Desarrollo de una solución *Business Intelligence* mediante un paradigma de *Data Lake*

José María Tagarro Martí
Grado de Ingeniería Informática

Consultor: **Humberto Andrés Sanz**
Profesor: **Atanasi Daradoumis Haralabus**

13 de enero de 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	Desarrollo de una solución Business Intelligence mediante un paradigma de Data Lake
Nombre del autor:	José María Tagarro Martí
Nombre del consultor:	Humberto Andrés Sanz
Fecha de entrega (mm/aaaa):	01/2019
Área del Trabajo Final:	Business Intelligence
Titulación:	*Grado de Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

Este trabajo implementa una solución de Business Intelligence siguiendo un paradigma de Data Lake sobre la plataforma de Big Data Apache Hadoop con el objetivo de ilustrar sus capacidades tecnológicas para este fin.

Los almacenes de datos tradicionales necesitan transformar los datos entrantes antes de ser guardados para que adopten un modelo preestablecido, en un proceso conocido como ETL (Extraer, Transformar, Cargar, por sus siglas en inglés).

Sin embargo, el paradigma Data Lake propone el almacenamiento de los datos generados en la organización en su propio formato de origen, de manera que con posterioridad puedan ser transformados y consumidos mediante diferentes tecnologías *ad hoc* para las distintas necesidades de negocio.

Como conclusión, se indican las ventajas e inconvenientes de desplegar una plataforma unificada tanto para análisis Big Data como para las tareas de Business Intelligence, así como la necesidad de emplear soluciones basadas en código y estándares abiertos.

Abstract (in English, 250 words or less):

This paper implements a Business Intelligence solution following the Data Lake paradigm on Hadoop's Big Data platform with the aim of showcasing the technology for this purpose.

Traditional data warehouses require incoming data to be transformed before being stored by means of an ETL process (Extract-Transform-Load) to adopt a predefined data model.

However, the Data Lake paradigm proposes storing first the organization's data in its original format in such a way that they can later be transformed and consumed by ad hoc processes based on the business needs.

Finally, there is a review of the advantages and disadvantages of deploying a unified data platform for both Big Data and traditional Business Intelligence use cases as well as the importance of using solutions based on open source and open standards.

Palabras clave (entre 4 y 8):

Business Intelligence, Data Lake, Data Warehouse, Big Data, Hadoop, ETL, Schema-on-read, Hive

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

El concepto de Business Intelligence engloba cualquier tipo de solución que permita procesar la información de manera que de soporte a la toma de mejores decisiones de negocio. Sin embargo, históricamente ha estado vinculado al despliegue de tecnologías *Data Warehouse* y OLAP con las que generar una variedad de informes periódicos mediante procesos ETL por lotes.

Además, y pese a la existencia de soluciones *Open Source* notables como Pentaho, estos despliegues se han venido basando en tecnologías propietarias desarrolladas por un reducido grupo de empresas tales como Oracle, IBM, Microsoft y SAP.

No obstante, en los últimos años y como consecuencia del advenimiento de las aplicaciones a escala Internet, han aparecido una serie de tecnologías orientadas al procesamiento de grandes volúmenes de datos en tiempos muy reducidos, englobadas bajo la denominación Big Data y que presentan algunas diferencias notables con respecto a la mencionada pila tecnológica de BI:

- La solución principal de Big Data, Apache Hadoop, se distribuye bajo el modelo *Open Source* y los líderes del sector (recientemente fusionados) operan bajo un modelo mixto *freemium* (Cloudera) y de pago por soporte (Hortonworks).

- Los datos pueden escribirse en cualquier formato y posteriormente leerse con otro distinto sin necesidad de procesos ETL y según las necesidades de la aplicación que los va a consumir, gracias a la característica de *schema-on-read* que posee el almacén de datos de Hadoop, Apache Hive.

- Los datos pueden procesarse ahora de forma distribuida y en tiempo cuasi-real sin necesidad de recurrir a procesos por lotes y con independencia de su volumen.

Tales ventajas han dado lugar a un nuevo modelo de gestión de la información donde los datos no necesitan ser transformados antes de guardarse y donde se modifican dinámicamente según sea necesario a la hora de ser consumidos por las distintas aplicaciones que los requieran.

En este nuevo paradigma, todos los procesos de negocio que generan datos en la organización envían una copia en su propio formato interno a un almacén global conocido como Data Lake, donde se guardan inmediatamente en ese mismo formato tanto si están estructurados, semi-estructurados o no estructurados. Las aplicaciones pueden acceder a los mismos mediante una diversidad de tecnologías y paradigmas, quedando la implementación concreta tras una capa de abstracción.

El modelo ha sido adoptado por muchas organizaciones, que han creado equipos de trabajo específicos para explorar las posibilidades de creación de valor y obtención de ventajas competitivas de los datos ya existentes mediante nuevas técnicas de análisis apoyadas en herramientas de aprendizaje automático, aunque sin embargo siguen operando sus despliegues Data Warehouse habituales, dando lugar a una duplicidad de soluciones para un mismo problema.

1.2. Objetivos del Trabajo

Se propone desarrollar una solución Business Intelligence mediante un paradigma de Data Lake implementado sobre Apache Hadoop en vez de un almacén de datos tradicional y que elimine la necesidad de contar con dos plataformas tecnológicas distintas en la organización, una orientada a DW+BI tradicional y otra orientada *a Big Data Analytics*.

En particular, como resultado del proyecto se obtendrán las siguientes capacidades tecnológicas y objetivos de negocio:

- Data Warehouse con integración de datos de distintos orígenes.
- *Reporting* y análisis OLAP para soporte a la toma de decisiones.
- *Dashboard* tanto histórico como en tiempo cuasi-real con herramientas de visualización avanzadas para soporte a la toma de decisiones y control del alineamiento estratégico.
- Herramientas de análisis predictivo y de minería de datos que den soporte a las actividades de investigación y desarrollo en *Big Data*.

La implementación se llevará a cabo sobre Hortonworks HDP, distribución de Hadoop completamente *Open Source*, utilizando Apache Hive como almacén de datos del Data Lake y Apache Kylin somo servidor OLAP.

1.3. Enfoque y método seguido

Se trabajará sobre una empresa modelo que a priori debería reunir muchas de las siguientes condiciones:

- Generar un número elevado de transacciones para que obtener un caso ilustrativo de *Big Data*. Se deberán evitar, por tanto, los productos de precio elevado o compra no periódica.
- Múltiples sedes, para ilustrar informes agregados o despliegues de BSC regionales.
- Un catálogo relativamente reducido de SKU, para evitar tener que generar un número elevado de datos que no serían relevantes desde el punto de vista del trabajo a desarrollar.

- Un tipo de productos que sea fácilmente categorizable, de manera que se facilite la generación de modelos predictivos.

En base a las restricciones mencionadas, se ha escogido una empresa modelo del sector *retail* de alimentación que se encuentra en rápida expansión y con multitud de aperturas de tiendas por toda la geografía nacional.

Dada la imposibilidad de contar con datos reales, se generará un juego de datos de simulación que permita alimentar el sistema a desarrollar y que aproveche cualquier información de dominio público que se encuentre disponible sobre las empresas del sector de interés, al objeto de aumentar el realismo en la medida de lo posible.

Además, se definirán los distintos orígenes de datos de la organización, así como los procesos cliente desplegados en los distintos niveles de la empresa modelo (operativo, gestión y estratégico) y de acuerdo con los objetivos establecidos en el apartado anterior. En este caso se tendrá en cuenta que el objetivo de este trabajo es la evolución tecnológica de las plataformas que soportan las soluciones Business Intelligence, por lo que no será necesario generar información sobre toda la cadena de valor y bastará con ejemplos ilustrativos, por ejemplo, del área comercial.

Una vez definido el problema, se propondrá la arquitectura de la solución, considerando las distintas soluciones tecnológicas en función de los objetivos ya planteados y los datos disponibles (volumen, frecuencia de actualización, disponibilidad, etcétera).

A continuación, se implementará la solución mediante Apache Hadoop y el despliegue de las distintas herramientas necesarias, siempre *Open Source* y sobre hardware virtualizado.

Finalmente, se llevará a cabo un análisis comparativo de la solución propuesta frente a las distintas alternativas existentes, como paso previo a la presentación de las conclusiones sobre el trabajo.

1.4. Planificación del Trabajo

1.4.1. Relación de tareas y diagrama de Gantt del proyecto

Tabla 1 Planificación de tareas

ID	Tarea	Inicio	Fin	Duración
0	Descripción del caso	15/10/18	18/10/18	4
6	Modelo de organización	15/10/18	15/10/18	1
2	Arquitectura existente	16/10/18	16/10/18	1
3	Modelo de datos OLTP	17/10/18	17/10/18	1
5	Desafíos actuales	18/10/18	18/10/18	1
7	Juego de datos de prueba	19/10/18	25/10/18	5
10	Venta en tienda	19/10/18	22/10/18	2

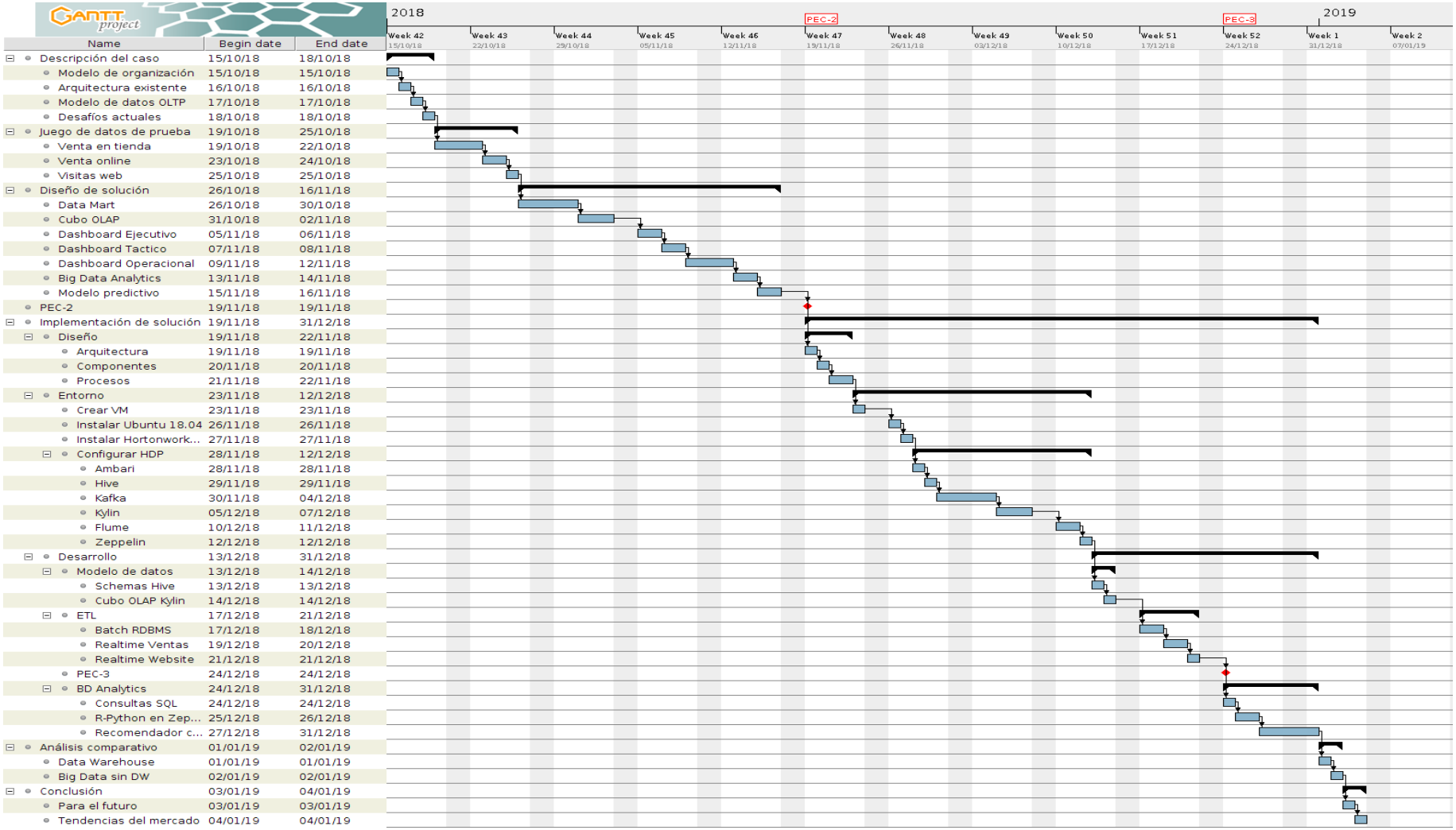
ID	Tarea	Inicio	Fin	Duración
11	Venta online	23/10/18	24/10/18	2
12	Visitas web	25/10/18	25/10/18	1
13	Diseño de solución	26/10/18	16/11/18	16
14	Data Mart	26/10/18	30/10/18	3
15	Cubo OLAP	31/10/18	02/11/18	3
16	Dashboard Ejecutivo	05/11/18	06/11/18	2
17	Dashboard Tactico	07/11/18	08/11/18	2
18	Dashboard Operacional	09/11/18	12/11/18	2
20	Big Data Analytics	13/11/18	14/11/18	2
19	Modelo predictivo	15/11/18	16/11/18	2
791	PEC-2	19/11/18	19/11/18	0
21	Implementación de solución	19/11/18	31/12/18	31
44	Diseño	19/11/18	22/11/18	4
22	Arquitectura	19/11/18	19/11/18	1
23	Componentes	20/11/18	20/11/18	1
24	Procesos	21/11/18	22/11/18	2
54	Entorno	23/11/18	12/12/18	14
57	Crear VM	23/11/18	23/11/18	1
56	Instalar Ubuntu 18.04	26/11/18	26/11/18	1
55	Instalar Hortonworks HDP	27/11/18	27/11/18	1
58	Configurar HDP	28/11/18	12/12/18	11
59	Ambari	28/11/18	28/11/18	1
60	Hive	29/11/18	29/11/18	1
61	Kafka	30/11/18	04/12/18	3
62	Kylin	05/12/18	07/12/18	3
63	Flume	10/12/18	11/12/18	2
64	Zeppelin	12/12/18	12/12/18	1
65	Desarrollo	13/12/18	31/12/18	13
66	Modelo de datos	13/12/18	14/12/18	2
45	Schemas Hive	13/12/18	13/12/18	1
49	Cubo OLAP Kylin	14/12/18	14/12/18	1
67	ETL	17/12/18	21/12/18	5
46	Batch RDBMS	17/12/18	18/12/18	2
47	Realtime Ventas	19/12/18	20/12/18	2
48	Realtime Website	21/12/18	21/12/18	1
69	BD Analytics	24/12/18	31/12/18	6
51	Consultas SQL	24/12/18	24/12/18	1
52	Spark-R en R-Studio	25/12/18	26/12/18	2
53	Recomendador compra	27/12/18	31/12/18	3

ID	Tarea	Inicio	Fin	Duración
858	PEC-3	24/12/18	24/12/18	0
70	Análisis comparativo	01/01/19	02/01/19	2
72	Data Warehouse	01/01/19	01/01/19	1
74	Big Data sin DW	02/01/19	02/01/19	1
71	Conclusión	03/01/19	04/01/19	2
76	Para el futuro	03/01/19	03/01/19	1
75	Tendencias del mercado	04/01/19	04/01/19	1

1.4.2. Diagrama de Gantt

En la página siguiente se presenta el diagrama de Gantt del proyecto con indicación de la programación de tareas en la línea temporal del proyecto, así como indicación de los hitos principales.

Ilustración 1 Diagrama de Gantt



Name	Begin date	End date
Descripción del caso	15/10/18	18/10/18
Modelo de organización	15/10/18	15/10/18
Arquitectura existente	16/10/18	16/10/18
Modelo de datos OLTP	17/10/18	17/10/18
Desafíos actuales	18/10/18	18/10/18
Juego de datos de prueba	19/10/18	25/10/18
Venta en tienda	19/10/18	22/10/18
Venta online	23/10/18	24/10/18
Visitas web	25/10/18	25/10/18
Diseño de solución	26/10/18	16/11/18
Data Mart	26/10/18	30/10/18
Cubo OLAP	31/10/18	02/11/18
Dashboard Ejecutivo	05/11/18	06/11/18
Dashboard Tactico	07/11/18	08/11/18
Dashboard Operacional	09/11/18	12/11/18
Big Data Analytics	13/11/18	14/11/18
Modelo predictivo	15/11/18	16/11/18
PEC-2	19/11/18	19/11/18
Implementación de solución	19/11/18	31/12/18
Diseño	19/11/18	22/11/18
Arquitectura	19/11/18	19/11/18
Componentes	20/11/18	20/11/18
Procesos	21/11/18	22/11/18
Entorno	23/11/18	12/12/18
Crear VM	23/11/18	23/11/18
Instalar Ubuntu 18.04	26/11/18	26/11/18
Instalar Hortonwork...	27/11/18	27/11/18
Configurar HDP	28/11/18	12/12/18
Ambari	28/11/18	28/11/18
Hive	29/11/18	29/11/18
Kafka	30/11/18	04/12/18
Kylin	05/12/18	07/12/18
Flume	10/12/18	11/12/18
Zeppelin	12/12/18	12/12/18
Desarrollo	13/12/18	31/12/18
Modelo de datos	13/12/18	14/12/18
Schemas Hive	13/12/18	13/12/18
Cubo OLAP Kylin	14/12/18	14/12/18
ETL	17/12/18	21/12/18
Batch RDBMS	17/12/18	18/12/18
Realtime Ventas	19/12/18	20/12/18
Realtime Website	21/12/18	21/12/18
PEC-3	24/12/18	24/12/18
BD Analytics	24/12/18	31/12/18
Consultas SQL	24/12/18	24/12/18
R-Python en Zep...	25/12/18	26/12/18
Recomendador c...	27/12/18	31/12/18
Análisis comparativo	01/01/19	02/01/19
Data Warehouse	01/01/19	01/01/19
Big Data sin DW	02/01/19	02/01/19
Conclusión	03/01/19	04/01/19
Para el futuro	03/01/19	03/01/19
Tendencias del mercado	04/01/19	04/01/19

1.5. Sumario de los productos obtenidos

Como resultado del trabajo desarrollado se obtendrán los siguientes productos:

- Descripción del caso, incluyendo el modelo de organización interna de la empresa, la arquitectura previa de sus Sistemas de Información, los principales modelos de datos de los OLTP desplegados y los desafíos a los que se enfrenta.

- Un juego de datos de simulación con tres orígenes distintos (ventas en tienda, online y visitas a la página web) y almacenados en diferentes sistemas y formatos: RDBMS y *flat files*.

- Diseño de la solución de *Business Intelligence* sobre *Data Lake* propuesta mediante una serie de casos de uso típicos:

    - *Data Marts* para distintas áreas de la empresa.

    - Cubo OLAP e informes asociados.

    - *Dashboard* para los distintos niveles de gestión de la organización: estratégico, táctico y operativo.

    - Un modelo predictivo de utilidad para el departamento de *marketing* y consistente en la optimización de una campaña mediante la información existente acerca de las preferencias de los clientes.

- Descripción de la arquitectura de la solución planteada, con detalle de los componentes y procesos necesarios.

- Implementación del *Data Lake* en la plataforma de *Big Data* de Hortonworks, incluyendo la creación de los distintos modelos de datos y sus procesos asociados, así como la generación de los cubos OLAP y ejemplos de casos de uso para consultas exploratorias y analítica predictiva.

- Análisis comparativo de una solución tradicional *Business Intelligence* basada en *Data Warehouse* frente a la solución basada en *Data Lake* descrita.

- Conclusiones del trabajo y consideraciones para el futuro.

1.6. Capítulos centrales de la memoria

- Capítulo 2. Descripción del caso empresarial y de la situación actual de los sistemas de información y desafíos de negocio en la empresa modelo.

- Capítulo 3. Juego de datos de simulación en los sistemas OLTP y de producción que servirán para demostrar las capacidades de la nueva plataforma.

- Capítulo 4. Descripción de los casos de uso escogidos para ilustrar la aplicación de la tecnología *Big Data* al *Business Intelligence* con introducción de las herramientas necesarias.

- Capítulo 5. Implementación de la solución propuesta: diseño, despliegue de los sistemas y herramientas necesarios y configuración de los procesos y productos instalados.

2. Descripción de la empresa modelo

2.1. Modelo de gestión de la información de partida

Para ilustrar la aplicación de las tecnologías *Big Data* en la el desarrollo de una solución *Business Intelligence* se recurrirá a caracterizar una empresa modelo que se encuentra en el primer nivel del *Business Intelligence Maturity Model* (Rajterič 2010, p. 55) de Gartner: "*unaware"*.

En este nivel no existe consistencia entre los datos de diferentes fuentes primarias ni tampoco una interpretación unificada de su significado debido a que las distintas áreas de la empresa consumen datos generados en sus propias herramientas y sistemas de información.

Con respecto a la metodología de trabajo para adquirir conocimiento, los datos se procesan mayoritariamente mediante hojas de cálculo a partir de volcados de tablas relacionales facilitadas por los administradores de las distintas bases de datos del nivel operativo.

El modelo descrito no es compatible con una rápida expansión de la organización ya que no se cuenta con las herramientas adecuadas para asegurar la implementación en los niveles táctico y operativo de las estrategias dictadas por la dirección, por lo que resulta idóneo para ilustrar las características de la nueva plataforma tecnológica.

2.2. Descripción del caso: Supermercados Sierra

Desde que comenzara como una pequeña tienda de barrio a finales de los años noventa, la cadena de supermercados Sierra ha experimentado un crecimiento vertiginoso: en la actualidad cuenta con 800 localizaciones, con 120 nuevas aperturas el año pasado y 250 más previstas para el presente ejercicio. Todo gracias al gran éxito de su modelo de negocio basado en proximidad, precios bajos y entrega a domicilio ultra-rápida.

2.2.1. Modelo de negocio

Tradicionalmente el pequeño supermercado no ha podido competir en precio con las grandes superficies debido a su menor poder de negociación con los proveedores (Nicholson y Young 2012, p. 2) pero Sierra ha conseguido ser competitivo gracias a su pertenencia a una gran central de compras a nivel europeo y el consumidor ha respondido con entusiasmo a la propuesta de buenos precios sin necesidad desplazarse.

De hecho, uno de los servicios de mayor éxito de la empresa es la posibilidad de realizar un pedido por Internet (también mediante un teléfono móvil inteligente) y recibir la compra en casa en menos de una hora, toda una revolución en el sector y que sólo ha sido posible gracias a la extrema proximidad que caracteriza a la cadena y que le ha permitido

utilizar técnicas habituales de reparto de comida rápida pero aplicadas a la entrega de cestas de compra de pequeño tamaño.

