

PAGINACIÓN DE SISTEMA DE **PROCESAMIENTO DE EVENTOS**

Antecedentes de la Invención

5 Los sistemas de procesamiento de eventos procesan los eventos continuamente en tiempo real a medida que son recibidos. Los eventos son elementos de datos que incluyen una carga útil de datos y un tiempo asociado. El procesamiento de eventos funciona con respecto a secuencias de eventos

10 asincrónicos o flujos de eventos, que son secuencias de eventos que ocurren a través del tiempo. Los ejemplos de las secuencias de eventos incluyen las variaciones mínimas (ticks) de valores que identifican los cambios en el precio de un valor y los mensajes en redes sociales. El procesamiento se

15 especifica sobre una secuencia de eventos con una consulta para habilitar la proyección, filtrado y agregación, por ejemplo. A diferencia de las consultas de bases de datos tradicionales, las consultas de procesamiento de eventos no se evalúan una vez que están sobre los datos almacenados. Por el contrario,

20 las consultas de procesamiento de eventos se evalúan continuamente sobre los eventos a medida que se producen los eventos. Para habilitar el procesamiento sustancialmente en tiempo real, los sistemas de procesamiento de eventos almacenan las consultas en la memoria y evalúan las consultas

25 en el sistema a medida que llegan los eventos.

Breve Descripción de la Invención

Lo siguiente presenta una breve descripción simplificada con el fin de proporcionar una comprensión básica de algunos aspectos de la materia objeto que se da a conocer. Esta breve descripción no es una perspectiva general extensa. No se pretende identificar elementos clave/críticos ni delimitar el alcance de la materia objeto reivindicada. Su único propósito es presentar algunos conceptos de una forma simplificada como un preludio a la descripción más detallada que se presenta más adelante.

Descrito de una forma breve, la presente invención se refiere a la paginación del sistema de procesamiento de eventos. Se genera una forma paginable de una consulta de procesamiento de eventos que soporta el intercambio de una consulta de procesamiento de eventos adentro y afuera de la memoria. Por ejemplo, la consulta de procesamiento de eventos se puede reescribir para incluir desencadenantes de paginación adentro y paginación afuera, así como la funcionalidad para soportar una carga diferida y la conservación de una consulta en una paginación afuera. En la operación, la consulta de procesamiento de eventos se carga en la memoria después de observar un desencadenante de paginación adentro y se descarga desde la memoria en un almacenamiento persistente después de observar el desencadenante de paginación afuera.

Para la realización de los fines anteriores y relacionados,

se describen en el presente documento ciertos aspectos ilustrativos de la materia objeto reivindicada en relación con la siguiente descripción y los dibujos anexos. Estos aspectos son indicativos de diversas maneras en que la materia objeto se puede practicar, en donde todos ellos pretenden estar en el ámbito de la materia objeto reivindicada. Otras ventajas y características novedosas pueden llegar a ser evidentes a partir de la siguiente descripción detallada al considerarse conjuntamente con los dibujos.

10 **Breve Descripción de los Dibujos**

La Figura 1 es un diagrama de bloques de un sistema de procesamiento de eventos.

La Figura 2 es un diagrama de bloques de un componente de paginación representativo.

15 La Figura 3 es un diagrama de bloques de un componente de reescritura representativo.

La Figura 4 es una ilustración gráfica de una reescritura con funciones de recopilación de estadísticas y paginación basadas en los datos históricos.

20 La Figura 5 es un diagrama de flujo de un método para generar una consulta paginable en un sistema de procesamiento de eventos.

La Figura 6 es un diagrama de flujo de un método de paginación de una consulta.

25 La Figura 7 es un diagrama de flujo de un método de

paginación adentro de una consulta.

La Figura 8 es un diagrama de flujo de un método de paginación afuera de una consulta.

La Figura 9 es un diagrama de bloques esquemático que
5 ilustra un entorno operativo adecuado para los aspectos de la presente invención.

Descripción Detallada de la Invención

Un sistema de procesamiento de eventos puede incluir un alojamiento y servicio a un gran número de suscripciones o
10 consultas de eventos por mucho tiempo. Por ejemplo, pueden existir muchas consultas para un único usuario, tales como aquéllas asociadas con las notificaciones de tráfico diarias, actualizaciones del clima, noticias sobre temas favoritos, actualizaciones de los marcadores en deportes, y notificaciones
15 de vuelos de líneas aéreas, entre otras cosas. Además, en el contexto de un sistema o servicio de procesamiento de eventos a gran escala, esto puede corresponder a millones, o a un orden de magnitud más grande, de consultas para su procesamiento. El procesamiento de eventos también puede
20 involucrar a un gran número de eventos además de un gran número de suscripciones.

El hecho de lograr el procesamiento en tiempo casi real implica almacenar en la memoria tanto código asociado con las consultas de procesamiento de eventos como sea posible. Como
25 resultado, cuando se produce un evento, no hay demora en la

carga de código en la memoria para procesar el evento.

Sin embargo, los sistemas de procesamiento de eventos pueden tener consultas pequeñas y no tan pequeñas que ocupan cantidades no triviales de memoria cuando se
5 mantienen residentes en la memoria. En vista de lo finito de la memoria y de la gran escala de procesamiento, el desafío es cómo intercambiar el código de procesamiento de eventos adentro y afuera de la memoria de una manera inteligente.

Los sistemas operativos emplean una técnica de
10 paginación para administrar la memoria mediante la lectura y escritura de páginas de datos hacia y desde el almacenamiento secundario, tal como el disco, para usarse en el almacenamiento primario, tal como la memoria. La paginación del sistema operativo, sin embargo, es independiente del
15 dominio y, por consiguiente, puede tomar decisiones subóptimas acerca de cuándo y qué paginar adentro y qué paginar afuera. Adicionalmente, las técnicas de paginación relacionadas con sistemas operativos de propósitos generales no se escalan correctamente para el procesamiento de eventos
20 de alto rendimiento y esto conduce a un acceso excesivo al disco y a la memoria.

Los siguientes detalles corresponden generalmente a la paginación del sistema de procesamiento de eventos. Las consultas de procesamiento de eventos paginables se generan
25 para soportar el intercambio de consultas, o partes de las

mismas, adentro y afuera de la memoria. Se puede proporcionar soporte para la paginación mediante la reescritura de las consultas de procesamiento de eventos de acuerdo con una modalidad. Por ejemplo, las expresiones de consulta se pueden reescribir automáticamente con desencadenantes de paginación adentro, desencadenantes de paginación afuera, así como con la funcionalidad para la carga diferida y conservación de la consulta. Además, se pueden especificar desencadenantes con respecto a flujos o secuencias de eventos, lo que permite la explotación de las características específicas del dominio de un sistema de procesamiento de eventos. También se puede insertar la funcionalidad para recolectar los datos con respecto a la ejecución de la consulta. Posteriormente, los datos recolectados se pueden analizar con el fin de buscar patrones útiles para especificar los desencadenantes de paginación adentro y paginación afuera. Además, la funcionalidad de paginación puede aprovechar las características de un sistema de procesamiento de eventos, incluyendo los operadores de consultas y los puntos de control, por ejemplo.

A continuación se describen con mayor detalle diversos aspectos de la presente invención con referencia a los dibujos anexos, en donde los numerales iguales se refieren en general a los elementos iguales o correspondientes a lo largo del texto. Sin embargo, se debe entender que los dibujos y la descripción detallada concerniente a los mismos, no pretenden limitar la

materia objeto reivindicada a la forma particular que se da a conocer. Más bien, la intención es cubrir todas las modificaciones, equivalentes y alternativas que caigan dentro del espíritu y alcance de la materia reivindicada.

5 Haciendo referencia inicialmente a la Figura 1, se ilustra el sistema de procesamiento de eventos 100. Conforme a un patrón de diseño de observadores, el sistema de procesamiento de eventos 100 se configura para aceptar suscripciones de uno o más observadores a una o más secuencias o flujos de eventos
10 observables, y para proporcionar notificaciones a los uno o más observadores en línea con las suscripciones correspondientes. Las suscripciones pueden corresponder a, o incluir, consultas en firme que ejecutan continuamente el procesamiento de datos a medida que llegan por la duración de la suscripción, de una
15 manera opuesta a las consultas *ad hoc* que están diseñadas para ejecutarse una vez. El tipo de notificación puede variar desde simples alertas hasta una secuencia de eventos que también pueden aceptar suscripciones.

 El sistema de procesamiento de eventos 100 incluye el
20 componente de evaluación 110, el componente optimizador 120, y el componente de paginación 130. El componente de evaluación 110 se configura para evaluar las consultas de procesamiento de eventos o, de una manera más simple, las consultas. Las consultas se pueden especificar en términos de
25 expresiones de consulta que incluyen una pluralidad de

operadores de consulta soportados por el sistema de procesamiento de eventos para llevar a cabo la proyección, el filtrado y la agregación, entre otras cosas. Las consultas se pueden cargar en la memoria y, una vez recibidos los datos, el
5 componente de evaluación 110 puede evaluar las consultas sobre los datos y producir las notificaciones.

El componente optimizador 120 se configura para optimizar las consultas recibidas con el fin de mejorar el desempeño de la evaluación. El componente optimizador 120
10 puede explotar las propiedades algebraicas de las consultas y los datos con relación a la carga de recursos del sistema y las capacidades de procesamiento, entre otras cosas, con el fin de generar una consulta modificada para la evaluación mediante el componente de evaluación 110. Como ejemplos no limitantes, el
15 componente optimizador 120 puede unir una serie de filtros en un único filtro, mover un filtro, o dividir una consulta para el procesamiento concurrente.

