 Invocación de operación de persistencia

La operación por lotes DataBaseIndexPersist.persistIndex() se ejecuta para realizar la copia de seguridad para todos los índices. El proceso para hacer que cada índice persista consiste en:

1. Leer el índice persistente actual
2. Leer datos nuevos o modificados de la tabla base
3. Generar un índice en memoria con los pasos 1) + 2) anteriores.
4. Guardar el índice en memoria recién generado en la base de datos.
5. Repetir los pasos del 1) al 4) para cada servicio de búsqueda.

12.4 Pruebas y consideraciones sobre las operaciones

Los índices persistentes, los índices FILE, están diseñados para conservar los índices creados tras cada reinicio del servidor.

Los datos también persisten entre las operaciones de reconstrucción de las bases de datos, y esto puede causar problemas para los probadores si los datos de índice se vuelven incoherentes con la base de datos actual.

De forma similar, en una situación operativa, si se producen actualizaciones de la base de datos sin que las actualizaciones del índice de búsqueda estén habilitadas en la aplicación (mediante la propiedad "curam.lucene.luceneOnlineSynchronizationEnabled") los datos del índice quedarán obsoletos y pueden producirse problemas.

En cualquiera de los casos anteriores, los datos persistentes se pueden eliminar manualmente de la base de datos descartando todas las tablas de bases de datos que empiezan con "GSS_" (habrá una tabla para cada servicio de búsqueda). Los índices persistentes se reconstruirán del modo normal cuando se ejecute una operación de extracción o persistencia.

En el caso de un índice FILE, el archivo puede suprimirse, y en el caso de un servicio de búsqueda de RAM que se encuentre con estos problemas, la reejecución del proceso de extracción solucionará los problemas.

12.5 Ajuste de rendimiento

En esta sección se describen los parámetros que influyen en el rendimiento de lectura y grabación del índice de búsqueda. Determinan cómo se construye el índice y cómo se deben grabar entradas nuevas en él.

12.5.1 Número máximo de documentos de fusión

`curam.searchserver.luceneadaptor.searcher.index.maxmergedocs`

Esta propiedad mejora los tiempos de búsqueda para valores superiores y para valores inferiores ofrece mejores resultados cuando un índice se encuentra con actualizaciones frecuentes. Los valores pequeños (por ejemplo, menos de 10.000) son mejores si el índice se actualiza frecuentemente, no obstante, el rendimiento de los tiempos de búsqueda se verá afectado. El valor predeterminado es 10000000. Si el rendimiento de la búsqueda es lo más importante, este valor debe ser grande, por ejemplo el valor predeterminado o, de lo contrario, si el rendimiento de actualización de los datos de búsqueda es más importante, el valor debe tener un valor pequeño, por ejemplo 10.000.

12.5.2 Factor de fusión

`curam.searchserver.luceneadaptor.searcher.pool.mergefactor`

Esta propiedad tiene un impacto sobre la RAM utilizada al actualizar un índice. El índice requiere la actualización como resultado de la búsqueda que afectan a las actualizaciones de datos de aplicación. Para los valores pequeños (inferiores a 10), las búsquedas serán más rápidas, sin embargo, las actualizaciones del índice de búsqueda serán más lentas. Con valores mayores (superiores a 10), se utiliza más RAM durante la actualización del índice y, aunque las búsquedas son más lentas, la actualización del índice es más rápida. El valor predeterminado es 10. Si el rendimiento de la búsqueda es lo más importante, este valor debe ser inferior a 10 o, de lo contrario, si el rendimiento de actualización de los datos de búsqueda es más importante, el valor deberá ser superior a 10.

12.5.3 Habilitar persistencia

```
curam.searchserver.server.index.persistence.enable
```

añada curam.searchserver.server.index.persistence.enable=true a Bootstrap.properties para habilitar la persistencia de índice.

Nota: Si esta propiedad está habilitada, durante la ejecución de la extracción de la base de datos también se generarán los nuevos índices persistentes.

