

San Francisco

Open Data



Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Alejandro Reina Reina

Tutor/es:
Alejandro Mate Morga y Juan Carlos Trujillo Mondéjar

Enero 2018

Trabajo de Fin de Grado

San Francisco Open Data

Ingeniería informática

Resumen

En este TFG se pretende realizar la integración de datos desde distintas fuentes, procesando y transformando los datos en información de la cual se nutrirá un sistema BI con tecnología Big Data. Dicho framework permitirá la computación en clúster mejorando la velocidad con respecto a otras herramientas tradicionales, permitiendo la exploración de datos ad hoc y una aplicación más sencilla de algoritmos machine learning y técnicas de minería de datos.

Motivación, justificación y objetivo general

Todo comienza a principios del 2017 cuando empiezo a conocer el mundo del Business Intelligence, gracias a la asignatura Ingeniería de Negocio y Gestión de Procesos impartida por el profesor Juan Carlos Trujillo de la Universidad de Alicante, un curso impartido en la UA por Teralco, y el curso de Google “Marketing Digital”, que aunque a priori pueda parecer que no tiene nada que ver, pero desde un perfil técnico como es el mío, pude ver desde el punto de vista de estudiantes de publicidad, y profesores especializados en la materia, como hacían uso de los datos y lo importante que eran para ellos para realizar buenas campañas de marketing, ya que de ello dependía un negocio, pero aun así, y a pesar de utilizar Google AdWords junto con otras herramientas, organizaban sus campañas y analizaban los resultados un poco a “ojo”, métodos poco fiables que deja mucho a la subjetividad, además de que había una gran cantidad de información que no era analizada y por ultimo un curso impartido en la universidad de Alicante por la escuela de verano Rafael Altamira, me permitieron ver desde distintos puntos de vista de que es BI, y la importancia de los datos que tenemos, que implica hacer un análisis de los datos, y la importancia que tiene esto, para cualquier negocio o institución. Puede parecer que el curso de Marketing Digital no tiene relación directa con los otros, pero precisamente en este curso, fue interesante ya que desde un perfil técnico como es el de los ingenieros, pude conocer el mundo de la publicidad y el marketing, donde a través de herramientas de Google como Google AdWords organizaban sus campañas y analizaban los resultados de redes sociales a “ojo”, para hacer mejores campañas en un futuro, pero ese método no era 100% fiable, dejaba mucho a la subjetividad, y no analizábamos empíricamente los datos que tenemos, además de que había una gran cantidad de información que no era analizada. De hecho, esto fue motivo de un TFG de mi compañero de carrera y amigo, Héctor Barrachina, donde monta un sistema BI tradicional nutriéndose de los mensajes de una página de Facebook y analizándolos, aunque ya hacia plantearnos otros tipos de preguntas como si la página que estas analizando es lo suficientemente grande, y tenemos en marcha el sistema durante cinco años, ¿cuál sería la cantidad de información que tendríamos para analizar? ¿Podremos analizarla? ¿Y si lo tenemos en marcha durante diez años? El conjunto de lo anteriormente expuesto, hizo que me decantase por esta rama de la ingeniería informática.

Aunque de manera práctica, hasta la fecha del desarrollo de este proyecto, solo había visto BI tradicional, y desconocía si era igual teóricamente o tecnológicamente para el mundo Big Data, fue un aliciente para mí para entrar en este mundo del análisis de la información, quizá menos valorado a principios de este siglo, pero hoy en día, nadie duda de la importancia de los datos en cualquier organización.

Para mi este TFG fue propuesto como un reto personal, ya que ha sido entrar de lleno en herramientas y tecnologías y, en definitiva, a un mundo que desconocía completamente, ya que, a lo largo de la carrera, BigData solo lo escuchabas entre compañeros, pero nadie sabía bien que era exactamente esto de Big Data.

Además, he podido hacer un contacto directo con lo que es un trabajo de investigación, que difiere mucho en una aplicación sobre marcos conceptuales y metodologías bien definidas como podría ser el desarrollo de una web, quizás haciendo uso de la definición de que es un ingeniero,

el cual con lo que sabe y tiene, debe dar solución a un problema específico, que en mi caso han sido mi esfuerzo y propia motivación por este mundo de los datos, al cual solo una asignatura hace referencia directa a dicha área.

Por lo que, a grandes rasgos, el objetivo de este proyecto, es definir una metodología y un marco de trabajo para implementar un BI sobre arquitectura Big Data, con el fin de suplir las limitaciones que tiene el BI tradicional, en rendimiento y escalabilidad.

Agradecimientos

Los agradecimientos de este trabajo están dedicados a:

- Mi madre, ejemplo de super-mujer, luchadora, fuerte, valiente, que se crece ante la adversidad y nunca se rinde, gracias por tus sacrificios y esfuerzos para que yo y mis hermanos pudiésemos formarnos y estudiar, sin ti, ninguno de los tres estaríamos donde estaríamos donde estamos.
- Mi hermano mayor Antonio, gracias por estar siempre ahí, y por ser también un ejemplo para mí, por tu culpa acabe metido en esto de la informática, aunque acabemos especializándonos en ramas distintas, gracias a ti descubrí una de mis pasiones.
- Mi hermana pequeña Claudia, que, aunque seas la enana, eres una persona de la que se puede aprender mucho, y ojalá hubiese más profesores de educación infantil apasionados y entregados con los niños como tú.
- A mis compañeros y amigos de rama Buba, Héctor, Jorge Espinosa y Jorge Cantó (“los jorges”), por enseñarme, aunque fuese al final de la carrera, lo que con un buen ambiente de equipo y una buena filosofía de trabajo se puede conseguir, parte de los resultados de este proyecto, se debe a vosotros, y ha sido un placer haberme encontrado con vosotros.
- A mi compañero Yeredai, y su Hermana Estefi, porque a pesar de ser “canarios” son grandes personas y amigos, con la que he compartido buenos y grandes momentos en estos años. Me alegro de que nuestras vidas se cruzasen y aunque hayáis vuelto a vuestra islilla nos volveremos a encontrar.
- A los tutores de este proyecto, Alejandro Mate, y Juan Carlos Trujillo por permitirme aprender e introducirme en el mundo del BI y Big Data y guiarme cuando lo necesitaba.
- A mis grandes amigas, Irene (la hierbas y mami), Eva (la bruja), Yoli (la Yoli) e Izaskun (la vasca), gracias por estar ahí siempre que lo he necesitado, aunque no abriese la boca y aguantarme. No hay nadie como vosotras. Os quiero jineteras.
- A mis amigos Farman, Salvatore y Darkness.
- A todo un Campeón y mi amigo Andrew Jefferson Rodríguez, con una gran pasión por los coches, esfuérzate y sé que algún día serás un gran ingeniero automovilístico, o cualquier cosa que te propongas.
- A otro gran amigo, Jorge, por esas entretenidas charlas de todo y nada.
- Y en general a todas las personas con las que de una manera u otra han influido para ser quien soy hoy en día.

Citas

Y hubo un día que comprendí que mi único enemigo era yo mismo. Fue ese día cuando empecé a ganar todas las batallas.

Anónimo

Nunca he conocido a nadie tan ignorante del que no pudiera aprender algo.

Buda

No pienses que no pasa nada, simplemente porque no ves tu crecimiento... las grandes cosas crecen en silencio.

Buda

No insistas en el pasado, no sueñes en el futuro, concentra tu mente en el momento presente.

Buda

El Big data es como el sexo en la adolescencia: todo el mundo habla de ello, nadie sabe realmente cómo hacerlo, todos piensan que los demás lo están haciendo, así que todos dicen que también lo hacen..."

Dan Ariely.

Con todo su poder, Big Data no deja de ser una deidad menor al servicio del dios supremo del beneficio económico.

Rafael Caballero.

Es un error capital el teorizar antes de poseer datos. Insensiblemente, uno comienza a deformar los hechos para hacerlos encajar en las teorías en lugar de encajar las teorías en los hechos.

Sherlock Holmes.

Índice de contenidos

Resumen.....	2
Motivación, justificación y objetivo general	3
Agradecimientos	5
Citas.....	6
Índice de figuras	11
Índice de tablas	15
1. Introducción	16
2. Metodología	17
3. Objetivos	20
4. Estado del arte.	21
4.1. Historia de Inteligencia de Negocio (Business Intelligence)	21
4.2. ¿Qué es Business Intelligence?	22
4.3. Arquitectura de un sistema BI.....	23
4.3.1. Fuentes de datos	24
4.3.2. Procesos ETL.....	24
4.3.3. Almacén de datos o Datawarehouse	25
4.3.4. Servidor de consultas	29
4.3.5. Visualización	30
4.4. Problemática del BI tradicional.	31
4.5. Inicios de BigData	31
4.6. ¿Qué es BigData?	32
4.7. Landscape de Big Data.	34
4.8. Data Lake	38
4.9. Hadoop.....	40
4.9.1. HDFS (Hadoop Distributed File System).....	41
4.9.2. MapReduce	43
4.9.3. HBase.....	44
4.9.4. Características de Hadoop.....	44
4.10. Cassandra	45
4.11. HBase vs Cassandra	46
4.12. MongoDB.....	47

4.13.	Hive.....	49
4.13.1.	Modelo de datos	49
4.14.	Spark.....	50
4.14.1.	Características de Spark	51
4.14.2.	Librerías importantes en Spark	53
4.14.3.	Funcionalidades DAG y RDD.....	55
4.14.4.	Spark 2.X.....	56
4.14.5.	Modelo de programación.....	57
4.14.6.	Tipos de transformaciones.....	58
4.14.7.	Lenguaje de programación a utilizar en Spark	60
4.15.	Spark vs Hadoop.....	64
4.16.	Kylin	65
4.17.	Zeppelin.....	66
4.18.	Docker	68
4.18.1.	Docker vs Máquinas virtuales	68
5.	Diseño.....	71
5.1.	Diseño de la arquitectura	71
5.2.	Despliegue de la arquitectura	73
5.2.1.	Java	73
5.2.2.	Scala.....	75
5.2.3.	Spark.....	76
5.2.4.	HBase.....	78
5.2.5.	Hive.....	80
5.2.6.	Zeppelin.....	86
5.2.7.	Kylin	89
5.2.8.	Docker	92
5.2.9.	Configuración adicional: Python	106
5.3.	Diseño de pruebas de rendimiento. Scala vs Python.....	108
5.3.1.	Metodos Scala	109
5.3.2.	Metodos Python.....	110
5.3.3.	Resultados de las pruebas.....	111
5.3.4.	Conclusiones.....	112
5.4.	Diseño del Datamart	113

5.4.1.	Exploración de los datos.....	113
5.4.2.	Definición de preguntas analíticas	117
5.4.3.	Diagrama de jerarquías	118
5.4.4.	Esquema estrella	119
5.5.	Diseño de ETL	120
5.5.1.	Lectura de ficheros	120
5.5.2.	Guardar un DataFrame en una tabla Hive	121
5.5.3.	Diseño de la dimension Horas.....	122
5.5.4.	Diseño de la dimensión Fecha.....	123
5.5.5.	Diseño de dimensiones y tabla de hechos	125
5.6.	Diseño del cubo MOLAP.....	128
5.6.1.	Carga de las tablas de Hive.....	129
5.6.2.	Diseño del modelo Multidimensional en Kylin	131
5.6.3.	Diseño del cubo en Kylin	135
5.6.4.	Compilar cubo	143
6.	Visualización.....	145
6.1.	Consulta del cubo con Kylin	145
6.2.	Consulta del cubo con Zeppelin	146
6.3.	Herramientas de cuadros de mando con Kylin	147
6.3.1.	Driver ODBC.....	147
6.3.2.	Conectar PowerBI a Kylin	149
6.4.	Police Department Incident	151
6.4.1.	Incidentes por Barrio.....	152
6.4.2.	Incidentes más frecuentes	152
6.4.3.	¿Cuántos robos se saldan con la detención de alguien? ¿En cuántos no ocurre nada?	153
6.4.4.	¿Cuándo es más frecuente en un día que ocurra un incidente?	153
6.4.5.	Meses del año que ocurren con mayor frecuencia un incidente.....	154
6.4.6.	¿Todos los tipos de robos ocurren por igual o su frecuencia se altera en el tiempo?	155
6.5.	Fire Department Calls for Service.....	155
6.5.1.	¿Se llama por igual de todos los vecindarios?.....	155
6.5.2.	¿Tipos de llamada tienen la misma proporción?	157
6.5.3.	¿Tiempo medio de respuesta por tipo de llamada?	157

6.5.4.	Evolución de las llamadas por año	158
6.5.5.	Evolución de las llamadas por tipo y meses del año	158
6.5.6.	¿Los batallones de bomberos responden el mismo número de llamadas?	160
6.5.7.	¿Hay batallones especializados en tipos específicos de llamadas?	160
7.	Conclusiones.....	162
8.	Posibles líneas de investigación futura.	164
8.1.	Herramienta de visualización de datos	164
8.2.	Clúster Hadoop.....	164
8.3.	Machine learning.....	164
8.4.	Ampliación del sistema BI	164
	Referencias.....	165

Índice de figuras

Figura 1. Tareas Backlog, In process, Done e investigación – Alejandro Reina Reina.....	17
Figura 2. Tareas Instalación e integración, Test y BI San Francisco – Alejandro Reina Reina.....	18
Figura 3. Entorno Virtual Box – Alejandro Reina Reina.....	19
Figura 4. Arquitectura BI – Apuntes INGP Juan Carlos Trujillo Mondéjar.....	24
Figura 5. Jerarquía de las dimensiones – Apuntes INGP Juan Carlos Trujillo.....	27
Figura 6. Esquema Estrella www.dataprix.com	28
Figura 7. Cubo multidimensional. www.dataprix.com	29
Figura 8. Dashboard PowerBI. Google Images	30
Figura 9. Las 3V's de BigData. www.blog.sqlauthority.com	33
Figura 10. Landscape BigData – Arquitectura. mattturck.com/bigdata2017	34
Figura 11. Landscape BigData – Analítica. mattturck.com/bigdata2017	35
Figura 12. Landscape BigData – Aplicaciones en industrias y empresas. mattturck.com/bigdata2017	36
Figura 13. Landscape BigData – infraestructura/analítica cruzada. mattturck.com/bigdata	36
Figura 14. Landscape BigData – OpenSource. mattturck.com/bigdata2017	37
Figura 15. Landscape BigData – Fuentes de datos y APIs. mattturck.com/bigdata2017/	37
Figura 16. Ecosistema Hadoop. Apache.org	41
Figura 17. Planes de ejecución para un Join. Momentotic.com	43
Figura 18. MapReduce count Word. cs.calvin.edu	44
Figura 19. Spark vs Hadoop. spark.apache.org	51
Figura 20. Librerías Spark. BBVAopen4u	54
Figura 21. DAG (Grafo Acíclico Dirigido). Geekytheory.com	55
Figura 22. Transformaciones de un RDD. Geekytheory.com	58
Figura 23. Tipos de transformaciones: Narrow y Wide. Geekytheory.com	59
Figura 24. Reordenamiento de los datos de un RDD. Geekytheory.com	60
Figura 25. Scala vs Python en Spark. Dezyre.com	62
Figura 26. The most popular Language for machine learning and data science. IBM	63
Figura 27. Grafica ofertas publicadas para el dominio DataScience, DataScientist, MachineLearning. Javier García Paredes	63
Figura 28. Arquitectura Kylin. Paper Online Analytical Processing on Hadoop using Apache Kylin	66
Figura 29. Lenguaje de Backend. Zeppelin.apache.org	67
Figura 30. Arquitectura VMs vs Docker. Docs.microsoft.com	69
Figura 31. Arquitectura VMs vs Docker. Ángel Luis Mula	70
Figura 32. Arquitectura BI tradicional. Fernando Berzal.....	71
Figura 33. Arquitectura BI con tecnología BigData. Alejandro Reina.....	72
Figura 34. Comprobar versión de Java (no instalado). Alejandro Reina	73
Figura 35. Añadir repositorios Java. Alejandro Reina	74
Figura 36. Actualización del sistema y dependencias. Alejandro Reina	74
Figura 37. Instalar Java8. Alejandro Reina	74
Figura 38. Abrir bashrc. Alejandro Reina	74

Figura 39. Añadir Path de Java8. Alejandro Reina	75
Figura 40. Comprobar versión de Java(Instalado). Alejandro Reina	75
Figura 41. Descarga de Scala. Alejandro Reina	75
Figura 42. Descomprimir Scala. Alejandro Reina	75
Figura 43. Renombrar y mover Scala. Alejandro Reina.....	76
Figura 44. Path Scala. Alejandro Reina.....	76
Figura 45. Scala version. Alejandro Reina	76
Figura 46. Path Spark. Alejandro Reina.....	76
Figura 47. Path Java en Spark-env.sh. Alejandro Reina	77
Figura 48. Spark Shell. Alejandro Reina.....	77
Figura 49. Spark webUI. Alejandro Reina.....	78
Figura 50. Hbase-env.sh path java. Alejandro Reina.....	79
Figura 51. Path HBase y Zookeeper. Alejandro Reina	79
Figura 52. Path HBase en bashrc. Alejandro Reina	79
Figura 53. Iniciar HBase. Alejandro Reina	79
Figura 54. Proceso HMaster. Alejandro Reina	80
Figura 55. HBase webUI. Alejandro Reina.....	80
Figura 56. Path Hive en bashrc. Alejandro Reina	80
Figura 57. Path Java en Hadoop-env.sh. Alejandro Reina.....	81
Figura 58. Path de Hadoop en bashrc. Alejandro Reina.....	81
Figura 59. Path de Hadoop en Hive-conf.sh. Alejandro Reina	81
Figura 60. Instalar MySQL. Alejandro Reina	81
Figura 61. Librerías Java para MySQL. Alejandro Reina	81
Figura 62. Conector de Java para Hive. Alejandro Reina	82
Figura 63. Ejecutar MySQL. Alejandro Reina.....	82
Figura 64. MySQL service start. Alejandro Reina	82
Figura 65. Usuario para Hive en MySQL. Alejandro Reina	82
Figura 66. Crear fichero hive-site.xml. Alejandro Reina.....	83
Figura 67. Hive upgradeSchema. Alejandro Reina	83
Figura 68. Script to UpgradeSchema. Alejandro Reina	84
Figura 69. Jars de Hive y MySQL para Spark. Alejandro Reina	85
Figura 70. Hive Shell. Alejandro Reina	85
Figura 71. Ejecutar hiveserver. Alejandro Reina	85
Figura 72. Hive webUI. Alejandro Reina.....	86
Figura 73. Anonymous allowed. Alejandro Reina	87
Figura 74. Zeppelin start. Alejandro Reina	87
Figura 75. Zeppelin webUI. Alejandro Reina	87
Figura 76. Intérpretes de Zeppelin. Alejandro Reina	88
Figura 77. Parámetros para el intérprete de Hive. Alejandro Reina	89
Figura 78. Kylin para HBase 1.X. Alejandro Reina	90
Figura 79. Descomprimir Kylin. Alejandro Reina.....	90
Figura 80. Guía oficial para instalar Kylin. Alejandro Reina	90
Figura 81. Scripts de comprobación de dependencias. Alejandro Reina.....	91
Figura 82. PATH adicionales de Kylin. Alejandro Reina.....	91
Figura 83. Iniciando Kylin. Alejandro Reina.....	91

Figura 84. Interfaz Kylin. Alejandro Reina	92
Figura 85. Actualizar sistema. Alejandro Reina.....	93
Figura 86. Añadir clave del servidor al repositorio. Alejandro Reina.....	93
Figura 87. Actualizar repositorio y políticas. Alejandro Reina	94
Figura 88. Instalar Docker. Alejandro Reina.....	94
Figura 89. Instalar Docker. Alejandro Reina.....	94
Figura 90. Comando docker version. Alejandro Reina	95
Figura 91. Mysql-hive.sql. Alejandro Reina	102
Figura 92. Init.sh. Alejandro Reina	103
Figura 93. Docker build. Alejandro Reina.....	104
Figura 94. Docker images. Alejandro Reina	104
Figura 95. Docker ps sin contenedores. Alejandro Reina	105
Figura 96. Docker run. Alejandro Reina	105
Figura 97. Docker run. Alejandro Reina	105
Figura 98. Docker attach. Alejandro Reina.....	105
Figura 99. Ejecución jps dentro del flujo del contenedor. Alejandro Reina.....	106
Figura 100. Acceso desde el host al container. Alejandro Reina	106
Figura 101. Descarga de Python. Alejandro Reina	107
Figura 102. Path Python. Alejandro Reina	107
Figura 103. Spark-env.sh Python. Alejandro Reina	107
Figura 104. Zeppelin-env.sh. Alejandro Reina.....	108
Figura 105. Zeppelin PySpark interpreter. Alejandro Reina.....	108
Figura 106. Datos Ejemplo. Alejandro Reina	108
Figura 107. Scala vs Python - Tiempos. Alejandro Reina	112
Figura 108. Scala vs Python - Tiempos. Alejandro Reina	112
Figura 109. Police_Department_Incidents.csv - data.sfgov.org	113
Figura 110. Fire departement calls for service.csv - data.sfgov.org.....	115
Figura 111. Diagrama de jerarquías Police Incidents – Alejandro Reina Reina	118
Figura 112. Diagrama de jerarquías Police Incidents – Alejandro Reina Reina	118
Figura 113. Diagrama estrella Police Incidents – Alejandro Reina Reina.....	119
Figura 114. Diagrama estrella Fire Departments Calls for Service – Alejandro Reina Reina	120
Figura 115. Datos dimension Horas – Alejandro Reina Reina.....	123
Figura 116. Datos dimension Fechas – Alejandro Reina Reina	125
Figura 117. Datos dimension fire_dim_location – Alejandro Reina Reina.....	126
Figura 118. Datos dimension fire_facttable – Alejandro Reina Reina	128
Figura 119. Nuevo proyecto Kylin – Alejandro Reina Reina.....	129
Figura 120. Cargar Tablas Hive – Alejandro Reina Reina	129
Figura 121. Seleccionar Tablas Hive – Alejandro Reina Reina.....	130
Figura 122. Tablas cargadas correctamente en Kylin – Alejandro Reina Reina	130
Figura 123. Tablas cargadas correctamente en Kylin – Alejandro Reina Reina	131
Figura 124. New Model Kylin – Alejandro Reina Reina.....	131
Figura 125. Model Info Kylin – Alejandro Reina Reina.....	132
Figura 126. Añadir Tabla de hechos y lookup tables – Alejandro Reina Reina	132
Figura 127. Añadir lookup tables – Alejandro Reina Reina	132
Figura 128. Paso Data Model – Alejandro Reina Reina.....	133

Figura 129. Seleccionamos columnas de dimension – Alejandro Reina Reina	133
Figura 130. Seleccionar medidas – Alejandro Reina Reina	134
Figura 131. Partición de tabla – Alejandro Reina Reina	134
Figura 132. Modelo creado correctamente – Alejandro Reina Reina.....	134
Figura 133. Editar Modelo – Alejandro Reina Reina	135
Figura 134. Visualización del modelo – Alejandro Reina Reina	135
Figura 135. Nuevo Cubo – Alejandro Reina Reina	136
Figura 136. Info del cubo – Alejandro Reina Reina	136
Figura 137. Add Dimensions – Alejandro Reina Reina	137
Figura 138. Add dimensions en la tabla de hechos – Alejandro Reina Reina	137
Figura 139. Add dimensions en lookup tables – Alejandro Reina Reina.....	138
Figura 140. Add dimensions resumen – Alejandro Reina Reina	138
Figura 141. Añadir medidas al cubo – Alejandro Reina Reina	138
Figura 142. Cubo incremental – Alejandro Reina Reina.....	139
Figura 143. Grupos de agregación de las dimensiones del cubo – Alejandro Reina Reina.....	140
Figura 144. Grupos de agregación de las dimensiones del cubo – Alejandro Reina Reina.....	141
Figura 145. Motor de construcción del cubo y otras opciones – Alejandro Reina Reina	141
Figura 146. Resumen de construcción del cubo – Alejandro Reina Reina.....	142
Figura 147. Resumen de construcción del cubo – Alejandro Reina Reina.....	142
Figura 148. Visor del cubo – Alejandro Reina Reina	142
Figura 149. Acciones del cubo– Alejandro Reina Reina	143
Figura 150. Progreso de compilación del cubo – Alejandro Reina Reina.....	143
Figura 151. Error en el step Hive Cleanup – Alejandro Reina Reina	144
Figura 152. Error en el step Hive Cleanup – Alejandro Reina Reina	145
Figura 153. Propiedades del interprete Kylin en Zeppelin – Alejandro Reina Reina	146
Figura 154. Error en el step Hive Cleanup – Alejandro Reina Reina	147
Figura 155. Agregar DSN de sistema – Alejandro Reina Reina.....	147
Figura 156. Seleccionar driver ODBC de Kylin – Alejandro Reina Reina	148
Figura 157. Apache Kylin DSN Configuration – Alejandro Reina Reina.....	148
Figura 158. Seleccionar driver ODBC en PowerBI – Alejandro Reina Reina.....	149
Figura 159. Seleccionar DSN en PowerBI – Alejandro Reina Reina.....	149
Figura 160. Seleccionar tablas para cargar en PowerBI – Alejandro Reina Reina.....	150
Figura 161. Nueva relación en PowerBI – Alejandro Reina Reina.....	150
Figura 162. Relaciones de tablas en PowerBI – Alejandro Reina Reina	151
Figura 163. Consulta con PowerBI a Kylin – Alejandro Reina Reina	151
Figura 164. Grafica Número de incidentes por barrio – Alejandro Reina Reina.....	152
Figura 165. Gráfica Incidentes más frecuentes – Alejandro Reina Reina	152
Figura 166. Gráfica Resoluciones de Robos/hurto – Alejandro Reina Reina	153
Figura 167. Gráfica Incidentes por día de la semana – Alejandro Reina Reina.....	153
Figura 168. Gráfica Frecuencia de incidentes por día de la semana – Alejandro Reina Reina .	154
Figura 169. Gráfica Frecuencia de incidentes por meses – Alejandro Reina Reina	154
Figura 170. Gráfica Incidentes por año – Alejandro Reina Reina.....	155
Figura 171. Gráfica Llamadas por barrio – Alejandro Reina Reina.....	155
Figura 172. Gráfica Llamadas por dirección – Alejandro Reina Reina.....	156
Figura 173. Gráfica Llamadas por dirección – Alejandro Reina Reina.....	156

Figura 174. Gráfica Tipos de llamada – Alejandro Reina Reina..... 157

Figura 175. Gráfica Tiempo medio de Respuesta – Alejandro Reina Reina 157

Figura 176. Gráfica Evolución de las llamadas por año – Alejandro Reina Reina 158

Figura 177. Gráfica Evolución de las llamadas por meses – Alejandro Reina Reina 158

Figura 178. Gráfica Evolución de las llamadas de suicidio por meses – Alejandro Reina Reina..... 158

Figura 179. Gráfica Evolución de las llamadas de asalto por meses – Alejandro Reina Reina . 159

Figura 180. Gráfica Evolución de las llamadas de drogas y narcóticos por meses – Alejandro Reina Reina..... 159

Figura 181. Gráfica Número de llamadas por batallones – Alejandro Reina Reina 160

Figura 182. Gráfica Llamadas más frecuentes por batallones – Alejandro Reina Reina..... 160

Figura 183. Gráfica Tipos de llamadas por batallón – Alejandro Reina Reina 161

Figura 184. Clúster NASA – Alejandro Reina Reina 163

Índice de tablas

Tabla 1. Ejemplo valores de una tabla de hechos – Alejandro Reina Reina 27

Tabla 2. Ejemplo tabla de dimensión – Alejandro Reina..... 28

Tabla 3. Principales comandos Docker – Alejandro Reina Reina 95

Tabla 4. Principales comandos de un script Dockerfile – Alejandro Reina Reina 96

1. Introducción

Uno de los puntos clave en la evolución y desarrollo del ser humano ha sido su capacidad para observar, reunir información y aprender. Hacia el año 3000 a.C. se tienen datos de que los babilonios usaban pequeñas tablillas de arcilla para recopilar datos en tablas sobre la producción agrícola y de los géneros vendidos o cambiados mediante trueque. (1) Estos datos servían para analizar que ocurría, y aprender. A lo largo de la historia, estos métodos han ido desarrollándose y perfeccionándose, junto con la estadística. Y ya en el siglo XX con la era de la informática comenzará lo que en el siglo XXI se conocerá como la revolución de los datos.

Hoy en día nadie puede negar la importancia de tener los datos, todos utilizamos datos, desde las empresas más sofisticadas para entender mejor que ocurre en su empresa y aprender continuamente, a unos padres de familia que comparan cual es la mejor guardería para su hijo, o un niño que piensa que juguete prefiere.

Todos tienen en común que recogen datos según sus intereses, ya sean efectividad de campañas de marketing o como de productiva es su empresa, cual tiene las mejores referencias y si esta cerca de casa o el trabajo, o que accesorios lleva el juguete que le gusten más. Todos reunimos datos de una manera u otra para aprender y tomar la mejor decisión posible.

Hoy en día, en la segunda década del siglo XXI, la informática juega un papel clave en el análisis y tratamiento de los datos, ya que permite analizar una cantidad de información que antes era impensable, y cuantos más datos tengamos (siempre y cuando tengan valor), mejor información podemos obtener y más podemos aprender.

Por ello, este proyecto tratará de como montar un sistema para poder tratar y analizar los datos a gran escala, es decir, un sistema orientado al análisis masivo de estos datos, aplicando las últimas tecnologías existentes, orientado a grandes empresas y organizaciones.

2. Metodología

Este proyecto ha sido un proyecto de investigación, el cual no solo la materia si no el entorno que la rodea era completamente desconocido para mí.

Para enfrentarme a el problema que en este proyecto se plantea, emplee una metodología , por lo que la metodología empleada para resolver este problema ha sido Kanban, ideada por Ohno, su nombre en japonés significa “Kan” visible o visual, y “Ban”, que significa tarjeta o tablón, donde apoyándome en Trello, pagina que ofrece un servicio de tableros basado en la metodología Kanban, fui añadiendo de manera iterativa una serie de tareas o actividades que la realización de dichas actividades permitía alcanzar los objetivos del proyecto.

La definición de estas tareas ha sido estudiar en un primer momento el amplio Landscape de tecnologías que existen en el mundo de Big Data, permitiéndome tener una visión global, y después ir profundizando más en tecnologías más concretas, entendiendo la finalidad de dichas tecnologías y sus casos de uso, ya que no existe ninguna metodología ni ningún estándar que nos indique que cuando y como utilizar ninguna de estas herramientas.

Una vez estudiado y aprendido sobre dichas tecnologías, planifique las actividades que permitiría instalar e integrar con el resto de tecnologías y cómo utilizar dichas herramientas.

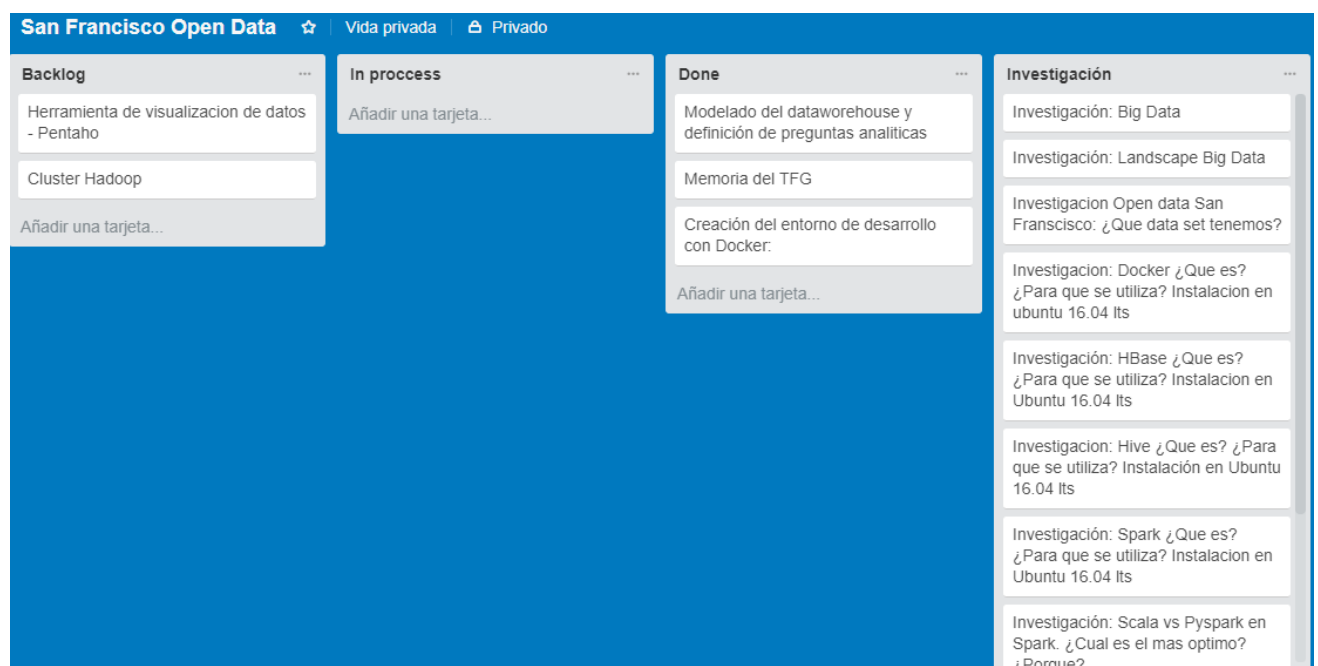


Figura 1. Tareas Backlog, In process, Done e investigación – Alejandro Reina Reina

Instalación e Integración	Test	BI San Francisco
Instalación Ubuntu 16.04 lts	Ejecutar Spark a través de Zeppelin	diseño conceptual
Instalación HBase	Ejecutar Hive a través de Zeppelin	esquema estrella
Instalar e integrar Spark	Pruebas de rendimiento: Scala vs Pyspark	ficheros reducidos de prueba
Instalación e integración de Hive	Carga de datos en Spark	ETL Police Department Incidents
Instalación e integración de Zeppelin	Escritura de datos en Hive a través de Spark	ETL Fire Department calls for service
Instalación e integración de Hive con MySQL	Cubo Kylin	Diseño del cubo en Kylin
Instalación e integración de MySQL	Consultar Cubo	Carga con los ficheros originales
Instalación e integración de Kylin	Ejecutar Kylin a través de Zeppelin	Carga del cubo en Kylin con los ficheros originales
Añadir una tarjeta...	Prueba de las tecnologías con docker	Graficas de las preguntas analíticas
	Añadir una tarjeta...	Exploración de los datos y diseño de preguntas analíticas
		Añadir una tarjeta...