El componente de paginación 130 se configura para intercambiar consultas adentro y afuera de la memoria. Más
20 específicamente, el componente de paginación 130 se configura para cargar y descargar el código y los datos asociados con una consulta adentro y afuera de la memoria. En el presente documento, la carga de una consulta adentro de la memoria se conoce como paginación adentro de una consulta, y la descarga
25 de una consulta desde la memoria se denomina como

paginación afuera de una consulta. Aun cuando en el presente documento se utiliza paginación adentro y paginación afuera, esto no necesariamente significa que un bloque de almacenamiento denominado como una página sea cargado en la memoria o descargado desde la memoria como se hace convencionalmente mediante un sistema operativo. Más bien, el bloque de almacenamiento puede ser sustancialmente de cualquier tamaño asociado con los datos y el código de consulta en particular.

De acuerdo con una modalidad, el componente de paginación 130 puede reutilizar las características y la funcionalidad de un sistema de procesamiento de eventos para realizar la paginación. Un planteamiento consiste en construir la paginación en un motor de ejecución de consulta particular, o el componente de evaluación 110 descrito en el presente documento. Esto, sin embargo, implica la producción de un sistema de paginación de bajo nivel con poca abstracción que involucra a programadores, administradores de memoria y administradores de entrada/salida, por ejemplo. Un segundo planteamiento, descrito en el presente documento, es construir sobre las capacidades del procesamiento de evento por evento, e implementar la paginación con los operadores de consulta. Como se describirá adicionalmente más adelante, una expresión de consulta que comprenda uno o más operadores de consulta se puede reescribir para incluir el soporte para la paginación.

Por otra parte, se pueden explotar las características específicas del dominio de alto nivel de un sistema de procesamiento de eventos para la paginación, las cuales no están disponibles para los mecanismos de paginación de los
5 sistemas operativos de bajo nivel.

Por ejemplo, considere una consulta de seguimiento de vuelo que notifica a una persona si el vuelo de esa persona va a estar retrasado. Esta consulta se podría someter cuando se haga la reserva de un vuelo, pero sólo necesita estar activa un
10 día antes de la salida del vuelo. Esta consulta en firme tiene una larga vida, potencialmente de meses, entre el momento en que se somete la consulta y el tiempo en que la consulta está activa. Debido a que el código correspondiente a la consulta no va a reaccionar a ningún evento durante un período, el código
15 se puede paginar afuera y posteriormente se puede paginar adentro un día antes de la salida del vuelo. Éste es un ejemplo muy trivial debido a que se trata de tiempo explícito en la consulta. Sin embargo, este ejemplo pone de relieve el uso de conocimientos del dominio de las consultas y el espacio de
20 eventos para tomar decisiones inteligentes con respecto a cuándo paginar adentro y cuándo paginar afuera.

Poniendo atención a la Figura 2, se ilustra un componente de paginación 130 representativo. El componente de paginación 130 incluye un componente de reescritura 210, un
25 almacenamiento de estadísticas 220, un componente de proceso

230 y un componente de actualización 240. El componente de reescritura 210 se configura para reescribir las expresiones de consulta de una manera que añada soporte para la paginación. En otras palabras, las reescrituras hacen que las consultas se
5 conviertan en consultas paginables. Las reescrituras pueden desencadenar la paginación con base en la llegada de los eventos en una secuencia de eventos. Aunque no se limita a lo mismo, la paginación se puede desencadenar con base en una secuencia de eventos temporizados o las variaciones mínimas
10 (ticks). La reescritura puede implicar también agregar un mecanismo para desencadenar la paginación afuera. Las reescrituras adicionales pueden pertenecer a una carga diferida, así como a la recolección y utilización de los datos de ejecución para tomar decisiones de paginación. En particular,
15 las consultas o partes de las mismas se pueden inyectar con la funcionalidad para recolectar los datos con respecto a la ejecución de las consultas y para guardar los datos en el almacenamiento de estadísticas 220. Estos datos pueden capturar el número de eventos, así como cuándo se producen
20 los eventos, entre otras cosas. El componente de proceso 230 se configura para analizar y procesar los datos con el fin de detectar los patrones que puedan ser útiles para especificar una consulta paginable. Por ejemplo, si la actividad del procesamiento de eventos se lleva a cabo diariamente a las
25 8:00 a.m. (por ejemplo, debido a que un evento llegue en una

de las entradas de la consulta), el componente de proceso 230 puede detectar o inferir este patrón. Como resultado, puede estar un desencadenante de paginación adentro diariamente antes de las 8:00 a.m., y puede estar un desencadenante de paginación afuera algún tiempo después de las 8:00 a.m. Cuando dichos datos o información con respecto a los patrones está disponible para la consulta o para consultas similares, el componente de reescritura 210 puede utilizar la información para iniciar la reescritura de una consulta en una forma paginable. Sin embargo, el componente de reescritura también puede especificar la consulta de una manera que permita la actualización posterior. El componente de actualización 240 se configura para actualizar una consulta paginable con base en los patrones identificados mediante el componente de proceso 230. En particular, los desencadenantes de paginación adentro y de paginación afuera se pueden modificar para reflejar el conocimiento actual con respecto a una consulta y a los eventos relacionados.

Poniendo atención brevemente a la Figura 3, se ilustra con mayor detalle un componente de reescritura 210 de ejemplo. Como se muestra, el componente de reescritura 210 incluye el componente de paginación adentro 310, el componente de cálculo 320, y el componente de paginación afuera 330. El componente de paginación adentro 310 se configura para inyectar la funcionalidad en una expresión de consulta para

desencadenar una funcionalidad de paginación adentro que carga la memoria con código y los datos de estado potencialmente asociados con la operación de consulta. La funcionalidad de paginación adentro se puede desencadenar con base en la presentación de un evento, tal como una variación mínima (tick) de temporizador o algún otro mensaje. Además, el componente de paginación adentro 310 se configura para introducir la funcionalidad que permite actualizar cuándo desencadenar una paginación adentro, por ejemplo, con base en las estadísticas de ejecución anteriores. El componente de cálculo 320 se configura para inyectar la funcionalidad que permite que la cuando menos una parte de una expresión de consulta o de cálculo asociada con los mismos sea intercambiada hacia adentro y hacia afuera de la memoria. En una implementación, esto se puede llevar a cabo mediante el uso de la funcionalidad de carga diferida de un sistema de procesamiento de eventos, así como de la persistencia del estado de operador en la forma de la funcionalidad de punto de control. El componente de cálculo 320 también se configura para que la funcionalidad de inyección recolecte los datos relacionados con la ejecución y los guarde en el almacenamiento de estadísticas 220. El componente de paginación afuera 330 se configura para inyectar la funcionalidad que desencadena una operación de paginación afuera. Esto se puede lograr cuando menos en parte mediante

la explotación de la funcionalidad de puntos de control de un sistema de procesamiento de eventos para guardar el código de consulta y el estado en un almacenamiento persistente, e iniciar la descarga del código desde la memoria, por ejemplo, mediante el establecimiento de un puntero en nulo y permitiendo que el espacio sea reclamado mediante un mecanismo de recolección de basura. También se puede añadir la funcionalidad mediante el componente de paginación afuera 320 para permitir las actualizaciones con respecto a cuándo desencadenar una operación de paginación afuera, por ejemplo, con base en las estadísticas de ejecución.

En seguida se presentan los detalles asociados a una implementación de ejemplo de un aspecto de la invención. La descripción comienza con la paginación adentro del código de consulta y los datos con temporizadores. Esto es seguido por una descripción de los patrones de activación para paginar adentro el código. Por último, se proporciona una descripción para paginar afuera el código y los datos.

Se pueden someter varias consultas en firme mucho tiempo antes de que estén activas. Algunas de estas consultas pueden ser de larga vida. Considere una consulta de seguimiento de vuelo. Una consulta en firme de seguimiento de un vuelo por lo general solamente necesita durar veinticuatro horas, empezando cierto tiempo antes de registrarse (check-in), por ejemplo, para sugerir salir hacia el aeropuerto; o se puede

cancelar y durar hasta algún tiempo después del aterrizaje, por ejemplo, para sugerir recoger a alguien en el aeropuerto. Sin embargo, las consultas en firme para el seguimiento de vuelo pueden ser sometidas a un sistema de procesamiento de
 5 eventos para el procesamiento al momento de la reserva de un vuelo. Además, puede haber una consulta para varias personas interesadas en el vuelo.

Con el fin de describir estas consultas, los operadores basados en el tiempo se utilizan a menudo para gestionar el
 10 tiempo de ejecución de la consulta. Para el ejemplo de la consulta de vuelo, se puede utilizar un operador de "RetrasarSuscripción" o "SaltarHasta" para retrasar el procesamiento de eventos hasta algún tiempo antes de la salida del vuelo, y se puede utilizar un operador de "TomarHasta" para
 15 conseguir un efecto similar en cuanto a la eliminación de la consulta después de que el vuelo haya aterrizado. Por ejemplo:

```
vuelos.RetrasarSuscripción(salida - 12hr)
```

```
.Donde(f=>f.Id=="BA49")
```

```
.TomarHasta(Aterrizaje + 12hr)
```

```
20 .Suscribir(...)
```

La expresión de consulta anterior se puede reescribir para soportar la paginación con base en las propiedades algebraicas de la consulta. En primer lugar, el operador de "RetrasarSuscripción" generalmente puede conmutarse con
 25 otros operadores de consulta (por ejemplo, se puede necesitar

mantener algunas propiedades en ausencia de efectos secundarios debido a que una suscripción). En otras palabras, no importa el orden del operador de "RetrasarSuscripción" con respecto a otros operadores. De conformidad con lo anterior, el

5 operador de "RetrasarSuscripción" se puede empujar hacia abajo más cerca del consumidor. De hecho, el operador se puede mover justo frente a la llamada de "Suscribir" como sigue:

```

vuelos.Donde(f=>fId=="BA49")
10      .TomarHasta(Aterrizaje + 12hr)
      .RetrasarSuscripción (salida - 12hr)
      .Suscribir(...)

