BI en tiempo real

Àlex Caminals Sánchez de la Campa

PID_00197295

Universitat Oberta de Catalunya

Índice

Introducción

Una de las tendencias actuales más interesantes dentro del mundo de la inteligencia de negocio (BI) es el acceso en tiempo real a la información. El objetivo de este módulo es ofrecer información sobre el BI en tiempo real a nivel de organización sin entrar de lleno en la vertiente tecnológica. A pesar de esto, se introducen ciertos aspectos tecnológicos, que se consideran clave para poder entender cómo se comporta un sistema BI en tiempo real y qué diferencias existen entre este y un sistema BI tradicional.

En este módulo se explican los conceptos básicos del BI en tiempo real y las principales diferencias con respecto a un sistema BI tradicional, las diferentes alternativas de diseño y los beneficios e inconvenientes de cada una de ellas. Se proporciona información útil para poder determinar la tecnología necesaria para la implementación de una solución BI en tiempo real. Se incluyen también ejemplos y escenarios para un mayor entendimiento de cada una de las situaciones expuestas. Además, cuenta también con información sobre cómo calcular métricas de evaluación del proyecto como el ROI y el Payback Period, aplicados a un proyecto de estas características. Finamente, se incluyen casos de éxito que ofrecen una visión real del BI en tiempo real.

1. Conceptos básicos

1.1. Definición

Podemos definir conceptualmente un sistema de BI en tiempo real como un sistema donde los datos que se permiten analizar incluyen tanto información histórica como información que acaba de ocurrir. Es decir, un sistema de BI donde el desfase entre el almacenamiento de los datos en el sistema operacional y decisional es cercano a cero.

Una vez definido conceptualmente un sistema de BI en tiempo real, cabe preguntarnos en qué situaciones este tipo de sistema es viable y necesario. Para ello, tendremos en cuenta los tipos de sistemas de BI que existen en función del tipo de análisis a realizar y los perfiles de los usuarios a quienes va dirigido:

- Sistema de BI estratégico
- Sistema de BI táctico
- Sistema de BI operacional

Cada uno de ellos es necesario para cubrir unas necesidades a nivel de organización. Así se muestra en el siguiente diagrama:

Dentro de lo que son los dos niveles superiores (estratégico y táctico), la necesidad de contener datos actualizados con respecto al sistema operacional es, en la gran mayoría de los casos, nula. Esto se debe al hecho de que la mayoría de decisiones estratégicas (en una organización) y tácticas (a nivel de departamento), no suelen modificarse por unos datos concretos ocurridos en el pasado reciente (menos de un día), sino que suelen ser decisiones basadas en el

análisis de los datos en un periodo de tiempo bastante amplio, donde cierto patrón se repite de manera recurrente o donde se visualizan tendencias a lo largo del tiempo.

Sin embargo, para el BI de tipo operacional, esa necesidad se hace patente en algunos casos. Para ello debemos recordar qué es el BI operacional, cómo puede ser utilizado en una organización y qué beneficios aporta a esta. Después estudiaremos en qué casos un sistema operacional necesita de tiempo real, y finalmente definiremos los sistemas de BI operacional en tiempo real.

1.1.1. BI operacional

El BI operacional proporciona información a los empleados encargados del día a día de los procesos de una organización para poder reaccionar de manera rápida y eficiente a las diferentes situaciones derivadas de su trabajo diario. Estos usuarios representan a la gran mayoría de los empleados en una organización.

Para un usuario operacional, es necesario poder acceder a datos que reflejen la realidad del momento. No disponer de los datos del día en curso, sino tan solo de una imagen que contenga los datos de hasta el día anterior, puede suponer no ser capaces de tomar las decisiones correctas a nivel de proceso en sus tareas diarias.

Esta es la gran diferencia a nivel de datos entre el BI operacional y el estratégico y táctico, donde el hecho de no disponer de los últimos datos del sistema no supone, generalmente, un impedimento para la toma de decisiones, como ya se ha indicado anteriormente.

Una vez definido qué es el BI operacional, debemos identificar la necesidad de disponer de datos actualizados para la toma de decisiones. Para ello, es importante definir el tiempo de acción ($t_{acción}$).

1.1.2. Tiempo de acción

Definimos $t_{acción}$ como el tiempo necesario para un usuario o proceso para reaccionar a una situación concreta. Este concepto cobra una importancia muy relevante dentro del BI operacional, debido a que aquí encontramos los procesos de negocio con un $t_{acción}$ menor, cosa que afectará a la frecuencia de las cargas de datos.

Respuesta a casos abiertos en un centro de soporte

En un centro de soporte 7x24, donde se reciben peticiones de soporte de un paquete de software, hay un acuerdo de nivel de servicio con el cliente, de manera que el tiempo máximo de respuesta a peticiones de servicio abiertas debe ser menor de 15'.

Se ha observado que, cuando se publica una nueva versión del software y los usuarios empiezan a descargarla e instalarla, el número de peticiones de soporte de servicio aumenta.

Con un número constante de agentes atendiendo estas peticiones de servicio, cuando el número de peticiones aumenta, el tiempo de espera de los usuarios aumenta. Es decir, tanto los usuarios que llaman por teléfono como los que se comunican con el centro de soporte vía chat, deben esperar más tiempo para ser atendidos.

Dada esta situación, la solución radica en aumentar el número de agentes asignados a atender estas peticiones. Sin embargo, el encargado del turno del centro de soporte debe balancear esta necesidad con las necesidades de otras tareas que puedan llevarse a cabo en el centro de soporte.

En resumen, el encargado del centro de soporte, para poder tomar las decisiones correctas necesita obtener la siguiente información:

- Información actual:
  - El número de casos abiertos incluyendo sus detalles y su evolución desde la publicación de la nueva versión, y para las otras campañas abiertas en el centro de soporte.
  - Los agentes asignados a contestar las peticiones de servicio y su rendimiento actual.
- Información histórica:
  - Evolución del número de casos abiertos cuando hay una publicación de una nueva versión de software, y para las otras campañas abiertas en el centro de soporte.
  - Rendimiento histórico de los agentes actualmente asignados y de otros agentes que pueden añadirse al equipo.

En este caso, queda patente la necesidad de consultar información histórica y actual.

Para poder responder a las peticiones de servicio dentro del tiempo acordado con el cliente en el caso de ejemplo, se ha decidido fijar el $t_{acción}$ en 5', de manera que el responsable del centro de soporte tenga tiempo de balancear la carga de trabajo de las diferentes campañas activas.

Por tanto, cada 5' el encargado recibirá un informe con información histórica y actual para poder tomar las decisiones correctas por lo que respecta a la gestión del centro de soporte. Es decir, $\mathbf{t_{acción} = 5'}$.

Envío de productos a clientes

Una empresa vende sus productos en su página web a sus clientes. Cada vez que se hace una venta *online*, la empresa debe enviar el producto al cliente. En este caso, la planificación de la ruta del distribuidor se hace al final de la jornada, una vez hemos acumulado todos los pedidos del día. El motivo es que no tiene sentido reaccionar a cada venta para disparar un proceso que recalcule la ruta del distribuidor.

Por tanto, en este caso el $t_{acción}$ para el cálculo de la ruta del distribuidor no será de minutos, sino que es más elevado. De hecho, tan solo necesitaremos esa información al final del día.

Al darse esta situación, a pesar de hablar de un proceso de negocio operacional (del día a día), no habrá la necesidad de combinar información histórica y actual durante el día, sino que podríamos esperar a realizar la carga de datos al final del día para calcular la ruta del distribuidor. Es decir, $\mathbf{t_{acción} = 24\ horas}$.

1.1.3. Necesidad de información en tiempo real

Cuando el $t_{acción}$ tiende a 0, estamos en una situación de necesidad de información en tiempo real. Es decir, la información debe estar disponible para su análisis casi al mismo momento en que esta información es almacenada en un sistema de información.

Por tanto, podemos concluir que:

Existe necesidad de acceso a la información en tiempo real cuando el $t_{acción}$ para un proceso dentro de una organización es próximo a 0.

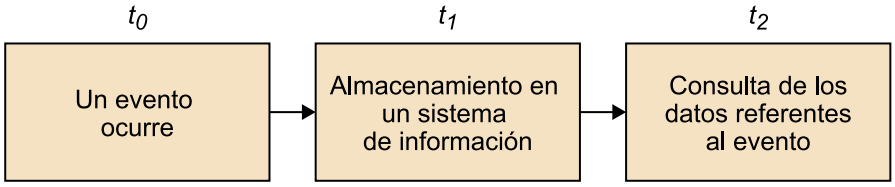
Supongamos un $t_{acción}$ de 5' y 24 horas para cada uno de los dos ejemplos anteriores. Teniendo en cuenta estos datos, podemos concluir lo siguiente:

Proceso	$t_{acción}$	Acceso a información en tiempo real
Respuesta a casos abiertos en un centro de soporte	5'	Sí
Envío de productos a clientes	24 horas	No

1.1.4. BI operacional en tiempo (casi) real

El acceso a la información en tiempo real es un mito. Siempre habrá un pequeño retraso entre un evento (t_0), el momento en que se almacena ese evento en un sistema de información (t_1) y la consulta de esos datos (t_2). Es decir, $t_0 < t_1 < t_2$.

Ciclo de vida de un evento hasta la consulta de sus datos



Es por eso por lo que, si quisiéramos ser puristas, deberíamos hablar de acceso a la información en tiempo "casi" real. Sin embargo, la expresión *tiempo real* ha ganado muchos adeptos en la comunidad internacional, con lo que ha pasado a ser una expresión estándar.

1.2. Acceso a la información según las necesidades de los usuarios

Generalmente, si se ofrece a un usuario la posibilidad de acceder a la información en tiempo real, o se le pregunta si así lo desea, el resultado siempre será el mismo:

"Sí, por favor, ¡quiero acceso en tiempo real a la información!"

Teniendo en cuenta la complejidad añadida que conlleva para un sistema BI el acceso a los datos en tiempo real, conviene abordar esta cuestión como una pregunta no tan directa. El objetivo es identificar la necesidad (si esta realmente existe), sin abrir la caja de Pandora.

Empezaremos viendo los factores a tener en cuenta a la hora de decidir el tipo de acceso a la información que deberemos realizar, en función del tipo de análisis a realizar (estratégico, táctico u operacional). Después analizaremos los distintos escenarios que podemos encontrarnos en función de estos factores, y vamos a ver cómo vamos a acceder a la información en cada uno de ellos. Finalmente, miraremos el contexto histórico detrás de esta necesidad para entender por qué ahora las organizaciones se están planteando el análisis conjunto de información histórica y actual, con la consiguiente necesidad de acceso a la información en tiempo real.

1.2.1. Factores de necesidad de información

Como ya hemos introducido, la necesidad de acceso a la información en tiempo real debe determinarse en función del $t_{acción}$ de los procesos a analizar por el usuario.

Sin embargo, para determinar el tipo de acceso a la información requerido, será necesario tener en cuenta también otros factores.

Los factores a tener en cuenta para determinar el tipo de acceso a la información son:

- $t_{acción}$
- Tipo de informes requeridos (analítico u operacional)
- Volumen de información

1) $t_{acción}$

El $t_{acción}$ determina la necesidad de acceso a información en tiempo real. En particular, un $t_{acción}$ menor que el tiempo de carga de datos en el *data warehouse* (t_{ETL}) indica necesidad de acceso a la información en tiempo real.

2) Tipo de informes requeridos (analítico u operacional)

Este factor, a menudo, no se tiene en cuenta en el análisis de necesidad de BI en tiempo real, pero su impacto puede llegar a ser increíblemente grande, ya que puede suponer la implementación de una solución de BI en tiempo real cuando, quizá, no es del todo necesaria.

3) Informes analíticos

La necesidad de análisis de información justifica por sí sola la existencia de una plataforma de BI con acceso a los datos del *data warehouse*.

El *data warehouse* permite el análisis de grandes volúmenes de datos con un rendimiento ampliamente superior al análisis de datos realizado directamente en las fuentes de datos transaccionales.

