

Big Data Architecture

Slide 3, Diseño. Bullet 1.

- ❑ **Lo primero a definir, es la analítica a realizar. Por ejemplo, construcción de un “tablón” de indicadores de Cliente.**
 - ❑ **Estos indicadores provendrán en algunos casos de datos de las BBDD con una calidad aceptable, y otras veces no.**

KPIs

La realización de la práctica será la realización de un Data Lake. A partir de un DWH la realización de dicho proyecto nos permitirá:

- Aumentar la facilidad para utilizar los datos de la organización, tanto para reporting como para analytics.
- Más agilidad gracias a que su trabajo se adapta en función de las necesidades del usuario, pudiendo procesar los datos en tiempo real si así fuera necesario.
- Mayor capacidad y flexibilidad en un entorno en el que confluyen los datos existentes en el DWH

El DWH es el primer paso para poder tomar decisiones en una empresa, por eso no hay que prescindir de él. Sin embargo, el Data Lake permite ir más allá, al mejorar algunas de las ineficiencias del repositorio tradicional:

- Requiere de demasiado tiempo para analizar las fuentes de datos y las necesidades del negocio, dando como resultado un modelo muy estructurado de los datos que es el único que permite construir informes.
- Para poder consumir datos, necesita que toda la información se someta a un proceso de modelado previo, a diferencia del Data Lake, donde no se pierde el tiempo en esa transformación, sino que se invierte la mayor parte en la ingesta de datos.
- Debido a su capacidad limitada, obliga a seleccionar muy bien el tipo de datos a almacenar. Además, hay que prestar atención a su formato y fuente de origen, puesto que no todos se pueden recoger en el DWH. Los Data Lakes permiten almacenar todos los tipos de datos (pdf, Excel, twitter, Facebook,gps,etc.)

En nuestro caso, vamos a realizar un análisis de datos del cliente obtenidos de diversas formas. En primer lugar, utilizaremos los datos que ya hay en el DWH Data101 donde se encuentran diversas bases de datos con sus respectivas tablas. Una de las bases de datos es ODS, la misma ha sido obtenida con la transformación y normalización de los datos que existían en las fuentes transaccionales de la empresa (CRM,Facturador y Centro de Llamadas). Además de estos datos utilizaremos webs que sean de nuestro interés para llevarnos los datos de estas al Data Lake y poder completar la información.

Y por último haremos uso de una red social, donde se podrá conocer la relevancia de nuestra empresa en las redes.

Los KPIs utilizados para presentar a gerencia en la empresa serán los siguientes (se encuentran en el Excel adjunto):

- Relación de TOTAL_FACTURADO por los clientes a lo largo del periodo seleccionado. Los datos vendrán de la base de datos relacional
- METODO_DE_PAGO utilizado por los clientes. Los datos vendrán de la base de datos relacional.
- Datos relativos al perfil del cliente de nuestra empresa (sexo, edad, etc). Los datos vendrán de la base de datos relacional.
- Relación entre el poder adquisitivo de un estado y lo que consumen nuestros clientes. Para este KPI necesitaremos información tanto de la base de datos relacional como de información que podamos encontrar en webs.
- Relación entre la edad de nuestros clientes y lo que consumen. Estos datos se podrán encontrar en la base de datos relacional.
- Relación de clientes que tiene nuestra empresa en cada estado en función del número total de habitantes que tiene el estado. Para este KPI necesitaremos información tanto de la base de datos relacional como de información que podamos encontrar en webs.
- Relevancia que tiene nuestra empresa en las redes sociales. Estos datos son los más diferentes del resto y necesitaremos fundamentalmente información que venga de las redes sociales y podamos capturarla en tiempo real.

Los KPIs que presentaremos a gerencia nos servirán para ser utilizados por un sistema de visualización y permitir que la dirección de la empresa pueda tomar las mejores decisiones para la empresa. En nuestro caso hemos seleccionado los KPIs que pensamos que permitirán a la empresa ver la evolución del negocio, tomar medidas correctoras en caso de que algún parámetro no sea cómo se había previsto anteriormente. Además, los KPIs que hemos seleccionado permitirán a la empresa la definición de su tipo de cliente y las acciones de marketing que puede tomar en función de datos como el número de clientes que tiene la compañía en un determinado estado en relación con la población total accesible de contratar los servicios de la compañía.

La obtención de los datos la realizaremos a través de 3 fuentes principales:

- La primera es el sistema de bases relacionales que dispone la empresa y que hace poco tiempo fue normalizado y revisado para que se pudiera extraer la mayor cantidad de información posible.
- La segunda es la utilización de webs que dispongan de información útil para la empresa. Este tipo de información se considera en la mayoría de los casos “no estructurada” al no estar tan bien definida como la anterior. Y aunque a veces nos servirá un archivo csv o json que podamos obtener, otras veces tendremos que hacer web scrapping para sacar normalmente mediante scripts de Python la información que nos interesa. De aquí sacaremos información sociodemográfica, que podemos encontrar de forma simple en la red.

