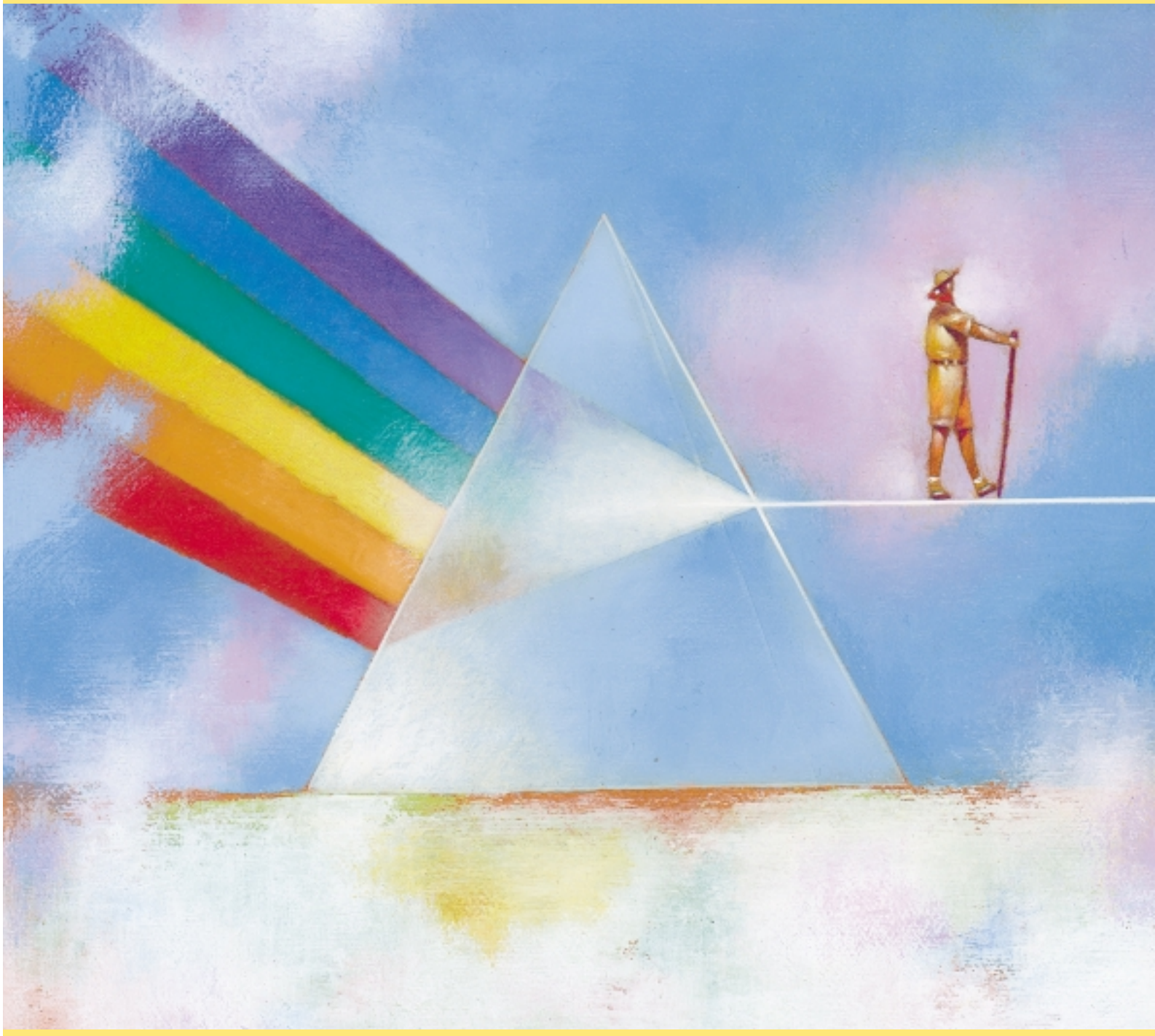
Guía tecnológica de Caché



InterSystems
CACHÉ™
Make Applications Faster

Contenidos

Introducción

El mundo informático ha entrado en la era "post-relacional".

Las aplicaciones actuales de proceso de transacciones tienen unos requisitos que superan las posibilidades de la tecnología relacional: deben abarcar grandes redes, dar servicio a miles de clientes y seguir proporcionando un rendimiento excelente, compatibilidad con la Web y operaciones sencillas con bajo coste. Cada vez en mayor medida, las empresas alcanzan los límites impuestos por sus bases de datos relacionales. Cuando intentan escalar las aplicaciones de proceso de transacciones, la complejidad se multiplica, el rendimiento disminuye y los costes se disparan. Por otra parte, ni siquiera las más recientes bases de datos "relacionales de objetos", que colocan una capa de objetos sobre un motor de datos relacional, pueden utilizar con eficacia las abundantes y complejas estructuras de datos que necesitan las aplicaciones de última generación.

Es más, las aplicaciones se tornan más complejas, como la tecnología que utilizan, ralentizando el proceso de desarrollo y reduciendo la fiabilidad. Ahora más que nunca, lo que se necesita es desarrollar rápidamente las aplicaciones.

Caché, la base de datos de alto rendimiento de la era post-relacional, constituye una nueva generación de tecnología de bases de datos que soluciona esas necesidades combinando un servidor de datos multidimensional con un servidor de aplicaciones versátil. Caché, que incorpora tecnología avanzada de objetos, desarrollo rápido para Web, SQL ampliado y tecnología exclusiva de almacenamiento de datos en memoria cache, proporciona niveles de rendimiento y escalabilidad que la tecnología relacional no puede alcanzar.

Además, la capacidad del Servidor de Aplicaciones Caché para integrar fácilmente el acceso a diversas tecnologías, su modelo de datos multidimensional para almacenar datos complejos y sus posibilidades de programación avanzada de objetos le permiten desarrollar rápidamente aplicaciones sofisticadas de bases de datos, multiplicando la productividad de programación y reduciendo el tiempo para su comercialización.

Caché admite todas las formas tradicionales de crear aplicaciones Web y añade una tecnología exclusiva, Caché Server Pages, o CSP, optimizada para el desarrollo rápido de sofisticados sistemas orientados a bases de datos. CSP incorpora una arquitectura avanzada de objetos, páginas de servidor dinámicas y conectividad con herramientas líderes en desarrollo para Web.

Esta guía tecnológica proporciona una visión general del Servidor de Datos y del Servidor de Aplicaciones Multidimensional de Caché. Explica cómo Caché se adapta al proceso básico de desarrollo de aplicaciones, permitiendo a los desarrolladores profesionales traspasar los límites de la tecnología relacional heredada para producir sofisticadas aplicaciones de alto rendimiento para proceso de transacciones en la Web y de otros entornos centrados en red.

La Guía tecnológica de Caché está dividida en cuatro secciones principales:

■ **Capítulo 1: Acceso relacional o mediante objetos**
Esta sección describe los pros y los contras de la tecnología relacional y valora el acceso mediante objetos a la base de datos.

■ **Capítulo 2: El servidor de datos multidimensional de Caché**
Esta sección describe la base de datos de Caché y las características principales asociadas a las operaciones de base de datos.

■ **Capítulo 3: El Servidor de Aplicaciones Caché**
Esta sección describe las características de soporte a lenguajes y conectividad de Caché.

■ **Capítulo 4: Caché Server Pages**
Esta sección describe el desarrollo para la Web y las características de ejecución de Caché.

Capítulo 1: Diseño de modelo de datos: Acceso relacional o mediante objetos

Al comenzar a diseñar una nueva aplicación, los desarrolladores deben decidir qué enfoque darle al modelo de datos. Para la mayoría, se trata de elegir entre el modelo de datos tradicional como tablas relacionales y el enfoque más reciente del modelo de objetos. Frente a la necesidad de manejar datos complejos, muchos desarrolladores creen que el modelo de objetos resulta más efectivo.

Caché soporta acceso a datos SQL y de objetos, y hay momentos en los que cada uno de ellos resulta adecuado. Para comprender los usos de cada modelo de datos y porqué el de objetos es el preferido normalmente por los desarrolladores modernos, sería útil conocer cómo y porqué se ha desarrollado cada uno de ellos.

Tecnología relacional

En los primeros días de la informática, el proceso de la información se realizaba en enormes sistemas mainframe y el acceso a los datos estaba, en gran medida, limitado a los profesionales de la TI. Las bases de datos tendían a ser de cosecha propia y recuperar o actualizar los datos de forma efectiva requería un profundo conocimiento de la base de datos y de su estructura. Si un usuario quería un informe especial, generalmente tenía que solicitarlo a un departamento central, siempre saturado de trabajo, y que normalmente no estaba disponible a tiempo para influir sobre las decisiones.

Con la llegada de los PCs, el mundo pasó a la era de la informática "centrada en el usuario". La tecnología de bases de datos relacionales se fue ampliando para satisfacer las necesidades de los que querían acceder a los datos corporativos centralizados desde sus puestos de trabajo: querían crear sus propios informes y consultas ad-hoc de la base de datos.

La tecnología relacional introdujo el lenguaje de consulta SQL, que permite utilizar un lenguaje uniforme para hacer preguntas sobre una amplia variedad de datos. SQL funciona estructurando los datos de una forma muy sencilla y estándar: una tabla bidimensional con filas y columnas. SQL se convirtió en la base subyacente de toda una generación de herramientas que permitían a los usuarios hacer preguntas y obtener informes.

Aunque este sencillo modelo de datos permitía la creación de un lenguaje de consulta elegante con el que hacer preguntas, llevaba consigo un pesado lastre. La complejidad inherente a las relaciones entre datos no se adapta de forma natural al simple formato de filas y columnas, así que los datos a menudo se fragmentan en muchas tablas que deben unirse ("join") para llevar a cabo incluso tareas muy sencillas. Esto provoca dos problemas: a) puede resultar muy complicado escribir las consultas debido a la necesidad de hacer uniones ("join") de muchas tablas, y b) la carga general de proceso necesaria cuando las bases de datos relacionales tienen que manejar datos complejos puede ser enorme.

SQL se ha convertido en el estándar para la interacción con las bases de datos y para las herramientas de generación de interoperatividad e informes. Sin embargo, es importante señalar que aunque SQL surgió a partir de las bases de datos relacionales, no tiene porque estar restringido por ellas. Caché soporta SQL como lenguaje de consulta utilizando una tecnología de base de datos multidimensional mucho más sólida.

Tecnología y bases de datos basadas en objetos

La programación de objetos y las bases de datos de objetos son el resultado práctico del trabajo que implicaba la necesidad de modelar las actividades complejas del cerebro. Se ha observado que el cerebro es capaz de almacenar tipos de datos muy distintos y complejos y a pesar de ello manipular de forma habitual esa información aparentemente distinta. Además, existía la necesidad de implementar comportamientos muy complejos en los programas a la vez que se ocultaba esa complejidad. Claramente, ambas características son cada vez más ciertas en las nuevas aplicaciones.

Acceso mediante objetos frente a acceso relacional

En la tecnología de objetos, la complejidad de los datos está contenida en el objeto, y se accede a los datos mediante una interfaz sencilla y uniforme. En contraste, la tecnología relacional también proporciona una interfaz sencilla y uniforme, pero almacena los datos de la forma más simple posible, y el usuario o programador es el responsable de manejar la complejidad de los datos.

Dado que los objetos permiten crear modelos de datos complejos con facilidad, la programación de objetos funciona mejor para programar aplicaciones complejas. De igual forma, el acceso a objetos de la base de datos funciona mejor para insertar datos y actualizar la base de datos (es decir, para el proceso de transaccional).

Caché complementa el acceso a los objetos con un lenguaje de consulta SQL ampliado a objetos. SQL es un potente lenguaje para buscar en una base de datos, y es ampliamente utilizado en herramientas de generación de informes. *Sin embargo, creemos que SQL es más adecuado para dicho fin (consultas e informes) que para el proceso transaccional (en el que se desenvuelve con lentitud y a menudo con poca eficacia).*

Visión general del modelo de datos de objetos y la programación orientada a objetos

El modelo de objetos de Caché se basa en el estándar ODMG (Object Database Management Group, Grupo de gestión de bases de datos de objetos) y soporta muchas características avanzadas, incluyendo la herencia múltiple.