4) Informes operacionales

En el caso de no necesitar informes analíticos sino operacionales (volúmenes de datos reducidos, formato estándar, sin navegación, etc.), la existencia de una plataforma de BI no estaría plenamente justificada. Por tanto, el acceso a los datos en tiempo real podría hacerse directamente sobre la fuente de datos como podría ser la base de datos de una aplicación transaccional.

Cabe destacar que hay múltiples factores que determinan la posibilidad de acceder directamente a la fuente de datos (o a una réplica) para informes operacionales. Los factores más importantes son:

- Impacto en el rendimiento del sistema y en la experiencia de los usuarios.
- Imposibilidad de modificar la fuente de datos (por ejemplo, para mejorar el rendimiento de algunas consultas).
- Complicación de la administración del sistema por la inclusión de nuevos usuarios.
- Problemas de seguridad en el acceso a la información.
- Razones políticas dentro de la organización.

En cualquier caso, el acceso directo a la fuente de datos para la extracción de información debe ser aprobada por el responsable de la fuente de datos.

5) Volumen de información

La cantidad de información necesaria para el análisis de datos en tiempo real condiciona la decisión final, ya que puede suponer un problema tecnológico.

Ejemplo

En el ejemplo anterior basado en un centro de soporte, si el análisis de la información se basa solamente en el número de peticiones de servicio, el volumen de datos será bajo. Si, en cambio, el análisis depende también de otros factores incluidos en cada petición de servicio (tipo de petición, subtipo de petición, descripción, estado, etc.), el volumen de datos a tener en cuenta para el análisis de información aumentará.

Supongamos que el diseño tecnológico a usar para una solución de BI en tiempo real consiste en microcargas (µ-cargas) ETL desde la base de datos transaccional hacia el *data warehouse*, ejecutándose cada 10'. No es lo mismo la transferencia de 1 MB (por ejemplo) de información cada 10' (en el caso de necesitar solamente el número de peticiones de servicio), que la transferencia de 10 GB (por ejemplo) en el caso de necesitar información extendida.

En el segundo caso, tal volumen de transferencia puede suponer un problema tecnológico. En ese caso, es posible que tengamos que reducir el volumen de datos y, por tanto, las capacidades de análisis.

Por tanto, en el caso de BI en tiempo real, puede ser que no tengamos todos los detalles de la información deseada en nuestro informe (algo que sí que tendremos en el *data warehouse* una vez se ejecute la carga diaria, por ejemplo). Esta es típicamente una limitación que los usuarios deben asumir.

Recordad

Una carga ETL es un proceso que permite cargar datos operacionales (orientados a facilitar las operaciones del día a día de una organización) a un *data warehouse* o base de de datos decisional (orientado al análisis analítico de los datos de una organización). Las siglas ETL vienen de los términos *extracción*, *transformación* y *carga* (*load* en inglés) y se refieren a las actividades a tener en cuenta en dicho proceso: extracción de datos de la base de datos operacional, transformación y carga en el *data warehouse.*

1.2.2. Escenarios

Supongamos que disponemos de un proceso diario de carga de datos (ETL) de un *data warehouse*. En este caso, el tiempo entre dos procesos ETL consecutivos es t_{ETL} = 24 horas.

Tal y como hemos visto anteriormente, los factores de necesidad de BI en tiempo real nos indican lo siguiente:

- Si $t_{acción} < t_{ETL}$, existe una necesidad de datos en tiempo real. En caso contrario, podemos conseguir los datos del *data warehouse*.

- Si el tipo de informes es analítico, necesitaremos una solución BI. En caso contrario, podemos conseguir los datos tanto de la base de datos operacional como del *data warehouse*.

- Si el volumen de información es bajo, podemos ir a buscar los datos tanto a la base de datos operacional como del *data warehouse*. En caso contrario, tendremos que ir al *data warehouse*.

La siguiente tabla muestra los diferentes escenarios que se derivan de los dos factores de necesidad de información en tiempo real cuando $t_{acción} < t_{ETL}$, que nos indica la necesidad de información en tiempo real:

	Volumen de información	
Tipo de informes	**Alto**	**Bajo**
Analítico	Reducir la información para poder aplicar BI en tiempo real	BI en tiempo real necesario

	Volumen de información	
Tipo de informes	**Alto**	**Bajo**
Operacional	Acceso a base de datos operacional sin BI	Acceso a base de datos operacional sin BI

1.2.3. Contexto histórico

Para poder entender la necesidad real de acceso a la información para su análisis en una organización y por qué esta situación se plantea ahora, es importante entender el contexto histórico del *data warehouse*. Para ello debemos hacer una retrospectiva histórica. En ella podemos distinguir tres estadios diferentes por lo que se refiere al tiempo de espera a la hora de poder cargar los datos en el *data warehouse* (lo que se conoce como la latencia de los datos). Estos tres estadios o fases son:

1) Primeros años: Durante los primeros años del *data warehousing,* la tecnología suponía una gran limitación en la latencia de los datos. Las cargas eran lentas y, por tanto, los usuarios debían aceptar una latencia de hasta varios días.

2) Madurez: El uso de los sistemas BI conllevó un aumento en las exigencias de los usuarios para el BI. Durante este periodo se produjo un aumento de los volúmenes de datos y el requerimiento de la reducción en la latencia de estos. La tecnología evolucionó, y los sistemas BI se perfeccionaron para lograr metas inalcanzables unos años antes. Durante este periodo, se estableció la creencia de que el *data warehouse* contenía datos con una latencia de un día.

3) El reto del acceso tiempo real: El éxito del *data warehouse* junto con el aumento de la competitividad en el mundo empresarial y la consiguiente necesidad de información cada vez más actualizada, ha impulsado a las organizaciones a demandar el acceso a la información con fines analíticos.

A esto hay que añadir que la evolución de la tecnología, tanto en la capacidad de almacenamiento como en la velocidad de proceso de datos, ha permitido superar las limitaciones que impedían históricamente acceder a los datos en tiempo real.

Todo ello desemboca en un nuevo reto dentro del mundo del BI: el acceso a la información para su análisis en tiempo real.

2. Obtención de los distintos tipos de información

En una solución de BI en tiempo real, los datos históricos y los actuales se combinan dando lugar a un único conjunto de datos que puede ser analizado por el usuario. A esta combinación de datos provenientes de diferentes fuentes de datos (*data warehouse* y sistema operacional) se le llama federación de datos o federación de fuentes de datos.

La federación de datos, sin embargo, conlleva una limitación en el tipo de información disponible.

2.1. Tipos de información devueltos en las consultas

Si nos remitimos a la naturaleza de la información obtenida en las consultas de usuario en una solución BI, podemos determinar tres tipos diferentes:

1) Datos con la **mínima granularidad** (máximo nivel de detalle). Por ejemplo, el detalle de las transacciones de venta. Los datos disponibles en los sistemas operacionales son datos con el máximo nivel de detalle. Estos datos son cargados en el *data warehouse* para su análisis. El usuario elige el nivel de detalle en cada análisis, y los datos le permiten la navegación hasta el máximo nivel de detalle.

2) Datos agregados. Por ejemplo, el total de ventas por línea de producto. Cuando la consulta del usuario se realiza a un nivel superior que el de máxima granularidad de los datos, el resultado debe ser agregado a ese nivel.

La agregación de datos la realiza el sistema gestor de bases de datos (en adelante SGBD) con el que estemos accediendo a los datos o, en el caso de otras fuentes de datos, esta agregación puede realizarla la misma herramienta de BI.

En el caso del *data warehouse*, es posible y recomendable disponer de agregaciones que mejoren el rendimiento de las consultas. Estas agregaciones son recalculadas cada vez que se ejecuta la carga de datos.

En cambio, en los sistemas operacionales no dispondremos de estas agregaciones, ya que cada inserción, modificación o borrado de datos implicaría el recálculo de dichos datos agregados, algo que requiere tiempo y que no es posible hacer por cada modificación de los datos fuente.

3) KPI. Por ejemplo, la ratio de oportunidades de venta cerradas con éxito. El caso de los KPI es muy similar al de los datos agregados, ya que, generalmente, se basan en fórmulas de agregación y/o funciones aritméticas que combinan dichos valores agregados.

2.2. Tratamiento de los tipos de información

Dentro del marco de un sistema BI en tiempo real, debido a la combinación de datos históricos y actuales, los tres tipos de información introducidos en el apartado anterior se tratan de una manera característica.

2.2.1. Datos con la mínima granularidad

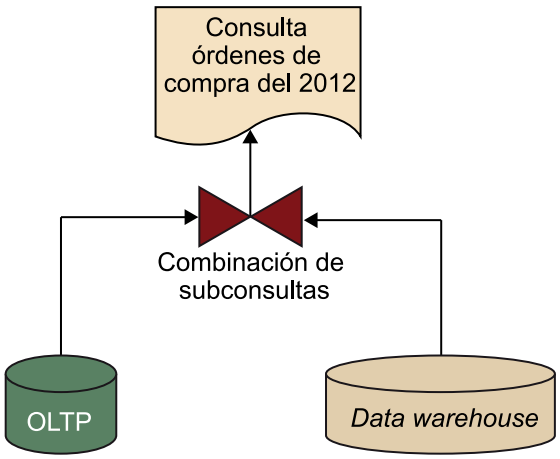
En el caso de una solución BI en tiempo real, obtendremos datos con la mínima granularidad tanto del *data warehouse* como de los sistemas operacionales.

Gracias a la federación de datos podremos combinar la información proveniente de ambas fuentes de datos.

Ámbito	Órdenes de compra.
Consulta	Identificador de compra, importe, cliente, fecha y producto de todas las órdenes de compra en el año 2012.
Descripción	Obtiene todos los registros de órdenes de compra de 2012 mostrando el identificador de compra, el importe, el nombre del cliente, la fecha y el código de producto de dicha compra.

La ejecución a alto nivel de esta consulta se muestra a continuación:

Consulta de datos con la mínima granuralidad



2.2.2. Datos agregados

Al no disponer la fuente de datos transaccional (*on line transactional processing, OLTP*) de datos agregados, la información se debe obtener al máximo nivel de detalle y agregar durante la consulta para poder mostrar los datos al usuario.

Esto se traduce en una degradación del rendimiento debido a dichas agregaciones. Esta degradación del rendimiento será mayor o menor en función de los volúmenes de datos y, sobre todo, de las funciones de agregación en cada consulta, ya que de estas dependerá en qué momento se hará la agregación de datos.

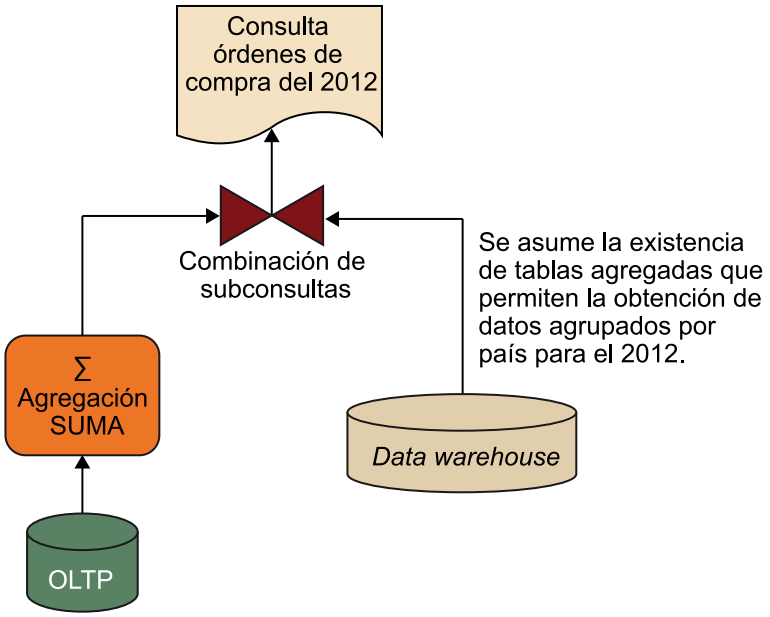
A continuación se muestran dos ejemplos de función de agregación: suma y mediana. En la suma, es posible la combinación de las subconsultas una vez hemos agregado los datos del sistema transaccional y el *data warehouse*. En cambio, en la mediana, la combinación y cálculo del resultado se debe hacer con el conjunto de valores individuales, lo cual evita una agregación previa y, por tanto, implica un coste más elevado del cálculo de la mediana.