- Por último, obtendremos la información que nos pueden proporcionar las redes sociales. Este tipo de información es también considerada como “no estructurada” y en la mayoría de los casos necesitaremos capturarla en tiempo real para que tengan mayor valor.

En definitiva, toda esta información dará a la empresa una “visión” de si se está cumpliendo su estrategia y de las medidas que se tienen que tomar en consecuencia. Debido a que la empresa para la cual vamos a realizar el Data Lake sólo disponía hasta la fecha de información a nivel transaccional, vamos a generar un primer tablón de KPIs que se irá actualizando y ampliando en función de las necesidades de la misma.

Lo que es importante es que la empresa ha comprendido que no sólo son útiles los datos que tenía y que provenían de un sistema relacional, sino que existen otros datos que son los considerados Big Data y que permiten ampliar las analíticas que se llevan a cabo dentro de una empresa.

El cambio de paradigma que ha posibilitado que podamos hacer todo esto es en primer lugar la utilización del schema on read. Es decir, ya no es como antes donde era necesario que la estructura del objeto de la base de datos estuviera definida previa la carga, si no que para estos tipos de datos que se salen de los tradicionales la estructura se define en la lectura en función del dato que se haya leído, con la flexibilidad de poder cambiar la misma en cada tipo de lectura.

Este tipo de utilización de datos Big Data está siendo posible gracias principalmente a disponer de un sistema de archivos distribuidos (HDFS) y herramientas que permiten el procesamiento en paralelo gracias al paradigma Map Reduce. Al final estos dos conceptos sobre los que se basan el ecosistema de herramienta Hadoop que es el más utilizado para llevar a cabo el tratamiento de Big Data posibilitan que el crecimiento de los clústeres con los que trabajaremos se produzca de forma horizontal y con hardware no necesariamente caro en contra de lo utilizado hasta su aparición que era el escalado mediante la utilización de superordenadores.

Muchas veces el tratamiento de este tipo de datos va asociado a bases de datos NoSQL que nos van a dar esa flexibilidad necesaria para trabajar con este tipo de dato. Existen muchos tipos de bases de datos NoSQL, pero HBase (que viene con Hadoop) es un buen ejemplo de las ventajas que nos ofrecen.

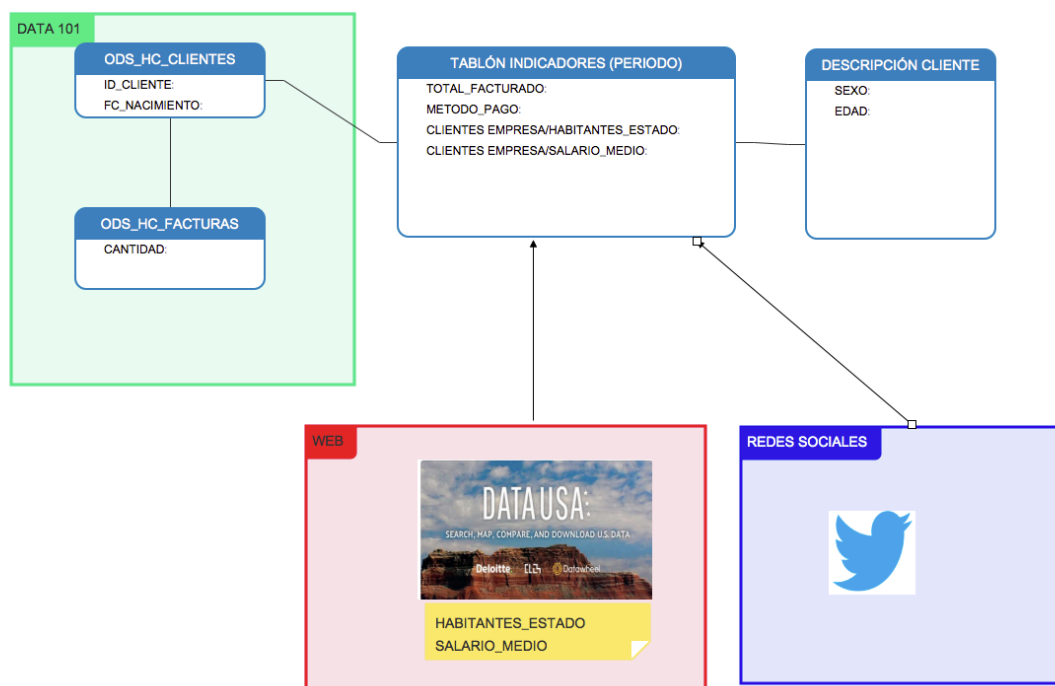
Es importante también señalar que gracias a herramientas como Hive los datos van a ser explotados por “SQL engines” que permiten a los usuarios de las bases de datos relacionales poder seguir utilizando SQL aunque bajo de esta capa se encuentre todo el paradigma Hadoop.

Slide 3, Diseño. Bullet 2.