Figura 2. Tareas Instalación e integración, Test y BI San Francisco – Alejandro Reina Reina

Cabe destacar, que la columna Backlog, ha sido utilizada como aquellas tareas pendientes, que todavía no se han comenzado, dando lugar también a tareas de futuras ampliaciones, In progress para aquellas tareas que se estaban realizando, destacando que el desarrollo del script en Docker y de la memoria ha sido un proceso que se ha desarrollado a lo largo de la realización de este proyecto. El resto de columnas se han utilizado para las distintas partes de las que esta compuesto este proyecto con un alto nivel de abstracción, donde se han ido incluyendo tareas más específicas, aunque manteniendo el nivel de abstracción.

Por otro lado, he empleado metodologías de desarrollo colaborativo y control de versiones, utilizando la plataforma GitHub, que, aunque este proyecto ha sido desarrollado individualmente, si ha permitido poder disponer de un repositorio donde alojar el código desarrollado, permitiéndome compartir dicho código entre los distintos entornos desplegados para el desarrollo del proyecto, en la siguiente URL, tenemos el repositorio Git empleado:

<https://github.com/konu90/SanFrancisco>

Y aunque quizás pueda no ser considerado como metodología, me he apoyado en virtual box y sus instantáneas para controlar las versiones del despliegue del entorno, con el fin de poder revertir la instalación en cualquier momento, o volver a un punto anterior a él, sin un alto coste, ya que desconocía si aunque hubiese integrado una tecnología con otra, iba a poder seguir integrándolo con las otras tecnologías, como de hecho me paso con Kylin, que tras haber instalado e integrado Hive con motor Derby, con Spark, HBase, Zeppelin, etc., fue imposible hacerlo funcionar, y tuve que revertir dicha instalación para instalar Hive con motor MySQL, y volver a integrar las distintas tecnologías entre sí, como veremos más adelante en la memoria de este proyecto. A continuación, podemos ver una imagen del entorno en Virtual Box.

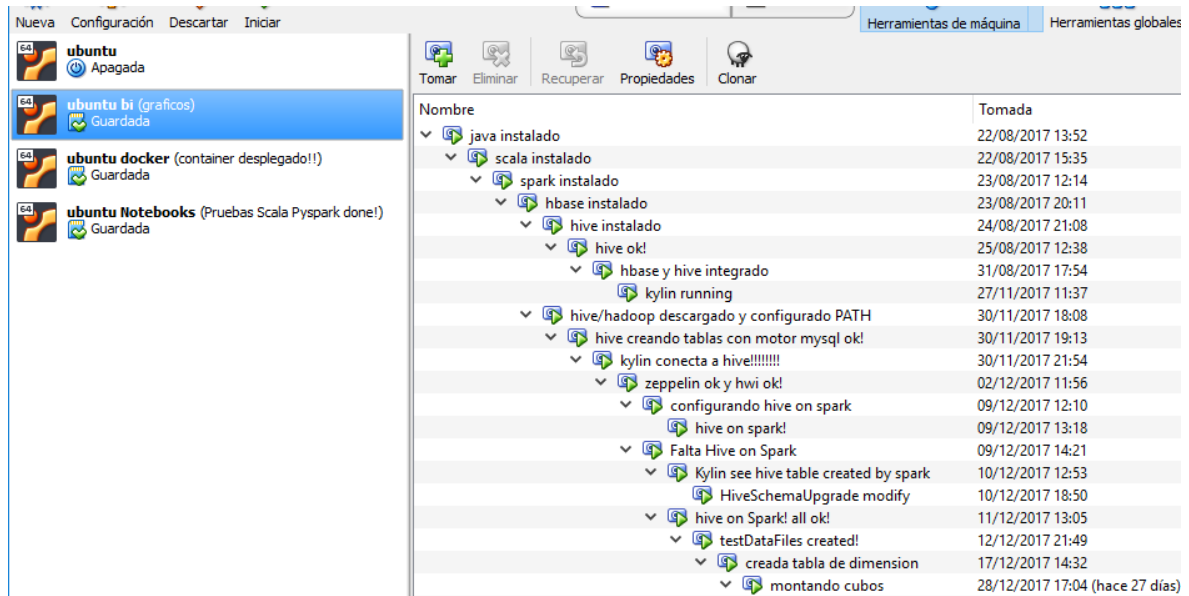


Figura 3. Entorno Virtual Box – Alejandro Reina Reina

3. Objetivos

El objetivo principal de este proyecto, es realizar la integración de datos de distintas fuentes, procesando y transformando los datos en información de la cual se nutrirá un sistema BI con tecnología Big Data, resolviendo la problemática de los sistemas BI tradicionales. La definición de dicho framework permitirá la computación en clúster mejorando la velocidad con respecto a otras herramientas tradicionales, permitiendo la exploración ad hoc y una aplicación más sencilla de algoritmos machine learning y técnicas de minerías de datos.

Por ello, definiremos un framework sobre tecnología BigData para realizar dicha computación en la nube o en un clúster optimizando considerablemente el rendimiento de lectura, escritura y procesamiento de los datos en un sistema BI.

Entre las partes del framework a definir, constara de las siguientes partes: Un repositorio donde almacenaremos los datos en bruto a analizar (fuentes de datos), una herramienta de procesamiento de datos(ETL), un sistema de datawarehouse, y por último una herramienta de generación de cubos multidimensionales permitiendo la consulta de información de esquemas multidimensionales, cabe destacar que las partes del framework no serán tecnologías tradicionales, si no tecnologías BigData.

Para ello, me apoyare en los datos open data publicados en el portal datasf.org, de la ciudad de San Francisco(California), para diseñar un modelo multidimensional para ejecutar en dicho framework.

Estos objetivos están sujetos a las limitaciones que tengo para realizar el proyecto, por lo que no incluirá un clúster, ya que solo se dispone de un portátil personal para la realización del proyecto y no un gran servidor y clúster donde montar el framework, pero es objetivo de este proyecto que dicho framework pueda ser exportado a servidores, ya sea en clúster o en la nube y ampliar su capacidad de procesamiento, utilizando tecnologías actuales para el despliegue de servidores.

4. Estado del arte.

4.1. Historia de Inteligencia de Negocio (Business Intelligence)

No se puede negar hoy en día la importancia de analizar los datos que disponemos. Desde la antigüedad una de las características que han permitido evolucionar al ser humano es su capacidad para observar, reunir información y aprender, ya sea por el simple hecho de la curiosidad de conocer lo que nos rodea o tomar mejores decisiones. Está presente desde la invención del fuego basada en la experiencia, en nuestros primeros días de vida cuando observamos sin ser conscientes siquiera que llorando nos dan de comer o cambian el pañal, en la medida del ancho de un terreno en la antigüedad para saber el espacio de las verduras a plantar o en la base de todas las ciencias, donde de una manera más formal y científica, se ha pretendido medir de manera más objetiva los sucesos que nos rodean, o en elegir a que colegio ira nuestro hijo. Junto con esta evolución del ser humano, los métodos para analizar estos hechos han ido evolucionando también con el fin de permitirnos mejorar en nuestros análisis. (2)

En la era moderna, en la era informática, se puede decir que todo empezó con la era de la computación, donde en 1940 Alan Turing matemático y científico de computación junto con Irving John Good matemático, consiguen que una maquina(enigma) sea capaz de analizar documentos cifrados por los alemanes en la Segunda Guerra Mundial y descifrarlos. En 1946 el proyecto Manhattan se utilizaron ordenadores para predecir y analizar el comportamiento que podría causar una reacción nuclear, aunque lamentablemente esto no sirvió para que el presidente Truman no lanzase aquella bomba sobre Hiroshima. (3) (4)

Mas adelante entre 1950 y 1969 se genera el primer modelo de predicción meteorológica con el análisis de datos, y en 1966 se crea un proyecto de investigación analítica financiado por el Ministerio de Agricultura de Estados Unidos que se conocerá más adelante como SAS Institute. De hecho, en 1958, Hans Peter Luhn, investigador de IBM acuño el termino Business Intelligence(BI) o Inteligencia de negocio en el artículo "A Business Intelligence System", definiendo negocio(Business) como "una colección de actividades realizadas para cualquier propósito, ya sea ciencia, tecnología, industria, ley, gobierno, defensa etc." e inteligencia(Intelligence) como "la capacidad de aprender las interrelaciones de los hechos presentados de manera que guie la acción hacia la meta deseada". Cuatro años después, el canadiense Kenneth Iverson desarrollo una nueva notación para operaciones sobre series numéricas, inventando el primer lenguaje de programación multidimensional, que será la base para el procesamiento analítico. (3) (4)

Entre 1970 y 1999 la popularidad del análisis de los datos crece, creándose modelos como Black-Scholes que consistía en estimar el valor de las acciones europeas en una fecha futura, siendo la década de los 70 cuando se desarrollan las primeras bases de datos y aplicaciones empresariales como SAP y en la década de los 80 se crea del concepto de Datawarehouse o almacén de datos por Ralph Kimball y Bill Inmon. En la década de los 90 aparecen los primeros proveedores de BI, pero estos resultan caros y requerían un equipo interno o delegarlo en terceros para manejar el registro de datos y los análisis. (3) (4)

Es ya en la primera década del siglo XXI, cuando se empiezan a consolidar las aplicaciones de BI, y empiezan a aparecer unas pocas plataformas como Oracle, IBM, SAP o Microsoft, permitiendo a las empresas obtener sus análisis sobre los datos a través de un tablero o Dashboard de fácil uso, permitiendo que cualquier persona de la empresa pudiese hacer uso de los datos sin un intermediario. Dicho de otra forma, son los propios usuarios que desde la aplicación final analizan, monitorizan y toman sus decisiones. (3) (4)

4.2. ¿Qué es Business Intelligence?

En la actualidad, el término Business Intelligence se define como todos los métodos, herramientas y procesos que permiten transformar los datos almacenados en información y conocimiento dirigido a un plan o estrategia empresarial permitiéndonos una mejor toma de decisiones. Dicho de otra manera, nos permite reunir, depurar y transformar los datos de los sistemas transaccionales e información desestructurada ya sea interna o externa a la compañía en información estructurada, para poder ser explotada de manera directa con técnicas de reporting y análisis OLTP / OLAP dando soporte a la toma de decisiones sobre un negocio. (5)

Hoy en día, la inteligencia de negocio actúa como un factor estratégico para una empresa u organización, ventaja competitiva a través de responder con la información a los problemas de negocio, como puede ser la entrada a nuevos mercados, promociones, ofertas de productos, control financiero, optimización de costes, planificación de la producción, análisis de clientes y un largo etc., optimizando la utilización de sus recursos y monitoreando el cumplimiento de los objetivos de la empresa día a día ayudando a la toma de decisiones y obtener mejores resultados. (5)

Algunos de los problemas que enfrentan las empresas a día de hoy es que tienen datos pero no tienen información, es importante almacenar los datos de los clientes, departamentos, compras, ventas, etc., pero toda esta información está repartida entre distintas aplicaciones, sistemas financieros o fuentes departamentos, pero esto solo nos otorga una visión puntual o particular sobre algo en concreto, pero si queremos que nuestra empresa tenga una mayor ventaja competitiva sobre los competidores, esta tarea no es suficiente, necesitamos también una visión global, y además profundizar el nivel de conocimiento de nuestros clientes, operaciones, empleados, para tener la capacidad de encontrar patrones de comportamiento, monitorear, entender, administrar o interrogar cuestiones propias de la empresa para permitir maximizar el rendimiento de la misma. (5)

Otro problema que afrontan las empresas es la fragmentación de sus datos, poseen aplicaciones independientes de los departamentos, por ejemplo, una aplicación para la venta, y otra aplicación distinta para el almacén y otra para el departamento financiero, careciendo de una visión global de la empresa, tal vez por la complejidad o incapacidad de las herramientas BI de integrar distintas fuentes de datos heterogéneas, limitando a la empresa en su toma de decisiones al tener la información fragmentada, conduciendo a lo que se conoce como distintas versiones de la verdad. Los gerentes de las empresas solicitan informes a los distintos departamentos obteniendo diferentes resultados de un mismo informe. Siendo clave la justificación de donde vienen esos datos y qué condiciones se utilizaron para la creación de dicho

informe, y si el gerente decide añadir alguna variable más al informe, esto se convierte en una tarea de semanas y volver a empezar. (5)

Ante la necesidad de generar estos infórmenes y análisis de negocio, lleva a extraer e integrar manualmente las distintas aplicaciones, pero esto es un proceso muy lento y costoso, y en definitiva con un bajo rendimiento, como dice el refrán “cuando me conocía todas las respuestas me cambiaron todas las preguntas...”, y en el mundo empresarial donde la incertidumbre y el cambio no son una excepción, hace que estos sistemas sean poco fiables. (5)

Otra de las ventajas que nos aportara un sistema BI es el control de costos y producción, que es una de las necesidades a las que se enfrenta una empresa, la necesidad de medir y controlar el gasto a distintos niveles de detalles identificando la línea de negocio, producto, costes de producción, entre otros. (5)

Además las empresas almacenan gran cantidad de información valiosa, provocando la necesidad de transformar esa información en conocimiento y entender mejor a sus clientes, y utilizarlo para obtener algún tipo de ganancia para la empresa, como puede ser fidelizar clientes, anticipar oportunidades, medición de campañas de marketing, identificar patrones de compras y el comportamiento de los clientes, así como también la medición de métricas u objetivos de negocio o Indicadores clave de rendimiento(KPI por sus siglas en ingles) que permitan monitorear y analizar si la empresa lleva el rumbo adecuado y está cumpliendo los objetivos definidos, permitiendo analizar la raíz de los problemas y estos indicadores, desde distintas perspectivas y niveles de detalle, permitiéndonos tener esa visión global de la empresa necesaria y tomar las decisiones oportunas para dirigir la organización o empresa en la dirección adecuada. (5)

Por lo que es necesario montar un servidor analítico propio separado de los sistemas transaccionales para conseguir la máxima agilidad, y que el análisis sea de la mayor calidad posible, permitiéndonos manejar el crecimiento de nuestros datos, permitiendo a la empresa cumplir uno de sus mayores retos, crecer y evolucionar, que en términos generales significa cambio, y deben de tener un sistema ágil que permita enfrentar esos cambios y las necesidades puntuales de la empresa.

4.3. Arquitectura de un sistema BI

Para resolver esta problemática, se monta un servidor analítico o servidor BI separado de las bases de datos operacionales, debido a que las consultas analíticas tienen un gran peso sobre el servidor de BD, y si estas se hacen directamente sobre el sistema transaccional, se ralentizaran las consultas del propio modelo de negocio. Dicho esto, en la siguiente imagen vemos un esquema típico de BI.

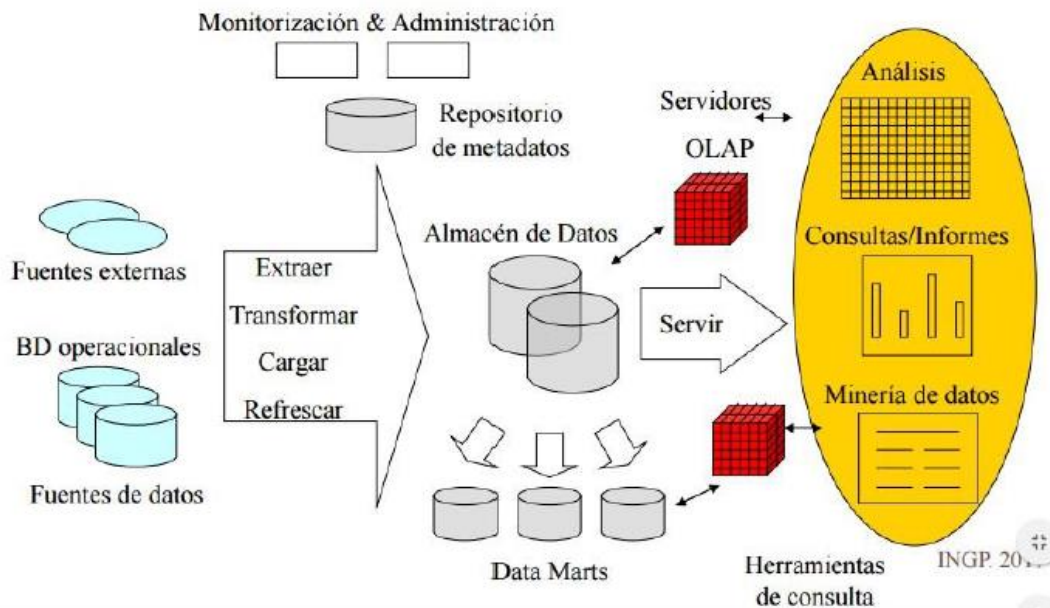


Figura 4. Arquitectura BI – Apuntes INGP Juan Carlos Trujillo Mondéjar.

Aunque podemos encontrar distintas imágenes de una arquitectura BI con más o menos semántica, todas tienen los mismos puntos en común, que de izquierda a derecha serán definidos a continuación:

4.3.1. Fuentes de datos

En primer lugar, tenemos las fuentes de datos, estas pueden ser distintas BD operacionales internas o externas, como los distintos departamentos ya sean Oracle, MySQL, MongoDB, documentos Excel, archivos XML, mainframe, etc. o incluso de la cantidad de datos ingente de internet, redes sociales etc.

4.3.2. Procesos ETL

Una vez definidas cuales van a ser nuestras fuentes de datos de las cuales se va alimentar nuestro sistema, haremos una copia de los datos de esta y los llevaremos al servidor BI dedicado, y una vez hecho esto, ya podemos empezar lo que se conoce como procesos ETL por sus siglas en inglés (Extraction, Transformation and Loading), esta fase conlleva entre el 60% y el 80% del esfuerzo en un sistema BI, en algunos diagramas aparece también Refrescar, con el objetivo de indicar que no es solo extraer transformar, cargar los datos y olvidarnos, si no que a medida que el negocio crece, y los datos van aumentando, debemos de ir incorporando estos datos también a nuestro sistema BI, ya que, solo tendríamos un reporte puntual, y el objetivo de un sistema BI es monitorizar diariamente nuestro negocio, pero en definitiva, esta fase es la que recoge todos los datos de las distintas fuentes de datos, teniendo como principales acciones las siguientes:

- **Validación de datos:** con esta acción se pretende hacer una primera exploración o buceo en los datos, e intentar detectar inconlucias en los mismos, de manera que detectemos si hay datos que tienen resultados anómalos, por ejemplo, numero de hora trabajadas negativo, costes negativos, que la hora de llegada de una patrulla sea anterior a la fecha de salida, estos errores generalmente se deben al error humano a la

hora de introducir los datos en las bases de datos operacionales, pero como científicos de datos debemos detectar estas anomalías y comunicarlas al cliente, de manera que nos diga cómo tratarlos y poder hacer un sistema analítico fiable, si esto no es así, podemos llegar a resultados erróneos, o a distintas versiones de la verdad, por lo tanto, es tan importante montar un servidor analítico potente, como detectar estas errores y tener unas fuentes de datos fiables y datos coherentes.

- **Extracción de datos:** son los procesos que recogen los datos necesarios del almacén de datos para que podamos tratarlos.
- **Limpieza y transformación de los datos:** Nuestro objetivo es limpiar esos errores o tratarlos como el cliente nos solicite, y transformar los datos ya que, al tener muchas fuentes de datos, hay una alta probabilidad de error y anomalías, como por ejemplo longitud de campos inconsistentes, o una descripción inconsistente, por ejemplo ¿qué es categoría?, la categoría del cliente, del producto etc. Valores nulos que tengamos en los datos o valores incoherentes, un ejemplo típico de esto es U. Alicante vs UA u Hombre/Mujer vs Masculino/Femenino.
- **Carga de datos:** Una vez hecho las acciones anteriores, los datos se cargan en el repositorio o almacén de datos, no confundir con el almacén de nuestras fuentes de datos, y lo normal es que esta acción lleve un pre-proceso antes de la carga, por ejemplo comprobar las reglas de identidad de nuevo, calcular datos agregados, tablas derivadas y virtuales e intermedias, construcción de índices, depende la necesidad específica del problema, además estará sujeta también a como hayamos diseñado el repositorio de datos donde haremos la carga, y también debemos de calcular el tiempo que este proceso lleva, hablamos de horas y frecuentemente se hace por la noche, aunque también hay casos en los que el negocio es una multinacional, y siempre hay en alguna parte de la empresa en la que se están cargando datos, esto junto con otros factores dará pie a la analítica de datos en tiempo real, aunque en esto entrare más adelante.

Como se ha podido comprobar, esta parte del proceso conlleva una parte significativa de todo el proceso, además otro factor que influye en la por ello requiere recursos, estrategia, habilidades especializadas y tecnológicas adecuadas.

4.3.3. Almacén de datos o Datawarehouse

Este repositorio donde hemos cargado los datos transformados se conoce en inglés como Datawarehouse, es otra de las partes claves en una arquitectura BI y está diseñado de una manera concreta para optimizar las operaciones de consultas, este concepto fue creado por Ralph Kimball y Bill Inmon, aunque su metodología a la hora de construirlo es diferente, Top-Down vs Botton-Up la definición y objetivo de las mismas es similar, podemos encontrar más información sobre la metodología en el libro “The data warehouse toolkit: the definitive guide to dimensional modeling” de Ralph Kimball, o “Building the data warehouse” de Bill Inmon, pero en definitiva, un datawarehouse será un repositorio que aporta una visión global y común de los datos de la empresa independientemente de cómo vayan a ser visualizados posteriormente, y de manera adicional tendremos Datamart, podemos entender esto como un subconjunto de los datos del datawarehouse con la finalidad de responder a determinados análisis de usuarios finales, por ejemplo no es la misma información ni consulta de los datos que tiene que hacer un gerente de almacén, o el gerente de una organización mundial. Además, el data mart puede ser

dependiente o independiente del datawarehouse, por ejemplo en el caso de los datawarehouse independientes se le puede aplicar machine learning o minería de datos para obtener patrones, u otro tipo de información de los datos, que no son aplicados directamente sobre los datos del datawarehouse, que un conjunto de usuarios finales específicos necesita, aunque estos datawarehouse no son tan frecuentes en un sistema tradicional de BI, si es más típico en arquitecturas BigData donde lo encontraremos más frecuentemente, pero ya sea dependiente o independiente, la diferencia principal con el datawarehouse es que los datamarts tienen la finalidad de cubrir las necesidades de un determinado departamento dentro de una organización. (6) (7) (8)

En este punto y en la manera de abordar y construir este datawarehouse y Datamarts es donde difieren Kimball y Inmon:

Inmon hace referencia a una metodología Top-Down donde los datos extraídos de nuestras fuentes de datos y tratados a través de los procesos ETL, son cargados primero en un datawarehouse, o dicho de otra manera, se cargan todos los datos de la compañía junto con los metadatos donde se documenta de forma muy clara y precisa el contenido del datawarehouse y después se crean los data mart obteniendo la información del datawarehouse aplicando sus propios procesos ETL sobre el datawarehouse, este enfoque es más difícil de desarrollar en proyectos grandes ya que se está intentando abordar directamente el todo para después ir al detalle, es decir, agrupar todos los datos de la empresa primero, para a partir de esta construir el de los departamentales. (7) (8) (9)

El otro enfoque que tenemos es de Ralph Kimball, el cual definiendo el datawarehouse como el conjunto de datamarts de una empresa, define primero los datamarts para posteriormente integrarlos todos en un datawarehouse global para la organización. Además, Kimball define el modelo multidimensional (no normalizado) para estos datamarts que incluye las dimensiones de análisis junto a sus atributos, por ejemplo, de un cliente, su dirección, sexo, edad entre otro tipo de información, así como los hechos o eventos del negocio que se quieren analizar, por ejemplo, una venta o la información recogida por los sensores en un momento determinado. Este enfoque es un enfoque bottom-up, de abajo a arriba, donde primero se construye los distintos datamarts para finalmente llegar al datawarehouse, permitiéndonos ir del detalle al todo, desde mi punto de vista, a una clara referencia a la técnica de divide y vencerás, donde abordaremos primero partes más pequeñas del problema para finalmente llegar a la solución final. (6) (8)

Kimball define un **modelo multidimensional** para estos datamarts diciendo que los datos se modelan en cubos de datos, estructuras multidimensionales o hipercubos cuyas operaciones más comunes serán Roll up y Drill down, es decir incremento y decremento en el nivel de agregación de los datos, Slice y Dice como reducción de la dimensionalidad de los datos mediante selección y proyección respectivamente. (6)

De esta manera los componentes principales de un modelado multidimensional serán dimensiones, hechos y medidas.

Se puede definir una **dimension** como la perspectiva desde la cual una empresa quiere mantener sus datos organizados, por ejemplo, tiempo, localización, clientes, proveedores, trabajadores,

tiendas, cada una de estas sería una dimensión. Cabe destacar que la dimensión tiempo es prácticamente obligada en cualquier sistema BI, ya que generalmente esos hechos se producen en un momento determinado y queremos mantener un histórico de los datos. (6)

Estas dimensiones estarán compuestas por miembros, por ejemplo, año, mes, día, día de la semana, trimestre son miembros de la dimensión tiempo, al igual que ciudad, provincia/estado, país son miembros de la dimensión localización. (6) (8)

A su vez estos miembros se suelen organizar en jerarquías, como podemos ver en la siguiente imagen, desplegándolo de izquierda a derecha Drill Down y de derecha a izquierda Roll up:

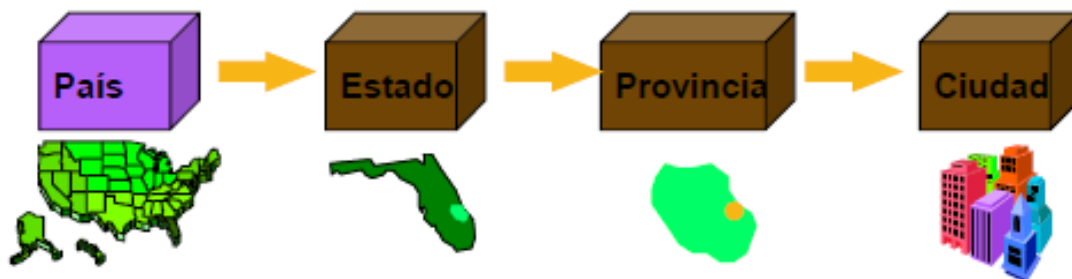


Figura 5. Jerarquía de las dimensiones – Apuntes INGP Juan Carlos Trujillo

Los **hechos** son las colecciones de datos relacionadas compuestas por medidas y su contexto, determinado por las dimensiones, es decir, cada hecho o evento en concreto está asociado a las distintas dimensiones sobre las cuales estamos midiendo el hecho. Esta tabla de hechos se compone de claves primarias compuestas de claves ajenas a las dimensiones. Generalmente la relación es N-N con las dimensiones y N-1 en particular con cada dimensión. Es importante destacar que con el fin de que obtener el máximo rendimiento, estas claves ajenas serán de tipo numérico, ya que a nivel de BD es mucho más rápido comparar si es el entero que buscamos, que comparar si es el string que buscamos. (6) (8)

Por último, tenemos las **medidas**, que es un atributo numérico asociado al hecho, es decir, las medidas son aquello que queremos medir, ya sea el número de ventas, tiempo de respuesta de una patrulla o el número de asientos disponibles en un avión asociadas al hecho. (6) (8)

Un ejemplo de valores de una tabla de hechos sería el siguiente:

Tabla 1. Ejemplo valores de una tabla de hechos – Alejandro Reina Reina

idVenta	idCliente	IdFecha	idLocalizacion	PrecioTotal	numArticulos
1	25	14	54	157,14	17
2	80	67	2	5,98	1
3	40	58	124	6,47	1
4	70	71	637	99,98	6

Siendo idVenta, idCliente, idFecha, idLocalizacion, las claves primarias que son clave ajena a las dimensiones, y PrecioTotal y numArticulos serian dos ejemplos de medida.

Un ejemplo de los valores a modo de tabla de una dimension con jerarquías sería el siguiente:

Tabla 2. Ejemplo tabla de dimensión – Alejandro Reina

id	País	Provincia	Ciudad
1	España	Alicante	San Vicente del Raspeig
2	España	Murcia	Yecla
3	España	Cataluña	Barcelona
4	España	Galicia	Santiago de Compostela

Finalmente formaríamos un diagrama similar a la siguiente imagen:

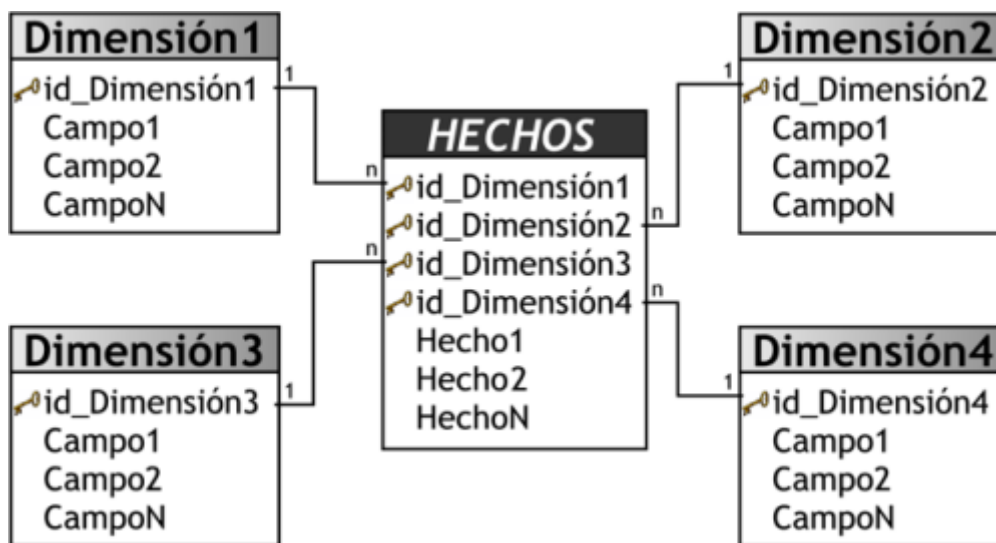


Figura 6. Esquema Estrella www.dataprix.com

Aunque hay variantes como el diagrama, como el diagrama de copo de nieve y constelaciones de hechos, no entrare en estas al ser menos comunes.

Es importante disponer también de un repositorio de metadatos, donde tendremos datos de los datos, es decir, de los datos que tenemos de los clientes, tendremos información sobre qué datos guardamos, donde los guardamos, que campos contiene la tabla o conjunto de tablas, de que fuentes de datos nos hemos provisto para conseguir esos datos, nivel de agregación, cuando se actualizan, y cuando fue la última actualización de los datos en el datawarehouse. (6) (10)

Por último, el servidor es un SGBD con la información de estos datamarts y del datawarehouse donde se almacenarán los datos, gestionando el repositorio propio del almacén de datos, se

coordinarán los procesos ETL que alimentan el datawarehouse y los datamarts, y donde se procesaran las consultas analíticas lanzadas sobre el almacén de datos, devolviendo datos. En BI tradicional generalmente serán servidores relacionales como MySQL u Oracle entre otros proveedores. (6) (8) (10)

4.3.4. Servidor de consultas

Siguiendo el diagrama de la arquitectura de un sistema BI, llegamos al servidor de consultas, generalmente se utiliza un servidor distinto al del almacén de datos por motivos de rendimiento y mantenimiento, y en este punto predominan dos tecnologías ampliamente utilizadas como son ROLAP y MOLAP, ambas sobre modelos multidimensionales.

Cabe destacar que ambas, tanto ROLAP como MOLAP están basadas sobre OLAP (On-Line Analytical Processing), son sistemas de ayuda a la toma de decisiones, por lo que tendremos principalmente consultas muy complejas con muchos datos, funciones de agregación(join) y con actualizaciones poco frecuentes (generalmente una vez al día y por la noche) y otra prioridad de estos sistemas es la optimización de las consultas y los tiempos de respuesta.

ROLAP (Relational On-Line Analytical Processing) están basados en sistemas y herramientas OLAP construidos sobre una base de datos relacional (MySQL, Oracle, etc.), y diseñado para realizar análisis de los datos mediante sistema relacionales clásicos, utilizando extensiones de SQL estándar para soportar el acceso multidimensional a los datos, con la ventaja de estar basado en el estándar MySQL. (10)

MOLAP (Multidimensional On-Line Analytical Processing) se diferencia en que requiere un procesamiento previo y almacenamiento de la información en estructuras de datos multidimensionales, es decir, matrices multidimensionales sobre las que se realizan directamente las operaciones OLAP. (10)

Independientemente de la técnica que utilicemos, ambas vienen a montar un cubo de datos, o estructuras multidimensionales, que podríamos ver de una forma gráfica en la siguiente imagen:

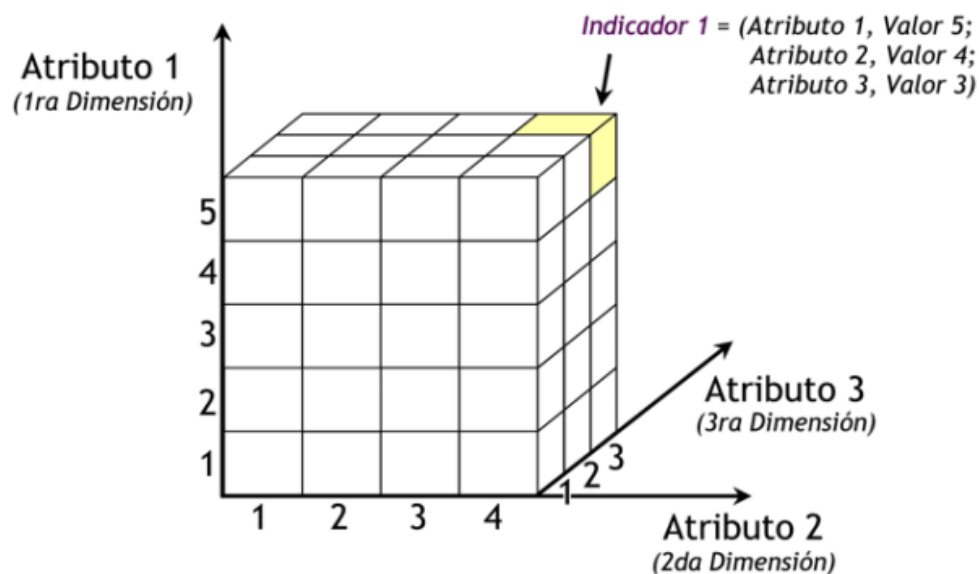


Figura 7. Cubo multidimensional. www.dataprix.com

Los atributos(dimensiones), existen a lo largo de varios ejes, siendo su intersección, el indicador que se está consultando.