12.5.4 Referencias

Para obtener más información sobre los parámetros que se describen en este apartado, consulte http://lucene.apache.org/java/2_2_0/api/index.html

12.6 Agrupación de búsquedas

En esta sección se describe cómo configurar agrupaciones de búsquedas y la influencia que tienen sobre el rendimiento de las búsquedas.

12.6.1 Visión general

Lucene tiene un mecanismo de almacenamiento en antememoria interno que realiza búsquedas utilizando objetos IndexSearcher de larga duración de forma más rápida que con búsquedas con instancias de IndexSearcher recientemente creadas. Una instancia de IndexSearcher compartida sería suficiente para obtener búsquedas rápidas en un entorno de un solo usuario, pero un caso de uso estándar en un entorno de servidor es que varios clientes buscan en el índice simultáneamente. Para evitar la secuenciación de las solicitudes de búsqueda en este valor, lo que degradaría el rendimiento de búsqueda individual, el GSS utiliza una agrupación IndexSearcher que mantiene un número definido de instancias de IndexSearcher para la reutilización mediante solicitudes de búsqueda simultáneas.

Un IndexSearcher sólo verá el índice como el "punto en el tiempo" en el que se abrió. Las actualizaciones del índice después de que se abriera IndexSearcher no son visibles hasta que el IndexSearcher se reabra. Cada instancia de IndexSearcher puede utilizar una cantidad muy importante de memoria en función del tamaño de índice y si el índice se ha actualizado mientras tanto o no. La agrupación de IndexSearcher se encarga de cerrar y reabrir instancias de IndexSearcher cuando se produce una actualización del índice.

12.6.2 Propiedades de configuración de la agrupación

La agrupación IndexSearcher tiene dos opciones básicas: el tamaño inicial y el tamaño máximo. El parámetro siguiente

```
curam.searchserver.luceneadaptor.searcher.pool.initialsize
```

especifica cuántas instancias de IndexSearcher estarán abiertas en el inicio y se mantendrán abiertas en todo momento para su uso por parte de clientes de búsqueda. Esta es una opción obligatoria y toma valores enteros positivos, incluido 0. Si no se especifica el valor predeterminado es "0". Generalmente, esta propiedad debe establecerse en el número máximo previsto de búsquedas de cliente simultáneas.

```
curam.searchserver.luceneadaptor.searcher.pool.maxsize
```

especifica cuál es el número máximo de instancias de IndexSearcher que se pueden abrir en cualquier momento específico. Si se producen más de este número de búsquedas en cualquier momento, se emitirá una excepción y se anotará con fines de diagnóstico. Esta opción toma números enteros positivos y si no se especifica el valor predeterminado es "100". También existe la opción

```
curam.searchserver.luceneadaptor.searcher.pool.maxsizeunbounded
```

asociado, que significa que el tamaño de agrupación máximo es ilimitado. La opción acepta valores de "true" o "false". Si no se especifica, el valor predeterminado es "true". Si esta opción se establece en "true" el valor de la opción curam.searchserver.luceneadaptor.searcher.pool.maxsize se ignorará. Una de estas dos opciones asociadas es obligatoria.

12.7 Limitaciones de RAM

Los índices del Servidor de búsqueda global se almacenan en la memoria si se ha configurado para hacerlo así. Si se utiliza una JVM de 32 bits, hay una limitación de memoria de ~3GB. No obstante, esta cifra no es solo la memoria disponible para GSS, sino también para todos los demás procesos del sistema. Es importante tener en cuenta que los índices del servicio de búsqueda de gran tamaño podrían superar la cantidad máxima de RAM disponible para el GSS y otros procesos desplegados.

12.7.1 Cálculo del tamaño de índice

El tamaño de índice es aproximadamente del 30% del texto indexado. Las propiedades indexadas y almacenadas del servicio de búsqueda (pueden obtenerse a partir de los atributos SearchServiceField, donde indexed=true y stored=true) se utilizan para calcular el tamaño del índice.