- ❑ Habrá que definir un modelo lógico para este “datamart” y luego un modelo físico sobre Hive, indicando de forma inicial el tipo de almacenamiento, compresión, particionamiento, bucketing, cuando se trata de tablas externas o gestionadas, etc.
 - ❑ Es importante justificar la decisión de diseño
 - ❑ Opcionalmente, puede plantearse un plan de pruebas para un piloto que permita refinar el modelo.

Modelo lógico

A continuación, pasamos a representar el modelo lógico:



Como se puede observar para montar el tablón los datos provendrán de las tablas clientes y facturas de la base de datos relacional que ya tenemos creada. Por otro lado, mediante la utilización de datos obtenidos de webs vamos a enriquecer mucho más el tablón final. Y por último el estudio de los tweets permitirá darnos un tipo de información sobre lo que se está comentando en nuestra empresa que sería imposible conseguir de otra forma. El conjunto de estos tres modelos de obtención de datos permitirá obtener los KPIs que gerencia estima más necesario para la toma de decisiones.

Para definir el modelo lógico mediante Hive (que será el “engine SQL” que utilizaremos de forma principal) tendremos en cuenta las siguientes consideraciones:

TIPOS DE TABLAS

Las tablas que utilizaremos serán gestionadas y externas. Las tablas externas las utilizaremos cuando los datos de las mismas vayan a ser utilizadas por otras herramientas. Además, todas las dimensiones de las tablas que tenemos en nuestra base de datos relacional serán tablas externas debido al poco o nulo cambio que sufren a lo largo del tiempo. Por otro lado, utilizaremos como tablas gestionadas aquellas de las que queramos gestionar de forma total el ciclo del dato y que sólo vayan a ser utilizadas por Hive.

TIPO DE FORMATO

Utilizaremos tablas ORC, orientadas a la lectura de las mismas en detrimento de la escritura. También se ha tenido en cuenta el alto nivel de compresión de las mismas, lo cual beneficiará al sistema. En nuestro caso, serán tablas que se escribirán de forma incremental y pocas veces. En cambio, las mismas serán leídas muchas más veces para las reuniones que tienen todos los departamentos de forma diaria.

Para la toma de esta decisión se han estudiado otros formatos como AVRO, PARQUET, etc.

PARTICIONAMIENTO

El particionamiento de las tablas está pensado para no tener que consultar toda la tabla cada vez que hagamos una query. Es recomendable particionar por las claves que más suelo consultar, de esta forma la query en vez de leerse todas las filas lo que hace es leer las particiones afectadas. En nuestro caso utilizaremos un particionamiento por ID_CLIENTE, que, aunque tenga una cardinalidad muy alta se le pondrá un módulo que será en que indicará cuantas filas guardará por archivos. Para el cálculo del módulo ideal se ha realizado mediante prueba/error hasta que se ha encontrado aquel que nos da un mejor rendimiento.

BUCKETS

Para acelerar todavía más la “performance” de las queries, hemos bucketeado nuestra tabla

Podemos hacer buckets de una columna en concreto. Por lo que si hacemos 5 buckets de la columna ID_CLIENTE, si buscamos un ID_CLIENTE determinado siempre estará

en el mismo bucket con independencia de las particiones. Por lo que si hacemos un join en función del ID_CLIENTE si este se encuentra ya en un bucket el sistema no tendrá que utilizar otros, mejorando la eficiencia de la query.

Y buena forma de bucketear sería en función de unos periodos de tiempo en los que hemos dividido el horizonte temporal, de esta forma si el sistema tiene la información de cada periodo en el mismo bucket.

COMPRESIÓN

Para nuestro caso no será necesario utilizar ningún tipo de compresión extra, únicamente la que viene de forma predeterminada con el tipo de archivo que hemos escogido.

OPTIMIZACIONES DE LOS JOINS A REALIZAR SOBRE HIVE

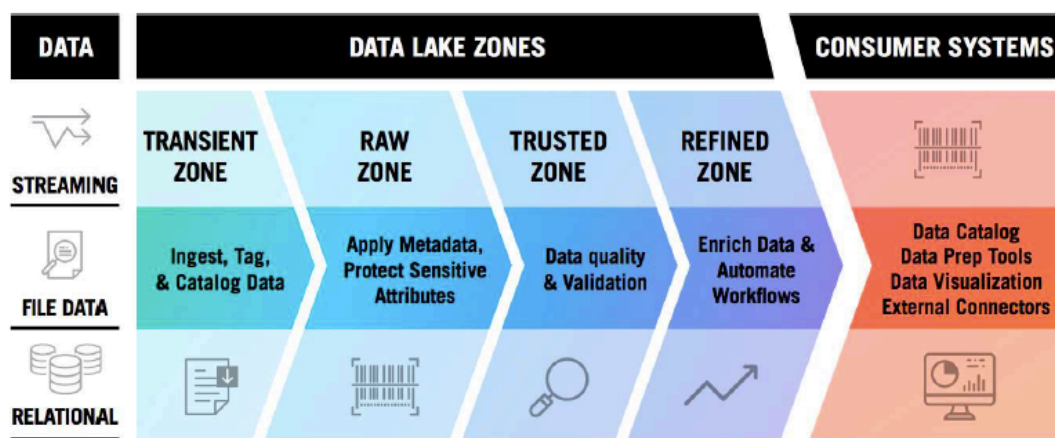
Existe una serie de consideraciones importantes para optimizar la realización de joins en Hive:

- Tablas que tengan el mismo bucketing realizarán los join de forma más eficiente.
- Realizar la query de tal forma que la tabla más pesada aparezca en el join la última.
- La utilización del formato correcto de archivos también permite mejorar la performance de la query.
- La correcta utilización de los parámetros SET.
- La partición adecuada para que no tenga que recorrer toda la tabla en cada consulta.