Agregación → Combinación de subconsultas

Ámbito	Órdenes de compra.
Consulta	Importe total de las órdenes de compra en el año 2012 por país.
Descripción	Obtiene la suma de los importes de todos los registros de órdenes de compra del 2012 agrupados por país.

La ejecución a alto nivel de esta consulta se muestra a continuación:

Consulta de datos agregados: Agregación→ Combinación de subconsultas



Teniendo en cuenta que el volumen de datos esperado en la fuente de datos OLTP no es muy grande, se puede asumir que la afectación al rendimiento de la consulta es baja.

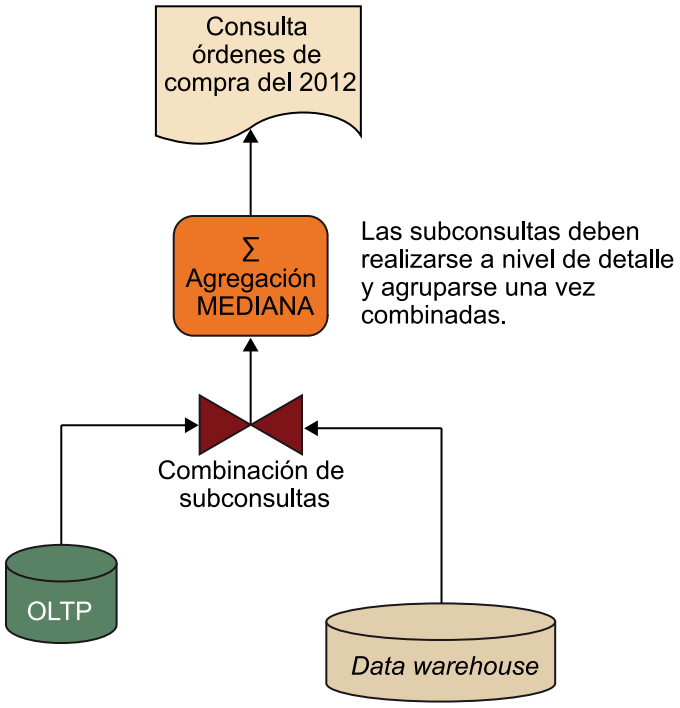
Combinación de subconsultas → Agregación

Ámbito	Órdenes de compra.
Consulta	Mediana del importe de las órdenes de compra en el año 2012 por país.
Descripción	Obtiene la mediana de los importes de todos los registros de órdenes de compra del 2012 agrupados por país.

Mediana

La mediana de un conjunto de valores no debe confundirse con la media de estos (por ejemplo, dado el conjunto de valores {1,2,3,4,5,10,20,30,40,50}, la mediana es 7.5, mientras que la media es 16.5).

Consulta de datos agregados: Combinación de subconsultas→ Agregación



En este caso, teniendo en cuenta los posibles volúmenes del *data warehouse*, el coste de agregación de todos los datos sería muy elevado. Y si se diera la situación en que los datos no se encuentran en el *data warehouse* con el nivel máximo de detalle, la extracción de datos desde el sistema OLTP debería extraer todas las transacciones del 2012. Esto supondría un gran impacto en el sistema origen debido a la extracción de datos e igualmente implicaría un alto coste en la agregación de todos estos datos.

2.2.3. KPI

El caso de los KPI es muy similar al de los datos agregados, ya que, generalmente, se basan en fórmulas de agregación y/o funciones aritméticas que combinan dichos valores agregados.

Por tanto, habrá situaciones donde encontremos graves problemas de rendimiento en las consultas con KPI cuando accedamos a información en tiempo real.

3. Factores a tener en cuenta durante el diseño

Cuando diseñamos un sistema de BI en tiempo real, hay que poner especial atención a estos dos factores:

- **Obtención de datos**: Cómo y cuándo se extraen los datos de las diferentes fuentes de datos.

- **Integración de** diferentes **fuentes de datos** en el *data warehouse*: cómo se combinan los datos obtenidos de las distintas fuentes de datos para crear una única versión de ellos.

A continuación vamos a analizar estos dos temas desde el punto de vista comparativo entre el BI tradicional y el BI en tiempo real.

3.1. Obtención de datos

La extracción de datos en tiempo real no es tan "simple" como la extracción de datos en un sistema BI tradicional. Hay que tener en cuenta que la frecuencia deseada de extracción y el método utilizado pueden entrar en conflicto. Además también hay que tener en cuenta el impacto sobre el sistema origen de los datos. Debido a la ejecución continua de los procesos de extracción de datos en un sistema BI en tiempo real, las extracciones pueden sobrecargar los sistemas orígenes provocando caídas en el rendimiento de estos, de lo que se resentirían los usuarios finales, tanto del sistema decisional como del sistema operacional.

Por todo esto, hay que considerar la obtención de los datos como una parte clave en el proceso de carga de los datos en un sistema BI en tiempo real, que puede acarrear muchos inconvenientes e incluso forzar limitaciones en nuestro sistema.

Las diferencias en la extracción de datos entre un sistema BI tradicional y un sistema BI en tiempo real son:

- Frecuencia y método de extracción

- Impacto en el rendimiento de la fuente de datos

3.1.1. Frecuencia y método de extracción

Sistema BI tradicional

En un sistema BI tradicional con una ejecución de la carga de datos típicamente ejecutándose una vez al día, el momento de la extracción suele coincidir con una ventana de tiempo en la que no existe actividad en el sistema (a excepción de sistemas 24x7 o similares).

Este hecho permite una extracción de los datos basada en procedimientos estándar. Estos procedimientos son, típicamente, consultas SQL para bases de datos, importación de ficheros planos de texto, API específicas de aplicación y acceso ODBC o JDBC mediante controladores específicos.

Estos métodos de extracción, permiten obtener e incorporar grandes cantidades de datos al flujo de la carga para su tratamiento y posterior inserción en el *data warehouse*.

La manera como se ejecutan estos procedimientos es de manera activa. Es decir, la carga de datos realiza una llamada a estos procedimientos y estos devuelven un conjunto de datos.

Sistema BI en tiempo real

En un sistema BI en tiempo real, utilizar estos métodos de extracción pueden introducir riesgos, debido al requerimiento intrínseco de un sistema BI en tiempo real de obtener datos continuamente.

Pongamos que nuestra solución para las cargas en tiempo real utiliza µ-cargas. Es decir, cargas continuas desde las fuentes de datos con relativamente poca cantidad de información para una ejecución eficiente de las cargas.

Cuando trabajamos con BI en tiempo real, el flujo de información desde el sistema transaccional hacia el sistema BI debe ser constante, con lo cual, la frecuencia de llamadas a las fuentes de datos para obtención de información debe ser muy alta.

Los métodos descritos anteriormente pueden ser procesos que requieren cierto tiempo para obtener los datos extraídos, con lo cual la frecuencia disminuirá. Esto nos lleva al primer problema: si el tiempo de respuesta de estas llamadas es demasiado alto, podríamos no tener tiempo de tratar esos datos entre dos ejecuciones consecutivas de la carga de datos en tiempo real.

Tomemos el siguiente escenario como ejemplo: Un proceso tiene un $t_{acción}$ de 10'. La extracción de datos se realiza mediante una llamada a un servicio web, que nos permite obtener un fichero en formato XML. El sistema operacional genera grandes volúmenes de información entre las 10:00 y las 11:00 de la mañana por la ejecución de ciertos procesos *batch* Debido a este incremento de datos en esta franja horaria, las extracciones de datos mediante el servicio web se extienden más allá de los 10' debido al tiempo de transferencia de dicha cantidad de datos.

Una solución a este problema puede ser reducir la frecuencia de las cargas, para permitir la finalización de una carga de datos en el caso de que requiera más tiempo de ejecución. Sin embargo, esta solución va en detrimento del requerimiento de "tiempo real". Cuanto

Batch

El termino *batch* se utiliza para designar los procesos que se ejecutan automática y periódicamente sin necesidad de asistencia humana.

menor sea la frecuencia, más latencia de los datos, y menos "tiempo real" en nuestra solución de BI.

Otra solución es cambiar el método de extracción de datos, cuando esto sea posible. Existe software especializado en la obtención de información de fuentes de datos que, mediante *streaming*, mandan esos datos al *data warehouse* casi en tiempo real. Este software suele ser multiplataforma, y suele permitir conectar de manera casi instantánea sistemas de orígenes de datos con sistemas de consumidores de datos (en nuestro caso el *data warehouse*). De esta manera podemos reducir el tiempo de extracción de los datos para poder centrarnos en su tratamiento. Esto nos permite reducir la ventana de ejecución de las cargas µ-batch.

El método utilizado en estos sistemas es el llamado *sniffing* de ficheros de logs (ya sea de bases de datos como de ficheros convencionales). Este software se conecta a estos ficheros y lee en tiempo real cualquier actividad llevada a cabo por el sistema origen y escrita en el fichero de *log* sin tener la necesidad de realizar ninguna consulta a la base de datos. Acto seguido, el tipo de operación (inserción, modificación o eliminación) y los datos asociados son trasmitidos al sistema consumidor de información (*data warehouse*), que los recibe en tiempo casi real, sin tener que esperar a la siguiente ejecución de una carga de datos.

De esta manera, podemos extraer información sobre las altas, bajas y modificaciones realizadas sin la necesidad de ejecutar consultas a la base de datos ni de crear disparadores en la base de datos para capturar estos eventos. Además, no habrá la necesidad de ejecutar consultas a la base de datos operacional para extraer los datos más recientes de manera periódica. Esto supone un ahorro de recursos y una reducción del impacto en el sistema operacional.

Streaming

El término *streaming* se utiliza para designar una técnica de transferencia de datos entre dos sistema informáticos mediante el cual los datos del primer sistema (el que genera los datos) son transferidos automáticamente al segundo sistema (también conocido como consumidor) cuando los datos son generados. La diferencia con una carga de datos ETL es que en esta, los datos son extraídos periódicamente de manera que un conjunto de datos es extraído y enviado cuando el proceso se ejecuta, mientras que en sistema basado en *streaming*, los datos se envían cuando estos son almacenados en el sistema origen.

Ejemplo

Ejemplos de software para integración de datos en tiempo real son Oracle Golden Gate, IBM InfoSphere Change Data Capture y SharePlex.

3.1.2. Impacto en el rendimiento de la fuente de datos

Sistema BI tradicional

En un sistema BI tradicional, el proceso de extracción y publicación de datos en el *data warehouse* es puntual y no continuo, siendo de una sola vez al día en la mayoría de los casos. Además, al ejecutarse en una ventana de tiempo donde no hay actividad en la fuente de datos, permite que haya margen para procedimientos de extracción que consuman elevados recursos del sistema origen, sin afectar a este.

En sistemas 24x7 o en sistemas donde la carga de datos se realiza más de una vez al día, en momentos en que los sistemas origen están activos, este margen desaparece. En estos casos, es muy importante que los procedimientos de extracción de datos sean eficientes.

Dado el caso en que no haya manera alguna de optimizar los procedimientos de extracción de datos, los usuarios se podrían ver afectados temporalmente (mientras dure la extracción de datos), por una bajada del rendimiento global de las aplicaciones para las cuales se estuviese ejecutando la extracción de datos.

Sistema BI en tiempo real

En el caso de un sistema BI en tiempo real con extracción de datos mediante cargas µ-batch, debido a la ejecución casi continua de la cargas µ-batch, el rendimiento de dichas extracciones debe ser óptimo. Si no se consigue un rendimiento óptimo en las extracciones, habrá un impacto sobre los usuarios fina-

les. Sin embargo, en este caso estaremos hablando de una afectación continua al sistema origen, con lo que el rendimiento de este estará constantemente por debajo de sus posibilidades.

En el caso de tratar con un sistema origen con soporte de base de datos, existe la posibilidad de ajustar el sistema mediante técnicas de administración de bases de datos, como por ejemplo la creación de índices y particiones para mejorar el rendimiento de las consultas. Sin embargo, estas técnicas tendrán también cierto impacto negativo en la aplicación, ya que añaden cierto trabajo extra a la base de datos en el caso de inserción, modificación y borrado de datos.