- 1 millones de registros de persona. donde 1 registro = 1 documento del índice.
- 1 documento puede contener las siguientes propiedades indexadas y almacenadas determinadas a partir de la tabla SearchServiceField para un servicio PersonSearch: refnumber(10) forename(20), surname(20), AddressLine1(30), AddressLine2(30), city(20), country(15), gender(10). donde (*) = tamaño de valor máximo en caracteres para ese campo.
- 1 documento = (155 caracteres para el valor almacenado) (66 caracteres para cada nombre de campo/término). = 221.
- Memoria 1M de documentos de persona y Java utilizando unicode de 16 bits por carácter. Total de texto indexado y devuelto 442MB * 30% = 132MB.

12.8 Configuración recomendada

La configuración recomendada para el Servidor de búsqueda genérico de Cúram es el uso de un tipo de índice FILE con la persistencia de índice desactivada como estándar. Esto debería proporcionar un buen rendimiento sin problemas de dimensionamiento. El servidor de búsqueda se debe desplegar como una aplicación por separado y no de forma conjunta con la aplicación Cúram (consulte el apartado Capítulo 11, “Despliegue del Servidor de búsqueda genérico”, en la página 39.

12.9 Configuración recomendada para el entorno de producción

El tipo de índice FILE es la única configuración soportada en el entorno de producción.

Apéndice A. Propiedades de configuración del Servidor de búsqueda genérico de Cúram

A.1 Propiedades de configuración

Antes de iniciar el desarrollo o de desplegar el Servidor de búsqueda genérico de Cúram, deben añadirse los valores siguientes al archivo `bootstrap.properties`, donde sea necesario.

Tabla 3. Valores de configuración básicos del Servidor de búsqueda genérico de Cúram

Nombre de la propiedad	Descripción
curam.searchserver.sync.interval	El intervalo en milisegundos entre invocaciones de sincronización del Servidor de búsqueda genérico. Es en efecto el tiempo máximo entre la actualización de los datos y su disponibilidad para la búsqueda. Si esta propiedad no está establecida, el valor predeterminado consiste en sincronizar cada 3 segundos.
curam.searchserver.sync.username	El nombre de usuario utilizado para iniciar sesión en la aplicación para realizar la sincronización. El usuario debe estar autorizado para ejecutar el identificador de función DoGSSSync.sync. Es necesario solo cuando se ejecuta en el servidor de aplicaciones WebSphere. Si se omite la especificación de esta propiedad y la contraseña asociada no se impedirá que la operación de sincronización se ejecute, pero se producirán avisos de seguridad que indicarán que se está escribiendo en los archivos de registro en cada sincronización.
curam.searchserver.sync.password	Contraseña asociada con el curam.searchserver.sync.username descrito en la entrada anterior. Esta contraseña debe estar cifrada con el destino de creación de cifrado de Cúram estándar.
curam.searchserver.environment.vendor	Esta propiedad debe establecerse en “ITD”, “IBM” o “de BEA”, en función de si está utilizando el servidor de búsqueda en modalidad de desarrollo o desplegándolo en WebSphere o WebLogic. Si esta propiedad no está establecida, el servidor de búsqueda utilizará de forma predeterminada la propiedad curam.environment.as.vendor.
curam.searchserver.server.host	El nombre de dominio o la dirección IP del servidor en el que se ejecuta el servidor de búsqueda. Se debe establecer para poder ejecutar el proceso de inicio del servidor desde la línea de comandos. Si esta propiedad no está establecida, el valor predeterminado es localhost.
curam.searchserver.server.port	El puerto en el que el servicio RMI del servidor de aplicaciones está disponible. Se debe establecer para poder ejecutar el proceso de inicio del servidor desde la línea de comandos.
curam.searchserver.autostartup.disabled	Con fines de prueba y desarrollo, el servidor de búsqueda inicializará sus índices en la primera solicitud de búsqueda, a menos que ya se haya iniciado. En un escenario de despliegue, puede que desee inhabilitar este comportamiento y asegurarse de que el proceso de inicio se ejecute desde la línea de comandos, para tener más control sobre el proceso. Si se establece esta propiedad en true se inhabilitará el comportamiento de inicio automático. Tenga en cuenta que el servidor de búsqueda generará una excepción en respuesta a cualquier intento de búsqueda que se produzca antes de que el inicio se haya completado.
curam.searchserver. luceneadaptor.searcher.index.maxmergedocs	Esta propiedad se utiliza para ajustar el rendimiento de lectura y escritura del índice. Los valores más grandes “1.000.000” son mejores para la escritura de índices por lotes y para acelerar las búsquedas. Los valores más pequeños “10.000” son mejores para la indexación interactiva en la que se producen numerosas actualizaciones de índice individuales.
curam.searchserver.luceneadaptor.document.flush.count	Indica el recuento de documentos que se deben actualizar antes de realizar un vaciado en el índice, al tratar con un gran lote de documentos. Si no se especifica, toma como valor predeterminado 1000 documentos. El ajuste de esta propiedad puede reducir el tiempo necesario para crear el índice inicialmente para la persistencia del índice o al iniciarse el servidor.
curam.searchserver.term.min.length	Longitud mínima permitida de un término de búsqueda. El valor predeterminado es de dos caracteres. Si se utilizan términos de búsqueda muy cortes el resultado será un rendimiento de la búsqueda deficiente, y normalmente una mala calidad de los resultados de la búsqueda.
curam.searchserver.directory.type	Especifica el tipo de almacenamiento que se utilizará para los servicios de búsqueda; puede ser DVD-RAM, FILE. RAM es el tipo de índice predeterminado y resulta adecuado para índices menores que requieren un rendimiento muy rápido. El valor FILE proporciona almacenamiento para índices de gran tamaño en el sistema de archivos.