Slide 4, Diseño. Bullet 1.

- ❑ Una vez definido el modelo, se identificarán las fuentes de datos estructurados y “no estructurados” que formarán parte del proyecto:
 - ❑ Hay que indicar la estrategia de carga inicial y periódica de cada fuente.
 - ❑ También es importante indicar que mecanismos se usarían para garantizar el ciclo de vida y la calidad del dato
 - ❑ Por último, se deberán seleccionar los componentes a utilizar en cada caso y justificar el por qué

Para la construcción del Data Lake vamos a utilizar el siguiente diagrama lógico:



Observamos que con independencia de donde venga el dato, se tendrá que tratar hasta que lo utilicemos.

Los datos pueden venir bien de bases de datos relacionales, de archivos que consideramos “no estructurados” (logs, emails, etc), y pueden llegar en streaming. Todo esto llegará la RAW ZONE, donde se irán guardando todos los archivos que lleguen de forma que se encuentren perfectamente nombrados para que podamos recuperarlos cuando sea necesario. Una actividad que será conveniente realizar será la de añadir un timestamp a dichos archivos para de esta forma tener un control de los mismos que nos permitan realizar determinadas acciones (copias incrementales) y evitar errores como la duplicidad de estos. Los archivos pueden permanecer en esta zona entre un año y cinco en función de la compañía.

Dentro de la zona anterior tenemos una segunda parte llamada STAGING. Esto es una zona temporal donde sólo habrá una porción de los datos (aquellos que vayamos a utilizar a continuación). Es decir, se cargará de forma incremental cada cierto tiempo una serie de datos para que se trabajen antes de poder ponerlos en la TRUSTED ZONE. Este staging se reiniciará cada intervalo de tiempo que tengamos predefinido (1 o 2 días) actualizándose con los datos nuevos que vengan del área anterior. En el STAGING se cargarán en tablas externas datos para ponerlos mediante la herramienta que

consideremos oportuno (Hive, Python,etc) de la forma que nos interese para la siguiente zona.

Una de las herramientas que podemos utilizar para automatizar la transición entre RAW y STAGING es OOZIE.

La TRUSTED ZONE será donde se crearán las tablas (gestionadas o externas) completamente validadas y con los datos que cumplan todas las características que queramos. Los datos vendrán de STAGING y tendrán ya la forma orientada a la analítica.

En nuestro caso tendremos al final un “tablón” desde donde el cual se podrá realizar la analítica de una forma lo más correcta posible.

Origen Dato

A continuación, se incluye una tabla para ver cual es el origen de cada uno de los datos (dicha tabla se puede encontrar en el Excel adjunto). Aquí podremos venir cada uno de los elementos de información que vamos a tener en nuestro tablón y cual es su lugar de origen. Según el sistema que estamos diseñando el dato permanecerá en RAW durante un largo periodo de tiempo, siendo sustituido si existen problemas de almacenamiento.

TABLA ORIGEN	COLUMNA ORIGEN	WEB ORIGEN	NOMBRE DATO	TRANSFORMACIÓN	TABLÓN GENERAL (PERIODO)
					CLAVE
ODS_HC_CLIENTES	ID_CLIENTE		ID_CLIENTE	NINGUNA	ID_CLIENTE
					INDICADORES
ODS_HC_FACTURAS2	CANTIDAD		TOTAL_FACTURADO	SUM()	TOTAL FACTURADO
ODS_HC_FACTURAS2	ID_METODO_PAGO		METODO_PAGO	NINGUNA	METODO_PAGO
					CLAVE
ODS_HC_FACTURAS2	CANTIDAD		TOTAL_FACTURADO	SUM()	TOTAL FACTURADO
					INDICADORES
		DATAUSA.IO	SALARIO_MEDIO	JSON A HIVE	PODER ADQUISITIVO ESTADO
ODS_HC_CLIENTES	FC_NACIMIENTO		FC_NACIMIENTO	NINGUNA	EDAD
					CLAVE
ODS_HC_CLIENTES	ID_CLIENTE			DISTINCT	Nº TOTAL CLIENTES
					INDICADORES
		WIKIPEDIA		BEAUTY SOUP	POBLACIÓN ESTADO
					CLAVE
TWITTER				FLUME	NOMBRE_EMPRESA

Estrategia de carga

Observando el cuadro anterior vamos a definir cuál será la estrategia de carga inicial y las periódicas.