Los repartidores son empleados externos a la tienda que reciben los pedidos mediante una aplicación móvil, acuden a la tienda, seleccionan los productos y los entregan sin apenas intervención del personal propio.

2.2.2. Dificultades del modelo

No obstante, el pequeño tamaño combinado con la diversidad geográfica cada vez mayor de las tiendas ha creado una situación no exenta de desafíos, sobre todo desde el punto de vista logístico:

- Las preferencias de los consumidores varían en función de la geografía por lo que la gama de productos debe poder adaptarse a los gustos locales.
- El espacio disponible en los lineales es muy reducido ya que la superficie media de los supermercados de la cadena no alcanza los 150m^2 y no cuentan con ningún espacio de almacenamiento.
- Las campañas de publicidad de las grandes marcas tienen un gran impacto en las preferencias de compra, por lo que la empresa necesita adaptarse rápidamente a una demanda cambiante.

2.2.3. Cadena de suministro

La solución ideada por Sierra ha sido, de nuevo, tomar prestada una solución de otro sector que también se ve afectado por problemas de espacio y una amplia red de tiendas: el farmacéutico.

Se alquila un pequeño almacén en una ubicación estratégica que se encuentre a menos de 30 minutos en coche de las tiendas a las que abastece. El almacén recibe mercancía diariamente mediante una ruta logística tradicional, pero sirve pedidos varias veces al día a los distintos supermercados mediante furgonetas de pequeño tamaño y en función de la demanda. La siguiente figura ilustra el proceso completo:

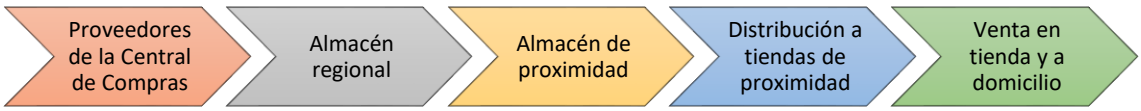


Ilustración 2 Cadena de suministro de Supermercados Sierra

En los últimos meses ha aumentado considerablemente el número de pedidos que no pueden ser entregados a tiempo por falta de stock en la tienda más cercana al cliente, lo que ha llevado a los responsables de la cadena a plantearse el despliegue de alguna solución que les permita adelantarse a la demanda y mantener su principal ventaja competitiva.

Por otra parte, también preocupa la falta de control efectivo sobre las tiendas, especialmente en la ejecución de acciones estratégicas, sobre todo de marketing. El motivo principal es la estructura de recursos humanos con la que cuenta la empresa, que debido al pequeño tamaño de las tiendas no cuenta con personal de gestión con dedicación completa y tiene un único coordinador de zona en cada localidad a las órdenes del delegado regional.

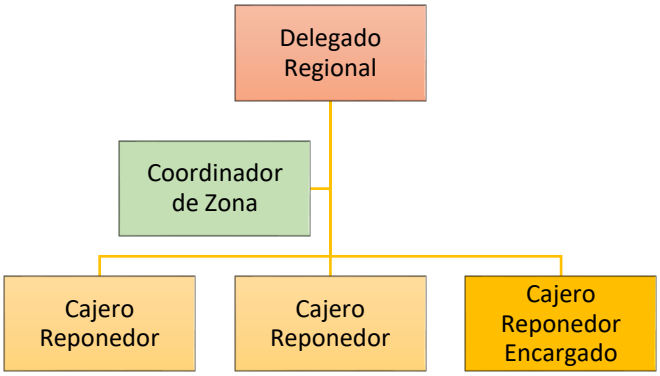


Ilustración 3 Cadena de mando del nivel de gestión

Se espera que la nueva solución sea capaz también de establecer objetivos claros al personal de tienda que permitan ejecutar de una manera eficaz las decisiones estratégicas de la dirección de Sierra.

2.3. Arquitectura de los Sistemas de Información

2.3.1. Hardware

La empresa cuenta con un pequeño centro de datos en su sede central para alojar los principales aplicativos, todos ellos paquetes comerciales estándar. Originalmente era un servidor *bare-metal* ejecutando Windows 2000 Server, pero en la actualidad todos los servicios están virtualizados mediante *hypervisors*.

Además, cuenta con varios servidores dedicados en *leasing* y con el servicio de gestión integral contratado al propio proveedor de alojamiento que ejecutan la solución *e-commerce* que permite capturar los pedidos de los clientes.

2.3.2. Software

En cuanto a software, la empresa cuenta con distintos paquetes que se han ido adquiriendo y actualizando a medida que se necesitaban en las distintas:

- Contabilidad y finanzas. Alojado en la sede central, acceso mediante servicios de terminal.

- Nóminas y RR.HH. Alojado en la sede central, acceso mediante servicios de terminal.

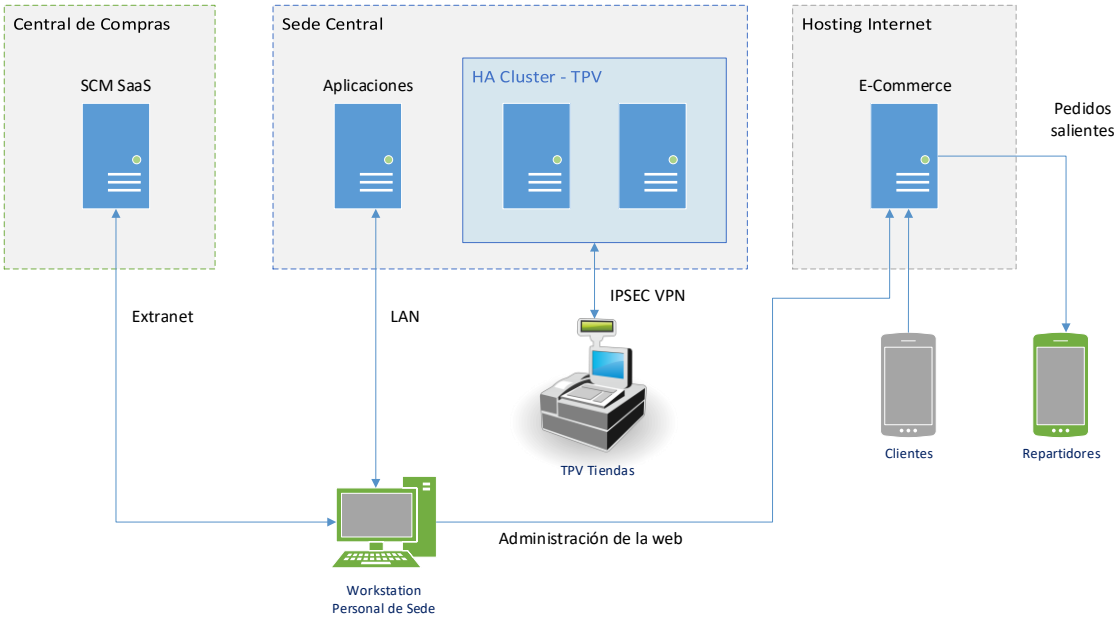
- Compras y almacén. Proporcionada por la propia central de compras a la que pertenece la cadena de supermercados por lo que se accede mediante su extranet.

- Facturación y TPV. Alojado en la sede central y siguiendo una arquitectura de alta disponibilidad. El acceso mediante servicios de terminal remoto.

2.3.3. Topología

El siguiente diagrama muestra la arquitectura actual de los sistemas desplegados en Supermercados Sierra, así como su relación con el proceso de venta.

Ilustración 4 Arquitectura de sistemas actual



2.4. Fuentes primarias de datos

Se deben seleccionar aquellos Sistemas de Información que mejor puedan ilustrar las aplicaciones de la nueva plataforma tecnológica de acuerdo con los objetivos del proyecto.

En particular, se han tenido en cuenta los siguientes criterios:

- Número elevado de transacciones para ilustrar la capacidad de procesar grandes volúmenes de información.

- Múltiples sedes en diferentes localizaciones geográficas para ilustrar la capacidad de agregación de datos.

- Catálogo reducido de SKU fácilmente categorizables para facilitar casos de uso de aplicación de modelos predictivos.

Como resultado, los Sistemas de Información y fuentes primarias de datos escogidos son:

- Aplicativo de TPV, que recoge la información de las transacciones en las tiendas físicas (base de datos relacional).
- Aplicativo de tienda online, que recoge la información de los pedidos a domicilio (base de datos relacional).
- Servidor web, que recoge la información las visitas (*flat files* con los registros de acceso).

2.4.1. Modelo de datos del aplicativo TPV

Nos basaremos en el juego de datos producido por la empresa de consultoría de ciencia de datos Dunnhumby, propiedad de la cadena de supermercados británica Tesco y liberado para su uso por la comunidad académica como parte del proyecto Source Files[1].

Se trata de información correspondiente al consumo a lo largo de dos años de 2.500 familias que compran con frecuencia en una cadena de supermercados e incluye información completa de cada *ticket* o cesta de la compra, así como indicadores demográficos del comprador.

Por otra parte, se incluye información sobre 30 campañas promocionales basadas en cupones descuento y en las que ha participado un subconjunto de las familias por lo que resulta una fuente de datos idónea para demostrar capacidades de análisis predictivo y segmentación.

2.4.2. Modelo de datos de la solución *e-commerce*

Debido a las características ya descritas del modelo de negocio, el proceso de compra en línea es análogo a un pedido por teléfono en el que opcionalmente es posible realizar el pago mediante tarjeta de crédito y dónde como resultado del pedido se genera una lista de compra que es notificada a los repartidores de servicio mediante una aplicación móvil.

Podemos modelarlo siguiendo un paradigma de publicación suscripción, en el que los pedidos online se publican en una cola a la que están suscritos los empleados de reparto del supermercado correspondiente, obteniendo así una oportunidad de demostrar las capacidades de ingestión y procesamiento de eventos en tiempo casi real de la plataforma tecnológica propuesta.

Así, un servicio web asociado a la tienda online publica los pedidos firmes a una cola a la que está suscrito el servicio web que alimenta la aplicación móvil que utilizan los repartidores.

[1] https://www.dunnhumby.com/careers/engineering/sourcefiles

2.4.3. Modelo de datos del registro de acceso del servidor web

Se basa en ficheros de texto plano que contienen una línea por cada petición de acceso que ha recibido el servidor por parte de un navegador web. La información capturada incluye:

- Fecha y hora.
- URL solicitada.
- Dirección IP desde la que se ha hecho la petición.
- *User-agent* o cadena identificativa del dispositivo, navegador y sistema operativo.

Se guarda un fichero de log por cada día y servidor incluido en el pool del balanceador de carga de la web, por lo que para tratar estos datos será necesario combinarlos en orden.

El siguiente diagrama muestra el proceso de creación de logs segmentados debido al uso de balanceadores de carga:

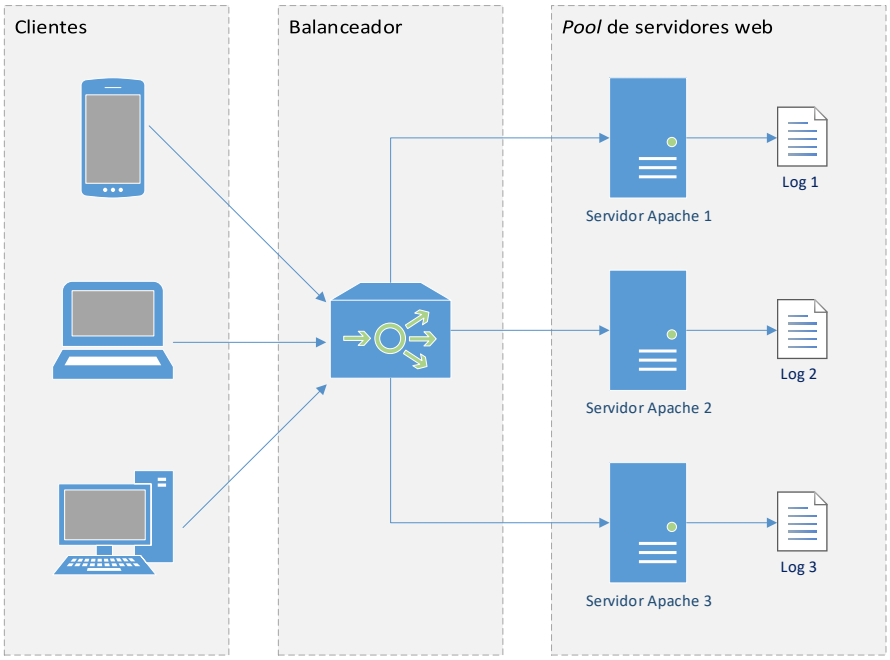


Ilustración 5 Generación de logs de Apache

3. Juego de datos de simulación

3.1. Introducción

Pese a que la propuesta original del trabajo se basaba en la generación de un juego de datos sintético a partir de la información pública disponible sobre empresas del sector de gran consumo, durante la fase de investigación se han localizado varios juegos de datos con información real y de uso libre para la comunidad académica.

Se trata de los juegos de datos publicados a través del proyecto Source Files de la empresa Dunnhumby[2], una consultora con presencia en treinta países y especializada en ciencia de datos aplicada al sector de gran consumo.

3.2. Juego de datos “The complete journey”

3.2.1. Descripción

Este apartado se basa en la información proporcionada con el propio juego de datos.

El juego de datos contiene información de las compras realizadas por 2.500 familias que durante dos años fueron clientes frecuentes de una determinada cadena de supermercados.

Además, dado que las compras se registraron por medio de la tarjeta de fidelidad de la cadena, se ha incluido información de 30 campañas de marketing de las que han sido objeto muchas de las familias, así como las compras asociadas a los cupones descuento incluidos en las campañas.

Por lo que respecta al volumen de datos, la distribución es la siguiente:

- Transacciones:
    - 2.581.075 líneas de recibo a lo largo de dos años, con 2.500 identificadores únicos de unidad familiar y 92.339 productos distintos.
- Familias:
    - 2.500 en total, incluyendo 801 con información demográfica adicional: rango de edad, estado civil, nivel de ingresos, propiedad de la vivienda, composición y tamaño de la familia, así como número de hijos.
- Campañas:

[2] https://www.dunnhumby.com/about-us

- 30 campañas en total, realizadas a lo largo de dos años y dirigidas a 1.584 de las familias, que promocionan 44.133 productos mediante 1.135 cupones.

- Productos:

  - 92.353 productos con descripción, categoría, departamento, marca y fabricante, de los cuales 68.377 productos cuentan con indicación de si han sido incluidos en acciones promocionales en algún momento y tienda dados.

3.2.2. Modelo de datos

El siguiente diagrama muestra las relaciones incluidas en el juego de datos, así como sus asociaciones con indicación de la multiplicidad. Además, se han separado en dos bloques, las tablas de datos frente a las de búsqueda.

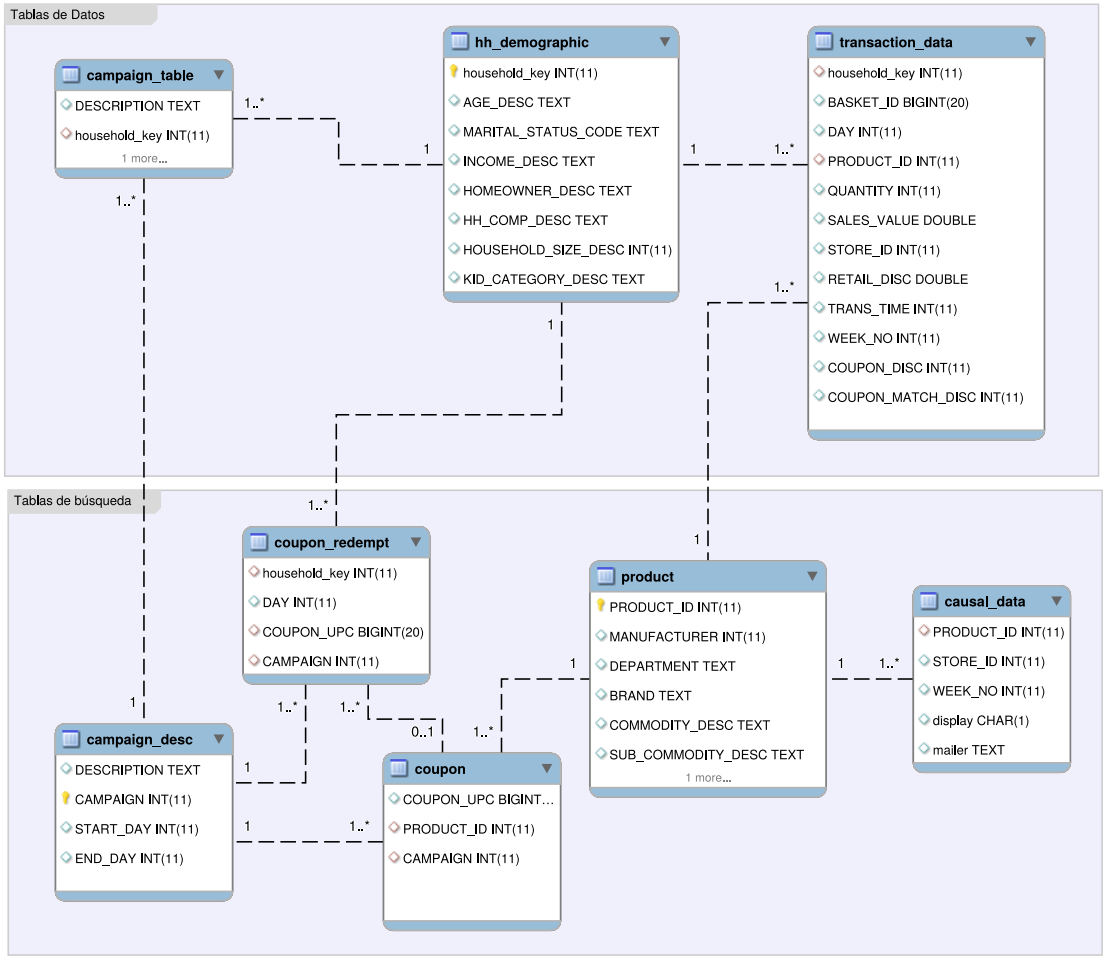


Ilustración 6 Modelo de la base de datos de Dunnhumby

3.2.3. Descripción detallada de las relaciones

Las tablas originales se proporcionan en formato CSV, uno por relación.

HH_DEMOGRAPHIC

Columna	Descripción
HOUSEHOLD_KEY	Identificador único de la unidad familiar
AGE_DESC	Rango de edad estimado
MARITAL_STATUS_CODE	Estado civil: casado (M), soltero (S), desconocido (U)
INCOME_DESC	Nivel de ingresos
HOMEOWNER_DESC	Situación de la vivienda: propiedad, alquiler
HH_COMP_DESC	Composición de la unidad familiar
HOUSEHOLD_SIZE_DESC	Tamaño de la unidad familiar
KID_CATEGORY_DESC	Número de niños

TRANSACTION_DATA

Columna	Descripción
HOUSEHOLD_KEY	Identificador único de la unidad familiar
BASKET_ID	Identificador del tique de compra
DAY	Dia de la transacción
PRODUCT_ID	Identificador único del producto
QUANTITY	Numero de productos del tique
SALES_VALUE	Cantidad de dólares recibidos por el vendedor
STORE_ID	Identificador único de tienda
COUPON_MATCH_DISC	Descuento aplicado para igualar cupón del fabricante
COUPON_DISC	Descuento aplicado por cupón del fabricante
RETAIL_DISC	Descuento aplicado por tarjeta de cliente del vendedor
TRANS_TIME	Hora del día en que ocurrió la transacción
WEEK_NO	Semana de la transacción (secuencial de 1 a 102)

PRODUCT

Columna	Descripción
PRODUCT_ID	Identificador único del producto
DEPARTMENT	Agrupación de productos similares – Nivel 1
COMMODITY_DESC	Agrupación de productos similares – Nivel 2
SUB_COMMODITY_DESC	Agrupación de productos similares – Nivel 3
MANUFACTURER	Código identificador del fabricante
BRAND	Marca blanca o gran marca
CURR_SIZE_OF_PRODUCT	Tamaño del paquete

CAMPAIGN_TABLE

Columna	Descripción
HOUSEHOLD_KEY	Identificador único de la unidad familiar
CAMPAIGN	Identificador único de campaña (de 1 a 30)
DESCRIPTION	Tipo de campaña: A, B o C

CAMPAIGN_DESC

Columna	Descripción
CAMPAIGN	Identificador único de campaña (de 1 a 30)
DESCRIPTION	Tipo de campaña: A, B o C
START_DAY	Fecha de inicio de la campaña
END_DAY	Fecha de fin de la campaña

COUPON

Columna	Descripción
CAMPAIGN	Identificador único de campaña (de 1 a 30)
COUPON_UPC	Identificador único del cupón (familia y campaña)
PRODUCT_ID	Identificador único del producto

COUPON_REDEMT

Columna	Descripción
HOUSEHOLD_KEY	Identificador único de la unidad familiar
DAY	Dia de la transacción
COUPON_UPC	Identificador único del cupón (familia y campaña)
CAMPAIGN	Identificador único de campaña (de 1 a 30)

CAUSAL_DATA

Columna	Descripción
PRODUCT_ID	Identificador único del producto
STORE_ID	Identificador único de tienda
WEEK_NO	Semana de la transacción (secuencial de 1 a 102)
DISPLAY	Ubicación de la oferta en la tienda
MAILER	Ubicación de la oferta en el folleto

DISPLAY, MAILER

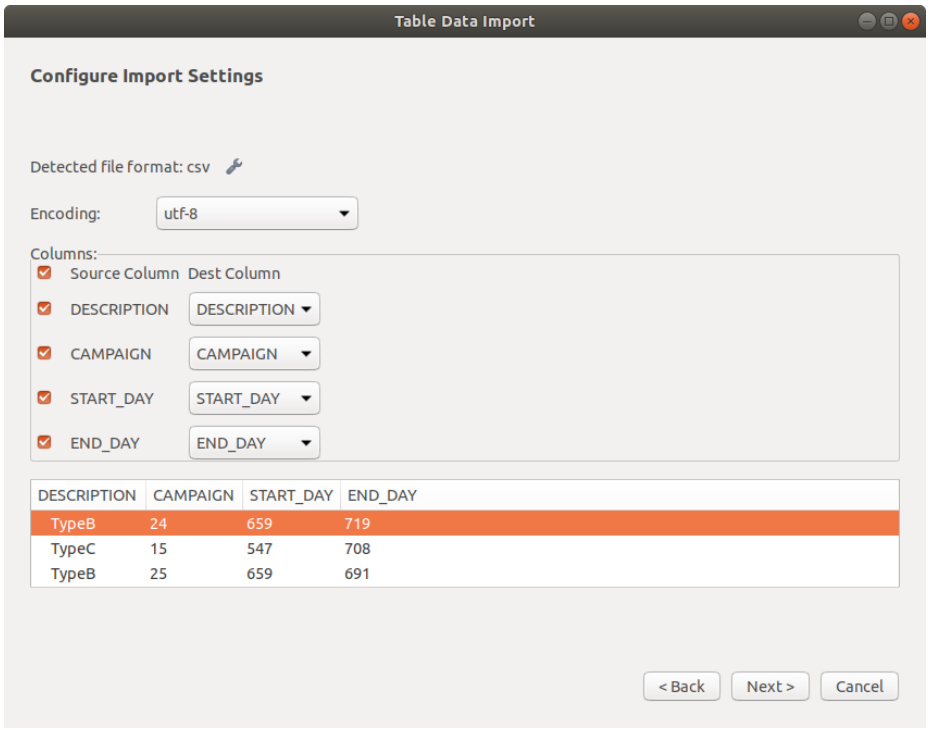
Display	Mailer
0 – No se muestra	0 – Sin anuncio
1 – Principio de tienda	A – Recuadro en página interior
2 – Final de tienda	C – Artículo en página interior
3 – Cabecera en principio de tienda	D – Recuadro en portada
4 – Cabecera en mitad de tienda	F – Recuadro en contraportada
5 – Cabecera en final de tienda	H – Desplegable en portada
6 – Lateral de cabecera	J – Desplegable cupón interior
7 - Pasillo	L – Desplegable en contraportada
9 – Ubicación secundaria	P – Cupón en página interior
A - Estantería	X – Suelto en página interior
	Z – Suelto en portada, contra o desplegable

3.2.4. Implementación en MySQL

El juego de datos se proporciona como ficheros de texto plano correspondientes a las distintas relaciones, por lo que, al objeto de replicar un entorno realista de fuente de datos primaria, se han importado en la base de datos relacional MySQL de Oracle.

```
total 808M
-rw-r--r-- 1 chema92 chema92  540 oct 18  2007 campaign_desc.csv
-rw-r--r-- 1 chema92 chema92  94K oct 18  2007 campaign_table.csv
-rw-r--r-- 1 chema92 chema92 664M oct 18  2007 causal_data.csv
-rw-r--r-- 1 chema92 chema92 2,7M oct 18  2007 coupon.csv
-rw-r--r-- 1 chema92 chema92  53K abr  7  2008 coupon_redempt.csv
-rw-r--r-- 1 chema92 chema92  44K abr  4  2008 hh_demographic.csv
-rw-r--r-- 1 chema92 chema92 6,2M ago 22  2008 product.csv
-rw-r--r-- 1 chema92 chema92 136M abr  7  2008 transaction_data.csv
```

Para ello, se ha utilizado la herramienta de gestión MySQL Workbench por su capacidad de inferir y editar el esquema de la base de datos a partir de los tipos presentes en las columnas del fichero CSV:



Sin embargo, la velocidad de importación de MySQL Workbench no es idónea[3], por lo que para generar los esquemas. La importación de datos se ha realizado mediante la herramienta de línea de comandos *mysqlimport* que permite cargar datos en formato CSV de una manera extremadamente rápida ya que escribe directamente en los ficheros asociados a la tabla sin utilizar ni la pila de red ni el *parser* SQL.