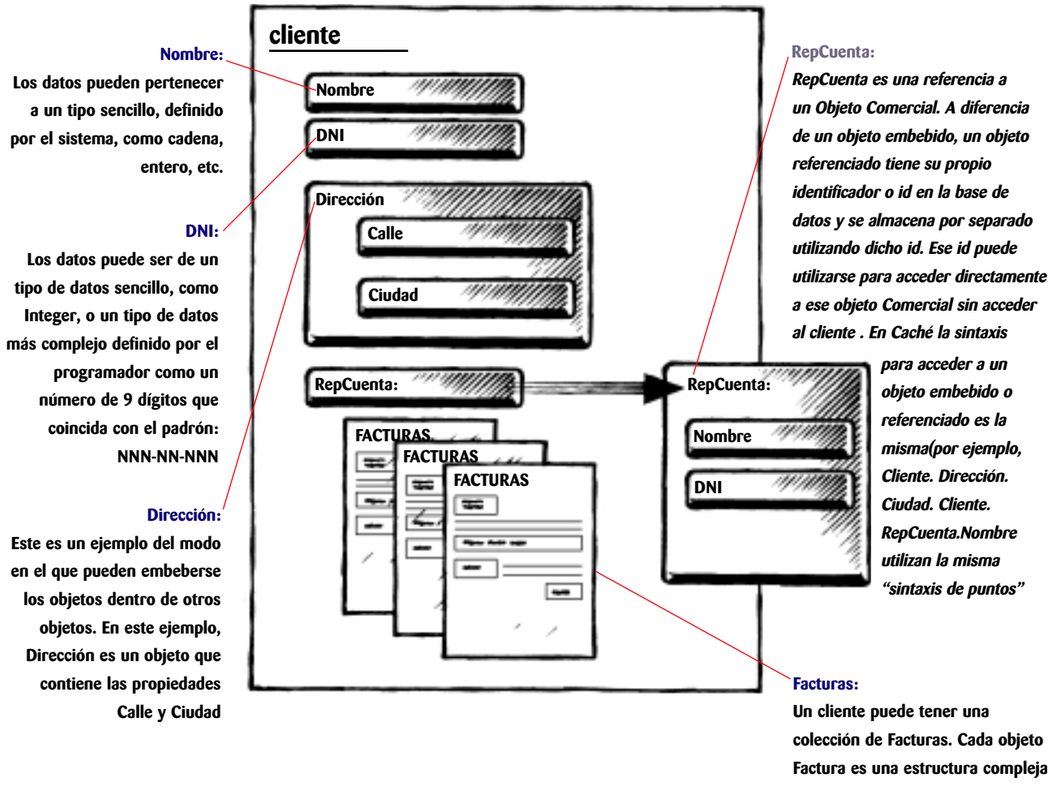
La tecnología de objetos intenta tratar y utilizar la información como lo hace la mente humana. Teóricamente, las unidades de información o entidades se ven como objetos que tienen un estado (representado por sus valores de datos) y un comportamiento (que se puede observar y modificar mediante su código). Un ejemplo sería un objeto Factura que tiene datos como un número de factura, un importe total y un código como Print().

A diferencia de las tablas relacionales, los objetos incorporan tanto los datos como el código. Conceptualmente, y a veces en la práctica, un objeto es un paquete que incluye los valores de los datos de dicho objeto ("propiedades") y una copia de todo su código ("métodos"). Los métodos de un objeto envían mensajes para comunicarse con otros métodos de éste y de otros objetos. Para reducir el espacio de almacenamiento, es habitual que los objetos de la misma clase compartan la misma copia del código (por ejemplo, sería poco realista que cada objeto Factura tuviera su propia copia de código). Además, en Caché, las llamadas a métodos suelen generan llamadas a funciones, en lugar de aumentar la sobrecarga de paso a mensajes. Sin embargo, estas técnicas de implementación quedan ocultas al programador; así que siempre es más adecuado pensar en términos de objetos pasando mensajes.

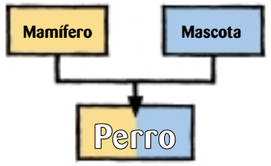
¿Cuál es la diferencia entre un objeto y una clase? Una clase es la estructura y el código definitorio proporcionados por el programador. Incluye una descripción de la naturaleza de los datos y de cómo se almacenan, además de todo el código, pero no contiene ningún dato. Un objeto es una entidad real que es una "instancia" particular de una clase. Por ejemplo, la factura Nº 123456 es un objeto de la clase Factura.

La tecnología de objetos también tiende a mostrar los datos de forma natural, sin restringir las propiedades a tipos de datos simples basados en la lógica informática. Los objetos pueden contener otros objetos o referencias a otros objetos, lo que facilita la generación de modelos de datos útiles y con sentido. He aquí un ejemplo sencillo de un objeto Cliente:

Objeto cliente:



También se pueden utilizar **relaciones** para conectar objetos; por ejemplo, en lugar de que Factura sea una colección embebida en Cliente, las facturas se pueden almacenar de forma independiente y habría una relación uno-con-muchos entre Cliente y Facturas. Las referencias son más ligeras que las relaciones; las relaciones son bidireccionales, soportan integridad referencial y requieren más tiempo de proceso.



HERENCIA MÚLTIPLE:
Subclases que pueden heredar atributos de una o mas superclases.

Conceptos clave sobre objetos

La **herencia** es la capacidad de que una clase de objetos proceda de otra. La nueva clase (una subclase) contiene todas las propiedades y métodos de su superclase, así como propiedades y métodos adicionales exclusivos de ella. Los objetos de la subclase deben tener una relación "es un/a" con su superclase. Por ejemplo, un perro "es un" mamífero, luego tiene sentido que la clase Perro herede todas las propiedades y métodos de la clase Mamífero. La subclase puede tener propiedades y métodos adicionales (por ejemplo, Perro puede tener una propiedad denominada NúmIDPerro que no esté presente en la clase Mamífero). Una subclase también puede modificar una definición heredada (por ejemplo, el método Print() para una subclase de la clase Factura puede ser distinto del método Print() de Factura).

La **herencia múltiple** significa que una subclase puede proceder de más de una superclase. Un perro "es un" mamífero y "es una" mascota, así que la clase de objeto "Perro" puede heredar atributos tanto de la clase "Mamífero" como de la clase "Mascota".

Encapsulación significa que, en la medida en que afecte a la aplicación, los objetos se pueden contemplar como una especie de "caja negra". Sin importar lo complejo que sea, una clase de objeto tiene un número limitado de propiedades y métodos **públicos**. Una vez definido, la aplicación no necesita conocer el funcionamiento interno de un objeto. Sólo maneja propiedades y métodos. Cualquier método puede acceder a las propiedades y métodos públicos, mientras que a las propiedades y métodos privados sólo tienen acceso los métodos de la misma clase.

Polimorfismo hace referencia al hecho de que los métodos utilizados en varias clases pueden compartir una interfaz común, incluso si su implementación subyacente es distinta. Por ejemplo, supongamos que una aplicación utiliza distintas clases de objetos: Carta, Etiqueta de correo, y Placa de identidad, y todos ellos contienen un método denominado ImprimirDirec. La aplicación no necesita contener instrucciones especiales sobre dar formato a una dirección para cada tipo de objeto. Simplemente llama al método ImprimirDirec del objeto. El polimorfismo asegura que cada objeto realiza la instrucción en el modo más adecuado para la clase a la que pertenece.

¿Por qué elegir objetos para el modelo de datos?

Para las nuevas aplicaciones de base de datos, la mayoría de los desarrolladores prefiere utilizar tecnología de objetos porque pueden desarrollar aplicaciones complejas de una forma más rápida y modificarlas posteriormente con facilidad. La tecnología de objetos proporciona muchas ventajas:

- Los objetos soportan una estructura de datos más completa que describe de forma más natural los datos del mundo real.
- Programar es más sencillo; Es más fácil hacer el seguimiento de lo que se está haciendo y de lo que se está modificando.
- Las versiones personalizadas de las clases pueden sustituir fácilmente a las estándares, facilitando la personalización y el soporte de una aplicación en sus nuevas versiones. Esta característica es especialmente importante para VARs que deben dar soporte a toda una gama de versiones personalizadas.
- El enfoque de "caja negra" en la encapsulación significa que el programador puede mejorar el funcionamiento interno de los objetos sin que afecte al resto de la aplicación.
- Los objetos proporcionan un método sencillo de conectar distintas tecnologías y distintas aplicaciones.
- Java utiliza de forma natural la tecnología de objetos, permitiendo un desarrollo Web más fácil y con las interfaces de usuario basadas en GUI.
- Muchas de las nuevas herramientas incorporan tecnología de objetos.
- Los objetos proporcionan un buen aislamiento entre la interfaz de usuario del resto de la aplicación. De este modo, cuando es necesario adoptar una nueva tecnología de interfaz de usuario (quizás tecnología Web o alguna tecnología futura aún no prevista) se puede reutilizar la mayor parte del código.

Almacenamiento de datos de objetos...

Desafortunadamente, aunque muchas aplicaciones se escriben ahora con lenguajes de programación de objetos, se intenta implementar a la fuerza los datos de los objetos en tablas relacionales planas. Esto afecta seriamente a las ventajas de la tecnología de objetos.

Caché proporciona una estructura de datos multidimensional que almacena de forma natural datos complejos de objetos. El resultado: más rapidez en el acceso a los datos y en la programación.

...y acceso relacional

Por supuesto, muchas herramientas (como los generadores de informes) utilizan SQL, no tecnología de objetos, para acceder a los datos, y es posible que algunos desarrolladores deseen usar alguna de esas herramientas.

Una característica única de Caché es que siempre que se define una clase de objetos, Proporciona automáticamente acceso SQL completo a esos datos. Por tanto, sin trabajo adicional, las herramientas basadas en SQL funcionarán inmediatamente con los datos de Caché, manteniendo el alto rendimiento del Servidor de Datos Multidimensional de Caché.

El proceso inverso también es válido. Cuando se importa una descripción DDL de una base de datos relacional, Caché generará automáticamente una descripción relacional/de objetos integrada de los datos, habilitando su acceso inmediato como objetos y a través de SQL.

La Arquitectura de Datos Unificada de Caché mantiene sincronizadas estas vías de acceso; sólo hay una descripción de datos que editar.

Capítulo 2: El servidor de datos multidimensionales de Caché

Modelo de datos multidimensional

Estructura de Datos Multidimensional Compleja

La base de datos de alto rendimiento de Caché utiliza un **modelo de datos multidimensional** que permite el almacenamiento eficaz y compacto de datos en una estructura de datos compleja. Con Caché es posible acceder a los datos o actualizarlos sin efectuar los "joins" complicados y lentos necesarios en las bases de datos relacionales.

Aunque en ocasiones se ha descrito como un "hipercubo" o "espacio n-dimensional", una descripción más precisa del modelo de almacenamiento de Caché sería una colección de arrays multidimensionales de tipo disperso denominados "globals". Los datos se pueden almacenar en un global con cualquier número de subíndices. Además, los subíndices son independientes, y pueden ser cualquier cosa:, String, integer, floating point, etc. Esto significa que un subíndice podría ser un entero, como 34, mientras que otro podría ser un nombre significativo, como "ArtículosLínea", incluso en el mismo nivel de subíndice.

Por ejemplo, una aplicación de inventario de stock que proporcione información sobre artículo, talla, color y estampado podría tener una estructura como la siguiente:

```
^Stock(artículo, talla, color, estampado) = cantidad
```

Algunos datos de ejemplo:

```
^Stock("vestido",4,"azul","floral")=3
```

Con esta estructura es muy sencillo determinar si hay vestidos azules de la talla 4 con estampado floral, simplemente accediendo a ese nodo de datos. Si un cliente desea un vestido de la talla 4 pero no está seguro del color y del estampado, es fácil mostrar una lista de todos ellos pasando por todos los nodos de datos bajo ^Stock("vestido", 4).

En este ejemplo, todos los nodos de datos eran de naturaleza similar, (almacenaban una cantidad), y todos se almacenaban en el mismo nivel de subíndice (4 subíndices) con subíndices similares (el tercer subíndice siempre era texto que representaba un color). Sin embargo no tiene por qué ser similar. No todos los nodos de datos tienen que tener el mismo número o tipo de subíndices, y pueden contener distintos tipos de datos.

A continuación viene un ejemplo de un global más complejo con datos de una factura que tiene distintos tipos de datos almacenados en distintos niveles de subíndices:

```
^Factura(n° factura,"Cliente") = Información del cliente

^Factura(n° factura,"Fecha") = Fecha factura

^Factura(n° factura,"Artículos") = n° de artículos de la factura

^Factura(n° factura,"Artículos",1,"Serie") = n° de serie del
primer artículo

^Factura(n° factura,"Artículos", 1,"Cantidad") = cantidad del
primer artículo

^Factura(n° factura,"Artículos",1,"Precio") = precio del primer
artículo

^Facturan° factura,"Artículos",2,"Serie") = n° de Serie del
segundo artículo
```

etc.

Múltiples elementos de datos por nodo

Con frecuencia, un solo elemento de datos se almacena en un nodo de datos, como una fecha o una cantidad, pero algunas veces resulta útil almacenar múltiples elementos de datos juntos como un solo nodo de datos. Esto es particularmente útil cuando existe un conjunto de datos relacionados a los que se accede en grupo con frecuencia. También puede mejorar el rendimiento al necesitar menos accesos a la base de datos, especialmente si se utilizan redes.

Por ejemplo, en la factura anterior cada artículo incluía el número de serie, la cantidad y el precio, todos ellos como nodos separados, pero se podrían almacenar en un solo nodo:

```
^Factura(n° factura,"ArtículosLínea",n° artículo)
```

Para simplificar esto, Caché dispone de la función $list(), que puede agrupar varios elementos de datos en una cadena byte de longitud delimitada y más tarde desagruparlos conservando el tipo de datos.

Las ventajas de Caché

Rendimiento

Utilizando un modelo de datos multidimensional eficaz con técnicas de almacenamiento mediante Arrays de tipo disperso en lugar de un laberinto inmanejable de tablas bidimensionales, el acceso a los datos y las actualizaciones se realizan con menos E/S en disco. La E/S reducida significa que las aplicaciones se ejecutarán más rápidamente.