```

A continuación, el operador de "RetrasarSuscripción" se puede ampliar en una forma más primitiva, mediante el uso de

15 la siguiente identidad algebraica:

```

xs.RetrasarSuscripción(t) ==
Temporizador(t).SeleccionarTantos(_ => xs)

```

Aquí, la fuente del operador de "RetrasarSuscripción", "xs", termina en el cuerpo lambda para la función de selección

20 del operador de "SeleccionarTantos". En este punto, se pueden emplear funcionalidades de puntos de control asociadas con un sistema de procesamiento de eventos, específicamente puntos de control para los operadores de orden superior, tales como "SeleccionarTantos". No es sino hasta que llega un evento

25 desde la fuente del operador de "SeleccionarTantos", en este

caso un pulso del temporizador, que se debe cargar el cuerpo de la función del selector en la memoria. El ejemplo actual se puede escribir como sigue:

```

5      Temporizador(salida - 12hr).SeleccionarTantos(_ =>
      vuelos.Donde(f => f.Id == "BA49")
      .TomarHasta(aterrizaje + 12hr))
      .Suscribir(...)
```

La parte en cursiva de la consulta se puede extraer de la consulta mediante la inserción de una expresión "trampa".
 10 Mediante el uso de la capacidad para definir un observable (el tipo de la parte en cursiva) como un artefacto con nombre, se puede eliminar la parte de código. Un nuevo artefacto se puede definir como sigue:

```

15      ctx.DefinirObservable(algúnId,vuelos.Donde(f => f.Id ==
      "BA49")
      .TomarHasta(aterrizaje + 12hr));
```

El artefacto "algúnId" es un observable (por ejemplo, tipo IObservable) que no contiene código sino solamente un identificador. Este identificador se puede utilizar como un
 20 soporte para la porción eliminada de la consulta, insertando de esta manera un nivel de dirección indirecta. Esto permite que la consulta original sea reescrita en su forma final:

```

      Temporizador(salida - 12hr).SeleccionarTantos(_ =>
      algúnId)
25      .Suscribir(...)
```

Esto significa que la suscripción con un consulta en particular se retrasa hasta doce horas antes de la salida. En ese momento, el operador de "SeleccionarTantos" recibirá un evento y el código de consulta será intercambiado hacia adentro, por ejemplo, mediante la lectura del código desde el disco, la compilación del código, y la carga del código hacia adentro de la memoria. El resultado es que el código residente para la lógica de la consulta se reduce significativamente. La forma final anterior asume que el componente de evaluación no realiza una puesta en línea apremiante de los artefactos mencionados en la consulta, especialmente si éstos se presentan en las posiciones internas (por ejemplo, expresiones lambda), sino que en su lugar, lo realiza de una forma lenta difiriendo la puesta en línea hasta que se necesite. Este comportamiento puede ser intrínseco al sistema o bien puede ser agregado por el componente de reescritura 210 como sigue:

```

    Temporizador(salida - 12hr).SeleccionarTantos(_ =>
        Diferir(algúnId))
        .Suscribir(...)

```

En y por sí mismo, el operador de "SeleccionarTantos" desacopla un observable de una "fuente" externa y las suscripciones a uno o más observables "internos". Las porciones internas se pueden marcar como puntos de control por separado de la porción externa, y solamente crean una instancia al recibir un evento desde la suscripción de la fuente

externa, tal como la secuencia del temporizador en este ejemplo. Sin embargo, esto se puede optimizar aún más mediante la reducción del patrón hasta un operador especial si resulta ser beneficioso en términos de utilización de los recursos. Por ejemplo, un operador de "ContinuarCon" se podría emplear como sigue:

```
Temporizador(salida - 12hr).ContinuarCon(algúnId)
    .Suscribir(...)
```

Se pueden hacer optimizaciones adicionales mediante la explotación de las propiedades del temporizador, por ejemplo, mediante la combinación de múltiples temporizadores en temporizadores principales de mayores intervalos que se pueden utilizar para desencadenar la creación de instancias de los temporizadores en el siguiente nivel. Esto puede ser beneficioso en el sentido de que se pueden reutilizar los objetos que se refieran al recurso del temporizador, en lugar de crear instancias de varios objetos de temporizador a través de diferentes consultas en firme. Estos objetos se pueden combinar con una lista de controlador que consuma un espacio de $O(n)$. Considere las siguientes consultas:

```
Temporizador(t1).SeleccionarTantos(_ => xs.Bar()).
Foo()).Suscribir(...)
Temporizador(t2).SeleccionarTantos(_ => ys. Qux()).
Baz()).Suscribir(...)
```

Utilizando las técnicas descritas anteriormente, las

porciones internas de los operadores de "SeleccionarTantos" se pueden paginar afuera. Sin embargo, los operadores de "SeleccionarTantos" residentes se pueden optimizar adicionalmente mediante temporizadores coalescentes. La reescritura algebraica para los temporizadores es la siguiente:

Temporizador(t) == Temporizador (N * (t/N)).

SeleccionarTantos(_ => Temporizador(t % N))

Aquí, un temporizador con el tiempo debido "t" en un temporizador de mayores intervalos con una resolución de "N".

Empleando la eliminación de la subexpresión común, se pueden combinar ahora los recursos del temporizador. Por ejemplo, si "N" se establece en veinticuatro horas, no es sino hasta la medianoche del tiempo debido que se crean las instancias de los temporizadores restantes. De esta manera, únicamente un temporizador de un día necesita ser residente por día. Asumiendo que los temporizadores "t1" y "t2", en el ejemplo, dan como resultado el mismo día "d", entonces:

Temporizador(d).SeleccionarTantos(_ => Temporizador (t1 % N))

.SeleccionarTantos(_ => xs.Bar().Foo()).Suscribir(...)
Temporizador(d).SeleccionarTantos(_ => Temporizador (t2 % N))

.SeleccionarTantos(_ => ys.Qux().Baz()).Suscribir(...)

Además, los operadores de "SeleccionarTantos" se pueden "voltear" usando la siguiente identidad (si se sostienen algunos

requisitos previos sobre los efectos secundarios):

```
xs.SeleccionarTantos(_ => ys).SeleccionarTantos(_ => zs)
== xs.SeleccionarTantos(_ => ys.SeleccionarTantos(_ => zs))
```

Como resultado, se producen los siguientes planes de

5 consulta:

```
Temporizador(d).SeleccionarTantos(_ => Temporizador(t1
% N)
```

```
.SeleccionarTantos(_ => xs.Bar().Foo()))
```

```
.Suscribir(...)
```

10 Temporizador(d).SeleccionarTantos(_ => Temporizador(t2
% N)

```
.SeleccionarTantos(_ => ys.Qux().Baz()))
```

```
.Suscribir(...)
```

No solamente se puede reutilizar el "Temporizador(d)",
15 sino que los cuerpos de los operadores de "SeleccionarTantos"
pueden ser paginados afuera. El componente de reescritura 210
también puede volver a aplicar de una forma recursiva esta
técnica para obtener el siguiente nivel del temporizador de
intervalos (por ejemplo, con una resolución $M < N$, tal como 1
20 hora).

Observe que el esquema de paginación anteriormente
descrito tiene la ventaja de obtener las expresiones requeridas
con anticipación en el tiempo, de tal manera que el costo de la
paginación adentro no afecta de una manera adversa la
25 precisión del temporizador. Este parámetro se puede ajustar,

por ejemplo, con base en un parámetro de tolerancia de tiempo pasado a la consulta. Por ejemplo, se podría redondear el tiempo en los temporizadores hasta una resolución de 1 minuto para aumentar la probabilidad de reuso.

5 Aunque con mucha frecuencia se puede utilizar el diferimiento de la carga del código con base en el temporizador, como se describe anteriormente, el diferimiento del código con base en el temporizador requiere que las consultas tengan una cierta forma. Las consultas, sin embargo, pueden tener una
10 forma arbitraria. Por ejemplo, se puede utilizar alguna secuencia o flujo de eventos que no sea un temporizador, como una fuente de consulta. En este caso, la reescritura algebraica por sí sola puede no ser suficiente.

 Por el contrario, las consultas pueden reescribirse con el
15 código para recopilar las estadísticas con respecto a la ejecución de la consulta, las cuales se pueden guardar en el almacenamiento de estadísticas 220. El almacenamiento de estadísticas 220 puede guardar de esta manera las estadísticas históricas con respecto a la ejecución de la consulta. El
20 componente de proceso 230 se configura para operar con respecto a las estadísticas históricas y determinar y/o inferir la información útil para la paginación, tal como cuándo y con qué frecuencia se presentan los eventos. El componente de reescritura 210 o el componente de actualización 240 puede
25 emplear datos o información a partir del componente de proceso

230 en la especificación de consultas para la paginación, es decir, en los desencadenantes de paginación adentro y paginación afuera.