[3] http://gdsotirov.blogspot.com/2018/07/mysql-wb-data-import-slow.html

El comando utilizado ha sido el siguiente, para cada fichero CSV:

```
mysqlimport -u root -p --fields-terminated-by ','\
--ignore-lines 1 sierra_online                    \
/var/lib/mysql-files/campaign_desc.csv
```

3.3. Juego de datos “Let’s get kind of real”

Como origen de datos se ha optado por recurrir de nuevo al proyecto Source Files de Dunnhumby, que ofrece otro juego de datos de gran volumen, pero en esta ocasión sintéticos.

3.3.1. Modelo de datos

En total se cuenta con 400 millones de transacciones durante 117 semanas en 760 tiendas, incluyendo 5.000 productos y 500.000 clientes distintos.

TRANSACTIONS

Columna	**Descripción**
SHOP_WEEK	Semana de la transacción en formato YYYYWW
SHOP_DATE	Fecha de la compra en formato YYYYMMDD
SHOP_WEEKDAY	Día de la semana, comenzando en Domingo (1-7)
SHOP_HOUR	Slot horario de la compra: 0 – 00:00 hasta 00:59
QUANTITY	Numero de productos distintos en la cesta
SPEND	Gasto asociado a los productos adquiridos
PROD_CODE	Identificador único de producto
PROD_CODE_10	Nivel de jerarquía del producto
PROD_CODE_20	Nivel de jerarquía del producto
PROD_CODE_30	Nivel de jerarquía del producto
PROD_CODE_40	Nivel de jerarquía del producto
CUST_CODE	Identificador del cliente
CUST_PRICE_SENSITIVITY	Sensibilidad a precios: bajo (LA), medio (MM), alto (UM), desconocido (UX)
CUST_LIFESTAGE	Situación personal del cliente: YA=Joven adulto, OA=Edad madura, YF=Familia joven, OF=Familia madura, PE=Pensionista, OT=Otros, XX
BASKET_ID	Identificador del tique de compra
BASKET_SIZE	Tamaño de la cesta: pequeño (S), mediano (M), grande (L)
BASKET_PRIZE_SENSITIVITY	Sensibilidad a precios: LA, MM, UM, XX
BASKET_TYPE	Tipo de cesta de la compra: Tienda pequeña, Reponer, Compra completa, XX
BASKET_DOMINANT_MISSION	Frescos, Alimentación, Mixta, No Comida, XX
STORE_CODE	Código de identificación de la tienda
STORE_FORMAT	Formato de la tienda: LS, MS, SS, XLS
STORE_REGION	Región a la que pertenece la tienda

La guía completa[4] adjunta al juego de datos incluye información adicional a la mostrada en este apartado.

3.3.2. Almacenamiento de los datos

Como se ha indicado en la presentación del caso, las ventas por Internet se servirán a la plataforma de *Business Intelligence* en forma de *stream* continuo mediante un paradigma de publicación-suscripción, se creará un script en lenguaje Python que lea los ficheros CSV y publique los datos en tiempo casi real en una cola de Apache Kafka[5], pero respetando los intervalos temporales presentes de origen.

Se ha escogido el lenguaje de programación Python por su creciente popularidad en el campo de análisis y procesamiento de información, así como la disponibilidad de bibliotecas que le permiten interactuar con las distintas aplicaciones del ecosistema Hadoop.

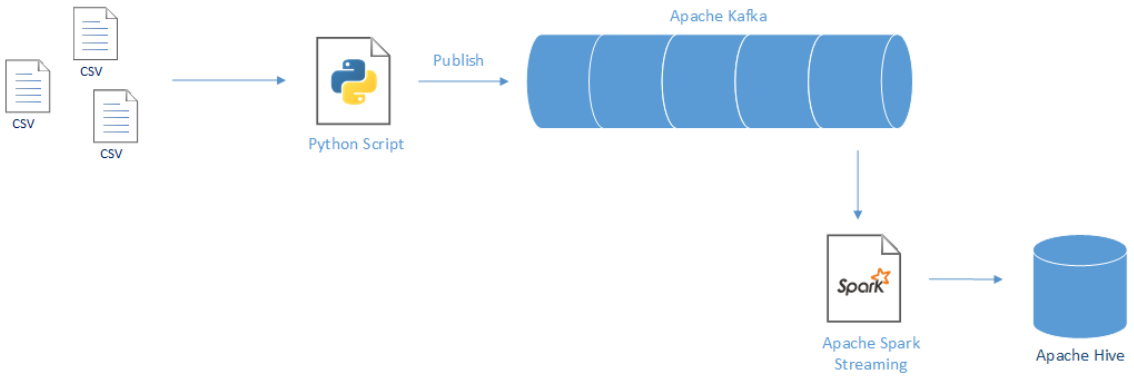


Ilustración 7 *Data streaming* desde flat files con Python y Kafka

En el diagrama se muestra un proceso de Apache Spark Streaming[6] como ejemplo, pero existirán distintos consumidores en función del uso que se haga de la información procedente de la tienda online.

Así, es posible tener un consumidor que actualizará en tiempo real un KPI de un cuadro de mando y otro consumidor que almacena datos históricos en Hive, el Data Warehouse de Hadoop, mediante un proceso por lotes.

Los ficheros con los datos en sí ocupan 40,3GB y se almacenarán en un volumen de red montado en el sistema operativo, es decir, no será necesario que se incluyan en el sistema de ficheros de Hadoop (HDFS) ya que tampoco sería el caso en el escenario propuesto, con los eventos siendo generados desde los sistemas de producción (ver Ilustración 4 Arquitectura de sistemas actual).

---

[4] https://www.dunnhumby.com/sites/default/files/filepicker/1/dunnhumby_-_Let_s_Get_Sort-of-Real_User_Guide.pdf
[5] https://kafka.apache.org/
[6] https://spark.apache.org/streaming/

3.4. Juego de datos EDGAR

Para elaborar datos sintéticos de visitas a la página web partiremos del conjunto de datos EDGAR ofrecido por la Division of Economics and Risk Analysis de la Securities and Exchange Comission de Estados Unidos.

Se ha escogido esta fuente porque los ficheros agregados por día, cuenta con un volumen de tráfico muy elevado y además opera mediante un balanceador de carga, de forma similar a la arquitectura planteada en el caso, lo que nos permitirá reproducir los procesos de unificación de logs desde las distintas instancias de Apache asociadas al *pool* del balanceador.

3.4.1. Estructura de los datos

Los datos se ofrecen en formato CSV e incluyen los siguientes campos:

APACHE LOG

Columna	**Descripción**
IP	Dirección IP desde la que se ha accedido
DATE	Fecha de acceso
TIME	Hora de acceso
ZONE	Identificación del servidor en el *pool* de balanceo
CIK	N/A
ACCESSION	N/A
DOC	N/A
CODE	Código HTTP devuelto por Apache
FILESIZE	N/A
NOREFER	Vale 1 si la solicitud no tiene campo *referrer*
NOAGENT	Vale 1 si la solicitud no tiene campo *user-agent*
FIND	N/A
CRAWLER	Vale 1 si la solicitud proviene de un buscador
BROWSER	Identificador del navegador y sistema operativo

Los campos indicados como N/A no son de aplicación en nuestro estudio ya que se trata de indicadores internos de los documentos oficiales recuperados mediante el servidor web.

3.5. Selección de datos

Una vez se han definido los distintos juegos de datos primarios a simular, es necesario establecer el volumen relativo de los mismos, de manera que el conjunto esté alineado con el caso planteado.

En particular, partiremos de los datos reales de compra en tienda de los que se dispone para estimar tanto las operaciones online como el tráfico del servidor web asociado.

3.5.1. Estimación del volumen de operaciones

La cadena de supermercados LIDL opera siguiendo un modelo de proximidad y cuenta[7] con aproximadamente 500 tiendas en España, con unas ventas netas por valor de 3.594 millones de euros en 2017 por lo que, dado que nuestro conjunto de datos de partida se basa en aproximadamente 700 tiendas, aunque de tamaño más reducido, podemos asimilar sus volúmenes de negocio.

Por otra parte, la muestra de 2.500 unidades familiares que compran con frecuencia y hacen uso de su tarjeta de fidelidad incluye ventas por aproximadamente 3,5 millones de euros por año, pero es necesario tener en cuenta de que se trata de una muestra sesgada que hace uso intensivo de la tarjeta de fidelidad de la cadena de tiendas.

Gracias a un estudio (Mauri 2003, p. 17) de los hábitos de uso de las tarjetas de fidelidad en el sector *retail* de alimentación, hemos obtenido siguientes resultados:

- Las ventas asociadas a tarjetas de fidelidad suponen un 70% del total de ventas del supermercado.
- Un 69% de las tarjetas emitidas llega a usarse en alguna ocasión.
    - De ellas, un 45% se utiliza con frecuencia y suponen el 88% del total de ventas asociadas a tarjetas.

Con esta información podemos estimar que el 31% de las ventas corresponden al segmento de compra frecuente y uso intensivo de la tarjeta de fidelidad y aportan el 62% de las ventas totales del supermercado.

Aplicando la estimación a las cifras de ventas netas de LIDL, obtenemos que 1.114 millones de euros corresponderían al segmento de uso intensivo de la tarjeta y compra frecuente, que estaría constituido por aproximadamente 800.000 unidades familiares.

Finalmente, dado que nuestro conjunto de datos de partida incluye 2.595.732 transacciones, el volumen total de transacciones anuales sería de aproximadamente 415 millones.

Con respeto al porcentaje de ventas de supermercado a través de Internet, la referencia está en el mercado asiático donde alcanza un 22% de cuota (“What’s in-store for online grocery shopping”, 2007) frente al 5% del europeo.

Por lo tanto, teniendo presente que el éxito de nuestra cadena de supermercados se basa en una fuerte relevancia de las ventas por

[7] https://www.lidl.es/es/lidl-espana.htm

Internet, podemos asumir que alcance un nivel similar al del mercado asiático en la actualidad.

Por lo tanto, obtenemos la distribución siguiente:

- 415 millones de transacciones, de las cuales:
    - 324 millones en tienda tradicional.
    - 91 millones por Internet.

Como la cesta media de acuerdo con el conjunto de datos “The complete journey” contiene 9 artículos, en el caso de Internet tendremos aproximadamente 10 millones de compras finalizadas al año. Por lo tanto, asumiendo ratio de conversión típico del 7% (Di Fatta, Patton y Viglia 2018, p. 165), tendremos un tráfico de 1.300 millones de visitantes, suponiendo que los registros de acceso de log del servidor web se limitan a páginas propias, no existen peticiones asíncronas y las imágenes se encuentran alojadas en la red de servidores de contenidos (CDN).

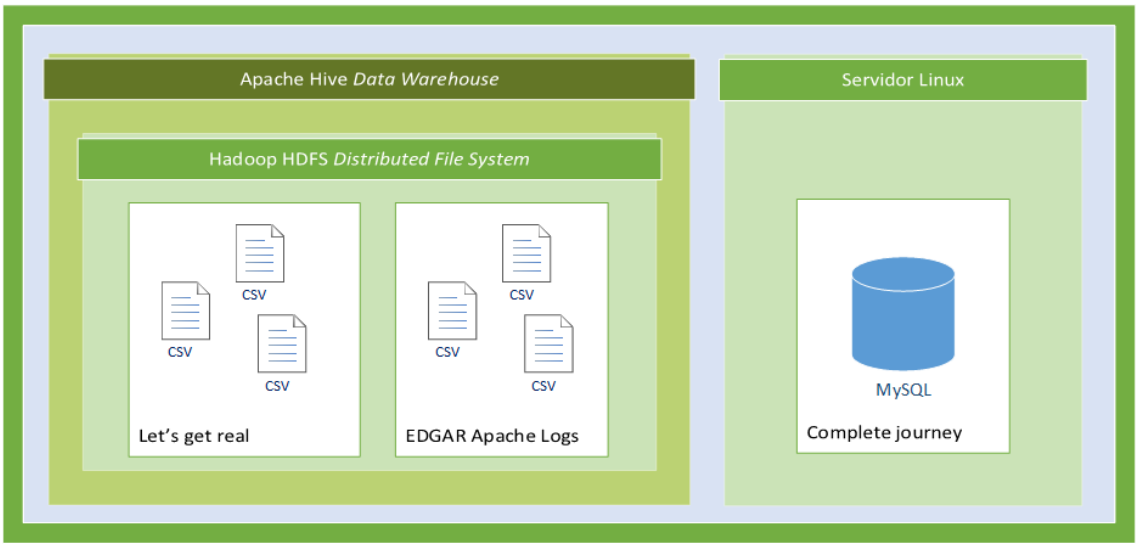
3.5.2. Implementación de los juegos de datos

El juego de datos reales “The complete journey” se ha importado directamente en una base de datos relacional, MySQL.

El juego de datos sintético “Let’s get real” se ha copiado en su formato original a Hadoop HDFS y se ha creado una tabla externa de Apache Hive, el *Data Warehouse* de Hadoop sobre los propios ficheros en su formato original CSV.

El juego de datos EDGAR de logs de Apache también se ha copiado a HDFS siguiendo el mismo procedimiento que para “Let’s get real”.

Ilustración 8 Implementación de los juegos de datos



4. Diseño de la solución y casos de uso típicos

4.1. Introducción a Apache Hadoop

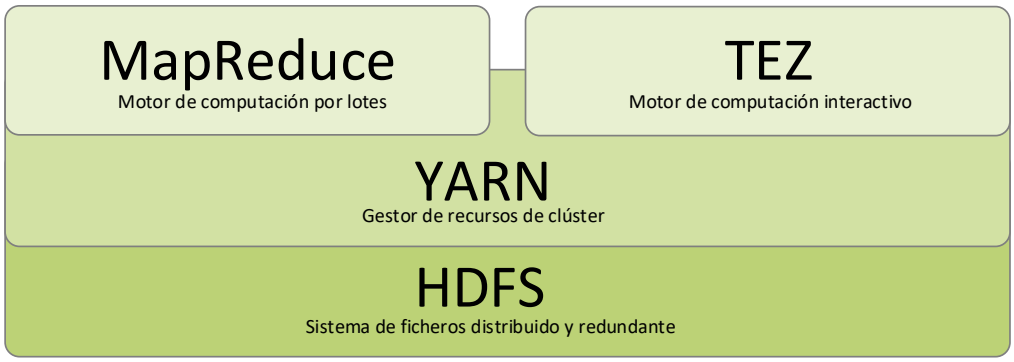
Las soluciones propuestas en este capítulo se basan en Apache Hadoop, un ecosistema de aplicaciones y tecnologías *Open Source* orientadas al procesamiento masivo de datos. Sus elementos principales son:

- Apache HDFS. Un sistema de ficheros distribuidos que opera mediante una API expuesta por un servicio que se ejecuta en cada nodo que forma parte del clúster. Incluye replicación automática de los bloques de datos para ofrecer tanto un mayor caudal de acceso como tolerancia a fallos.

- Apache YARN. Una capa de abstracción que ofrece los recursos de procesamiento de un servidor a distintos motores de computación distribuida, permitiendo que coexistan en el mismo clúster.

Sobre estos dos elementos se han ido creando una serie de aplicaciones que consumen sus servicios y operan siguiendo distintos paradigmas de manera que en función de los casos de uso existentes se utilizará una pila tecnológica ad hoc.

Ilustración 9 Arquitectura *core* de Hadoop



4.2. Data Marts

4.2.1. Introducción a Apache Hive

Apache Hive es el almacén de datos de Hadoop y está formado por los siguientes elementos principales:

- HCatalog, un servicio que permite leer y escribir archivos de datos en una multitud de formatos, tanto de texto plano (CSV, JSON) como binarios serializados.

- Hive Metastore, una base de datos relacional externa (por ejemplo, PostgreSQL) que almacena metadatos relacionales sobre los

datos almacenados en HDFS: nombres de columna, tipos de dato, relaciones, así como el formato del fichero y la relación entre ambos.

- Hive DDL, un *Data Definition Language* que es capaz de mapear los esquemas de Hive Metastore a los atributos de los objetos definidos mediante HCatalog y cuyas expresiones son parte de los metadatos almacenados en Metastore.

La arquitectura de Hive fue diseñada para constituir un almacén de datos *append-only*, pero en sucesivas versiones ha ido incorporando diferentes funcionalidades del estándar SQL demandadas por los usuarios, de manera que en la actualidad cumple total o parcialmente todas las directivas del estándar SQL:2016.

Por lo tanto, permite tanto definir los esquemas como realizar consultas mediante el lenguaje SQL.

4.2.2. *Schema-on-read* y el *Data Lake*

Gracias a la arquitectura flexible de Apache Hive definida en el apartado anterior es posible definir una multitud de esquemas relacionales que apuntan a los mismos ficheros de datos. Así, un mismo conjunto de ficheros, por ejemplo, CSV, puede tener asignados diferentes esquemas que incluyen la totalidad o un subconjunto de las columnas, de forma análoga a como una vista opera en un RDBMS.

Ilustración 10 Esquemas en RDBMS vs Hive[8]

Schema-on-write (RDBMS tradicional)	**Schema-on-read** (Apache Hive)

• **Modelo de datos prescriptivo**.
  • Primero se crea el esquema y se definen los tipos de datos.
  • Transforma los datos para el formato del RDBMS (ETL).
  • Consulta los datos en el formato del RDBMS.
• Las columnas nuevas deben añadirse antes de poder importar nuevos datos.
	• **Modelo de datos descriptivo**.
	  • Primero se copian los datos en su formato nativo.
	  • Crea un esquema adecuado a los tipos de los datos.
	  • Consulta los datos, que siguen en su formato nativo.
	• Los datos nuevos pueden añadirse en cualquier momento y aparecerán retroactivamente cuando un esquema los describa.

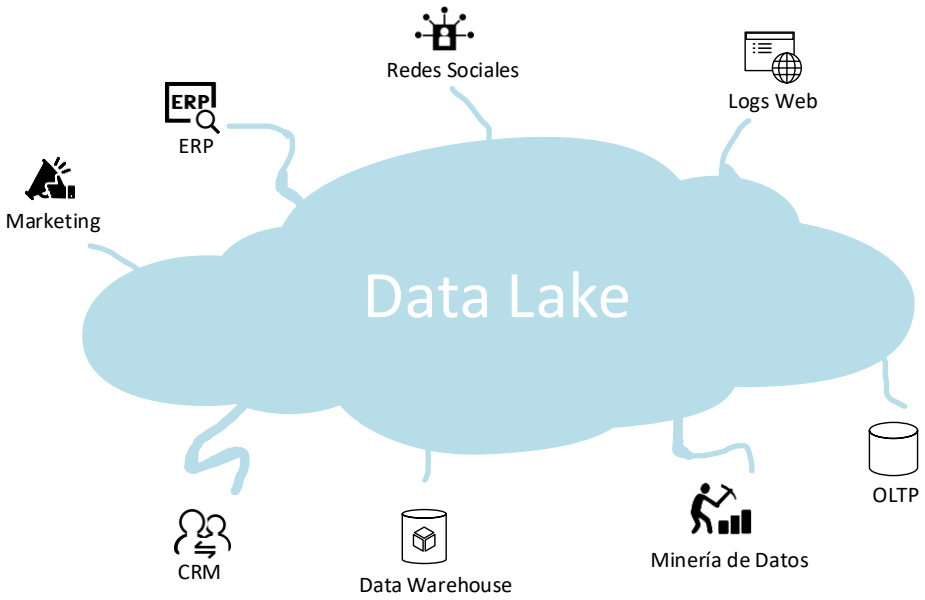
[8] Elaboración propia a partir de "Hive in enterprises" (Mark Grover, 2015). Disponible en https://www.slideshare.net/markgrover/hive-in-enterprises

De acuerdo con Grover[8], los esquemas que se crean antes de los datos son adecuados para escenarios donde ya sabemos lo que nos es desconocido, por lo tanto, tenemos toda la información para tomar una decisión acerca del formato y tipo de lo datos.