Escalabilidad

El modelo de datos multidimensional transaccional permite a las aplicaciones basadas en Caché escalarse a muchos miles de usuarios sin sacrificar el alto rendimiento. Esto se debe a que el acceso a los datos de un modelo multidimensional no se ve afectado de forma significativa por el tamaño o la complejidad de la base de datos, si lo comparamos con los modelos relacionales. Las transacciones acceden a los datos que necesitan sin realizar complicados "joins" o ir haciendo operaciones de tabla en tabla.

El uso del bloqueo lógico en Caché para las actualizaciones en lugar de bloquear las páginas físicas contribuye también de forma importante a la concurrencia, al igual que lo es su sofisticado almacenamiento en memoria intermedia cache a través de las redes.

Proceso de transacciones con gran número de usuarios

El acceso eficaz a los datos hace que el modelo multidimensional sea el modelo natural para el proceso transaccional. Los procesos de Caché no consumen tiempo haciendo un "join" de varias tablas, por lo que se ejecutan más rápidamente.

El bloqueo lógico favorece la alta concurrencia

En sistemas con miles de usuarios, la reducción de conflictos entre procesos concurrentes es crítica para conseguir alto rendimiento. Uno de los mayores conflictos se da entre transacciones que intentan acceder a los mismos datos.

Los procesos de Caché no bloquean páginas enteras de datos al ejecutar actualizaciones. Dado que las transacciones requieren acceso frecuente o cambios a pequeñas cantidades de datos, en Caché el bloqueo de la base de datos se realiza a nivel lógico. Los conflictos de la base de datos se reducen aún más utilizando operaciones individuales de suma y resta, que no requieren bloqueo. (Estas operaciones resultan especialmente útiles en los contadores incrementales utilizados para asignar números de id y para modificar contadores estáticos, que son "zonas calientes" de una base de datos y que con otro tratamiento producirían conflictos frecuentes entre transacciones concurrentes.)

Con Caché, las transacciones individuales se ejecutan más rápidamente, y se pueden ejecutar más transacciones simultáneamente.

El modelo multidimensional ofrece una descripción realista de los datos

El modelo multidimensional es también el sistema natural para describir y almacenar datos complejos. Los desarrolladores pueden crear estructuras de datos que representan de forma exacta los datos del mundo real, lo que hace que el desarrollo de aplicaciones sea más rápido y más fácil mantenerlas.

Datos de longitud variable en sparse arrays

Como los datos de Caché tienen de forma inherente una longitud variable, y se almacenan en sparse arrays, Caché suele necesitar menos de la mitad del espacio que necesita una base de datos relacional. Además de reducir los requerimientos de disco, el almacenamiento de datos compacto mejora el rendimiento porque se pueden leer y escribir más datos con una sola operación de E/S, y los datos se pueden almacenar en memoria intermedia cache de forma más eficiente.

Las declaraciones y definiciones no son necesarias

Las declaraciones, definiciones o asignaciones no son necesarias para acceder directamente a los datos o almacenarlos, y tampoco es necesario especificar el número o tipo de subíndices, ni el tipo ni el tamaño de los datos. Las arrays multidimensionales son inherentemente independientes de los datos, tanto en los datos como en los subíndices. Los datos globales simplemente se crean mientras se insertan datos con el comando SET.

Sin embargo, para utilizar el acceso de objetos y SQL de la base de datos, es necesaria la información del diccionario de datos. Al especificar el diccionario de datos para objetos y SQL, los desarrolladores tienen la opción de permitir que los asistentes (wizards) seleccionen automáticamente la estructura de datos multidimensional que mejor se adapte a sus datos, o pueden especificar directamente el mapeado.

Namespaces

En Caché, los datos y el código de Caché ObjectScript se almacenan en archivos de disco en archivos con el nombre CACHE.DAT (sólo uno por directorio). Cada archivo contiene numerosos "globals" (arrays multidimensionales). En un archivo, el nombre de cada global debe ser único, pero archivos diferentes pueden contener el mismo nombre de global. Sin mucho rigor, se pueden considerar estos archivos como bases de datos.

En lugar de especificar qué archivo CACHE.DAT utilizar, cada proceso de Caché utiliza un "namespace" para acceder a los datos. Un namespace es un mapa lógico que mapea los nombres de las arrays multidimensionales globales y el código de las rutinas a los archivos CACHE.DAT, incluyendo el Servidor de datos y el nombre del directorio del archivo. Si se mueve un archivo de una unidad de disco u ordenador a otro, se cambia el mapeo del namespace.

Generalmente, un namespace especifica la forma de compartir cierta información del sistema con otros namespaces, y el resto de los datos del namespace está en un solo archivo CACHE.DAT que únicamente utiliza ese namespace. Sin embargo, esta es una estructura flexible que permite el mapeos arbitrarios, y no es extraño que un namespace mapee el contenido de varios archivos CACHE.DAT.

Las ventajas de Caché

Desarrollo rápido

Caché permite el desarrollar más rápidamente porque la estructura de datos proporciona un almacenamiento natural y de fácil comprensión de datos complejos y no requiere gran cantidad de declaraciones ni definiciones largas o complicadas. El acceso directo a los globals es muy sencillo, lo que permite utilizar la misma sintaxis de lenguaje que para acceder a las arrays.

Rentabilidad

Comparadas con las aplicaciones relaciones de tamaño similar, las aplicaciones basadas en Caché requieren mucho menos hardware y ningún administrador de bases de datos. La gestión y utilización del sistema s sencilla.

"Realizamos algunos análisis de lo que supondría crear un modelo de entrada de solicitudes médicas en la base de datos relacional. El resultado eran más de 700 tablas."

Steve Flammini
Director de desarrollo de aplicaciones
Partners HealthCare System Inc.

Acceso integrado a base de datos

Varias formas de acceder a los datos

Caché proporciona a los programadores la libertad de almacenar y acceder a los datos mediante acceso a objetos, SQL o acceso directo a las estructuras multidimensionales. Independientemente del método de acceso, todos los datos de la base de datos de Caché se almacenan en las arrays multidimensionales.

Una vez almacenados los datos, se pueden utilizar simultáneamente los tres métodos de acceso sobre los mismos datos, con concurrencia total.

Definir una vez, utilizar en cualquier sitio. La Arquitectura de Datos Unificada de Caché define simultáneamente objetos y tablas.

Arquitectura de Datos Unificada

Una característica única de Caché es su Arquitectura de Datos Unificada. Siempre que se define una clase de objetos en la base de datos, Caché genera automáticamente una descripción relacional SQL de dicha clase. De forma similar, cuando se importa una descripción DDL de una base de datos relacional, Caché creará automáticamente junto con la descripción relacional, la definición de la clase equivalente, lo que posibilita el acceso inmediato como objetos. Caché mantiene estas descripciones unidas: En la práctica existe una única definición de datos.

Acceso mediante objetos

Algunos desarrolladores prefieren pensar en los datos como objetos. Los objetos agrupan datos ("propiedades") y código ("métodos"), permiten embeber objetos dentro de otros objetos y soportan colecciones (una propiedad puede contener múltiples valores o múltiples objetos embebidos). La programación orientada a objetos se ha convertido en el paradigma dominante para el desarrollo de nuevas aplicaciones.

El modelo de objetos de Caché se basa en el estándar ODMG. Caché soporta una serie completa de conceptos de programación orientada a objetos, incluyendo la encapsulación, los objetos embebidos, la herencia múltiple, el polimorfismo y las colecciones.

"Caché ofrece… orientación a objetos junto con acceso SQL e interacción con otras bases de datos. Lo mejor de todos los entornos posibles. Y Caché es rápido."

John Gantz
en ComputerWorld
10 de mayo de 1999

Acceso SQL

Otros desarrolladores prefieren pensar en los datos como tablas relacionales, que organizan los datos en filas y columnas y no contienen ningún código. Sólo se proporcionan las operaciones básicas Select, Insert, Update y Delete.

El acceso SQL es importante porque muchas aplicaciones y herramientas existentes utilizan SQL como lenguaje de consulta. Dado que los datos de Caché se almacenan realmente en eficientes estructuras multidimensionales, las aplicaciones que utilizan SQL obtienen un mejor rendimiento con Caché que cuando se ejecutaban sobre bases de datos relacionales tradicionales.

Caché SQL es el lenguaje de consulta para Caché. Es una implementación mejorada de objetos para SQL, y soporta ODBC, OCI y JDBC. Las consultas pueden embeberse en rutinas de Caché ObjectScript, y mediante ellas en los métodos de Caché Objects.

"Hemos estimado que con el mismo hardware que el sistema Sybase, Caché es 100 veces más rápido gestionando transacciones y 20 veces más rápido que Sybase en conjunto".

Rolf Streb
Director de IT
Ministerio de Justicia
Berna, Suiza

Modo de acceso directo global

Además de utilizar objetos o SQL, hay ocasiones en las que tiene más sentido utilizar directamente "referencias a globales" para definir y recuperar los datos de las arrays multidimensionales de Caché. Esto se puede conseguir utilizando la misma sintaxis que se utilizaría para establecer, recuperar o buscar en arrays privados.

El acceso global directo se usa normalmente cuando se necesitan utilizar estructuras poco comunes y muy especializadas y no hay necesidad de proporcionar acceso a objetos o SQL a ellas, o cuando se necesita el mayor rendimiento posible.

Las ventajas de Caché

Flexibilidad
Los modos de acceso a los datos de Caché – Objetos, SQL y Directo – se pueden utilizar concurrentemente sobre los mismos datos. Esta flexibilidad permite a los programadores desarrollar aplicaciones que se adaptan a las necesidades de sus usuarios finales. También ofrece a los programadores la libertad de pensar en los datos de la forma que tenga más sentido para ellos.

Respuesta SQL más rápida
Las aplicaciones relacionales pueden disfrutar un rendimiento significativamente mayor utilizando Caché SQL para conectar con la eficiente base de datos post-relacional de Caché.

Desarrollo rápido de aplicaciones
La tecnología de objetos constituye una potente herramienta para aumentar la productividad del programador. Los desarrolladores pueden pensar en objetos (incluso objetos tremendamente complejos) y utilizarlos de forma sencilla y cercana a la realidad, acelerando el proceso de desarrollo de la aplicación. Además, la modularidad e interoperabilidad innata de los objetos simplifican el mantenimiento de la aplicación y permiten al programador aprovechar su trabajo para otros muchos proyectos.

Caché está totalmente preparado para objetos, y proporciona toda la potencia de la tecnología de objetos a los desarrolladores de aplicaciones de proceso de transaccional de alto rendimiento.

ALMACENAMIENTO DE BASES DE DATOS DE OBJETOS

Como se mencionó anteriormente, el Servidor de datos multidimensional de Caché almacena todos los datos en forma de estructuras de datos multidimensionales. Gracias a su naturaleza multidimensional, estas estructuras se adaptan muy bien al almacenamiento de datos complejos.

Definiciones de almacenamiento y clases de almacenamiento

En Caché Objects, la forma en que se almacenan los datos y se accede a ellos se controla mediante una "Definición de almacenamiento" que utiliza una "Clase de almacenamiento". Una Definición de almacenamiento contiene una descripción de la estructura de base de datos para esa clase y especifica una Clase de almacenamiento. Una Clase de almacenamiento contiene código que genera los métodos Load, Save y Delete de una clase de objetos basándose en la Definición de almacenamiento de esa clase.

El uso de Clases de almacenamiento y de Definiciones de almacenamiento para controlar las estructuras de almacenamiento y cómo se guardan los datos es muy potente, pero si el programador no quiere preocuparse en controlar estas estructuras de almacenamiento y este mecanismo de archivo, Caché los generará automáticamente.

Con Caché se proporcionan tres Clases de almacenamiento

La opción más común y de más alto rendimiento es **CachéStorage**. Los desarrolladores definen los datos o importan una descripción de éstos y, durante la compilación, Caché generará las estructuras de datos multidimensionales necesarias y los métodos de acceso a la base de datos necesarios. La estructura de almacenamiento puede generarla totalmente el compilador o el programador puede, hasta cierto límite, diseñar la estructura de datos.

Los desarrolladores pueden crear también métodos de acceso **CustomStorage** específicos de la clase. En este caso existe un mapa de almacenamiento. Esta técnica es particularmente útil para acceder a estructuras de datos altamente especializadas o para almacenamiento que no sea en disco, pero no se podrá utilizar SQL sobre esa clase.

Una tercera opción es **CachéSQLStorage**. En este caso, Caché genera automáticamente los métodos de acceso a la base de datos con código SQL embebido. No es la forma más eficaz de implementar la persistencia, pero permite a una aplicación almacenar datos en una base de datos relacional.

Si estas tres clases no son suficientes, el programador también puede implementar su propia clase de almacenamiento.

No todas las clases tienen por qué utilizar la misma Clase de almacenamiento. La mayoría de las clases probablemente utilice CachéStorage, pero algunas pueden usar CustomStorage o CachéSQLStorage. La especificación de una Clase de almacenamiento se hace clase por clase.

El almacenamiento de Caché es incluso lo suficientemente flexible para permitir a las subclases utilizar Clases de almacenamiento diferentes. Por ejemplo, una clase Cliente podría ser una clase abstracta con una subclase que utilice CachéStorage y otra subclase que utilizase CachéSQLStorage. Esto permitiría acceder a algunos clientes en una base de datos relacional y a otros en una base de datos de Caché sin que el programador se preocupase sobre dónde está almacenado el cliente. Sin embargo, por lo general no es recomendable utilizar estas divisiones de subclases, porque dificultan la generación de consultas SQL.