La carga de datos la realizaremos de forma diferente en función del tipo de dato:

- Los datos que vienen de los transaccionales se realizará mediante SQOOP
- Los datos que traemos de las webs las realizaremos mediante librerías de PYTHON
- Los datos que vienen de las redes sociales los cargaremos mediante FLUME

La primera carga será realizada para los tres flujos en un mismo momento y a partir de ahí en función del dato se cargará con mayor o menor frecuencia.

Si cargando desde transaccional o web nos encontramos errores, procederemos a volver a cargar los datos. En cambio, si son datos que proceden de twitter, permitiremos un margen de error de un 3%. En caso de que sea menor no realizaremos ninguna acción y si es mayor procederemos a estudiar como resolver el conflicto.

Gobierno del dato

- Meter una fecha cuando se cargue en staging para que la tabla no se duplique (añadir datos de control a las filas). Para no cargar 2 veces poner en el nombre de la tabla el periodo.
- El tablón indicarle un campo(mensual) para que sea única
- Poner las tablas que no se puedan borrar como externas
- A veces puede ser necesario cargar una tabla auxiliar, y actualizarla periódicamente de forma online
- Para auditar el acceso de cada elemento de información y tener un control del acceso a los mismos utilizaremos RANGER. Este permite o deniega el acceso en función de quien eres a todos los recursos de HADOOP. De esta forma permitiremos únicamente a los administradores manipular parámetros de administración, a ciertos usuarios almacenar datos en RAW, a otros realizar las transformaciones pertinentes sobre los mismos, etc.
- Se realizará un backup de forma diaria e incremental.

Además, para validar los distintos pasos que tienen los datos (inicial,abierto,cargando,cerrado,disponible,inválido,suspendido,archivado,removido) utilizaremos los metadatos indicando el estado de la tabla. Y a esto le añadiremos el mecanismo de seguridad que nos proporciona RANGER.

Para nuestra empresa, llevaremos a RAW una copia de los transaccionales cada semana, de los obtenidos a través de webs una vez al mes y de los provenientes de las redes sociales una vez al día. Desde allí se le añadirá a las tablas un timestamp cuando pasen a STAGING. Esto nos permitirá trabajar de forma incremental, es decir, con la diferencia entre la última tabla creada y la anterior necesitando muchos menos recursos.

Slide 5, Construcción. Bullet 1 al 4.

El código que pondremos en Hive tanto para la creación de las tablas como para la carga de las mismas se encuentra en el siguiente enlace de Github:

<https://github.com/jampol71/BDA>

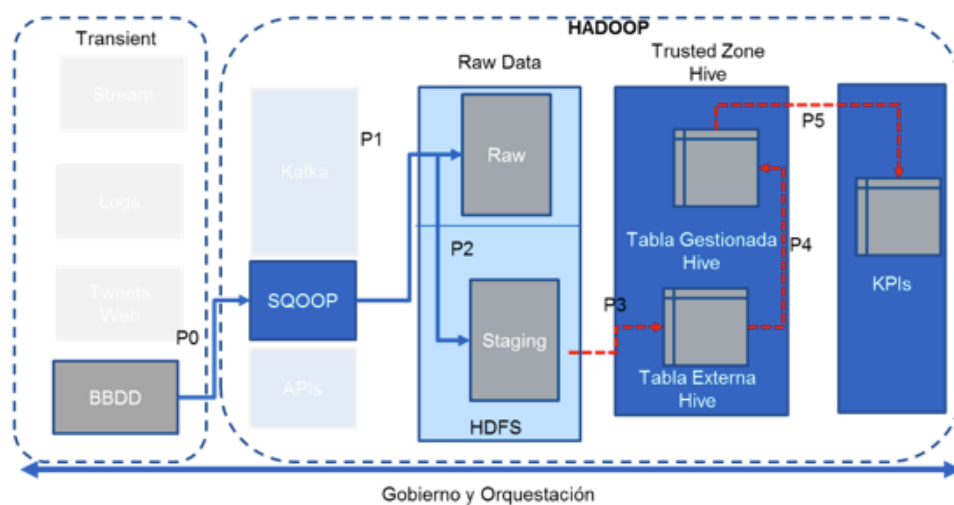
Está dividido en los siguientes archivos:

- codigo_BDA20180114. Archivo de texto plano con el código utilizado.
- BDA_20180114. Archivo pdf con las respuestas de la práctica
- BDA_20180114. Archivo Excel con la trazabilidad de datos

Para la parte práctica, hemos simplificado alguno de los pasos por cuestiones técnicas. En primer lugar, todos los datos, aterrizarán a RAW que será lo mismo que STAGING en nuestro caso.

Para realizar la carga de datos, lo hemos realizado mediante SQOOP para los datos de las bases de datos transaccionales. En lugar de cargar todas las cargas de ODS que teníamos sólo hemos cargado las que necesitábamos para crear nuestro tablón final.