Consideremos una fuente que produce eventos con poca
 5 frecuencia. Aquí, el resto de la consulta podría ser cargada de una forma diferida. De una manera más en general, las consultas pueden tener muchas fuentes, pero las técnicas descritas en este documento son igualmente aplicables. Se consideran las consultas de una sola fuente para mayor
 10 simplicidad. Por ejemplo, supongamos la siguiente expresión de consulta:

```
Clima("MAR").Donde(w=>w.Temperatura>30°C).Bar().Foo()
```

Mientras que el flujo parametrizado para el clima en
 Seattle puede producir eventos con frecuencia, las
 15 posibilidades de un evento pasando el predicado (temperatura sobre los 30 grados Celsius) son más bien escasas. Dicha información se puede obtener mediante la instrumentación de la consulta para la adquisición de las estimaciones de cardinalidad (o más bien de velocidad, dada la importancia de la dimensión
 20 de tiempo en un sistema de procesamiento de eventos) que se pueden tomar en cuenta al reescribir la consulta:

```
Si xs = Clima("MAR").Donde (w=>w.Temperatura>30°C) en  

  xs.Medida().Bar().Foo()
```

Aquí, el operador de "Medida" toma "xs" como la fuente,
 25 no solamente para suscribirse a ella, sino también para asociar

las mediciones con esa subexpresión. Por ejemplo, el operador de "Medida" puede almacenar los registros históricos de la tasa de eventos producidos mediante la subexpresión. Si "Clima("MAR")" se fuera a instrumentar de una manera similar, también se podría calcular la proporción de ingreso y egreso de la subexpresión. Se debe apreciar que se pueden obtener muchas estadísticas a partir de esto, de varios órdenes de derivadas (conteo de eventos, velocidad, aceleración, arranque; tiempo de proceso, etcétera).

10 Tómese en consideración que la parametrización del flujo de "clima" proporciona también el filtrado, pero se aplicarían las mismas observaciones si el flujo fuera una enorme "manguera" que necesitara de un filtrado usando el operador de "Donde":

15 Clima.Donde(w => w.Ciudad == "Mar"&&w.Temperatura >30°C). Bar().Foo()

En tales casos, podrían normalizarse las consultas, por ejemplo, con varias cláusulas consecutivas de operadores de "Donde", de tal forma que pongan el predicado más restrictivo al principio. Para lo que sigue, se asume que la condición de temperatura causa un poco frecuente desencadenamiento de eventos, o más generalmente, en:

xs.Donde(f).Seleccionar(p)...

La fuente "xs" produce eventos con poca frecuencia, según lo determinado por el componente de proceso 230. En

este punto, tendría sentido hacer que el flujo del evento de "xs" desencadenara la paginación adentro del resto de la lógica de la consulta. Para ello, se pueden utilizar técnicas similares a los temporizadores, pero teniendo en cuenta que "xs" puede
 5 tener una cardinalidad de más de uno, de esta manera es deseable paginar adentro el cálculo una vez y no para cada mensaje. En una forma general, es posible la siguiente reescritura:

```
xs.F()==xs.Publicar(ys=>ys.Primer().SeleccionarTantos(y
10 =>ys.EmpezarCon(y).F()))
```

Esto puede parecer bastante complicado, pero es necesario en el caso general donde son posibles los efectos secundarios. En particular, puede no ser deseable suscribirse a "xs" múltiples veces, debido a que la creación de una
 15 suscripción puede tener un efecto secundario, tal como el ponerse en contacto con un servicio externo. Es ahí donde entra el uso del operador de "Publicar". Si "xs" fuera un flujo muy activo, significando que se producirían notificaciones independientemente de las suscripciones, en lugar de comenzar
 20 a producir notificaciones en respuesta a una suscripción, entonces se puede eliminar el operador de "Publicar". Para simplificar el código anterior, supongamos que "xs" es un flujo muy activo al que uno se puede suscribir tantas veces como se desee:

```
25 xs.F() == xs.Primer().SeleccionarTantos(x =>
```

```
xs.EmpezarCon(x). F())
```

Ahora llega a quedar más claro que la carga es diferida para la expresión "F", la cual podría ser arbitrariamente grande (por ejemplo, `Donde(f).Seleccionar(p)`), hasta que llegue el primer evento en "xs". Con el fin de que la expresión de la consulta original "F" no falle en ese evento, el componente de reescritura 210 puede volver a insertar la expresión al frente de la nueva suscripción a "xs". Esto supone que la implementación del operador de "EmpezarCon" es sincrónica y que no fallará en los eventos recibidos en "xs" entre cuando se recuperó "x" y cuando se hace la suscripción interior a "xs". Si no existe esta garantía, se pueden entretelar otros operadores para mantener una memoria intermedia de mensajes.

De cualquier manera, de aquí en adelante, se aplican las mismas técnicas que aquéllas que se han descrito para los temporizadores. La porción interior del operador de "SeleccionarTantos" se puede paginar en una forma lenta a la llegada de un mensaje (para los temporizadores, esto es una marca (tick)). Se puede construir un operador especializado de "PaginarCuandoLlegaPrimerEvento" para hacer todo esto:

```
xs.PaginarCuandoLlegaPrimerEvento(ys=>ys.F())
```

Este operador se puede construir como la composición de operadores tal como se muestra anteriormente, o se puede construir como su propio primitivo que utiliza las mismas clases de base del operador de orden superior que el operador de

"SeleccionarTantos" para volver a utilizar los mecanismos de carga diferida y puntos de control existentes. Es decir, no existe costo alguno para los puntos de control y la residencia en memoria para "ys.F()" sino hasta que llega el primer evento.

- 5 Adicionalmente, el parámetro para este operador se puede abstraer en un artefacto cargado de una forma diferida como sigue:

```
ctx.DefinirObservable(algúnId, ys=>ys.F());
```

```
xs.PaginarCuandoLlegaPrimerEvento(algúnId)
```

- 10 La llegada de eventos que desencadenan la paginación puede conducir a demoras en el procesamiento de mensajes debido al tiempo de carga. En colaboración con la recopilación de estadísticas, esto se podría mitigar si se pudiera predecir la llegada del siguiente mensaje, por ejemplo, a partir de los datos
- 15 históricos. Esta información se puede utilizar entonces para crear una señal antes de tiempo. Independientemente de esto, todavía es necesario el desencadenamiento basado en los eventos en el caso de que la predicción sea errónea. Con esta observación, la técnica de paginación se puede generalizar para
- 20 permitir un evento de paginación adentro que vaya más allá de "xs.Primer()" o un temporizador. Es decir, considere el siguiente operador:

```
xs.PaginarAdentro(ys=>paginarAdentroEvento,ys=>ys.F())
```

- Aquí, "xs" se alimenta a dos expresiones lambda, cuyo
- 25 alias es "ys". La segunda expresión lambda es donde se inserta

el código paginable, como se describió anteriormente. La primera expresión lambda proporciona una señal, que puede ser una expresión observable arbitraria que puede usar la fuente "xs" (con alias "ys"). Por ejemplo, si se desea paginar adentro
 5 el cálculo cuando llega el primer evento o cuando vence un temporizador predictivo, "xs.F()" se puede reescribir como sigue:

```
xs.PaginarAdentro(ys=>Unir(ys.Primer(),Temporizador(t),
ys=>ys.F()))
```

10 Mediante la reutilización de la rica composición de operadores de procesamiento de eventos, se proporciona un medio muy flexible para construir un sistema de paginación.

Teniendo en cuenta el mecanismo descrito anteriormente para paginar adentro el código que representa un cálculo de
 15 procesamiento de evento, se dan como resultado dos artefactos del tiempo de ejecución, es decir, el código que se paginó adentro y el estado acumulado por un cálculo recién iniciado. Cuando un cálculo resultante se alimenta con poca frecuencia con nuevos eventos entrantes (y suponiendo que los
 20 temporizadores se han diseñado también como flujos de eventos), es conveniente poder paginar afuera el cálculo, incluyendo el código y el estado.

Un sistema de procesamiento de eventos puede incluir un mecanismo de punto de control de estado. Por consiguiente, el
 25 soporte del operador existente para instalar el puntos de control

de estado se puede utilizar para soportar la paginación con el fin de que se pague afuera de una manera efectiva un cálculo en ejecución.

Con el fin de proporcionar flexibilidad, el operador de "PagarAdentro" descrito anteriormente se puede incorporar en un operador de "Pagar" de ejemplo, el cual también recibe la señal de paginación afuera:

```
IObservable<T>Pagar<T,I,R,O>(IObservable<T>fuente,
    Func<IObservable<T>,IObservable<I>>paginarAdentro,
    Func<IObservable<T>,IObservable<R>>calcular,
    Func<IObservable<T>,IObservable<O>>paginarAfuera);
```

La funcionalidad del operador se puede construir de una manera compositiva, como se ilustra anteriormente para la porción de pagar adentro. Operacionalmente, el operador de "Pagar" hace varias cosas durante un ciclo de vida. El operador se suscribe a una fuente introduciendo un tema para difundir mensajes a las funciones suministradas sin duplicar los efectos secundarios de las suscripciones hacia la fuente, y sin perder los mensajes ya sea mediante el uso de esquemas de bloqueo o bien limitando el almacenamiento en la memoria intermedia. Adicionalmente, el operador de "Pagar" realiza repetidamente varias acciones. En primer lugar, alimenta la fuente difundida a la función de "PagarAdentro" y se suscribe a su resultado. Al recibir una notificación, alimenta la fuente difundida a la función de "calcular", alimenta los eventos

resultantes a un observador corriente abajo, y alimenta la fuente difundida a la función de "PaginarAfuera". La función de "PaginarAfuera" también podría ser alimentada con más datos para tomar decisiones, a manera de ejemplo y no de limitación, y el resultado del cálculo se puede pasar al operador de "PaginarAfuera" para permitir la medición de la tasa de un evento y paginar afuera cuando la tasa se haya mantenido en cero durante un período prolongado previamente determinado. Al recibir una notificación asociada con el estado de cálculo actual del punto de control de la función de "PaginarAfuera", se pagan afuera los artefactos de tiempo de ejecución, y se repite el ciclo de vida.