Los nombres lógicos de las Clases de almacenamiento facilitan el soporte de varias ubicaciones

Una clase dada puede tener incluso más de una Definición de almacenamiento, y existen varias razones por las que sería deseable recompilar una clase utilizando una Definición de almacenamiento distinta. Un VAR puede tener varios clientes, cada uno de ellos con una estructura de almacenamiento ligeramente distinta, de modo que quizá sea necesario disponer de varias Definiciones de almacenamiento para la clase Factura utilizando CachéStorage.

Además, aunque CachéStorage ofrezca dar el mayor rendimiento, puede ser importante la capacidad de recompilar la aplicación para que se ejecute en otra base de datos (relacional).

Para que sea más sencillo recompilar una aplicación para ubicaciones diferentes, Caché soporta nombres lógicos para las Definiciones de almacenamiento. Basta con cambiar unos pocos valores de nombres lógicos y recompilar, y una aplicación completa se puede volver a compilar para utilizarla con una combinación distinta de estructuras de almacenamiento y Clases de almacenamiento.

Acceso a bases de datos relacionales con Caché Relational Gateway

Caché Relational Gateway permite que una solicitud SQL que se origine en Caché se envíe a otras bases de datos para su proceso. Utilizando el gateway, una aplicación de Caché puede recuperar y actualizar datos almacenados en bases de datos relacionales anteriores.

Además, si se compilan las estructuras de datos de Caché con CachéSQLStorage, el gateway permite a las aplicaciones de Caché utilizar bases de datos relacionales. La posibilidad de ejecutar aplicaciones de Caché sobre bases de datos relacionales anteriores proporciona flexibilidad adicional de instalación. Sin embargo, las aplicaciones se ejecutarán más rápidamente y serán más escalables si acceden a la base de datos post-relacional de Caché.

Las ventajas de Caché

Potencial de crecimiento
La Arquitectura de datos unificada de Caché describe automáticamente los datos como objetos y como tablas, lo que permite una integración transparente de las tecnologías relacional y de objetos. De esta forma, los programadores relacionales pueden ampliar sus conocimientos y aplicaciones existentes e ir introduciendo capacidades de objeto a medida que evolucionan los productos.

Compatibilidad con las herramientas existentes
Por medio de la Arquitectura de Datos Unificada, se puede acceder a todos los datos de Caché automáticamente mediante SQL. El acceso Caché SQL proporciona interoperatividad con aplicaciones relacionales convencionales, incluyendo muchas herramientas conocidas de análisis de datos y generación de informes.

Flexibilidad de implantación e independencia de la base de datos
Caché Relational Gateway permite que las aplicaciones de Caché se ejecuten sobre bases de datos relacionales convencionales, proporcionando opciones de implantación para clientes con grandes sistemas relacionales in *situ, y* para aquellos para los que el rendimiento no es un tema clave.

Gestión sencilla del sistema/Operación de la base de datos las 24 horas

Caché se ha diseñado para estar en ejecución las 24 del día, todos los días del año, en sistemas que van desde 4 usuarios hasta decenas de miles de usuarios. Muchas de las instalaciones las efectúan los VARs, que con frecuencia necesitan instalar grandes sistemas pero no pueden proporcionar un soporte continuado y extensivo de la gestión del sistema. Estos requerimientos hacen que se haga hincapié en la necesidad de simplicidad a la hora de dar soporte para el funcionamiento de grandes instalaciones.

En Caché no hay ninguna necesidad de realizar operaciones como la regeneración periódica de índices para mejorar el rendimiento o volver a cargar una base de datos cuando se instala una nueva versión, como suele ocurrir con algunos sistemas relacionales. Estas operaciones nunca son sencillas ni propicias para el funcionamiento las 24 horas.

Caché incluye un juego completo de utilidades de gestión del sistema, a las que se puede acceder de forma remota, y las principales funciones de gestión del sistema pueden programarse mediante scripts para efectuarlas en modo desatendido.

Copia de seguridad automática mientras se actualiza la base de datos

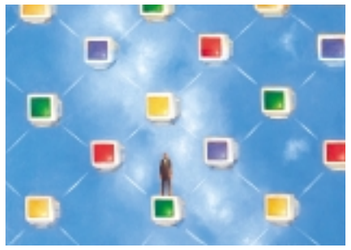
Caché soporta copias de seguridad completas, incrementales e incrementales acumulativas. Estas copias de seguridad pueden efectuarse mientras la base de datos está en funcionamiento, incluso cuando se están produciendo actualizaciones en la base de datos. Además pueden programarse mediante scripts para poder realizarlas automáticamente sin operador.

Los mapas de namespaces permiten efectuar cambios de configuración del hardware y de la base de datos sin tiempos de inactividad

En todos los sistemas, Caché mantiene “Mapas de namespaces” que especifican dónde se almacenan el código y los datos. Se pueden especificar en un Mapa de namespaces que informe de la ubicación de todos los Servidores de Aplicaciones y de datos de Caché.

Los cambios en el sistema, incluso cambios importantes como añadir de un servidor de base de datos o cambiar a un servidor de respaldo para que se puedan efectuar operaciones de servicio en el servidor de datos, pueden hacerse modificando el Mapa de namespaces maestro. Los cambios de configuración planificados se pueden realizar dinámicamente, sin tiempos de inactividad, de forma totalmente transparente para las aplicaciones.

Al crear Mapas de namespaces para contingencias, los administradores del sistema pueden cambiar la configuración de la base de datos para resolver toda una serie de escenarios de fallo.



Características de recuperación de fallos para mejorar la robustez en caso de fallo del hardware

Caché soporta muchas características avanzadas de recuperación de fallos para asegurar la robustez de las aplicaciones, incluyendo Servidores de duplicación y Clusters de base de datos.

La utilización de Servidores Shadow ofrece una potencia superior. Con la duplicación, cada servidor de datos tiene un servidor de respaldo que lee constantemente el registro diario del servidor principal y actualiza la base de datos del servidor de respaldo. Este servidor de respaldo es útil para diversos fines:

- En caso de que falle el servidor principal, el servidor de respaldo puede utilizarse inmediatamente, aunque se deshará cualquier transacción abierta.
- El servidor de respaldo puede utilizarse para los informes y las consultas. No se utiliza para actualizaciones cuando el servidor principal está en funcionamiento.
- Se pueden realizar las copias de seguridad de las bases de datos en el servidor de respaldo, descargando al servidor principal de esa tarea.

En el caso de que se pierda temporalmente la conexión entre los servidores, el servidor de respaldo se actualiza al restaurarse la conexión.

Los Clusters de base de datos soportan características automáticas de recuperación de fallos. En un cluster de bases de datos, varios ordenadores acceden a las mismas unidades de disco y utilizan software de gestión de clusters para coordinar el acceso compartido o exclusivo a bloques de disco. Si un ordenador falla, sus procesos se pierden, pero los otros ordenadores siguen funcionando. Las transacciones de usuario que estaban procesándose en el ordenador que han fallado se deshacen, y los usuarios podrán conectarse a otro ordenador. A menudo, se utiliza una distribución equilibrada de carga para asignar los usuarios dinámicamente a los ordenadores en cluster. Aunque los Clusters de base de datos proporcionan flexibilidad operativa y mayor fiabilidad, generalmente exigen más gestión del sistema que otros sistemas, y necesitan un hardware especial.

Caché ViewPoint para control y análisis del rendimiento

Caché ViewPoint es un conjunto de herramientas de análisis del rendimiento que se licencia separadamente y que permite a los desarrolladores supervisar, analizar y generar informes automáticamente sobre toda una serie de factores de rendimiento de las bases de datos y las aplicaciones de Caché, en tiempo real o histórico. Utiliza una arquitectura cliente/servidor para evitar la carga general de un host que esté analizando a un PC, reservando así los recursos del sistema para que los utilicen las aplicaciones.

Las ventajas de Caché

Mantenimiento simplificado
Con Caché, las aplicaciones no requieren mucha administración para seguir en funcionamiento a gran velocidad. En Caché no se efectúan operaciones como regeneración de índices ni otras similares de las bases de datos relacionales, y muchas funciones operativas pueden programarse mediante scripts para efectuarlas en modo desatendido.

Fiabilidad y funcionamiento las 24 horas
La copia de seguridad en línea, los Servidores Shadow y los Clusters de base de datos soportan el funcionamiento las 24 horas del día y la recuperación perfecta de fallos. Se pueden añadir clientes y servidores y volver a configurar las redes “sobre la marcha” utilizando el Mapeo dinámico de namespaces.

Rendimiento demostrable
Además de su clara utilización como herramienta de diagnóstico y optimización del rendimiento, Caché ViewPoint también permite a los desarrolladores realizar un seguimiento del rendimiento para asegurarse de que se cumplen los compromisos de rendimiento. En la actualidad, las medidas de rendimiento suelen forman parte de las especificaciones de las aplicaciones. Caché ViewPoint permite a los desarrolladores probar que sus aplicaciones tienen un rendimiento que cumple o supera dichas medidas.

Capítulo 3: El Servidor de aplicaciones Caché

El Servidor de aplicaciones de Caché es una capa de código que ofrece posibilidades avanzadas de programación de objetos, sofisticado almacenamiento de datos en memoria intermedia cache y fácil acceso a diversas tecnologías. El Servidor de Aplicaciones de Caché permite desarrollar rápidamente aplicaciones de base de datos sofisticadas, obtener alto rendimiento al operar con ellas y darles soporte con facilidad.

De forma específica, el Servidor de Aplicaciones de Caché proporciona:

- Acceso a los Servidores de datos multidimensionales de Caché el mismo u otros ordenadores.
- Acceso a bases de datos relacionales, aunque con menor rendimiento que con el acceso a los Servidores de Datos de Caché.
- El lenguaje de programación Caché ObjectScript.
- Software de conectividad con almacenamiento con la memoria intermedia cache en la parte cliente, para permitir el acceso rápido a Caché Objects desde toda una serie de tecnologías, incluyendo Java, C++, ActiveX, Visual Basic, Delphi y CORBA. Caché se encarga automáticamente de los procesos en red, si es necesario.
- Software de conectividad para acceso SQL a la base de datos de Caché mediante ODBC y JDBC y OCI, incluyendo almacenamiento sofisticado en memoria intermedia cache en las capas cliente y del servidor de aplicaciones para conseguir mayor rendimiento y la posibilidad de combinar automáticamente datos de varios Servidores de Datos de Caché en una sola consulta.
- Caché Server Pages para crear aplicaciones Web de alto rendimiento y fáciles de programar.
- Las herramientas de desarrollo Caché Object Architect, Caché Studio y Visual Caché, para desarrollar y depurar rápidamente aplicaciones con Caché.
- Almacenamiento en memoria intermedia del código de Caché ObjectScript de modo que el código no tiene que instalarse en más de un ordenador. Los cambios en el código se propagan automáticamente a todos los servidores de aplicaciones.
- Caché efectúa automáticamente todos los procesos de red, de modo que el código de la aplicación no dependa de la ubicación de los datos, tanto si los datos están en el mismo ordenador como en uno remoto.

Cómo accede el Servidor de Aplicaciones Caché al Servidor de Datos Caché

El Servidor de aplicaciones de Caché está completamente integrado con el Servidor de datos Multidimensional de Caché.

No es necesario que Servidores de Aplicaciones y de Datos de Caché estén en el mismo ordenador. Si están en el mismo ordenador, los procesos de las aplicaciones llaman directamente al código del Servidor de Datos para acceder a la base de datos utilizando un buffer en memoria cache de disco compartido.

Cuando el Servidor de Aplicaciones necesita acceder a un Servidor de Datos Caché remoto, se utiliza el Protocolo de Memoria Cache Distribuida (DCP, Distributed Caché Protocol) de Caché. DCP accede a la base de datos remota y almacena los datos en memoria intermedia cache en el Servidor de Aplicaciones, de modo que las solicitudes futuras de datos pueden a menudo resolverse sin utilizar la red.

Caché Object Architect

Las Clases de Caché Object suelen crearse, editarse y compilarse con Caché Object Architect. Utilizando Object Architect, los desarrolladores pueden especificar las propiedades, codificar los métodos de los objetos y definir tipos de datos especializados.

Si se desea, Object Architect también se puede utilizar para especificar la estructura de datos para almacenamiento (o Caché elegirá una) y características especiales de SQL como triggers SQL e información de optimización de consultas.

Importar/exportación de modelos de datos

Además de utilizar Object Architect, hay varias formas de importar y exportar definiciones de clases a y desde el Diccionario de datos de Caché, respectivamente.

Caché RoseLink permite definir clases utilizando la herramienta de creación de modelos de objetos Rose de Rational Software para importarlas en Caché. De forma similar, las definiciones de clase de Caché se pueden exportar a Rose para utilizarlas dentro del entorno de creación de modelos de Rose.

Caché también puede crear objetos a partir de archivos relacionales DDL. Las clases de objetos resultantes serán muy sencillas: sus propiedades serán tipos de datos definidos por el sistema con un solo valor que se corresponderán con los campos de la tabla relacional, y sus únicos métodos serán los métodos persistentes necesarios para mover los datos a disco y desde el disco. Sin embargo, incluso estas clases simples están inmediatamente disponibles para su utilización con lenguajes de programación de objetos, y se pueden utilizar como bloques de construcción de modelos de datos más complejos.