Sea cual sea el soporte físico de los datos y el procedimiento de extracción, el objetivo debe ser el de lograr un rendimiento óptimo para la extracción de los datos. Hay que tener en cuenta que, dependiendo de los procedimientos de extracción de datos, esto puede que no sea posible y que nos encontremos con una limitación bloqueante para nuestro sistema de BI en tiempo real.

3.2. Integración de diferentes fuentes de datos en el *data warehouse*

La integración de información de múltiples fuentes de datos en un sistema de BI en tiempo real requiere tener en cuenta los siguientes aspectos:

- **Frecuencia de extracción**: No poder extraer de datos de un sistema origen con la misma frecuencia que las cargas puede llevar a problemas de datos, donde no se refleje la realidad de estos.

- **Sincronización de datos entre fuentes de datos**: Por su parte, no poder sincronizar datos entre diferentes sistemas origen provocará la desconexión de datos y la imposibilidad de analizar los datos correctamente.

En los dos casos el problema es el mismo, el sistema BI puede ofrecer información incorrecta, ya que su información es incompleta o desactualizada. A continuación se exponen estos aspectos con más detalle.

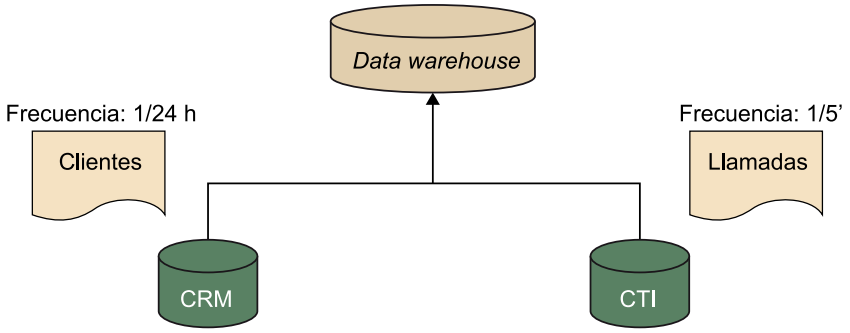
3.2.1. Frecuencia de extracción

Cuando hablamos de integración de fuentes de datos en una solución de BI en tiempo real, hay que tener en cuenta que todas las fuentes de datos deben poder ser accedidas por la carga de datos en tiempo real. Si hubiese alguna fuente de datos que no permitiese acceso para la extracción de datos en tiempo real (por ejemplo, tan solo existen extracciones diarias de información de un sistema de información concreto), existe el riesgo de que los datos extraídos en global no sean un reflejo de la realidad en el momento de la carga de datos.

La frecuencia de extracción debe pues ser consistente en todas las fuentes de datos necesarias para la construcción de un área de análisis de datos.

El siguiente diagrama muestra un ejemplo de esta situación:

BI en tiempo real con extracción de datos de múltiples sistemas origen



CRM (*costumer relationship management*)
Un sistema de administración de la relación con los clientes suele ser la fuente de datos corporativa de clientes (máster de clientes).

CTI (*computer telephony integration*)
Un sistema de integración de telefonía informática es un sistema informático destinado a la interacción entre una llamada telefónica y un ordenador de manera coordinada.

En este caso, podríamos cargar llamadas sin disponer de la información de los clientes, ya que esta se almacena en un sistema de gestión de relación con el cliente (CRM). Dada esta situación, no podríamos hacer un análisis completo de los datos.

Ved también

Para más información sobre la frecuencia de extracción en sistemas de BI tradicionales y soluciones de BI en tiempo real, se puede consultar el apartado anterior.

Sistema CRM

Un sistema CRM (*customer relationship management*) permite la gestión de datos de los clientes de una organización y de las interacciones entre ambos, como por ejemplo las oportunidades de venta, las peticiones de servicio del cliente mediante correo electrónico o llamada telefónica, y las campañas de marketing y las respuestas de cada cliente.

3.2.2. Sincronización de datos entre fuentes de datos

Como ya hemos visto en el diagrama anterior, es posible que la extracción de datos necesaria para dar repuesta a análisis de datos de un área de negocio afecte a múltiples fuentes de datos.

En el ejemplo anterior, cuando un cliente nuevo llama a un centro de llamadas, el sistema CTI obtendrá la información de la llamada, mientras que la aplicación que verá el operador telefónico será el CRM, donde deberá dar de alta al nuevo cliente y donde también registrará la petición de servicio del nuevo cliente, incluyendo el motivo de la llamada entre otras cosas.

Llegados a este punto donde tenemos información distribuida en diferentes fuentes de datos, necesitamos una interfaz entre ambos sistemas para poder asociar una llamada del CTI con el cliente, la petición de servicio y la actividad de tipo llamada entrante que queda registrada en el sistema CRM. Esta interfaz de intercambio de información entre distintas fuentes de datos es muy impor-

tante para poder tener la capacidad de cruzar datos entre ambos sistemas. El tratamiento de esta sincronización en un entorno BI en tiempo real es más estricto que en un sistema BI tradicional debido a la frecuencia de las cargas de datos. A continuación se analizan ambos casos.

Sistema BI tradicional

En un sistema BI tradicional, suponiendo que la carga de datos se ejecuta una sola vez al día, la sincronización puede efectuarse a partir de un proceso *batch* que también se ejecute con la misma periodicidad. En este caso, ese proceso de sincronización deberá ejecutarse antes de las cargas de datos al *data warehouse* para que la información cargada esté sincronizada, compartiendo los identificadores de las entidades comunes.

Estos identificadores comunes serán los que permitirán, durante la carga de datos, asociar los de una fuente de datos con otra.

Por ejemplo, podremos asociar las llamadas entrantes (disponibles en el sistema CTI), con los datos del cliente (disponibles en el CRM).

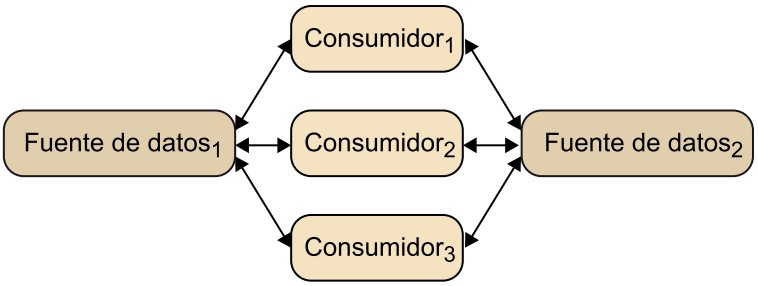
Sistema BI en tiempo real

En el caso de un sistema BI en tiempo real con extracción de datos mediante cargas µ-batch, debido a la ejecución casi continua de estas, la sincronización entre diferentes fuentes de datos debe ejecutarse casi instantáneamente. De no ser así, cabe la posibilidad de no tener una visión completa de los datos cargados en la última ejecución de la carga de datos.

Siguiendo con el ejemplo anterior, si la sincronización fuese como en el caso del sistema BI tradicional, durante las cargas ejecutadas a lo largo del día, podríamos cargar información del CTI, pero no podríamos asociarla con la información del CRM. Por esta razón, en el caso de una solución BI en tiempo real, es necesaria una sincronización de las fuentes de datos también en tiempo real. Esta sincronización se puede conseguir mediante diferentes técnicas. A continuación se muestran las dos más habituales:

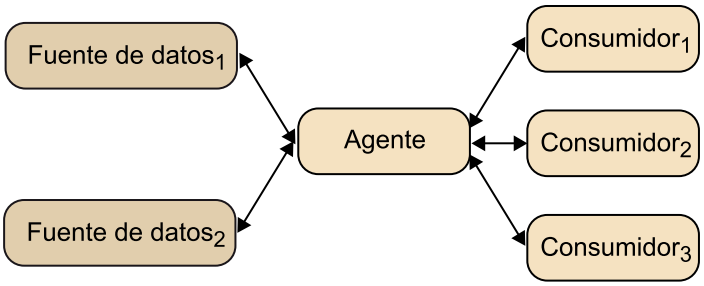
- ***Streaming***: Los datos, una vez son almacenados en el sistema origen, son enviados a otros sistemas destino para realizar la sincronización de datos de los diferentes sistemas. El sistema origen es el que activamente distribuye la información. Un cambio en la lista de subscriptores (o sistemas destino) implica un cambio en la aplicación origen para poder distribuir esta información a los sistemas consumidores.

Distribución de datos mediante *streaming*



- **Suscripción y distribución mediante un agente**: Los datos son enviados por el sistema origen a un agente que se encarga de su distribución a una lista de suscriptores. Los sistemas consumidores se suscriben a los diferentes servicios de datos del agente que actúa como punto de gestión y distribución de los cambios en los datos. De esta manera, la carga en los sistemas origen se reduce, así como los cambios en este cada vez que hay modificaciones en la lista de suscriptores. Por parte de los suscriptores, también hay beneficios al existir solamente un sistema desde el cual se reciben datos (agente), en lugar de tener que lidiar con diversos sistemas origen con posiblemente diferentes conectores e interfaces.

Distribución de datos mediante agente



Es muy importante tener en cuenta en ambos casos lo siguiente:

- Los sistemas origen y destino deben ser modificados para distribuir los datos a otros sistemas u obtenerlos en el caso de que sean consumidores de esta información. Esto implica cambios en el código de las aplicaciones que pueden suponer la violación de licencias de software (lo cual supone perder el soporte del fabricante). Además, estas modificaciones requieren la existencia y disponibilidad de personal especializado para modificar los mencionados sistemas informáticos. Por tanto, se trata claramente de un inconveniente a tener en cuenta a la hora de planificar un sistema BI en tiempo real.

- Ambos sistemas tendrán una ligera sobrecarga debido al tratamiento adicional que deben hacer en el caso de la distribución o recepción de datos. Si el flujo de información es alto, esta sobrecarga puede llegar a afectar al rendimiento del sistema.

4. Evolución de sistemas BI tradicionales a tiempo real

Un sistema BI tradicional con carga de datos diaria generalmente no puede evolucionar a tiempo real mediante el incremento de la frecuencia de las cargas. Es cierto que la frecuencia de las cargas puede normalmente incrementarse hasta cierto punto y que ello permite disponer de datos actualizados con más frecuencia que con la carga diaria tradicional. No obstante, existe un límite en lo que respecta a la frecuencia de carga, debido a cómo se diseña el proceso de carga de datos.

4.1. Carga de datos tradicional

Una carga desde las fuentes de datos hacia el *data warehouse* es un proceso que se divide en las siguientes fases:

1) Extracción de datos de las fuentes de información. Este proceso requiere de consultas a las fuentes de datos para obtener los datos actualizados desde la última fecha de carga. Esto incluye inserciones, modificaciones y borrado de información. Podemos hacer extracciones totales (obtención de todos los datos en el sistema operacional) o incrementales (solo los datos actualizados desde la última extracción).

2) Limpieza de datos. En esta fase se examinan los datos para detectar anomalías en el formato y valores, ya sea por sí mismos (por ejemplo, un código postal en España debe tener cinco números) o basados en reglas de negocio (por ejemplo, un cliente moroso solo puede tener ventas contra reembolso), y arreglarlos cuando sea posible (por ejemplo, si el formato de los números de teléfono debe incluir el código de país y tan solo aceptar dígitos, el número de teléfono '+34 (93) 555-55-55' será convertido a '34935555555' para cumplir con el formato adoptado).

3) Integración de las distintas fuentes de datos. Esto es lo que se conoce en inglés como *conform*. Durante esta fase se consolida la información referente a un mismo concepto que se halla dispersa en diferentes fuentes de datos.

> Por ejemplo, si los datos de los clientes se hallan en un sistema de gestión de relaciones con el cliente (CRM) pero son posteriormente enriquecidos por un indicador de morosidad proveniente de una aplicación de riesgos, ambos sistemas serán fuentes de datos y deberemos obtener datos de ambos sistemas durante la carga diaria. Durante esa carga, hay que integrar la información de ambos sistemas en un solo conjunto de datos, donde los datos de cada sistema se alinean con la información específica de cada uno, reunida en un solo contenedor o registro de datos.