Método 1: BBDD RELACIONALES

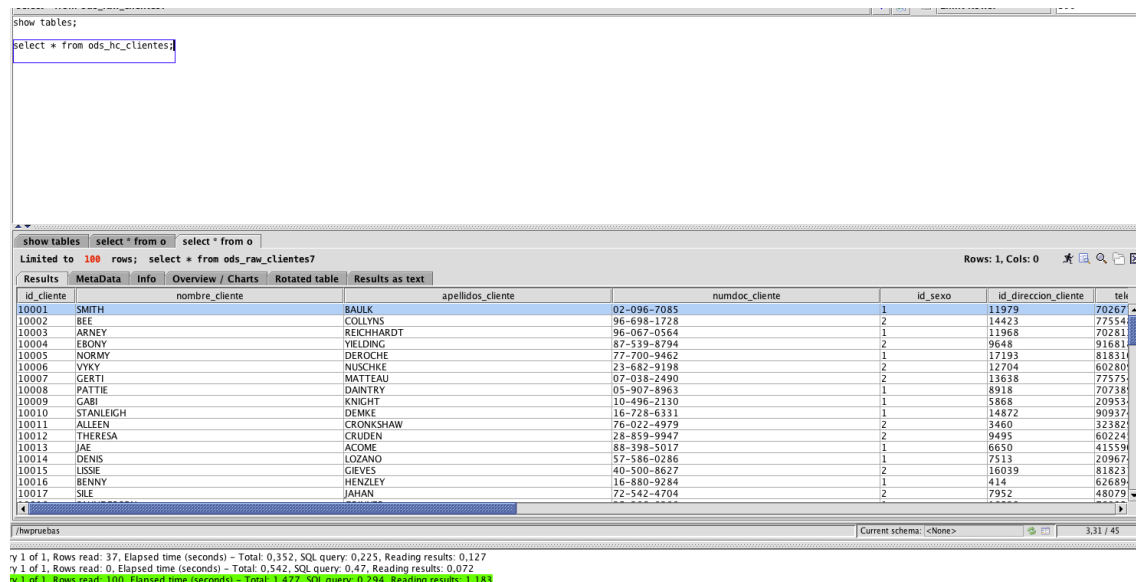


P0/P1:

Para trabajar con en Hadoop con bases de datos relacionales, lo primero que tenemos que hacer es coger los datos de las mismas. Para ello, se ha utilizado SQOOP el cual mediante un comando me ha pasado las tablas de mysql a HDFS:

```
sqoop import --connect jdbc:mysql://52.178.182.123/ODS --driver  
com.mysql.jdbc.Driver --username name --password pwd --table  
ODS_HC_FACTURAS --target-dir /raw
```

Una vez en RAW he examinado la tabla con la CLI para comprobar que estaba correcta. Después he creado las tablas que he puesto como externas/ internas en HIVE (ver archivo con el código en el repositorio señalado de Github). Una vez se estaban todas en HIVE he realizado las queries con SQuirreL (cliente SQL) como se aprecia a continuación:



The screenshot shows the SQuirreL SQL client interface. The top panel displays the command 'select * from ods_raw_clientes;'. The bottom panel shows the query results in a table format. The table has 8 columns: id_cliente, nombre_cliente, apellidos_cliente, numdoc_cliente, id_sexo, id_direccion_cliente, and telc. The results are limited to 100 rows. The status bar at the bottom indicates 'Current schema: <None>' and '3,31 / 45'.

id_cliente	nombre_cliente	apellidos_cliente	numdoc_cliente	id_sexo	id_direccion_cliente	telc
10001	SMITH	BAULK	02-096-7085	1	11979	70267
10002	BEE	COLLYNS	96-698-1728	2	14423	77554
10003	ARNEY	REICHARDT	96-067-0564	1	11968	70281
10004	EBONY	YELDING	87-539-8794	2	9648	91681
10005	NORMY	DEROCHE	77-700-9462	1	17193	81831
10006	VYKY	NUSCHKE	23-682-9198	2	12704	60280
10007	CERTI	MATTEAU	07-038-2490	2	13638	77575
10008	PATTIE	DANTRY	05-907-8963	1	8918	70738
10009	GABI	KNIGHT	10-496-2130	1	5868	20953
10010	STANLEIGH	DEMKE	16-728-6331	1	14872	90937
10011	ALLEEN	CRONSHAW	76-022-4979	2	3460	32382
10012	THERESA	CRUDEN	28-859-9947	2	9495	60224
10013	IAE	ACOME	88-398-5017	1	6650	41559
10014	DENIS	LOZANO	57-586-0286	1	7513	20967
10015	LISSE	GIEVES	40-500-8627	2	16039	81823
10016	BENNY	HENZLEY	16-880-9284	1	414	62689
10017	SILE	JAHAN	72-542-4704	2	7952	48079

P3/P4:

La creación de las tablas se ha realizado desde la consola de comandos, encontrándose el código en el repositorio Github.

P5:

Y una vez que teníamos las tablas externas/internas cargadas, se ha creado una tabla general para resolver los KPIs que hemos descrito al principio de la práctica. Cómo todos los indicadores, además de las respectivas claves tenían que verse a lo largo del tiempo, se han creado cinco tablas con los cuatrimestres que abarcan las facturas. A partir de esta tabla ya es sencillo sacar las informaciones de los KPIs.