CDL es el lenguaje de definición de clases de InterSystems. Un archivo CDL es un archivo de texto que contiene toda la información necesaria para definir una clase de Caché Object, y proporciona otros medios para transportar las definiciones de clases desde una aplicación a otra.

Caché Studio

Las rutinas también se pueden editar y compilar utilizando Caché Studio. Caché Studio resulta particularmente útil en la depuración y la edición del código que no se haya creado mediante Object Architect.

Las ventajas de Caché

Desarrollo rápido de aplicaciones
Caché Object Architect y Caché Studio son herramientas muy productivas para crear Caché Objects y rutinas de Caché ObjectScript. Al soportar conceptos de programación de objetos, se facilita el desarrollo rápido ya que permite a los desarrolladores crear modelos de datos complejos de forma natural.

Opciones para creación de modelos de aplicaciones
Caché permite a los desarrolladores modelar sus aplicaciones del modo que más sentido tenga para ellos. Las estructuras de datos multidimensionales se pueden crear directamente en Caché ObjectScript. Las definiciones de clase pueden crearse con Caché Object Architect o importándolas de la herramienta de creación de modelos de objetos Rose de Rational. Caché acepta incluso modelos relacionales en forma de archivos DDL.

Flexibilidad
Caché ObjectScript ofrece a los desarrolladores la flexibilidad de utilizar cualquiera o todos los modos de acceso a los datos de Caché:a objetos, SQL o acceso global directo dentro de la misma aplicación y la misma rutina e incluso la misma línea de código. Los desarrolladores puede adaptar el acceso a los datos para adecuarlo a las necesidades de la aplicación.

Integración de Caché con otras tecnologías de objetos

Utilización de Caché Objects con otras tecnologías

Caché soporta varias tecnologías de programación para crear la lógica de la aplicación y el interface de usuario. Los **Servidores Caché Object** interpretan las clases de Caché Objects como clases ActiveX, Java o C++, en función del servidor Caché objects que se utilice. Los objetos de Caché también se pueden utilizar con CORBA.

Estos servidores de objetos disponen de una capa de software en la parte cliente, incluyendo almacenamiento en memoria intermedia cache para obtener un mayor rendimiento, y una capa en el Servidor de Aplicaciones. Es transparente para el programador, tanto si están en el mismo ordenador como en ordenadores distintos. Además, es transparente para el cliente en lo que ese refiere al Servidor de Datos Caché que contiene los datos, o incluso si los datos del objeto están almacenados en una base de datos relacional a la que se accede a través del Servidor de aplicaciones Caché.

Normalmente, un cliente está conectado sólo a un Servidor de Aplicaciones de Caché, y dicho servidor es responsable de obtener los datos del Servidor de Datos correcto.

ActiveX

El servidor Caché Object para ActiveX muestra las clases de objetos de Caché como clases de ActiveX para utilizarlas con herramientas como Visual Basic, Delphi de Inprise y cualquier software compatible con la interfaz COM.

Soporte de Java

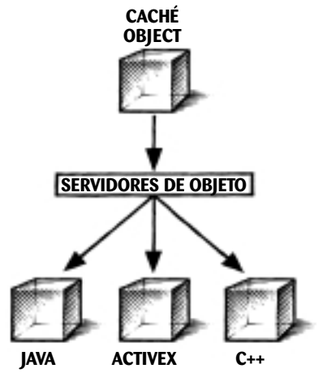
El servidor Caché Object para Java muestra las clases Caché como Java Beans.

Soporte de C++

Caché ObjectServer para C++ crear interpretaciones C++ de las clases Caché. Además, una vez definidas y compiladas las clases Caché Object, pueden generarse los archivos ODL (Lenguaje de Definición de Objetos), que permiten rellenar la librería de tipos de una aplicación C++.

Soporte para CORBA

Se puede acceder a los objetos de Caché mediante CORBA y estos pueden acceder a otros objetos compatibles con CORBA. La utilización de CORBA puede no dar el mismo nivel de rendimiento que se obtendría con otras tecnologías de objeto.



Los objetos de Caché pueden presentarse a aplicaciones y herramientas de desarrollo como Objetos Java, Activex o C++.

Visual Caché

Como complemento del Servidor Caché Object para ActiveX, **Visual Caché** proporciona un enlace entre Caché Objects y Visual Basic para desarrollo rápido. Visual Caché incluye **Caché Object Link Control (Control de Enlace con Caché Objects)**, que automatiza el enlace entre una clase Caché y los controles de la interfaz de usuario utilizados para mostrar la clase en un formulario VB. El Control soporta la interfaz de control de datos de VB, de modo que se puede conectar a un control con información de datos con sólo especificar su Data Source (una clase Caché) y su Data Field (el atributo de clase Caché). El resultado: Cualquiera que sepa Visual Basic puede utilizarlo para crear la interfaz de usuario para una aplicación que emplee Caché Objects, incluso si no saben nada en absoluto sobre la tecnología de objetos.

Para facilitar aún más las cosas, Visual Caché también incluye **el Caché Wizard Form (Asistente de formularios de Caché)**, que crea automáticamente formularios en Visual Basic. Los desarrolladores sólo tienen que identificar la clase y los atributos que desean incluir y el Asistente genera un formulario listo para ejecutarse que puede usarse tal cual o personalizarse con VB.

Visual Caché se suele utilizar para aplicaciones sencillas. Las aplicaciones de VB más complejas suelen programarse utilizando las capacidades de acceso a objetos de VB para acceder a Caché Objects.

"Estamos muy impresionados con el nuevo sistema de reserva que nos ha proporcionado Innsite [unVAR de InterSystems. Para cumplir nuestros criterios, Innsite tenía que proporcionarnos un sistema que fuera tan rápido y sencillo de instalar como de utilizar; hicieron esto y mucho más dentro de un periodo de diez semanas."

Stephen Thirlwell
Director de marca
Welcome Lodges

Las ventajas de Caché

Libertad de elección

Caché permite a los programadores utilizar cualquiera de los entornos de desarrollo que elijan para la programación UI (Interfaz de Usuario) y las reglas de negocio. Caché incluye unas características especiales que proporcionan enlaces para conseguir mayor productividad con algunas de las herramientas más populares.

Mediante los servidores Caché Objects y gateways, las aplicaciones basadas en Caché pueden interactuar con otros programas utilizando objetos ActiveX, Java o C++.

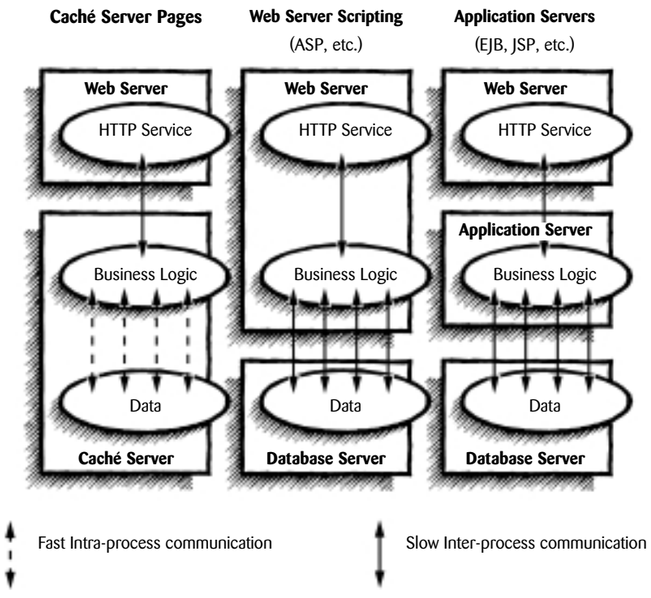
Los usuarios de Visual Basic pueden utilizar Visual Caché Object Control Link (Control de Enlace con Objetos de Caché) y el Caché Wizard (Asistente de Formularios Caché) para ser productivos al instante. InterSystems ha llegado a acuerdos de colaboración con otros proveedores importantes de herramientas, lo que facilita el uso de Caché con herramientas como Delphi de Inprise, y Visual Café de Symantec.

Conectividad Web (ver también: Capítulo 4)

Para la conectividad Web, Caché emplea el concepto de páginas de servidor, es decir, las páginas se crean y llenan con datos "sobre la marcha" a medida que las solicita un navegador. Pero a diferencia de otras arquitecturas Web, las **Caché Server Pages** se ejecutan en el servidor de datos de Caché, junto a los datos que utilizan. La conexión con el servidor Web se realiza utilizando APIs rápidos estándares.

Comparación de Arquitecturas Web

Las aplicaciones basadas en Caché son más rápidas y altamente escalables porque se ejecutan en el servidor de la Base de Datos Caché



Las Caché Server Pages están realizadas en HTML o XML estándar, así que se pueden crear o modificar fácilmente utilizando cualquier herramienta predefinida de creación de páginas Web o su editor de textos favorito. La funcionalidad se añade incorporando Caché Application Tags, CATs (Marcadores de Aplicaciones Caché) o Hyper-Events™.

Caché Application Tags

Caché Applications Tags (CATs) son como etiquetas HTML, excepto en que en lugar de dar formato al texto, ejecutan funciones en el servidor de datos Caché y/o en el navegador. Los Application Tags que acompañan a Caché se pueden utilizar para leer y escribir en la base de datos, realizar cálculos, bucles, actuar como contadores, gestionar la coordinación de múltiples frames, etc. Lo mejor de todo es que las CATs son ampliables. Los desarrolladores pueden crear las suyas propias para adaptarlas a las necesidades específicas de sus aplicaciones.

Eventos de Servidor

Los eventos del servidor permiten que los eventos que suceden en el navegador (clics del ratón, movimientos del ratón, cambios de valor de los campos, final de tiempos de espera, etc.) llamen a operaciones del servidor y actualicen la página actual sin redibujarla. Incluyendo los Hyper-Events, las e-aplicaciones son mucho más interactivas y con mayor capacidad de respuesta.

Gestión de sesiones simplificada

Uno de los retos a los que se enfrentan los desarrolladores de aplicaciones de Internet es la naturaleza carente de información de estado propia de la Web. Los desarrolladores generalmente tienen que utilizar gran cantidad de código a nivel del sistema para hacerse cargo de los problemas de gestión de sesiones que surgen cuando las aplicaciones necesitan mantener el estado entre páginas Web. Caché facilita la gestión de estados porque proporciona objetos especiales de gestión de sesiones. En ellos se encapsula todo el código a nivel de sistema, de forma completamente transparente para los desarrolladores. Los objetos de gestión de sesiones trabajan tanto con HTTP como con HTTPS (para transacciones seguras). Caché también admite procesos de sesión dedicados, para aquellas aplicaciones que requieran bloqueos de múltiples páginas de la base de datos.

"En el área de aplicaciones basadas en la Web, no hay punto de comparación entre Caché y otras tecnologías de base de datos que hemos utilizado. Caché las supera a todas."

Uwe Wagner
Cofundador
Schmidt & Wagner

"Caché es un producto bien diseñado que deberían considerar seriamente las organizaciones de desarrollo de aplicaciones Web y empresariales con complejos requisitos de proceso de transacciones".

International Data Corporation

Las ventajas de Caché

Aplicaciones Web rápidas y escalables

La tecnología Web de Caché aporta el rendimiento superior y la escalabilidad masiva de la base de datos de Caché para las Intranets y la red más grande del mundo: Internet. Caché Server Pages proporciona una conexión de alto rendimiento entre la base de datos post-relacional de Caché y un servidor Web, utilizando interfaces estándar. Las Caché Server Pages se ejecutan en el servidor de datos de Caché e interactúan con la base de datos mediante comunicaciones rápidas entre los procesos, de modo que el rendimiento mejora. Y la escalabilidad se amplía porque el servidor Web ya no está saturado de procesos de consumo intensivo de recursos.

Desarrollo Web rápido y simplificado

Los desarrolladores pueden utilizar cualquier editor de texto o sus herramientas predefinidas de creación de páginas Web para crear Caché Server Pages. Caché Applications Tags (CATs) favorece la reutilización del código y el desarrollo en colaboración; unos desarrolladores crearán Caché Applications Tags (CATs) que necesitan las aplicaciones y otros utilizarán esas CATs para crear sofisticadas aplicaciones Web. Caché también reduce el tiempo de desarrollo simplificando la gestión de sesiones.

Caché ObjectScript

Caché ObjectScript es un potente lenguaje de programación orientado a objetos diseñado para el desarrollo rápido de aplicaciones de base de datos. A continuación se describen algunas de las características clave de este lenguaje.

Estructura general

Caché ObjectScript está orientado a los comandos; por tanto su sintaxis es similar a:

```
set x=a+b
do rotate(a,3)
if x>3
```

Existe un conjunto de funciones de sistema integradas, cuyos nombres empiezan con el carácter '$'(para distinguirlos de los nombres de variables y de arrays) que son especialmente potentes para el trabajo con texto. Incluyen funciones como:

```
$extract(cadena,desde,hasta) // extrae un conjunto de caracteres
                             de una cadena

$length(cadena)              // determina la longitud de una cadena
```

Las expresiones utilizan la precedencia de operadores de izquierda a derecha, al igual que la mayoría de las calculadoras de mano, excepto en los casos en que los paréntesis cambian el orden de evaluación.

Llamadas al código

Las llamadas al código se realizan generalmente utilizando el comando DO; por ejemplo:

```
do rotate(a,3)
```

Al código que devuelve un valor también se le puede llamar como una función; por ejemplo:

```
set x=a+$$insertar(3,y)
```

llama a la subrutina 'insertar' escrita por el programador.