Al igual que en el diseño del datawarehouse, es posible montar cubos específicos basado en un Datamart o en el datawarehouse, dependiendo del usuario final que consulte la información. (10)

4.3.5. Visualización

En la última parte de la arquitectura de un sistema BI, pero no por ello menos importante, tenemos la parte de visualización de datos, que será la parte que estará en contacto con el usuario final y realizara las consultas, llamados cuadros de mando(Dashboard) donde se realizaran las consultas, mostrando los datos base en términos de negocio y la generación de

En la siguiente imagen podemos ver un típico Dashboard o cuadro de mando.



Figura 8. Dashboard PowerBI. Google Images

Volviendo al diagrama de arquitectura de un sistema BI, desde un punto de vista desde las herramientas que utilizamos para montar un sistema de este tipo actualmente, para los ETL, se utilizaran herramientas como Pentaho Data Integration siendo esta open source, Oracle Data Integration y SAP BI-BW de pago entre otras, para el diseño del cubo multidimensional una de las herramientas más utilizadas es Mondrian, y en la parte de visualización de datos tenemos un gran abanico de aplicaciones para generar esos informes y Dashboard para los usuarios finales,

teniendo JPivot de Pentaho, PowerBI de Microsoft, Tableau, QlikView entre muchísimas otras opciones.

4.4. Problemática del BI tradicional.

Aunque una arquitectura es totalmente válida y robusta para su propósito, tiene una serie de limitaciones.

Uno de sus problemas principales es el volumen de datos, hay que pensar que estos sistemas están montados para consultar datos históricos, y aunque son capaces de consultar en grandes cantidades de datos, hablamos de millones de filas, pero ¿Qué ocurre cuando queremos procesar gran cantidad de información? Las bases de datos solo alcanzan un límite de inserciones por segundo, por lo tanto, es una limitación de estos sistemas, de hecho, las tecnologías tradicionales no soportan un sistema BI con más del orden de miles de filas por dimensión y tablas de hechos a partir de 10 millones de filas, a partir de ahí el sistema empieza a colapsar, por lo que, en resumen, las tecnologías tradicionales o bases de datos relacionales son a menudo incapaces de gestionar un gran conjunto de datos o dataset.

Otro punto débil de estos sistemas, es que tienen toda la información almacenada en un servidor central del cual se extrae la información, esto puede generar cuellos de botellas, además de la sobrecarga en el procesamiento en un único servidor central, donde la escalabilidad de este servidor puede ser menor y mucho más costosa.

También otro aspecto a tener en cuenta es cuando se requiere la información en tiempo real, debemos definir muy bien que es tiempo real, es importante destacar que nunca vamos a tener los datos en tiempo real, ya que se requiere un tiempo de cómputo de los datos, pero cuando hablamos de analítica en tiempo real, queremos reducir ese tiempo de cómputo al máximo, y servir los datos lo antes posible, dependiendo del volumen de los datos, hablamos de minutos o segundos, en un BI tradicional, este proceso de carga en bases de datos relacionales es más lento, además de las limitaciones en cuanto a ingestión y rendimiento mencionadas anteriormente, por lo que las tecnologías BI tradicionales no soportan análisis de datos en tiempo real.

Por último, otro gran problema que tienen las tecnologías BI tradicionales, es que solo realizan un tratamiento de datos estructurados, es decir, los almacenados en bases de datos tradicionales o documentos en formato tabla, como puede ser un documento Excel, o un fichero csv, entre otros, pero no pueden procesar datos semiestructurados o no estructurados al estar basados sobre bases de datos relacionales. (8) (11)

4.5. Inicios de BigData

Hasta 2011/12 no se empezó a popularizar el término Big Data, pero ¿cómo comienza toda esta corriente? Todos los autores, coinciden en situarlo en Google.

Google tenía un problema del page range a la hora de procesar los datos, cuyo proceso le tardaba cerca de 8 horas en terminar, y cuando paralelizaban era muy fácil que el proceso fallase y tuviesen que volver a empezar desde el principio, por lo que empezaron a desarrollar herramientas para arreglar este problema. Es en 2003 cuando Sanjay Ghemawat, Howard

Gobioff y Shun-Tak Leung publican un paper donde explicaba su sistema de ficheros distribuidos Google File System(GFS) titulado “The Google File System”. Es ya en 2008 cuando Jeffrey Dean y Sanjay Ghemawat, dos ingenieros de Google publican un artículo llamado “MapReduce: Simplified Data Processing on Large Clusters”, donde hablan de un nuevo modelo de programación que permite simplificar el procesamiento de grandes volúmenes de datos que será conocido como MapReduce, que utilizaban internamente en Google para procesar los grandes volúmenes de datos que manejaban como documentos, referencias web, paginas entre otras, de hecho, esto para Google fue algo necesario que permitió a Google ser quien es hoy en día, puesto que lo utilizaban para obtener una serie de métricas para medir el posicionamiento de una página, que más tarde se popularizo en industrias como el SEO y SEM. La idea principal de MapReduce es montar un esquema paralelo de computación que permita distribuir el trabajo de procesamiento de datos entre diferentes maquinas o nodos dentro de una red, para reducir considerablemente el tiempo de procesamiento, aplicando uno de los paradigmas de la informática, divide y vencerás, reduciendo un gran trabajo en tareas más pequeñas y estas siendo paralelizadas en diferentes nodos. (12) (13)

A raíz de esta publicación, y que la comunidad vio lo que estaban haciendo, fue cuando comienza a extenderse, sentando la base para la mayoría de tecnologías NoSQL, de hecho a raíz de esta publicación la comunidad de apache puso en marcha el proyecto Hadoop, un proyecto open source que sentó las bases para lo que hoy conocemos como BigData, de hecho lo que Google llamo Google File System(GFS), en Hadoop se conocerá como Hadoop Distributed File System(HDFS), BigTable que era el servicio de base de datos NoSQL de Google, paso a ser conocido como HBase en Hadoop, aunque en esto entrare más adelante.

4.6. ¿Qué es BigData?

Aquí es cuando empieza la gran controversia en el mundo de BigData, es en 2001 cuando Doug Lane de la consultora Gartner define BigData como un término aplicado al conjunto de grandes datos caracterizados por 3V, Volumen, velocidad y variedad que superaban la capacidad del software habitual para ser capturados, gestionados y procesados. Desde que se definió el termino, otras V's han ido añadiéndose, como “Veracidad de los datos” y “Valor de los datos”, y otros autores aun han extendido más la definición añadiendo otras 2V's “Visibilidad de los datos” y “Viabilidad de los datos”, pero independientemente de las V's que usen los autores para su definición, las 3V's principales son las que van a caracterizar las necesidades de una arquitectura BigData, Volumen, Velocidad y Variedad. (14)

En cuanto al volumen, podemos remitirnos a un artículo Chris Mattman, investigador principal de la iniciativa de BigData en el Laboratorio de propulsión a Chorro de la Nasa en Pasadena, California, para Information Weekly en 2013 donde decía que la NASA en total estaría manejando probablemente cientos de petabytes de datos acercándose a exabytes si unificaba todas las ciencias y disciplinas de la ciencia planetaria y espacial, donde además decía que hoy en día para misiones o proyectos individuales era habitual recolectar cientos de terabytes de información. Aunque en concreto, ellos dividen esto como dos tareas “independientes” o mejor dicho, proyectos con objetivos diferentes. Por un lado, uno de los objetivos es archivar los datos, es decir, el almacenamiento de esos datos y su administración, por ejemplo, los datos del Sistema de Observación de la Tierra(EOS), donde esta información es almacenada en las

instalaciones del Centro de archivos Distribuidos activos (DAAC), es un proyecto bastante grande y su trabajo es garantizar que los datos se conserven.

Por otro lado, otros proyectos dependen más del análisis de estos datos, como por ejemplo en radioastronomía el Square Kilometer Array(SKA) que incluye miles de telescopios en Australia y Sudáfrica para explorar la formación temprana de galaxias, los orígenes del universo y otros misterios. (15)

Otra de las V's Velocidad se define por su propio nombre, queremos procesar este gran conjunto de información y lo queremos procesar de la manera más rápida posible. Es indudable que el crecimiento de los datos y la explosión de las redes sociales como Facebook o Twitter han cambiado la forma en la que queremos y vemos los datos, es decir, lo que antes los datos de ayer eran recientes, ahora lo que interesa son los datos del último segundo, el ultimo tweet. El movimiento de la actualidad tiende al tiempo real, como es en el caso de las redes sociales, buscando que la ventana de actualización de los datos sean fracciones de segundos. (16)

Por último, la otra gran V de BigData es la variedad de los datos en la que ya no se pretende analizar datos estructurados almacenados en bases de datos relacionales o ficheros tipo Excel, CSV, si no que a veces los datos los encontraremos en formato video, audio, PDF o cualquier otro formato que podamos imaginar. Por lo que existe la necesidad de organizar todos estos datos y obtener información de ellos. Por lo que este es otro de los grandes desafíos del BigData. (16)

En la siguiente imagen podemos ver una representación de lo anterior:

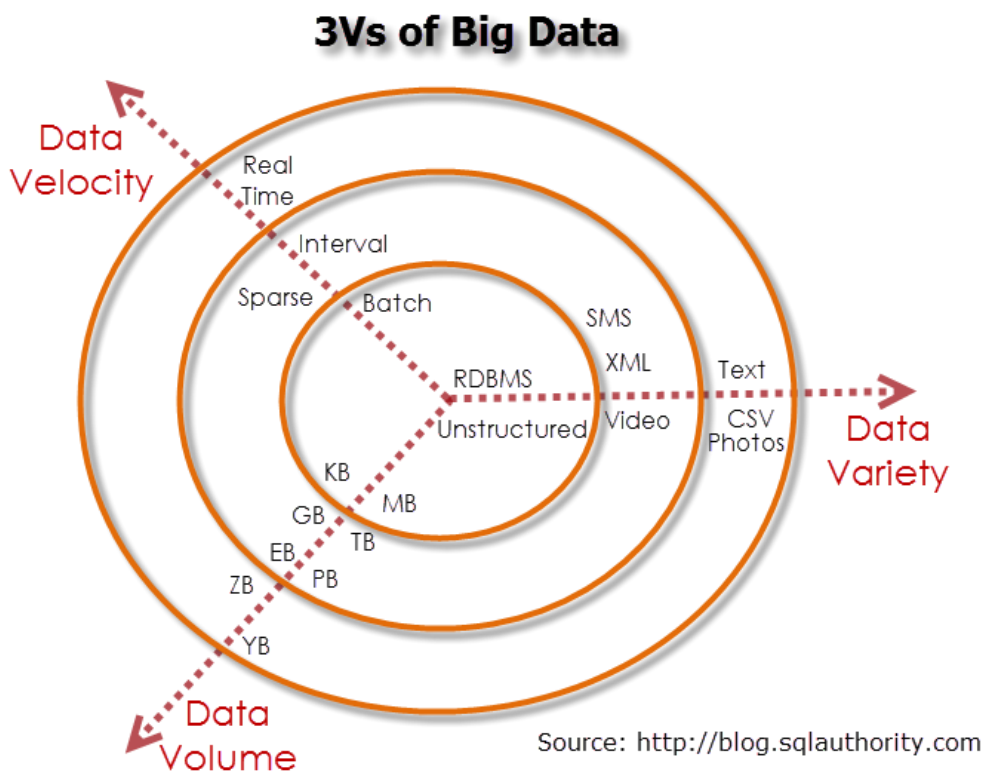


Figura 9. Las 3V's de BigData. www.blog.sqlauthority.com

4.7. Landscape de Big Data.

Para empezar este punto quiero hacer referencia a charlas a las que he asistido relacionadas con el mundo de Big Data, una impartida en la UA por un profesor de la UOC o el curso de Big Data impartido en la UA por la escuela de verano “Rafael Altamira”, que contó con varios exponentes de la UOC y profesionales del sector, todos comenzaban o coincidían en algo similar cuando comenzaban sus charlas, a la siguiente cita “Big Data es como el sexo en la adolescencia: todo el mundo habla de él, nadie sabe cómo hacerlo, todos creen que los demás lo están haciendo y, claro, todos dicen que lo hacen”. Esto ya nos permite hacernos una idea de lo podemos encontrar.

Si buscamos landscape de BigData en Google, fácilmente nos encontraremos con imágenes como las siguientes publicada por Matt Turck de First Mark con la colaboración de Jim Hao, de hecho, solo la familia Apache tiene más de 30 proyectos referentes a Big Data. (17)



Figura 10. Landscape BigData – Arquitectura. mattturck.com/bigdata2017



Figura 11. Landscape BigData – Analítica. mattturck.com/bigdata2017

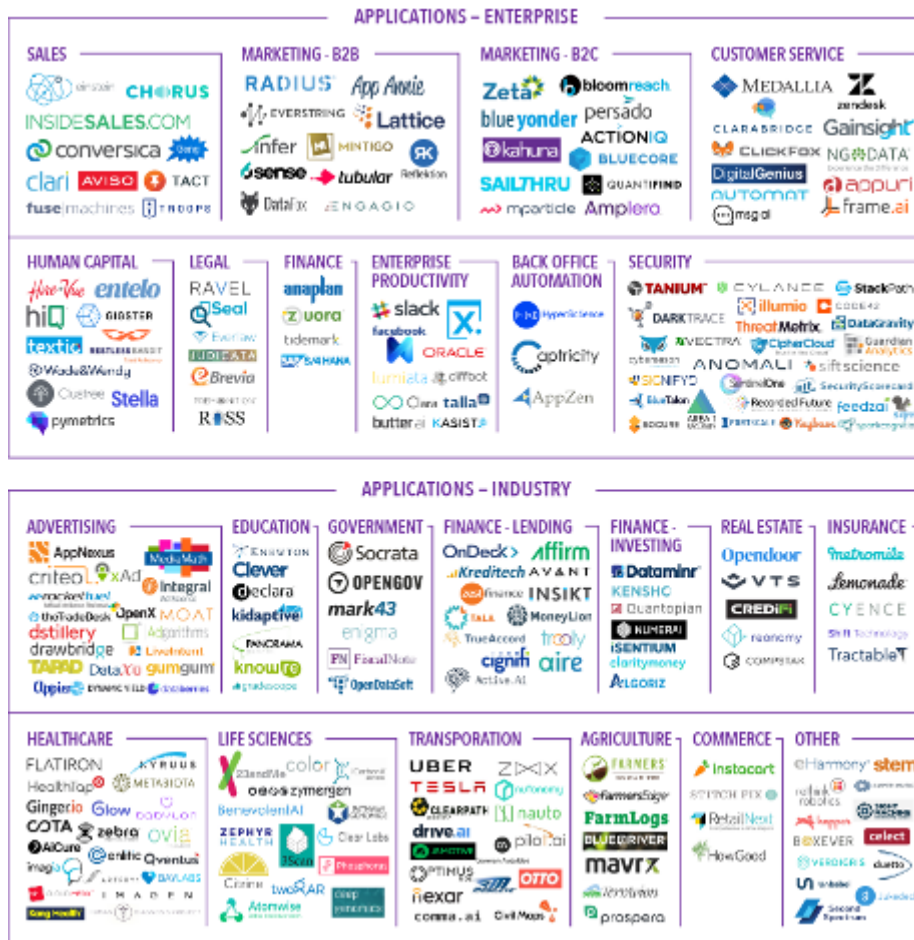


Figura 12. Landscape BigData – Aplicaciones en industrias y empresas. mattturck.com/bigdata2017



Figura 13. Landscape BigData – infraestructura/analítica cruzada. mattturck.com/bigdata

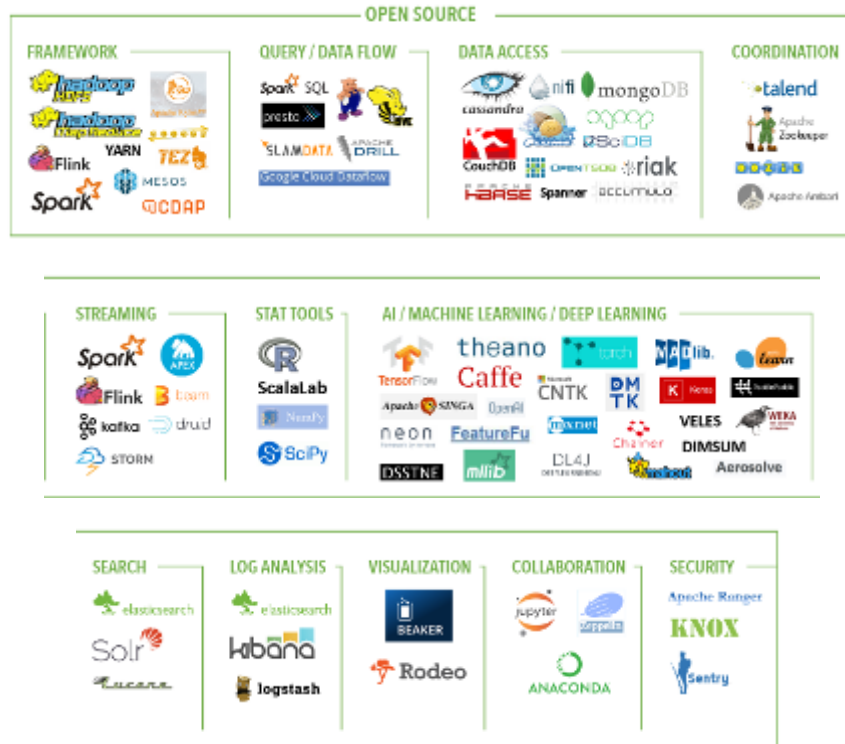


Figura 14. Landscape BigData – OpenSource. mattturck.com/bigdata2017

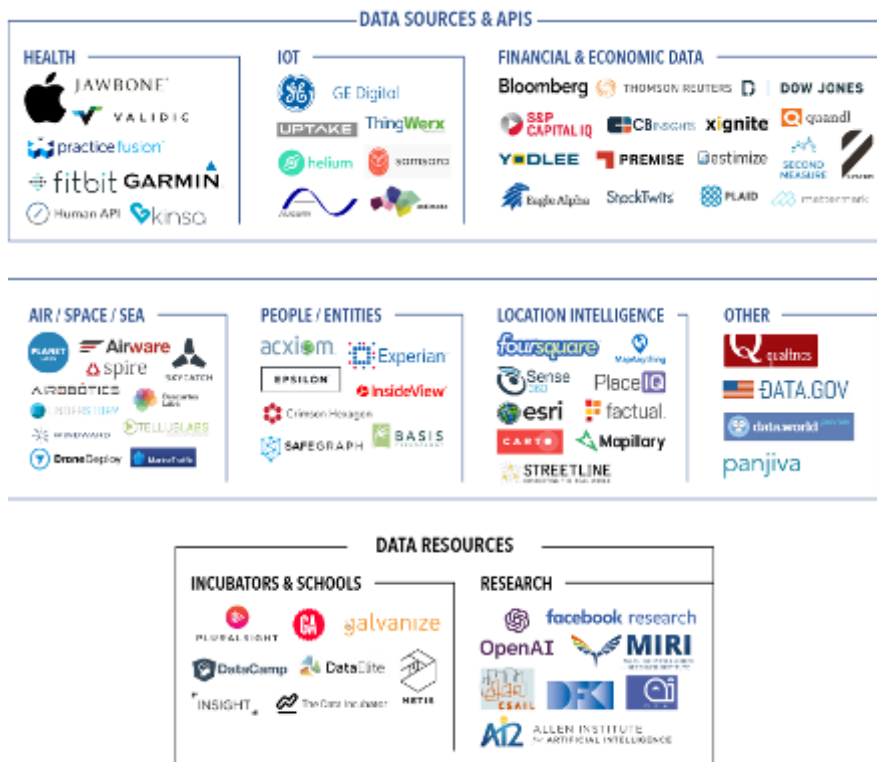


Figura 15. Landscape BigData – Fuentes de datos y APIs. mattturck.com/bigdata2017/

Remitiéndome a un artículo publicado por Javier Lahoz Sevilla, responsable de Big Data de Orange y Profesor del instituto de economía digital, donde explica los motivos y cuál es el perfil de un experto en big data, las condiciones que requiere y su contribución en los equipos. (18)

Destacando en primer lugar, que él hace referencia al Landscape de 2016, que es más pequeño que el mostrado en las anteriores imágenes, y en su primera línea ya hace referencia a que el Landscape podría parecer una de las famosas imágenes del libro “¿Dónde está Wally? ya que el conjunto de tecnologías bajo el paraguas de Big Data es muy grande. (18)

En el artículo además comenta que Big Data es un término abstracto que es necesario contextualizar para acotarlo, dividiendo la arquitectura Big Data por capas, donde es necesaria una capa de infraestructura sobre la que se desplegaran plataformas de Big Data, otra capa compuestas por distintas tecnologías con la capacidad de capturar datos, procesarlos y almacenarlos. Y simplificándolo la arquitectura Big Data necesitaríamos una capa de explotación y análisis de los datos, y solo acotándolo a estas capas, salen muchas tecnologías. (18)

Enumera las tecnologías más representativas por capas, sin ser ni mucho menos todas las tecnologías existentes y sin detallar la parte de infraestructura que abre otro debate con las soluciones iCloud.

- Captura de datos: Flume, Logstash, Kafka, RabbitMQ. (18)
- Procesamiento de datos: MapReduce, Hive, Pig, Spark, Storm, Flink, Samza.
- Almacenamiento de datos: HDFS, HBase, MongoDB, Cassandra, Neo4J, Elasticsearch, Solr.
- Explotación de datos: Kibana, Grafana, Pentaho.
- Análisis de datos: R, Python, Weka, RapidMiner.

Además comenta que con este mapa tecnológico tan amplio se necesita una figura que tenga experiencia en algunas de las tecnologías y conozca en mayor o menor medida el resto de ellas, y que sea capaz de identificar su aplicabilidad según los diferentes casos de uso que defina el negocio, requiriendo también que los expertos en arquitectura Big Data suelen estar en constante formación de las diferentes actualizaciones en las tecnologías existentes y de las emergentes, requiriendo una continua dedicación y esfuerzo adicional. (18)

Esto se debe a que no hay un estándar que defina o nos ayude a como deberíamos o que deberíamos aplicar según un caso de uso, si no que tenemos que adentrarnos en cada una de las aplicaciones, incluso algunas muy recientes con mucha falta de documentación, para poder discernir o situar las distintas tecnologías en nuestra arquitectura según su funcionamiento y finalidad.

4.8. Data Lake

Antes de entrar en materia con las tecnologías principales dentro de este gran Landscape de BigData, creo que es necesario matizar que, aunque el concepto Datawarehouse procedente de BI sigue existiendo, entra en juego el concepto de Data lake o lago de datos.

Este término es introducido en 2010 por James Dixon, CTO de Pentaho, donde antes de introducirlo el termino, había hablado con muchas compañías que estaban utilizando Hadoop, encontrando los siguientes puntos en común: (19)

- El 80-90% de las compañías tratan con datos estructurados o semiestructurados.
- La fuente de los datos es normalmente una única aplicación o sistema.
- Los datos son típicamente sub-transaccionales o no transaccionales.
- Tienen preguntas que hacerle a los datos
- No saben las preguntas que podrán necesitar en un futuro
- Hay múltiples comunidades de usuarios que tienen preguntas para los datos.
- Los datos son de una escala o volumen diario tal que no se adaptan técnicamente o económicamente en un sistema RDBMS.

En el pasado, la manera estándar de manejar la generación de informes y el análisis de esos datos era identificar los atributos más interesantes y agregarlos en un Datamart(BI tradicional), identificando que de esta manera solo se examina un subconjunto de los atributos, por lo que solo se responden preguntas determinadas y los datos son agregados de manera que se pierda visibilidad en los niveles más bajos, en base a estos problemas define el concepto Data Lake como “Si piensas en un Datamart como una tienda de agua embotellada, limpia, empaquetada y estructurada para un fácil consumo, el Data Lake es un gran cuerpo de agua en un estado más natural. El contenido del lago de datos fluye desde una fuente para llenar el lago y varios usuarios del lago pueden venir a examinar, sumergirse o tomar muestras”. (19)

Cuatro años más tarde, hace otra publicación, en revisión del término debido a que a raíz de la publicación anterior, se estaba extendiendo definiciones erróneas o que no eran del todo correctas, para ampliar esta definición también se apoya en dos videos de YouTube de una de sus conferencias donde explica el termino (20) (21) destacando que un Data lake consiste en una sola fuente de datos pre-agregados, sin tratar, y que muchas empresas solo tienen una fuente de datos que cumple los requisitos. En un Data lake almacenaremos todos los datos (de una sola fuente) ya que no sabemos de antemano todas las preguntas para los datos, destacando el problema de los Datamart y datawarehouse en el que la pre-agregación limita las preguntas que pueden ser realizadas, por eso usando data lake, los datamarts y datawarehouses pueden ser alimentado de agregaciones del lago de datos, pudiendo ser respondidas también las preguntas ad-hoc.

Un data lake no reemplaza una fuente de datos, un Datamart o un datawarehouse, por lo que en conclusión, un solo lago de datos almacena datos de una fuente, pudiendo tener múltiples lagos, pero esto no hace que sean iguales a un data mart o datawarehouse. Destaca también que a se pueden utilizar data warehouse con soluciones Big Data, aunque probablemente no deberías hacerlo (en 2010, aunque actualmente esto no es así, a lo largo del desarrollo de este trabajo, veremos que gracias algunas bondades de soluciones big data superamos problemas del análisis de datawarehouse tradicionales). Comenta también que una gran cantidad de datos de uno o dos sistemas no son un data warehouse, en su mejor aproximación será un Datamart.

Por último, destaca que un data lake no es un data warehouse almacenado en Hadoop, si almacenamos datos de muchos sistemas y los cruzamos, entonces tendremos un jardín de agua no un lago de datos. Comenta además que eligió con mucho cuidado el termino Data lake y prestando atención a la analogía y la metáfora, pero aun así hoy hay gente usando el termino sin prestar mucha atención, haciendo referencia a que en caso de que hagamos un lago de datos con todos los datos de una empresa, deberemos usar bases de datos transaccionales para propósitos transaccionales, bases de datos analíticas para propósitos analíticos y encontrar la mejor herramienta para una determinada situación, encontrándonos seguramente híbridos con diferentes almacenamiento de datos (22) . También en otro artículo al que no voy a entrar, podemos encontrar los casos de uso de un data lake. (23)

4.9. Hadoop

Ante la falta de definición de un estándar que nos indique que usar y como usarlo, voy a empezar a hacer un estudio de las tecnologías más relevantes de las que disponemos en Big Data desarrolladas por apache, es decir open source desarrolladas por la comunidad.

Hadoop es un sistema de código abierto utilizado para almacenar, procesar y analizar grandes volúmenes de datos, cientos de terabytes, petabytes o incluso más.

Es un proyecto de Apache.org y surge como iniciativa open source (software libre) a raíz de la publicación de varios *papers* de Google sobre sus sistemas de archivo (Google File System GFS), su herramienta de mapas y el sistema BigTable Reduce. Como resultado nació un conjunto de soluciones en el entorno de apache. De GFS nace Hadoop Distributed File system(HDFS), Apache MapReduce de la publicación del nuevo modelo de programación que simplifica el procesamiento de grandes volúmenes de datos y de BigTable aparece Apache HBase, estas tres tecnologías son las que se conocen como Hadoop, con herramientas como Sqoop (para importar datos estructurados en Hadoop clúster) o NoSQL (para realizar el análisis de los datos no estructurados) entre otras. (24)

Hadoop se trata de una librería de software y un marco (framework) que permite el procesamiento distribuido de grandes conjuntos de datos, generalmente conocidos como Big Data, a través de miles de sistemas convencionales que ofrecen potencia de procesamiento y espacio de almacenamiento, es decir, procesamiento distribuido sobre un clúster. Hadoop es en esencia, el diseño más poderoso en el espacio de los analíticos de Big Data. (25)

En la creación de su framework participan varios módulos y entre los principales podemos encontrar los siguientes:

- **Hadoop Common** (Utilidades y librerías que dan soporte a otros módulos Hadoop).
- **Hadoop Distributed File Systems (HDFS)**, sistema de distribución de ficheros en clúster.
- **Hadoop YARN** (Yet Another Resource Negotiator), tecnología de administración de clústeres.
- **Hadoop MapReduce** (modelo de programación que soporta la computación paralela masiva).

Si bien los cuatro módulos arriba citados componen el núcleo central de Hadoop, existen otros más. Entre ellos, están Ambari, Avro, Cassandra, Hive, Pig, Oozie, Flume y Sqoop. Todos ellos sirven para ampliar y extender la potencia de Hadoop e incluirse en aplicaciones big data y procesamientos de grandes conjuntos de datos. A continuación, podemos ver una imagen de un ecosistema Hadoop. (26)

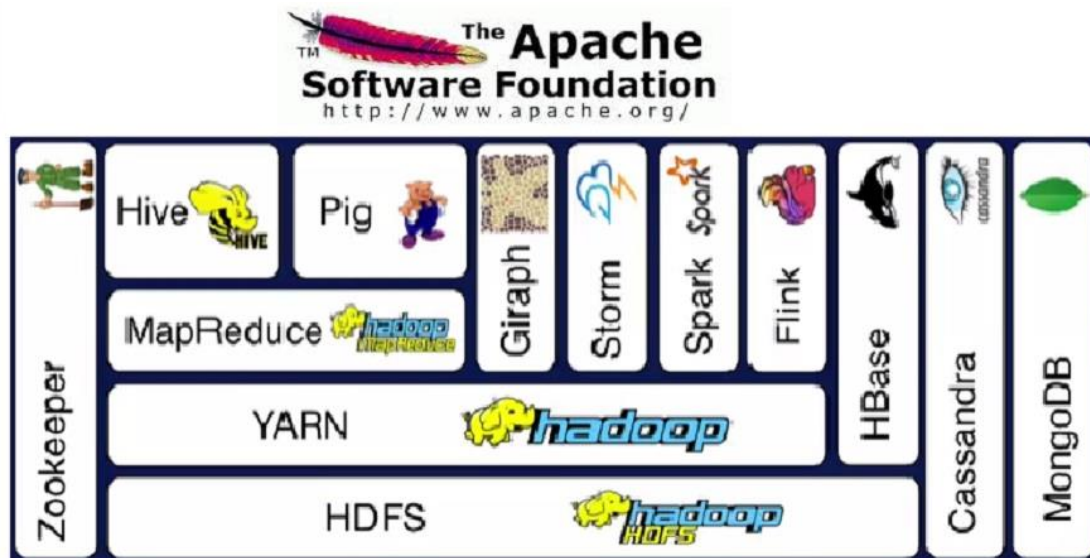


Figura 16. Ecosistema Hadoop. Apache.org

Muchas compañías utilizan Hadoop para sus grandes conjuntos de datos y analíticos, por ejemplo, la NASA (15), Facebook, Amazon o Yahoo! (27) este último con un clúster de 4500 nodos(2x4cpu boxes, 4*1TB HDD y 16GB de RAM), convirtiéndose en el estándar de todas las aplicaciones big data.

Fue originalmente diseñado para manejar funciones de crawling (exploración de sitios Web) y realizar la búsqueda de millones de páginas Web mientras recolectaba información en una base de datos. El resultado de ese deseo de explorar y buscar en la Web terminó siendo Hadoop HDFS y su motor de procesamiento distribuido, MapReduce. (26)

Hadoop resulta útil para las empresas cuando los conjuntos de datos son tan grandes y tan complejos que las soluciones con las que ya cuentan (BI tradicional) no pueden procesar la información de forma efectiva y en lo que las necesidades del negocio definen como tiempos razonables mientras que MapReduce es un excelente motor de procesamiento de textos y eso se debe a que tanto el crawling como la búsqueda en la Web, sus primeros desafíos, son tareas basadas en texto. (26)

Entrando un poco más en el ecosistema y centrándonos en sus principales tecnologías encontraremos como ya he mencionado, HDFS, HBase y MapReduce.

4.9.1. HDFS (Hadoop Distributed File System)

HDFS es un sistema de ficheros distribuido, escalable y portátil escrito sobre Java y diseñado para trabajar con ficheros de gran tamaño. Una de sus principales características es un tamaño

de bloque muy superior al habitual, por defecto es 64 MB, pero puede ser ampliado hasta 128 o 256MB, para no perder tiempo en los accesos de lectura. Los ficheros que normalmente van a ser almacenados o ubicados en este tipo de sistema de ficheros siguen el patrón “Write once read many” (escribe una vez y lee muchas), de este modo, está especialmente indicado para procesos *batch* de grandes ficheros de logs o a los datos recogidos de una sonda espacial que serán escritos una vez y leído muchas veces para poder extraer información y analizar su contenido. Igualmente, se puede aplicar este concepto a bases de datos orientadas al almacenamiento masivo de datos como HIVE o Cassandra que entrare más adelante. (28) (29) (30)

Hadoop se basa en una arquitectura Maestro/Esclavo con tipos de nodos: nodo máster (maestro) y los nodos Slave (esclavos). Un clúster Hadoop tiene un sólo nodo máster y varios nodos Slave.