Tabla 3. Valores de configuración básicos del Servidor de búsqueda genérico de Cúram (continuación)

Nombre de la propiedad	Descripción
curam.searchserver.file.index.location	Esta propiedad indica dónde se debe almacenar el índice de archivo en el sistema de archivos si curam.searchserver.directory.type=FILE con más datos. Si se realiza el despliegue en varias máquinas, la ubicación del archivo debe existir en cada máquina de destino.

Tabla 4. Valores de la agrupación de buscadores del Servidor de búsqueda genérico de Cúram

Nombre de la propiedad	Descripción
curam.searchserver.luceneadaptor.searcher.pool.initialsize	Esta propiedad inicializa el número de buscadores de la agrupación de buscadores durante el inicio. El valor predeterminado es 0.
curam.searchserver.luceneadaptor.searcher.pool.maxsize	Esta propiedad indica el número máximo de IndexSearchers de la agrupación de buscadores. El valor predeterminado es 100.
curam.searchserver.luceneadaptor .searcher.pool.maxsizeunbounded	Esta propiedad establecida en "true" altera temporalmente curam.searchserver.luceneadaptor.searcher.pool.maxsize e indica que no hay número máximo de IndexSearchers permitidos en la agrupación de buscadores. El valor predeterminado es "true".
curam.searchserver.luceneadaptor.searcher.pool.mergefactor	Esta propiedad se utiliza para ajustar el rendimiento de lectura y escritura del índice. El valor predeterminado es "10". El valor mínimo es "2". Los valores más altos causan un uso superior de la memoria RAM, búsquedas más lentas pero una escritura de índice más rápida.

Tabla 5. Valores de persistencia del Servidor de búsqueda genérico de Cúram

Nombre de la propiedad	Descripción
curam.searchserver.server.index.persistence.enable	Esta propiedad debe establecerse en "true" para habilitar la persistencia de índice. Si esta propiedad no está establecida, el valor predeterminado es "false".
curam.searchserver.custom.db.init	Esta propiedad debe establecerse en "true" al personalizar las tablas de base de datos de persistencia de índice. Indica que las tablas de persistencia de índice predeterminadas no se utilizarán y el archivo `CustomDBSearchServices.properties` debe utilizarse para configurar estas tablas.