4) Publicación de datos. Cálculo y presentación de los datos que van a ser consultados por los usuarios en las tablas del *data warehouse*.

5) Cálculo de agregaciones. Las tablas agregadas permiten un acceso más rápido a los datos, cosa que acelera el tiempo de respuesta de las consultas. Este es un proceso que suele requerir una cantidad importante de recursos.

Si bien es cierto que el rendimiento de las cuatro primeras fases dependerá de la cantidad de información a procesar en cada carga y que con una frecuencia de carga muy alta esta cantidad de información se reducirá considerablemente, esto no pasará con la fase 5, ya que el cálculo de las agregaciones podría, en función de la frecuencia, solaparse con la próxima ejecución.

Por tanto, la frecuencia de las cargas no puede generalmente elevarse hasta obtener un $t_{ETL} \rightarrow 0$.

Para poder obtener una solución de BI en tiempo real, debemos rediseñar las cargas y permitir la consulta de datos históricos y datos actuales mediante una solución transparente para los usuarios conocida como federación de datos.

Ved también

Ver con más detalle la federación de datos en el apartado 5.1.3. "Federación de datos".

Federación de datos

La federación de datos consiste en la integración de diferentes esquemas de bases de datos base en un esquema único. El usuario ejecuta una sola consulta sobre dicho esquema único, pero el sistema ejecuta consultas diferentes basadas en los esquemas de datos base, combinando los resultados de cada consulta en un único conjunto de resultados.

4.2. Caso práctico

Veamos un ejemplo de cómo el cálculo de agregaciones limita la frecuencia de las cargas.

Supongamos una carga de datos diaria con las siguientes duraciones parciales:

1) Extracción-limpieza-integración-publicación: 4 horas

2) Cálculo de agregaciones: 2 horas

Existe el requerimiento de incrementar la frecuencia de las cargas para poder proporcionar datos más actualizados a los usuarios para su análisis. El objetivo es poder ofrecer datos con un máximo de 1 hora de antigüedad.

En una primera aproximación al objetivo, se incrementa la frecuencia de las cargas de datos a una ejecución cada 6 horas.

$$f_{ETL} = \frac{1 \text{ ejecución}}{6 \text{ horas}}$$

Las duraciones parciales observadas se muestran a continuación:

1) Extracción-limpieza-integración-publicación: 1 hora

2) Cálculo de agregaciones: 2 horas

Con la idea de reducir el tiempo entre dos cargas de datos consecutivas, se incrementa la frecuencia de las cargas de datos a una ejecución cada 3 horas.

$$f_{ETL} = \frac{1 \text{ ejecución}}{3 \text{ horas}}$$

Las duraciones parciales observadas se muestran a continuación:

1) Extracción-limpieza-integración-publicación: 30'

2) Cálculo de agregaciones: 2 horas

Como se puede observar, las fases que dependen del número de registros actualizados se benefician del incremento de frecuencia de la ETL. Esto se debe al menor número de registros a tratar, lo cual agiliza la ejecución de estas fases. En cambio, el cálculo de agregaciones depende del volumen total en cada una de las estructuras de datos del *data warehouse*. Esto explica que la duración del cálculo de agregaciones no se reduzca al incrementar la frecuencia de extracciones.

En este caso, la solución pasa por la federación de datos.

Solamente esta decisión ya puede implicar cambios a nivel de herramienta de BI en el caso de que la herramienta actual no soporte federación de datos. Suponiendo que la herramienta de BI utilizada sea válida para federación, deberemos rediseñar los datos de la capa de negocio para combinar datos de múltiples fuentes de datos, y mapear cada uno de sus modelos de datos. Si además hemos optado por µ-cargas, deberemos tener en cuenta todo el trabajo de diseño e implementación de estas. A esto hay que añadir las opciones de réplica de base datos y *streaming* de datos, que complicarán la solución.

5. Tecnología

5.1. Arquitectura

Si tenemos en cuenta el retraso entre el momento en que un evento ocurre y el momento en que está disponible para el análisis de datos, hay tres grandes tipos de soluciones técnicas para BI:

1) *Reporting* sobre sistema transaccional (OLTP)

2) *Data warehouse*

3) Federación de datos

Cada una de las soluciones técnicas anteriores tiene implicaciones propias a nivel técnico, de latencia de datos y de costes de implementación. Estas implicaciones deben ser tenidas en cuenta a la hora de decidir la opción tecnológica de una solución BI en tiempo real.

A continuación se detallan cada una de ellas.

5.1.1. *Reporting* sobre OLTP

Esta opción consiste en ejecutar las consultas directamente sobre la base de datos transaccional, sin tener en cuenta cargas de datos sobre ningún *data warehouse*. La latencia de los datos es baja, ya que los datos se obtienen directamente de la fuente original de almacenamiento. Sin embargo, el rendimiento es también bajo, ya que la fuente de datos no está diseñada para el análisis de un gran número de datos sino para transacciones (diseño OLTP).

5.1.2. *Data warehouse* (DWH)

En el caso de usar tan solo el *data warehouse*, no dispondremos de una solución de BI en tiempo real.

La latencia de los datos es la misma que la periodicidad de la carga del *data warehouse*. Es decir, la latencia es alta.

El rendimiento es el mejor de todas las opciones presentadas, ya que no hay ningún tipo de ejecución posterior a la consulta para tratar los datos y combinarlos con otros datos provenientes de ninguna otra consulta.

5.1.3. Federación de datos

La federación de datos consiste en la integración de diferentes modelos de datos base en un modelo único. El usuario ejecuta una sola consulta sobre dicho modelo único, pero el sistema ejecuta consultas diferentes basadas en los modelos de datos base, combinando los resultados de cada consulta en un único conjunto de resultados.

Una de las características de un sistema de BI en tiempo real es la capacidad de ofrecer una visión única de los datos. Gracias a la federación es posible ofrecer un modelo único de acceso a los datos, independientemente de dónde estén almacenados, bien en el *data warehouse*, bien en el sistema operacional.

Existen varias herramientas de BI en el mercado. Sin embargo, no todas ellas son capaces de proporcionar federación de datos de múltiples fuentes de datos. Una de estas fuentes de datos sería el *data warehouse* y las otras vendrían del sistema operacional o de otras fuentes de información, como por ejemplo de Internet.

Sin esta capacidad, las consultas al *data warehouse* y el sistema operacional deben realizarse por separado para, posteriormente, combinarse manualmente y proporcionar el resultado deseado. Este procedimiento suele ser complejo y necesita de conocimientos técnicos, con lo cual no disponer de la capacidad de federación de datos puede llegar a ser causa de fracaso del proyecto de BI en tiempo real.

Las herramientas de BI con soporte para federación de datos de múltiples fuentes de datos permiten definir un modelo de datos lógico basado en el modelo de negocio. Ese modelo de negocio se mapea en los modelos de datos específicos de las distintas fuentes de datos. Estos datos pueden venir de distintas fuentes de datos, como por ejemplo bases de datos, ficheros de texto, ficheros MS Excel y cubos con tecnología OLAP.

Dentro de la federación de datos, existen dos tipos, dependiendo de dónde se hallan los datos que contienen la información más actual:

1) Federación pura, cuando los datos que permiten tiempo real se hallan en el mismo sistema OLTP.

2) Federación con µ-cargas, cuando los datos que permiten tiempo real se hallan en una área del *data warehouse* específica para estos datos. Esta área de almacenamiento de datos se denomina partición de tiempo real.

Federación pura

Esta opción consiste en ejecutar *N* consultas independientes. Una consulta será sobre el *data warehouse* y *n-1* consultas sobre las distintas *n-1* fuentes de datos. Notad que *N* es el número de fuentes de datos a sincronizar. Finalmente se combinarán todos los resultados obtenidos y se mostrarán al usuario como si del resultado de una sola consulta se tratara.

La latencia de los datos es la misma que la solución de *reporting* sobre OLTP, ya que los datos se obtienen directamente de la fuente original de almacenamiento.

En este caso, el rendimiento suele ser un poco mejor que en el acceso únicamente a la fuente de datos OLTP. El motivo es que el acceso a los datos del sistema transaccional se reduce solamente a la parte no cargada en el *data warehouse,* que suele ser una pequeña porción de los datos. Por tanto, el acceso a grandes volúmenes de datos se efectúa en el *data warehouse* y se complementa con los pocos datos extraídos de la base de datos OLTP. Debe tenerse en cuenta también el incremento de tiempo debido a la combinación de las consultas independientes y a la "conversión" del modelo OLTP al modelo dimensional existente en el *data warehouse*, que afecta negativamente al rendimiento.

Federación con µ-cargas

En la federación con µ-cargas de datos, se cargan los nuevos datos disponibles en el entorno OLTP en un espacio intermedio entre este y el *data warehouse*: la partición de tiempo real. Este espacio intermedio permite una mayor integración con los datos del *data warehouse*, con lo cual la federación se realiza entre este espacio intermedio y el *data warehouse*, resultando en una respuesta más rápida y eficiente.

La latencia de los datos es un poco más elevada, ya que depende de la frecuencia de la µ-carga. Esta frecuencia debe ser menor que la de la carga del *data warehouse*, y suele ir desde una ejecución cada *s* segundos hasta una ejecución cada 12 horas (asumiendo cargas diarias del *data warehouse*).

Desde el punto de vista del rendimiento, esta opción es más efectiva que las anteriores, ya que, a pesar de no disponer de los datos de la µ-carga totalmente integrados en el *data warehouse,* estos están estructurados de manera que facilitan su análisis y combinación con el *data warehouse* (se utiliza un diseño dimensional). Además, en las consultas de usuario no interviene en ningún momento el sistema transaccional.

La ejecución de µ-cargas implica el acceso reiterado al sistema transaccional. El acceso a esta fuente de datos debe ser extremadamente eficiente para no afectar al rendimiento del sistema transaccional.

A continuación se muestran varias opciones para no sobrecargar el sistema origen:

1) Creación de índices en la base de datos OLTP. Los índices en las bases de datos actúan como aceleradores en el acceso a los datos. En el caso de las extracciones en las µ-cargas, la existencia de índices por un campo de fecha que acceda de manera eficiente a los datos a extraer suele incrementar el rendimiento de las extracciones.

2) Creación de una partición *online* en la base de datos OLTP. Una partición de datos en una base de datos permite disponer de los datos en un contenedor aislado del resto de los datos.

> Por ejemplo, una partición con los datos de peticiones de servicio del último mes permite tener dichos datos separados de toda la historia de peticiones de servicio. Esta reducción de volúmenes implica una mejora sustancial en el acceso a los datos (equivaldría a poder encontrar un libro en una estantería concreta en vez de hacerlo en una biblioteca).

3) Replicación de bases de datos. La opción de tener una copia de la base de datos sincronizada con la base de datos maestra permite acceder a una fuente de datos que no es utilizada por los usuarios finales. Al no haber usuarios utilizando el sistema, las consultas para la extracción de datos en las µ-cargas se ejecutan de manera más eficaz. En este caso, los recursos de la base de datos utilizada son asignados enteramente al proceso de extracción en vez de compartirlos con los usuarios. Esta opción permite una afectación cero al sistema operacional (aparte de la sobrecarga de la replicación de la base de datos).

4) Transferencia *online* de registros mediante *streaming*. Esta técnica evita las consultas periódicas a la base de datos. Los datos actualizados son transferidos a una partición *online* en otro sistema. Una vez en esta partición, podemos lanzar la µ-carga sin impactar la base de datos operacional.

Consideraciones técnicas en la federación de datos

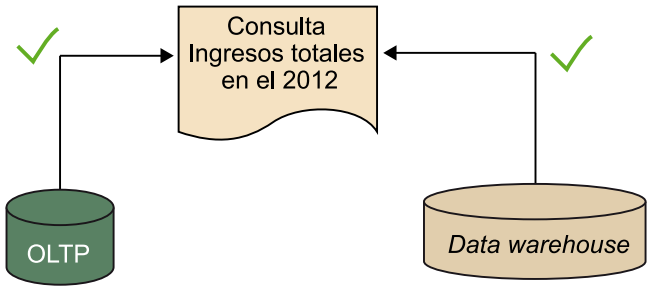
Si se desea implementar una solución de BI en tiempo real con federación de datos, hay que tener en cuenta que la herramienta de BI elegida pueda soportar esta funcionalidad. Es decir, que dinámicamente se puedan elegir qué particiones de la federación se han de consultar para poder obtener los resultados deseados.