4.9.1.1. Nodo Máster

Es el encargado de almacenar el metadato asociado a sus nodos Slave dentro del rack del que forma parte.

El nodo máster es el responsable de mantener el estatus de sus nodos Slave, estableciendo uno de ellos como nodo pasivo, que se convertirá en nodo máster, si por cualquier motivo el nodo master se quedara bloqueado.

Uno de los problemas que tiene Hadoop es que a veces el nodo pasivo no está sincronizado con el nodo máster original, al asumir las funciones de éste dentro del proceso.

Los servicios que proporciona son los siguientes:

- **NameNode:** Encargado de administrar el almacenamiento HDFS.
- **SecondaryNameNode:** Encargado de realizar los checkpoints (puntos de control) periódicos del NameNode.
- **ResourceManager:** Supervisa la programación de tareas y gestiona los recursos del clúster Hadoop. (29)

4.9.1.2. Rack

En Hadoop se denomina rack a la combinación de “nodos de datos”. Un rack puede tener máximo de 40 nodos máster. Cada rack tiene un switch que le permite comunicarse con los distintos racks del ecosistema, sus nodos y procesos cliente. (24)

4.9.1.3. Proceso cliente

Un proceso cliente es un proceso que se lanza a petición de un nodo máster, ya sea para almacenamiento de archivo nuevo(entrada) o recuperación de un archivo en el clúster Hadoop(salida). El nodo máster se comunica directamente con el proceso cliente y actúa según el tipo de petición que este le realiza. (24) (28)

En la siguiente imagen vemos un escenario sobre los planes de ejecución para una transformación JOIN.

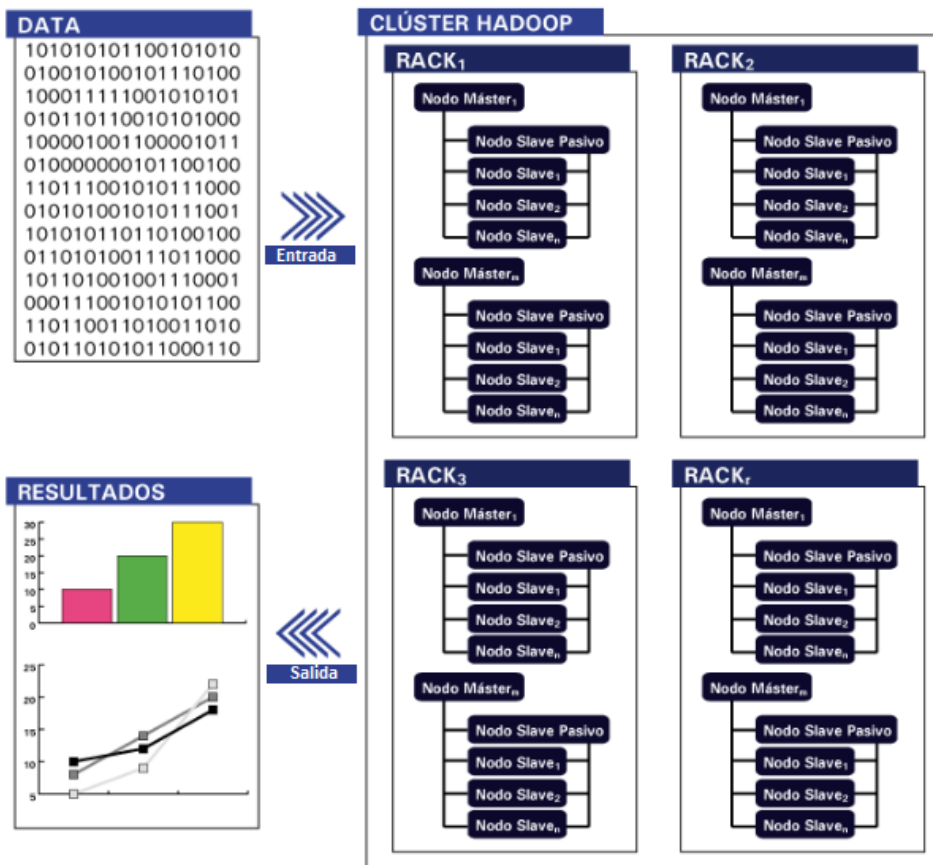


Figura 17. Planes de ejecución para un Join. Momentotic.com

A grandes rasgos para poder montar un sistema Hadoop, una vez hecha una configuración previa en la que definamos sus variables de entorno, necesitaremos crear los distintos servidores que conformaran nuestro clúster, generalmente uno por nodo, es decir una IP por cada nodo, independientemente de que sea master o Slave y estableceremos una relación de confianza ssh por seguridad. Instalaremos Hadoop en cada uno de los nodos, y pasaríamos a configurar cada uno de los nodos para que cumpla la función asignada dentro de este ecosistema. Podemos encontrar más información sobre la creación de un clúster, en la página oficial de Hadoop. (25)

4.9.2. MapReduce

El concepto MapReduce lo introdujo Google en 2004 en el paper "MapReduce: Simplified Data Processing on Large Clusters". El objetivo principal de MapReduce era permitir la computación paralela sobre grandes colecciones de datos permitiendo abstraerse de los grandes problemas de la computación distribuida. (13)

MapReduce consta de 2 fases: Map y Reduce que se aplican sobre pares de datos (clave, valor).

Map toma como entrada un par (clave, valor) y devuelve una lista de pares (clave2, valor2). Esta operación se realiza en paralelo para cada par de datos de entrada. Luego el framework MapReduce (como Hadoop MapReduce) agrupa todos los pares generados con la misma clave de todas las listas, creando una lista por cada una de las claves generadas.

Reduce se realiza en paralelo tomando como entrada cada lista de las obtenidas en el Map y produciendo una colección de valores. (13) (25) (29)

El ejemplo más simple es “cuenta palabras”.

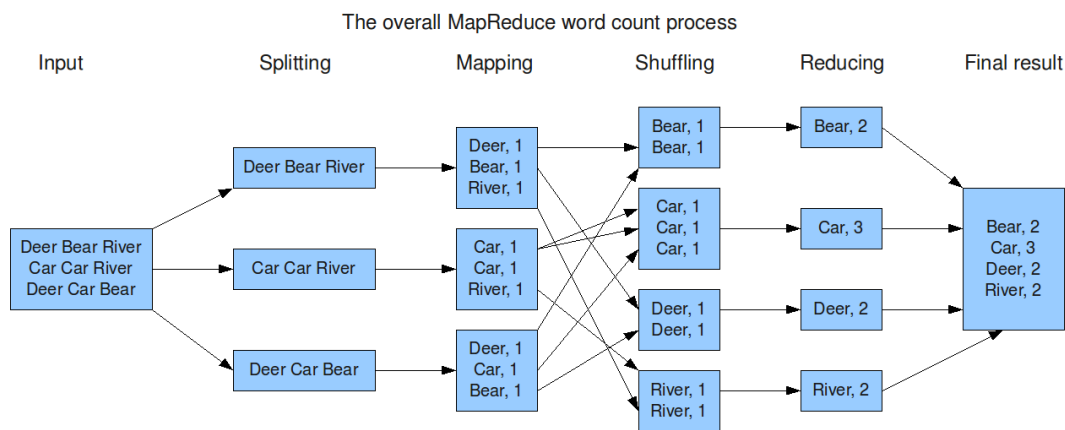


Figura 18. MapReduce count Word. cs.calvin.edu

4.9.3. HBase

HBase es una base de datos distribuida NoSQL modelada a partir de Google BigTable y escrita en Java. Su desarrollo forma parte del proyecto Hadoop de Apache y se ejecuta sobre HDFS proporcionando capacidades al estilo de BigTable para Hadoop. Es decir, proporciona una forma tolerante a fallos de almacenar grandes cantidades de datos dispersos (pequeñas cantidades de información inmersas dentro de una gran colección de datos inexistentes o poco significativos, tales como la búsqueda de mayor gasto en una factura sobre 5.000 millones de registros o la búsqueda de un elemento concreto que represente menos del 0,1% de una enorme colección. (31)

Además, incluye operaciones de compresión en memoria, y filtro de Bloom (Howard Bloom,1970), que es una estructura de datos probabilística usada para verificar si un elemento es miembro de un conjunto, en base a cada columna como se propone en el artículo original sobre BigTable. Las tablas en HBase pueden servir como entrada o salida para tareas MapReduce en Hadoop. Es un almacén de datos orientado a columnas de tipo clave-valor. HBase se ejecuta sobre HDFS y es adecuado para acelerar operaciones de lectura y escritura en los grandes conjuntos de datos con un alto rendimiento y una baja latencia de entrada/salida. (31)

4.9.4. Características de Hadoop

Hadoop es capaz de almacenar toda clase de datos: estructurados, no estructurados, semiestructurados; archivos de registro, imágenes, video, audio, comunicación, etc.

Por otra parte, Hadoop también destaca por tener una arquitectura con capacidad de asegurar una alta disponibilidad y recuperación de los datos que ingesta. Para realizar esta labor eficientemente se basa en su característica de “Replication”, la cual permite una alta

disponibilidad de los datos en Hadoop, esto es posible debido a que lleva implícita la replicación de datos en un clúster Hadoop. (25)

Un bloque de archivo se replica en varios “nodos de datos” en función del “factor de replicación” del clúster Hadoop, que podría ser 1, 2, 3... Un factor de replicación de 1 indica que un bloque de archivo residirá en un sólo “nodo de datos”. Un factor de replicación de 2 indica que un bloque de archivo residirá en dos “nodos de datos”, ya sea dentro del mismo rack o en uno que esté físicamente a miles de kilómetros de distancia, etc. (25)

Aunque HDFS utiliza HBase por defecto, podemos sustituir o complementar este componente con Cassandra o MongoDB. (25)

Un último punto importante a destacar sobre Hadoop, es la alta interoperabilidad que tiene, almacenando datos de toda clase de archivos, ya sean estructurados o no estructurados, y una alta integración con otras herramientas como Ambari que proporciona, gestiona y monitoriza clúster de Hadoop, Avro es un sistema de serialización de datos, Cassandra como base de datos NoSQL, Hive como almacén de datos, Pig para análisis de datos mediante lenguaje de alto nivel o Spark como motor general y rápido para el procesamiento de grandes cantidades de datos. (25)

4.10. Cassandra

El desarrollo inicial de Cassandra tiene su origen en Facebook, el cual lo diseñó para potenciar la funcionalidad de búsqueda, siendo en 2008 liberado como proyecto open source, convirtiéndose en 2010 como proyecto de la fundación Apache. Está inspirado e influenciado por los papers de Amazon Dynamo de 2007 y de Google BigTable de 2006. (32)

Apache Cassandra es una base de datos NoSQL distribuida, es decir, que la información está repartida en los nodos de un clúster, basada en un modelo de almacenamiento de clave-valor, de código abierto y escrito en Java, permitiendo el almacenamiento de grandes volúmenes de datos de forma distribuida, ofreciendo una alta disponibilidad de manera que si alguno de los nodos se cae el servicio no. Por ejemplo, es la tecnología utilizada por Twitter para su plataforma. (33)

Escala linealmente y de forma horizontal, es decir, linealmente en rendimiento respecto al número de nodos que añadamos, por ejemplo, si con 1 nodo soportamos 50.000 operaciones por segundo, con 2 nodos soportaremos 100.000. Y de forma horizontal se refiere a la manera de escalar el sistema añadiendo nuevos nodos hardware de bajo coste. (32)

La arquitectura distribuida de Cassandra está basada en una serie de nodos iguales que se comunican con un protocolo P2P con lo que la redundancia es máxima, proporcionando tolerancia a particiones y disponibilidad a cambio de ser eventualmente, pudiendo ser configurado incluso a nivel de Query, según las necesidades. (32)

También ofrece soporte para múltiples centros de datos con una comunicación peer-to-peer eliminando los puntos de fallo único sin la necesidad de un nodo maestro, permitiendo operaciones de baja latencia para todos los clientes, y que cualquier nodo pueda ser el

coordinador para una Query, y agrupa los nodos lógicamente para tener los datos más cerca del usuario. (32)

Cassandra también ofrece un gran rendimiento. En 2012, investigadores de la Universidad de Toronto que estudian los sistemas NoSQL concluyeron que "En términos de escalabilidad, hay un claro ganador a través nuestros experimentos. Cassandra logra el más alto rendimiento para el número máximo de nodos en todos los experimentos", aunque "esto tiene como precio una alta latencia de escritura y lectura". (34)

Cassandra, al igual que Hadoop, reparte los datos a lo largo de un clúster, con la política de replicación que le configuremos y dispone de su propio lenguaje de acceso a datos, Cassandra Query Language(CQL), donde los datos están desnormalizados, por lo que el concepto de joins o subqueries desaparece frente a bases de datos relacionales. (32)

En cuanto a modelado de datos, combina propiedades de una base de datos clave-valor y una orientada a columnas, organizando la información de manera que toda la fila tiene una clave única y una serie de pares de clave-valor para las columnas, siendo importante tener esto en mente a la hora de diseñar el modelo de datos, y siendo necesario hacer un análisis de las queries que se pretenden utilizar de manera de que se pueda diseñar un modelo más eficiente y sacar todo el rendimiento a Cassandra. También se deben considerar cual va a ser el patrón de consulta intentando minimizar el número de particiones a las que se van a acceder durante su lectura para evitar cuellos de botella. (32)

En conclusión, aunque Cassandra es una gran solución para muchos casos de uso, no es adecuado para alojar un datawarehouse tradicional, ya que hay que tener claro desde el principio los casos de uso y el tipo de consultas que se realizaran para diseñar una base de datos coherentemente, y obtener el máximo rendimiento en grandes volúmenes de datos.

4.11. HBase vs Cassandra

Al contrario de lo que pueda parecer, a la hora de comparar estas dos tecnologías, es más propio comparar HBase con Cassandra y no Hadoop con Cassandra, puesto que Hadoop está compuesto de tres tecnologías fundamentales (HDFS, MapReduce y HBase), por lo que acorde a su funcionalidad, la parte de Hadoop similar a Cassandra es HBase. (33) (35)

HBase como he mencionado anteriormente es una base de datos distribuida NoSQL que se ejecuta sobre HDFS de Hadoop, siendo su punto fuerte la alta consistencia, es decir, cualquier escritura se reflejara inmediatamente en lecturas secuenciales, siendo utilizado HBase para lecturas/escrituras escalables sobre HDFS. (33) (35)

Cassandra es también una base de datos distribuida NoSQL, a diferencia de HBase, Cassandra tiene como punto fuerte la alta disponibilidad y una administración mínima, es decir, si algún nodo cae, Cassandra garantiza la disponibilidad continua de los datos, siendo su punto fuerte también las lecturas/escrituras rápidas en un entorno distribuido. (33) (35)

Aunque distintos, hay muchas similitudes entre HBase y Cassandra, ambos son almacenes de datos distribuidos NoSQL con punto fuerte las lecturas/escrituras escalables, ambos con una escalabilidad lineal mejorando su capacidad de almacenamiento y rendimiento directamente

proporcional a la cantidad de nodos en el clúster. Y tanto HBase como Cassandra se apoyan en la replicación para evitar la pérdida de datos, y siendo tolerante a fallos. (33) (35)

Pero, aunque Cassandra y HBase tienen un rol similar en un ecosistema Hadoop, hay algunas diferencias que los distinguen. HBase se basa en un nodo maestro para manejar las funciones administrativas y asignación de las regiones. Estos servidores regionales administran los nodos de datos realizando lecturas/escrituras sobre HDFS. HBase además usa un Zookeeper para coordinar el estado y las operaciones del servidor, es decir, si está disponible y quien accede y que ejecuta. Cassandra por el contrario almacena datos fuera del HDFS y del clúster de Hadoop, a diferencia con HBase todos los nodos en Cassandra comparten un rol similar, tiene una arquitectura descentralizada y no existe el rol Master-Slave, si no múltiples semillas, pudiendo cualquier nodo realizar cualquier operación. (33) (35)

Acuerdo a su rendimiento dependiendo el caso de uso, es cuando vamos a encontrar las diferencias, HBase se prefiere para lecturas a gran escala, trabajando con MapReduce y el procesamiento por lotes, mientras que Cassandra es buena en lecturas de una sola fila y esta optimizada para escrituras, ofreciendo un rendimiento uniforme en entornos tolerantes a fallos y siendo un componente ideal para el procesamiento de transacciones ofreciendo una disponibilidad continua a miles de usuarios concurrentes y un rendimiento uniforme en un entorno tolerante a fallos, haciendo que Cassandra sea el componente ideal para el procesamiento de trabajos de naturaleza transaccional proporcionando disponibilidad continua y tiempo de actividad a miles de usuarios, y un análisis en caliente generados por aplicaciones web, móviles y “Internet of Things”(IoT). (33) (35)

4.12. MongoDB

El desarrollo de MongoDB empieza en octubre de 2007 por la compañía de software 10gen, hoy en día es una base de datos NoSQL orientada a documentos muy diferente al concepto de bases de datos relacionales, desarrollada bajo open source. (36)

Pero ¿que entendemos por NoSQL? Significa que en vez de guardar los datos en tablas relacionales como se hacen en las bases de datos tradicionales, MongoDB utiliza estructuras de datos en documentos tipo JSON almacenados en BSON (una representación binaria de JSON). En MongoDB no tenemos tablas ni registros, ni tenemos SQL. (36)

Una de las principales diferencias con las bases de datos relacionales, es que no es necesario definir un esquema, los documentos de una misma colección (colección sería similar al concepto tabla de una base de datos relacional) pueden tener esquemas diferentes. (37)

Un ejemplo de que tipo de documentos almacena MongoDB en una colección sería el siguiente:

```
{
  Nombre: "Alejandro",
  Apellidos: "Reina Reina",
  Edad: 27,
  Aficiones: ["BigData", "Business Intelligence", "Viajar"],
}
```


Frente a otro documento como el siguiente con otro formato(campos).

```
{
  Nombre: "Antonio"
  Estudios: "Grado Ingeniería Informática",
  Amigos: 15
}
```

Como hemos podido ver, este último documento no sigue el mismo modelo que el primero, tiene menos campos y campos nuevos que no existen en el anterior, algo que en las bases de datos relacionales es impensable, es totalmente valido en MongoDB.

MongoDB está escrito en C++ y las consultas se realizan pasando objetos JSON como parámetro, como en el siguiente ejemplo:

```
db.Clientes.find({Nombre:"Alejandro"});
```

Pero, ¿hasta dónde llega la usabilidad o finalidad de MongoDB? O, dicho de otra manera, ¿en qué proyectos o casos de uso podemos utilizarlos? En principio, MongoDB puede ser utilizado en cualquier aplicación que necesite almacenar datos semiestructurados, como es el ejemplo de aplicaciones CRUD o desarrollos web actuales. Pero, aunque las colecciones de MongoDB no necesitan definir un esquema, es importante diseñar la aplicación para que siga un esquema, pensando también si es necesario normalizar los datos, desnormalizarlos o utilizar una aproximación entre ambas, ya que si afectara al rendimiento de la aplicación. Dicho de otra manera, el esquema lo definirán las consultas que se vayan a realizar con más frecuencia con el objetivo de optimizarlas. (37)

También dispone de una gran escalabilidad, con sus opciones de replicación y sharding, pudiendo conseguir que escale de manera horizontal sin muchas dificultades.

Es importante destacar que MongoDB es una base de datos donde no existen las transacciones, aunque la aplicación utilice alguna técnica para simularlo, como por ejemplo mantener los datos en cache hasta que todos hayan sido recibidos antes de enviar la petición a la base de datos, MongoDB no tiene esta capacidad de manera nativa. Solo garantiza las operaciones atómicas a nivel de documento. (37)

Además en MongoDB no existe la operación JOIN, para consultar datos relacionados de varias colecciones es necesario hacer más de una consulta, y aunque dispone de un framework para realizar consultas de agregación llamado Aggregation Framework, y puede utilizar MapReduce, estos métodos no llegan a la potencia de un sistema relacional, por lo que si vamos a explotar infórmenes complejos y muchas relaciones por lo que será mejor utilizar otro sistema, aunque esto es algo en lo que MongoDB está mejorando en cada versión y es posible que en un futuro cercano esto deje de ser un problema. (37)

4.13. Hive

El software Apache Hive data warehouse facilita lectura, escritura y manejar grandes dataset alojados en almacenamientos distribuidos y consultados usando sintaxis SQL. Construido sobre Hadoop, provee las siguientes características:

- Herramienta para facilitar el acceso a los datos vía SQL lo que permite tareas de datawarehousing como los ETL, reporting y análisis de datos.
- Un mecanismo de imponer estructuras sobre una variedad de formatos de datos.
- Acceso a archivos almacenados directamente en HDFS u otro almacenamiento de datos como HBase.
- Ejecución de queries vía Apache Tez, Apache Spark o MapReduce.
- Lenguaje procedural con HPL-SQL

Hive provee funcionalidades del estándar SQL incluyendo muchas características posteriores de SQL2003 y SQL2011 para análisis. El SQL de Hive también se puede ampliar con código de usuario a través de funciones definidas por el usuario(UDF), agregaciones definidas por el usuario(UDAFs) y funciones de tabla definidas por el usuario(UDTF). (38)

No hay un solo formato de Hive en el cual debas almacenar los datos, ya que viene con conectores para ficheros (CSV/TSV), o ficheros de texto, Apache Parquet (39) y otros formatos, pudiéndose extender también los conectores para otros formatos. (38)

Hive no está diseñado para cargas de trabajo de tipo OLTP (On-Line Transaction Processing), es mejor para tareas de almacenamiento de datos tradicionales y diseñado para maximizar la escalabilidad (escala añadiendo dinámicamente más maquinas al clúster de Hadoop), rendimiento, extensibilidad, tolerancia a fallos y bajo acoplamiento con los formatos de entrada. (38)

Hive también incluye entre sus componentes HCatalog y WebHCat:

- HCatalog es una tabla y una capa de administración de almacenamiento para Hadoop que permite a los usuarios el procesamiento de datos con diferentes herramientas como Pig, MapReduce para facilitar la lectura y escritura de los datos.
- WebHCat provee un servicio para correr Hadoop MapReduce (o YARN), Pig, trabajos Hive o realizar operaciones sobre los metadatos de Hive usando una interfaz HTTP. (38)

4.13.1. Modelo de datos

Los datos en Hive son organizados en:

- **Tablas:** Estas son análogas a tablas de bases de datos relacionales. Cada Tabla tiene su correspondencia en el directorio HDFS. Los datos en las tablas son serializados y almacenados en archivos dentro de ese directorio. Los usuarios pueden asociar tablas con formato de serialización de los datos subyacentes. Hive provee de formatos integrados que explotan la

compresión y la deserialización lenta. Los usuarios pueden agregar soporte para nuevos formatos de datos definiendo una serialización personalizada y métodos deserializados llamados SerDe's escritos en Java (40). El formato de serialización para cada tabla es almacenado en un sistema de catálogos y es automáticamente usado por Hive durante la compilación y ejecución de una Query. También añade soporte para tablas externas sobre datos almacenados en HDFS, NFS o directorios locales. (38)

- **Particiones:** Cada tabla puede tener una o más particiones que determinan la distribución de los datos dentro de subdirectorios de un directorio de tablas. con una determinada distribución de los datos dentro de subdirectorios. Supongamos que los datos de una tabla T están en el directorio *"/ejemplo/T"*, si la tabla T esta particionada sobre columnas id y país, entonces los datos con un determinado valor *"id"* por ejemplo 20090101 y un valor *"pais"* como *"US"*, podrá ser almacenada en archivos dentro del directorio *"/ejemplo/T/id=20090101/pais=US"*. (38)
- **Cubos:** Los datos dentro de cada partición pueden a su vez dividirse en cubos basados en el hash de una columna en la tabla. Cada cubo es almacenado en un archivo dentro de un directorio particionado. (38)

4.14. Spark

Spark es la otra estrella de la corona en cuanto a Big Data se refiere. Apache Spark combina un sistema de computación distribuida a través de un clúster, creado en la universidad de Berkely (California), es considerado el primer software de código abierto que hace la programación distribuida realmente accesible a los científicos de datos. Sus desarrolladores lo presentan como "un motor general y rápido para el procesamiento de datos en gran escala (Ken Hess, Feb 2016)". (26)

Es sencillo entender Spark si lo comparamos con su predecesor, MapReduce, el cual revolucionó la manera de trabajar con grandes conjuntos de datos ofreciendo un modelo relativamente simple para escribir programas que se podían ejecutar paralelamente en cientos y miles de máquinas al mismo tiempo. Gracias a su arquitectura, MapReduce logra prácticamente una relación lineal de escalabilidad, ya que si los datos crecen es posible añadir más máquinas y tardar lo mismo. (26)

Spark también puede realizar procesamiento batch, sin embargo, su performance se luce cuando se trata de cargas tipo streaming, consultas interactivas y lo que denominamos aprendizaje basado en máquinas (Machine Learning), ya que tiene un sistema avanzado de ejecución de DAG (Directed Acyclic Graph) que soporta flujos de datos cíclicos y cálculo en memoria, haciendo ambas características que Spark sea muy rápido. (26)

La gran consigna de Spark es la de su capacidad de procesamiento de datos en tiempo real, lo que compara con el desempeño de MapReduce basado en discos como motor de procesamiento batch (en lotes). (26)

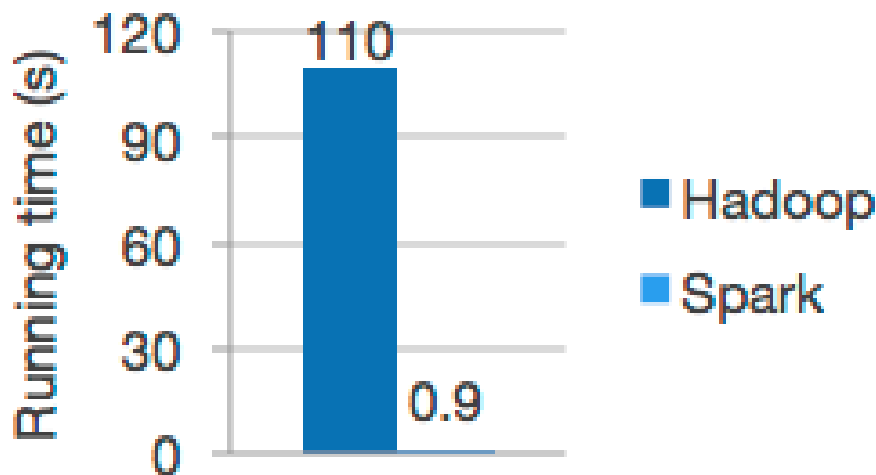


Figura 19. Spark vs Hadoop. spark.apache.org

Es un framework de computación en clúster, aunque no tiene su propio sistema de archivos distribuidos, puede usar HDFS. Spark utiliza memoria y puede usar también discos para el procesamiento de datos, mientras que MapReduce es por el momento una tecnología basada estrictamente en discos. La principal diferencia entre MapReduce y Spark consiste, en que MapReduce utiliza almacenamiento persistente y Spark utiliza RDDs (Resilient Distributed Datasets). (26)

4.14.1. Características de Spark

A continuación voy a detallar cuales son las principales características que hacen a Spark una gran herramienta para el mundo del Big Data.

4.14.1.1. Performance

La razón por la que Spark es tan veloz es porque procesa todo en memoria. También puede apoyarse en discos físicos cuando todos los datos que está procesando no caben en la memoria RAM. (26)

El procesamiento in-memory de Spark permite contar con analíticas en tiempo casi real. Esto es útil para campañas de marketing, aprendizaje de máquinas, sensores de IoT (Internet of Things), monitoreo de actividad de logs, analíticos de seguridad y sitios de medios sociales. (26)

4.14.1.2. Facilidad de Uso

Spark ha ganado reconocimiento por su velocidad y también es algo conocida su facilidad de uso gracias a su API (Application Programming Interface) amigable para SCALA, su lenguaje nativo y también para Java, Python, R y Spark SQL, aunque esto abre un debate en el que entrare más adelante. Spark SQL es muy parecido a SQL 92, de modo que prácticamente no existe una curva de aprendizaje para utilizarlo. (26)

Spark también tiene un modo interactivo con el que tanto desarrolladores como usuarios por igual pueden tener respuestas inmediatas para consultas u otras acciones. (26)

4.14.1.3. Costos

Es un proyecto de Apache.org. Esto significa que son productos open source de software libre. Si bien no existe costo a la hora de adquirir el software, si existen costos asociados a la utilización de la plataforma en términos de personal y hardware. Los dos productos han sido diseñados para funcionar en hardware estándar o commodity, tal como lo son los sistemas llamados de caja blanca (compatibles) o servidores de bajo costo, al igual que otras tecnologías como HDFS, Cassandra entre otros. (26)

Por otra parte, Spark requiere enormes cantidades de memoria, pero puede funcionar con espacios de disco considerados estándar en términos de capacidad y velocidad. Algunos usuarios, se han quejado de la sobrecarga que representa la limpieza de los archivos temporales. Estos archivos temporales generalmente se guardan por siete días con el objeto de acelerar el eventual procesamiento con conjuntos de datos ya utilizados. El espacio de disco es de menor costo y como Spark no utiliza el Input/Output de disco para procesar, el espacio de disco utilizado puede ser bien aprovechado tanto en esquemas SAN (Storage Area Networks) como NAS (Network Attached Storage). (26)

Lo cierto es que los sistemas para Spark cuestan más porque necesitan grandes cantidades de memoria RAM. Pero también es cierto que la tecnología de Spark reduce la cantidad de sistemas para funcionar. Por eso, se termina teniendo una menor cantidad de sistemas que cuestan más. Posiblemente haya un punto de inflexión el que Spark reduzca los costos por unidad de computación aun con la memoria RAM adicional que requiere. (26)

4.14.1.4. Compatibilidad

MapReduce y Spark son ambos compatibles entre sí y Spark comparte todas las compatibilidades de MapReduce para fuentes de datos, formatos de archivos y herramientas de BI (Business Intelligence) vía JDBC (Java Database Connectivity) y ODBC (Open Database Connectivity). (26)

4.14.1.5. Seguridad

La seguridad de Spark es un poco escasa en términos de recursos. Actualmente sólo soporta autenticación vía password. Si se corre Spark sobre HDFS, se pueden usar los permisos HDFS ACL y a nivel de archivo. Además, Spark puede utilizar a YARN para alcanzar la capacidad de utilizar autenticación Kerberos. (26)

4.14.1.6. Procesamiento de Datos

MapReduce es un motor de procesamiento batch. Opera en etapas secuenciales leyendo los datos del clúster, realizando sus operaciones en los datos y escribiendo los datos nuevamente en el clúster; leyendo los datos actualizados en el clúster, realizando la siguiente operación de datos, grabando esos resultados nuevamente en el clúster y así sucesivamente. Spark realiza similares operaciones, pero lo hace en un mismo paso y trabajando en memoria. Lee los datos del clúster, hace las operaciones con los datos y luego graba los resultados en el mismo. (26)

Spark también incluye su propia librería de computación Graph: GraphX permite que los usuarios puedan ver los mismos datos como graphs y como colecciones. También pueden transformar y unir graphs con Resilient Distributed Datasets (RDDs). (26)

4.14.1.7. Tolerancia a Fallos

Spark utiliza RDDs que son colecciones de elementos con tolerancia a fallos y que pueden ser operados en paralelo. Los RDDs pueden referenciar a un conjunto de datos en un sistema externo de almacenamiento, tal como un file system compartido, HDFS, HBase o cualquier otra fuente de datos que tenga el InputFormat de Hadoop. Spark puede crear RDDs desde cualquier fuente de almacenamiento soportada por Hadoop, incluyendo file systems locales u otros de los antes mencionados. (26)

Un RDD cuenta con las siguientes propiedades:

- Una lista de particiones
- Una función de computar cada Split
- Una lista de dependencias con otros RDDs

Opcionalmente, un Particionador para los RDDs con valor de clave y una lista de ubicaciones preferenciales para computar cada Split (por ejemplo, en un block de archivo HDFS). (26)

Los RDDs puede ser persistentes y guardar un conjunto de datos en cache durante las operaciones. Esto permite que las futuras acciones sean mucho más fáciles. El caché de Spark es tolerante a fallos en cuanto si alguna partición o RDD se pierde, automáticamente será reprocesada utilizando las transformaciones originales. (26)

4.14.1.8. Escalabilidad

Spark es escalable utilizando HDFS. La pregunta es qué tan grande puede llegar a ser un clúster Hadoop, por ejemplo. Tenemos el caso de Yahoo!, que tiene un clúster Hadoop de 44.000 nodos, por lo que no se puede decir que exista un límite. Por el lado de Spark trabajando en el modo standalone, por ejemplo, Facebook tiene dos clústers, uno de 1100 máquinas, con 8800 nodos y cerca de 12 PB de almacenamiento y otro clúster de 300 máquinas con 2400 nodos y cerca de los 3PB de almacenamiento. (26) (27)

4.14.2. Librerías importantes en Spark

Al igual que MapReduce en su día, Spark surge como un framework base sobre el cual se van desarrollando cada vez más aplicaciones y más avanzadas, como vemos en la siguiente imagen, entre las que destacaremos las siguientes:

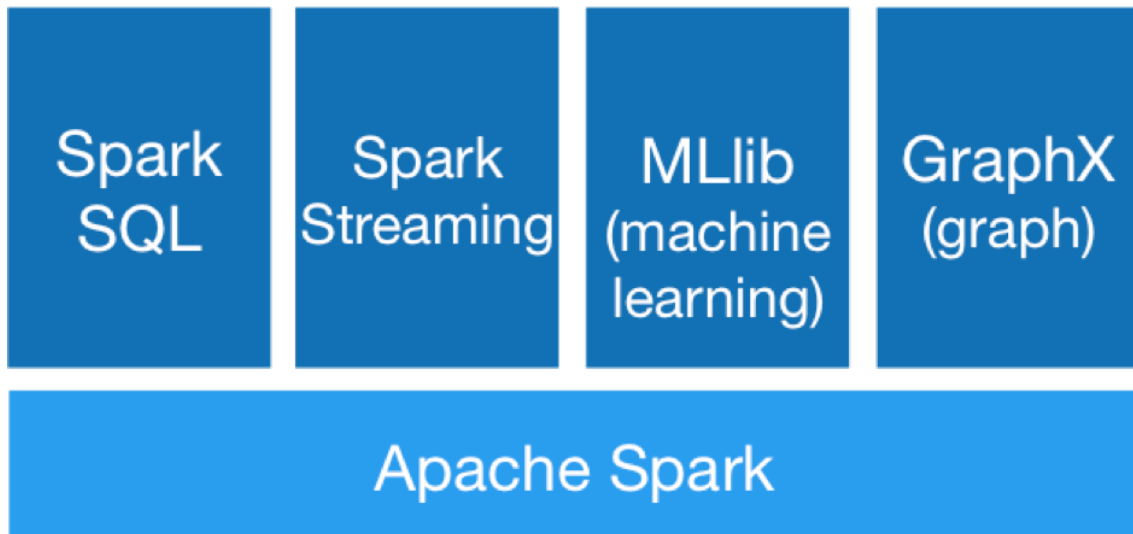


Figura 20. Librerías Spark. BBVAopen4u

4.14.2.1. Spark SQL

Spark SQL es una librería de Spark que añade una abstracción de datos llamada DataFrames, que proporciona soporte para datos estructurados y semiestructurados. Spark SQL está habilitado para manipular DataFrames en Scala, Java, Python o R. También provee de soporte de lenguaje SQL, con interfaces de línea de comandos y servidor ODBC / JDBC. (41)

4.14.2.2. MLlib

MLlib es una biblioteca básica de Spark que proporciona varias utilidades que ayudan en las tareas de aprendizaje automático (Machine Learning), incluye utilidades que son adecuadas para:

- Clasificación
- Regresión
- Agrupación en clúster
- Modelado de tema
- Descomposición en valores singulares (SVD) y Análisis de los componentes principales (PCA).
- Comprobación de hipótesis y cálculo de estadísticas de ejemplo. (42)

4.14.2.3. GraphX

GraphX es una librería de procesamiento de gráficos distribuido. Al basarse en RDD, los gráficos son inmutables y por lo tanto GraphX no es adecuado para los gráficos que necesitan ser actualizados. A diferencia de su predecesor Bagel, que fue formalmente obsoleto en Spark 1.6, GraphX tiene soporte completo para gráficos de propiedades (gráficos donde las propiedades pueden ser adjuntadas a bordes y vértices).