Apéndice B. Listados DMX de muestra: PersonSearch

B.1 Registro del servicio de búsqueda

```
      <?xml version="1.0" encoding="UTF-8"?>
<table name="SEARCHSERVICE">

  <column name="
      searchServiceId
      " type="text" />
  <column name="
      name
      " type="text" />
  <column name="
      extKeyName
      " type="text" />
  <column name="
      analyzer
      " type="text" />
  <column name="
      locked
      " type="bool" />
  <column name="
      forcedReindexTimeStamp
      " type="timestamp" />
  <column name="
      mapperName
      " type="text" />
  <column name="
      prstBlobSize
      " type="text" />
  <row>
    <attribute name="searchServiceId">
      <value>
      PersonSearch
      </value>
    </attribute>
    <attribute name="name">
      <value>
      PersonSearch
      </value> </attribute>
    <attribute name="extKeyName">
      <value>
      ConcernRoleID
      </value> </attribute>
    <attribute name="analyzer">
      <value>
      STANDARD
      </value>
    </attribute>
    <attribute name="locked">
      <value>
      0
      </value>
    </attribute>
    <attribute name="forcedReindexTimeStamp">
      <value>
      SYSTIME
      </value>
    </attribute>
    <attribute name="mapperName">
      <value>
      curam.core.impl.PersonSearchMapper
```

```
      </value>
    </attribute>
    <attribute name="prstBlobSize">
      <value>
      50M
      </value>
    </attribute>
  </row>
</table>
```

B.2 Registro de campo del servicio de búsqueda

```
      <?xml version="1.0" encoding="UTF-8"?>
<table name="SEARCHSERVICEFIELD">

  <column name="
      searchServiceFieldId
      " type="text" />
  <column name="
      searchServiceId
      " type="text" />
  <column name="
      name
      " type="text" />
  <column name="
      indexed
      " type="bool" />
  <column name="
      type
      " type="text" />
  <column name="
      stored
      " type="bool" />
  <column name="
      entityName
      " type="text" />
  <column name="
      analyzerName
      " type="text" />
  <column name="
      untokenized
      " type="bool" />

  <row>
    <attribute name="searchServiceFieldId">
      <value>
      field0
      </value>
    </attribute>
    <attribute name="searchServiceId">
      <value>
      PersonSearch
      </value>
    </attribute><attribute name="name">
      <value>
      primaryAlternateID
      </value>
    </attribute><attribute name="indexed">
      <value>
      1
      </value>
    </attribute><attribute name="type">
      <value>
      String
      </value>
    </attribute><attribute name="stored">
      <value>
```

```
      1
      </value>
    </attribute>
    <attribute name="entityName">
      <value>
      Person
      </value>
    </attribute>
    <attribute name="analyzerName">
      <value></value>
    </attribute>
    <attribute name="untokenized">
      <value>
      1
      </value>
    </attribute>
  </row>

  <row>
    <attribute name="searchServiceFieldId">
      <value>
      field1
      </value>
    </attribute>
    <attribute name="searchServiceId">
      <value>
      PersonSearch
      </value>
    </attribute><attribute name="name">
      <value>
      firstForename
      </value>
    </attribute><attribute name="indexed">
      <value>
      1
      </value>
    </attribute><attribute name="type">
      <value>
      String
      </value>
    </attribute>
    <attribute name="stored">
      <value>
      1
      </value>
    </attribute>
    <attribute name="entityName">
      <value>
      AlternateName
      </value>
    </attribute>
    <attribute name="analyzerName">
      <value>
      STANDARD
      </value>
    </attribute>
    <attribute name="untokenized">
      <value>
      0
      </value>
    </attribute>
  </row>

......

</table>
```

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en los Estados Unidos. Es posible que IBM no ofrezca los productos, servicios o características que se describen en este documento en otros países. Solicite información al representante local de IBM acerca de los productos y servicios disponibles actualmente en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implica que sólo pueda utilizarse ese producto, programa o servicio de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ningún derecho de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM. IBM podría tener patentes o solicitudes de patentes pendientes relacionadas con el tema principal que se describe en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

EE.UU.

Para consultas sobre licencias relacionadas con información de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM de su país o envíe sus consultas, por escrito, a:

Intellectual Property Licensing

Legal and Intellectual Property Law.

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokio 103-8510, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde las disposiciones en él expuestas sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERABILIDAD, COMERCIALIZACIÓN O IDONEIDAD PARA UN PROPÓSITO DETERMINADO. Algunos países no permiten la renuncia a garantías explícitas o implícitas en determinadas transacciones, por lo que puede que esta declaración no sea aplicable en su caso.