Sin embargo, para realizar tareas exploratorias, un esquema que se aplica retroactivamente sobre datos que se cargan en su formato nativo aporta flexibilidad y agilidad, permitiendo que en conjunto el paradigma de *Data Lake* sea idóneo para el almacenamiento de una información en un contexto de evolución constante de la tecnología y donde los sistemas de información desplegados deben ser actualizados a menudo, una operación que no resulta sencilla ni económica.

Finalmente, el Data Lake cuenta con capacidad de almacenamiento virtualmente ilimitada, ya que basta con añadir más nodos al clúster de manera dinámica para obtener mayor capacidad de almacenamiento y de computación, sin necesidad siquiera de reiniciar el clúster ya que basta con reiniciar los servicios de localización de información en los nodos.

Ilustración 11 El paradigma del Data Lake



Por otra parte, la flexibilidad del DDL de Hive junto con HCatalog permite definir formatos de datos optimizados para diferentes usos:

- Formato AVRO, que serializa por filas y por tanto está optimizado para recuperar filas completas.
- Formato ORC, que serializa por columnas y es útil cuando se desea seleccionar o filtrar por un subconjunto de estas.

Además, numerosas aplicaciones del ecosistema de Hadoop también se han integrado[9] con Hive Metastore, incluso aunque no utilicen paradigmas relacionales de acceso a la información y con independencia de si procesan la información por lotes o en tiempo casi real[10].

Como resultado de lo expuesto, Apache Hive permite definir Data Marts con gran flexibilidad y mediante tanto paradigmas de acceso a datos como modelos de proceso de datos (por lotes, tiempo real) distintos.

En particular, la funcionalidad de modificación del formato de los datos en disco ofrece gran potencia a la hora de definir data marts no sólo en base a la información necesaria sino también al modo en el que va a ser consumida, y aunque distintos formatos ofrecen rendimientos diferentes en función del tipo de consulta, como se ha visto.

4.2.3. *Data Marts* en Supermercados Sierra

En base a los desafíos planteados en la presentación del caso, podemos definir diferentes *data marts* asociados a los casos de uso que se describen más adelante en este mismo capítulo:

- Cuadros de mando estratégicos y tácticos con información agregaciones en el tiempo relativamente prolongadas que permitan identificar tendencias.
- Cuadros de mandos operativos con información en tiempo casi real y agregaciones cortas asociadas a los periodos de cumplimiento de objetivos.
- Análisis exploratorio de datos con acceso a formatos que permiten hacer consultas sobre toda la serie histórica de datos con independencia del volumen almacenado.
- Análisis predictivo, ofreciendo formatos específicos para computación paralela en función de los algoritmos a emplear.

4.3. Cubo OLAP

Definiremos un esquema en estrella y su cubo asociado con el propósito de demostrar la capacidad de la plataforma para utilizar las herramientas tradicionales de la inteligencia de negocio.

Se partirá del juego de datos “The complete journey”.

4.3.1. Esquema en estrella

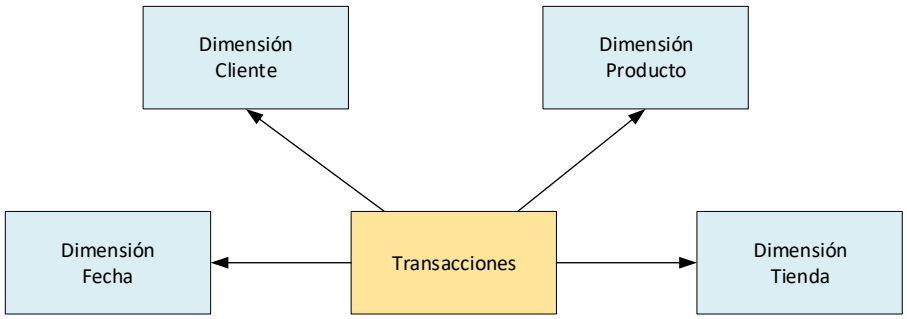
[9] https://cwiki.apache.org/confluence/display/HCATALOG/HCatalog+HBase+Integration+Design
[10] https://henning.kropponline.de/2015/01/24/hive-streaming-with-storm/

La tabla de hechos estará constituida por la tabla de transacciones y se utilizarán las dimensiones cliente, producto, fecha y tienda.

- Transacciones*: transaction_data*
- Cliente: *hh_demographic*
- Producto: *product*
- Fecha: *transaction_data.day*
- Tienda*: transaction_data.store_id*

En el caso de fecha y tienda se generarán tablas dimensionales ad hoc.

Ilustración 12 Esquema en estrella



4.3.2. Dimensiones y medidas del cubo

Se establece una correspondencia entre las dimensiones del esquema en estrella y las dimensiones del cubo OLAP

Con respecto a las medidas de las dimensiones, se tendrán en cuenta una sola jerarquía por cada una ya que los fines son demostrativos de la tecnología. Por lo tanto, resultan los siguientes niveles y miembros:

- Cliente: *income_desc* (bajo, medio, alto, desconocido)
- Producto: *department*, *commodity_desc*, *sub_commodity_desc*
- Fecha: año, trimestre, mes, día.
- Tienda: *store_id*

4.4. Cuadro de mando ejecutivo

Proporcionará una visión general del estado de salud de la cadena de valor en lo que respecta a la red comercial objeto de este trabajo, tanto en tienda física como por Internet.

Ofrecerá agregaciones en distintos periodos de tiempo, así como datos en tiempo real.

4.5. Cuadro de mando de gestión

Tendrá como objetivo fundamental el cumplimiento de niveles de venta definidos por la dirección, tanto para productos concretos como para categorías o marcas de producto que se consideren estratégicos.

La información estará agregada por tienda bajo su responsabilidad, así como por el total para su zona o región correspondiente.

Finalmente, incluirá alertas de previsiones de demanda que le permitan tomar decisiones anticipándose a las situaciones críticas, como por ejemplo actuando sobre la planificación del trabajo de los repartidores.

4.6. Cuadro de mando operativo

El objetivo de este cuadro de mando será triple:

- Informar del nivel de pedidos por Internet para preparar las cestas de la compra cuando el nivel de servicio en tienda lo permita.

- Informar de los niveles de stock de forma anticipada en función de los pedidos por Internet, de manera que el personal de la tienda pueda reservar productos para los repartidores y retirarlos de la venta directa en tienda.

- Informar del grado cumplimiento de objetivo de ventas por día, semana y mes, de terminados productos que se consideran estratégicos de acuerdo con la dirección de la empresa.

4.7. Big Data Analytics

Se presentarán diversos ejemplos de análisis exploratorio de los datos, de manera que se demuestren las capacidades tecnológicas de la plataforma para llevar a cabo consultas SQL arbitrarias sobre cualquier volumen de datos.

Para ello se utilizará la herramienta Hive View de Apache Ambari, la interfaz web de uso y administración de Apache Hadoop. En particular, se trata de obtener información que no estaría accesible sin recurrir a programación a medida, como puede ser una consulta que combina datos de distintos orígenes y formatos:

- Tablas asociadas a las transacciones en tienda.

- Tablas asociadas a los CSV de logs del servidor web apache.

Un ejemplo posible sería la consulta que devuelva el número de visitas en la web y el número de ventas online por hora, de manera que se pueda establecer si hay relación entre ambos.

4.8. Modelo predictivo

Se propone utilizar la información disponible en los juegos de datos de simulación sobre para entrenar y desplegar un sistema de optimización de la ubicación del surtido en tienda en base a un análisis de cesta de la compra mediante algoritmos de implementación de modelos de reglas de asociación.

5. Implementación de la solución

5.1. Diseño

Se utiliza una distribución de Hadoop porque proporciona todos los componentes de *software* necesarios para alcanzar los objetivos del proyecto, a excepción de una solución OLAP que se incorpora por separado al paquete.

Precisamente el hecho que las distribuciones de Hadoop no incorporen herramientas de uso común en el ámbito de *Business Intelligence* ha dado lugar a un nicho de mercado con soluciones propietarias que intentan incorporar la escalabilidad de Hadoop a las técnicas clásicas de la industria, como Datameer[11], que permiten ofrece una experiencia de usuario similar a la de una hoja de cálculo, pero con un *back-end* Big Data.

Por ese motivo, en el diseño de la solución se han tenido en cuenta los siguientes objetivos adicionales:

- Solución basada en *software* libre que ofrezca flexibilidad e independencia de los vendedores, permitiendo encontrar el *mix* de herramientas idóneas para cada caso de uso propuesto por los clientes internos.

- Solución con soporte comercial disponible, ya que la complejidad de las interacciones entre las distintas herramientas desaconseja su uso en entornos de producción sin contar con un SLA adecuado.

5.1.1. Arquitectura

La solución propuesta sigue la arquitectura Lambda ("How to beat the CAP theorem" 2011]), formulada por Nathan Marz, creador de Apache Storm, el motor de procesado de eventos distribuido, tolerante a fallos y en tiempo real que utiliza Twitter y que forma parte del ecosistema de aplicaciones de Apache Hadoop.

El teorema CAP (Brewer 2000) establece que no es posible obtener a la vez las propiedades de consistencia, disponibilidad y tolerancia a particiones en una base de datos y sólo es factible obtener dos de ellas de manera simultánea (). Sin embargo, Marz plantea que se pueden utilizar dos mecanismos diferentes de procesado de datos (uno por lotes y otro en tiempo real) para conseguir en la práctica las tres propiedades conjuntamente.

[11] https://www.datameer.com/

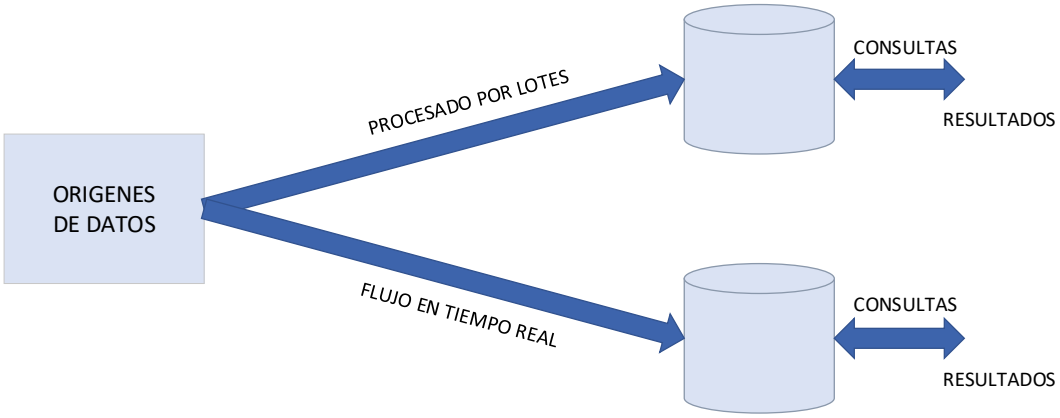


Ilustración 13 Arquitectura Lambda

Así, los datos siguen dos procesos de ingestión paralelos, uno mediante procesos por lotes, con frecuencias relativamente bajas (por ejemplo, diario, semanal, mensual o trimestral) y otro mediante motores de procesado de eventos en tiempo casi real, de manera que estarán siempre disponibles a medida que se generan en los sistemas de origen (Ilustración 13 Arquitectura Lambda).

La tolerancia a particiones se incluye en ambos flujos de datos, ya que el sistema almacenamiento por lotes más popular es Apache Hive que como se ha indicado es un sistema distribuido. Por otra parte, los almacenes de datos en tiempo real también cuentan con múltiples nodos para soportar mayor concurrencia y ancho de banda, teniendo como ejemplo más significativo Apache Hbase.

Para que se trata de un sistema completo de procesado de datos que cumpla con las tres propiedades del teorema CAP, falta por definir una última capa, la de servicio, que abstrae la implementación concreta de la que se obtiene cada dato del consumidor. Un ejemplo de implementación de esta capa es Apache Druid[12], una base de datos para tareas analíticas en tiempo casi real.

5.1.2. Componentes

Distribución de Hadoop

Hortonworks HDP[13] versión 2.6.5.0, completamente basada en código libre[14] y con soporte comercial disponible, de acuerdo con los requisitos del diseño de la implementación.

La distribución incluye los siguientes subcomponentes:

[12] http://druid.io/
[13] https://hortonworks.com/products/data-platforms/hdp/
[14] https://github.com/hortonworks

- HDFS, el sistema de ficheros distribuido y tolerante a fallos de Apache.

- YARN, capa de abstracción de los recursos de computación del clúster que opera como un servicio que gestiona las tareas solicitadas por las distintas aplicaciones que los requieren.

- MapReduce2, motor de computación basado en el paradigma homónimo creado por investigadores de Google en 2004 (Dean y Ghemawat 2008) y que ofrece paralelización de tareas como servicio.

- Tez, *framework* que se sitúa entre las aplicaciones que necesitan procesar datos de manera distribuida y el motor de computación MapReduce, que elabora un plan de ejecución siguiendo un modelo DAG (grafo acíclico dirigido, por sus siglas en inglés) complejo y que como resultado permite obtener rendimientos mayores que MapReduce por sí solo.

- Hive, almacén de datos utilizado tanto para la realización de consultas *ad hoc* como para el análisis de conjuntos de datos virtualmente ilimitados.

- HBase, base de datos distribuida, versionada y no relacional que permite realizar consultas con baja latencia sobre tablas en el orden de miles de millones de filas y millones de columnas. Su diseño sigue el de BigTable (Chang et al. 2008), la base de datos que da soporte al buscador *web* de Google.

- Sqoop, herramienta que permite la transferencia de datos masiva entre Hadoop y bases de datos relacionales, en ambos sentidos y con excelente rendimiento debido a que utiliza los recursos de computación distribuida de la plataforma.

- ZooKeeper, es un servicio centralizado de configuraciones que permite la orquestación de las distintas herramientas que forman parte del ecosistema Hadoop. Pese a que opera como un servicio con un único punto de acceso, está implementado de manera distribuida por lo que ofrece tolerancia a fallos.

- Kafka, sistema de mensajes distribuido que implementa dos de los principales paradigmas de manera simultánea: cola de mensajes y publicación-suscripción. Creado por Linkedin, ofrece un excelente rendimiento («Benchmarking Apache Kafka» 2014), del orden de millones de escrituras por segundo con tan sólo tres nodos de pequeño tamaño.

- Oozie, sistema para la coordinación de los distintos flujos de ejecución de las tareas de Hadoop. Permite programar la ejecución (sustituyendo a las tareas cron de Unix) y monitorizar el estado aplicando reglas lógicas para la recuperación desde los distintos estados de fallo.

- Spark2, motor de procesado de propósito general que ofrece excelente rendimiento y escalabilidad.

Zeppelin Notebook

Se trata de una herramienta de análisis de datos basada en la *web* que permite la creación de documentos interactivos que incorporan consultas, tratamiento y visualización de datos mediante una variedad de lenguajes, incluyendo SQL.

Su principal ventaja es la integración con Hadoop, que permite alterar el flujo de trabajo habitual en los equipos de análisis de Big Data y que tradicionalmente consiste en prototipar con un conjunto de datos reducido y un lenguaje dinámico (como puede ser R) y posteriormente desarrollar una versión más potente con otro lenguaje más rápido y robusto (como puede ser Java).

Así, con Zeppelin es posible combinar distintos lenguajes tanto de consulta como de programación para lograr los objetivos de la tarea propuesta, utilizando el conjunto completo de datos y procesándolo de manera distribuida.

En particular, permite:

- Ingestión de datos desde diversas fuentes, estructuradas o no, relacionales o no.

- Exploración de datos, gracias a que soporta más de 20 lenguajes de programación distintos.

- Análisis de datos a escala Hadoop, ya que cuenta con conectores para los motores de computación distribuida de Hadoop.

- Visualización y colaboración, ya que incorpora una serie de componentes para la realización de gráficas interactivas, así como la posibilidad de compartir la URL de un cuaderno con colegas o clientes.

Spark-R

Se trata de una biblioteca para el lenguaje de programación R que permite utilizar datos almacenados en Hadoop directamente desde R y que se incluye con la distribución estándar de Apache Spark.

Apache Druid

Se trata de un almacén de datos distribuido y organizado por columnas que ofrece tiempos de respuesta por debajo de un segundo en consultas complejas y soporta alta concurrencia gracias a la incorporación del paradigma de índice inverso, presente habitualmente en los motores de búsqueda de textos.

Está integrado con el bróker de mensajes de Kafka, por lo que su aplicación directa es el análisis de flujos de eventos en tiempo real, quneu su relevancia para los objetivos del proyecto se fundamenta en su capacidad para sustituir la necesidad de pre-computar cubos en aplicaciones OLAP.

Al generar índices inversos asociados a las relaciones entre las distintas dimensiones en el momento en que se insertan los datos, Druid permite ejecutar consultas analíticas arbitrarias con rendimientos superiores a las BDD relacionales.

Su principal desventaja es la carencia de operaciones eficaces de unión de conjuntos de datos, debido a las limitaciones en el propio concepto en que se basa su diseño.

Apache Superset

Aplicación de Business Intelligence desarrollada por Airbnb[15], basada en la *web* y que ofrece características similares a herramientas propietarias que se han convertido en estándares de facto en el sector, como pueden ser Tableau o Qlikview.

Desarrollada en lenguaje Python, soporta una gran variedad de orígenes de datos con distintos paradigmas:

MySQL	Sybase	SQLite
Postgres	IBM DB2	Greenplum
Vertica	Exasol	Firebird
Oracle	MonetDB	MariaDB
Microsoft SQL Server	Snowflake	Redshift
Clickhouse	Apache Kylin	**Apache Druid**

Su integración con Apache Druid es particularmente estrecha y en el caso de análisis OLAP, Superset ocuparía el lugar de los exploradores de cubos, pero de nuevo a escala Internet y sin procesos por lotes previos.

[15] https://airbnb.io/projects/superset/

Con respecto a sus características, ofrece las siguientes funcionalidades:

- Creación de cuadros de mando interactivos.
- Batería de componentes de visualización.
- Granularidad completa, permitiendo llegar desde una visualización hasta los datos individuales que la alimentan, pasando por distintos niveles de agregación en función de las dimensiones.
- Editor SQL que sirve además como punto de partida para la generación de visualizaciones.
- Control de acceso basado en reglas con alta granularidad, así como integración con diversos mecanismos de autenticación: OpenID, LDAP, OAuth.
- Capa semántica que abstrae las fuentes de datos que son presentadas al usuario de manera que se pueden generar diferentes vistas sin necesidad de acceder a los sistemas de origen.
- Alto rendimiento, en particular en aplicaciones integradas con Apache Druid.

Apache Kylin

Motor de análisis multidimensional (OLAP) distribuido para conjuntos de datos a escala Internet, desarrollado originalmente por eBay.

Se utiliza para ilustrar la capacidad de generación de cubos tradicionales en la plataforma Hadoop, así como para comparar y evaluar sus capacidades frente a soluciones emergentes como Apache Druid.

5.1.3. Procesos

MySQL – Apache Hive

Se utiliza Apache Sqoop para importar una base de datos relacional completa, distinguiendo entre dos tipos de carga:

- Carga completa, para relaciones susceptibles de ser actualizadas, como por ejemplo la tabla de productos o clientes.
- Carga incremental, para relaciones *append-only*, como pueden ser las transacciones en tienda.

El juego de datos escogido es DunnHumby “The Complete Journey” dado que incluye un esquema completo, así como sus relaciones y asociaciones, de manera que se puedan ilustrar los dos tipos de carga mencionados.

Sqoop opera en dos pasos diferenciados, como se observa en Ilustración 14 Proceso de importación de MySQL en Hive:

1. Obtención de metadatos de la base de datos de origen, fundamentalmente estadísticas de la clave primaria.

2. División de los datos en distintos grupos en función su clave, de manera que se generen tantas tareas de importación como grupos, al objeto de paralelizar y acelerar la carga.

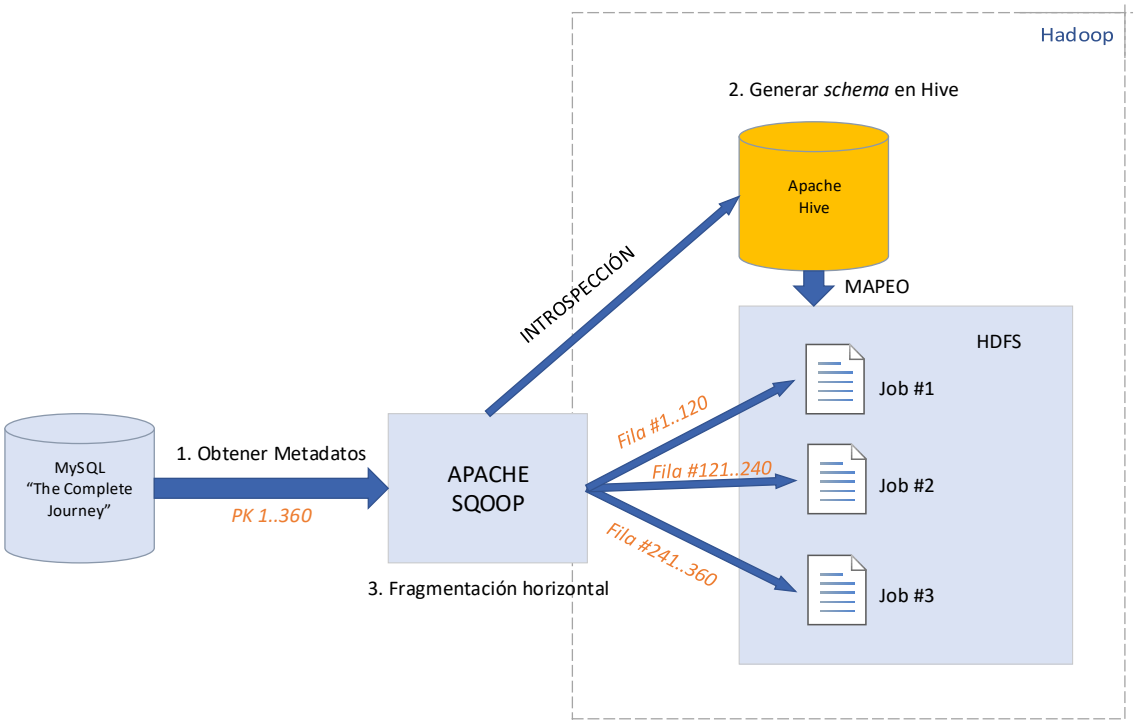


Ilustración 14 Proceso de importación de MySQL en Hive

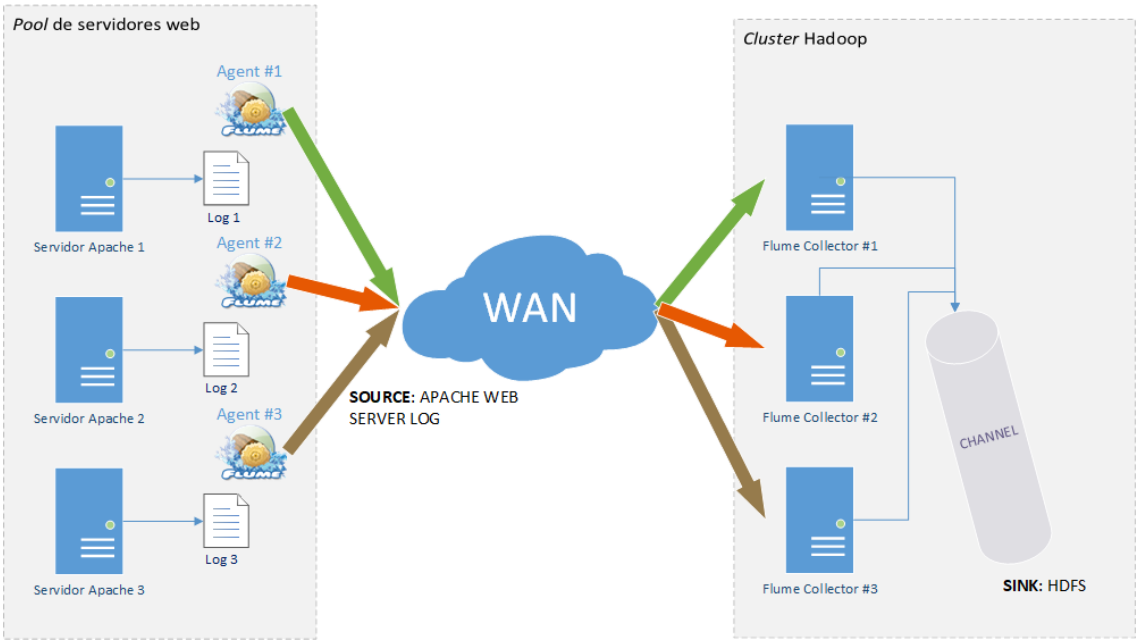
Servidor web – Apache Hive

Apache Flume es un servicio diseñado para obtener datos de distintas fuentes "*sources*" mediante un agente desplegado en el sistema dónde se generan, enviarlos a un almacenamiento intermedio "*channel*" y posteriormente transformarlos y almacenarlos en uno de entre varios destinos posibles "*sinks*".