Al igual que Apache Spark, GraphX inicialmente comenzó como un proyecto de investigación en AMPLab y Databricks de UC Berkeley, y más tarde fue donado a Apache Software Foundation y al proyecto Spark. (43)

4.14.2.3.1. Spark Streaming

Spark Streaming puede ingerir datos de una amplia variedad de fuentes, incluyendo flujos provenientes de sensores y dispositivos conectados por medio de sockets TCP. También se pueden procesar datos almacenados en sistemas de archivos como HDFS o Amazon S3.

Spark Streaming puede procesar datos utilizando una variedad de algoritmos y funciones tales como Map, Reduce, Join y Window. Una vez procesados, los datos son enviados a archivos en file systems o para poblar dashboards en tiempo real. (44)

A grandes rasgos, lo que hace Spark Streaming es tomar un flujo de datos continuo y convertirlo en un flujo discreto —llamado DStream— formado por paquetes de datos. Internamente, lo que sucede es que Spark Streaming almacena y procesa estos datos como una secuencia de RDDs (Resilient Distributed Data). Así que cuando usamos Spark Streaming para alimentar un stream a Spark Core, éste último los analiza de forma normal, sin enterarse de que está procesando un flujo de datos, porque el trabajo de crear y coordinar los RDDs lo realiza Spark Streaming.

4.14.3. Funcionalidades DAG y RDD

Spark mantiene la escalabilidad lineal y la tolerancia a fallos de MapReduce, pero amplía sus bondades gracias a varias funcionalidades: DAG y RDD.

4.14.3.1. DAG (Directed Acyclic Graph)

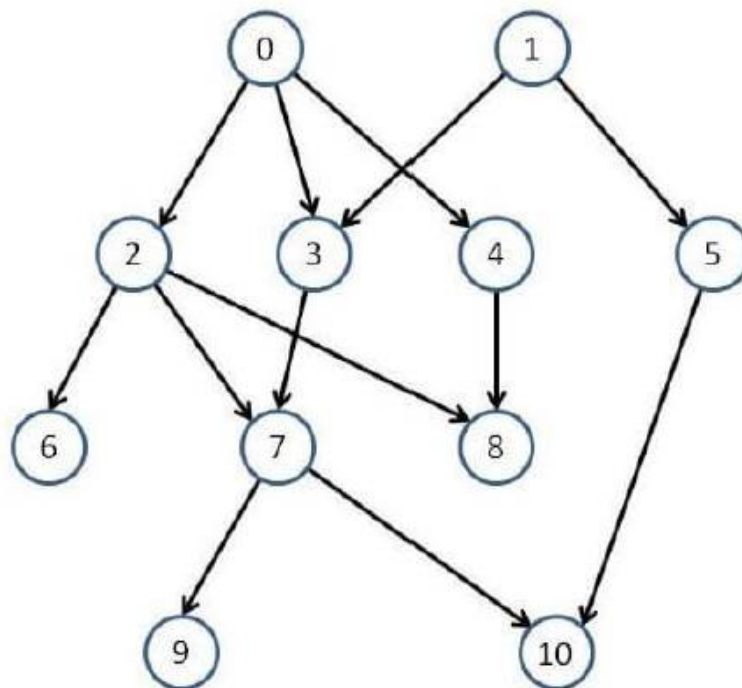


Figura 21. DAG (Grafo Acíclico Dirigido). Geekytheory.com

DAG (Grafo Acíclico Dirigido) es un grafo dirigido que no tiene ciclos, es decir, para cada nodo del grafo no hay un camino directo que comience y finalice en dicho nodo. Un vértice se conecta a otro, pero nunca a sí mismo.

Spark soporta el flujo de datos acíclico. Cada tarea de Spark crea un DAG de etapas de trabajo para que se ejecuten en un determinado clúster. En comparación con MapReduce, el cual crea un DAG con dos estados predefinidos (Map y Reduce), los grafos DAG creados por Spark pueden tener cualquier número de etapas. Spark con DAG es más rápido que MapReduce por el hecho de que no tiene que escribir en disco los resultados obtenidos en las etapas intermedias del grafo. MapReduce, sin embargo, debe escribir en disco los resultados entre las etapas Map y Reduce.

Gracias a una completa API, es posible programar complejos hilos de ejecución paralelos en unas pocas líneas de código. (45)

4.14.3.2. RDD (Resilient Distributed Dataset)

Apache Spark mejora con respecto a los demás sistemas en cuanto a la computación en memoria. RDD permite a los programadores realizar operaciones sobre grandes cantidades de datos en clúster de una manera rápida y tolerante a fallos. Surge debido a que las herramientas existentes tienen problemas que hacen que se manejen los datos ineficientemente a la hora de ejecutar algoritmos iterativos y procesos de minería de datos. En ambos casos, mantener los datos en memoria puede mejorar el rendimiento considerablemente.

Una vez que los datos han sido leídos como objetos RDD en Spark, pueden realizarse diversas operaciones mediante sus APIs. Los dos tipos de operaciones que se pueden realizar son:

Transformaciones: tras aplicar una transformación, obtenemos un nuevo y modificado RDD basado en el original.

Acciones: una acción consiste simplemente en aplicar una operación sobre un RDD y obtener un valor como resultado, que dependerá del tipo de operación.

Dado que las tareas de Spark pueden necesitar realizar diversas acciones o transformaciones sobre un conjunto de datos en particular, es altamente recomendable y beneficioso en cuanto a eficiencia el almacenar RDDs en memoria para un rápido acceso a los mismos. Mediante la función cache, se almacenan los datos en memoria para que no sea necesario acceder a ellos en disco.

El almacenamiento de los datos en memoria caché hace que los algoritmos de machine learning ejecutados que realizan varias iteraciones sobre el conjunto de datos de entrenamiento sea más eficiente. Además, se pueden almacenar versiones transformadas de dichos (45)

4.14.4. Spark 2.X

Uno de los grandes saltos que ha dado Spark en los últimos tiempos ha sido el paso de la versión 1.X a la 2.X de Spark.

Aunque en el anterior punto se han definido las funcionalidades DAG y RDD, hacían falta muy buenos conocimientos en programación para optimar el rendimiento a la hora de leer los datos.

En la nueva versión de Spark 2.X el mayor cambio que se produce es la unión del API Dataset y DataFrame en Scala y Java, añadiendo una nueva API para tipos de agregaciones en Datasets, siendo la primera versión de Spark centrada en ETL, y en versiones sucesivas sacaran más librerías y operadores para ETL. En Python y R al ser lenguajes no tipados, por defecto utilizan la API DataFrame. (46)

Hasta Spark 2.X, los RDD al ser datos sin formatos y no tener una estructura predefinida, las optimizaciones deban ser hechas por parte del programador, requiriendo más esfuerzo para mejorar el rendimiento en el procesamiento de los datos. Dataset es un tipo muy diferente a RDD ya que es una API que construye queries relacionales, y DataFrame es un tipo de dato basado en RDD, que traslada lenguaje SQL a expresiones de lenguaje de dominio específico optimizadas a bajo nivel para operaciones RDD. Ahora bien, con la unión de la API Dataset y DataFrame, ahora DataFrame comparte la base de código de Dataset, permitiendo que tenga las mismas optimizaciones básicas, optimizando la creación de código y conversiones transparentes al formato basado en columnas y una interfaz SQL (46) (47)

Spark 2.x se centrará en una combinación de Parquet y almacenamiento en cache para mejorar su rendimiento, mejorando el escaneo de Parquet a través de la vectorización (Parquet es un formato columnar soportado por muchos sistemas de procesamiento de datos), y en mejoras en el streaming de datos estructurados, en librerías como MLib, SparkR, SparkSQL entre otros cambios que pueden ser consultados en la documentación. (46)

4.14.5. Modelo de programación

Un programa típico se organiza de la siguiente manera:

1. A partir de una variable de entorno llamada context se crea un objeto RDD leyendo datos de fichero, bases de datos o cualquier otra fuente de información.
2. Una vez creado el RDD inicial se realizan transformaciones para crear más objetos RDD a partir del primero. Dichas transformaciones se expresan en términos de programación funcional y no eliminan el RDD original, sino que crean uno nuevo.
3. Tras realizar las acciones y transformaciones necesarias sobre los datos, los objetos RDD deben converger para crear el RDD final. Este RDD puede ser almacenado.

Un pequeño ejemplo de código en Python que cuenta el número de palabras que contiene un archivo sería el siguiente:

```
my_RDD = spark.textFile("hdfs://...")
words = my_RDD.flatMap(lambda line : line.split(" "))
.map(lambda word : (word, 1))
.reduceByKey(lambda a, b : a + b)
words.saveAsTextFile("hdfs://...")
```

Cuando el programa comienza su ejecución crea un grafo similar al de la figura siguiente en el que los nodos son objetos RDD y las uniones entre ellos son operaciones de transformación. El

grafo de la ejecución es un DAG y, cada grafo es una unidad atómica de ejecución. En la figura siguiente, las líneas rojas representan transformación y las verdes operación.

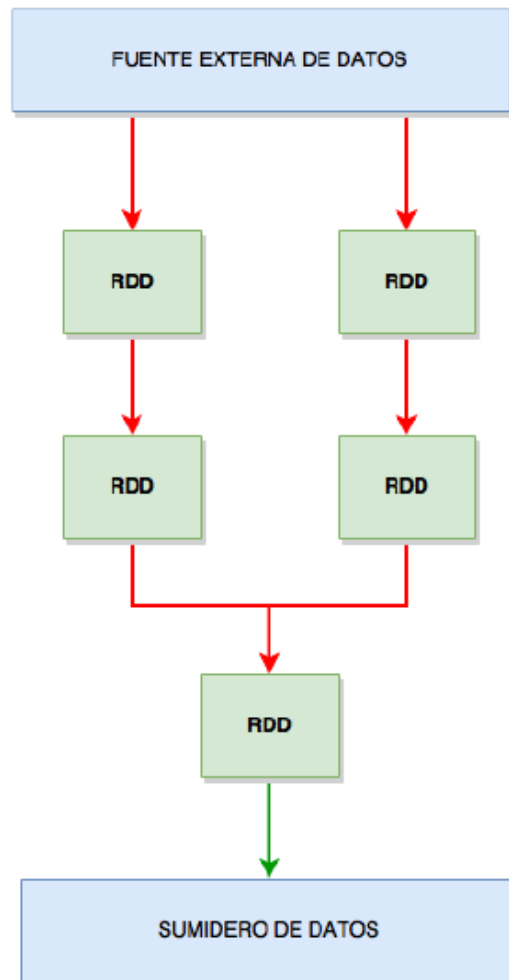


Figura 22. Transformaciones de un RDD. Geekytheory.com

4.14.6. Tipos de transformaciones

Es muy posible que los datos con los que se necesite tratar estén en diferentes objetos RDD, por lo que Spark define dos tipos de operaciones de transformación: narrow transformation y wide transformation.

Narrow transformation: se utiliza cuando los datos que se necesitan tratar están en la misma partición del RDD y no es necesario realizar una mezcla de dichos datos para obtenerlos todos. Algunos ejemplos son las funciones filter(), sample(), map() o flatMap().

Wide transformation: se utiliza cuando la lógica de la aplicación necesita datos que se encuentran en diferentes particiones de un RDD y es necesario mezclar dichas particiones para

agrupar los datos necesarios en un RDD determinado. Ejemplos de wide transformation son: `groupByKey()` o `reduceByKey()`. (45)

Una representación gráfica de ambos tipos de transformaciones es la que se puede apreciar en la figura siguiente:

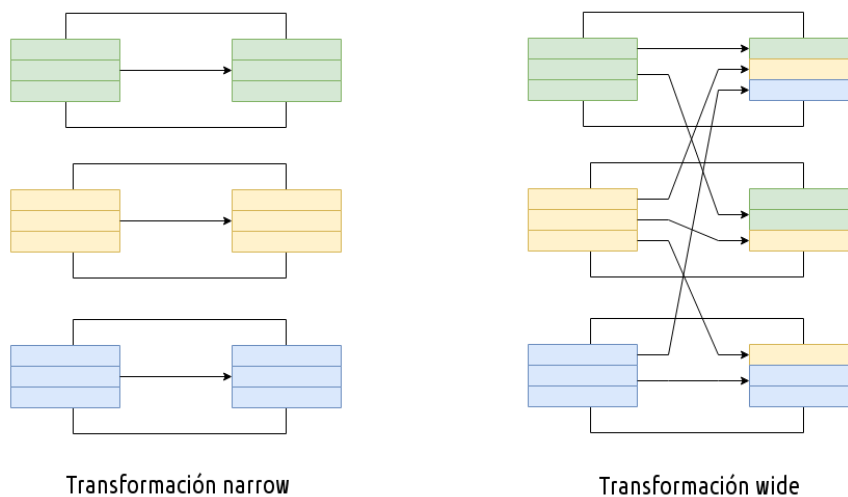


Figura 23. Tipos de transformaciones: Narrow y Wide. *Geekytheory.com*

En algunos casos es posible realizar un reordenamiento de datos para reducir la cantidad de datos que deben ser mezclados. A continuación, se muestra un ejemplo de un JOIN entre dos objetos RDD seguido de una operación de filtrado.

Por ejemplo, dados dos objetos RDD (RDD1 y RDD2), con variables 'a' y 'b', se va a realizar una operación de JOIN entre ambos conjuntos de datos para los casos en los que 'a' sea mayor que 5 y 'b' sea menor que 10:

```
SELECT a, b FROM RDD1 JOIN RDD2 WHERE a>5 AND b<10
```

Esta operación puede realizarse de dos maneras, tal y como se aprecia en la imagen de abajo. La primera opción consiste en una implementación muy simple en la que primero se realiza el JOIN entre los objetos RDD y luego se filtran los datos. Sin embargo, en la segunda opción, primero se realiza el filtrado por separado en ambos RDD y luego se hace el JOIN. (45)

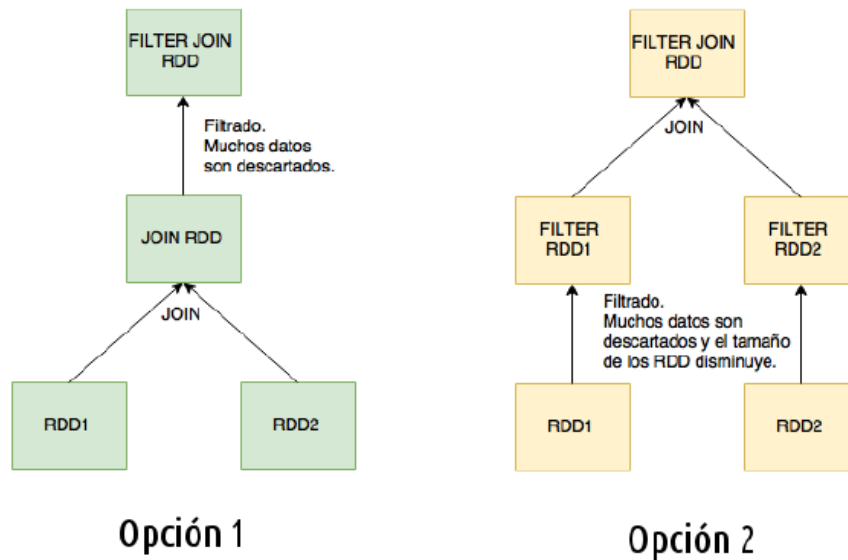


Figura 24. Reordenamiento de los datos de un RDD. *Geekytheory.com*

La segunda opción es más eficiente debido a que el filtrado y posterior unión de los datos se hace por separado. Podría decirse la mezcla o barajado de datos es la operación que más coste tiene, por lo que Apache Spark proporciona un mecanismo que genera un plan de ejecución a partir de un DAG que minimiza la cantidad de datos que son mezclados. El plan de ejecución es el siguiente:

1. Primero se analiza el DAG para determinar el orden de las transformaciones.
2. Con el fin de minimizar el mezclado de datos, primero se realizan las transformaciones narrow en cada RDD.
3. Finalmente se realiza la transformación wide a partir de los RDD sobre los que se han realizado las transformaciones narrow. (45)

4.14.7. Lenguaje de programación a utilizar en Spark

Este es otro punto importante y con gran controversia dentro del mundo de Spark, ya que Spark permite la programación de cuatro lenguajes sobre Spark, que son Java, Scala, Python(PySpark) y R, y dependiendo del lenguaje que utilicemos tendremos unas ventajas y/o inconvenientes a la hora de utilizar este framework.

Una de las cosas que he detectado en el tiempo que llevo estudiando estas tecnologías, es la alta volatilidad que hay en cuanto a que mañana aparece algo nuevo y lo que hoy era referente empieza a ser menos utilizado, lo cual requiere de una visión analítica sobre la información que existe y no volverse loco con las modas.

De hecho, remitiéndome a un artículo de BBVA sobre Spark de Souryna Luangsay(Ene,2015), matiza una anécdota que le ocurre a un instructor de Cloudera mientras enseñaba Spark a sus alumnos, en la que los propios alumnos descubrieron un bug en Spark, a lo que el instructor de

Cloudera abrió una incidencia en el mismo curso, algo que no le había pasado hasta entonces, matizando después que Spark era todavía un software joven y que no tenía la estabilidad que tiene Hadoop-MapReduce, de hecho en 2015 no tenían clientes que estuviesen utilizando esta tecnología en producción, aunque si estaban ya realizando pruebas sobre esta tecnología. (48)

Un ejemplo que pone es sobre la velocidad a la que está evolucionando, en un año paso de la versión 0.7 a la versión 1.0(2015), actualmente en 2017 estamos en la versión 2.2.1 lanzada el 1 de diciembre de este mismo año. También comenta que por una parte esto es bueno porque significa que hay muchas mejoras que se van incluyendo y que el proyecto está muy vivo, por otro lado, tantos cambios en las versiones denotan una cierta falta de madurez y de estabilidad, que a fecha actual va asentándose cada vez más, aunque no en lo que a lenguaje de programación a utilizar se refiere. (48)

Otro ejemplo de este continuo cambio, es la diversidad de soluciones SQL en Hadoop, al principio estuvo Hive, Cloudera saco Impala para aportar “tiempo real” en Hadoop, Hortonworks desarrolla la iniciativa Stinger para mejorar Hive y competir contra Impala, se desarrolla Shark encima de Spark para conseguir una especie de “Hive in memory”, luego Databricks decide dejar de soportar Shark y refactorizar todo en un nuevo proyecto que se conocerá como SparkSQL, apareciendo también una iniciativa de Hive encima de Spark. (48)

Toda esta amalgama de soluciones SQL sobre Hadoop no es de extrañar cuando hablamos de tecnologías tan novedosas, fomentando la competencia y por tanto el desarrollo, al menos al principio. A día de hoy esta tecnología está más consolidada, pero siguen existiendo cuestiones sin resolver y sigue evolucionando, por lo que es una razón de peso para no precipitarse y analizar con detenimiento el asunto, como es el asunto de lenguaje de programación a utilizar sobre Spark. (48)

Esta línea de investigación de los lenguajes de programación surge a partir de un artículo que encuentro también en BBVA, publicado en 2015.

En este artículo habla de la comunidad de Apache Spark haciendo referencia a que esta es cada vez más activa, en septiembre de 2013 había 113.000 líneas de código, un año después superan las 296.000 líneas y en septiembre de 2015 marcaban la cifra de 620.300 líneas de código. Actualmente hay 1.241.183 líneas de código. También hace otra referencia a como han ido evolucionando el crecimiento de esta comunidad en cuanto a contribuyentes se refiere, en junio de 2012 se dan de alta cuatro contribuyentes, en junio de 2015 ese número fue de 128, siendo a día de hoy 1592 desde marzo de 2010 que empezó. (49)

En primer lugar, hay que destacar que Spark está programado sobre Scala, por lo que este es su lenguaje nativo, a pesar de eso, podemos programar Java, R, PySpark o Scala, pregunta típica que podemos encontrar en internet sobre que lenguaje debo de aprender para programar sobre Spark.

Según un artículo de Quora.com, Java es eliminado de la elección por ser demasiado detallado, y no soporta Read-Evaluate-Print-Loop(REPL), es decir, un bucle de lectura, evaluación, impresión también conocidos como consola de lenguaje, necesita compilar antes de ejecutar. (50) (51)

Además, encontramos una buena comparación entre Scala y Python.

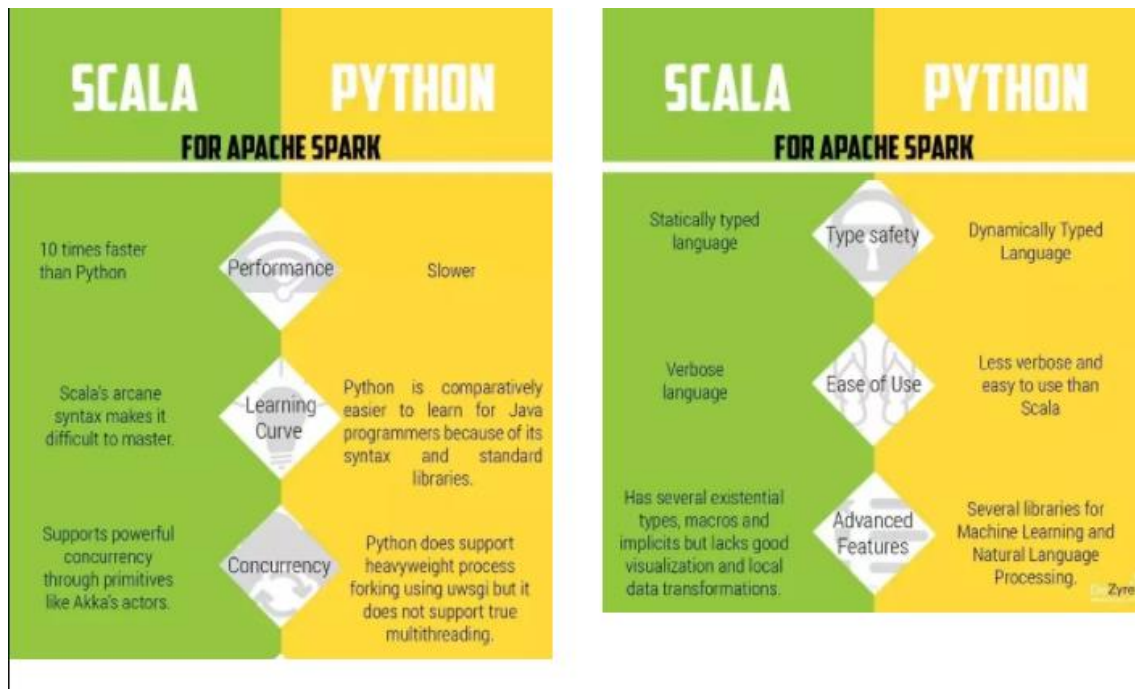


Figura 25. Scala vs Python en Spark. Dezyre.com

En cuanto a rendimiento, Scala es 10 veces más rápido, que el mismo código en Python para análisis de datos, principalmente esto se debe al tiempo consumido cuando Python es utilizado para llamar a librerías de Spark, si hay más procesamiento que código en Python este se vuelve más lento. (50) (51)

Si nos fijamos en la curva de aprendizaje de estos dos lenguajes, Python es más fácil de aprender por una sintaxis más simple y una gran cantidad de librerías estándar para su uso, que además reducen el tiempo de programación para un mismo código, frente a Scala que su sintaxis es un poco más compleja y no dispone de tantas librerías para su uso. (50) (51)

En términos de concurrencia y paralelismo, Scala soporta muy bien la concurrencia basada en primitivas, frente a Python que no soporta la concurrencia y el multihilo muy bien. (50) (51)

Además, Scala es un lenguaje tipado, frente a Python que es dinámicamente tipado (Python infiere el tipo de dato de manera dinámica).

Referente a la usabilidad, aunque tanto Scala como Python, son lenguajes igual de expresivos, Python utiliza un lenguaje ligeramente más amistoso que Scala y es menos expresivo también, lo que lo hace fácil de aprender para proyectos sobre Spark, aunque este factor es más subjetivo, y depende del presupuesto y del equipo que se disponga. (50) (51)

En cuanto a funcionalidades avanzadas, Scala tiene tipos existenciales, implícitos y macros, y Scala siempre será más poderosa en términos de framework, librerías, macros etc., mientras que Python es la elección natural para procesar lenguaje natural (Natural Processing Language) y machine learning por las herramientas de las que dispone. (50) (51)

En conclusión, dependiendo del problema concreto que tengamos y su finalidad, el equipo y el tiempo de desarrollo del que dispongamos, elegiremos un lenguaje u otro, pero general, Scala es la mejor elección para Stream, Streaming, Macros mientras que Python es la elección segura si nuestro objetivo es machine learning y procesar lenguaje natural.

De hecho, en la siguiente imagen, obtenida de un artículo de Javier García Paredes, Director de Estrategia y CEO del grupo Solutio, podemos ver la tendencia del lenguaje de programación utilizado para machine learning: (52)

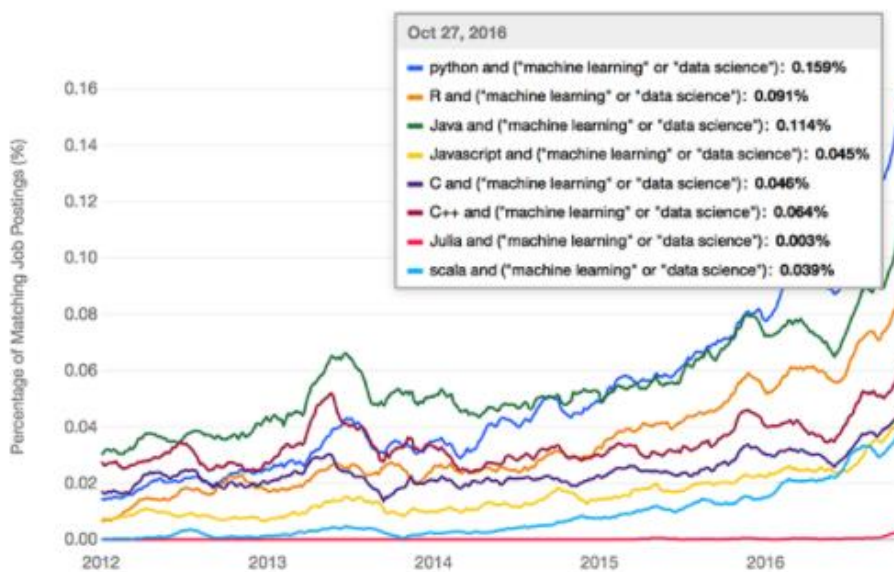


Figura 26. The most popular Language for machine learning and data science. IBM

Me parece interesante destacar algo que ocurre en España al margen de toda esta corriente, en la siguiente imagen vemos un estudio sacado de las ofertas publicadas en InfoJobs y LinkedIn por lenguaje para los términos Data Sciences, Data Scientist y machine learning.

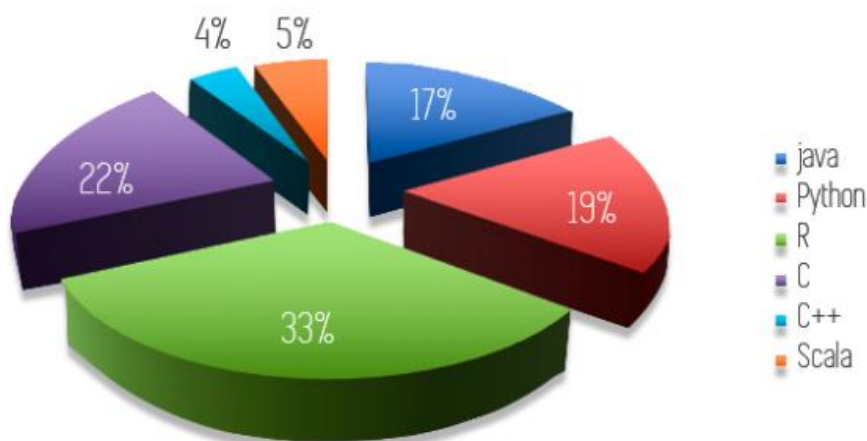


Figura 27. Grafica ofertas publicadas para el dominio DataScience, DataScientist, MachineLearning. Javier García Paredes

Donde el lenguaje más utilizado está siendo R seguido de C y Python, aunque no voy a entrar a comentar a que se puede deber esto, ya que forma parte de los objetivos de este TFG.

El ultimo lenguaje que podemos utilizar es R, que es utilizado principalmente en machine learning, y sería más propio comparar R con Python, aunque no voy a extenderme más sobre este tema, ya que Machine Learning queda fuera de los objetivos del TFG, pero si nuestro objetivo fuese machine learning, una comparación entre estos dos lenguajes sería fundamental para elegir el lenguaje adecuado para nuestra aplicación.

Cabe destacar que los lenguajes mencionados anteriormente, cada uno está trabajando para paliar sus debilidades, porque una cosa esta clara, Spark ha venido a quedarse.

4.15. Spark vs Hadoop

Una vez explicado más materia, podemos entrar a comparar Spark y Hadoop. Son dos diseños tecnológicos que, aunque parecen alternativos, en realidad fueron diseñados con la idea de operar en un mismo contexto. Aunque en sus inicios se apoyasen en MapReduce, a día de hoy, unidos, proporcionan la solución más potente para gran cantidad de procesamiento de datos en tiempo real. Una comparación entre Hadoop y Spark es algo difícil porque los dos realizan muchas cosas similares, aunque tienen algunas zonas en las que sus funcionalidades no se superponen. (26)

Para comenzar, Spark no tiene su propia administración de archivos y por eso debe apoyarse en el HDFS (Hadoop's Distributed File System) o alguna otra solución sustitutiva. Lo más indicado sería comparar a Hadoop Map Reduce con Spark, ya que son más comparables en su carácter de motores de procesamiento de datos. (26)

A medida que la ciencia de datos ha ido madurando en los últimos años, lo mismo ha ocurrido con la necesidad de contar con nuevas formas para el manejo de los datos en gran escala. Existen aplicaciones comerciales en las que Hadoop supera a la performance del recién llegado Spark, pero a su vez, Spark tiene su propio lugar en el espacio de big data debido a su velocidad y facilidad de uso. (26)

Ahora analizaremos los conjuntos de atributos que les son comunes a cada plataforma, incluyendo performance, tolerancia a fallos, costo, facilidad de uso, procesamiento de datos, compatibilidad y seguridad.

Lo más importante a tener en cuenta en relación a Hadoop y Spark es que su utilización no depende de un escenario de uno u otro, ya que no son excluyentes entre sí. Ninguno de los dos es un reemplazo total del otro. Los dos son compatibles entre sí y eso hace que, unidos, se esté frente a una solución extremadamente poderosa en una variedad de aplicaciones big data. (26)

En definitiva, Spark debería ser la elección por defecto para cualquier aplicación big data. Sin embargo, ese no es el caso. MapReduce ha logrado afianzarse en el mercado de big data para empresa que necesitan manejar grandes conjuntos de datos y tenerlos bajo control en sistemas estándar. La velocidad de Spark, su agilidad y relativa facilidad de uso son complementos excelentes para el bajo costo de operación de MapReduce. (26)

Hadoop ofrece elementos que Spark no tienen, tal como sistemas de archivos distribuidos y Spark ofrece procesamiento in-memory en tiempo real para aquellos conjuntos de datos que así lo requieran. El escenario perfecto de big data es el que sus diseñadores pensaron, el de Hadoop y Spark trabajando juntos en un mismo equipo.