La información de este documento puede incluir imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede reservarse el derecho de realizar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento sin previo aviso.

Cualquier referencia incluida en esta información a sitios web que no sean de IBM sólo se proporciona para su comodidad y en ningún modo constituye una aprobación de dichos sitios web. El material de esos sitios web no forma parte del material de este producto de IBM y la utilización de esos sitios web se realizará bajo su total responsabilidad.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente. Los titulares de licencias de este programa que deseen obtener información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y el uso mutuo de información que se haya intercambiado, deben ponerse en contacto con:

IBM Corporation

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

EE.UU.

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos el pago de una cuota.

IBM proporciona el programa bajo licencia que se describe en este documento y todo el material bajo licencia disponible para el mismo bajo los términos del Acuerdo de cliente de IBM, el Acuerdo internacional de licencias de programas de IBM o cualquier acuerdo equivalente entre las partes.

Los datos de rendimiento incluidos aquí se determinaron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar considerablemente. Algunas mediciones podrían haberse realizado en sistemas en desarrollo y, por lo tanto, no existe ningún tipo de garantía de que dichas mediciones sean las mismas en los sistemas con disponibilidad general. Además, es posible que algunas mediciones se hayan calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a sus entornos específicos.

La información relacionada con productos que no son de IBM se ha obtenido de los proveedores de dichos productos, de sus anuncios publicados o de otras fuentes de disponibilidad pública.

IBM no ha probado estos productos y no puede confirmar la precisión de rendimiento, compatibilidad ni otras afirmaciones relacionadas con productos que no son de IBM. Las preguntas relativas a las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de dichos productos.

Las afirmaciones relativas a las intenciones futuras de IBM están sujetas a cambio o retirada sin previo aviso, y sólo representan objetivos

Todos los precios de IBM que se muestran son precios de distribuidor recomendados por IBM, corresponden al momento actual y están sujetos a cambios sin aviso previo. Los precios de los distribuidores pueden variar.

Esta información se ofrece con fines de planificación únicamente. La información incluida en este documento puede cambiar antes de que los productos descritos estén disponibles.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos de la manera más completa posible, los ejemplos incluyen los nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con nombres y direcciones utilizados por empresas comerciales reales son mera coincidencia.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir los programas de ejemplo de cualquier forma, sin tener que pagar a IBM, con intención de desarrollar, utilizar, comercializar o distribuir programas de aplicación que estén en conformidad con la interfaz de programación de aplicaciones (API) de la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni implicar la fiabilidad, capacidad de servicio o función de estos programas. Los programas de ejemplo se proporcionan "TAL CUAL", sin garantía de ningún tipo. IBM no es responsable de ningún daño resultante de la utilización de los programas de ejemplo por parte del usuario.

Todas las copias o fragmentos de las copias de estos programas de ejemplo o cualquier trabajo que de ellos se derive, deberán incluir un aviso de copyright como el que se indica a continuación:

© (el nombre de la empresa) (año). Algunas partes de este código proceden de los programas de ejemplo de IBM Corp.

© Copyright IBM Corp. _escriba el año o los años_. Reservados todos los derechos.

Si visualiza esta información en una copia software, es posible que no aparezcan las fotografías ni las ilustraciones en color.

Marcas registradas

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., registradas en muchas jurisdicciones en todo el mundo. Otros nombres de productos y servicios pueden ser marcas registradas de IBM u otras empresas. Encontrará una lista actual de marcas registradas de IBM en la web en "Copyright and trademark information" en http://www.ibm.com/legal/us/en/copytrade.shtml.

Apache es una marca registrada de Apache Software Foundation.

Oracle, WebLogic Server, Java y todas las marcas registradas y logotipos basados en Java son marcas registradas de Oracle y/o sus filiales.

Otros nombres pueden ser marcas registradas de sus respectivos propietarios. Otros nombres de empresas, productos o servicios pueden ser marcas registradas o de servicio de terceros.

Impreso en España