También se puede llamar al código como método de objeto; por ejemplo:

```
set dinero=factura.Total()  // Total() devuelve el importe total
                            de la factura
```

o bien

```
do serie.Incremento()       // Incremento() no devuelve ningún
                            valor de interés
```

Se admiten las llamadas por valor como por referencia para los parámetros.

Almacenamiento flexible de datos

Una de las características más exclusivas de Caché ObjectScript es el alto grado de flexibilidad y dinamismo. Los datos se pueden almacenar en:

- propiedades de objetos.
- variables.
- sparse arrays multidimensionales que permitan cualquier tipo de datos para los subíndices.
- archivos de base de datos ("globals") que se asemejan a los sparse arrays multidimensionales.

Con raras excepciones, en cualquier parte del lenguaje en la que se puedan utilizar variables se pueden utilizar arrays, propiedades de objeto o referencias a global.

En la mayoría de los lenguajes informáticos, los tipos de datos son una extensión de los conceptos de almacenamiento del hardware (integer, float, char, etc.). Sin embargo, Caché ObjectScript asume la filosofía de que las personas no piensan utilizando esos tipos de almacenamiento y que esos tipos de datos "centrados en la informática" son un impedimento para el desarrollo rápido de aplicaciones. La necesidad de esas declaraciones y sentencias de dimensión provoca muchos más errores de los que ayuda a prevenir (errores como desbordamiento de entero de 2 bytes, o cuando una cadena sobrepasa su asignación de memoria y daña otra variable). Sin embargo, la creación de tipos de objeto como Persona, Factura, Animal, Coche, etc., son muy apreciados y coherentes con el modo en que piensan los seres humanos.

Por eso, en Caché ObjectScript las propiedades de objetos son muy dependientes de los tipos, pero los otros tres tipos de almacenamiento (variables, arrays y nodos globales) son entidades polimórficas e independientes de los tipos y no necesitan ser declaradas ni definidas. Simplemente surgen cuando se las utiliza, y se modelan a sí mismas según las necesidades de los datos que almacenan y el modo en que se utilizan en una expresión. Ni siquiera las arrays necesitan una especificación de tamaño, ni de dimensión, ni de tipos de subíndices o datos. Por ejemplo, un desarrollador podría crear una array denominada Persona estableciendo simplemente lo siguiente:

```
set Persona("Sánchez","Luis")="Soy una buena
persona"
```

En este ejemplo, los datos se han almacenado en una array bidimensional que utiliza datos de cadena como subíndices. Otros nodos de datos de esta array podrían tener un número distinto de dimensiones y entremezclar string, integer u otros tipos de datos como subíndices. Por ejemplo, uno podría almacenar datos en:

```
abc(3)
abc(3,-45.6,"Sí")
abc("Contar")
```

todos en la misma array.

Acceso directo a la base de datos

Una referencia directa a la base de datos (una "referencia a global") es en esencia una referencia de array multidimensional precedida por el carácter del acento circunflejo '^'. Ese carácter indica que es una referencia a los datos almacenados en la base de datos en lugar de datos privados de proceso temporal. Cada array de la base de datos se denomina "global".

Al igual que con las arrays multidimensionales y las variables, no se necesitan declaraciones, definiciones ni reservas de almacenamiento para acceder a los datos o almacenarlos en la base de datos; los datos del global comienzan a existir desde el momento en que se almacenan. Por ejemplo, para almacenar información en la base de datos, se podría escribir:

```
set ^Persona("Sánchez","Luis")="Soy muy buena persona"
```

y más tarde recuperarla mediante un código como el siguiente:

```
set x=^Persona("Sánchez","Luis")
```

El programador tiene completa flexibilidad para estructurar estas arrays de datos de globales. De este modo, los datos de la factura podrían almacenarse como:

```
^Factura(n° factura,"Cliente") = Información del cliente

^Factura(n° factura,"Fecha") = Fecha factura

^Factura(n° factura,"Artículos") = n° de artículos de la factura

^Factura(n° factura,"Artículos",1,"Serie") = n° de serie del primer
artículo

^Factura(n° factura,"Artículos",1,"Cantidad") = cantidad del primer
artículo

^Factura(n° factura,"Artículos",2,"Serie") = n° de serie del
segundo artículo
```

etc.

Referencias a objetos

Caché Objects implementa el modelo de datos ODMG, con potentes extensiones.

En Caché ObjectScript se utiliza una "oref" (referencia a objeto) para acceder a un objeto ("oref" es generalmente una variable cuyo valor especifica a qué objeto de la memoria se está haciendo referencia). A "oref" le sigue un punto y el nombre de una propiedad o método. Las referencias a objetos pueden utilizarse siempre que se pueda utilizar una expresión. Por ejemplo:

```
set nombre=persona.Nombre   // 'persona' es una variable cuyo valor
                            es una oref

                            // el nombre de la persona se incluye
                            en la variable 'Nombre'

if persona.Edad>x           // comprobar si la edad de la persona
                            es mayor que 'x'

set dinero=factura.Total()  // 'Total()' es un método que calcula
                            la suma de

                            // todos los artículos de la factura
```

Los métodos también se pueden ejecutar con un comando DO, cuando no se precisa un valor de retorno. Por ejemplo:

```
do serie.Incremento()       // Incremento() es un método cuyo valor
                            de devolución

                            // si existe, no es de interés
```

La "oref" no es lo mismo que un id de objeto de la base de datos. El id de objeto es un valor asociado de forma permanente al objeto de la base de datos; se utiliza para recuperar y almacenar un objeto de base de datos. Una vez que un objeto está en la memoria, se le asigna, como a todos los objetos de la memoria, un valor oref reutilizable que después se utilizará para acceder a los datos del objeto. La próxima ocasión en que el mismo objeto de la base de datos vuelva a la memoria, probablemente se le asignará un valor oref distinto.

Modelos diferentes de acceder a la base de datos

Los Objetos Caché que son persistentes generan código que los puede almacenar y recuperar de la base de datos como una serie de uno o más nodos globals. Además, colocan automáticamente una descripción de sí mismos en el diccionario de datos de SQL, de modo que el acceso SQL también está permitido. El lenguaje de consulta para Caché Objects es Caché SQL, un SQL mejorado para objetos.

Por tanto, hay tres formas de acceder a las base de datos integrada:

- Sintaxis de base de datos de objetos.
- Caché SQL, incluyendo acceso ODBC y JDBC para herramientas de uso común.
- Acceso mediante referencia directa a globales de arrays multidimensionales.

Acceso HTML y SQL

HTML para aplicaciones Web y SQL pueden estar embebidos en el código de Caché ObjectScript.

Rutinas

En algunos lenguajes de objetos, todo el código debe ser parte de algún método. Caché ObjectScript no tiene esa restricción. Parte del código se llama mediante métodos, y el resto del código se llama mediante subrutinas.

El código Caché ObjectScript está organizado en un conjunto de "rutinas". Cada rutina (generalmente de hasta 32 KB de tamaño) es un todo, en el sentido de que puede editarse, almacenarse, compilarse y ejecutarse de forma independiente. Las rutinas se enlazan dinámicamente en tiempo de ejecución; el programador no tiene que efectuar un paso adicional de enlace. El código de rutina se almacena en la base de datos; así, las rutinas pueden llamarse de forma dinámica a través de la red en lugar de tener que estar instaladas en cada ordenador.

En una rutina, el código está organizado como un conjunto de subrutinas. Para llamar a una subrutina, el nombre de la subrutina debería tener al final el nombre de la rutina. Por ejemplo,

```
do admit^pat3()        ; llama a la subrutina 'admit', que está en
                         la rutina 'pat3'
```

Al llamar a código que está en la misma rutina, sólo se necesita el nombre de la subrutina.

```
do discharge()         ; llama a la subrutina 'discharge', que está
                         en la misma rutina
```

Si la subrutina devuelve un valor que es de interés, se debe llamar a la subrutina con la función "$$" de la sintaxis. Por ejemplo:

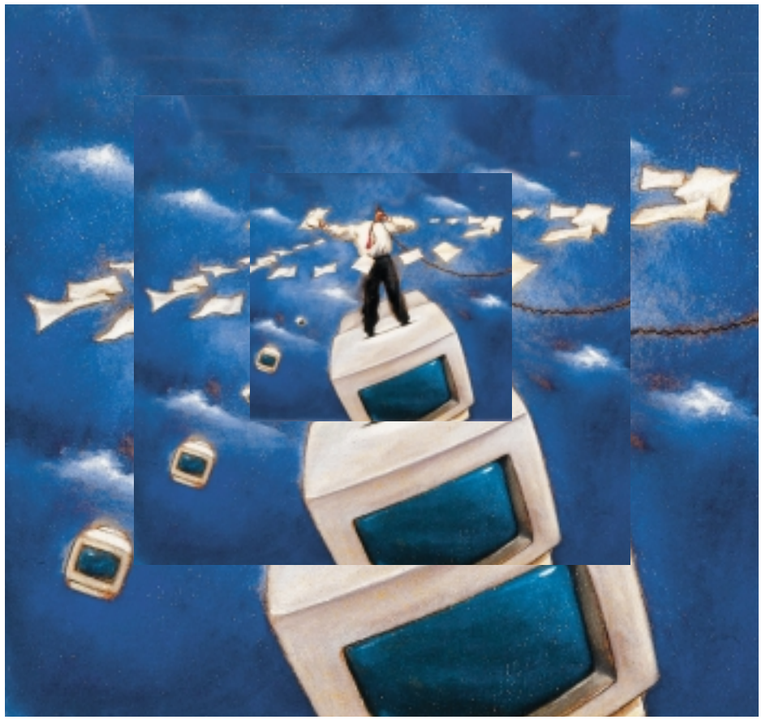
```
Set x=$$admitir^pat3(); la subrutina 'admitir' que está en la
                        rutina 'pat3'

Set y=$$descargar()    ; la subrutina 'descargar que está en la
                         misma rutina
```

Las rutinas pueden editarse y compilarse utilizando Caché Studio.

La definición de los objetos y la edición de sus métodos se realiza en Caché Object Architect (aunque también se pueden importar mediante CDL). Estas definiciones y el código de los métodos se almacenan en archivo de datos globales, y Class Compiler utiliza estas definiciones para generar un conjunto de rutinas. Por ejemplo, la compilación de la clase Paciente podría producir un conjunto de rutinas denominadas ooPacienteR1, ooPacienteR2, etc., que incluyen todo el código de método de Paciente. Los métodos son simplemente subrutinas de esa rutina, y el sistema sabe cómo enviar una llamada a un método hacia la subrutina apropiada dentro la rutina apropiada.

Además de la versión en tiempo de ejecución de una rutina, existe generalmente una versión en código fuente. Las utilidades emplean sufijos para distinguir entre el código del objeto y las versiones de código fuente: a).MAC hace referencia al código fuente de macro; b) INT indica el código fuente intermedia tras la ejecución de preprocesador de macros; y c) OBJ es la versión de ejecución compilada. Si un desarrollador escribe una rutina denominada "Admit", la fuente de la macro estará en Admit.Mac, que se compila mediante el preproceso de la macro en Admit.Int, donde se sustituyen todas las macros, y finalmente Admit.Obj, que es el código de ejecución. Los sufijos sólo los emplean las utilidades, no el código del programa; nunca se programa una llamada a subrutina como "do sub^routine.obj".



"Al Ministerio de Sanidad de Ontario le gustó tanto la aplicación que expandieron el proyecto para incluir 170 hospitales. Nunca nos preocupó el problema de la escalabilidad. Con Caché, sabíamos que la aplicación se ejecutaría bien sin importar el tamaño que alcanzara ...Tenemos experiencia en tecnología relacional, incluyendo Oracle y DB2, y ninguna otra tecnología de base de datos que conozcamos es comparable a Caché."

Joseph Scaglione
Copresidente
Rincon Technologies, Inc.

Rendimiento escalable en sistemas distribuidos

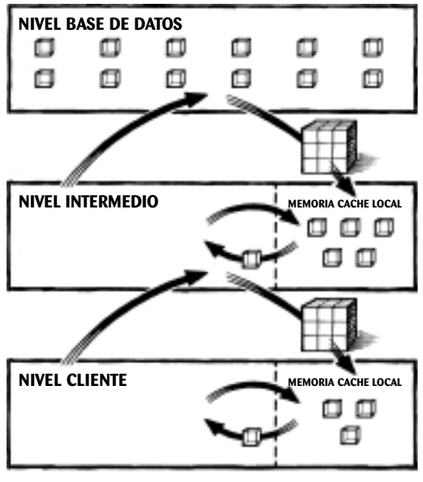
A menudo, en las arquitecturas distribuidas, la red en sí misma (el volumen de tráfico que debe gestionar y la ruta entre los clientes y los datos que solicitan) desempeña un papel importante en el rendimiento de las aplicaciones de proceso transaccional. La clave del alto rendimiento es almacenar información en memoria intermedia de cache, donde el cliente puede alcanzarla fácilmente.

Protocolo de Memoria Cache Distribuida

Caché incluye el exclusivo Protocolo de cache distribuido de (DCP) de InterSystems, que reduce drásticamente el tráfico de los sistemas en red.