4.16. Kylin

Apache Kylin es un motor de análisis distribuido de código abierto diseñado para proporcionar una interfaz SQL y análisis multidimensional(OLAP) sobre Hadoop que admite grandes conjuntos de datos, desarrollado originalmente por eBay Inc. Kylin permite consultar conjuntos de datos masivos a una latencia inferiores al segundo mediante tres pasos: Identificar un esquema estrella sobre Hadoop, la construcción de un cubo multidimensional de las tablas identificadas, y consulta ANSI-SQL a través de ODBC, JDBC o una api RESTful. (53)

En la era del Big Data, es necesario remodelar el tradicional almacenamiento de datos y procesamiento analítico. Muchos desafíos se plantean a las plataformas tradicionales debido a que los datos son cada vez mayores. Kylin es un motor OLAP que construye cubos de los datos presentes en un clúster Hadoop y almacena los cubos en HBase para un análisis posterior. Los cubos almacenados en HBase son analizado mediante consultas analíticas similares a SQL, permitiendo la generación de informes y paneles generados de los cubos proporcionando una información valiosa de los datos de una compañía. (11)

Kylin es un motor OLAP extremadamente rápido, diseñado para reducir la latencia de consultas sobre Hadoop de más de 10.000 millones de filas, ofreciendo respuestas rápidas a las consultas relacionadas con el negocio en comparación con otras opciones como Hive, los cubos se construyen utilizando Kylin que almacena los resultados precalculados para una respuesta de consulta más rápida (sacrificamos espacio frente a rendimiento). Se pueden definir cubos MOLAP, definiendo un modelo de datos y precompilarlo en Kylin, almacenados en HBase admite analítica en tiempo real, además de una escalabilidad horizontal y tolerancia a fallos, pudiéndose integrar con informes y visualización de datos de otras herramientas como Tableau, PowerBi, ClivView, Pentaho entre otros. (11) (53)

En la siguiente imagen podemos ver el propósito de la arquitectura del sistema.

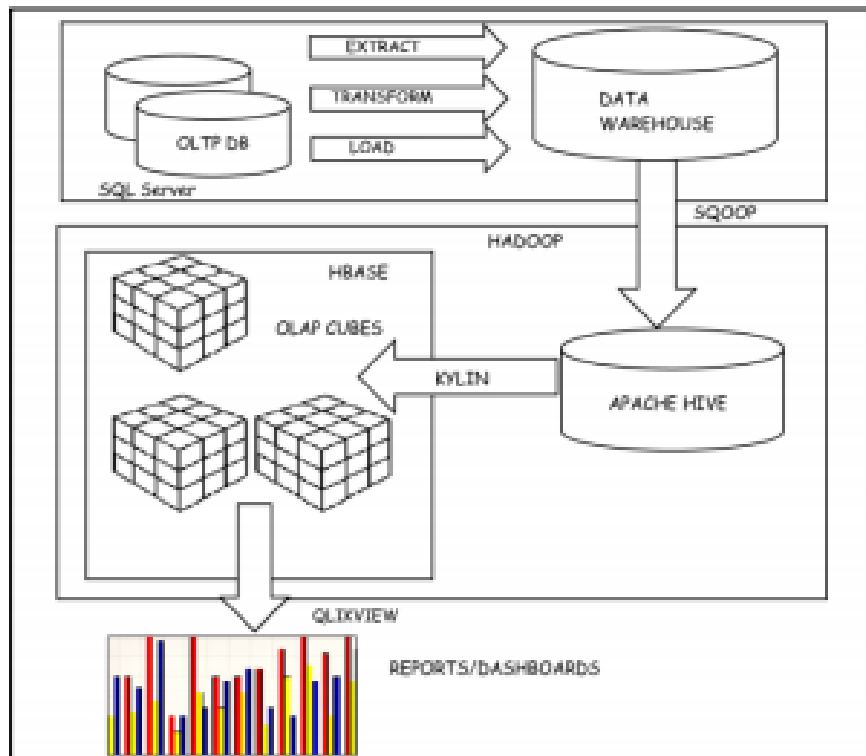


Figura 28. Arquitectura Kylin. Paper Online Analytical Processing on Hadoop using Apache Kylin

Donde en una primera fase, a través de los procesos ETL, alimentamos un Datawarehouse sobre Hive, Kylin consultara los datos almacenados en tablas Hive para generar sobre HBase los cubos OLAP precalculados, y esos cubos podrán ser consultados por herramientas de reporting y Dashboard. (11) (53)

4.17. Zeppelin

Apache Zeppelin es un bloc de notas(notebook) interactivo open source basado en web, que permite el análisis interactivo de datos, ofreciendo múltiples propósitos como ingestión de datos, exploración de datos, visualización, uso compartido y colaborativo para Hadoop, Spark y otras muchas tecnologías.

Los notebooks interactivos permiten a los ingenieros, analistas y científicos de datos ser más productivos desarrollando, organizando, ejecutando y compartiendo código de datos y visualizando los resultados sin la necesidad de utilizar línea de comandos ni necesitar detalles del clúster. Los notebooks permiten a los usuarios no solo la ejecución si no también trabajar de forma interactiva con grandes flujos de trabajo. Admite Python, pero también una creciente lista de lenguajes de programación como Scala, Hive, SparkSQL, Shell entre otros.

Los diversos lenguajes de programación o idiomas son compatibles a través de intérpretes en Zeppelin. (54) (55)

El descubrimiento de datos, exploración, informes y visualización son componentes clave del flujo de trabajo de las ciencias de datos. Zeppelin ofrece un “estudio de ciencia de datos moderno” soportando múltiples backends de lenguajes que tienen soporte en un ecosistema

creciente de fuentes de datos. Podemos ver una colección de los cuadernos o lenguajes que soporta Zeppelin en la siguiente imagen: (54)



Figura 29. Lenguaje de Backend. Zeppelin.apache.org

Apache Zeppelin proporciona varias API REST para la interacción y activación remota de funcionalidades de Zeppelin. Todas las API REST están disponibles a través de la siguiente url: [http://\[Zeppelin-server\]:\[Zeppelin-port\]/api](http://[Zeppelin-server]:[Zeppelin-port]/api). Hay que tener en cuenta que las API REST de Apache Zeppelin reciben o devuelven objetos JSON. Podemos obtener más información sobre roles y grupos en los interpretes o como definir nuestro propio interprete o conector en la siguiente referencia. (54)

Zeppelin tiene algunas visualizaciones básicas como charts entre otras. Las visualizaciones no están limitadas a queries en SparkSQL, cualquier salida de cualquier lenguaje de backend puede ser reconocido y visualizado.

Permite también la creación de algunos formularios de entrada dinámicos, además la URL del notebook puede ser compartida por muchos colaboradores, luego Apache transmite cualquier cambio en tiempo real, como la colaboración en Google docs. En adicción, Apache Zeppelin

proporciona una URL para mostrar solo el resultado de esa página, no incluye menús ni botones dentro del notebook, pudiendo ser insertado fácilmente como un iframe dentro de un sitio web. (56)

4.18. Docker

Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estándar llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluida bibliotecas, herramientas de sistema código y tiempo de ejecución, pudiendo ser ejecutados estos contenedores en la nube o en local. Con Docker se pueden implementar y ajustar la escalabilidad de las aplicaciones rápidamente en cualquier entorno con la certeza de saber que el código se ejecutara. (57)

Un contenedor no es más que una imagen en ejecución a la que le otorgamos recursos. Estas imágenes están compuestas por capas de solo lectura a las que se le añade una capa superior de escritura que perdura solo mientras el contenedor está en ejecución. En la construcción de la imagen se crea una capa por cada cambio. Nuestra imagen contiene solo lo necesario para ejecutar la aplicación que reside dentro de él. No necesitamos librerías, ejecutables, módulos o drivers que nuestra aplicación no utilice, es decir, garantizándonos que se ejecutara igual en cualquier host que tenga el motor de Docker, ya sea un portátil, una máquina virtual, la nube o un clúster. Esto se traduce en una reducción de costes de manera directa, ya que necesitamos menos recursos para ejecutar nuestros servicios, y aporta mucha agilidad en todo el ciclo de vida de la aplicación permitiendo desarrollar, probar y desplegar más rápido que con cualquier otra tecnología. (58)

4.18.1. Docker vs Máquinas virtuales

La comparación de Docker frente a máquinas virtuales nos va a permitir entender cómo funciona y las ventajas que aporta frente a otros sistemas.

Un contenedor Docker necesita muchos menos recursos, por ejemplo, no necesita un sistema operativo completo para funcionar, siendo más fáciles de implementar y más rápidos en ejecutarse, permitiendo ejecutar más servicios en la misma unidad de hardware y reduciendo los costos.

El objetivo principal de una imagen Docker es lo que hace el entorno, es decir, las dependencias. Una imagen es una manera de empaquetar una aplicación o servicio e implementarlo de manera confiable y reproducible, por lo que, en este sentido, Docker no es solo una tecnología sino también una filosofía y un proceso.

Si comparamos la arquitectura que tiene una máquina virtual frente a Docker, nos permitirá mejor entender cuáles son las mejoras que aporta frente a las máquinas virtuales, y cuáles son los beneficios de utilizar Docker, dichas arquitecturas las vemos en la siguiente imagen: (59) (60)

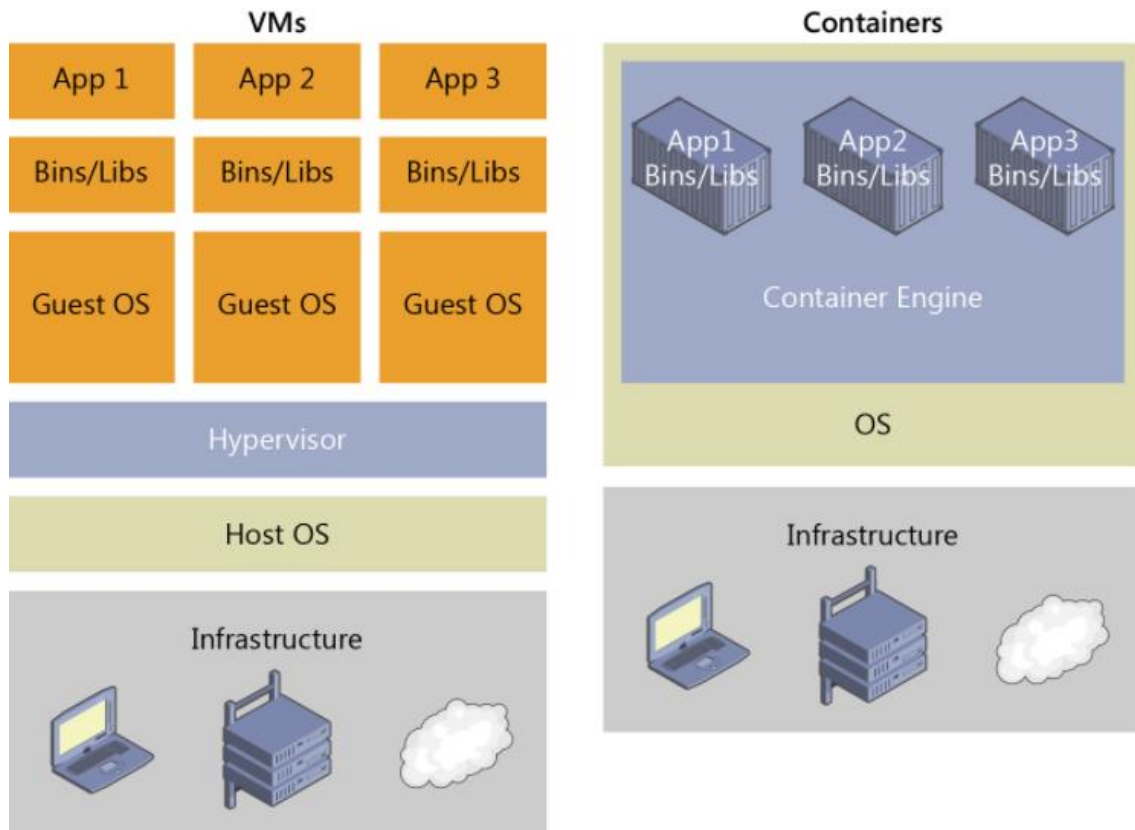


Figura 30. Arquitectura VMs vs Docker. Docs.microsoft.com

En la parte izquierda de la imagen vemos la arquitectura típica de las máquinas virtuales, donde sobre una infraestructura (servidor, nube o un ordenador personal), tenemos el sistema operativo sobre el cual se instala un host de máquinas virtuales como puede ser VirtualBox, cuando instalamos un host de máquinas virtuales el funcionamiento de estas es montar otro sistema sobre el sistema operativo base, ha sido típico durante la carrera que sobre un sistema operativo base Windows, montar una máquina virtual Linux para realizar prácticas de programación, asignándole recursos a este nuevo entorno, y siendo administrados los recursos por un hipervisor. Sobre este nuevo sistema operativo tendríamos las dependencias del sistema operativo, como librerías, archivos binarios, drivers etc., y finalmente montaríamos la aplicación. Por lo que, en definitiva, para ejecutar una aplicación estamos dividiendo los recursos de los que disponemos entre dos sistemas operativos. (59) (60)

Cuando entra en acción Docker, esto se agiliza, porque sobre una infraestructura (servidor, nube o ordenador personal) con su sistema operativo se instala el servidor Docker, y sobre este servidor se crean los contenedores que contienen todo lo necesario para correr nuestra aplicación. En ningún momento montamos un nuevo sistema operativo con el consumo de recursos que eso conlleva, si no que Docker utiliza los mismos recursos que el sistema operativo base, sin dividir los recursos que tengamos. (59) (60)

Podemos ver esto claramente en la siguiente imagen:

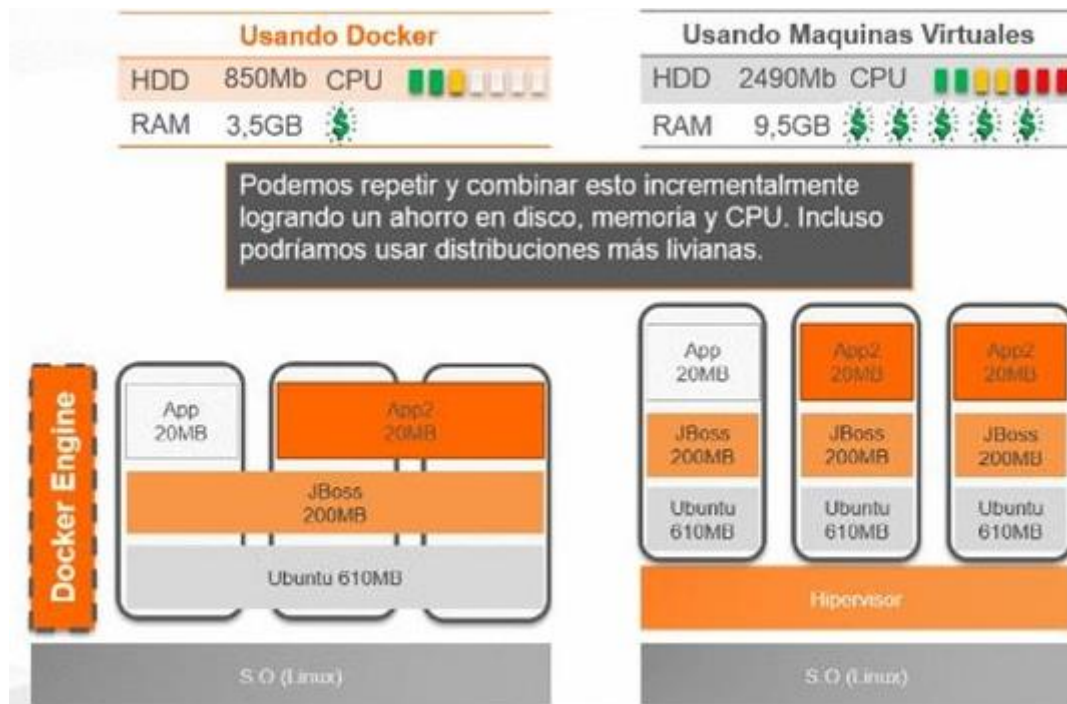


Figura 31. Arquitectura VMs vs Docker. Ángel Luis Mula

5. Diseño

En esta parte del proyecto, vamos a definir y explicar el diseño que se ha realizado para resolver la problemática y cumplir con los objetivos del proyecto.

Para ello, empezare definiendo el modelo de la arquitectura que vamos a diseñar de manera que de arriba abajo vayamos definiendo y diseñando dicha arquitectura que resuelve la problemática del BI tradicional.

5.1. Diseño de la arquitectura

En primer lugar, quiero volver a hacer referencia a la arquitectura BI tradicional, ya que, aunque hemos roto con esta arquitectura en cuestión de tecnologías, el marco teórico (el modelo multidimensional), definido por Kimball (6), seguirá siendo válido, y muchos de los conceptos teóricos que define volverán a aparecer en el sistema a diseñar.

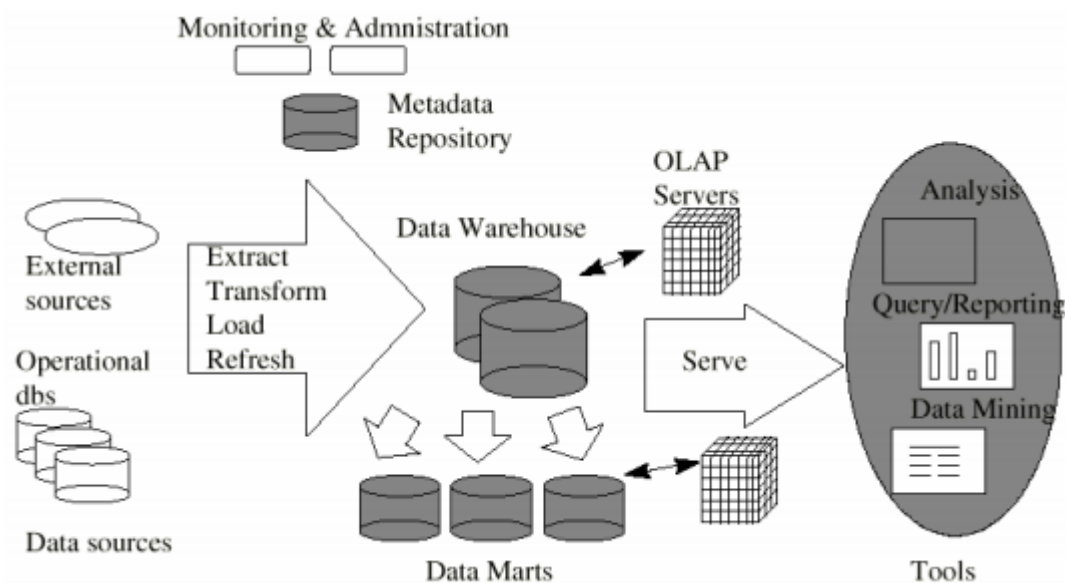


Figura 32. Arquitectura BI tradicional. Fernando Berzal.

Partiendo de la anterior imagen, arquitectura de un Sistema BI, donde a grandes rasgos, tenemos cinco partes (fuentes de datos, herramienta de procesamiento ETL, servidor datawarehouse, servidor OLAP, y por último, un servidor de reporting y análisis de la información), donde en el estado del arte ya mencioné las tecnologías tradicionales utilizadas en BI, modificare dicha arquitectura de manera que incluya las tecnologías Big Data a utilizar para diseñar dicha arquitectura con tecnología Big Data.

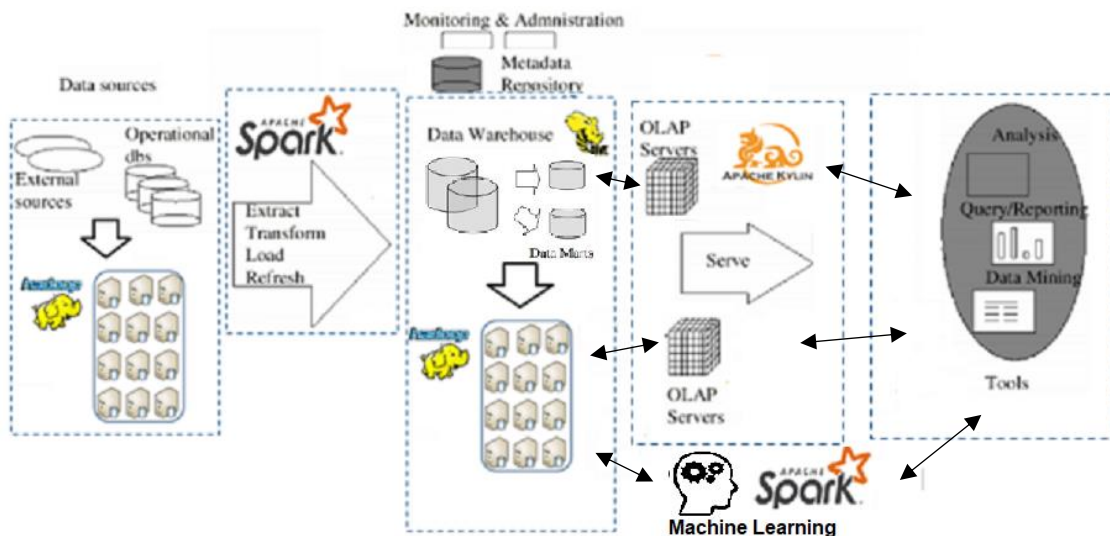


Figura 33. Arquitectura BI con tecnología BigData. Alejandro Reina

Al igual que en una arquitectura tradicional de BI, podríamos dividir la arquitectura en cinco partes (fuentes de datos, herramienta de procesamiento ETL, servidor datawarehouse, servidor OLAP, por último, un servidor de reporting y análisis de la información).

Cabe destacar, que, aunque en el diseño de la arquitectura aparece Hadoop como tecnología de comunicación de clúster, en este proyecto, me voy a abstraer del diseño de ese clúster por motivo de recursos, se diseñara un único nodo sobre una de las tecnologías que componen el ecosistema de Hadoop, que es **HBase**, sobre el cual se almacenaran los archivos de nuestras fuentes de datos, de la funcionalidad de HDFS no haremos uso en este proyecto, aunque si permitirá hacer uso cuando sea desplegado en un clúster o en la nube.

En la parte de procesamiento de datos(ETL), nos vamos a apoyar en el framework **Spark**, el cual conectara con HBase para obtener los datos en bruto, procesarlos en memoria y volcarlos en un datawarehouse.

En el apartado de datawarehouse, ocurre algo similar que con las fuentes de datos, **Hive**, permite volcar los datos del datawarehouse sobre un clúster Hadoop de manera que evitemos ese servidor central donde se consultan todos los datos, por motivos de recursos, ese clúster distribuido de nuestro datawarehouse no vamos a tenerlo, tendremos un único nodo donde volcamos los datos y luego serán analizados, aunque en un despliegue en un clúster o nube, si entraría en juego dicha distribución de la información.

A continuación, pasaríamos a la parte del servidor OLAP, en este caso será Kylin, el cual una vez diseñado el modelo multidimensional sobre un Datamart o datawarehouse en Hive, precalculara todas las combinaciones de los datos, y los volcará físicamente sobre HBase, para que ante cualquier pregunta analítica que hagamos, ya tenga el resultado precalculado. Kylin sacrifica espacio en beneficio de rendimiento, es decir, poder responder rápidamente a una determinada pregunta analítica en cuestión de sub segundos.

En la última parte de la arquitectura, podemos utilizar un gran abanico de herramientas como puede ser Pentaho, PowerBI, Tableau, entre muchas otras, facilitando la exploración y consulta de esos datos, pudiendo montar un sistema de reporting o dashboards.

Una vez tengamos desplegada la arquitectura, la encapsularemos dentro de un contenedor Docker para permitir su despliegue rápido en cualquier servidor.

5.2. Despliegue de la arquitectura

En primer lugar, destacar que esta arquitectura va a ser desplegada sobre el S.O. Linux 16.04 LTS, siendo la última versión estable y utilizada como SO base de muchos servidores, aunque no voy a entrar en detalles sobre la instalación del SO. Es importante también remarcar, que antes de utilizar una tecnología de despliegue en servidor como Docker, debemos de entender bien cómo funciona cada servicio y configurarlo correctamente. Podría hacerse directamente sobre Docker, que tiene alguna particularidad en la que entrare más adelante en esta memoria, pero a nivel personal, con el fin de simplificar toda la amalgama de tecnologías que entran en juego y las pruebas de integración, me ha resultado más fácil y cómodo instalar y configurar primero los servicios en una máquina virtual y una vez entendidos, utilizar Docker para generar dicho script de despliegue rápido que podremos exportar a cualquier servidor. Como ultima mención, se instalarán las últimas versiones estables sobre cada uno de los servicios mencionados.

También con el fin de realizar una instalación genérica que pueda correr sobre cualquier servidor Linux, las instrucciones y localización de los servicios será en el directorio `"/usr/local/directorioServicio/"` ya que esta ruta de directorios es común a cualquier instalación de Linux.

5.2.1. Java

En la fase inicial del despliegue de la arquitectura, y antes de empezar a instalar cualquier servicio, necesitaremos instalar Java 8, ya que muchas de las tecnologías lo requieren, como Scala, Spark o Hive.

Por defecto no tendremos Java instalado, aun así, podemos comprobarlo con el siguiente comando:

```
alex@alex-bi:~$ java -version
El programa «java» puede encontrarse en los siguientes paquetes:
* default-jre
* gcj-5-jre-headless
* openjdk-8-jre-headless
* gcj-4.8-jre-headless
* gcj-4.9-jre-headless
* openjdk-9-jre-headless
Intente: sudo apt install <paquete seleccionado>
```

Figura 34. Comprobar versión de Java (no instalado). Alejandro Reina

Como podemos ver en la imagen no está instalado, por lo que el siguiente paso será añadir los repositorios para poder realizar la instalación, como vemos en la siguiente imagen:

```
alex@alex-bi:~$ sudo apt-add-repository ppa:webupd8team/java
[sudo] password for alex:
Oracle Java (JDK) Installer (automatically downloads and installs Oracle JDK7 /
JDK8 / JDK9). There are no actual Java files in this PPA.

Important -> Why Oracle Java 7 And 6 Installers No Longer Work: http://www.webup
d8.org/2017/06/why-oracle-java-7-and-6-installers-no.html

Ubuntu 16.10 Yakkety Yak is no longer supported by Canonical (and thus, Launchpa
d and this PPA). The PPA supports Ubuntu 17.10, 17.04, 16.04, 14.04 and 12.04.

More info (and Ubuntu installation instructions):
- for Oracle Java 7: http://www.webupd8.org/2012/01/install-oracle-java-jdk-7-in
-ubuntu-via.html
- for Oracle Java 8: http://www.webupd8.org/2012/09/install-oracle-java-8-in-ubu
ntu-via-ppa.html
```

Figura 35. Añadir repositorios Java. Alejandro Reina

Una vez añadido el repositorio, actualizaremos el sistema y sus dependencias con el siguiente comando:

```
alex@alex-bi:~$ sudo apt-get update
Des:1 http://ppa.launchpad.net/webupd8team/java/ubuntu xenial InRelease [17,6 kB
]
Obj:2 http://es.archive.ubuntu.com/ubuntu xenial InRelease
Des:3 http://es.archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Des:4 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Des:5 http://es.archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Des:6 http://ppa.launchpad.net/webupd8team/java/ubuntu xenial/main amd64 Package
s [2.896 B]
Des:7 http://ppa.launchpad.net/webupd8team/java/ubuntu xenial/main i386 Packages
[2.896 B]
Des:8 http://ppa.launchpad.net/webupd8team/java/ubuntu xenial/main Translation-e
n [1.260 B]
Descargados 331 kB en 1s (181 kB/s)
Leyendo lista de paquetes... Hecho
```

Figura 36. Actualización del sistema y dependencias. Alejandro Reina

Tras la actualización el árbol de dependencias, podemos ejecutar el comando para la instalación de Java, en mi caso será Java8.

```
alex@alex-bi:~$ sudo apt-get install oracle-java8-installer
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
```

Figura 37. Instalar Java8. Alejandro Reina

Luego, añadiremos el PATH (variable de entorno) a nuestro fichero bashrc con el siguiente comando:

```
alex@alex-bi:~$ nano ~/.bashrc
```

Figura 38. Abrir bashrc. Alejandro Reina

Añadiendo la siguiente línea al final del fichero como vemos en la siguiente imagen, estableciendo la ruta donde se encuentra Java8:

```

root@alex-bi: /usr/local
GNU nano 2.5.3 Archivo: /root/.bashrc
# Set Java_Home
export JAVA_HOME=/usr/lib/jvm/java-8-oracle

```

Figura 39. Añadir Path de Java8. Alejandro Reina

Si todo ha ido bien, podremos volver a ejecutar el comando para comprobar la versión, si se ha instalado correctamente veremos algo como la siguiente imagen:

```

alex@alex-bi:~$ java -version
java version "1.8.0_144"
Java(TM) SE Runtime Environment (build 1.8.0_144-b01)
Java HotSpot(TM) 64-Bit Server VM (build 25.144-b01, mixed mode)

```

Figura 40. Comprobar versión de Java(Instalado). Alejandro Reina

Con esto concluiríamos la instalación y configuración de Java8 en nuestro sistema.

5.2.2. Scala

Antes de comenzar a instalar otros servicios como Spark, debemos de preparar las dependencias necesarias que requieren nuestros servicios, por ejemplo, el lenguaje de programación que ejecutaremos en Spark. Spark está construido en Scala 2.11 por defecto, por lo que será esta la versión de Scala que instalaremos.

En este apartado, vamos a ver como instalar y configurar correctamente Scala para que después podamos utilizar Scala sobre Spark, para ello, buscaremos en la página oficial de Scala la versión que queramos descargar, en mi caso instalare la versión 2.11.6 de Scala (61) como vemos en la siguiente imagen:

```

alex@alex-bi: /usr/local$ sudo wget http://www.scala-lang.org/files/archive/scala-2.11.6.tgz
--2017-08-22 14:51:59-- http://www.scala-lang.org/files/archive/scala-2.11.6.tgz
Resolviendo www.scala-lang.org (www.scala-lang.org)... 128.178.154.159
Conectando con www.scala-lang.org (www.scala-lang.org)[128.178.154.159]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 27130723 (26M) [application/x-gzip]
Grabando a: "scala-2.11.6.tgz"

scala-2.11.6.tgz 100%[=====] 25,87M 8,00MB/s in 3,4s
2017-08-22 14:52:03 (7,67 MB/s) - "scala-2.11.6.tgz" guardado [27130723/27130723]

```

Figura 41. Descarga de Scala. Alejandro Reina

Una vez descargado, lo descomprimiremos y moveremos al directorio anteriormente mencionado `/usr/local/`, aunque se podría usar otro, además lo renombraremos para mayor comodidad en la navegación de ficheros a través de la línea de comandos, esto se hará para todos los servicios, como vemos en las siguientes imágenes:

```

alex@alex-bi: /usr/local$ sudo tar xvf scala-2.11.6.tgz
scala-2.11.6/
scala-2.11.6/man/
scala-2.11.6/man/man1/
scala-2.11.6/man/man1/scala.1

```

Figura 42. Descomprimir Scala. Alejandro Reina

```
alex@alex-bi:/usr/local$ ls
bin  etc  games  include  lib  man  sbin  scala-2.11.6  scala-2.11.6.tgz  share  src
alex@alex-bi:/usr/local$ sudo mv scala-2.11.6 scala
alex@alex-bi:/usr/local$ ls
bin  etc  games  include  lib  man  sbin  scala  scala-2.11.6.tgz  share  src
```

Figura 43. Renombrar y mover Scala. Alejandro Reina

A continuación, utilizaremos el comando “`rm -R scala-2.11.6.tgz`” para liberar espacio, ya que el archivo comprimido ya no es de utilizad.