A continuación se muestran tres consultas diferentes y cómo una herramienta con federación de datos actuaría en cada una de ellas:

Escenario

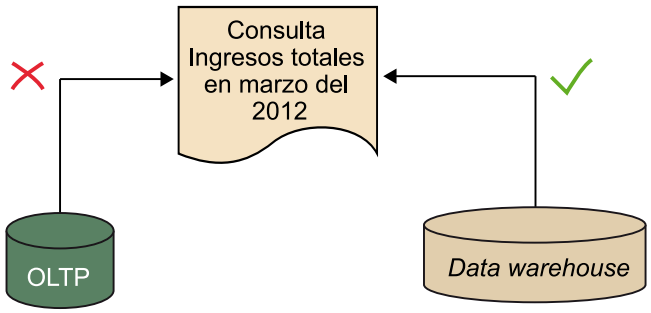
- OLTP (sistema operacional): Contiene todos los datos históricos en un modelo de datos transaccional no optimizado para el análisis de los datos.

- *Data warehouse*: Contiene datos hasta el 30 de septiembre del 2012 (fecha de transacción < 1 de octubre del 2012).



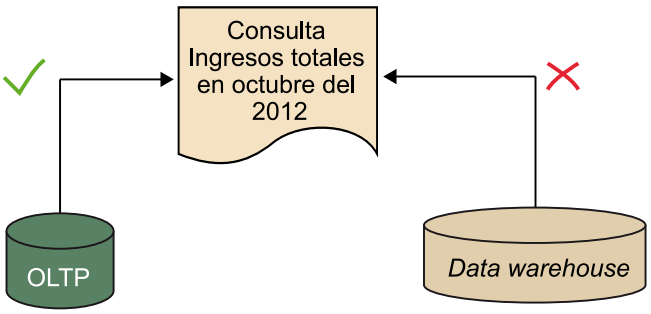
Fecha de transacción >= 1 octubre 2012 Fecha de transacción < 1 octubre 2012

La herramienta BI debe realizar tan solo una consulta sobre la partición histórica



Fecha de transacción >= 1 octubre 2012 Fecha de transacción < 1 octubre 2012

La herramienta BI debe realizar tan solo una consulta sobre la partición histórica



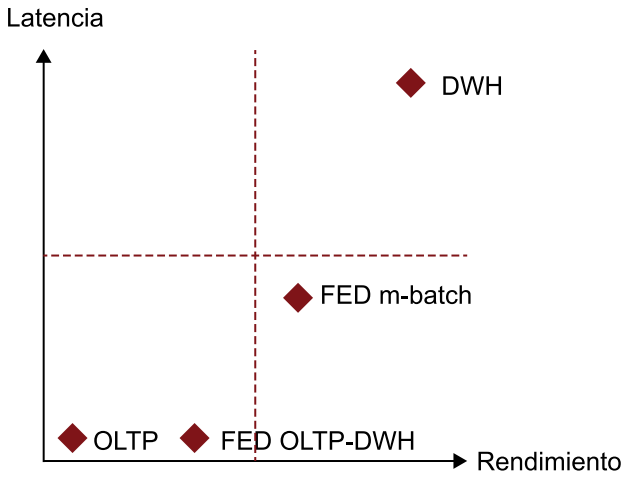
Fecha de transacción >= 1 octubre 2012 Fecha de transacción < 1 octubre 2012

La herramienta BI debe realizar tan solo una consulta sobre la partición en tiempo real

Una herramienta de BI sin federación realizaría dos consultas en cada uno de los casos anteriores y combinaría los datos. En el caso de los escenarios 2 y 3, las consultas innecesarias (✗) también se ejecutarían, no devolviendo ningún registro, con lo cual el resultado de la consulta sería correcto, pero estaríamos consumiendo recursos sin ninguna necesidad.

5.2. Resumen de opciones de arquitectura

El siguiente cuadrante resume las cuatro opciones de arquitectura anteriores.



5.3. Consideraciones técnicas generales

La definición tanto de la arquitectura como de la tecnología y periodicidad de las cargas de la solución BI va a depender tanto de los requerimientos funcionales del sistema como de los requerimientos técnicos proporcionados por el departamento de sistemas informáticos. En todo caso se necesita un balance entre ambos para poder obtener una solución que pueda satisfacer a ambas partes.

Conviene tener muy en cuenta las limitaciones a las que nos enfrentamos, ya que estas influirán en la decisión final. Estas limitaciones son:

1) Limitaciones de los sistemas fuente

- **Disponibilidad de la base de datos**. Las bases de datos suelen tener periodos de indisponibilidad reservados para tareas de mantenimiento. Este tipo de tareas incluye la realización de copias de seguridad, aplicación de parches del sistema y cambios de discos entre otras. Durante los periodos de indisponibilidad, las extracciones en las fuentes de datos no son posibles, lo que limita las cargas de datos hacia el *data warehouse*.

- **Carga de trabajo de la base de datos**. La interacción de los usuarios y de otros procesos existentes con el sistema operacional genera una carga de trabajo en la base de datos origen. Cuando esta carga es elevada, los recursos disponibles para las cargas de datos de la solución BI en tiempo real son reducidos. Siendo el objetivo de las extracciones no impactar el sistema operacional y la experiencia de usuario, la limitación de recursos condicionará la técnica de extracción de datos y la periodicidad de las cargas.

- **Imposibilidad de modificar el diseño lógico**. Algunos sistemas operacionales son cerrados y no permiten la modificación del diseño lógico de la base datos. Esto no nos permitirá la creación de índices de bases de datos o particiones para acelerar la extracción de datos. En este caso, el rendi-

miento de las extracciones puede no ser óptimo, haciendo que las cargas se ejecuten durante más tiempo del inicialmente planeado. Esto podría modificar la decisión final sobre la arquitectura.

- **No disponibilidad de réplicas de la BD en tiempo real**. El acceso a la base de datos operacional en lugar de una réplica limita los recursos disponibles. Ello obligará a reducir el impacto en la base de datos para no afectar a los procesos existentes y la experiencia de usuario.

2) Limitaciones de seguridad. La imposibilidad de transmitir datos sin encriptación es un requerimiento que puede bloquear una solución técnica. Las políticas de seguridad en algunas organizaciones no permitirán la transmisión de información por la red sin un protocolo seguro.

3) Limitaciones de red. La velocidad de transferencia es clave en una solución de BI en tiempo real cuando el tiempo entre dos cargas consecutivas de datos es reducido y el volumen de datos es elevado. Una red de comunicaciones con una carga de velocidad de transferencia baja hará que el tiempo de carga sea elevado, aumentando el riesgo de superposición de cargas y de obtención de datos más allá del $t_{acción}$.

4) Limitaciones de hardware. Los recursos del hardware deben ser suficientemente holgados para el proceso de los datos dentro de la ventana de la µ-carga o para la combinación de los resultados de las subconsultas en una solución con federación de datos.

5) Limitaciones de software

- **Federación de datos no soportada**. En el caso de optar por una solución con federación de datos, el software de BI debe soportar esta funcionalidad. Algunas soluciones BI no soportan esta funcionalidad. Si se dispone de software de BI que no soporta la federación de datos, se deberá cambiar la solución técnica o plantearse un cambio de herramienta de BI.

- **Base de datos sin opción de particionamiento**. Algunos paquetes comerciales de base de datos del mercado no soportan el particionamiento de tablas. Otros paquetes lo ofrecen como extensión opcional al software básico. Es conveniente saber si el software de base de datos existente en la organización (usualmente se da continuidad al software conocido dentro de una organización) soporta esta funcionalidad.

- **Base de datos sin opción de índices avanzados**. Existen diferentes tipos de índices que permiten un acceso eficaz a los datos. En estos índices se incluyen los índices de tipo bitmap y los índices basados en funciones. Es conveniente identificar los índices necesarios para una extracción eficaz y rápida, y comprobar si la base de datos del sistema operacional puede soportar este tipo de índices. Si no fuese el caso, se debería cambiar la

técnica de extracción o esperar un rendimiento inferior, que podría ser crítico.

Por tanto, es necesario incluir personal con conocimientos en todos estos ámbitos técnicos en el diseño de la arquitectura de un sistema BI en tiempo real.

5.4. BI en tiempo real y la nube (*cloud*)

Llegados a este punto, es el momento de introducir un componente que ya es una realidad en el mundo de la tecnología: la nube (*cloud* en inglés).

No podemos obviar el papel determinante del *cloud* desde el punto de vista tecnológico a la hora de plantearnos una solución tecnológica para un sistema informático, incluido un sistema de BI en tiempo real.

En este capítulo vamos a ver las consideraciones a tener en cuenta cuando queremos desarrollar una solución de BI en tiempo real con componentes de la arquitectura presentes en la nube. Empezaremos hablando del tiempo de acceso a los datos en función de su ubicación para continuar después hablando de la extracción de información de las diferentes fuentes de datos.

5.4.1. Tiempo de acceso a los datos (t_{acceso})

El *cloud*, por el hecho de estar ubicado fuera de un centro de proceso de datos local, introduce un retraso considerable en el acceso a los datos. Cualquier acceso a un centro de proceso de datos será mucho más eficiente en lo que se refiere al acceso a los mismos.

5.4.2. Arquitecturas de extracción de información

La arquitectura del sistema BI, si bien es también muy importante en un sistema BI tradicional (con una frecuencia de ejecución diaria), no llega al punto de criticidad del sistema BI en tiempo real, ya que la ventana de ejecución es mayor, y la ejecución de la carga de datos puede absorber con más margen los retrasos producidos por el acceso a los datos.

En cambio, en un sistema BI en tiempo real, donde el flujo de información puede ser prácticamente constante y podemos necesitar un t_{acceso} a los datos muy bajo, deberemos tener muy en cuenta la arquitectura de nuestro sistema.

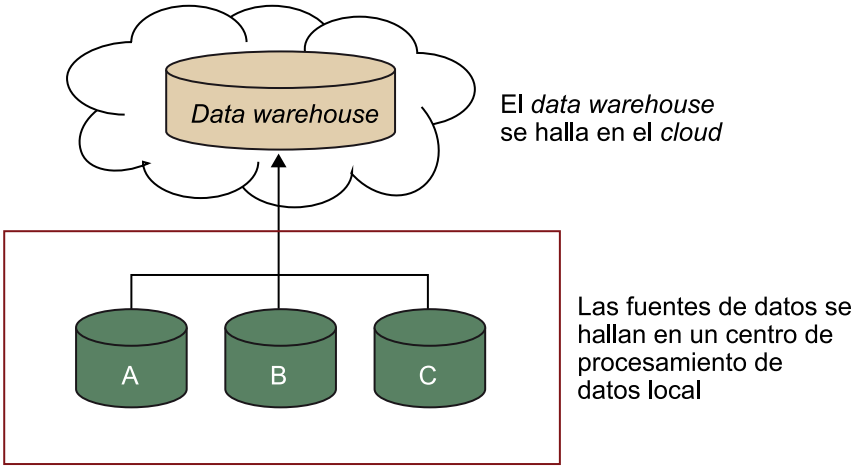
A continuación mostraremos una serie de escenarios de arquitectura y comentaremos cómo pueden afectar al rendimiento de nuestro sistema de BI en tiempo real.