DCP es eficaz porque los datos se transportan en paquetes. Cuando se solicita información a través de la red, el paquete de datos de respuesta incluye además de los datos deseados, datos adicionales relacionados. La relaciones de datos naturales inherentes a los objetos y al modelo de datos multidimensional de Caché permiten identificar e incluir información relacionada con los datos originales solicitados. Esta información "asociada" se almacena localmente en memoria intermedia cache en el cliente o en el servidor de aplicación. Normalmente, las solicitudes posteriores de datos pueden resolverse con los datos de la memoria de cache local, evitando otro viaje a través de la red la red. Si el cliente cambia datos, sólo las actualizaciones vuelven a enviarse al servidor de bases de datos.

El Protocolo de Memoria Cache Distribuida puede reducir radicalmente el tráfico de red. El rendimiento mejora porque, para la mayoría de las solicitudes, los clientes utilizan los objetos y datos almacenados en memoria cache local, y un Servidor de Datos puede dar soporte a más usuarios y a más servidores de aplicaciones.



Con DCP, las peticiones a través de la red son respondidas incluyendo más datos de los solicitados generando memorias cache locales. Los objetos y los datos almacenados en memoria cache local en el cliente y en los servidores de Nivel Intermedio, se utilizan a menudo para responder a posteriores peticiones asociadas.

Las ventajas de Caché

Rendimiento

El Protocolo de Memoria Cache Distribuida exclusivo de InterSystems reduce drásticamente el tráfico de red, lo que significa que las transacciones pueden completarse más rápidamente. Además, la mayor parte del tiempo, los clientes utilizan los objetos y datos almacenados en memoria cache local, lo que optimiza aún más el rendimiento.

Escalabilidad

Gracias a DCP, las aplicaciones basadas en Caché se pueden escalar a miles de usuarios sin sacrificar el rendimiento. El mapeo dinámico de namespaces permite expandir las redes de forma totalmente transparente para las aplicaciones que se ejecutan sobre ellas

"Caché supera significativamente en rendimiento y en nivel a sus rivales."

Bloor Research

Implantación y mantenimiento de una aplicación.

Plataformas de hardware y sistemas operativos

Las aplicaciones Caché se ejecutan generalmente sin ninguna modificación sobre una amplia gama de plataformas de hardware y sistemas operativos, incluyendo Windows 9x/ME/2000/NT, todas las plataformas importantes de UNIX, OpenVMS y Linux.

Configuraciones de hardware

Una aplicación Caché se puede implantar en configuraciones de hardware de uno, dos o tres niveles, o igual-a-igual, sin programar ni compilar. En sistemas de gran tamaño, puede haber varios Servidores de Aplicaciones Caché Application Servers, así como varios Servidores de Datos multidimensionales de Caché, pero en general, la carga de proceso es mucho menor cuando el Servidor de Aplicaciones accede a datos que están en el mismo ordenador que el Servidor de Datos.

Normalmente, un cliente conecta con un solo Servidor de Aplicaciones de Caché, y es responsabilidad de ese Servidor de Aplicaciones obtener los datos del Servidor de datos correcto.

Las grandes configuraciones quizá necesiten un ordenador que actúe principalmente como Servidor de Datos y varios ordenadores que sean los Servidores de aplicaciones. Si es posible, en esta arquitectura de tres niveles, se deberá limitar el Servidor de Datos a un solo ordenador para evitar la carga general asociada a la coordinación de varios servidores de datos.

Otra arquitectura que se utiliza habitualmente con Caché es la arquitectura igual-a-igual, en la que dos (o más) ordenadores actúan tanto de Servidor de Aplicaciones como de Servidor de Datos. Esta arquitectura resulta especialmente funcional cuando cada ordenador da servicio a un grupo de usuarios que acceden sobre todo a los datos almacenados en ese ordenador, pero que ocasionalmente necesitan acceder a los datos de otro. Por ejemplo, en un hospital, un ordenador se puede dedicar principalmente al laboratorio y otro a los trabajos administrativos, pero ocasionalmente necesitarían compartir datos.

Caché también admite clusters hardware, en los que varios ordenadores comparten acceso a las mismas unidades de disco y coordinan el acceso compartido y exclusivo a los bloques de disco. Esta arquitectura proporciona una fiabilidad y un rendimiento superior que un solo ordenador, pero la gestión y las operaciones del sistema suelen ser más complicadas. Igual que en una red igual-a-igual, el rendimiento óptimo de este entorno se consigue cuando ordenadores diferentes se ocupan de aplicaciones distintas, reduciendo los conflictos entre ordenadores en el acceso a los mismos bloques de disco.

Los servidores de respaldo(Shadow Servers) son también bastante populares, y constituyen un buen modo de mejorar la fiabilidad, a la vez que se descarga de algunas actividades de generación de informes y consultas.

Estas son las arquitecturas principales que pueden combinarse para formar un amplio rango de arquitecturas hardware.

Volver a instalar una aplicación Caché con una base de datos relacional

Gracias a la Arquitectura de datos unificada y a SQL Gateway de Caché, una aplicación Caché se puede compilar para ejecutarla sobre una base de datos relacional. Aunque el rendimiento resultante no será tan óptimo, esto representa una opción de implantación importante para VARs que necesitan ofrecer una solución relacional para que nuevos clientes potenciales les tomen en consideración.

Instalación de una aplicación e introducción de cambios en un sistema en funcionamiento

Una vez que se ha escrito una aplicación, esta debe instalarse y modificarse posteriormente. Con miles de PCs y varios servidores, esto puede ser un problema real con la mayoría de las tecnologías.

Caché simplifica la instalación del software de las aplicaciones. El código de Caché ObjectScript sólo necesita instalarse en un único Servidor de datos. Los otros ordenadores obtienen las copias utilizando DCP y almacenándolo la memoria cache.

Para introducir un cambio en una rutina Caché ObjectScript en un sistema en funcionamiento, basta con cargar la rutina con el código fuente modificado en el Servidor de datos donde reside el código. El código fuente se compilará automáticamente y se notificará al Servidor de aplicaciones que necesita volver a cargar la rutina modificada. Por supuesto, esos cambios deben introducirse con precaución; un proceso que hiciese una llamada desde la versión anterior a otra rutina obtendrá un error al intentar volver a la rutina modificada.

Operación las 24 horas

Caché se ha diseñado para ejecutarse las 24 del día, todos los días del año, y se utiliza de ese modo en muchas ubicaciones con miles de usuarios. Caché soporta dicho funcionamiento exigiendo una mínima gestión operativa y del sistema. Consulte “Gestión sencilla del sistema/funcionamiento de la base de datos las 24 horas”.

Capítulo 4: Creación de aplicaciones Web rápidas a gran velocidad – Caché Objects para la Web

"Creación de aplicaciones Web rápidas a gran velocidad" incide doblemente en la rapidez. Eso es porque es posible crear sofisticadas aplicaciones Web orientadas a base de datos con CSP más rápidamente que con cualquier solución tradicional, y además porque la base de datos incorporada de Caché es la más rápida del mundo, capaz de dar soporte a sistemas con decenas de miles de usuarios simultáneos.

Hay muchas formas de escribir aplicaciones Web con Caché, incluyendo todos los métodos tradicionales que utilizan SQL para acceder a la base de datos. En este capítulo vamos a describir otra solución más directa denominada Caché Server Pages ("CSP").

CSP es una tecnología que se proporciona como parte del Servidor de aplicaciones Caché. Es el medio principal por el que las aplicaciones escritas en Caché se desarrollan e interactúan con la Web, y proporciona:

■ Una solución avanzada de desarrollo orientado a objetos para aplicaciones de base de datos, y

■ Rendimiento y escalabilidad muy altos en tiempo de ejecución.

CSP soporta HTML, XML, WML y otros "lenguajes de marcas" orientados a la Web.

CSP no es una herramienta de diseño, aunque se puede utilizar con ellas. Mientras las herramientas de diseño Web generalmente se concentran sólo en la producción de HTML estático, CSP va más allá del aspecto de las páginas para facilitar el desarrollo de la lógica de la aplicación. También proporciona la configuración que permite la ejecución del código en el Servidor de aplicaciones Caché.

CSP da soporte a un potente entorno de programación por procedimientos, de modo que las aplicaciones se pueden escribir con un nivel de sofisticación y precisión que supera las posibilidades de las tecnologías centradas en la generación de aplicaciones. Sin embargo, también soporta el desarrollo rápido mediante su arquitectura de clases, que produce "bloques de construcción" de código que se pueden combinar y que, mediante el uso de asistentes, pueden producir con rapidez versiones sencillas de código parametrizado. El resultado es la capacidad de desarrollar rápidamente aplicaciones de base de datos muy sofisticadas.

Estas son algunas de las características de Caché Server Pages

■ **Páginas de servidor dinámicas** – Como las páginas se crean dinámicamente en el servidor de aplicaciones mediante el código de la aplicación, en lugar de tener un servidor Web que simplemente devuelve código estático, las aplicaciones responden rápidamente a distintos tipos de solicitudes y adaptan las páginas de resultados que se envían de vuelta al navegador.

■ **Modelo de sesión** – Todo el proceso relacionado con las páginas de un solo navegador se considera parte de una sesión – desde la primera solicitud del navegador hasta que la aplicación termine o exceda el tiempo de respuesta(timeout). Este modelo de sesión permite el proceso de transacciones en la Web y muchas de las otras características de CSP.

■ **Conservación del estado del servidor** – En una sesión, los datos de la aplicación que están en el servidor (e incluso el contexto completo de la aplicación) se pueden conservar automáticamente en las solicitudes del navegador, facilitando el desarrollo y la ejecución de aplicaciones complejas.

■ **Arquitectura de objetos** – Como cada página corresponde a una clase, el código y otras características comunes a muchas páginas se pueden incorporar fácilmente mediante herencia. Normalmente también se hace referencia a los datos mediante objetos con todas las ventajas de la programación orientada a objetos.

■ **XML** – Caché soporta totalmente XML, como alternativa potente a HTML para crear páginas Web y como formato universal para mover datos entre aplicaciones y sistemas. El Asistente de XML se puede utilizar para generar automáticamente descripciones XML de los objetos de base de datos, incluyendo código para la importación y exportación de objetos utilizando documentos XML.

■ **Caché Application Tags para generación automática de código en el servidor de aplicaciones** – Estas etiquetas HTML extendidas son fáciles de utilizar como etiquetas HTML tradicionales. Cuando se añaden a un archivo HTML, generan código de aplicación sofisticado que proporciona amplia funcionalidad, como abrir objetos, ejecutar consultas y controlar el flujo del programa. Estas etiquetas son extensibles: los desarrolladores pueden crear las suyas propias para adaptarlas a sus necesidades especificas.

■ **Integración con herramientas conocidas de diseño Web** – CSP funciona con varias herramientas que facilitan la organización visual de una página. Con Dreamweaver, CSP da un paso más gracias a la capacidad de añadir Caché Application Tags mediante una sencilla interacción "señalar y pulsar". CSP también incluye un Asistente que facilita la visualización o edición de los datos de una base de datos Caché.

■ **Métodos del servidor que se pueden llamar desde el navegador** – Para facilitar el desarrollo de aplicaciones interactivas más dinámicas, CSP simplifica la llamada a métodos del servidor. Cuando se produce un evento en el navegador, generalmente porque el usuario lleva a cabo una acción), se puede llamar al código de la aplicación en el servidor y generar una respuesta al evento, todo ello sin la carga que supone transmitir y cargar una nueva página entera.

■ **Cifrado** – Caché aplica cifrado automáticamente a todos los datos de la URL, para ayudar a autentificar las solicitudes y evitar la manipulación. La clave de cifrado se conserva únicamente en el servidor, y sólo es válida para el transcurso de una sesión.

Todo esto es tecnología avanzada, pero no tiene porqué ser difícil utilizarla. Nos hemos centrado en que el uso de la tecnología sea fácil, haciendo que todas estas posibilidades trabajen de forma automática para el usuario. Nuestra filosofía es la potencia mediante la simplicidad; la complejidad debe resolverse en nuestra implementación, no en su programación.

Las ventajas de Caché

La programación orientada a objetos por procedimientos más los Asistentes de Caché dan como resultado el desarrollo rápido de aplicaciones sofisticadas de base de datos.

Tecnologías Web tradicionales

En la tecnología Web tradicional, se envía una solicitud al servidor Web y el servidor Web recupera un archivo secuencial de HTML y lo envía de vuelta al navegador. Cuando las aplicaciones incluyen datos variables, el desarrollo se complica, y los programadores normalmente utilizan CGI (Common Gateway Interface) ,con lenguajes como Perl o tcl, en el servidor Web y realizan consultas SQL y solicitudes de procedimientos almacenados a la base de datos. Como entorno de programación, esto deja mucho que desear, y la ejecución (especialmente con gran número de usuarios) puede ser completamente ineficaz a medida que el servidor Web se sobrecarga.

Con CGI, cada solicitud del navegador crea habitualmente otro proceso. Para evitar esta sobrecarga, los programadores a veces enlazan el código de la aplicación directamente con el servidor Web, con un desafortunado efecto secundario: un error de ese código puede provocar la caída del servidor.

Páginas de servidor dinámicas

CSP utiliza una solución distinta para la programación y la ejecución: Tecnología de páginas de servidor dinámicas. El contenido (HTML, XML, hojas de estilo, imágenes y de otro tipo) se genera por programación en tiempo de ejecución en el servidor de aplicaciones, en lugar de proceder de archivos secuenciales, lo que permite mucha más flexibilidad al responder a las solicitudes de las páginas.