En nuestro caso la fuente seleccionada serán ficheros de texto plano, en particular los logs de acceso a la página web de Supermercados Sierra, como canal se podrá utilizar tanto HDFS como Kafka, en función del volumen de tráfico que se genere, y como destino final se utilizará HDFS o Druid, en función de si se necesitan acceder a los datos en tiempo casi real o no.

En Ilustración 15 Recogida de logs de Apache con Flume se presenta un diagrama de la arquitectura a desarrollar.

Ilustración 15 Recogida de logs de Apache con Flume



Pese a que existen muchas soluciones posibles para el procesamiento masivo de datos, la característica principal de Flume es la distribución de la propia ingestión, de manera que no existe un único punto de fallo y los agentes actúan como distribuidores de la carga de trabajo entre todos los recolectores disponibles.

TPV – Kafka – Druid

Mediante *scripts* de Python que simulan ser un terminal de punto de venta de un supermercado o de la web, se envían transacciones a Kafka.

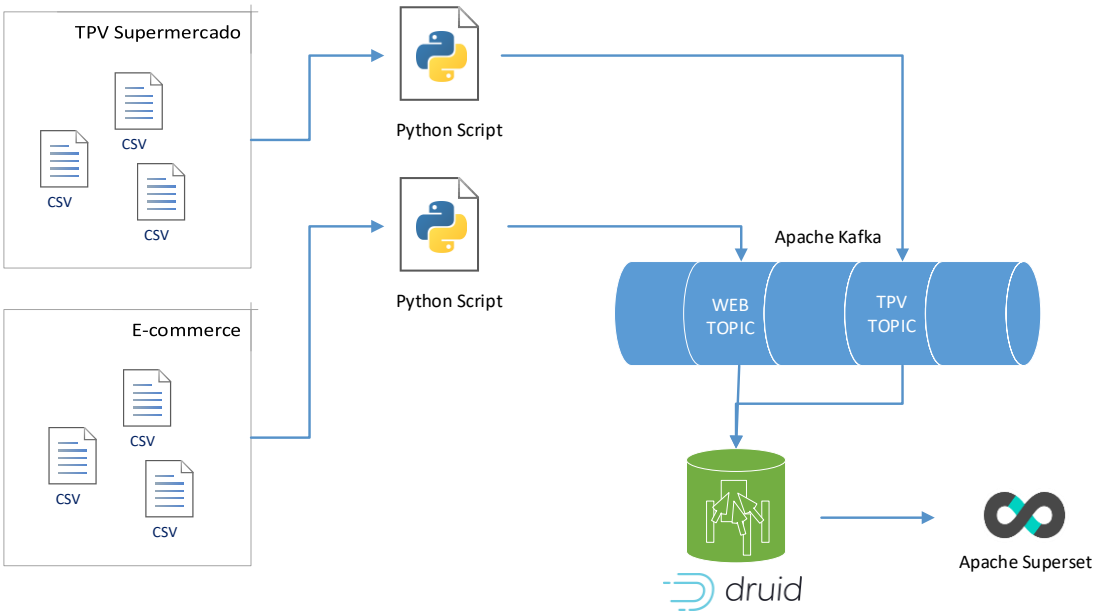


Ilustración 16 Flujo de transacciones de venta

Los eventos de cada tópico serán consumidos por Apache Druid y almacenados en diferentes relaciones con el objetivo de obtener disponibilidad en tiempo casi real de la información ingerida, tanto para consultas *ad hoc* como para alimentar cuadros de mando elaborados mediante Superset, la herramienta de visualización de datos interactiva de Apache.

En los tópicos se especifica su tamaño mediante el periodo de retención de estos, estando situado por defecto en 4 días. Así, los mensajes que superan dicho umbral son eliminados automáticamente, pero se mantiene un periodo de gracia que permite recuperar estados de error en los datos de origen que se hayan detecten a tiempo.

5.2. Entorno de *hardware* y *software* de base

Se han utilizado cuatro servidores dedicados alojados en el proveedor OVH con la siguiente configuración por equipo:

- Intel(R) Xeon(R) CPU E5-1650 0 @ 3.20GHz
  Cores: 12, Cache: 12288KB
- RAM: 4x 16384MB (64GB en total)
- Disks: 2 x 2000 GB (4TB en total)

El sistema operativo es Ubuntu Server 14.04 LTS y la conectividad de red entre los equipos cuenta con 1 Gbps de ancho de banda. El acceso remoto se ha realizado mediante SSH (Secure-Shell).

5.2.1. Instalación de Hortonworks HDP

La instalación se ha llevado a cabo mediante la aplicación Ambari de Apache, una herramienta especializada en la instalación, configuración, monitoreo y mantenimiento de clústeres Hadoop.

Instalación de Apache Ambari

Los pasos que se han seguido son:

1. Instalación de *curl*, un cliente de línea de comandos para la biblioteca *libcurl* que permite realizar peticiones HTTP desde *scripts*.

2. Creación de pares de claves RSA para OpenSSH en el usuario administrador de todos los nodos, de manera que Ambari pueda tomar el control del sistema durante la instalación.

3. Aumentar el número máximo de ficheros que se le permite abrir a un mismo usuario (directiva *ulimit*).

4. Configuración de archivo *hosts* de cada nodo, estableciendo un Fully Qualified Domain Name (FQDN) que permita a los miembros del cluster localizarse unos a otros sin necesidad de depender de un servidor DNS.

5. Configuración de acceso sin contraseña mediante SSH entre todos los nodos del clúster, mediante el fichero de clave privada RSA.

6. Configuración de un nuevo repositorio de paquetes en el sistema de gestión de paquetes del sistema operativo, en nuestro caso Aptitude, disponible en las distribuciones basadas en Debian.

7. Instalación del paquete *ambari-server*, que incluye la descarga del *runtime* Java de Oracle.

8. Inicio del demonio ambari-server y comprobación de acceso a la herramienta mediante el navegador web, en el puerto 8080 de la máquina escogida como instalador.

Instalación de HDP

Una vez lanzado Ambari y tras autenticarse con el usuario y contraseña por defecto, el asistente de instalación solicita los nombres de host de los equipos que van a formar parte del cluster, en nuestro caso se ha especificado mediante una expresión regular: *hadoop[1-4].local*

A continuación, el sistema solicita la clave privada RSA generada en el apartado anterior y comprobará que cuenta con acceso administrador a todos los equipos en los que se instalará HDP, como paso previo a la configuración de repositorios de paquete adicionales en todos los nodos y la instalación del agente Ambari, que le permitirá monitorizar el estado del sistema en todo momento.

El siguiente paso es la selección de las aplicaciones del ecosistema Hadoop que se desean instalar, seguido de la asignación de los diferentes servidores y clientes de cada solución a los distintos nodos, en función de sus capacidades de almacenamiento y de memoria.

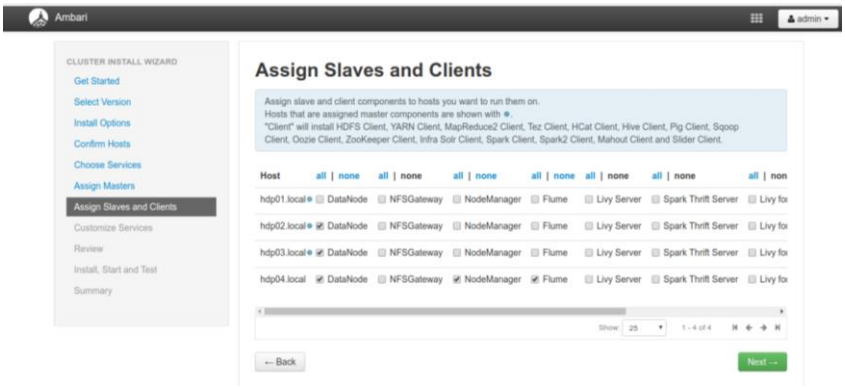


Ilustración 17 Asignación de servicios en nodos

Consideraciones clave de este paso:

- **Servicio HDFS**
    - Se deben escoger al menos tres nodos que ejecuten el servicio de HDFS como almacén de datos "*datanode*", de manera que se pueda garantizar el nivel mínimo de replicación de datos que proporciona Hadoop y que por defecto es 3.
    - Se deben escoger dos nodos de gestión del servicio o "*namenodes*" que mantienen la base de datos con los metadatos que relacionan los ficheros en HDFS con los bloques distribuidos entre los distintos *datanode*. En caso de no escoger al menos dos, no existe garantía de alta disponibilidad y la pérdida del único namenode supondría la caída del clúster completo.

Finalmente, se deben especificar diferentes parámetros de configuración para aquellas decisiones que Ambari no puede tomar de manera automática.

En su mayor parte, se trata de los parámetros de conexión para las distintas bases de datos que muchas de las herramientas del ecosistema utilizan para almacenar sus metadatos:

- Ambari por defecto instala una base de datos PostgreSQL para almacenar su propia configuración y estado.
- Apache Hive incluye la base de datos Apache Derby para almacenar los metadatos de las distintas tablas, así como la localización (nodo y bloque) de los propios datos en el clúster,

aunque no se recomienda su uso en entornos de producción, por lo que en nuestro caso hemos creado y configurado una base de datos MySQL que ha requerido de la instalación adicional del conector de Java JDBC para MySQL y la configuración de su ubicación en el servicio *ambari-server*.

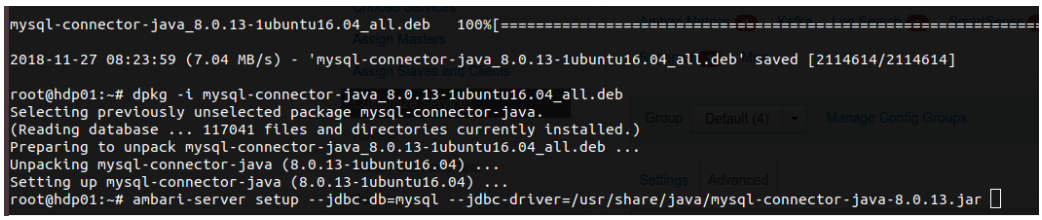


Ilustración 18 Configuración del conector MySQL en Ambari

Consideraciones clave de este paso:

- **Permisos de las bases de datos**
    - Cuando se crean las distintas bases de datos indicadas en la configuración, es importante crear además un usuario específico con privilegios completos en ellas y asegurar que estos son válidos desde cualquier nodo del clúster.
    - La mayoría de SGBD relacionales autorizan por defecto a usuarios de la máquina local.

Una vez completada la configuración y tras lanzar el proceso de despliegue, obtenemos el cuadro de mando de Ambari:

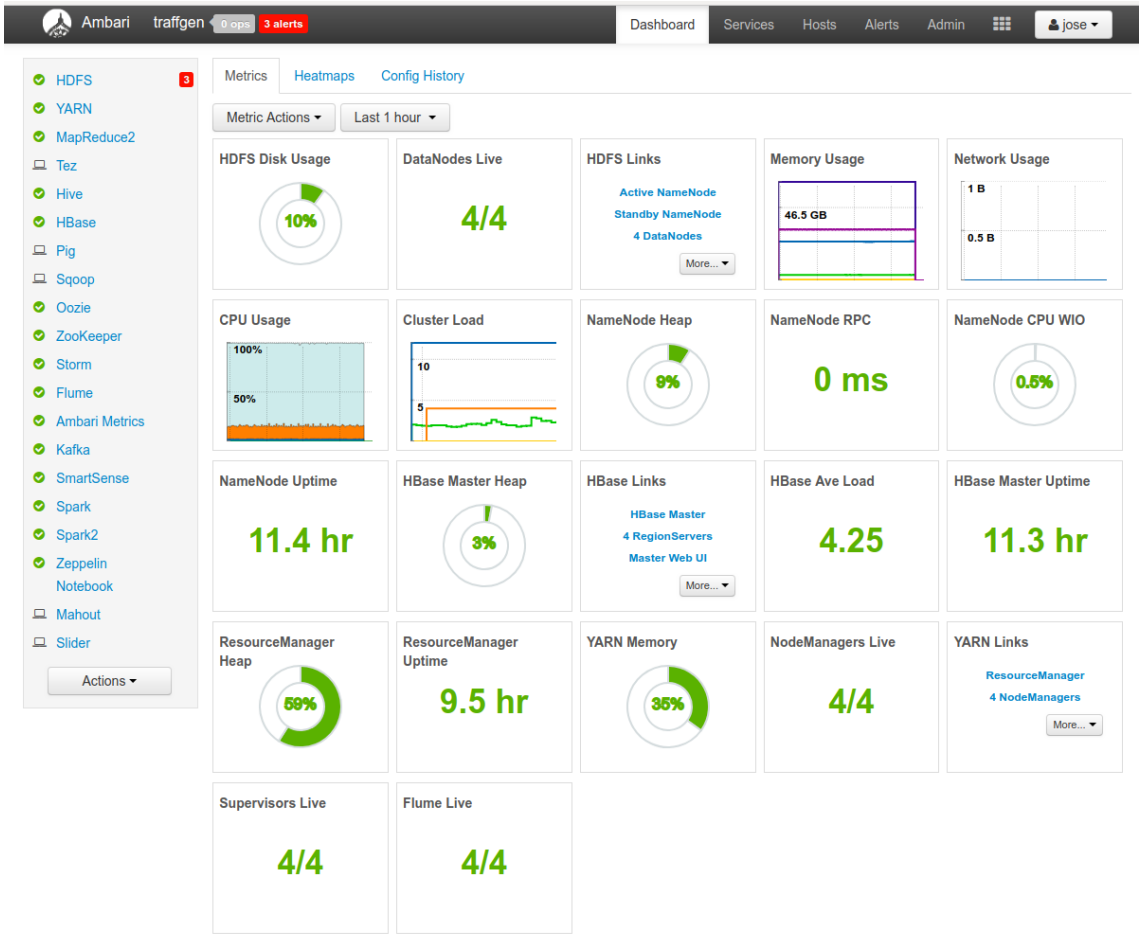
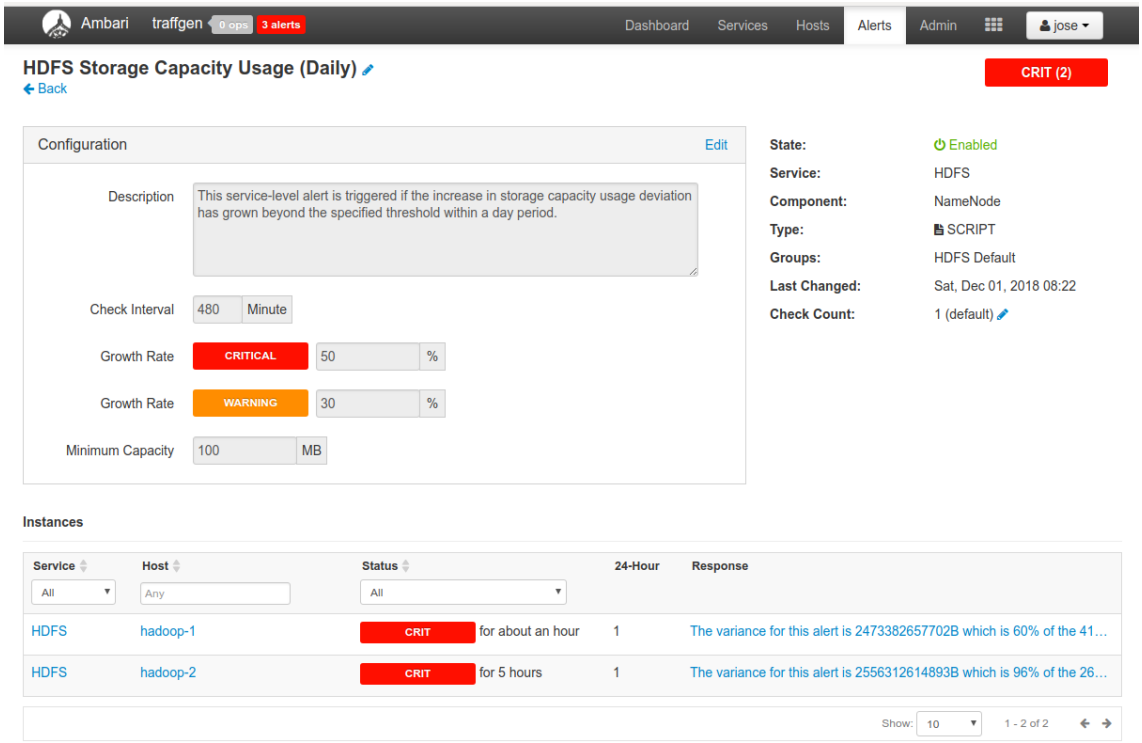


Ilustración 19 Instalación de HDP completada

Las tres alertas que aparecen en el servicio de HDFS se deben a la monitorización de actividad inusual, basada en cambios bruscos en la varianza de parámetros clave como el ancho de banda consumido o el espacio de almacenamiento ocupado. Como resultado, en clústeres recién instalados es sencillo lanzar estas alertas al realizar cargas de datos de prueba.

En Ilustración 20 Alertas de varianza fuera de límites en HDFS se puede observar el caso.

Ilustración 20 Alertas de varianza fuera de límites en HDFS



5.2.2. Configuración de HDP

Las configuraciones por defecto de Ambari, Hive, Kafka, Flume y Zeppelin son adecuadas para los fines ilustrativos de la tecnología del proyecto y no ha sido necesario modificarlas.

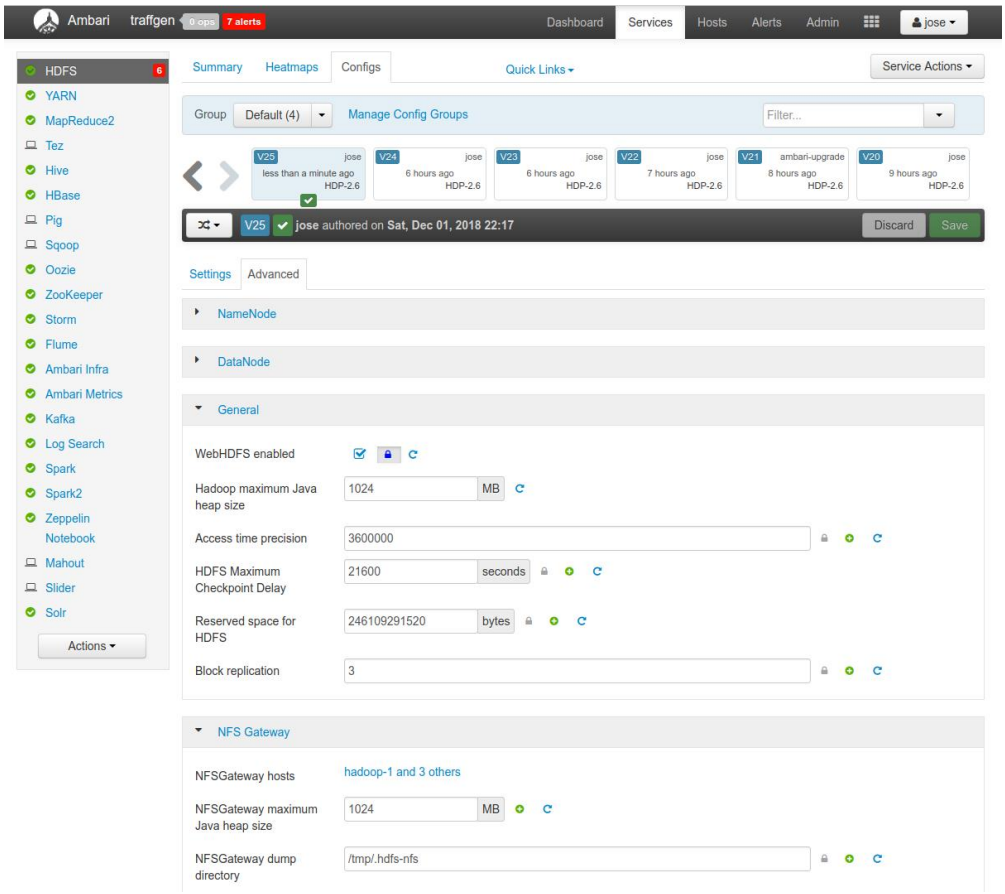
NFS Gateway

El acceso al sistema de ficheros de Hadoop (HDFS) se realiza bien programáticamente mediante la API para Java proporcionada, bien mediante una herramienta específica de línea de comandos que permite realizar las tareas más habituales: creación de ficheros, copia hacia y desde HDFS desde el sistema de ficheros local o borrado.