A continuación se muestra un ejemplo de uso:

```
Paginar(fuente: xs,
15      PaginarAdentro: ys => Unir(ys.Primer(),
      Temporizador(t1)),
      calcular: ys => ys.F(),
      PaginarAfuera: ys => ys.Regular(t2))
```

Incluso la función de "PaginarAfuera" se puede paginar afuera hasta que la función de "PaginarAdentro" haya producido un mensaje. Es decir, el componente de reescritura 210 puede poner argumentos cargados de una forma diferida, tal como sigue:

```
Paginar(fuente: xs,
25      PaginarAdentro: ys => Unir(ys.Primer(),
```

```

    Temporizador(t1)),
    calcular: Diferir(algúnId1),
    PaginarAfuera: Diferir(algúnId2))

```

Aquí, "algúnId1" y "algúnId2" se definen como se describió
 5 anteriormente. Operacionalmente, este operador se suscribe a
 "xs" y paginará adentro el cálculo al recibir un primer evento o
 a la llegada de un evento predictivo. El cálculo se pagina afuera
 cuando la secuencia de entrada regulada recibe un evento; es
 decir, si un evento de entrada no fue seguido por otro evento
 10 dentro de un tiempo "t2", se propaga en la función de paginar
 afuera y hace una paginación afuera debido al tiempo de
 inactividad mayor que "t2".

El componente de actualización 240 se configura para
 actualizar o cambiar la operación de la paginación con base en
 15 los datos actuales o en la información obtenida a partir del
 componente de proceso 230 en respuesta a los datos históricos
 de ejecución de consulta capturados. El uso de artefactos
 definidos o de identificadores proporciona un nivel de
 direccionamiento indirecto que se puede utilizar para cambiar
 20 dinámicamente las funciones de paginación asociadas con el
 operador de "Paginar".

```

    Paginar(fuente: xs,
            PaginarAdentro: Diferir(algúnId3),
            calcular: Diferir(algúnId1),
            PaginarAfuera: Diferir(algúnId2))

```

25

En el fragmento anterior, también se ha insertado una carga diferida para "PaginarAdentro". Mientras que esto puede ser un desperdicio para una suscripción inicial, puede ser útil si la consulta hace la paginación afuera y de regreso adentro repetidas veces. En ese momento, el componente de reescritura 210 o un componente de actualización 240 independiente designado para las actualizaciones, puede volver a definir "algúnId3" y "algúnId2" en algún momento para hacer que el comportamiento de paginar afuera sea menos agresivo y/o para hacer que el comportamiento de paginar adentro sea más proactivo. Con el fin de habilitar la actualización, las expresiones que representan estas funciones de paginar no deben ponerse ávidamente en línea y por consiguiente, se presenta el uso de un operador de "Diferir".

La Figura 4 es una ilustración gráfica de una reescritura con funciones de recopilación de estadísticas y paginación basadas en los datos históricos. Empezando a la izquierda, se recibe una consulta de geo-cercas 400 inicial. La consulta de geo-cercas 400 establece una barrera virtual y genera un evento cuando "Bart sale de casa", basándose en los parámetros "Bart", "sale" y "casa". La consulta de geo-cercas 400 se puede reescribir para producir una versión paginable 410 de la consulta de la geo-cerca. La función de paginación puede ser una generalización de un operador de "Seleccionar Tantos". Aquí, la consulta de geo-cercas 400 se reescribe en

dos partes: la primera parte 420 representa el código que debe ser residente en la memoria, y la segunda parte 430 representa el código que no tiene que ser residente en la memoria excepto entre la presentación de un evento de paginación adentro y un evento de paginación afuera. El código de recopilación de estadísticas se agrega también a la segunda parte 430 de la consulta, el cual recolecta los datos estadísticos con respecto a la ejecución de la consulta y guarda los datos en el almacenamiento de estadísticas históricas 440. Las reescrituras también inyectan la funcionalidad de paginación adentro 450 y la funcionalidad de paginación afuera 460 que controla cuando la segunda parte 430 de la consulta se intercambia hacia adentro y hacia afuera de la memoria. Adicionalmente, la funcionalidad de paginación adentro 450 y la funcionalidad de paginación afuera 460 se determinan con base en los datos estadísticos. Como se muestra en el almacenamiento de estadísticas históricas, se pueden recopilar estadísticas basándose en la ejecución de la consulta previa observando cuando la geo-cerca se rompe normalmente para los parámetros "Bart", "sale" y "casa". Se puede determinar o deducir, con base en los datos, que la geo-cerca se rompe normalmente alrededor de las 8:00 a.m. De conformidad con lo anterior, la funcionalidad de paginación adentro 450 se puede configurar para desencadenarse antes de las 8:00 a.m. (por ejemplo, a las 7:45 a.m.), y la funcionalidad de la paginación afuera se puede

configurar para desencadenarse después de las 8:00 a.m. (por ejemplo, a las 8:15 a.m.). Utilizando un análisis más avanzado, tal como el aprendizaje de máquina, se puede inferir que el desencadenante de la consulta se da alrededor de las 8:00 a.m. en los días laborables y alrededor de las 10:00 a.m. los fines de semana. De esta manera, la secuencia de eventos de paginación adentro y paginación afuera puede cambiar desde un temporizador periódico de 24 horas (por ejemplo, cada día a las 7:45 a.m. paginar adentro) hasta una secuencia más compleja que represente el patrón descubierto.

Volviendo de nuevo brevemente a la Figura 1, observe que el componente optimizador 120 y el componente de paginación 130 se ilustran como entidades separadas. Es posible que el componente optimizador 120 pueda sugerir una optimización de consulta que sea esencialmente deshecha por el componente de paginación 130. En otras palabras, el componente de paginación 130 puede desoptimizar u optimizar a la inversa una consulta optimizada para hacer que la consulta sea paginable. Por ejemplo, el componente optimizador 120 puede delinear dobles operadores de "SeleccionarTantos", pero el componente de paginación 130 puede revertir y poner en línea los operadores. De acuerdo con otra modalidad, la funcionalidad del componente de paginación 130 se puede incrustar dentro del componente optimizador 120, de tal manera que el resultado del componente optimizador sea una consulta paginable

optimizada.

Los sistemas, arquitecturas, entornos y similares anteriormente mencionados, se han descrito con respecto a la interacción entre varios componentes. Se debe apreciar que
5 estos sistemas y componentes pueden incluir los componentes o subcomponentes que se especifican en los mismos, algunos de los componentes o subcomponentes especificados, y/o componentes adicionales. Los subcomponentes también se podrían implementar como componentes comunicativamente
10 acoplados a otros componentes, en lugar de incluirse dentro de los componentes progenitores. Todavía además, uno o más componentes o subcomponentes se pueden combinar en un solo componente para proporcionar una funcionalidad agregada. La comunicación entre sistemas, componentes y/o subcomponentes
15 se puede llevar a cabo de acuerdo con un modelo de empujar y/o de jalar (push & pull). Los componentes también pueden interactuar con uno o más componentes diferentes no específicamente descritos en el presente documento en aras de la brevedad, pero conocidos por las personas expertas en este
20 campo.

Además, diversas porciones de los sistemas que se dan a conocer anteriormente y de los métodos que se encuentran más adelante, pueden incluir o emplear inteligencia artificial, aprendizaje de máquina, o componentes, subcomponentes,
25 medios, procesos, metodologías, o mecanismos basados en

conocimientos o en reglas (por ejemplo, máquinas de vectores de soporte, redes neurales, sistemas expertos, redes de creencia bayesianas, lógica difusa, motores de fusión de datos, clasificadores...). Estos componentes, entre otras cosas, pueden automatizar ciertos mecanismos o procesos ejecutados mediante los mismos con el fin de hacer que las porciones de los sistemas y métodos sean más adaptables, así como eficientes e inteligentes. A manera de ejemplo y no de limitación, el componente de proceso 230 puede emplear estos mecanismos para deducir un período de tiempo activo para que una consulta en firme facilite la determinación de cuándo paginar adentro y paginar afuera.

Teniendo en cuenta los sistemas de ejemplo descritos, las metodologías que se pueden implementar, de acuerdo con la materia objeto que se da a conocer, serán mejor apreciadas con referencia a los diagramas de las Figuras 5 a 8. Mientras que para efectos de simplicidad de explicación, las metodologías se muestran y se describen como una serie de bloques, se debe entender y apreciar que la materia objeto reivindicada no está limitada por el orden de los bloques, debido a que algunos bloques se pueden presentar en diferentes órdenes y/o simultáneamente con otros bloques, de una manera distinta de lo que se ilustra y se describe en este documento. Por otra parte, pueden no requerirse todos los bloques ilustrados para implementar los métodos descritos posteriormente en la

presente.