Vamos a ver el KPI TOTAL_FACTURADO:

Es un sum(cantidad) que si se realiza mediante subqueries

```
select primer_periodo.cantidad, segundo_periodo.cantidad, tercer_periodo.cantidad, cuarto_periodo.cantidad,quinto_periodo.cantidad
from
(select sum(cantidad) cantidad from general_primer_periodo) as primer_periodo
,(select sum(cantidad) cantidad from general_segundo_periodo) as segundo_periodo
,(select sum(cantidad) cantidad from general_tercer_periodo) as tercer_periodo
,(select sum(cantidad) cantidad from general_cuarto_periodo) as cuarto_periodo
,(select sum(cantidad) cantidad from general_quinto_periodo) as quinto_periodo;
```

select * from o

show tables

select * from o

select * from d

select distinct

select primer_p

select primer_p

select primer_p

select sum(c

describe dataus

select datausa.

select dist

Rows 1: select primer_periodo.cantidad, segundo_periodo.cantidad, tercer_periodo.cantidad, cuarto_periodo.cantidad,quinto_periodo.cantidad

Results

MetaData

Info

Overview / Charts

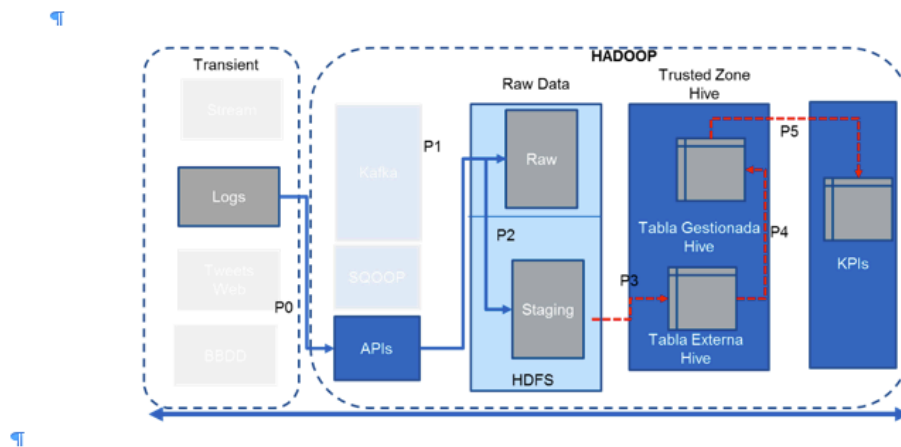
Rotated table

Results as text

cantidad	cantidad	cantidad	cantidad	cantidad
2523420.0	2526623.0	2524779.0	2526618.0	2520109.0

En la imagen anterior se puede observar el TOTAL_FACTURADO para cada uno de los periodos.

Método 2: WEB SCRAPPING



Para realizar esta parte, hemos trabajado con una página web: datausa.io. En ella podemos encontrar bases de datos que son de nuestro interés. Uno de los KPIs que pretendíamos conocer es la relación entre los ingresos medios de un estado y el gasto que realizan los clientes de nuestra que se encuentran en ese estado.

Para ello, mediante Python hemos sacado de la url: http://api.datausa.io/api/csv/?show=geo&sumlevel=state&required=avg_wage el fichero csv, que hemos enviado a Hive. El problema de este csv es el siguiente:


```
[{u'avg_wage': 41185.2, u'geo': u'04000US01', u'year': 2014},
 {u'avg_wage': 51959.1, u'geo': u'04000US02', u'year': 2014},
 {u'avg_wage': 43582.8, u'geo': u'04000US04', u'year': 2014},
 {u'avg_wage': 38570.2, u'geo': u'04000US05', u'year': 2014},
 {u'avg_wage': 51851.5, u'geo': u'04000US06', u'year': 2014},
 {u'avg_wage': 49679.3, u'geo': u'04000US08', u'year': 2014},
 {u'avg_wage': 60977.6, u'geo': u'04000US09', u'year': 2014},
 {u'avg_wage': 47620.6, u'geo': u'04000US10', u'year': 2014},
 {u'avg_wage': 71820.1, u'geo': u'04000US11', u'year': 2014},
 {u'avg_wage': 42217.0, u'geo': u'04000US12', u'year': 2014},
 {u'avg_wage': 44957.1, u'geo': u'04000US13', u'year': 2014},
 {u'avg_wage': 44833.1, u'geo': u'04000US15', u'year': 2014},
 {u'avg_wage': 38357.1, u'geo': u'04000US16', u'year': 2014},
 {u'avg_wage': 49691.5, u'geo': u'04000US17', u'year': 2014},
 {u'avg_wage': 41072.2, u'geo': u'04000US18', u'year': 2014},
 {u'avg_wage': 41975.1, u'geo': u'04000US19', u'year': 2014},
 {u'avg_wage': 43453.9, u'geo': u'04000US20', u'year': 2014},
 {u'avg_wage': 39573.7, u'geo': u'04000US21', u'year': 2014},
 {u'avg_wage': 43441.9, u'geo': u'04000US22', u'year': 2014},
 {u'avg_wage': 40992.2, u'geo': u'04000US23', u'year': 2014},
 {u'avg_wage': 57025.9, u'geo': u'04000US24', u'year': 2014},
 {u'avg_wage': 58299.3, u'geo': u'04000US25', u'year': 2014},
 {u'avg_wage': 43528.2, u'geo': u'04000US26', u'year': 2014},
 {u'avg_wage': 48758.6, u'geo': u'04000US27', u'year': 2014},
 {u'avg_wage': 37638.8, u'geo': u'04000US28', u'year': 2014},
 {u'avg_wage': 42215.5, u'geo': u'04000US29', u'year': 2014},
 {u'avg_wage': 37732.0, u'geo': u'04000US30', u'year': 2014},
 {u'avg_wage': 40511.7, u'geo': u'04000US31', u'year': 2014},
 {u'avg_wage': 41918.2, u'geo': u'04000US32', u'year': 2014},
 {u'avg_wage': 51126.1, u'geo': u'04000US33', u'year': 2014},
 {u'avg_wage': 59716.3, u'geo': u'04000US34', u'year': 2014},
 {u'avg_wage': 38680.4, u'geo': u'04000US35', u'year': 2014},
 {u'avg_wage': 55258.8, u'geo': u'04000US36', u'year': 2014},
 {u'avg_wage': 42602.3, u'geo': u'04000US37', u'year': 2014},
 {u'avg_wage': 44515.7, u'geo': u'04000US38', u'year': 2014},
 {u'avg_wage': 43077.9, u'geo': u'04000US39', u'year': 2014},
 {u'avg_wage': 41001.7, u'geo': u'04000US40', u'year': 2014},
```