Local-to-cloud

Por arquitectura *local-to-cloud* entendemos lo siguiente:

- Fuentes de datos en un centro de procesamiento de datos local
- *Data warehouse* en el *cloud*

Arquitectura *local-to-cloud*



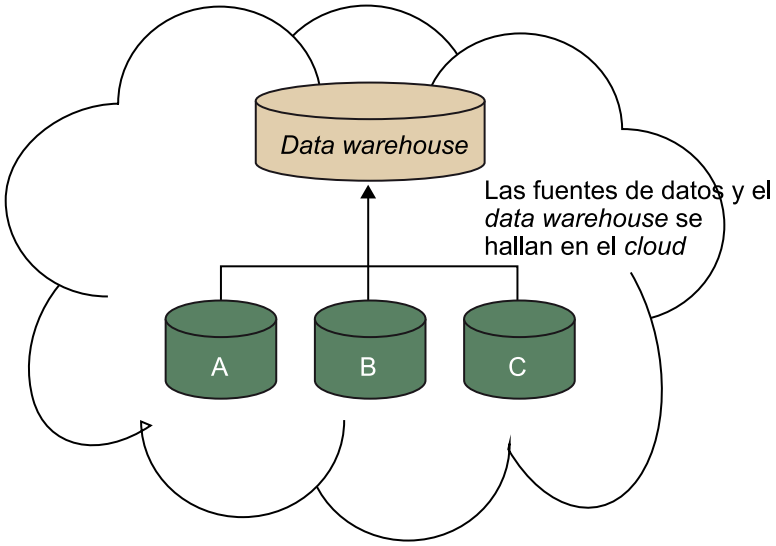
Esta arquitectura implica un t_{acceso} a los datos afectado por la transferencia de datos al *cloud*. Este hecho provocará cierto retraso en la disponibilidad de los datos. En el caso en que el t_{acceso} a los datos > $t_{acción}$, esta solución no será efectiva y deberemos buscar otra solución arquitectónica para nuestro sistema de BI en tiempo real.

Cloud-to-cloud

Por arquitectura *cloud-to-cloud* entendemos lo siguiente:

- Fuentes de datos en el *cloud*
- *Data warehouse* en el *cloud*

Arquitectura *cloud-to-cloud*



Esta arquitectura implica un t_{acceso} a los datos afectado por la transferencia de datos dentro del *cloud*.

Si hemos optado por una solución en un centro de *cloud computing*, con los diferentes componentes de la arquitectura en el mismo centro de procesamiento de datos en la nube, la transferencia de datos entre componentes debe ser equivalente al de un centro de procesamiento de datos tradicional, con lo que no habrá impacto en las cargas de datos.

Si los diferentes componentes de la arquitectura se hallan en distintos centros de procesamiento de datos, el retraso en el t_{acceso} a los datos sería equivalente al de la arquitectura *local-to-cloud*.

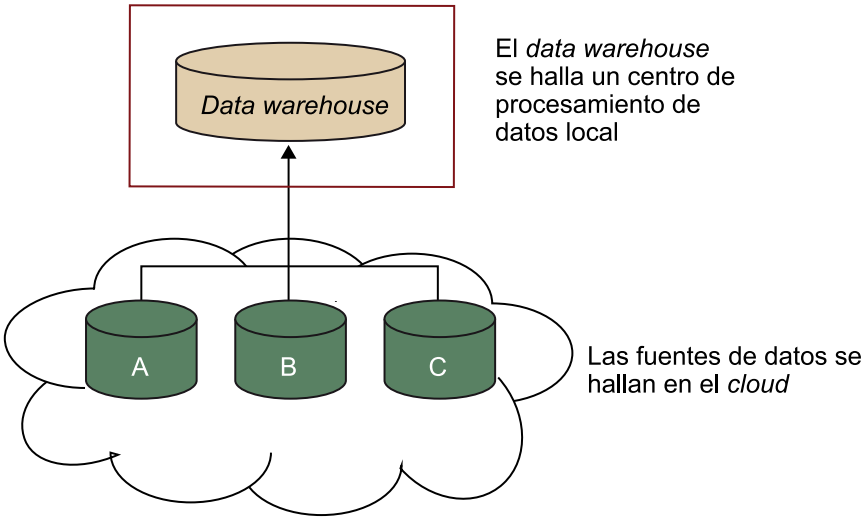
Cloud-to-local

Por arquitectura *cloud-to-local* entendemos lo siguiente:

- Fuentes de datos en el *cloud*

- *Data warehouse* en un centro de procesamiento de datos local

Arquitectura *cloud-to-local*



En este caso, el t_{acceso} a los datos sería equivalente al de la arquitectura *local-to-cloud*. Los datos deben transferirse entre el *cloud* y un centro de procesamiento de datos en local. Este hecho provocará cierto retraso en la disponibilidad de los datos. En el caso en que el t_{acceso} a los datos > $t_{acción}$, esta solución no será efectiva y deberemos buscar otra solución arquitectónica para nuestro sistema de BI en tiempo real.

5.4.3. Paquetes de software en el *cloud*

Cada día es más habitual licenciar una solución de software disponible en el *cloud*, ya que proliferan en la red paquetes de CRM y ERP ofreciendo no solo una solución software sino también el *hosting* de estas soluciones con un precio único, habitualmente por usuario y periodo (mes, año...).

El uso de estos paquetes de software en el *cloud* facilita en gran parte el mantenimiento y la administración del sistema, pero a la vez limitan la libertad de movimientos en una solución disponible en local y en consecuencia su explotación en tiempo real.

En algunos casos, el uso de una aplicación en el *cloud* puede venir acompañada de una limitación en el ancho de banda utilizado o un coste basado en el uso de este. Si este es el caso, la extracción de datos de manera masiva desde la aplicación en el *cloud* puede suponer costes adicionales que repercutirían de manera continua en el proyecto.

Interfaces de extracción de información en el *cloud*

En los sistemas *cloud, e*n la mayoría de casos no disponemos de acceso a la fuente de datos (base de datos), sino que los datos están disponibles mediante otras interfaces, siendo las más comunes los ficheros planos y la existencia de API específicas de aplicación.

Vamos ver con más detalle estos tres tipos de fuentes de datos:

- **Ficheros planos**: En el caso de los ficheros planos, la aplicación suele generar extracciones periódicas de datos. En función del proveedor del paquete de software, es posible configurar las extracciones con los datos en los cuales estamos interesados, lo que permite un poco de flexibilidad a dichas extracciones. La periodicidad en la generación de los ficheros de texto, en el mejor de los casos puede negociarse con el proveedor, pero no suelen permitir una frecuencia muy alta, como la necesaria en el caso del BI en tiempo real. Por tanto, esta solución limita en gran medida una solución de BI en tiempo real.

- **API específica de la aplicación**: Algunas soluciones en el *cloud* incorporan una API que puede ser accedida para obtener datos. Esta API puede responder a llamadas de servicios web, por ejemplo. Con esta solución superamos la limitación de los ficheros de texto sobre la periodicidad de las extracciones, ya que los datos se obtendrán cuando se invoque la llamada de la API. Sin embargo, es muy habitual que la API sea cerrada y no ofrezca flexibilidad sobre los datos extraídos. El hecho de trabajar con una API suele limitar en gran medida la información obtenida en las extracciones. A esto hay que añadir el coste del desarrollo a medida, derivado de la extracción de datos mediante una API, cosa que complicará las cargas de datos y elevará el coste de la solución de BI en tiempo real. Por tanto, esta solución también limita en gran medida una solución de BI en tiempo real.

- **Base de datos**: En el caso de disponer de acceso a una base de datos de la aplicación en el *cloud*, el uso que podemos hacer de ella suele ser muy limitado. Hay que tener en cuenta que es probable que dicha base de datos sea compartida por los diferentes clientes de la aplicación. En consecuencia, no dispondremos de acceso ni en modo administrador ni en modo propietario del esquema de bases de datos, sino tan solo en modo lectura en el caso de que gocemos de ese privilegio.
  Tampoco podremos modificar el esquema propietario para añadir disparadores que nos permitan capturar altas, bajas y modificaciones, ni crear tablas específicas para la carga de datos, ni añadir índices o particionar tablas para conseguir un acceso más eficiente a los datos durante su extracción. Tampoco podremos instalar software de captura de transacciones de base de datos mediante la inspección de los ficheros de *log* de la base de datos. Por tanto, esta solución puede limitar en gran medida una solución de BI en tiempo real.

Nota

Cuando se habla de privilegio en el caso de tener acceso de lectura a la base de datos, debemos dejar claro el porqué. Hay que tener en cuenta que el proveedor del software no puede controlar el uso que un cliente hará de dicho acceso, con lo cual podría estar cargando el sistema con consultas pesadas o muy frecuentes. Por tanto, queda claro que es un privilegio poder tener acceso de lectura a la base de datos.

6. Métricas de evaluación

6.1. Análisis de coste (TCO)

Hablar de TCO en una solución de BI en tiempo real es similar a hablar de TCO en un proyecto de BI tradicional, salvo por algunas diferencias específicas al BI en tiempo real. Para analizar las diferencias, debemos hacer una comparativa de componentes específicos de un proyecto de BI tradicional y una solución de BI en tiempo real. Hay que tener en cuenta que típicamente implementaremos una solución de BI tradicional con una capa adicional para soportar el BI en tiempo real. Raramente veremos una solución únicamente de BI en tiempo real, a pesar de que podría darse el caso.

En la siguiente tabla se muestran estos componentes específicos para el BI tradicional y las dos alternativas de diseño más utilizadas en el BI en tiempo real:

- **Federación pura**: Combinación de los resultados de las consultas al *data warehouse* y a la fuente de datos (sistema operacional).

- **Federación con µ-cargas**: Combinación de los resultados de las consultas al *data warehouse* y a una partición *online* cargada mediante µ-cargas (extracciones de datos realizadas periódicamente con un tiempo entre cargas reducido).

Componente	BI tradicional	BI en tiempo real	
		Federación pura	Federación con µ-cargas
Base de datos (*data warehouse*)	X		
Herramienta ETL	X		
Herramienta de carga *online* de registros (*streaming*)			X
Herramienta BI	X		
Herramienta BI con soporte para federación de datos en múltiples fuentes de datos		X	X

6.1.1. Consideraciones en el cálculo del TCO

La solución tecnológica determina los componentes del sistema. El TCO debe incluir el coste de esos componentes a nivel de licencias de producto y mantenimiento.

Sin embargo, hay que tener en cuenta todo el coste derivado de la utilización de este software. A continuación se muestra una lista de áreas donde podemos incurrir en coste para cada uno de este software:

- Hardware.
- Servidores donde se instalará el software. Hay que tener en cuenta los múltiples entornos que podemos tener: desarrollo, integración, producción…
- Tiempo de dimensionamiento, búsqueda y compra de servidores.
- Tiempo de instalación y configuración de servidores.
- Posibles ampliaciones futuras.
- Software.
- Licencias de las herramientas citadas anteriormente.
- Licencias de otro software necesario. Esto incluye el sistema operativo.
- Licencias de soporte y mantenimiento para años venideros.
- Instalación de software.
- Instalación de futuros *upgrades* y *patches*.
- Migración.
- Migraciones de software si el software actual no dispone de las capacidades deseadas y se requiere un cambio de herramienta BI (por ejemplo).
- Servidores donde realizar la migración.
- Desarrollo de proyecto de migración.
- Mitigación de riesgos.
- Desarrollo de la solución BI en tiempo real.
- Proyecto completo.
- Formación de equipo.
- Coste de contratación de servicios o personal, de manera opcional.
- Formación al departamento de explotación.
- Coste de personal del equipo del proyecto y otros recursos (*management*, personal de negocio, etc.).

Conviene ser muy estrictos en el cálculo del TCO para poder calcular con precisión el retorno de inversión y el periodo de *payback*. Estos dos valores se describen y detallan a continuación.

6.2. Análisis de retorno de inversión y *payback* (*ROI* & *payback*)

Como en todo proyecto, una solución de BI en tiempo real debe hacer un estudio del retorno de inversión (ROI) y el *payback* para poder tomar una decisión sobre su viabilidad a cierto plazo.

Empecemos haciendo un repaso a las definiciones de *ROI* y *payback*.