Sin embargo, HDFS incorpora compatibilidad con el sistema de ficheros en red de Unix (NFS) de manera que rutas de HDFS se pueden montar de manera local como si fuesen una ruta de red cualquier y para facilitar el uso de cualquier herramienta que no esté preparada para usar la API de HDFS directamente.

La pasarela NFS viene activada por defecto en HDP, aunque es necesario modificar un parámetro de configuración para asegurar su compatibilidad con las versiones más recientes de los clientes NFS, se trata de “Access time precisión” ya que las latencias de red propias de HDFS no son las esperadas por NFS.

Ilustración 21 Configuración de la pasarela NFS



Apache Kylin

Kylin no forma parte de la distribución de Hadoop de Hortonworks, por lo que ha sido necesario descargarlo desde la página web del proyecto y proceder a su instalación manual.

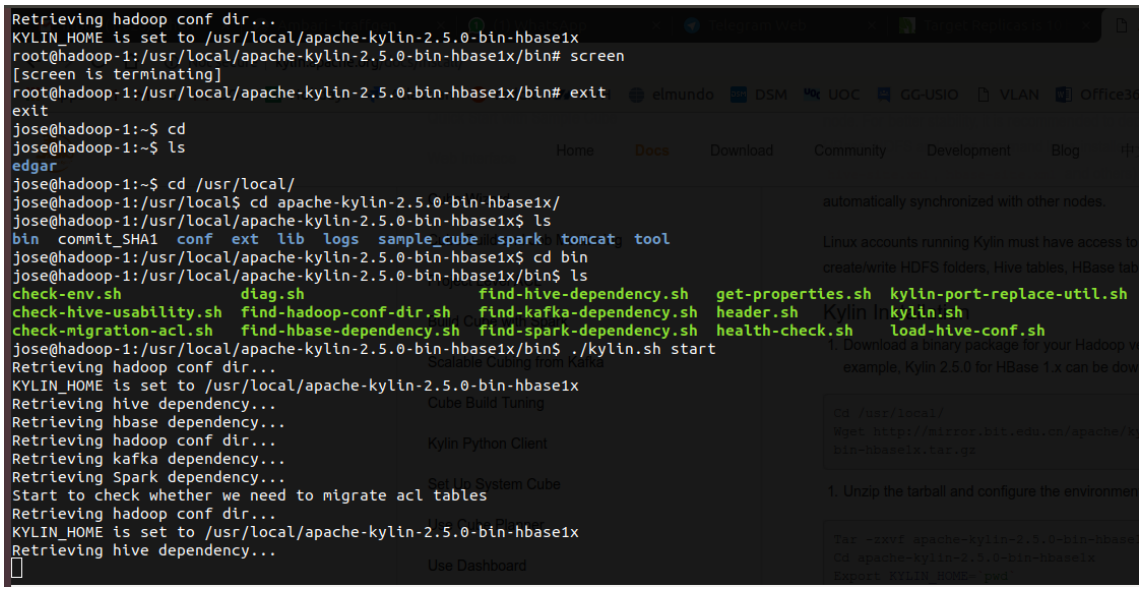
En primer lugar, ha sido necesario instalar Apache Hbase ya que Kylin lo utiliza como almacén de datos intermedio para la creación de sus cubos.

Consideraciones clave de este paso:

- **HBase** sigue un modelo de tolerancia a fallos en computación distribuida que impide que el clúster se inicie cuando no se detectan nodos operativos. Esto implica que cuando se desea lanzar el servicio debe hacerse nodo por nodo, ya que de hacerlo de manera simultánea entre en un estado de error cíclico.

No obstante, Kylin es compatible con HDP y basta con descomprimir el paquete de instalación y ejecutarlo para que el servicio esté accesible en el puerto 7070 mediante el navegador web:

Ilustración 22 Detalle de instalación de Kylin



Una vez lanzado y tras autenticarse con los valores por defecto, es posible cargar un juego de datos de prueba proporcionado con el paquete y generar un cubo:

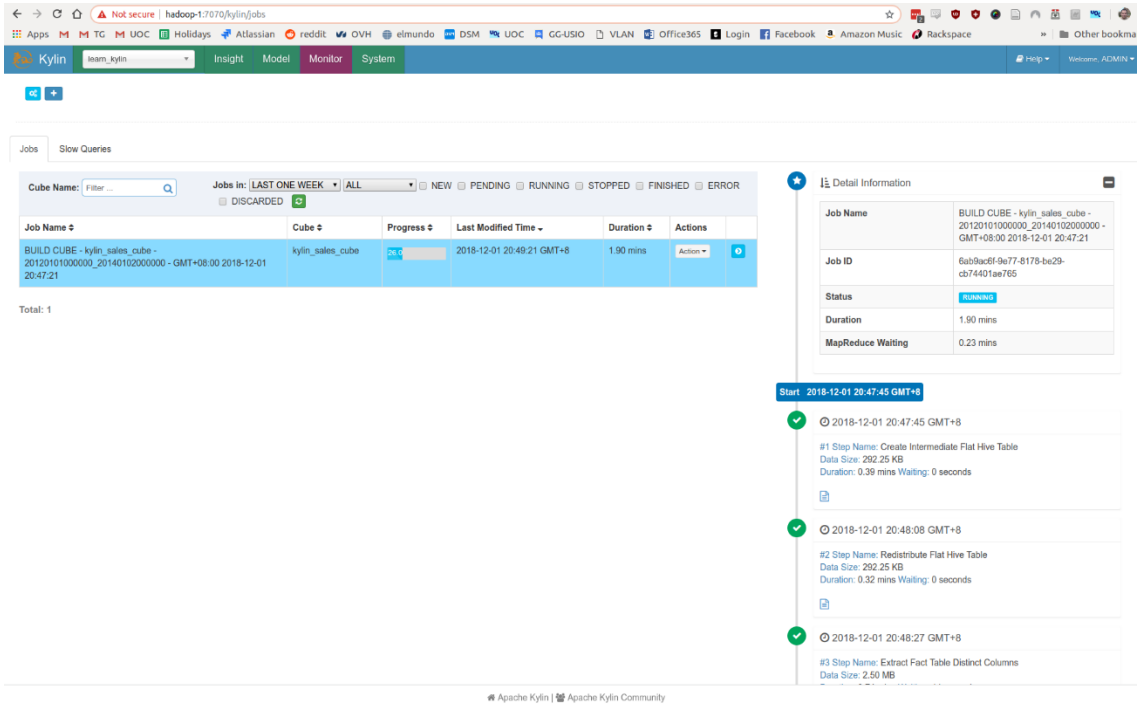


Ilustración 23 Generación de cubo OLAP en progreso

5.3. ETL

5.3.1. Modelo de datos

Se han creado los siguientes esquemas en Hive:

DunnHumby – **Let's get real** – TRANSACTION_DATA

```
CREATE EXTERNAL TABLE `dh_lgr_transactions`(
  `shop_week` decimal(10,0),
  `shop_date` decimal(10,0),
  `shop_weekday` decimal(10,0),
  `shop_hour` decimal(10,0),
  `quantity` decimal(10,0),
  `spend` decimal(10,0),
  `prod_code` string,
  `prod_code_10` string,
  `prod_code_20` string,
  `prod_code_30` string,
  `prod_code_40` string,
  `cust_code` string,
  `cust_price_sensitivity` string,
  `cust_lifestage` string,
  `basket_id` decimal(10,0),
  `basket_size` string,
  `basket_price_sensitivity` string,
  `basket_type` string,
  `basket_dominant_mission` string,
  `store_code` string,
  `store_format` string,
  `store_region` string)
COMMENT 'Dunnhumby LetsGetReal Transactions'
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  'hdfs://traffgen/user/jose/dunnhumby/real'
```

Se ha indicado que se trata de una tabla externa, por lo que los datos sólo residen en los ficheros CSV y no se copian a la base de datos aunque son plenamente accesibles mediante el paradigma de *schema-on-read*.

Además, se ha especificado el delimitador del fichero plano CSV así como la ruta en HDFS del directorio que contiene los ficheros CSV.

También se indica el formato del fichero en HDFS (*TextInputFormat*) así como el formato de datos con el que deben ser interpretados por Apache Hive (*HiveIgnoreKeyTextOutputFormat*).

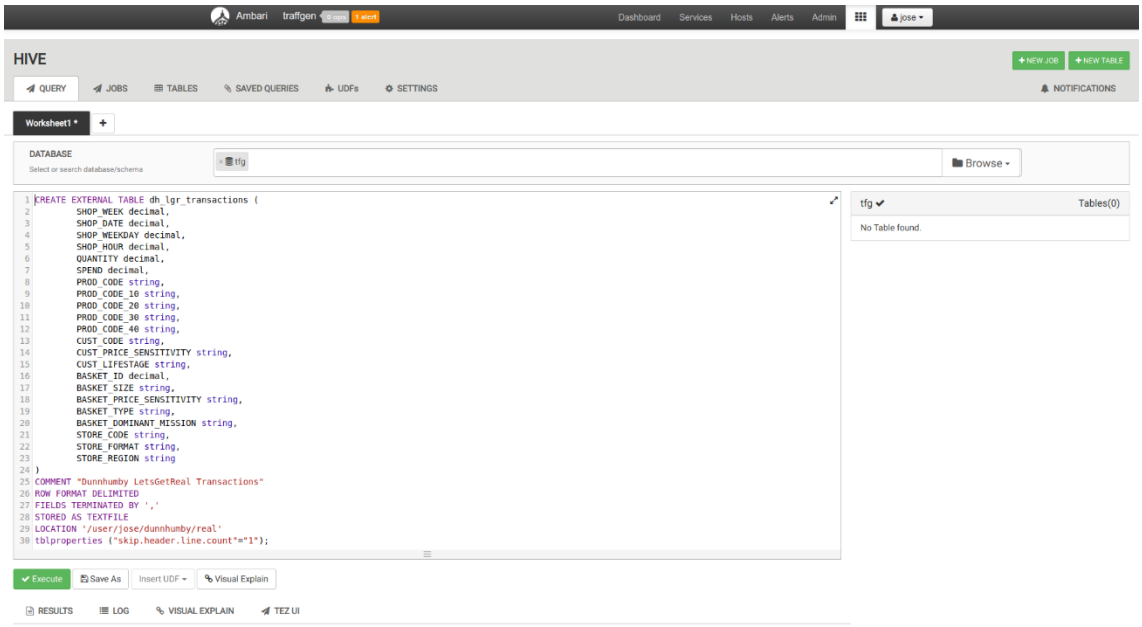
EDGAR – **Log de Apache**

```
CREATE TABLE `edgar_apache_log`(
  `ip` string,
  `dt` string,
  `time` string,
  `zone` decimal(10,0),
  `cik` decimal(10,0),
  `accession` string,
  `extention` string,
  `code` decimal(10,0),
  `size` decimal(10,0),
  `idx` decimal(10,0),
  `norefer` decimal(10,0),
  `noagent` decimal(10,0),
  `find` decimal(10,0),
  `crawler` decimal(10,0),
  `browser` boolean)
COMMENT 'Edgar Apache Log'
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  'hdfs://traffgen/user/jose/edgar'
```

Para la base de datos relacional *Dunnhumby – The Complete Journey* el esquema se importa de los datos de origen por lo que no es necesaria su especificación previa.

La figura Ilustración 24 Creación de la tabla TRANSACTION_DATA muestra el uso de la herramienta de administración de Apache Hive en Apache Ambari para crear tablas SQL externas asociadas a un directorio con ficheros de tipo plano (CSV, en este caso).

Ilustración 24 Creación de la tabla TRANSACTION_DATA



Una vez creada los datos están disponibles inmediatamente ya que están asociados a los ficheros presentes en la ruta indicada, por lo que no se habla de ETL (*Extract-Transform-Load*) sino de ELT (*Extract-Load-Transform*).

Para ilustrar este hecho, resulta útil generar las estadísticas de la tabla dónde además se parecía como las consultas SQL se transforman en tareas distribuidas MapReduce que son gestionadas por el motor optimizado de Apache Tez, como se observa en Ilustración 25 Tareas asociadas al comando ANALYZE TABLE ... COMPUTE STATISTICS:

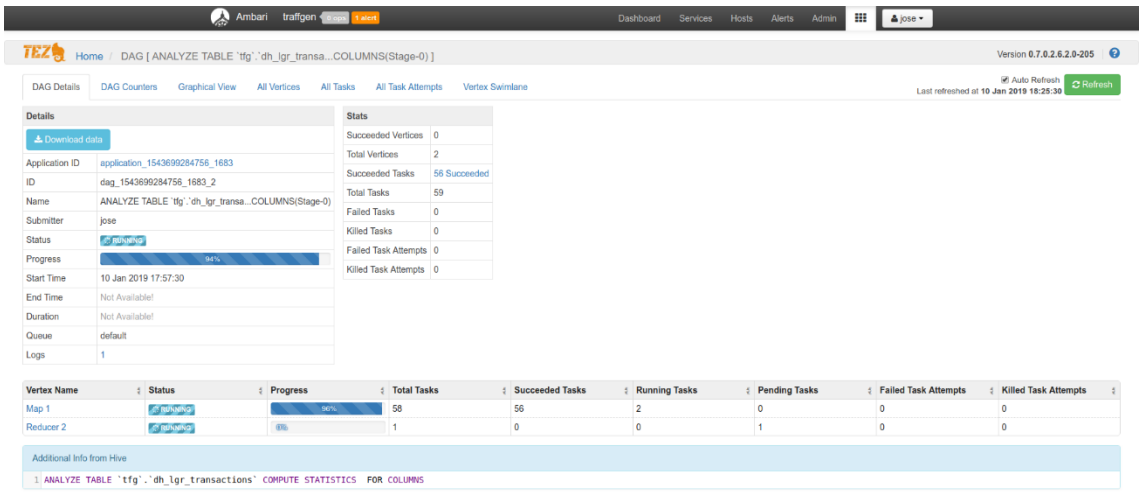


Ilustración 25 Tareas asociadas al comando ANALYZE TABLE

En particular, se aprecia que el análisis de la tabla ha dado lugar a 58 tareas *Map* que se distribuyen entre los nodos disponibles en el clúster, de las cuales 2 ya han sido completadas y por tanto quedan 56 restantes.

En Ilustración 26 Resultado del análisis de la tabla se observan las estadísticas ya computadas, indicando más de 307 millones de filas (tiempo de procesamiento aproximado: 20 minutos).

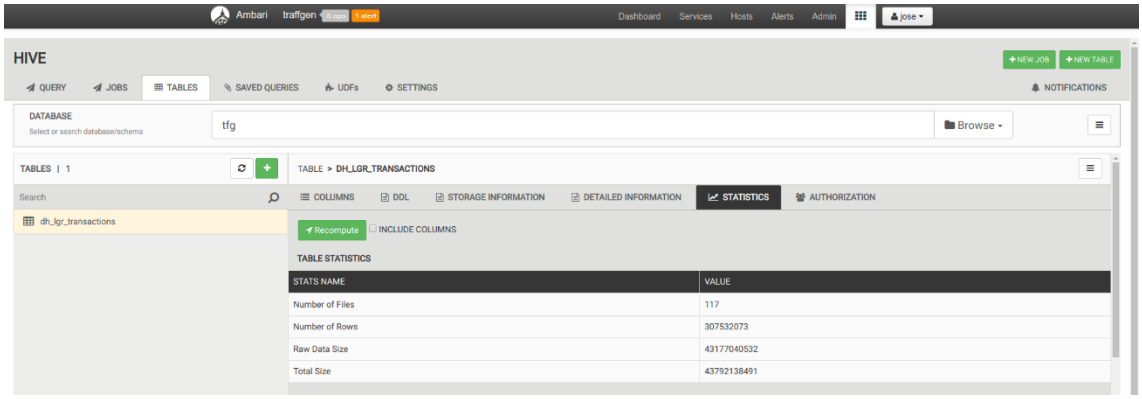


Ilustración 26 Resultado del análisis de la tabla

5.3.2. Carga de datos por lotes

Se utiliza la herramienta Apache Sqoop, invocada mediante línea de comandos, para especificar la base de datos de origen, la URI de conexión y los parámetros necesarios para la carga. La XXX muestra el comando utilizado para la carga del juego de datos *Dunnhumby – The Complete Journey* desde MySQL en Apache Hive.

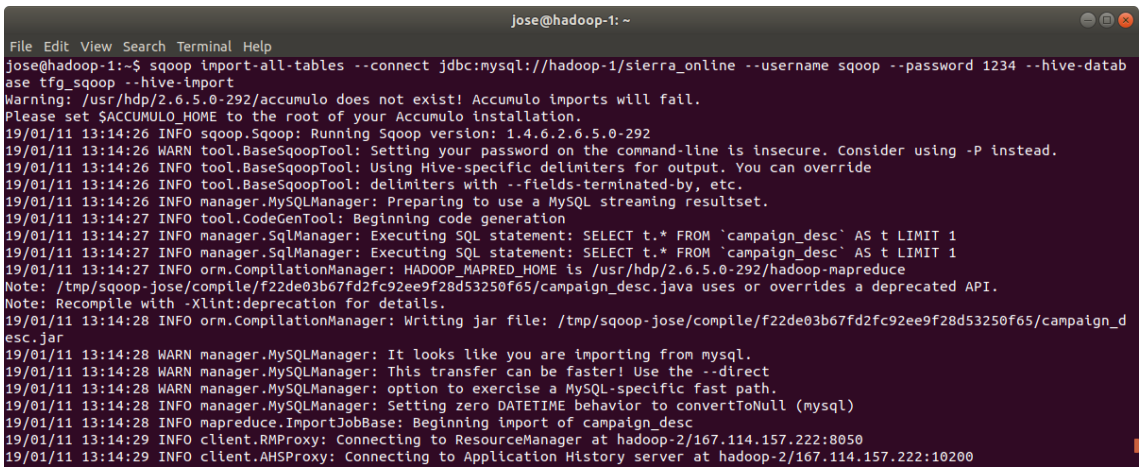


Ilustración 27 Invocación de Apache Sqoop en CLI

Se ha indicado el argumento *hive-import* para indicarle a Sqoop que no cargue los datos en HDFS de forma previa (comportamiento por defecto) para posteriormente asociarlos a una tabla externa, sino que los cargue directamente en una tabla propia de Hive.

Al utilizar tablas internas, Apache Hive gestiona directamente la ubicación de la información por lo que aunque también está persistida en HDFS, se pueden llevar a cabo optimizaciones a la hora de almacenarlo que mejoran el rendimiento de las consultas en cierta medida

Se debe tener en cuenta que Apache Sqoop, como cualquier otro procesamiento en Hadoop, transformará los comandos invocados en una serie de tareas distribuidas del paradigma MapReduce, por lo que las cargas se realizan en paralelo y su rendimiento es proporcional al número de nodos del clúster empleado.

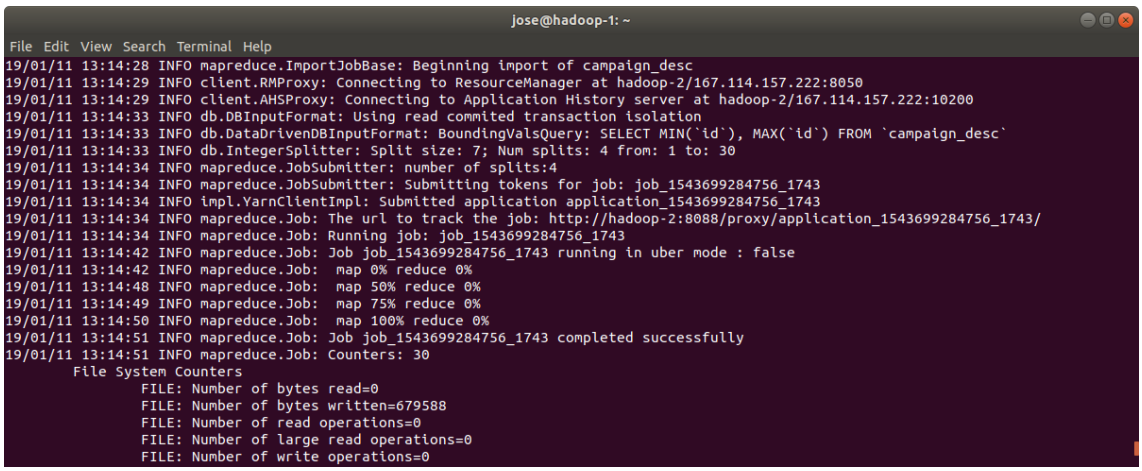


Ilustración 28 Distribución de la carga en Sqoop

En Ilustración 28 Distribución de la carga en Sqoop se aprecia que al importar la tabla *campaign_desc*, que cuenta con 30 filas, se han realizado 4 particiones (*splits*) que se corresponden con otros tantos nodos del clúster, de manera que cada nodo emita una cuarta parte de las filas durante la fase *Map* del algoritmo *MapReduce*.

Consideraciones clave de este paso:

- Las relaciones que se deseen importar mediante Sqoop **siempre deben tener una restricción de tipo clave primari**a ya que de otra forma Sqoop no es capaz de distribuir la carga entre varios nodos.

- Alternativamente, es posible indicarle a Sqoop qué **columna** se desea utilizar para particionar la carga en tareas, sin que sea necesario que se haya declarado clave primaria, aunque sí debe tener **valores únicos**.

5.3.3. Carga de datos en tiempo casi real

Tienda

Sitio web

5.4. Consultas exploratorias SQL

Apache Hive implementa un servicio JDBC[16], la API de conectividad a base de datos de Java por lo que cualquier aplicación compatible puede realizar consultas directamente.