Podemos ver el identificador del estado y los ingresos medios, pero nos ha hecho falta una tabla intermedia que estaba en la misma página web y que también hemos subido a Hive:

Locations	Industries	Occupations	Education
Nation	State	County	Public Use Microdata Area
			Metropolitan Statistical Area
			Census Place
ID ↕	Name ↕		
04000US01	Alabama		
04000US02	Alaska		
04000US04	Arizona		
04000US05	Arkansas		
04000US06	California		
04000US08	Colorado		
04000US09	Connecticut		
04000US10	Delaware		
04000US11	District Of Columbia		
04000US12	Florida		
04000US13	Georgia		
04000US15	Hawaii		
04000US16	Idaho		
04000US17	Illinois		
04000US18	Indiana		
04000US19	Iowa		
04000US20	Kansas		

Para saber a qué estado hacía referencia cada ID. El problema de análisis ha ocurrido cuando hemos querido ver cuantos de estos estados estaban en nuestros registros de clientes de las bases relacionales. Para ello, hemos ejecutado la siguiente consulta:

```
select distinct est.de_estado
from ods_raw_ciudades_estados as est
join dimension_estados5 as dimension on (dimension._c1 = est.de_estado);
```

Que nos devuelve lo siguiente:

```
select distinct est.de_estado
from ods_raw_ciudades_estados as est
join dimension_estados5 as dimension on (dimension._c1 = est.de_estado);
```

describe dimens describe dataus select datausa. select distinct show tables s

select * from d select * from

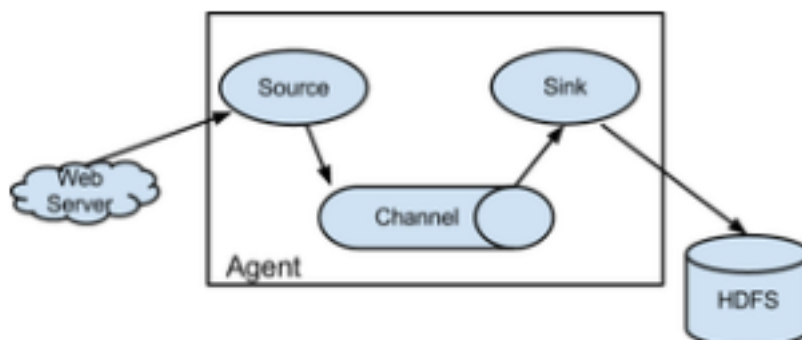
Rows 3; select distinct est.de_estado from ods_raw_ciudades_estados as est join

Results MetaData Info Overview / Charts Rotated table Results as text

de_estado
ARIZONA
CALIFORNIA
NEVADA

Lo cual indica que solo tenemos clientes de tres estados diferentes lo que convierte este KPI en no apto para sacar conclusiones aceptables.

Método 3: FLUME



En un principio estaba pensada la realización de un flujo de trabajo de donde se pudieran coger mensajes relacionados con unas keywords determinadas, pero debido a que la sandbox de Hortonworks no me permite modificar una serie de parámetros internos dejamos esta parte del proyecto para una siguiente etapa.

Sin embargo, se reflejará a nivel teórico las distintas formas encontradas para llevar a cabo esto.