Haciendo referencia a la Figura 5, se ilustra un método 500 para generar una consulta paginable en un sistema de procesamiento de eventos. En el numeral de referencia 510, se
5 recibe, se recupera, o se obtiene o se adquiere de otra manera una consulta en firme de procesamiento de eventos. La consulta en firme de procesamiento de eventos es continuamente evaluada sobre una secuencia o flujo de eventos a través del tiempo.

10 En el numeral 520, se inyecta en la consulta un desencadenante de paginación adentro. El desencadenante de paginación adentro identifica cuándo cargar una consulta, o una porción de la misma, en la memoria. Aunque el desencadenante de paginación adentro se puede definir después de la
15 presentación de un evento que sea procesado por una consulta, es preferible que el desencadenante de paginación adentro sea especificado antes de la presentación del evento que sea procesado por la consulta. Esto se puede determinar con base en los parámetros y las propiedades algebraicas de los
20 operadores de consultas. De una manera adicional o alternativa, se pueden utilizar los datos recolectados relacionados con la ejecución de la consulta para determinar y actualizar el desencadenante de paginación adentro.

En la referencia 530, se introduce la funcionalidad para
25 paginar adentro o, en otras palabras, para cargar en la

memoria, una consulta o una porción de la misma. De acuerdo con una implementación, la funcionalidad de paginación adentro puede aprovechar las facilidades del sistema de procesamiento de eventos, incluyendo la carga lenta o diferida de las

5 secuencias internas de un operador de "SeleccionarTantos", así como un mecanismo de puntos de control para cargar la consulta o la porción de la misma, incluyendo el código y el estado de la captura de datos. Adicionalmente, se puede inyectar la funcionalidad para adquirir los datos con respecto a

10 la ejecución de la consulta. A partir de estos datos, se pueden predecir los tiempos activos, entre otras cosas, y se pueden utilizar para establecer o actualizar los desencadenantes de paginación adentro y paginación afuera.

Se inyecta un desencadenante de paginación afuera en el

15 numeral 540. El desencadenante de paginación afuera indica cuándo se puede retirar o descargar la consulta desde la memoria. Por ejemplo, esto puede ser después de un tiempo previamente determinado o después de un tiempo de inactividad previamente determinado, entre otras cosas. Además, el

20 desencadenante de paginación afuera se puede generar y actualizar basándose en los datos de ejecución recolectados, tales como los tiempos de actividad predichos o el tiempo de procesamiento.

En el numeral de referencia 550, se puede introducir la

25 funcionalidad para paginar afuera la consulta. De acuerdo con

una implementación, la funcionalidad de paginación afuera puede aprovechar la persistencia de estado asociada con los puntos de control proporcionados por el sistema de procesamiento de eventos. Por ejemplo, el código y el estado
5 del código se pueden guardar en un almacenamiento de datos persistente. Posteriormente, la consulta se puede retirar o descargar de la memoria. Esto se puede hacer directamente, por ejemplo, mediante la desasignación de memoria, o indirectamente, por ejemplo, mediante el establecimiento de un
10 puntero nulo o mediante la disminución de un contador de referencia hasta cero para iniciar la recolección de basura.

La Figura 6 representa un método 600 para paginar la consulta adentro y afuera de la memoria. En la referencia 610, se detecta un desencadenante de paginación adentro. En la
15 referencia 620, se pagina adentro una consulta en firme de procesamiento de eventos, o una porción de la misma, que comprende el código y potencialmente el estado de la captura de datos. En otras palabras, el código requerido para evaluar una consulta sobre los datos que previamente no estaba en la
20 memoria, se carga en la memoria. De acuerdo con una implementación, se puede emplear la funcionalidad de un sistema de procesamiento de consultas que pertenece a la carga diferida del código, para paginar adentro una consulta. En el numeral de referencia 630, se puede detectar un
25 desencadenante de paginación afuera. A continuación, en el

numeral 640, se puede paginar afuera la consulta o una porción de la misma. En una implementación, se pueden utilizar las facilidades de los puntos de control de un sistema de procesamiento de consultas para guardar el código y el estado en un almacenamiento de datos persistente. Posteriormente, la consulta se puede retirar de la memoria directamente mediante la desasignación de memoria, o indirectamente estableciendo un puntero en nulo, o disminuyendo un contador de referencia para iniciar la recolección de basura. De acuerdo con una modalidad, no se necesita paginar afuera toda la consulta, o una porción de la misma, sino que más bien se puede paginar afuera un subconjunto de la consulta. Por ejemplo, en una situación donde hay consultas anidadas, se puede emplear una estrategia para paginar afuera selectivamente las partes en cualquier nivel, tal como paginar afuera el subárbol de más alto consumo de memoria.

La Figura 7 ilustra un método de paginación adentro de una consulta. En el numeral de referencia 710, se lee un identificador. De acuerdo con un aspecto de la materia objeto que se da a conocer, se puede emplear un nivel de direccionamiento indirecto en donde únicamente hay una referencia a una consulta, a saber, un identificador, residente en la memoria. Esto proporciona un mecanismo para diferir la carga de la consulta. En el numeral de referencia 720, se localiza una consulta o una porción de la misma, que está

asociada con el identificador. Por ejemplo, se puede guardar la consulta en una estructura de almacenamiento de diccionario o en un almacenamiento de valores clave. En este caso, el identificador se utiliza como la clave para adquirir la consulta almacenada como datos. En la referencia 730, la consulta o la porción de la misma se compila en código ejecutable. En el numeral 740, el código ejecutable está en línea en la memoria. Dicho de una manera diferente, el código ejecutable se carga en la memoria reemplazando al identificador. Por último, en el numeral de referencia 750, se adquiere el estado relacionado con la consulta a partir del almacenamiento persistente y se restaura. Desde luego, si es la primera vez que se ha paginado adentro esta consulta, el estado puede ser algún estado inicial o ningún estado en absoluto.

La Figura 8 es un diagrama de flujo de un método de paginación afuera 800 de una consulta. En el numeral de referencia 810, se pone el punto de control de la consulta o de la porción de la misma. Esto puede corresponder a guardar en un almacenamiento persistente un identificador relacionado con la consulta, la consulta por sí misma, y el estado de la consulta. De acuerdo con una modalidad, esta funcionalidad de puntos de control se puede proporcionar mediante un sistema de procesamiento de eventos y se puede invocar simplemente para usarse en la paginación afuera de una consulta. En el numeral de referencia 820, se identifican porciones de la

consulta para retirarse de la memoria. De acuerdo con una modalidad, se puede retirar de la memoria sustancialmente la consulta entera o todas las porciones, excepto cualesquiera ganchos necesarios para la paginación adentro de la consulta.

- 5 De acuerdo con otra modalidad, se retira de la memoria únicamente un subconjunto de la consulta, tal como la porción de más alto consumo de memoria. En un caso, las consultas se pueden anidar, y se puede implementar una estrategia para retirar selectivamente porciones en cualquier nivel. En el
- 10 numeral de referencia 830, se retiran de la memoria las porciones identificadas de la consulta. Por ejemplo, la consulta se puede cancelar directamente de la memoria o se puede recolectar indirectamente como basura mediante el establecimiento de un puntero de referencia en nulo o mediante
- 15 la disminución de un contador de referencia hasta cero.

Los datos que pueden ser representados como una secuencia de eventos o un flujo de eventos, se pueden utilizar como parte de un proceso de decisión para una capacidad de paginación adentro y paginación. En un caso particular, los

20 eventos del sistema, por ejemplo, aquéllos asociados con la memoria física y la unidad de procesamiento central, se pueden representar como una secuencia de eventos. De una manera convencional, los mecanismos de paginación basados en el sistema operativo tratan exclusivamente con este tipo de

25 métricas del sistema. Estas métricas son secundarias al

proceso de decisión específico del dominio descrito en el presente documento. En efecto, el conocimiento del dominio reduce los casos de borde donde se utilizan métricas del nivel del sistema para paginar las consultas adentro y afuera de la memoria. Sin embargo, para las tormentas de eventos, que son situaciones donde un gran número de eventos se producen durante un período relativamente corto de tiempo, o en los tiempos cuando no es bueno el poder predictivo, el sistema de procesamiento de eventos puede recurrir de nuevo a la paginación con base en los eventos del sistema para evitar la sobrecarga. Además, en un entorno distribuido, se puede utilizar el procesamiento de eventos basado en los eventos del sistema para activar la equilibración de la carga mediante la migración de la computación hacia diferentes máquinas.

Los aspectos de este documento de divulgación son aplicables al procesamiento distribuido a gran escala como un procesamiento de dispositivos a pequeña escala. Por ejemplo, los aspectos de esta divulgación se pueden implementar en conjunto con un sistema distribuido de procesamiento de eventos o con un servicio que procese muchas consultas de muchos usuarios. En este escenario, la paginación puede aumentar la densidad de memoria para ayudar a procesar una gran cantidad de consultas. Los aspectos también son aplicables a un solo dispositivo de computación, tal como un teléfono. Aquí, la paginación se puede emplear para retirar una

o más consultas de los usuarios desde la memoria y para un procesamiento en segundo plano con el fin de conservar la energía de la batería, por ejemplo.

La presente invención es compatible con diversos
5 productos y procesos que lleven a cabo, o que se configuren para llevar a cabo, diversas acciones con respecto a la paginación en un sistema de procesamiento de eventos. En lo siguiente se presentan uno o más sistemas y métodos de ejemplo.