Para los fines ilustrativos del presente trabajo, se ha empleado la aplicación de administración de la Hive que incorpora Apache Ambari, como se puede ver en Ilustración 29 Consulta SQL sobre Apache Hive en Ambari:

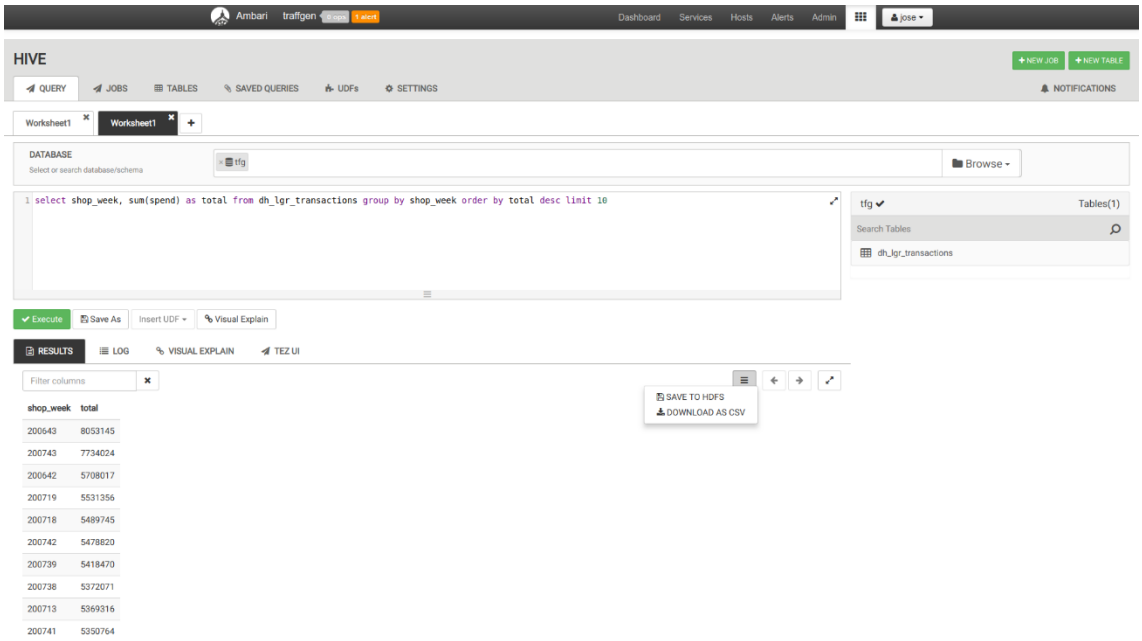


Ilustración 29 Consulta SQL sobre Apache Hive en Ambari

La consulta mostrada en la imagen ha tenido un tiempo de ejecución de 351 segundos, pese a que no contar con índices y sus 40GB de datos están almacenados en 117 ficheros planos de tipo CSV.

[16] https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/

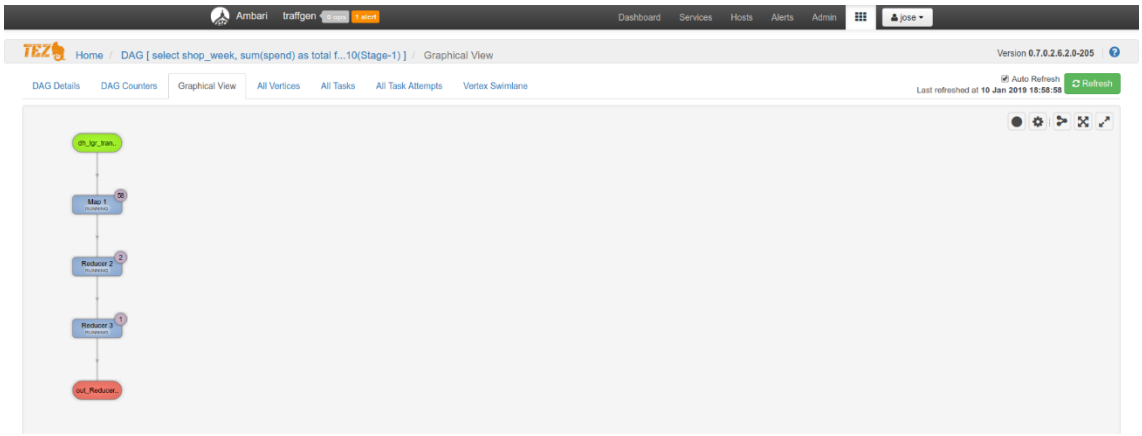


Ilustración 30 Grafo de ejecución de la consulta SQL

En Ilustración 30 Grafo de ejecución de la consulta SQL se observa que Apache Tez ha decidido generar una tarea *Map* y dos tareas *Reduce* en este caso, que a su vez serán distribuidas entre los distintos nodos como se indica en la siguiente figura:

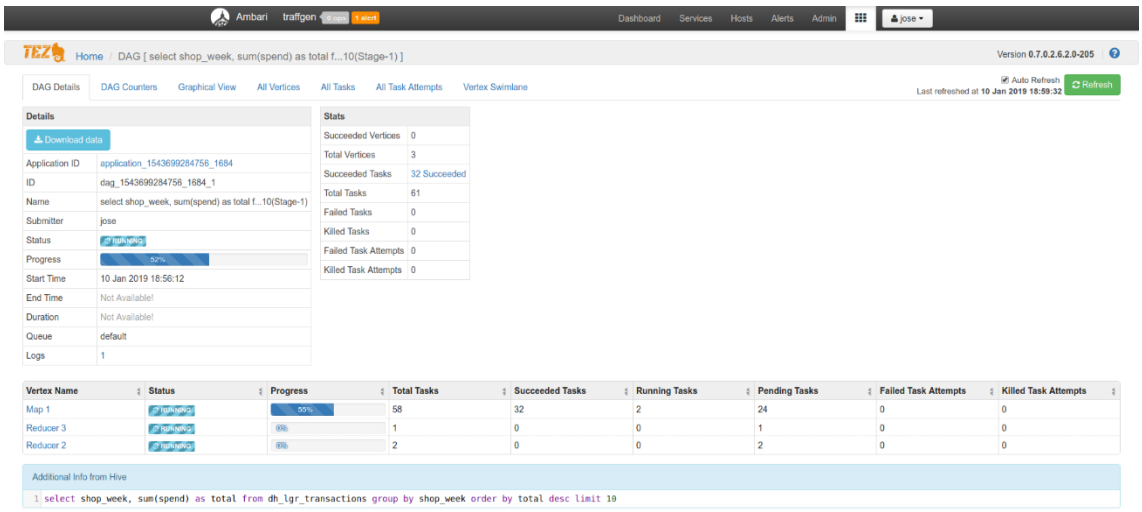


Ilustración 31 Tareas MapReduce resultado de la consulta de agregación

El rendimiento mostrado da idea de la capacidad de la plataforma para la realización de consultas exploratorias con granularidad completa y sobre conjuntos de datos completos.

Con respecto a la creación de Data Marts, resulta sencillo puesto que se pueden generar definiciones de tablas accesibles mediante el lenguaje SQL con independencia del tipo o la ubicación de los datos, y se pueden establecer multitud de esquemas, incluso parciales, sobre los mismos ficheros.

En cuanto a la seguridad de la información, el control de acceso de realiza en varios niveles:

- Control de acceso en la base de datos, mediante privilegios de usuario.

- Control de acceso en el controlador de acceso a la base de datos, de manera que incluso en caso de fallo en las restricciones impuestas por la base de datos, el controlador de acceso a los datos utilizado por las aplicaciones implementa sus propias políticas de seguridad mediante el servicio Apache Ranger.

- Control de acceso en el sistema de ficheros, de manera que no se pueden realizar consultas sobre ficheros de los que se carece de los permisos de acceso, lectura y/o escritura.

5.5. Cubo OLAP en Apache Kylin

Apache Kylin incorpora compatibilidad completa con las tablas declaradas en Apache Hive, por lo que toda fuente de datos susceptible de ser consultada por Hive no necesita ser cargada en Kylin.

Así, se elimina el proceso ETL del flujo de vida habitual del Data Warehouse y sólo es necesario indicar a Kylin qué tablas de Hive se desean sincronizar (y como hemos visto, las tablas de Hive pueden apuntar a simples ficheros CSV), un proceso instantáneo ya que sólo se cargan los metadatos:

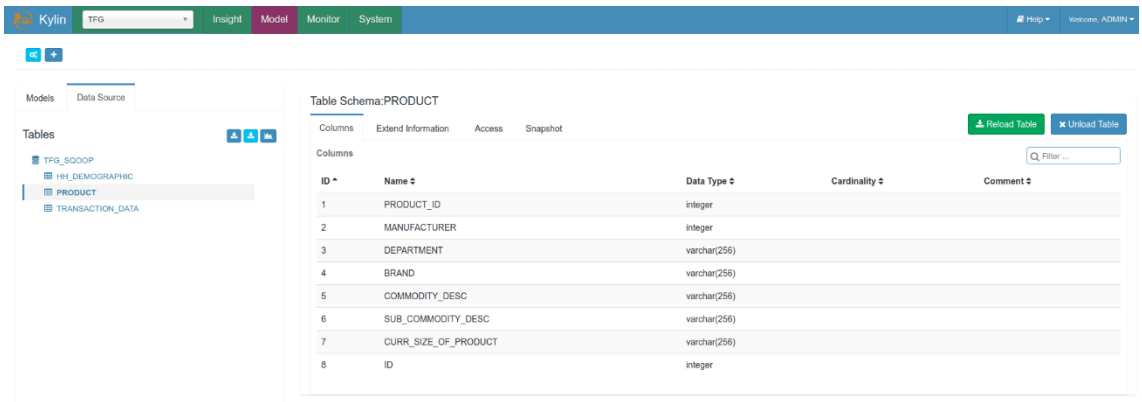


Ilustración 32 Sincronización de tablas de Hive en Kylin

El siguiente paso es la creación del modelo dimensional, requisito previo para la generación de cubos, para lo que se indican las relaciones asociadas.

Paso 1 – Definición de tablas de hechos y dimensiones

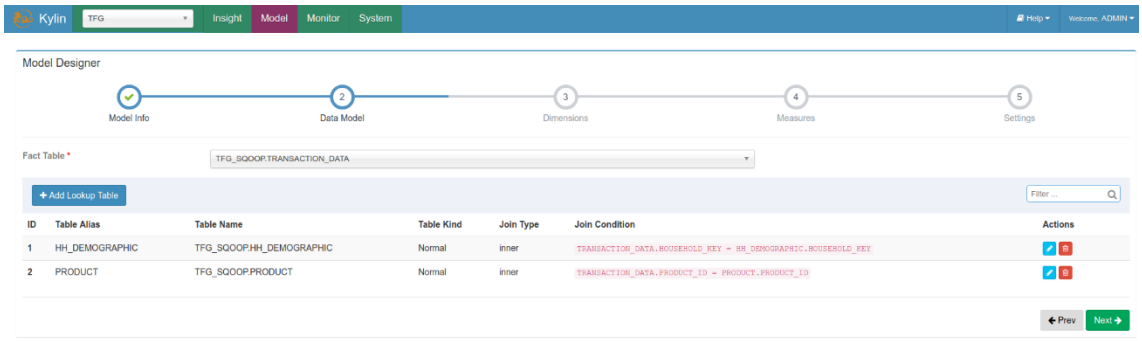


Ilustración 33 Definición de hechos y dimensiones

Paso 2 – Definición de las columnas asociadas a las distintas dimensiones en las tablas relacionadas en el paso anterior.

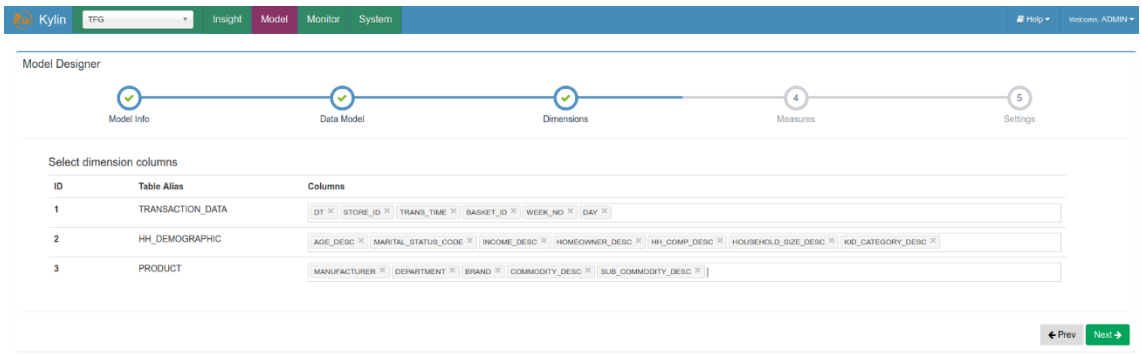


Ilustración 34 Selección de las columnas dimensión

Paso 3 - Se indican las métricas que se desean pre-calcular:

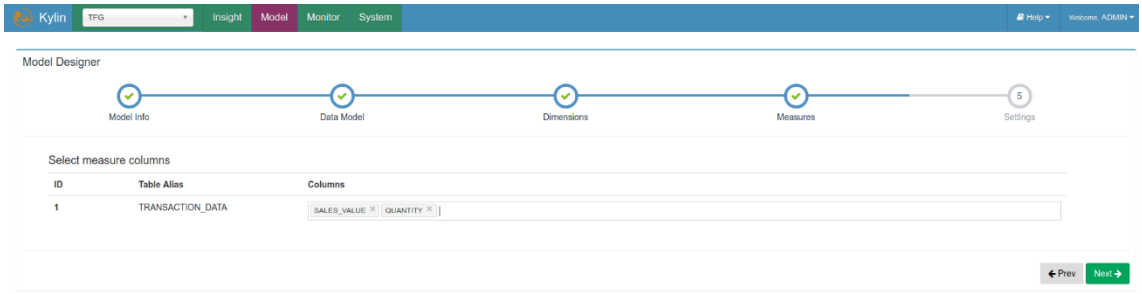


Ilustración 35 Definición de las métricas

Paso 4 – Agregaciones válidas:

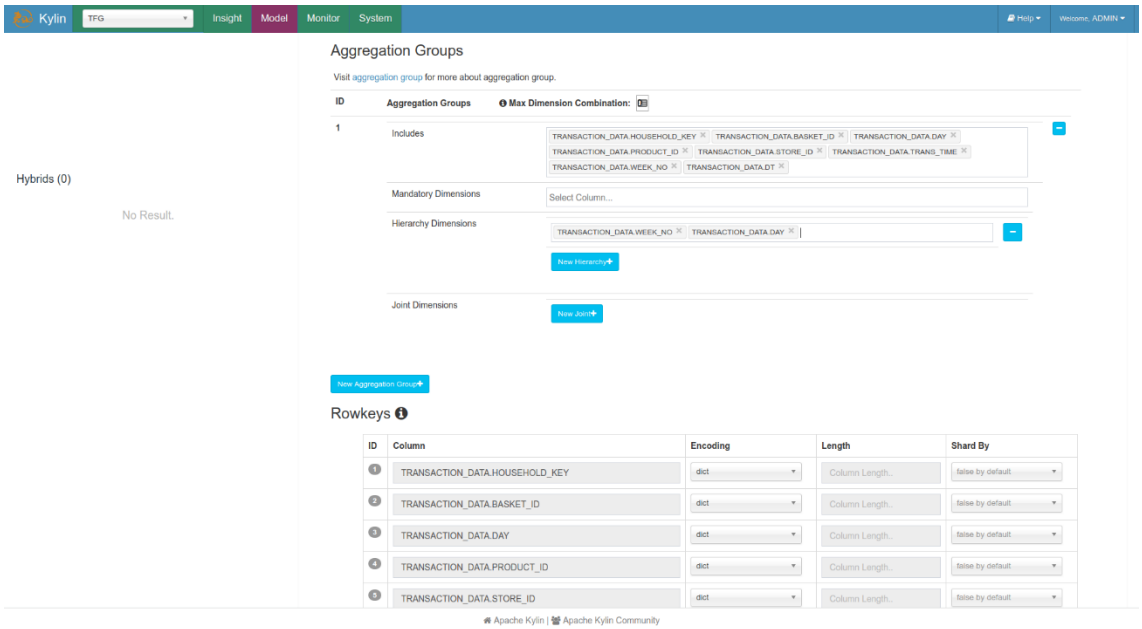


Ilustración 36 Agregaciones válidas

Paso 5 – Parámetros de configuración del cubo:

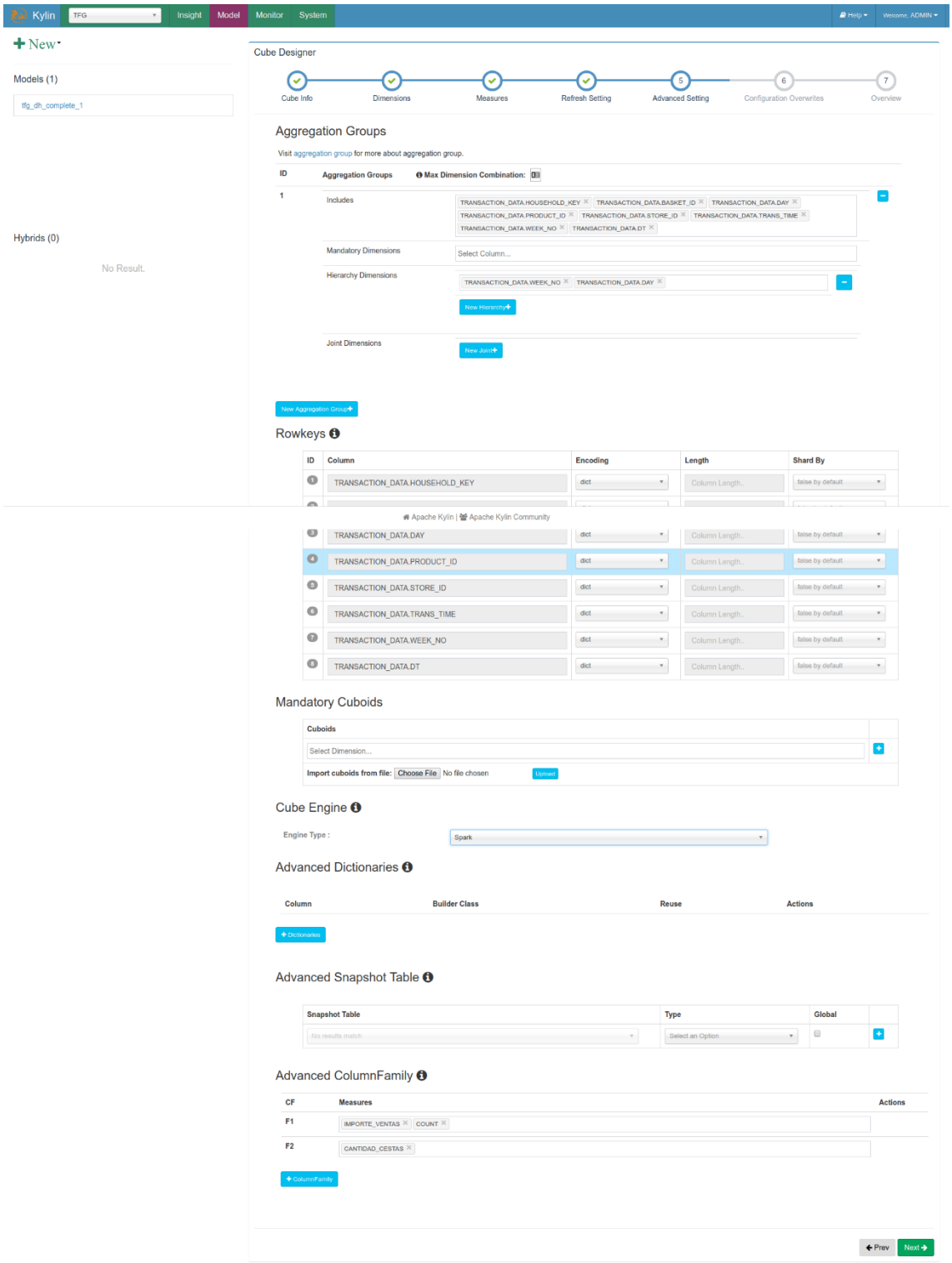


Ilustración 37 Configuración del cubo

Con la configuración generada se puede proceder a la construcción el cubo, que hace uso de la base de datos no relacional Apache HBase como capa de persistencia de las computaciones.

El cubo resultante puede consultarse mediante lenguaje SQL, ya que Kylin proporciona una interfaz JDBC, así como un cliente basado en la web para realizar consultas interactivas.