El último paso, es añadir las siguientes líneas para establecer el PATH de Scala en nuestro fichero `bashrc`.

```
# Set SCALA_HOME
export SCALA_HOME=/usr/local/scala
export PATH=$SCALA_HOME/bin:$PATH
```

Figura 44. Path Scala. Alejandro Reina

Si todo ha ido correctamente, cuando ejecutemos el comando “`scala -version`” veremos algo como en la siguiente imagen:

```
alex@alex-bi:/usr/local$ scala -version
Scala code runner version 2.11.6 -- Copyright 2002-2013, LAMP/EPFL
```

Figura 45. Scala version. Alejandro Reina

5.2.3. Spark

Una vez tenemos las dependencias de Spark preparadas, lo descargaremos de la web oficial de Spark (62), en mi caso instalare la versión de Spark 2.2.0 que viene con la versión de Hadoop 2.7. Spark viene con una versión de Hadoop preconstruida, ya que Spark utiliza las librerías de cliente de Hadoop para HDFS y YARN. Se podría utilizar una propia distribución de Hadoop añadiendo el CLASSPATH de Spark. (63)

Una vez descargado, descomprimido, movido al directorio `/usr/local/` y eliminado el fichero comprimido, estableceremos el PATH de Spark.

```
# Set SPARK_HOME
export SPARK_HOME=/usr/local/spark
export PATH=$SPARK_HOME/bin:$PATH
```

Figura 46. Path Spark. Alejandro Reina

Con el fin de evitar problemas, en el fichero “`/conf/Spark-env.sh`” añadiremos al final el path de Java como vemos en la siguiente imagen:

```
GNU nano 2.5.3 Archivo: spark-env.sh
# - SPARK_HISTORY_OPTS, to set config properties only for the history server (e$
# - SPARK_SHUFFLE_OPTS, to set config properties only for the external shuffle $
# - SPARK_DAEMON_JAVA_OPTS, to set config properties for all daemons (e.g. "-Dx$
# - SPARK_PUBLIC_DNS, to set the public dns name of the master or workers

# Generic options for the daemons used in the standalone deploy mode
# - SPARK_CONF_DIR      Alternate conf dir. (Default: ${SPARK_HOME}/conf)
# - SPARK_LOG_DIR       Where log files are stored. (Default: ${SPARK_HOME}/lo$
# - SPARK_PID_DIR       Where the pid file is stored. (Default: /tmp)
# - SPARK_IDENT_STRING  A string representing this instance of spark. (Default:$
# - SPARK_NICENESS       The scheduling priority for daemons. (Default: 0)
# - SPARK_NO_DAEMONIZE  Run the proposed command in the foreground. It will not$

JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

Figura 47. Path Java en Spark-env.sh. Alejandro Reina

Si todo ha ido correctamente ejecutaremos el comando Spark-shell y veremos algo como en la siguiente imagen:

```
alex@alex-bi:~$ spark-shell
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/08/23 12:26:22 WARN NativeCodeLoader: Unable to load native-hadoop library for your platfo
pplicable
17/08/23 12:26:23 WARN Utils: Your hostname, alex-bi resolves to a loopback address: 127.0.1.
  enp0s3)
17/08/23 12:26:23 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
17/08/23 12:26:36 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjec
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-1503483985011).
Spark session available as 'spark'.
Welcome to

       _ _ _ _ _ _ _ _ _ _
      / /   / /   / /   / /
     /_/   /_/   /_/   /_/
    _/____/_/____/_/____/_/____
   /_____/_____/_____/_____/
  /_____/_____/_____/_____/

version 2.2.0

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_144)
Type in expressions to have them evaluated.
Type :help for more information.

scala> █
```

Figura 48. Spark Shell. Alejandro Reina

A pesar de que en la imagen vemos warning, en este caso no es por ningún problema, son mensajes de información, por ejemplo, el primer warning “NativeCodeLoader” es porque la librería de Hadoop nativa esta compilada en 32 bit, y yo estoy utilizando Hadoop sobre Linux 64bit. Este warning se puede eliminar bajando el código fuente de Hadoop y recompilando “libhadoop.so.1.0.0” sobre un sistema de 64 bit, y reemplazar dicha librería de Spark por la nueva, aunque no tiene ningún impacto en nuestro sistema.

Además, Spark dispone de una interfaz webUI, a la que accederemos a la dirección `localhost:4040`, donde podremos consultar de los Jobs ejecutados, tiempos y otro tipo de información, como vemos en la siguiente imagen:

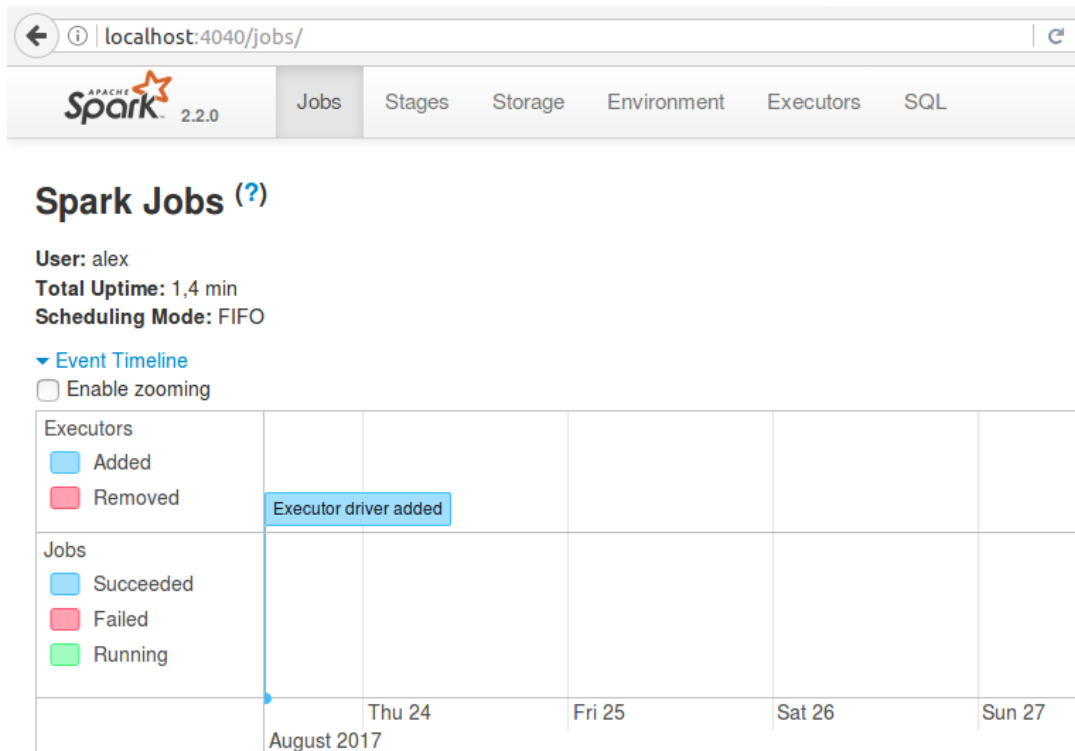


Figura 49. Spark webUI. Alejandro Reina

5.2.4. HBase

En el caso concreto de este TFG, donde no vamos a instalar y configurar, un clúster, y con el fin de optimizar al máximo el servidor, no instalaremos Hadoop completo, descargaremos uno de los módulos que componen Hadoop, que es HBase.

HBase tiene tres modos de instalación:

- Standalone: En este modo, HBase no utiliza HDFS, utiliza el sistema de archivos locales, y un Zookeeper que se conecta a un puerto para que los clientes puedan hablar con HBase.
- Pseudo_distributed: Este modo es como Fully-distributed, pero corriendo sobre un solo host.
- Fully-distributed: Este es el modo normal en entornos de producción, donde múltiples instancias de HBase corre sobre múltiples nodos de un clúster.

En nuestro caso, vamos a instalarlo en modo Standalone, para ello descargaremos la última versión estable de HBase, la versión 1.2.6 que podemos descargar de la página oficial de Apache HBase (64), una vez descargado haremos la misma operación que para los otros servicios, descomprimiremos, moveremos al directorio `"/usr/local/"`, y liberaremos espacio.

Una vez hecho esto, debemos configurar algunas cosas, en primer lugar, abriremos el fichero `"hbase/conf/hbase-env.sh"`, y le incluiremos el PATH de java como vemos en la siguiente imagen:

```
GNU nano 2.5.3 Archivo: /usr/local/hbase/conf/hbase-env.sh
# This script sets variables multiple times over the course of s
# so try to keep things idempotent unless you want to take an ev
# into the startup scripts (bin/hbase, etc.)
# The java implementation to use. Java 1.7+ required.
export JAVA_HOME=/usr/lib/jvm/java-8-oracle/
```

Figura 50. Hbase-env.sh path java. Alejandro Reina

Ahora abriremos el fichero “hbase/conf/hbase-site.xml”, y estableceremos la ruta donde HBase y el Zookeeper de HBase almacenaran sus archivos, como vemos en la siguiente imagen:

```
<configuration>
  //Establecemos el path donde hbase almacenara los archivos
  <property>
    <name>hbase.rootdir</name>
    <value>file:/usr/local/hfiles</value>
  </property>

  //Establecemos el path donde hbase almacenara sus archivos de zookeeper
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/usr/local/zookeeper</value>
  </property>
</configuration>
```

Figura 51. Path HBase y Zookeeper. Alejandro Reina

Por último, estableceremos el Path en el fichero bashrc como vemos en la siguiente imagen:

```
# Set HBASE_HOME
export HBASE_HOME=/usr/local/hbase
export PATH=$HBASE_HOME/bin:$PATH
```

Figura 52. Path HBase en bashrc. Alejandro Reina

Una vez configurado, podemos iniciaremos el servicio, con el siguiente comando:

```
root@alex-bi:/usr/local/hbase/bin# start-hbase.sh
starting master, logging to /usr/local/hbase/logs/hbase-root-master-alex-bi.out
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support
was removed in 8.0
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; sup
port was removed in 8.0
root@alex-bi:/usr/local/hbase/bin#
```

Figura 53. Iniciar HBase. Alejandro Reina

Si todo ha ido correctamente, cuando ejecutemos el comando “jps” (lista los procesos java que están corriendo en ese momento) veremos algo como la siguiente imagen:


```
alex@alex-bi:/$ sudo jps
3257 HMaster
3757 Jps
alex@alex-bi:/$
```

Figura 54. Proceso HMaster. Alejandro Reina

HBase también dispone de una interfaz webUI como Spark, para acceder utilizaremos la siguiente dirección “localhost:16010”, como vemos en la siguiente imagen:

ServerName	Start time	Version	Requests Per Second
alex-bi_38089,1511610941242	Sat Nov 25 12:55:41 CET 2017	1.2.6	0
Total:1			0

Figura 55. HBase webUI. Alejandro Reina

5.2.5. Hive

Ahora al igual que con los otros servicios, descargaremos la última versión estable de Hive, en mi caso será la versión de Hive 2.1.1, una vez descargado, descomprimido, puesto en el directorio correspondiente, estableceremos el PATH de Hive como en la siguiente imagen:

```
# Set HIVE_HOME
export HIVE_HOME=/usr/local/hive
export PATH=$HIVE_HOME/bin:$PATH
```

Figura 56. Path Hive en bashrc. Alejandro Reina

Hive tiene dependencias con algunas librerías de Hadoop para su funcionamiento, por lo que descargaremos Hadoop, descomprimiremos y moveremos a nuestro directorio “/usr/local/”, una vez hecho esto, abriremos el fichero “hadoop/etc/hadoop/hadoop-env.sh” y añadiremos el path de Java como en la siguiente imagen:

```
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-8-oracle/
```

Figura 57. Path Java en Hadoop-env.sh. Alejandro Reina

Una vez hecho esto, estableceremos en nuestro fichero bashrc los PATH de Hadoop necesarios para que Hive encuentre las librerías que necesita para su funcionamiento como vemos en la siguiente imagen:

```
# Set HADOOP variables
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

Figura 58. Path de Hadoop en bashrc. Alejandro Reina

Ahora añadiremos el Path de Hadoop en el fichero de configuración “hive/bin/hive-config.sh”, como vemos en la siguiente imagen:

```
#Hadoop home variable
export HADOOP_HOME=/usr/local/hadoop
```

Figura 59. Path de Hadoop en Hive-conf.sh. Alejandro Reina

Hive por defecto utiliza Derby (una base de datos relacional open source sobre java de Apache), con propósitos de testing, pero esta no es recomendable para entornos de producción ya que no permite más de un usuario activo a la misma vez. Por ello instalaremos MySQL como motor relacional de Hive, con el siguiente comando:

```
alex@alex-bi:/usr/local/hive/bin$ sudo apt-get install mysql-server
```

Figura 60. Instalar MySQL. Alejandro Reina

Una vez instalado, necesitamos instalar también algunas librerías de Java para MySQL, para esto ejecutaremos el siguiente comando:

```
alex@alex-bi:/usr/local$ sudo apt-get install libmysql-java
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
```

Figura 61. Librerías Java para MySQL. Alejandro Reina

Con las librerías instaladas, ahora copiaremos el conector a las librerías de Hive con el siguiente comando:

```
alex@alex-bi:/usr/local$ sudo ln -s /usr/share/java/mysql-connector-java.jar $HIVE_HOME/lib/mysql-connector-java.jar
```

Figura 62. Conector de Java para Hive. Alejandro Reina

Y arrancaremos MySQL con el siguiente comando como vemos en la imagen:

```
root@alex-bi:/usr/local# service mysql start
root@alex-bi:/usr/local# service mysql status
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since mar 2018-01-09 12:10:13 CET; 7s ago
     Process: 11522 ExecStartPost=/usr/share/mysql/mysql-systemd-start post (code=exited, status=0/SUCCESS)
     Process: 11515 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
    Main PID: 11521 (mysqld)
      CGroup: /system.slice/mysql.service
             └─11521 /usr/sbin/mysqld

ene 09 12:10:10 alex-bi systemd[1]: Starting MySQL Community Server...
ene 09 12:10:13 alex-bi systemd[1]: Started MySQL Community Server.
```

Figura 63. Ejecutar MySQL. Alejandro Reina

Una vez ya tenemos copiado el conector de Java en Hive, arrancaremos MySQL con el siguiente comando:

```
root@alex-bi:/usr/local# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 148
Server version: 5.7.20-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Figura 64. MySQL service start. Alejandro Reina

Una vez hecho esto, crearemos en MySQL el usuario para Hive, en mi caso el usuario será “hive” y contraseña “bigdata” como vemos a continuación:

```
mysql> CREATE USER 'hive'@'%' IDENTIFIED BY 'bigdata';
Query OK, 0 rows affected (0,00 sec)

mysql> GRANT all on *.* to 'hive'@localhost identified by 'bigdata'
-> ;
Query OK, 0 rows affected, 1 warning (0,02 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0,01 sec)

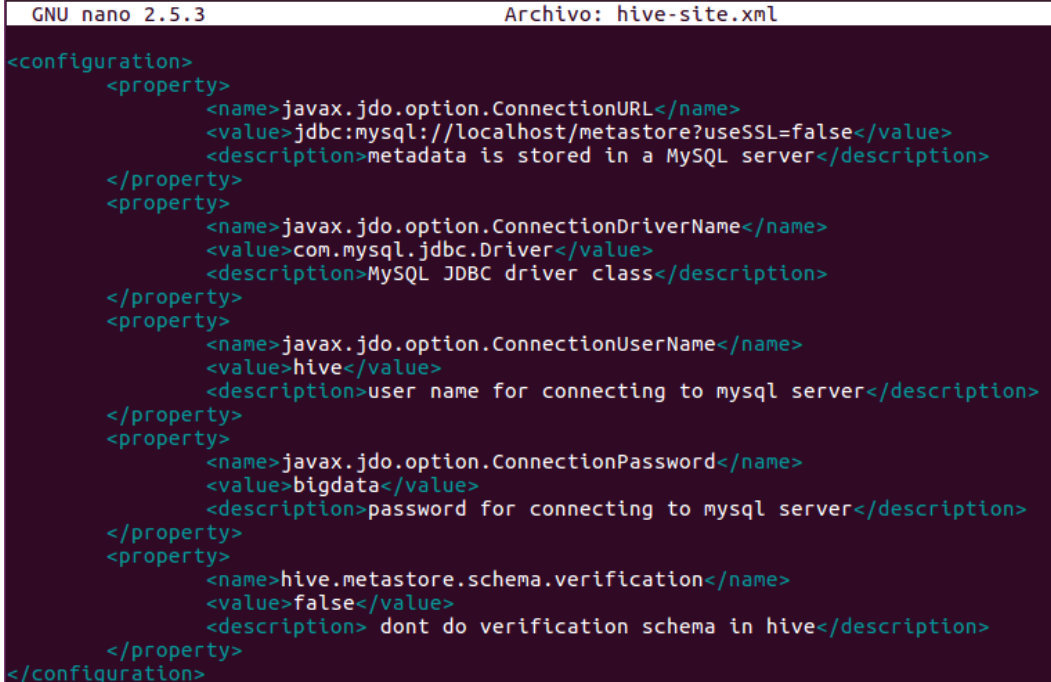
mysql>
```

Figura 65. Usuario para Hive en MySQL. Alejandro Reina

Este punto fue un poco confuso para mí, ya que me costó entender porque Hive necesitaba MySQL, ya que esa BD relacional como servidor central era uno de los problemas del BI tradicional, y no conseguía entender la diferencia entre Hive y MySQL, la diferencia principal está en que Hive es una especialización construida para propósitos analíticos(OLAP), frente a

MySQL que tiene un propósito general para procesos transaccionales(OLTP) y para propósitos analíticos(OLAP). Técnicamente la principal diferencia es la ausencia de la funcionalidad de actualización y borrados. Hive ofrece un lenguaje de script como MySQL sobre MapReduce, y la capacidad de escalar linealmente. (65).

Una vez creado el usuario para hive, creamos el fichero `"/usr/local/hive/conf/hive-site.xml"`, con la siguiente información:



```

GNU nano 2.5.3 Archivo: hive-site.xml
<configuration>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://localhost/metastore?useSSL=false</value>
    <description>metadata is stored in a MySQL server</description>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
    <description>MySQL JDBC driver class</description>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>hive</value>
    <description>user name for connecting to mysql server</description>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>bigdata</value>
    <description>password for connecting to mysql server</description>
  </property>
  <property>
    <name>hive.metastore.schema.verification</name>
    <value>false</value>
    <description>dont do verification schema in hive</description>
  </property>
</configuration>

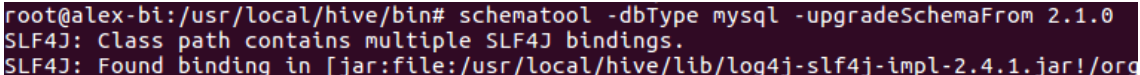
```

Figura 66. Crear fichero hive-site.xml. Alejandro Reina

La primera propiedad hace referencia a la URL para conectar a MySQL server, en la segunda, establecemos el driver que va a utilizar Hive para conectar a MySQL, en la tercera y cuarta propiedad estableceremos el usuario y la contraseña con la que Hive se conectara a MySQL.

La última propiedad es utilizada para que cuando arranquemos Hive no verifique que todas las tablas dentro de Hive tienen el mismo esquema de metadatos. Esta propiedad fue necesaria incluir, ya que en una primera instalación de Hive, cree una tabla desde MySQL versión de metadatos 2.1.0 y más adelante en el desarrollo del proyecto, al volcar las tablas de Spark a Hive, estas se guardaban en una versión 1.2.0, aunque esto no tiene impacto sobre la funcionalidad, esto hacía que Hive no pudiese arrancar al detectar distintas versiones, con esta opción indicamos a Hive que no compruebe esto.

De manera adicional, Hive tiene una orden para actualizar la versión de los metadatos a una versión que nosotros le indiquemos por ejemplo con el siguiente comando:



```

root@alex-bi:/usr/local/hive/bin# schematool -dbType mysql -upgradeSchemaFrom 2.1.0
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org

```

Figura 67. Hive upgradeSchema. Alejandro Reina

Pero aquí encontré un “bug”, porque al intentar actualizar, lo que hace es crear nuevas tablas, algunas que al tener ya una tabla con versión 2.1.0, ya estaban creadas, fallando el script de actualización, y no pudiendo unificar el esquema de los metadatos, la solución está en modificar los script que encontramos en el directorio “/hive/scripts/metastore/upgrade/mysql/”, donde están todos los script que utiliza Hive internamente para inicializar un schema(Hive-schema-2.1.0.mysql.sql) como los script para actualizar el esquema, como por ejemplo “upgrade-1.2.0-to-1.3.0.mysql.sql”. Estos scripts a su vez llaman a otros scripts como por ejemplo “024-HIVE-12814.mysql.sql”.

```

root@alex-bi:~/usr/local/hive/scripts/metastore/upgrade/mysql# ls
001-HIVE-972.mysql.sql      027-HIVE-12819.mysql.sql    hive-schema-2.0.0.mysql.sql
002-HIVE-1068.mysql.sql    028-HIVE-12821.mysql.sql    hive-schema-2.1.0.mysql.sql
003-HIVE-675.mysql.sql     029-HIVE-12822.mysql.sql    hive-txn-schema-0.13.0.mysql.sql
004-HIVE-1364.mysql.sql    030-HIVE-12823.mysql.sql    hive-txn-schema-0.14.0.mysql.sql
005-HIVE-417.mysql.sql     031-HIVE-12831.mysql.sql    hive-txn-schema-1.3.0.mysql.sql
006-HIVE-1823.mysql.sql    032-HIVE-12832.mysql.sql    hive-txn-schema-2.0.0.mysql.sql
007-HIVE-78.mysql.sql      034-HIVE-13076.mysql.sql    hive-txn-schema-2.1.0.mysql.sql
008-HIVE-2246.mysql.sql    035-HIVE-13395.mysql.sql    README
009-HIVE-2215.mysql.sql    036-HIVE-13354.mysql.sql    upgrade-0.10.0-to-0.11.0.mysql.sql
010-HIVE-3072.mysql.sql    hive-schema-0.10.0.mysql.sql upgrade-0.11.0-to-0.12.0.mysql.sql
011-HIVE-3649.mysql.sql    hive-schema-0.11.0.mysql.sql upgrade-0.12.0-to-0.13.0.mysql.sql
012-HIVE-1362.mysql.sql    hive-schema-0.12.0.mysql.sql upgrade-0.13.0-to-0.14.0.mysql.sql
013-HIVE-3255.mysql.sql    hive-schema-0.13.0.mysql.sql upgrade-0.14.0-to-1.1.0.mysql.sql
014-HIVE-3764.mysql.sql    hive-schema-0.14.0.mysql.sql upgrade-0.5.0-to-0.6.0.mysql.sql
016-HIVE-6386.mysql.sql    hive-schema-0.3.0.mysql.sql  upgrade-0.6.0-to-0.7.0.mysql.sql
017-HIVE-6458.mysql.sql    hive-schema-0.4.0.mysql.sql  upgrade-0.7.0-to-0.8.0.mysql.sql
018-HIVE-6757.mysql.sql    hive-schema-0.4.1.mysql.sql  upgrade-0.8.0-to-0.9.0.mysql.sql
019-HIVE-7784.mysql.sql    hive-schema-0.5.0.mysql.sql  upgrade-0.9.0-to-0.10.0.mysql.sql
020-HIVE-9296.mysql.sql    hive-schema-0.6.0.mysql.sql  upgrade-1.1.0-to-1.2.0.mysql.sql
021-HIVE-7018.mysql.sql    hive-schema-0.7.0.mysql.sql  upgrade-1.2.0-to-1.3.0.mysql.sql
022-HIVE-11970.mysql.sql   hive-schema-0.8.0.mysql.sql  upgrade-1.2.0-to-2.0.0.mysql.sql
023-HIVE-12807.mysql.sql   hive-schema-0.9.0.mysql.sql  upgrade-2.0.0-to-2.1.0.mysql.sql
024-HIVE-12814.mysql.sql   hive-schema-1.1.0.mysql.sql  upgrade.order.mysql
025-HIVE-12816.mysql.sql   hive-schema-1.2.0.mysql.sql
026-HIVE-12818.mysql.sql   hive-schema-1.3.0.mysql.sql
root@alex-bi:~/usr/local/hive/scripts/metastore/upgrade/mysql#

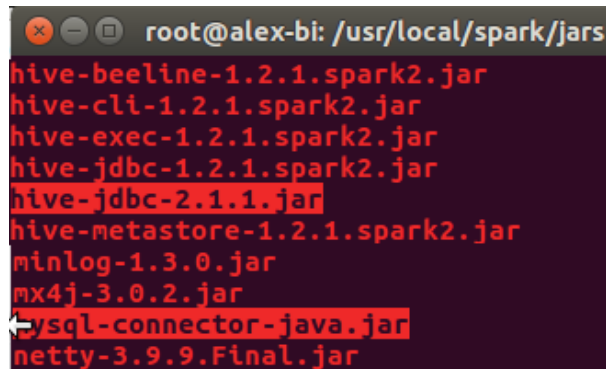
```

Figura 68. Script to UpgradeSchema. Alejandro Reina

Por lo que deberíamos ir inicializando los scripts de upgrade uno a uno y si alguno falla abrirlo y ver que script final está utilizando para ejecutar esa actualización, para modificar dicho script. Todos los errores que da son porque intenta crear una tabla que ya existe, esto podría solucionarse modificando “CREATE TABLE” por “CREATE TABLE IF NOT EXISTS”. Pero de momento Hive no ha solucionado dicho “bug”. (66)

Una vez aclarado este punto, vamos a copiar algunos conectores de Hive y MySQL a Spark para que este pueda conectar e integrarse con Hive, y guardar los datos procesados en el datawarehouse.

Para ello copiaremos los ficheros “mysql-conector-java.jar” y “hive-jdbc-2.1.1” al directorio de Spark “/spark/jars/” como vemos en la siguiente imagen:



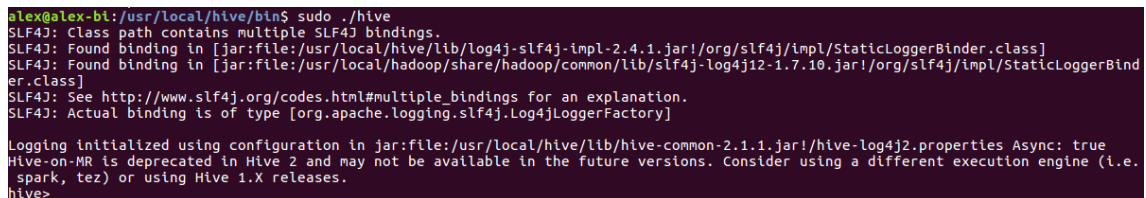
```

root@alex-bi: /usr/local/spark/jars
hive-beeline-1.2.1.spark2.jar
hive-cli-1.2.1.spark2.jar
hive-exec-1.2.1.spark2.jar
hive-jdbc-1.2.1.spark2.jar
hive-jdbc-2.1.1.jar
hive-metastore-1.2.1.spark2.jar
minlog-1.3.0.jar
mx4j-3.0.2.jar
mysql-connector-java.jar
netty-3.9.9.Final.jar

```

Figura 69. Jars de Hive y MySQL para Spark. Alejandro Reina

Una vez hecho esto, ya podemos ejecutar hive-shell, si hemos hecho los pasos anteriores correctamente veremos algo como lo siguiente:



```

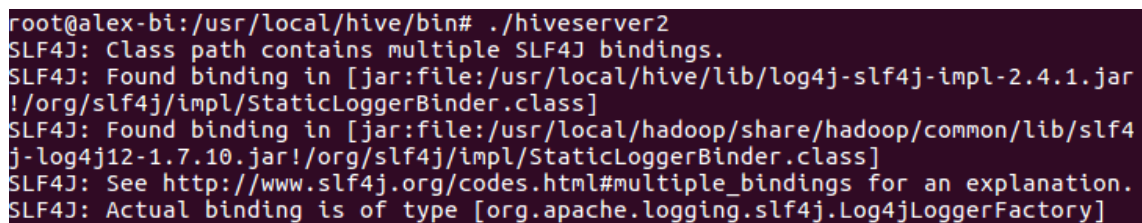
alex@alex-bi: /usr/local/hive/bin$ sudo ./hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-common-2.1.1.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive>

```

Figura 70. Hive Shell. Alejandro Reina

Hive al igual que HBase y Spark, dispone de una interfaz webUI donde tendremos información de la sesión activa, las últimas queries en Hive, log, métricas parámetros de configuración y trazas de todos los hilos activos, además de permitir a otras aplicaciones conectarse a Hive, para ello necesitamos arrancar hiveserver como haremos en el siguiente comando:



```

root@alex-bi: /usr/local/hive/bin# ./hiveserver2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

```

Figura 71. Ejecutar hiveserver. Alejandro Reina

Y ya podremos acceder a la interfaz web en la dirección “localhost:10002”

HiveServer2

Active Sessions

User Name	IP Address	Operation Count	Active Time (s)	Idle Time (s)
Total number of sessions: 0				

Open Queries

User Name	Query	Execution Engine	State	Opened Timestamp	Opened (s)	Latency (s)	Drilldown Link
Total number of queries: 0							

Figura 72. Hive webUI. Alejandro Reina

5.2.6. Zeppelin

Ahora vamos a descargar notebook Zeppelin, el cual nos provee de una interfaz webUI para programar integrando los distintos servicios previamente configurados. Para ello vamos a descargar la última versión estable de Zeppelin, en mi caso es la versión 0.7.3 (67).

Una vez descargado, descomprimido, movido y renombrado al directorio `"/usr/local/zeppelin/"`, vamos a realizar una serie de configuraciones.

En primer lugar, renombraremos el fichero `"/zeppelin/conf/zeppelin-env.sh-template"` a `"/zeppelin/conf/zeppelin-env.sh"`, en este fichero se pueden añadir opciones adicionales de configuración, por ejemplo, si desde Spark fuésemos a utilizar Python en vez de Scala, pero en nuestro entorno final solo lo copiaremos para que Zeppelin disponga de este fichero.

Otra modificación necesaria, es para poder acceder con un usuario y contraseña a Zeppelin, Zeppelin por defecto tiene habilitado el acceso anónimo, pero en ocasiones no funciona correctamente, por lo que bloquearemos el acceso anónimo para ello, modificaremos el fichero `"/zeppelin/conf/zeppelin-site.xml"`, estableciendo como valor false a los permisos de acceso anónimo como en la siguiente imagen:

```
GNU nano 2.5.3 Archivo: zeppelin-site.xml
-->
<property>
  <name>zeppelin.server.allowed.origins</name>
  <value>*</value>
  <description>Allowed sources for REST and WebSocket requests
</property>

<property>
  <name>zeppelin.anonymous.allowed</name>
  <value>>false</value>
  <description>Anonymous user allowed by default</description>
</property>
```

Figura 73. Anonymous allowed. Alejandro Reina

También renombraremos el fichero que tenemos en la ruta “/zeppelin/conf/shiro.ini.template” a “/zeppelin/conf/shiro.ini”, este fichero contiene la configuración de los usuarios que pueden acceder a Zeppelin, pudiendo añadir nuevos usuarios en este fichero, por defecto el administrador es de usuario “admin” y contraseña “password1”.

Una vez configurado, iniciaremos Zeppelin con el siguiente comando:

```
root@alex-bi: /usr/local/zeppelin/bin
Archivo Editar Ver Buscar Terminal Ayuda
root@alex-bi: /usr/local/zeppelin/bin# ./zeppelin-daemon.sh start
Zeppelin start [ OK ]
root@alex-bi: /usr/local/zeppelin/bin#
```

Figura 74. Zeppelin start. Alejandro Reina

Y una vez arrancado, ya podremos acceder con el usuario a su interfaz webUI a través de la url “localhost:8080” como vemos en la imagen:

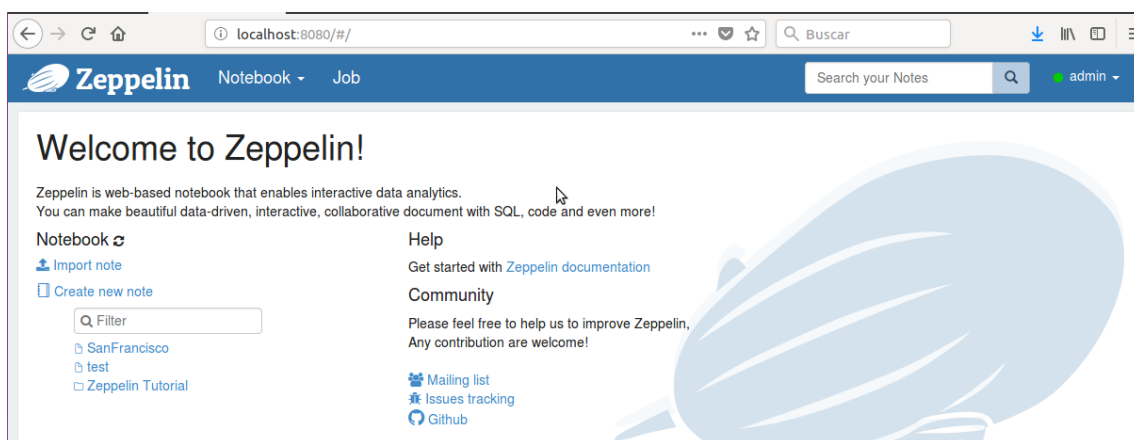


Figura 75. Zeppelin webUI. Alejandro Reina

Una vez aquí, debemos crear un nuevo interprete para Hive, para poder integrar Hive con Zeppelin y realizar consultas a Hive desde Zeppelin. Para ello iremos, a la esquina superior derecha haciendo clic en admin->interpreter como en la siguiente imagen:

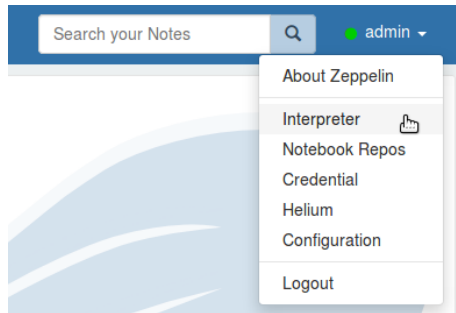


Figura 76. Intérpretes de Zeppelin. Alejandro Reina

Donde veremos todos los interpretes de que tiene Zeppelin por defecto. En nuestro caso, aunque Hive puede utilizar el conector JDBC que trae Zeppelin, con el fin de tener una mayor abstracción, crearemos nuestro propio interprete para Hive, para ello, una vez estemos en la interfaz de "Interpreter", haremos clic en la esquina superior derecha sobre el botón "create" y rellenaremos el nuevo interprete con los siguientes parámetros:

Zeppelin Notebook Job Search your Notes

Create new interpreter

Interpreter Name
Hive

Interpreter group
jdbc

Option
The interpreter will be instantiated Globally in shared process.
 Connect to existing process
 Set permission

Properties

name	value	description	action
common.max_count	1000	Max number of SQL result to display.	✕
default.driver	org.apache.hive.jdbc.HiveDriver	JDBC Driver Name	✕
default.password	bigdata	The JDBC user password	✕
default.url	jdbc:hive2://localhost:10000	The URL for JDBC.	✕
default.user	hive	The JDBC user name	✕
zeppelin.jdbc.auth.type		If auth type is needed, Example: KERBER OS	✕
zeppelin.jdbc.concurrent.max_connection	10	Number of concurrent execution	✕
zeppelin.jdbc.concurrent.use	true	Use parallel scheduler	✕
zeppelin.jdbc.keytab.location		Kerberos keytab location	✕
zeppelin.jdbc.principal		Kerberos principal	✕
			+

Dependencies

artifact	exclude	action
org.apache.hive:hive-jdbc2.1.1	(Optional) comma separated groupId:artifactId list	✕
org.apache.hadoop:hadoop-common2.7.0	(Optional) comma separated groupId:artifactId list	+

Figura 77. Parámetros para el intérprete de Hive. Alejandro Reina

Una vez hecho esto ya podremos ejecutar consultas Hive desde Zeppelin.

5.2.7. Kylin

Aquí está el servicio que con diferencia es el que más problemas me ha dado. El principal problema de Kylin es que es una tecnología muy nueva, con escasa documentación más que su página principal (68) y esta no está completamente actualizada a las últimas versiones, además de en algunos puntos como la instalación la información que nos proporcionan está incompleta.

En primer lugar, vamos a descargar la última versión disponible de Kylin, en mi caso sera la versión 2.2.0. con los binarios de HBase 1.X, si descargamos otra versión que no venga incluida con HBase, no funcionara en nuestro entorno, podemos ver el archivo descargado en la siguiente imagen:

2.2.0

- This is a major release after 2.1, with more than 70 bug fixes and enhancements. For the detail list please check release notes.
- [Release notes](#) and [upgrade guide](#)
- Source download: [apache-kylin-2.2.0-src.tar.gz](#) [asc] [md5]
- Binary download:
 - for HBase 1.x (includes HDP 2.3+, AWS EMR 5.0 - 5.7, Azure HDInsight 3.4 - 3.6) - [apache-kylin-2.2.0-bin-hbase1x.tar.gz](#) [asc] [md5]
 - for CDH 5.7-5.10 - [apache-kylin-2.2.0-bin-cdh57.tar.gz](#) [asc] [md5]

Figura 78. Kylin para HBase 1.X. Alejandro Reina

Una vez descargado, lo descomprimos, cabe destacar que, aunque la extensión del archivo es tar.gz no pude descomprimirlo con el mismo comando que los otros servicios que también traían esta versión, para descomprimir utilice el siguiente comando:

```
root@alex-bi:/usr/local# tar -zxvf apache-kylin-2.2.0-bin-hbase1x.tar.gz
apache-kylin-2.2.0-bin/
apache-kylin-2.2.0-bin/bin/
apache-kylin-2.2.0-bin/bin/check-env.sh
apache-kylin-2.2.0-bin/bin/check-migration-acl.sh
```

Figura 79. Descomprimir Kylin. Alejandro Reina

Una vez descomprimido, lo movemos al directorio `"/usr/local/Kylin/"` y establecemos los PATH.

Cabe destacar que esta es la información de la documentación oficial:

Install Kylin

1. Download latest Kylin binaries at <http://kylin.apache.org/download>
2. Export KYLIN_HOME pointing to the extracted Kylin folder
3. Make sure the user has the privilege to run hadoop, hive and hbase cmd in shell. If you are not so sure, you can run `bin/check-env.sh`, it will print out the detail information if you have some environment issues.
4. To start Kylin, run `bin/kylin.sh start`, after the server starts, you can watch `logs/kylin.log` for runtime logs;
5. To stop Kylin, run `bin/kylin.sh stop`

Figura 80. Guía oficial para instalar Kylin. Alejandro Reina

En la documentación oficial de Kylin, nos indica establecer el PATH de KYLIN_HOME, y probar con el fichero `"check-env.sh"`, el cual es un script que comprueba si la configuración del entorno es correcta.

Tras establecer el PATH seguía sin funcionar, y aunque ejecutemos el fichero `"check-env.sh"`, donde está el error, la información del error que nos da es insuficiente para saber qué es lo que está fallando, y si tienes un error no hay una documentación, foro o comunidad donde encontrar soluciones.

El fichero check-env.sh ejecuta otros scripts individuales sobre cada una de las dependencias de Kylin, llamados “find-*” como vemos en la siguiente imagen:

```
root@alex-bi: /usr/local/kylin/bin# ls
check-env.sh          find-spark-dependency.sh  load-hive-conf.sh
check-migration-acl.sh  get-properties.sh        meta
diag.sh              header.sh                metastore.sh
find-hadoop-conf-dir.sh health-check.sh          sample.sh
find-hbase-dependency.sh hive-metastore-2.1.1.jar sample-streaming.sh
find-hive-dependency.sh kylin-port-replace-util.sh
find-kafka-dependency.sh kylin.sh
```

Figura 81. Scripts de comprobación de dependencias. Alejandro Reina

Explorando estos ficheros, viendo la línea del script en la que fallaba junto con información sobre su arquitectura (11), pude inferir que Kylin además necesitaba variables de entorno para el directorio de configuración, librerías, y HCatalog de Hive.

Por tanto, definiremos los PATH o las nuevas variables de entorno necesarias para Kylin, como vemos en la siguiente imagen.