10 Un sistema comprende un procesador acoplado a una memoria, en donde el procesador se configura para ejecutar los siguientes componentes ejecutables por computadora almacenados en la memoria: un primer componente configurado para recibir una consulta de procesamiento de eventos; y un
15 segundo componente configurado para reescribir la consulta de procesamiento de eventos en una forma paginable que soporta el intercambio de cuando menos una porción de la consulta hacia adentro y hacia afuera de la memoria. El sistema reivindicado también comprende un tercer componente
20 configurado para inyectar un desencadenante de paginación adentro que inicia la carga de cuando menos la porción de la consulta de procesamiento de eventos en la memoria. En un caso, el desencadenante de paginación adentro se genera basándose en los datos históricos relacionados con la ejecución
25 de la consulta de procesamiento de eventos. El sistema además

puede comprender un cuarto componente configurado para inyectar un desencadenante de paginación afuera que inicie el retiro de la cuando menos una porción de la consulta de procesamiento de eventos desde la memoria. En un caso, el desencadenante de paginación afuera se genera basándose en los datos históricos relacionados con la ejecución de la consulta de procesamiento de eventos. El sistema además puede comprender un quinto componente configurado para recolectar los datos con respecto a la ejecución de la consulta de procesamiento de eventos.

Un método comprende emplear cuando menos un procesador configurado para ejecutar las instrucciones ejecutables por computadora almacenadas en una memoria para llevar a cabo las siguientes acciones: recibir una consulta de procesamiento de eventos; y generar una consulta paginable con base en la consulta de procesamiento de eventos, en donde la consulta paginable soporta el intercambio de cuando menos una porción de la consulta de procesamiento de eventos hacia adentro y hacia afuera de la memoria. El método además comprende inyectar un desencadenante de paginación adentro para iniciar la carga de la cuando menos una porción de la consulta de procesamiento de eventos en la memoria. El método además comprende inyectar un desencadenante de paginación afuera para iniciar el retiro de la cuando menos una porción de la consulta de procesamiento de eventos desde la memoria. El

método además comprende introducir un mecanismo para diferir la carga de la cuando menos una porción de la consulta de procesamiento de eventos en la memoria hasta que sea desencadenada. El método además comprende introducir un

5 mecanismo para recolectar los datos relacionados con la ejecución de la consulta de procesamiento de eventos. El método además comprende emplear el aprendizaje de máquina con respecto a los datos para inferir cuando menos uno de un desencadenante de paginación adentro o un desencadenante de

10 paginación afuera. El método además comprende introducir un mecanismo para guardar el estado de la por lo menos una porción de la consulta de procesamiento de eventos en relación con el intercambio de la consulta de procesamiento de eventos hacia afuera de la memoria. El método además comprende

15 introducir un mecanismo para recuperar el estado de la cuando menos una porción de la consulta de procesamiento de eventos en relación con el intercambio en la consulta de procesamiento de eventos hacia adentro de la memoria. El método además comprende introducir un mecanismo para intercambiar hacia

20 afuera de la memoria un subconjunto de la por lo menos una porción de la consulta de procesamiento de eventos.

Un medio de almacenamiento legible por computadora que tiene instrucciones almacenadas en el mismo, las cuales hacen posible que por lo menos un procesador lleve a cabo un método

25 cuando se ejecutan las instrucciones, en donde el método

comprende: detectar un desencadenante de paginación adentro con respecto a una secuencia de eventos; y cargar cuando menos una porción de una consulta de procesamiento de eventos en la memoria en respuesta a la detección del desencadenante de paginación adentro. El método para cargar la consulta de procesamiento de eventos puede comprender la localización de una representación de los datos de la por lo menos una porción de la consulta de procesamiento de eventos con base en un identificador; y compilar la representación de los datos en un código ejecutable. El método puede comprender además detectar un desencadenante de paginación afuera con respecto a la secuencia de eventos; y descargar la cuando menos una porción de la consulta de procesamiento de eventos desde la memoria en respuesta a la detección del desencadenante de paginación afuera. El método para descargar la consulta de procesamiento de eventos puede comprender iniciar una operación de captura de estado, la cual captura el estado actual de la consulta de procesamiento de eventos. El método para descargar la consulta de procesamiento de eventos puede comprender la descarga de únicamente un subconjunto de la por lo menos una porción de la consulta de procesamiento de eventos.

Los aspectos de la presente invención pertenecen al problema técnico de gestión de memoria de computadora en conjunto con el procesamiento de consultas en firme. Las

características técnicas asociadas con la resolución de este problema incluyen el empleo de un sistema de procesamiento de eventos para gestionar la memoria reescribiendo consultas para soportar la paginación adentro y afuera de la memoria. De conformidad con lo anterior, los aspectos de la divulgación exhiben los efectos técnicos con respecto al uso eficiente de los recursos, incluyendo, pero no limitándose a, la conservación de la memoria y una mayor velocidad de procesamiento.

Las palabras "de ejemplo", o las diferentes formas de las mismas, se utilizan en el presente documento para significar que sirve como ejemplo, instancia o ilustración. Cualquier aspecto o diseño descrito en este documento como "ejemplar" no debe interpretarse necesariamente como preferido o conveniente con respecto a otros aspectos o diseños. Además, los ejemplos se proporcionan exclusivamente con fines de claridad y comprensión, y no están destinados para limitar o restringir la materia objeto reivindicada o porciones pertinentes de esta divulgación en cualquier forma. Se debe apreciar que se podrían haber presentado grandes cantidades de ejemplos adicionales o alternativos de un alcance variable, pero se han omitido para fines de brevedad.

Como se utilizan en el presente documento, los términos "componente" y "sistema", así como diversas formas de los mismos (por ejemplo, componentes, sistemas, subsistemas...) pretenden referirse a una entidad relacionada con la

computadora, ya sea hardware, una combinación de hardware y software, o software en ejecución. Por ejemplo, un componente puede ser, pero no se limita a ser, un proceso que se ejecute en un procesador, un procesador, un objeto, una instancia, un ejecutable, un hilo de ejecución, un programa y/o una computadora. A manera de ilustración, tanto una aplicación que se ejecute en una computadora como la computadora, puede ser un componente. Uno o más componentes pueden residir dentro de un proceso y/o hilo de ejecución, y un componente se puede localizar en una computadora y/o se puede distribuir entre dos o más computadoras.

La conjunción "o", como se utiliza en esta descripción y en las reivindicaciones adjuntas, pretende significar un "o" inclusivo en lugar de un "o" exclusivo, a menos que se especifique de otra manera o que quede claro a partir del contexto. En otras palabras, "'X' o 'Y'" pretenden significar cualesquiera permutaciones inclusivas de "X" e "Y". Por ejemplo, si "'A' emplea 'X'", "'A' emplea 'Y'", o "'A' emplea tanto 'X' como 'Y'", entonces se cumple "'A' emplea 'X' o 'Y'" en cualquiera de las instancias anteriores.

Además, en la medida en que se utilicen los términos "incluye", "contiene", "tiene", "que tiene", o las variaciones en la forma de los mismos, ya sea en la descripción detallada o bien en las reivindicaciones, se pretende que estos términos sean inclusivos de una manera similar al término "que

comprende" como se interpreta "que comprende" cuando se emplea como una palabra de transición en una reivindicación.

Con el fin de proporcionar un contexto para la materia objeto reivindicada, la Figura 9, así como la siguiente
5 discusión, pretenden proporcionar una breve descripción general de un entorno adecuado en donde se pueden implementar diversos aspectos de la materia objeto. Sin embargo, el entorno adecuado es solamente un ejemplo y no pretende sugerir ninguna limitación con respecto al alcance de
10 uso o funcionalidad.

Aunque el sistema y los métodos que se dan a conocer anteriormente se pueden describir en el contexto general de instrucciones ejecutables por computadora de un programa que se ejecuta en una o más computadoras, los expertos en la
15 materia reconocen que también se pueden implementar algunos aspectos en combinación con otros módulos de programas o similares. Generalmente, los módulos de programas incluyen rutinas, programas, componentes, estructuras de datos, entre otras cosas, que realizan tareas particulares y/o implementan
20 tipos particulares de datos abstractos. Por otra parte, aquellos expertos en este campo podrán apreciar que los sistemas y métodos anteriores se pueden practicar con diferentes configuraciones de sistemas de computadora, incluyendo sistemas de computación un solo procesador, de múltiples
25 procesadores, o de un procesador con múltiples núcleos, mini-

dispositivos de computación, computadoras centrales, así como computadoras personales, dispositivos de computación portátiles (por ejemplo, asistente personal digital (PDA), teléfono, reloj...), electrónica de consumo o industrial basada en microprocesador o programable, y similares. Los aspectos también se pueden practicar en entornos de computación distribuidos en donde las tareas son realizadas por dispositivos de procesamiento remoto que estén vinculados a través de una red de comunicaciones. Sin embargo, algunos, si no es que todos los aspectos de la materia objeto reivindicada, se pueden practicar en computadoras independientes. En un entorno de computación distribuido, los módulos de programas se pueden localizar en uno o ambos de los dispositivos de memoria locales y remotos.

Con referencia a la Figura 9, se ilustra una computadora para propósitos generales de ejemplo o un dispositivo de computación 902 (por ejemplo, de escritorio, laptop, tablet, reloj, servidor, de mano, electrónica de consumo o industrial programable, decodificador, sistema de juegos, nodo de computación...). La computadora 902 incluye uno o más procesadores 920, la memoria 930, la barra colectora (bus) del sistema 940, dispositivos de almacenamiento masivo 950, y uno o más componentes de interfaz 970. La barra colectora (bus) del sistema 940 acopla de una manera comunicativa por lo menos los componentes del sistema anteriores. Sin embargo, se

debe apreciar que, en su forma más simple, la computadora 902 puede incluir uno o más procesadores 920 acoplados a la memoria 930, los cuales ejecutan diversas acciones ejecutables por computadora, instrucciones, y/o componentes almacenados en la memoria 930.