Kylin detecta automáticamente cuando una consulta SQL es susceptible de utilizar el cubo pre-computado y ofrecerá los resultados sin comunicarse con Apache Hive, como se aprecia en la siguiente figura:

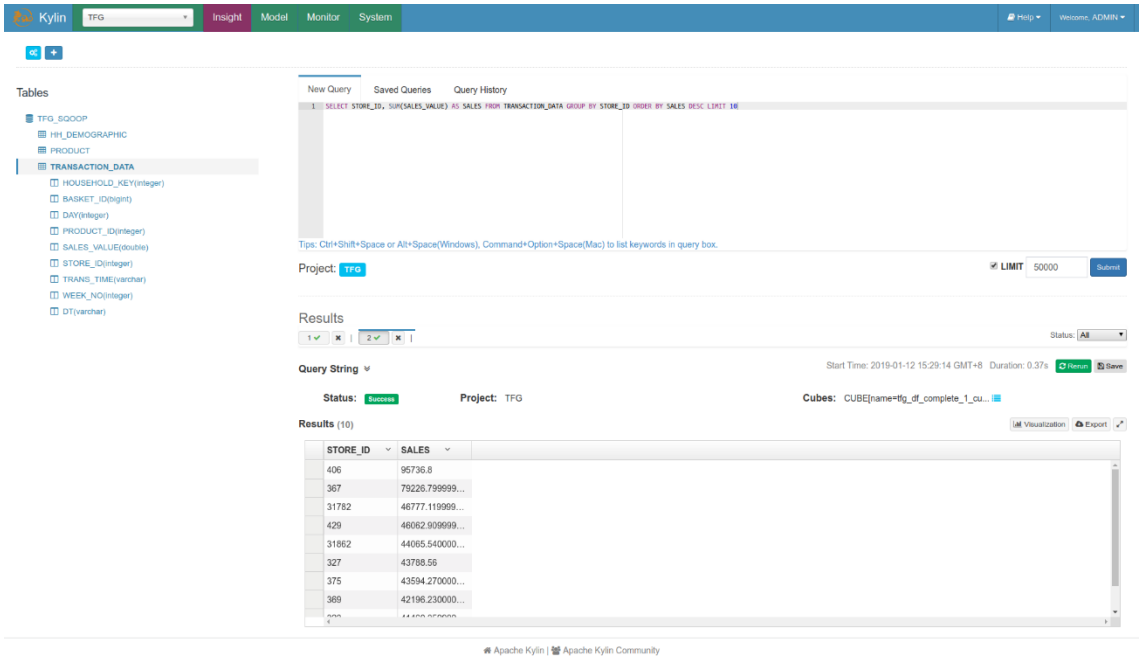


Ilustración 38 Consulta SQL sobre un cubo Kylin

La herramienta web también permite pivotar los resultados directamente:

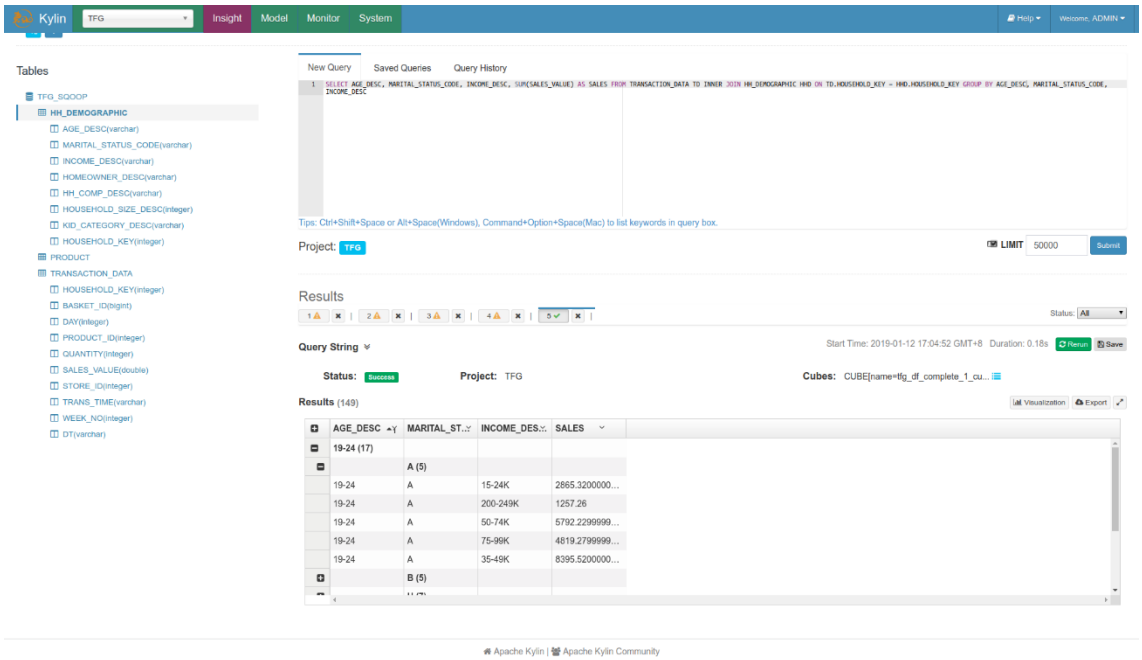


Ilustración 39 Pivotación de resultados SQL

Así como la generación de gráficas directas desde los resultados SQL:

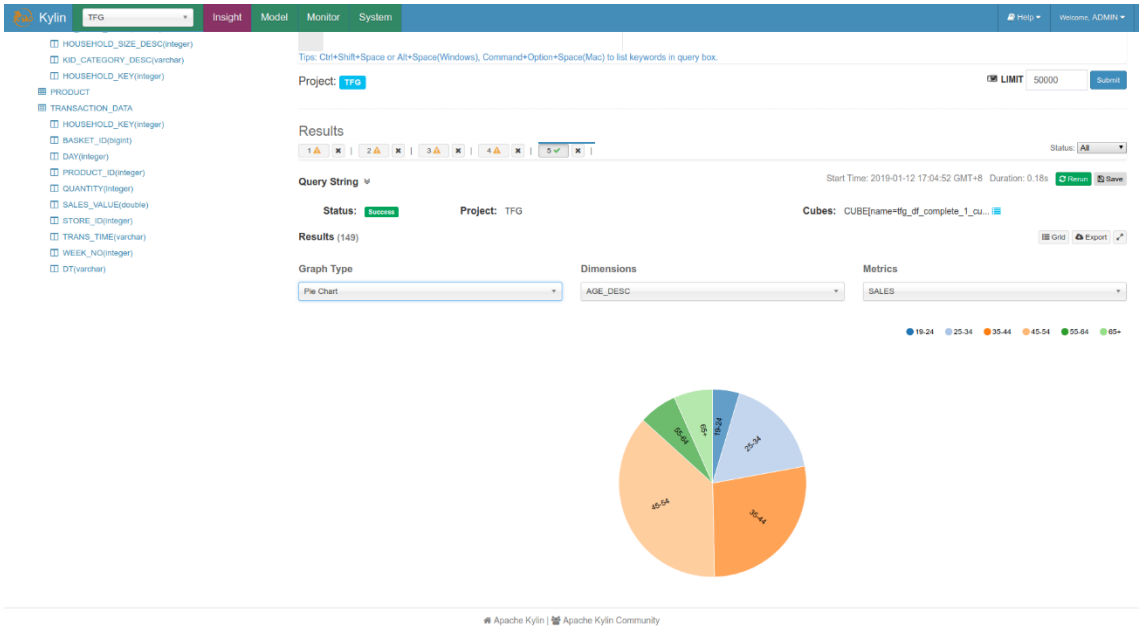


Ilustración 40 Generación de gráficas en Kylin

La principal ventaja de Apache Kylin es su integración en el ecosistema de Hadoop, ya que se beneficia de la flexibilidad del paradigma de Data Lake y *schema-on-read* de Hive, así como del uso de un lenguaje estándar y común a todas las herramientas de la plataforma como es SQL, todo ello ofreciendo funcionalidades de generación de cubos OLAP tradicionales.

Además, Kylin ofrece capacidades de ingestión de datos en flujo, de manera que se pueden producir cubos con información actualizada en tiempo casi real gracias a la utilización de Apache Hbase como almacenamiento de las agregaciones de sus cubos.

Así, cuando se reciben nuevos datos, resultan en comandos UPDATE sobre Apache Hbase, un almacenamiento clave-documento distribuido que soporta concurrencias de escritura extremadamente altas con rendimientos sub-segundo.

5.6. Modelo predictivo avanzado: análisis de cesta de la compra

Dado que el caso escogido pertenece al sector *retail* de alimentación, para ilustrar las capacidades de minería de datos de la plataforma se ha escogido un modelo de análisis de cesta de la compra, basado en el algoritmo *FP-Growth* (Wang et al. 2002).

La solución se ha implementado en lenguaje R[17] mediante la librería SparkR[18] que permite el acceso a los datos almacenados en Hive desde R. En particular se ha utilizado el entorno de desarrollo RStudio[19] Server, que permite desarrollar aplicaciones R desde el propio servidor Hadoop.

Se ha decidido utilizar RStudio en vez de Apache Zeppelin debido a que la solución completa se implementa en R y la principal ventaja de Zeppelin es precisamente la posibilidad de ejecutar una diversidad de lenguajes en un mismo cuaderno, algo no necesario en este caso.

5.6.1. Cuaderno RStudio

En primer lugar, se inicializa la conexión de R con Spark:

```
Sys.setenv(SPARK_HOME = "/usr/hdp/current/spark2-client")
library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
sc <- sparkR.init(master = "local[*]", sparkEnvir = list(spark.driver.memory="2g"))
```

```
## Launching java with spark-submit command /usr/hdp/current/spark2-client/bin/spark-submit   --driver-memory "2
g" sparkr-shell /tmp/RtmpezRL96/backend_port52ac1111ff34
```

A continuación, se carga la tabla de Hive correspondiente a los 307 millones de transacciones de *Dunnhumby – Let's Get Real*, seleccionando únicamente el identificador de ticket o cesta y una combinación de le descripción del producto y su identificador, de manera que el resultado sea legible directamente:

```
hiveContext <- sparkRHive.init(sc)
results <- sql(hiveContext, "select basket_id, CONCAT_WS(' - ', sub_commodity_desc, t.product_id) as item from tf
g_sqoop.transaction_data t inner join tfg_sqoop.product p on t.product_id = p.product_id")
```

Se verifica que se han cargado los datos correctamente:

```
head(results)
```

```
##     basket_id                              item
## 1 26984851472 POTATOES RUSSET (BULK&BAG) - 1004906
## 2 26984851472    ONIONS SWEET (BULK&BAG) - 1033142
## 3 26984851472                    CELERY - 1036325
## 4 26984851472                  BANANAS - 1082185
## 5 26984851472           ORGANIC CARROTS - 8160430
## 6 26984851516             HAMBURGER BUNS - 826249
```

Transformamos los datos para que sean listas de productos asociadas a una cesta única:

```
data = agg(groupBy(results, "basket_id"), collect_list(results$item))
```

[17] https://www.r-project.org/
[18] https://spark.apache.org/docs/latest/sparkr.html
[19] https://www.rstudio.com/products/rstudio

En este punto los datos están en el formato adecuado para poder entrenar el modelo, dónde se utiliza un soporte y una confianza del 1%:

```
model <- spark.fpGrowth(data, minSupport = 0.01, minConfidence = 0.01, itemsCol = "collect_list(item)")
```

Como primer resultado, podemos obtener una lista de los 5 productos más frecuentes en las listas de la compra:

```
most_freq <- spark.freqItemsets(model)
showDF(most_freq, 5)
```

```
## +--------------------+-----+
## |               items| freq|
## +--------------------+-----+
## | [BANANAS - 1082185]|29778|
## |[GASOLINE-REG UNL...|19820|
## |[FLUID MILK WHITE...|14430|
## |[FLUID MILK WHITE...| 3447|
## |[FLUID MILK WHITE...|12542|
## +--------------------+-----+
## only showing top 5 rows
```

Cuyo resultado indica que el producto más frecuente es el plátano, seguido de la gasolina sin plomo y tres marcas distintas de leche.

Finalmente, obtenemos las reglas de asociación del modelo:

```
association_rules <- spark.associationRules(model)
showDF(association_rules)
```

```
## +--------------------+--------------------+--------------------+
## |          antecedent|          consequent|          confidence|
## +--------------------+--------------------+--------------------+
## |[FLUID MILK WHITE...| [BANANAS - 1082185]|0.23847871152926167|
## |[FLUID MILK WHITE...| [BANANAS - 1082185]| 0.2388773388773389|
## | [BANANAS - 1082185]|[FLUID MILK WHITE...|  0.115756598831352|
## | [BANANAS - 1082185]|[FLUID MILK WHITE...|0.10044328027402781|
## +--------------------+--------------------+--------------------+
```

La conclusión de las reglas es que la leche y los plátanos parece que se compran juntos con relativa frecuencia, por lo que se puede recomendar al departamento de operaciones que utilice este conocimiento para aumentar las ventas, mediante reposicionamiento del surtido o creación de paquetes promocionales indivisibles.

Consideraciones clave:

- El hecho de que puedan utilizar los lenguajes de prototipado habituales en la industria directamente desde el clúster de Hadoop posibilita desarrollar modelos de los juegos de datos completos sin las restricciones de recursos (memoria, almacenamiento) propias de las estaciones de trabajo.

- Tanto los analísticas de negocio como los científicos de datos pueden compartir una misma plataforma, evitando duplicidades.

6. Conclusiones

6.1. Descripción de las conclusiones

El paradigma Data Lake aporta una serie de ventajas que lo hacen interesante para su uso en organizaciones que deseen desplegar una estrategia global de obtención del conocimiento a partir de los datos, con independencia de los objetivos específicos y de su alcance:

- El mecanismo de *schema-on-read* permite guardar primero los datos y pensar luego qué hacer con ellos, evitando tener que orientar toda la estrategia de procesamiento alrededor de un objetivo de negocio específico para el que se ajusta el modelo de datos escogido.

- Como resultado, diferentes consumidores pueden hacer uso de los datos para fines diversos sin interferir con posibles transformaciones de estos que hayan sido realizadas por otros interesados para sus propios fines.

- El departamento de sistemas debe operar y mantener una única plataforma, con su correspondiente ahorro de costes.

- Aumento de la seguridad, al contar con una solución centralizada de autenticación y autorización para toda la organización, implementable en todas las aplicaciones con acceso al Data Lake.

- Facilita la obtención de ventajas competitivas ya que la plataforma permite combinar fuentes de datos heterogéneas sin necesidad de intervención del área de sistemas, lo que facilita la minería de datos y la realización de experimentos y comprobación de hipótesis.

- Facilita la puesta en producción del conocimiento adquirido, por sus características avanzadas de intercambio de información desde y hacia bases de datos relacionales y no relacionales de los sistemas de producción.

Sin embargo, la adopción de una solución basada en Data Lake no está exenta de desventajas:

- La heterogeneidad de las tecnologías empleadas requiere contar con personal altamente especializado para la gestión de la plataforma.

- La heterogeneidad de las herramientas disponibles también requiere que los perfiles de analista cuenten con una formación

multidisciplinar y no se puedan limitar a dominar una única herramienta líder en el mercado, como sucede en muchos casos.

- La combinación de los dos puntos anteriores recomienda contar con perfiles que no existen como tales en la actualidad: informáticos con amplios conocimientos de análisis y analistas con amplios conocimientos informáticos, dada la combinación de habilidades que requieren los roles de implementación de soluciones basadas en aprendizaje automático y Big Data.

No obstante, en un entorno de transformación digital generalizada en todos los sectores, las áreas de Business Intelligence ofrecen grandes oportunidades de obtención de ventajas competitivas dado que la gran mayoría del mercado está siendo servido por un reducido grupo de soluciones BI, lo que diluye las ventajas estratégicas de contar con una solución de este tipo y lo convierte en una necesidad operativa, tal y como sucede con los despliegues ERP.

En este sentido, una estrategia que adopte un *mix* específico de soluciones de tratamiento de datos *best of breed* sobre una plataforma Data Lake puede diferenciar una empresa y otorgar ventajas competitivas significativas, sobre todo en un entorno donde el software se incorpora en cada vez más productos, lo que facilita la implementación y puesta en producción del conocimiento adquirido, por ejemplo mediante una actualización remota de un producto tradicional.

6.2. Consecución de los objetivos planteados

El objetivo fundamental de ilustrar las ventajas del paradigma Data Lake para aportar flexibilidad a la hora de diseñar soluciones orientadas a la obtención de conocimiento se ha conseguido suficientemente, poniendo de relevancia tanto las ventajas como los inconvenientes y demostrando facetas de ambas durante el propio desarrollo de la implementación.

Sin embargo, los objetivos secundarios de desarrollo de una solución Business Intelligence mediante Hadoop a través de un caso de empresa y diversos juegos de datos de simulación sólo se han conseguido con fines ilustrativos, ya que la propia implementación tecnológica es lo bastante compleja como para dejar la implementación de negocio parcialmente fuera del alcance del trabajo, que tendría entidad propia para constituir un proyecto propio.

En particular, el objetivo de desarrollo de *dashboards* mediante la herramienta Apache SuperSet no se ha conseguido debido a la complejidad de su implementación y la relativa escasez de información técnica para resolver los problemas encontrados, en particu

Apache Superset pertenece la incubadora de proyectos de Apache y como resultado su versión actual (enero de 2019) es 0.29, por lo que no se considera listo para su utilización en entornos de producción.

La intención original era conectar SupetSet con Apache, pero la versión proporcionada por Hortonworks en su distribución HDP incluye conectores y documentación únicamente para Apache Druid.

Apache Druid también es un proyecto de la incubadora de Apache y su versión estable es la 0.12, por lo que tampoco se considera preparado para su uso en entornos de producción ni cuenta con una base de usuarios suficientemente amplia como para permitir la resolución rápida de los distintos problemas encontrados durante la implementación.

Ambos proyectos se incluyeron en la planificación debido a que el hecho de ser soportados por Hortonworks otorgaba una cierta garantía de estabilidad, pero a la vista de los acontecimientos se ha tratado de un error.

Como conclusión, se debe mejorar la investigación del grado de madurez de las distintas herramientas propuestas para el desarrollo de un proyecto de esta naturaleza.

6.3. Análisis crítico de la metodología del proyecto

La complejidad de las soluciones tecnológicas propuestas y su heterogeneidad ha causado un *gap* significativo entre planificación e implementación, por lo que la decisión de diseñar de manera teórica la solución como paso previo a su implementación puede ser mejorada.

En particular, diseñar el cronograma y plan de trabajo antes de la fase de investigación y diseño de la solución no es una aproximación óptima a menos que se conozcan de antemano las características detalladas de las herramientas a utilizar, pero como se ha indicado, la complejidad de una solución Big Data hace relativamente inabarcable este problema.

No obstante, la propia flexibilidad del ecosistema de aplicaciones de Hadoop ha permitido en la mayoría de los casos obtener una solución equiparable a la originalmente diseñada, si bien mediante una técnica o herramienta diferente. Se trata, en definitiva, de un ejemplo más de las ventajas de operar con plataformas abiertas, ya que una dependencia de características específicas de un producto comercial hubiese resultado más difícil de subsanar en caso de encontrar dificultados de implementación.

Como conclusión, parece recomendable evitar mencionar herramientas específicas y apelar a la funcionalidad que implementan en la medida de lo posible.

6.4. Líneas de trabajo futuro

El presente trabajo se ha centrado en la faceta tecnológica de la solución propuesta, pero sin duda existen otras consideraciones que cabría tener en cuenta, pero han quedado fuera del alcance del proyecto:

- Impacto del despliegue de una plataforma de datos única en la estructura organizativa de las áreas de Inteligencia de Negocio y Big Data Analytics, dado que la separación de sus objetivos de negocio es relativamente difusa pero habitualmente operan de manera independiente.

- Impacto en la estrategia digital de la organización, ya que las posibilidades de análisis en tiempo casi real de los datos generados por las áreas operativas dan lugar a nuevos usos inmediatos del conocimiento adquirido, de nuevo en las áreas operativas, sin intervención de las capas táctica y estratégica, restando relevancia a los mandos intermedios y aumentando la criticidad de una estrategia de SI a largo plazo que adopte un entorno dinámico.

- Comparativas de rendimiento de soluciones comerciales que implementan el paradigma indicado, tanto de actores tradicionales que han adoptado Big Data (como por ejemplo Teradata) como por nativos digitales (como por ejemplo Databricks).

7. Glosario

AVRO – Formato de serialización y almacenamiento de datos por filas en Hadoop.

DDL – Data Definition Language, utilizado para definir los esquemas de tablas en Hive.

DW – Data Warehouse, paradigma de almacenamiento de información en un formato que facilite su análisis, habitualmente siguiendo un modelo de datos en estrella.

ETL – Extracción, Transformación y Carga de datos, por sus siglas en inglés.

HDFS – Sistema de ficheros distribuido de Hadoop, que opera como un servicio y programado en Java.

HDP – Hortonworks Data Platform, producto de código libre de Hortonworks para la gestión de almacenes de datos hadoop.

JSON – Formato de intercambio de información basado en los tipos de datos nativos de Javascript y que se utiliza habitualmente en la implementación de API REST, de manera similar a XML en API SOAP en el pasado.

LDAP – Protocolo de autenticación empleado por Microsoft y soportado por Hadoop.

OAUTH – Protocolo abierto de autenticación implementado como estándar.

OLAP – Base de datos analítica, orientada a la obtención de conocimiento.

OLTP – Base de datos transaccional, orientada a entornos operativos.

ORC – Formato de serialización y almacenamiento de datos por columnas, creado por Microsoft y liberado como código libre.

RDBMS – Sistema de gestión de base de datos relacional.

RSA – Algoritmo de cifrado estándar.

CSV – Formato de texto plano delimitado.

SKU – Unidad inventariable en un almacén.

SQL – Lenguaje procedimental para la gestión de información almacenada en una base de datos relacional.

SSH – Protocolo y aplicación de conexión remota a consolas de línea de comandos.

8. Bibliografía

130115_relationship_between_supermarkets.pdf [en línea], [sin fecha]. S.l.: s.n. [Consulta: 17 noviembre 2018]. Disponible en: http://www.promarca-spain.com/pdf/130115_relationship_between_supermarkets.pdf.

Benchmarking Apache Kafka: 2 Million Writes Per Second (On Three Cheap Machines). [en línea], 2014. [Consulta: 12 diciembre 2018]. Disponible en: https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines.

BREWER, E.A., 2000. Towards robust distributed systems. PODC. S.l.: s.n.,

CHANG, F., DEAN, J., GHEMAWAT, S., HSIEH, W.C., WALLACH, D.A., BURROWS, M., CHANDRA, T., FIKES, A. y GRUBER, R.E., 2008. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), vol. 26, no. 2, pp. 4.

CORR, L. y STAGNITTO, J., 2014. Agile data warehouse design : collaborative dimensional modeling, from whiteboard to star schema. S.l.: s.n. ISBN 978-0-9568172-0-4. /z-wcorg/

CURTO, J., 2017. Introducción al Business Intelligence. S.l.: Editorial UOC. Manuales.

DEAN, J. y GHEMAWAT, S., 2008. MapReduce: simplified data processing on large clusters. Communications of the ACM, vol. 51, no. 1, pp. 107-113.

DI FATTA, D., PATTON, D. y VIGLIA, G., 2018. The determinants of conversion rates in SME e-commerce websites. Journal of Retailing and Consumer Services, vol. 41, pp. 161-168.

DONGEN, J. van. y BOUMAN, R., 2013. Pentaho solutions : business intelligence and data warehousing with pentaho and mysql. [en línea]. Disponible en: http://rbdigital.oneclickdigital.com.

KIMBALL, R. y ROSS, M., 2013. The data warehouse toolkit : the definitive guide to dimensional modeling. [en línea]. Disponible en: http://www.books24x7.com/marc.asp?bookid=56391.

MARZ, NATHAN, 2011. How to beat the CAP theorem - thoughts from the red planet - thoughts from the red planet. [en línea]. [Consulta: 11 diciembre 2018]. Disponible en: http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html.

MAURI, C., 2003a. Card loyalty. A new emerging issue in grocery retailing. Journal of Retailing and Consumer Services, pp. 13.

MAURI, C., 2003b. Card loyalty. A new emerging issue in grocery retailing. Journal of Retailing and Consumer Services, vol. 10, no. 1, pp. 13-25.

NICHOLSON, C. y YOUNG, B., 2012. The relationship between supermarkets and suppliers: What are the implications for consumers. Consumers International, vol. 1, pp. 7-8.

Nielsen Global Connected Commerce Report January 2017.pdf [en línea], [sin fecha]. S.l.: s.n. [Consulta: 19 noviembre 2018]. Disponible en: https://www.nielsen.com/content/dam/nielsenglobal/de/docs/Nielsen%20Global%20Connected%20Commerce%20Report%20January%202017.pdf.

NORI, S. y SCOTT, J., 2016. BI & Analytics on a Data Lake. S.l.: MAPR.

RAJTERIČ, I.H., 2010. OVERVIEW OF BUSINESS INTELLIGENCE MATURITY MODELS. Management, vol. 15, no. 1, pp. 47-67.

WANG, K., TANG, L., HAN, J. y LIU, J., 2002. Top down fp-growth for association rule mining. Pacific-Asia Conference on Knowledge Discovery and Data Mining. S.l.: Springer, pp. 334-340.

What's in-store for online grocery shopping [en línea], 2017. enero 2017. S.l.: Nielsen. Disponible en: https://www.nielsen.com/content/dam/nielsenglobal/de/docs/Nielsen%20Global%20Connected%20Commerce%20Report%20January%202017.pdf.