La mayoría del código de la aplicación se ejecuta en el Servidor de aplicaciones Caché, que puede estar o no en el mismo ordenador que el servidor Web. Parte del código (normalmente JavaScript o Java) se puede ejecutar en el navegador, generalmente para dar soporte a operaciones como validación de datos básicos, aplicar formato de nuevo o llamar al código del servidor.

Con esta solución, no hay porqué crear procesos para cada solicitud del navegador (como se hace en la solución CGI tradicional), lo que aumenta el rendimiento. Y como el código de la aplicación no está enlazado con el servidor Web, un error de la aplicación no provoca la caída del servidor.

Sesiones – el modelo de proceso

Todo el proceso relacionado con las páginas de un solo navegador se considera parte de una sesión – desde la primera solicitud del navegador hasta que la aplicación termine o se produzca un error de tiempo de espera (timeout). Cuando el servidor Web recibe una solicitud de página (“URL”) con la extensión de archivo “.csp”, esa solicitud se envía (mediante una fina capa de código Caché asociada al servidor Web) al Servidor de aplicaciones Caché adecuado, que puede estar en un ordenador distinto. (Si hay varios Servidores de aplicación Caché, el URL lógicamente especifica cuál utilizar).

Cuando el Servidor de aplicaciones Caché recibe la solicitud, determina si ya existe una sesión abierta para ese navegador. Si no es así, se inicia una automáticamente. A continuación, Caché ejecuta el código de aplicación asociado a esa página concreta, llevando a cabo las acciones solicitadas por el usuario y creando mediante programación el contenido HTML, XML, de imagen o de otro tipo que se envía de vuelta al navegador.

Conservación del estado

En la programación Web tradicional, cuando un navegador está ejecutando una aplicación, no hay forma efectiva (sin gran cantidad de programación) de retener en el servidor la información de una solicitud para la siguiente. En lugar de eso, las aplicaciones normalmente envían al navegador toda la información de estado que necesitan retener, ya sea en URLs o en campos de formulario ocultos. No es una técnica muy efectiva para las aplicaciones más complejas, que pueden recurrir a guardar datos temporalmente en archivos o bases de datos. Desafortunadamente, esto supone una carga significativa sobre el servidor, al guardar y almacenar el contexto de sesión de cada solicitud.

Una de las ventajas del modelo de sesiones es que permite a Caché conservar de forma automática y eficaz los datos entre llamadas de un navegador. CSP proporciona un objeto Session que contiene la información general de la sesión y propiedades que permiten al programador controlar diversas características de la sesión. La aplicación también puede almacenar en el objeto Session sus propios datos, que se conservan automáticamente entre una solicitud y la siguiente.

La aplicación determina en qué medida se conserva el estado estableciendo la propiedad Preserve del objeto Session en 0 o en 1. (El valor por defecto es 0, y se puede modificar dinámicamente en tiempo de ejecución.)

- **0** - Los datos almacenados en el objeto Session se retienen. (Los datos se definen como una propiedad multidimensional que acepta datos de cualquier tipo y permite cualquier número de subíndices, incluyendo subíndices de cadena con valores, sin ningún tipo de declaración.)
- **1** - Caché dedica un proceso a la sesión, de modo que se retiene todo el estado del proceso, incluyendo todas las variables, (no sólo las del objeto Session), los dispositivos de E/S y los bloqueos.

Hay diversas opiniones en cuanto a qué valor utilizar para Preserve, que reflejan las distintas necesidades de aplicación y filosofías de desarrollo. Un valor de 0 permite la partición lógica de todos los datos conservados y que varias sesiones compartan un solo proceso, conservando los recursos de la máquina, pero conserva el estado en menor medida. Un valor de 1 resulta más sencillo para el programador y proporciona una gama más amplia de posibilidades, a costa de utilizar más recursos del servidor.

Las ventajas de Caché

El uso de Páginas de servidor dinámicas y del Servidor de aplicaciones Caché aporta mayor flexibilidad a la hora de responder a las solicitudes, ejecución más rápida sin riesgo de que los errores de la aplicación provoquen la caída del servidor y un entorno de programación más productivo.

Arquitectura de clases de las páginas Web

Para cada página Web, existe su correspondiente clase, que contiene métodos (código) para generar el contenido de la página. Como estas clases no contienen propiedades, (datos), no hay necesidad de crear un objeto página para cada solicitud.

Cuando se recibe una solicitud, su URL se utiliza para identificar una clase de página y se llama al método "Page()" de esa clase. Normalmente, las clases de página proceden de una clase de página Web estándar denominada "%CSP.Page", que proporciona a cada página varias posibilidades integradas, como la generación de cabeceras y el cifrado. Estas posibilidades estándar se pueden omitir por diversos medios: heredando de otra superclase, utilizando la herencia múltiple o simplemente omitiendo métodos específicos.

Esta arquitectura de clases facilita el cambio de comportamiento de una aplicación y la utilización de un estilo común. También aporta todas las demás ventajas de la programación de objetos al desarrollo para la Web.

La clase de página contiene código para ejecutar la acción solicitada y generar y enviar una respuesta al navegador, pero no todo el código que se ejecuta se encuentra en esa clase de página. De hecho, la mayoría del código ejecutado está habitualmente en métodos de varias clases de bases de datos y quizás en clases adicionales de reglas de negocio. **De esta forma, el proceso de desarrollo consiste en desarrollar clases de página y de base de datos (y quizás clases adicionales de reglas de negocio).**

En general, recomendamos que las clases de página contengan sólo la lógica del interface. Las reglas de negocio y de la base de datos deben incluirse en clases distintas, de modo que exista una separación clara entre el código del interface de usuario y la lógica de la base de datos, y que sea más fácil añadir interfaces de usuario adicionales más adelante.

Múltiples Estrategias de desarrollo

El proceso global de desarrollo implica dos pasos. En primer lugar, Caché Object Architect se utiliza para crear las clases de base de datos que describen los datos que se almacenarán en la base de datos. Por ejemplo, puede haber una clase cuenta, con un nombre de cuenta y una dirección de facturación, y una clase persona con un nombre de persona y un número de teléfono.

A continuación, se define un conjunto de páginas Web para implementar el interface de usuario. Hay dos formas de crear las páginas Web con CSP:

■ **Diseño Web** - Utilizar una herramienta de diseño Web, o un editor de texto para, producir un archivo HTML con Caché Application Tags embebidas, y compilarlo para generar la clase de página. La programación de métodos adicionales de otras clases, se realiza de forma independiente.

■ **Programación** – Se programa una clase completa de la página codificando los métodos adecuados, normalmente con Caché Object Architect. La página enviada al navegador puede utilizar HTML, XML, WML u otros lenguajes de marcas.

Ambas soluciones ofrecen la misma calidad de aplicación con las mismas posibilidades. La solución deseada a menudo refleja la capacidad y la experiencia de las personas que trabajan en el proyecto, así como sus preferencias.

Diseño Web - Utilización de herramientas Web o HTML para crear la página Web

Con esta estrategia, se crea un archivo secuencial con HTML y Caché Application Tags (normalmente con una herramienta de diseño para Web, aunque también se puede utilizar un editor de texto). Ese archivo secuencial no se envía directamente al navegador. En su lugar, se utiliza como conjunto de especificaciones que lee el compilador Caché Web, que a su vez genera el código de la aplicación (la clase de la página Web) que se ejecuta siempre que se solicita la página.

La solución de diseño Web es potente porque:

■ Los creadores Web pueden diseñar el formato visual mientras los programadores se concentran en el código.

■ Gran parte del interface de usuario se puede programar sin procedimientos en un entorno visual y se puede mantener aislado de las reglas de negocio y de la base de datos.

■ A menudo es más fácil personalizar una aplicación para un usuario concreto permitiendo modificar la presentación visual y añadir funcionalidad sencilla sin necesidad de programación.

Aunque de esta forma se puede crear una aplicación completa sencilla, lo más frecuente es que un programador suministre el código adicional. Este código adicional se proporciona escribiendo métodos en otra clase y haciendo que el archivo HTML incluya una Caché Application Tag que llame al código o especifique las clases adicionales que deben incluirse ("heredarse") en el momento de la compilación. Es un buen ejemplo de la potencia de la herencia múltiple en la arquitectura de objetos. Como las especificaciones visuales de la aplicación se mantienen separadas de la mayoría de la lógica, es relativamente sencillo cambiar el aspecto sin necesidad de programar de nuevo. Basta con editar el archivo HTML y recompilar la página.

Con algunas herramientas de diseño Web, como Dreamweaver, Caché proporciona soporte "señalar y pulsar" para añadir Caché Application Tags, así como un Asistente de formularios de Caché que genera automáticamente el código necesario para ver o editar un objeto de base de datos. El usuario simplemente pulsa la clase de base de datos de su interés, y a continuación pulsa el conjunto de propiedades (campos) que se va a visualizar. El Asistente de Caché hace el resto, añadiendo HTML y Caché Application Tags a la página. Como la salida del Asistente es HTML, si el resultado no es exactamente el deseado, es fácil editarlo.

Las ventajas de Caché

Al conservar automáticamente la información de estado en el servidor, el tráfico de red y la carga sobre el servidor son menores, ya que la aplicación no tiene que seguir archivando y accediendo a los datos en cada solicitud de página. Y programar la aplicación es más sencillo.

Caché Application Tags

Se pueden añadir Caché Application Tags a los archivos HTML utilizados en la solución de diseño Web. Se utilizan como etiquetas HTML estándar, pero realmente son instrucciones para que el Compilador Caché Web genere código de aplicación que proporciona funcionalidad diversa, como abrir objetos, ejecutar consultas, controlar el flujo del programa y especificar directamente el código Caché Object Script. Por ejemplo, una etiqueta CSP:Object activa las propiedades y métodos de un objeto Caché que se va a utilizar en una página, y una etiqueta CSP:Search proporciona posibilidades de búsqueda en base de datos.

Las Caché Application Tags no están embebidas en el código HTML que se envía a n navegador, sólo están en el archivo ".csp" que lee el Compilador Caché Web. Ese compilador las transforma automáticamente en código HTML estándar, que puede procesar cualquier navegador.

Las Caché Application Tags son extensibles: los desarrolladores pueden crear las suyas propias para adaptarlas a sus necesidades especificas.

Métodos de servidor

CSP también permite que el navegador llame directamente a los métodos del servidor Caché. Una vez realizada la acción adecuada, ese método del servidor puede devolver código para que lo ejecute el navegador, generalmente JavaScript.

Si se utiliza la solución de diseño Web, basta con incluir en el código HTML una expresión con la sintaxis "#server(…)#". Entre los paréntesis se incluye el nombre del método al que llamar y los parámetros de entrada que necesita. El compilador Web sustituirá esta sintaxis con código JavaScript que, al ejecutarlo en el navegador, llamará al método del servidor Caché. Por ejemplo, supongamos que, cuando el usuario pulsa una imagen de carro de la compra, queremos llamar a un método del servidor denominado AddToCart(). La definición HTML de la imagen podría incluir: onClick="#server(..AddToCart())#"

HTML y XML

HTML es el "lenguaje" tradicional de la Web, y es uno de los lenguajes de programación más ampliamente conocidos del mundo.

Sin embargo para la transmisión de datos y comunicación entre sistemas está indicada la utilización de XML. Caché se adapta especialmente bien a XML porque puede producir automáticamente documentos XML. XML soporta la estructura de objetos y la representación de los datos mediante objetos.

Con XML, en lugar de enviar una descripción completa de la página al navegador, sólo se envían los datos en un formulario estructurado. Además, se indica al navegador que utilice una "Hoja de estilos" concreta para esa página, que contiene una descripción de lo que hacer con los datos. Una vez que el navegador tiene la hoja de estilos, puede recordarla y utilizarla de nuevo más adelante si la página (u otra página con la misma hoja de estilos) la utiliza.

XML es realmente eficaz para aplicaciones basadas en transacciones que usan de forma repetitiva las mismas páginas con datos distintos. Sólo deben transmitirse al navegador los datos, no la descripción HTML completa. Otro punto interesante de XML es que proporciona una separación definida entre el contenido de los datos (codificado en XML) y el comportamiento para visualizarlos y editarlos (codificado en las hojas de estilo).

CSP funciona igual con XML que con HTML. En lugar de indicar HTML en el método Page(), se indica XML. El código XML se deriva fácilmente de las clases de base de datos ejecutando el Asistente de XML de Object Architect: basta con especificar los nombres de las propiedades que se incluirán y el Asistente genera un método para proporcionar el documento XML.

Con XML hay que definir una hoja de estilos. Se puede proporcionar como clase de página independiente cuyo método Page() simplemente indica al navegador la hoja de estilos o, más frecuentemente, suele ser un sencillo archivo secuencial que el servidor Web transfiere al navegador como otro archivo cualquiera.

InterSystems Spain
Avda. de Europa
12 – Edif. Mónaco
Parque Empresarial de la Moraleja
28108 Alcobendas
Madrid
Teléfono: 91 484 18 80
Fax: 91 662 60 84

www.InterSystems.com

InterSystems Caché es una marca comercial de InterSystems Corporation. Otros nombres de productos son marcas comerciales de sus respectivos proveedores. Copyright © 2001 InterSystems Corporation. Todos los derechos reservados.
L07-705