6.2.1. Retorno de inversión (*ROI*)

El *ROI* se define como la eficiencia de una inversión en función del gasto realizado. El *ROI* se expresa como un porcentaje y se calcula mediante la siguiente fórmula:

$$ROI\% = \frac{\text{Beneficios derivados de la inversión} - \text{Coste de la inversión}}{\text{Coste de la inversión}} \times 100$$

Beneficios derivados de la inversión	100.000 €
Gastos de inversión	40.000 €

$$ROI\% = \frac{100.000\ € - 40.000\ €}{40.000\ €} \times 100 = 150\%$$

6.2.2. *Payback period*

El periodo de *payback* se define como el periodo de tiempo necesario para recuperar la inversión realizada en un proyecto. Se expresa en unidades de tiempo y se calcula mediante la siguiente fórmula:

$$Payback\ \ period = \frac{\text{Gastos de inversión}}{\text{Beneficio neto de inversión por unidad de tiempo}}$$

Gastos de inversión	40.000 €
Beneficio neto de inversión por unidad de tiempo	5.000 € / 2 meses

$$Payback\ \ period = \frac{40.000\ €}{5.000\ € / 2\ \text{meses}} = 16\ \text{meses}$$

6.2.3. Consejos para calcular el *ROI* y el *payback period*

A pesar de disponer de una fórmula simple para el cálculo del *ROI* y el *payback period*, el problema con el que nos podemos topar en este momento es el de saber qué valores vamos a introducir en cada caso.

Estos consejos van dirigidos a obtener las variables de las fórmulas de *ROI* y *payback period*.

1) Beneficios

Como beneficio se entiende la cuantificación en la unidad de medida moneda que aporta la solución de BI. Para poder cuantificar estos beneficios, debemos encontrar maneras de traducir los beneficios no tangibles a esa unidad de medida moneda.

A continuación se muestran varios ejemplos de cómo calcular beneficios en soluciones de BI en tiempo real.

Tiempo ahorrado en la búsqueda y análisis de la información

Sin BI en tiempo real, los usuarios deben estar consultando los datos antes del $t_{acción}$ para poder extraer la información necesaria para hacer una toma de decisiones apropiada. Este tiempo puede ser relativamente corto, pero la periodicidad de dicho proceso hace que, al cabo de muchas repeticiones, la cantidad de tiempo total sea muy elevada.

$$\text{Beneficio} = \text{Tiempo invertido tradicionalmente} \times \text{Coste horario recurso} \times \text{Núm. veces en el periodo}$$

Disminución de penalizaciones por violación de acuerdos de nivel de servicio

Poder reaccionar de manera rápida y eficiente a los cambios de los datos puede traducirse en la disminución o total desaparición de penalizaciones por violación de acuerdos de nivel de servicio. Si ello se produce, podemos calcular los beneficios como la suma de las penalizaciones no ocurridas.

$$\text{Beneficio} = (\text{Núm. penalizaciones anterior} - \text{Núm. penalizaciones posterior}) \times \text{Coste de penalización}$$

Incremento de ventas por incremento en la agilidad de reacción

En los casos en que poder reaccionar rápidamente ante peticiones de compra puede suponer el éxito en esa operación, puede realizarse una estimación de dicho incremento de ventas a partir de los datos previos a la nueva solución de BI en tiempo real y los actuales, obtenidos con la implantación de la solución.

$$\text{Beneficio} = (\text{Ingresos actuales} - \text{Ingresos anteriores}) \times \text{Factor asumible al BI en tiempo real}$$

Incremento en la retención de clientes

$$\text{Beneficio} = \text{Número de clientes retenidos} \times \text{Ingresos medios obtenidos por cliente retenido}$$

Cuando la solución de BI en tiempo real permite incrementar la retención de clientes, podemos estimar los beneficios en función del número de clientes retenidos y los ingresos no perdidos de estos.

Reducción del número de productos fabricados con defectos

$$\text{Beneficio} = (\text{Núm. defectos pre BI} - \text{Núm. defectos post BI}) \times \text{Coste de un producto defectuoso}$$

Donde:

- Núm. defectos pre BI = Estimación del número de productos defectuosos antes de que se implantara el sistema de BI en tiempo real.
- Núm. defectos post BI = Número de productos defectuosos necesitados para la detección de un error en la fabricación, una vez que la solución BI está disponible.

En una fábrica, la detección de productos defectuosos por el departamento de calidad es muy importante, ya que puede suponer enormes pérdidas si la producción continúa durante cierto periodo de tiempo. En este caso, el beneficio no es tal, sino que se trata del ahorro producido al detectar un defecto repetido en la cadena de montaje.

2) Gastos de inversión

Los gastos de inversión son los derivados de la implantación de la solución de BI en tiempo real y su mantenimiento.

La siguiente lista incluye varias partidas a tener en cuenta:

- Coste de consultoría (si existe)
- Coste de personal interno (TI)
- Coste de personal interno (no IT)
- Coste de compra de licencias (primer año)
- Coste de licencias de mantenimiento (años siguientes)
- Coste de hardware y mantenimiento
- Coste de formación

7. Casos de éxito

A continuación se muestran casos de éxito de BI en tiempo real para ilustrar los beneficios que puede aportar una solución de este tipo.

7.1. Evolución diaria de las ventas y comparación con el histórico reciente

Groupalia es una compañía encuadrada en el área de e-business, especializada en la venta de planes de ocio con descuentos. En la actualidad, cuenta con más de seis millones de usuarios en los dos países donde opera: España e Italia.

En el pasado, Groupalia disponía de una solución de BI tradicional con cargas diarias. Esto les proporcionaba información sobre las ventas del día anterior con una latencia de 24 horas.

La necesidad de poder reaccionar rápidamente a los cambios para poder tomar decisiones de manera eficaz llevaron a Groupalia a plantearse una solución de BI en tiempo real, donde pudiesen ver la evolución de las ventas durante el día, con una latencia de los datos no superior a 15'. De esta manera, Groupalia podría detectar la detención del flujo de ventas, analizar los motivos y ofrecer soluciones en un tiempo mínimo, con una mínima afectación a la operativa diaria.

Inicialmente se planteó un aumento de la frecuencia de las cargas de datos existentes (carga diaria). Sin embargo, la duración de la carga excedía los 15' debido principalmente a la necesidad de mantener las tablas agregadas sincronizadas. Por ello, este planteamiento fue descartado.

Debido a la imposibilidad de ejecutar una carga cada 15', se optó por un rediseño de la solución BI tradicional existente para su evolución a un sistema de BI en tiempo real. Para ello se optó por una solución con federación de datos basado en:

- ***Data warehouse***: Cargas diarias durante la noche.

- **Partición *online***: Cargas de ventas actuales cada 15' sobre la partición *online* que va acumulando los datos de las ventas realizadas durante el día. Cabe destacar que las ventas cargadas en esta partición *online* son utilizadas durante la carga diaria para cargar el *data warehouse* y evitar la reextracción y tratamiento de los datos. De esta manera se consiguen optimizar las cargas diarias y reducir su duración.

La arquitectura está basada en tecnología Oracle y se compone de:

- **Base de datos**: Oracle Database Enterprise Edition 11g. Contiene el área temporal de datos utilizada durante la ETL, las tablas de hechos y dimensiones del *data warehouse*, las tablas agregadas del *data warehouse* y la partición *online* con las ventas diarias.

- **Herramienta ETL**: Oracle Warehouse Builder 11g. Esta herramienta ETL está integrada en la base de datos Oracle Database.

- **Herramienta BI**: Oracle Business Intelligence Enterprise Edition 11g. Permite la federación de datos. En este caso accede tanto al *data warehouse* como a la partición *online* y combina ambas consultas para proporcionar un resultado único al usuario.

Gracias a esta solución, Groupalia cuenta con información de ventas en tiempo casi real, que compara con información histórica sobre ventas. Además proyecta esa información a la hora de cierre contable, con lo que puede anticipar resultados con cierto margen de seguridad.

Esta información facilita a Groupalia detectar anomalías en la evolución de las ventas y le permite llevar a cabo acciones correctivas durante el día, sin tener que esperar a los resultados la mañana siguiente.

7.2. Mantener nivel de acuerdo de servicio en un centro de soporte

La corporación Thomson Reuters es la compañía líder en publicación y distribución de información en el mundo de los negocios y profesionales de la información.

Con el fin de mantener el nivel de servicio de soporte en su centro de soporte, Thomson Reuters se hallaba ante la necesidad de disponer de información casi en tiempo real sobre el número de casos abiertos que no habían recibido atención. El objetivo era responder a las peticiones de soporte en un máximo de 5' desde la creación del caso.

Con un $t_{acción}$ tan reducido, la ejecución de la carga diaria de su solución BI tradicional era imposible. La solución a este problema fue la evolución de la solución BI tradicional a una solución de BI en tiempo real.

En este caso, el tipo de análisis a realizar era reducido en comparación con el análisis de los casos de clientes que se estaba realizando habitualmente: tan solo se necesitaba información básica del caso, las actividades relacionadas e

información básica del cliente. Por tanto, la cantidad de información a obtener en tiempo real era mucho menor que la disponible en el *data warehouse*. Esto planteaba dos alternativas:

- Acceso directo a la base de datos operacional
- Creación de µ-cargas

Thomson Reuters es una multinacional con presencia global y con distintos centros de soporte esparcidos por todo el mundo. La carga de trabajo del sistema operacional era bastante elevada en su momento, lo que no permitía sobrecargar el sistema con constantes consultas. Este factor fue decisivo para descartar el acceso directo a la base de datos operacional.

Finalmente se creó una solución basada en µ-cargas para cargar la partición *online* del *data warehouse* con información básica sobre las peticiones de soporte, las actividades asociadas a estas y los clientes. La ejecución de estas cargas se planificó con una frecuencia de una ejecución cada 2', con lo cual había margen para las acciones pertinentes por el personal del centro de soporte sin superar el $t_{acción}$ para ese proceso de negocio.

Debido al gran volumen de datos a extraer del CRM que actuaba como fuente de datos, se tuvo que reducir la cantidad de información incluida en las µ-cargas para que estas pudiesen ejecutarse dentro de la ventana acordada (30 segundos de ejecución). Esta ventana reducida estaba influenciada por la urgencia en conseguir los datos y también por el objetivo de reducir el impacto en el sistema CRM origen de los datos. Como se ha indicado anteriormente, esta reducción de información en la partición *online* respecto a la información del *data warehouse*, no era crítica debido al tipo de análisis requerido para información *online*. Eso permitió la aceptación de los usuarios de la solución planteada.

Por lo que respecta a los datos cargados en la partición *online*, los datos de esta eran eliminados al iniciar un nuevo día, volviendo a cargar los datos diarios en dicha partición. Por su parte, al ejecutar la carga diaria del *data warehouse*, estos datos eran cargados nuevamente pero con toda la información necesaria en el *data warehouse*. Es decir, los datos de la partición *online*, en este caso, no servían como fuente de datos para el *data warehouse*.

El resultado de aplicar una solución de BI en tiempo real mediante la federación de datos con µ-cargas sobre una partición *online* fue un sistema donde, en tiempo casi real, el personal de los centros de soporte global de Thomson Reuters podía disponer de información acerca de los casos abiertos y no atendidos. Con esta información, los agentes de estos centros de soporte global pueden atender estos casos de manera urgente para cumplir el acuerdo de nivel de servicio vigente.

Resumen

En este módulo se ha definido el concepto de sistema de BI en tiempo real a partir de la necesidad de cubrir unos requerimientos analíticos dentro de un periodo de tiempo máximo llamado $t_{acción}$.

Hemos visto cómo algunos procesos dentro de una organización tienen un $t_{acción}$ menor de 1 día y que estos, por tanto, necesitan una frecuencia de carga más alta que el ofrecido por el BI tradicional basado en cargas diarias. Y dentro de este grupo de procesos, algunos tienen un $t_{acción}$ que tiende a 0'. En este caso, la ventana de ejecución de las cargas es tan pequeña que no permite la ejecución de una carga de datos. En este momento es donde aparece un nuevo diseño del sistema BI para dar una solución a dichos requerimientos: la federación de datos. Dentro de la federación de datos se han mostrado alternativas de diseño para federación de datos y la transferencia de estos.

Se han indicado los beneficios y los posibles problemas de dichas alternativas, las diferentes técnicas y las herramientas de software necesarias para poder implementar una solución BI en tiempo real.

Durante este módulo también se han dado consejos para el cálculo de las métricas de evaluación ROI y *payback period*.

Por último, se han mostrado casos de éxito para una mejor comprensión de los beneficios que el BI en tiempo real puede aportar a una organización.