Los procesadores 920 se pueden implementar con un procesador para propósitos generales, un procesador de señales digitales (DSP), un circuito integrado de aplicación específica (ASIC), un campo de matriz de puertas programables (FPGA), u otro dispositivo lógico programable, puerta discreta o lógica de transistor, componentes de hardware discretos, o cualquier combinación de los mismos diseñada para llevar a cabo las funciones descritas en el presente documento. Un procesador para propósitos generales puede ser un microprocesador, pero en la alternativa, el procesador puede ser cualquier procesador, controlador, microcontrolador, o máquina de estado. Los procesadores 920 también se pueden implementar como una combinación de dispositivos de computación, por ejemplo, una combinación de un procesador de señales digitales (DSP) y un microprocesador, una pluralidad de microprocesadores, procesadores de múltiples núcleos, uno o más microprocesadores en conjunto con un núcleo de procesador de señales digitales (DSP), o cualquier otra de estas configuraciones. En una modalidad, los procesadores pueden ser un procesador de gráficos.

La computadora 902 puede incluir o de otra manera interactuar con una variedad de medios legibles por computadora para facilitar el control de la computadora 902 con el fin de implementar uno o más aspectos de la materia objeto
5 reivindicada. Los medios legibles por computadora pueden ser cualquier medio disponible al que se pueda acceder mediante la computadora 902, e incluyen los medios volátiles y no volátiles, y los medios extraíbles y no extraíbles. Los medios legibles por computadora pueden comprender dos tipos distintos y
10 mutuamente excluyentes, es decir, medios de almacenamiento de computadora y medios de comunicación.

Los medios de almacenamiento de computadora incluyen los medios volátiles y no volátiles, extraíbles y no extraíbles, implementados en cualquier método o tecnología para el
15 almacenamiento de información, tal como instrucciones legibles por computadora, estructuras de datos, módulos de programas, u otros datos. Los medios de almacenamiento de computadora incluyen dispositivos de almacenamiento tales como dispositivos de memoria (por ejemplo, memoria de acceso
20 aleatorio (RAM), memoria de sólo lectura (ROM), memoria de sólo lectura programable y borrable eléctricamente (EEPROM)...), dispositivos de almacenamiento magnéticos (por ejemplo, discos duros, disquetes, casetes, cintas...), discos ópticos (por ejemplo, un disco compacto (CD), disco versátil
25 digital (DVD)...), y los dispositivos de estado sólido (por

ejemplo, unidad de estado sólido (SSD), una unidad de memoria flash (por ejemplo, tarjeta, memoria USB, unidad de claves...)...), o cualesquiera otros medios similares que almacenen, en oposición a que transmitan o comuniquen, la
5 información deseada accesible mediante la computadora 902. De conformidad con lo anterior, los medios de almacenamiento de computadora excluyen las señales de datos moduladas, así como aquéllos descritos con respecto a los medios de comunicación.

10 Los medios de comunicación incorporan instrucciones legibles por computadora, estructuras de datos, módulos de programas, u otros datos, en una señal de datos modulada, tal como una onda portadora u otro mecanismo de transporte, e incluyen cualquier medio de entrega de información. El término
15 "señal de datos modulada" significa una señal que tiene una o más de sus características fijadas o cambiadas de una manera tal como para codificar información en la señal. A manera de ejemplo, y no de limitación, los medios de comunicación incluyen los medios cableados, tales como una red cableada o
20 una conexión de cableado directo, y los medios inalámbricos, tales como los medios acústicos, de radiofrecuencia (RF), infrarrojos, y otros medios inalámbricos.

Los dispositivos de memoria 930 y de almacenamiento masivo 950 son ejemplos de los medios de almacenamiento
25 legibles por computadora. Dependiendo de la configuración

exacta y del tipo de dispositivo de computación, la memoria 930 puede ser volátil (por ejemplo, RAM), no volátil (por ejemplo, ROM, memoria flash...) o alguna combinación de las dos. A manera de ejemplo, el sistema básico de entrada/salida (BIOS),
5 que incluye las rutinas básicas para transferir información entre los elementos dentro de la computadora 902, tal como durante el arranque, se puede almacenar en la memoria no volátil, mientras que la memoria volátil puede actuar como una memoria caché externa para facilitar el procesamiento mediante los
10 procesadores 920, entre otras cosas.

Los dispositivos de almacenamiento masivo 950 incluyen medios de almacenamiento de computadora extraíbles/no extraíbles, volátiles/no volátiles para el almacenamiento de grandes cantidades de datos en relación con la memoria 930.
15 Por ejemplo, los dispositivos de almacenamiento masivo 950 incluyen, pero no se limitan a, uno o más dispositivos, tales como una unidad de disco magnético u óptico, una unidad de disquete, una memoria flash, una unidad de estado sólido, o una memoria USB.

20 Los dispositivos de memoria 930 y de almacenamiento masivo 950 pueden incluir, o pueden tener almacenados en los mismos, el sistema operativo 960, una o más aplicaciones 962, uno o más módulos de programas 964, y datos 966. El sistema operativo 960 actúa para controlar y asignar los recursos de la
25 computadora 902. Las aplicaciones 962 incluyen uno o ambos

de software del sistema y de la aplicación, y pueden aprovechar la gestión de recursos por medio del sistema operativo 960 a través de los módulos de programas 964 y los datos 966 almacenados en los dispositivos de memoria 930 y/o de
5 almacenamiento masivo 950, para llevar a cabo una o más acciones. De conformidad con lo anterior, las aplicaciones 962 pueden convertir una computadora para propósitos generales 902 en una máquina especializada de acuerdo con la lógica proporcionada por las mismas.

10 Toda o una porción de la materia objeto reivindicada se puede implementar usando las técnicas convencionales de programación y/o ingeniería para producir software, firmware, hardware, o cualquier combinación de los mismos, con el fin de controlar una computadora para llevar a cabo la funcionalidad
15 que se da a conocer. A manera de ejemplo y no de limitación, el sistema de procesamiento de eventos 100 o las porciones del mismo, pueden ser, o pueden formar parte, de una aplicación 962, e incluyen uno o más módulos 964 y los datos 966 almacenados en los dispositivos de memoria 930 y/o de
20 almacenamiento masivo 950, cuya funcionalidad se puede producir cuando son ejecutados por medio de uno o más procesadores 920.

De acuerdo con una modalidad particular, los procesadores 920 pueden corresponder a un sistema en chip
25 (SoC) o a una arquitectura similar, incluyendo, o en otras

palabras, la integración tanto de hardware como de software en un solo sustrato de circuito integrado. En este escenario, los procesadores 920 pueden incluir uno o más procesadores, así como una memoria, por lo menos similares a los procesadores 5 920 y a la memoria 930, entre otras cosas. Los procesadores convencionales incluyen una cantidad mínima de hardware y software y se basan en gran medida en el hardware y el software externos. Por el contrario, una implementación de procesador de sistema en chip (SoC) es más potente, debido a 10 que incorpora hardware y software en el mismo, los cuales hacen posible tener una funcionalidad particular con una mínima o ninguna dependencia de hardware y software externos. Por ejemplo, el sistema de procesamiento de eventos 100 y/o la funcionalidad asociada, se pueden incorporar dentro 15 del hardware en una arquitectura de sistema en chip (SoC).

La computadora 902 también incluye uno o más componentes de interfaz 970 que se acoplan de una manera comunicativa a la barra colectora (bus) del sistema 940, y facilitan la interacción con la computadora 902. A manera de 20 ejemplo, el componente de interfaz 970 puede ser un puerto (por ejemplo, en serie, en paralelo, PCMCIA, USB, Fire Wire...) o una tarjeta de interfaz (por ejemplo, de sonido, video...), o similar. En una implementación de ejemplo, el componente de interfaz 970 se puede incorporar como una interfaz de 25 entrada/salida de usuario para hacer posible que un usuario

introduzca comandos e información en la computadora 902, por ejemplo, a través de uno o más gestos o voz, a través de uno o más dispositivos de entrada (por ejemplo, dispositivo de puntero, tal como un ratón, trackball, lápiz, teclado táctil, 5 teclado, micrófono, joystick, control de juegos, antena satelital, escáner, cámara, otra computadora...). En otra implementación de ejemplo, el componente de interfaz 970 se puede incorporar como una interfaz periférica de salida para proporcionar la salida a pantallas (LCD, LED, plasma...), altavoces, impresoras, 10 y/u otras computadoras, entre otras cosas. Todavía además, el componente de interfaz 970 se puede incorporar como una interfaz de red para permitir la comunicación con otros dispositivos de computación (no mostrados), tal como sobre un enlace de comunicaciones por cable o inalámbrico.

15 Lo que se ha descrito anteriormente incluye ejemplos de los aspectos de la materia objeto reivindicada. Desde luego, no es posible describir cada combinación concebible de componentes o metodologías para los fines de descripción de la materia objeto reivindicada, pero un experto ordinario en la 20 técnica puede reconocer que son posibles muchas otras combinaciones y permutaciones de la materia objeto que se da a conocer. De conformidad con lo anterior, la materia objeto dada a conocer pretende abarcar todas las alteraciones, modificaciones y variaciones que caigan dentro del espíritu y 25 alcance de las reivindicaciones adjuntas.