```
export HIVE_CONF=/usr/local/hive/conf
export HCAT_HOME=/usr/local/hive/hcatalog
export HIVE_LIB=/usr/local/hive/lib
```

Figura 82. PATH adicionales de Kylin. Alejandro Reina

A pesar de que, en este punto, todos los servicios estaban integrados correctamente y en funcionamiento, seguía con serios problemas para hacer funcionar este servicio.

En este punto tuve que revertir la instalación del entorno, hasta aquí tenía Hive con motor Derby, y a pesar de que estaban todos los servicios integrados y funcionando (Spark accedía a Hive, Hive podía ver las creadas en HBase), tuve que revertir la instalación ya que fue imposible conectar Kylin a Hive con motor Derby.

Es importante que el fichero “Hive-site.xml” este muy bien configurado, driver, url de conexión, usuario de acceso, como indique en el apartado de configuración de Hive, si no es así, aunque otros servicios funcionen correctamente, Kylin no lo hará. Una vez revertida la instalación, y pudiendo configurar mejor dicho fichero, como detalle anteriormente, pude finalmente arrancar el servicio.(estos pasos no han sido incluidos en la memoria por motivos de claridad, reflejando solo en este trabajo la información de la configuración final, aunque es mencionado a fin de aportar más información sobre esta tecnología, y los problemas que un usuario puede encontrarse al intentar configurarlo adecuadamente)de manera oficial más información sobre y de utilizar motor MySQL para Hive, y pudiendo configurar mejor dicho fichero, como detalle anteriormente, pude finalmente arrancar el servicio.

Si todo ha ido de manera correcta, tras un chequeo de las dependencias y de establecer conexión a través del Zookeeper de HBase, veremos algo como en la siguiente imagen:

```
A new Kylin instance is started by root. To stop it, run 'kylin.sh stop'
Check the log at /usr/local/kylin/logs/kylin.log
Web UI is at http://<hostname>:7070/kylin
root@alex-bi: /usr/local/kylin/bin#
```

Figura 83. Iniciando Kylin. Alejandro Reina

Kylin al igual que el resto de servicios, y como vemos en la anterior imagen, también dispone de una interfaz webUI, podemos acceder a dicha interfaz en la ruta “localhost:7070/kylin”, e introducir el usuario y contraseña por defecto de Kylin que es el usuario “ADMIN” y contraseña “KYLIN”, una vez accedemos, veremos algo como en la siguiente imagen:

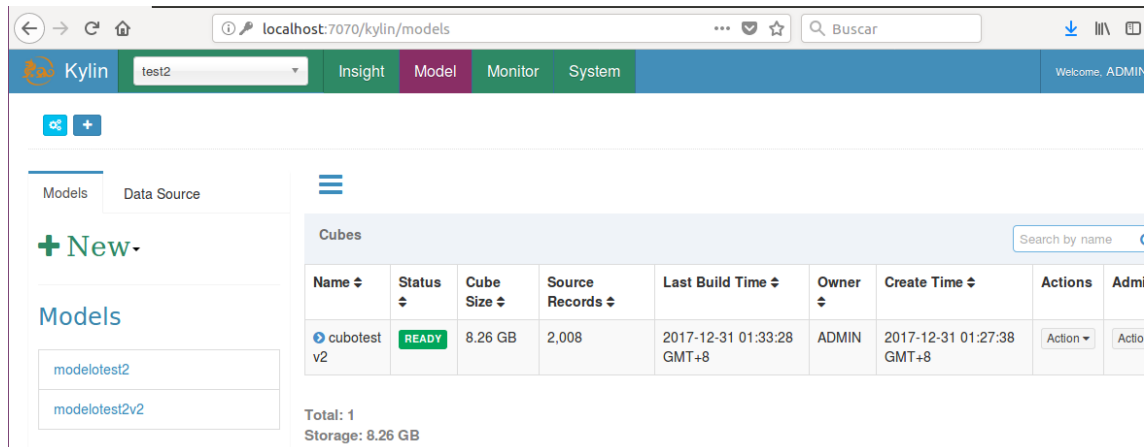


Figura 84. Interfaz Kylin. Alejandro Reina

5.2.8. Docker

Una vez entendidos y configurados todos los servicios, diseñare el script completo el cual ejecutaremos en servidores para un despliegue rápido de la arquitectura.

Antes de empezar, me gustaría destacar las distintas maneras con las que podemos trabajar con Docker y particularidades de su funcionamiento.

Cuando ejecutamos un script de Docker (Dockerfile o Docker Compose), lo que ocurre es que se crea una imagen con los ficheros necesarios para ejecutar dicha imagen en un contenedor en cualquier otra máquina que tenga instalado Docker o crear una imagen compuesta de otras imágenes de la comunidad. La imagen puede ser creada sobre una imagen con el sistema operativo base(Ubuntu, FreeBSD u otras más livianas), y sobre el contenedor en ejecución ir descargando e instalando los servicios, llegando finalmente a la imagen final que desplegaremos en otros servidores, pero esta manera de trabajar no es la más adecuada, ya que si debemos de hacer algún cambio, tras haberlo hecho, debemos de descargar esa imagen con su peso(pueden ser Gigas), en todos los servidores, y volver a arrancar, esto actúa como caja negra para los propios desarrolladores, porque físicamente solo disponen de la imagen final, y no de los posibles cambios que haya podido tener el entorno, con lo cual dificulta considerablemente solucionar algún problema que puede surgir(y surgirá, Ley de Murphy), por lo que tendremos serios problemas.

De hecho, esto es un problema al apoyar un entorno de producción sobre las imágenes ya creadas por la comunidad de Docker, ya que hay imágenes que son composición de otras imágenes, así que, claramente cualquier problema que tengamos en ese punto difícilmente solucionaremos.

La otra manera de crear una imagen es a través de un Dockerfile mencionado anteriormente, que es un documento de texto(script) que contiene todos los comandos que Docker tiene que ejecutar para ensamblar una imagen. (69) Para mi esta es la manera ideal, ya que no solo tienes la imagen final, si no que sabes paso por paso los comandos ejecutados para construir esa imagen, pudiendo depurar mejor cualquier problema que podamos tener o ampliar dicho servidor, aunque a pesar de esto, la metodología empleada a la hora de generar este script también es importante, ya que algunos script copiaban directamente una carpeta con los ficheros de un determinado servicio(por ejemplo, copiar las carpetas de los servicios anteriormente detallados en este proyecto) esto no nos aporta ninguna información sobre la construcción de dicho servicio, por lo que estaríamos en el mismo caso anterior, donde tenemos una caja negra a la hora de solucionar algún problema, o de hacer alguna modificación.

Por lo que la manera ideal de crear una imagen es ir haciendo las modificaciones una por una, aunque desarrollar esto de esta manera es más lento en términos de producción, aporta una traza sobre los pasos de despliegue del servidor.

Una vez aclarado las distintas metodologías a la hora de desplegar una imagen Docker, pasamos a instalar Docker, en mi caso, va a ser instalado sobre otra máquina virtual distinta a la utilizada anteriormente para configurar los servicios, por lo cual será un entorno limpio.

Antes de instalar Docker, es importante que tengamos actualizado nuestro sistema, podemos actualizarlo con el siguiente comando:

```
alex@alex-docker:~$ sudo apt-get update
Obj:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu xenial InRelease
Des:3 http://es.archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Des:4 http://es.archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Descargados 204 kB en 1s (201 kB/s)
Leyendo lista de paquetes... Hecho
```

Figura 85. Actualizar sistema. Alejandro Reina

Una vez actualizado nuestro sistema, añadiremos la clave del servidor a nuestro repositorio para poder descargar e instalar Docker, como vemos en la siguiente imagen:

```
alex@alex-docker:~$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADB76221572C52609D
Executing: /tmp/tmp.6u40Rg1DKi/gpg.1.sh --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADB76221572C52609D
gpg: solicitando clave 2C52609D de hkp servidor p80.pool.sks-keyservers.net
gpg: clave 2C52609D: clave pública "Docker Release Tool (releasedocker) <docker@docker.com>" importada
gpg: Cantidad total procesada: 1
gpg: importadas: 1 (RSA: 1)
```

Figura 86. Añadir clave del servidor al repositorio. Alejandro Reina

Una vez hecho esto, añadida la clave, volveremos a actualizaremos el repositorio y actualizaremos las políticas para bajar del repositorio oficial de Docker:

```
alex@alex-docker:~$ sudo apt-get update
Obj:1 http://es.archive.ubuntu.com/ubuntu xenial InRelease
Des:2 http://es.archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Obj:3 http://security.ubuntu.com/ubuntu xenial-security InRelease
Des:4 http://es.archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Des:5 https://apt.dockerproject.org/repo ubuntu-xenial InRelease [48,7 kB]
Des:6 https://apt.dockerproject.org/repo ubuntu-xenial/main amd64 Packages [4.17
7 B]
Descargados 257 kB en 2s (120 kB/s)
Leyendo lista de paquetes... Hecho
alex@alex-docker:~$ apt-cache policy docker-engine
docker-engine:
  Instalados: (ninguno)
  Candidato: 17.05.0~ce-0~ubuntu-xenial
```

Figura 87. Actualizar repositorio y políticas. Alejandro Reina

Ahora si instalaremos Docker con el siguiente comando:

```
alex@alex-docker:~$ sudo apt-get install -y docker-engine
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
 aufs-tools cgroupfs-mount git git-man liberror-perl
Paquetes sugeridos:
 git-daemon-run | git-daemon-sysvinit git-doc git-el git-e
 gitweb git-arch git-cvs git-mediawiki git-svn
```

Figura 88. Instalar Docker. Alejandro Reina

Docker funciona como un servicio de Linux, por lo que, si ejecutamos el comando “*systemctl status docker*”, podemos verificar que se ha instalado correctamente y que lo tenemos en funcionamiento, como vemos en la siguiente imagen:

```
alex@alex-docker:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
  Active: active (running) since lun 2017-09-04 16:43:00 CEST; 31s ago
  Docs: https://docs.docker.com
  Main PID: 7581 (dockerd)
  CGroup: /system.slice/docker.service
          └─7581 /usr/bin/dockerd -H fd://
          └─7594 docker-containerd -l unix:///var/run/docker/libcontainerd/doc
```

Figura 89. Instalar Docker. Alejandro Reina

Podemos utilizar el comando “*docker version*” para comprobar la versión que tenemos de cliente y de servidor, esto es útil para conectar entre distintos servidores Docker.

```
alex@alex-docker:~$ sudo docker version
Client:
Version:      17.05.0-ce
API version:  1.29
Go version:   go1.7.5
Git commit:   89658be
Built:        Thu May  4 22:10:54 2017
OS/Arch:      linux/amd64

Server:
Version:      17.05.0-ce
API version:  1.29 (minimum version 1.12)
Go version:   go1.7.5
Git commit:   89658be
Built:        Thu May  4 22:10:54 2017
OS/Arch:      linux/amd64
Experimental: false
alex@alex-docker:~$
```

Figura 90. Comando docker version. Alejandro Reina

5.2.8.1. Comandos Docker para imágenes y contenedores

Antes de entrar a crear el script, me gustaría detallar algunos comandos importantes que son necesarios a la hora de crear y ejecutar imágenes.

Tabla 3. Principales comandos Docker – Alejandro Reina Reina

Comando	Descripción
docker ps -a	Lista todos los contenedores
docker images	Lista las imágenes que tenemos en local
docker version	Muestra la versión del cliente y del servidor
docker rmi -f nombreImagen	Fuerza el borrado de una imagen.
docker rm -f nombreContenedor	Fuerza el borrado de un contenedor.
docker build -t="nombreImagen"	Construye una imagen a partir de un fichero Dockerfile, este comando debemos ejecutarlo desde el directorio donde este el fichero Dockerfile.
docker run -it nombreImagen	Ejecuta el modo interactivo de una imagen, pudiendo hacer modificaciones en dicha imagen de manera manual (no recomendado para crear imágenes, aunque si nos permite hacer pruebas sobre una imagen). Podemos mapear también los puertos que utilizara esta imagen añadiendo -p puertoOrigen:puertoDestino al comando.
docker create --name nombreContenedor -it nombreImagen	Crea un contenedor basado en la imagen "nombreImagen".

<code>docker start nombreContenedor</code>	Inicia un contenedor.
<code>docker exec -it nombreContenedor /bin/bash/</code>	Ejecuta en un contenedor una instrucción, en el ejemplo ejecutamos la consola.

Hay muchos más comandos, pero estos son los principales a la hora de crear y ejecutar una imagen/contenedor con Docker, que podemos encontrar en la documentación oficial de Docker. (69)

5.2.8.2. Comandos Dockerfile

También quiero destacar con el fin de entender mejor el script algunas palabras reservadas que utiliza Docker en sus scripts.

Tabla 4. Principales comandos de un script Dockerfile – Alejandro Reina Reina

Palabra reservada	Finalidad
FROM ubuntu 16.04	Indica el sistema operativo base sobre el cual se montará nuestra imagen, en el ejemplo es Ubuntu 16.04
MAINTAINER Alex "arr62@alu.ua.es"	Datos del administrador de la imagen
RUN apt-get install -y XXXXX update && \apt-get update	Ejecuta un comando, utilizamos los símbolos && \ para concatenar comandos en una sola ejecución, el símbolo \ indica salto de línea
ENV	Establece una variable de entorno
COPY	Copia un fichero
ADD	Añade un fichero
EXPOSE	Indica que puertos escuchara la imagen

5.2.8.3. Script Dockerfile

Una vez aclarado los principales comandos a utilizar, veremos el fichero Dockerfile creado, es decir el script que creara la imagen:

```
#Version: 1.0
FROM ubuntu:16.04
MAINTAINER Alejandro Reina Reina "alex.konu@gmail.com"
#Actualizacion de la lista de paquetes y "actualizaciones" inicial
RUN apt-get update
```

```
#####JAVA#####  
  
#Version de Java: 8  
  
#Instalamos los paquetes necesarios para ejecutar el comando apt-add-repository  
#y resincronizamos los paquetes  
  
RUN apt-get install -y software-properties-common && \apt-get update  
  
#Añadimos el repositorio de Java y resincronizamos paquetes  
  
RUN apt-add-repository -y ppa:webupd8team/java && \apt-get update  
  
#Establecemos opciones de configuracion para aceptar la licencia de java  
#automaticamente  
  
RUN echo oracle-java8-installer shared/accepted-oracle-license-v1-1 select  
true | /usr/bin/debconf-set-selections  
  
#Instalar Java  
  
RUN apt-get install -y oracle-java8-installer  
  
#Establecer PATH JAVA  
  
ENV export JAVA_HOME = /usr/lib/jvm/java-8-oracle  
  
#Prueba de vida  
  
RUN java -version  
  
#####SCALA#####  
  
#Version de scala: 2.11.6  
  
#Descarga de Scala  
  
RUN wget http://www.scala-lang.org/files/archive/scala-2.11.6.tgz  
  
#Descomprimos, movemos al directorio /usr/local y liberamos espacio del tgz  
  
RUN tar xvf scala-2.11.6.tgz && \  
    mv scala-2.11.6 /usr/local/scala && \  
    rm -R scala-2.11.6.tgz  
  
#Establecemos PATH de Scala  
  
ENV SCALA_HOME=/usr/local/scala  
  
ENV PATH=$SCALA_HOME/bin:$PATH  
  
#Prueba de vida  
  
RUN scala -version  
  
#####APACHE SPARK #####  
  
#Version de Spark: 2.2.0(Jul 11 2017) con Pre-built para Hadoop 2.7  
  
#Descarga de Spark
```

```
RUN wget https://d3kbcqa49mib13.cloudfront.net/spark-2.2.0-bin-hadoop2.7.tgz
#Descomprimir, mover al directorio /usr/local/spark y liberar espacio
RUN tar xvf spark-2.2.0-bin-hadoop2.7.tgz && \
    mv spark-2.2.0-bin-hadoop2.7 /usr/local/spark && \
    rm -R spark-2.2.0-bin-hadoop2.7.tgz
#Establecer PATH de Spark
ENV SPARK_HOME=/usr/local/spark
ENV PATH=$SPARK_HOME/bin:$PATH
COPY spark-env.sh /usr/local/spark/conf
#Prueba de vida
RUN spark-shell
##### HBASE #####
#version de hbase 1.2.6
#Descarga de Hbase
RUN wget http://apache.rediris.es/hbase/1.2.6/hbase-1.2.6-bin.tar.gz
#Descomprimir y mover al directorio /usr/local/hbase
RUN tar xvf hbase-1.2.6-bin.tar.gz && \
    mv hbase-1.2.6 /usr/local/hbase && \
    rm -R hbase-1.2.6-bin.tar.gz
#Copiamos ficheros de configuracion, modo STANDALONE
#Este fichero se ha modificado para añadir el PATH de JAVA
COPY hbase-env.sh /usr/local/hbase/conf
#Este fichero establece donde hbase guardara los archivos
# y los archivos de zookeeper
COPY hbase-site.xml /usr/local/hbase/conf
#Establecemos PATH de HBase
ENV HBASE_HOME=/usr/local/hbase
ENV PATH=$HBASE_HOME/bin:$PATH
#Arrancamos el servicio de HBase
RUN /usr/local/hbase/bin/start-hbase.sh
#####HIVE#####
#version de hive:2.1.1
```

```
#DEPENDENCIAS

#Notas de la instalacion: Hive requiere librerias de hadoop para su
#funcionamiento

# ademas sustituiremos el motor derby de Hive por MySQL

#####HADOOP#####

#version de hadoop: 2.7.4

#Descarga de Hadoop

RUN wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.4/hadoop-
2.7.4.tar.gz

#Descomprimir y mover al directorio /usr/local/hadoop

RUN tar xvf hadoop-2.7.4.tar.gz && \
    rm -R hadoop-2.7.4.tar.gz && \
    mv hadoop-2.7.4 /usr/local/hadoop

#Fichero modificado para añadir el PATH de java en Hadoop
COPY hadoop-env.sh /usr/local/hadoop/etc/hadoop

#Establecer PATH de hadoop

ENV HADOOP_HOME=/usr/local/hadoop
ENV PATH=$PATH:$HADOOP_HOME/bin
ENV PATH=$PATH:$HADOOP_HOME/sbin
ENV HADOOP_MAPRED_HOME=$HADOOP_HOME
ENV HADOOP_COMMON_HOME=$HADOOP_HOME
ENV HADOOP_HDFS_HOME=$HADOOP_HOME
ENV YARN_HOME=$HADOOP_HOME
ENV HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

#####MYSQL#####

#version de mysql:5.7

#Establecer debconf para mysql

RUN echo "mysql-server-5.7 mysql-server/root_password password bigdata" |
debconf-set-selections

RUN echo "mysql-server-5.7 mysql-server/root_password_again password bigdata"
| debconf-set-selections

#Instalacion de mysql

RUN apt-get update && apt-get install -y mysql-server
```

```
#creamos usuario para Hive en MySQL
ADD mysql-hive.sql /usr/local/
RUN service mysql start && \
    mysql -u root -pbigdata < /usr/local/mysql-hive.sql
#####HIVE#####
#Descarga de Hive
RUN wget https://apache.rediris.es/hive/hive-2.1.1/apache-hive-2.1.1-
bin.tar.gz
#Descomprimir y mover al directorio /usr/local/hive
RUN tar xvf apache-hive-2.1.1-bin.tar.gz && \
    rm -R apache-hive-2.1.1-bin.tar.gz && \
    mv apache-hive-2.1.1-bin /usr/local/hive
#Establecer el PATH de Hive
ENV HIVE_HOME=/usr/local/hive
ENV PATH=$HIVE_HOME/bin:$PATH
#Establecemos path adicionales de Hive para Kylin
ENV HIVE_CONF=/usr/local/hive/conf
ENV HCAT_HOME=/usr/local/hive/hcatalog
ENV HIVE_LIB=/usr/local/hive/lib
#Fichero de configuracion de hive con el PATH DE hadoop
COPY hive-config.sh /usr/local/hive/bin
#Fichero de configuracion hive site
COPY hive-site.xml /usr/local/hive/conf
#Integracion de Hive con MySQL y Spark
#Librerias Java para MySQL
RUN apt-get install libmysql-java && \
    ln -s /usr/share/java/mysql-connector-java.jar /usr/local/hive/lib/
#Librerias Hive para Spark
RUN cp /usr/local/hive/lib/mysql-connector-java.jar /usr/local/spark/jars/ && \
    cp /usr/local/hive/lib/hive-jdbc-2.1.1.jar /usr/local/spark/jars/
#Borramos la bd por defecto, ya que da problemas
RUN rm -R metastore_db
```

```
#Inicializamos la bd por defecto a MySQL
RUN service mysql start && \
    /usr/local/hive/bin/schematool -initSchema -dbType mysql
#Prueba de vida
RUN service mysql start && \
    /usr/local/hive/bin/hive
#####ZEPPELIN#####
#Version: 7.3.0
#Copiamos una carpeta Zeppelin previamente configurada(con un interprete para
Hive)
ADD zeppelin /usr/local/zeppelin
#Ficheros modificados:
#zeppelin-env.sh establecemos PATH de Java
#zeppelin-site.xml desactivamos el acceso anonimo a la aplicacion
#shiro.ini informacion de usuarios por defecto admin/password1
#####KYLIN#####
#version 2.2.0
#Descargamos Kylin, descomprimos y movemos al directorio /usr/local/
RUN wget http://ftp.cixug.es/apache/kylin/apache-kylin-2.2.0/apache-kylin-
2.2.0-bin-hbase1x.tar.gz && \
    tar -zxvf apache-kylin-2.2.0-bin-hbase1x.tar.gz && rm -R apache-kylin-
2.2.0-bin-hbase1x.tar.gz && \
    mv apache-kylin-2.2.0-bin /usr/local/kylin/
#Establecemos variables de entorno
ENV KYLIN_HOME=/usr/local/kylin
ENV PATH=$KYLIN_HOME/bin:$PATH
#Añadimos script de inicio del contenedor
ADD init.sh /usr/local/
#Otorgamos permisos al fichero
RUN chown root:root /usr/local/init.sh && chmod 700 /usr/local/init.sh
#Abrimos puertos de los servicios
#Spark 4040
#HBase 16010
```

```
#Hive 10002
```

```
#Zeppelin 8080
```

```
#Kylin 7070
```

```
EXPOSE 4040 7070 8080 10002 16010
```

```
#Automatizamos el arranque de los servicios cuando iniciemos el contenedor
```

```
CMD /usr/local/init.sh && /bin/bash
```

Sobre dicho script, cabe destacar un par de puntos, donde me he apoyado en otros scripts para poder ejecutar la configuración completa del script.

La instalación de MySQL mediante un script es diferente a las demás, ya que necesitamos establecer una variable de entorno en la instalación para cuando se pida un usuario y contraseña para root, sean introducidas de manera automática, ya que la instalación de MySQL, a diferencia de otros servicios, tiene una instalación guiada, y ejecutada normalmente en el script, este se quedaba a la espera de la entrada de datos para poder continuar, y ya que buscamos una automatización completa, establecemos las siguientes variables debconf.

```
RUN echo "mysql-server-5.7 mysql-server/root_password password bigdata" | debconf-set-selections
```

```
RUN echo "mysql-server-5.7 mysql-server/root_password_again password bigdata" | debconf-set-selections
```

Una vez resuelto este punto, nos encontramos con otro punto a destacar, Hive utilizara MySQL como motor, pero para ello necesita acceder mediante un usuario y contraseña y una BD física, esto implica que, en la automatización del despliegue del entorno, deberemos de crear una BD, usuario y contraseña, esto implica abrir MySQL, introducir datos y cerrarlo para poder seguir con el despliegue del entorno, por lo que nos apoyaremos en el fichero mysql-hive.sql, el cual vemos a continuación:

```
GNU nano 2.5.3 Archivo: mysql-hive.sql
create database metastore;
use metastore;
CREATE USER 'hive'@'%' IDENTIFIED BY 'bigdata';
GRANT all on *.* to 'hive'@localhost identified by 'bigdata';
flush privileges;
```

Figura 91. Mysql-hive.sql. Alejandro Reina

Este script será ejecutado mediante la siguiente línea:

```
RUN service mysql start && \
```

```
mysql -u root -pbigdata < /usr/local/mysql-hive.sql
```

Cabe puntualizar, que cuando creamos la imagen y ejecutamos RUN, si se ejecuta correctamente comitea el resultado de ejecutar dicho comando, o dicho de otra manera, comitea los archivos modificados de ejecutar el comando, pero la ejecución de un servicio no se guarda, por lo que cada vez que queramos hacer un cambio que requiera que un servicio este inicializado, deberemos ejecutarlo previamente en la misma línea, como es en el ejemplo anterior, donde ejecutamos MySQL, lo arrancamos e introducimos el script que crea el usuario, y al acabar la ejecución del comando, cierra y continua con la siguiente línea del Dockerfile.

El propósito del comando CMD, es el de proporcionar un valor predeterminado para la ejecución de un contenedor, es decir, ejecutara el comando no en la creación de la imagen, si no cuando despluguemos el contenedor, de esta manera, automatizaremos también el arranque de los servicios y datos que necesite nuestro contenedor/servidor.

Si tenemos más de un CMD en nuestro Dockerfile, solo se ejecutará el ultimo comando dentro del fichero, por lo crearemos un script ejecutado con CMD que automatizará el despliegue de los servicios de nuestro servidor, init.sh.

```
GNU nano 2.5.3 Archivo: init.sh
service mysql start
/usr/local/hbase/bin/start-hbase.sh
hive --service hiveserver2 &
sleep 20
/usr/local/zeppelin/bin/zeppelin-daemon.sh start
/usr/local/kylin/bin/kylin.sh start
```

Figura 92. Init.sh. Alejandro Reina

El servicio de Hiveserver2 es especial, ya que no es un servicio que se inicia y te devuelve el flujo de ejecución, para que puedas seguir utilizando, es un servicio similar a un servidor web tomcat, el cual se queda a la escucha, por lo que al desplegar nuestro servidor, necesitamos dos Shell, una sobre la cual ejecutamos los comandos, y otra subshell que ejecutara hiveserver2 quedándose a la escucha, pero devolviendo el flujo a la Shell principal donde seguiremos ejecutando comandos, es fundamental hacerlo así, ya que Kylin requiere que hiveserver este desplegado, para poder cargar la configuración de este y poder acceder a Hive, por eso además, hacemos un sleep 20, para asegurarnos que Hive ha sido desplegado antes de arrancar Kylin.

Finalmente, este script lo lanzaremos con la siguiente línea:

```
CMD /usr/local/init.sh && /bin/bash
```

Además, ejecutamos el comando /bin/bash/ que después de ejecutar el script abrirá una terminal para su uso, esto es necesario ya que, por defecto, cuando ejecutamos un comando sobre un contenedor, este muere al finalizar la ejecución de dicho comando, de esta manera, ejecutamos una consola, para detener la finalización de dicho contenedor, ya que queremos un contenedor escuchando donde ejecutaremos nuestros servicios. Esto ocurre ya que los servicios instalados, son tecnologías o programas con un determinado propósito, y no un servicio como tal.

5.2.8.4. Construcción de la imagen y despliegue del contenedor

Una vez explicado el Dockerfile, veremos cómo ejecutar dicho script y arrancar un contenedor para tener nuestro servidor en marcha, en primer lugar, construiremos la imagen utilizando un fichero Dockerfile, para ello, debemos de situarnos en la carpeta donde tenemos el fichero Dockerfile y los ficheros necesarios para su ejecución como vemos en la siguiente imagen.

```

root@alex-docker:/usr/local/dockerfile# ls
Dockerfile          hbase-env.sh      hive-site.xml     spark-env.sh
Dockerfile.save    hbase-site.xml   init.sh           zeppelin
hadoop-env.sh      hive-config.sh   mysql-hive.sql
root@alex-docker:/usr/local/dockerfile# docker build -t="bigdata" .
Sending build context to Docker daemon 1.33GB
Step 1/66 : FROM ubuntu:16.04
--> ccc7a11d65b1
Step 2/66 : MAINTAINER Alejandro Reina Reina "alex.konu@gmail.com"
--> Using cache
--> 26762d765a84
Step 3/66 : RUN apt-get update
--> Using cache

```

Figura 93. Docker build. Alejandro Reina

Si nuestro fichero Dockerfile es correcto, nos indicara que se ha creado correctamente, y al ejecutar el comando “docker Images”, donde veremos todas las imágenes que tenemos creadas y disponibles en nuestro repositorio, si alguna imagen no se ha creado correctamente, se le asigna una ID temporal, que guarda la ejecución apoyándose en commit de la imagen hasta antes de que se produjese el fallo, de esta manera, agiliza la construcción de las imágenes en construcciones posteriores. Podemos ver la ejecución del comando en la siguiente imagen:

```

root@alex-docker:/usr/local/dockerfile# docker images
REPOSITORY          TAG              IMAGE ID          CREATED
SIZE
bigdata             latest          b4b5babd3239     About a minute ago
5.21GB
test               latest          3d3cdb7e3d02     16 hours ago
3.9GB
zeppelin           latest          3b8b10fc441e     2 months ago
4.67GB
hive               latest          9382517149b4     2 months ago
2.89GB
hbase              latest          c70d1681e8ab     2 months ago
1.82GB
spark              latest          6b65e2cb6584     3 months ago
1.37GB
scala              latest          c9782eb0aa00     4 months ago
930MB
java               latest          831d425d0c6e     4 months ago
872MB
holamundo          latest          7903f190928f     4 months ago
159MB
ubuntu             16.04          ccc7a11d65b1     5 months ago
120MB

```

Figura 94. Docker images. Alejandro Reina

Una vez creada la imagen, si ejecutamos el comando “docker ps” vemos todos los contenedores en ejecución:

```

root@alex-docker:/usr/local/dockerfile# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS             PORTS              NAMES
root@alex-docker:/usr/local/dockerfile#

```

Figura 95. Docker ps sin contenedores. Alejandro Reina

En este caso no hemos ejecutado todavía nuestro container, para ello utilizaremos el siguiente comando, donde nos devolverá una id si no le hemos asignado un nombre predeterminado, como vemos en la siguiente imagen:

```

root@alex-docker:~# docker run -it -p 4040:4040 -p 7070:7070 -p 8080:8080 -p 100
02:10002 -p 16010:16010 -d bigdata
a59b6e86128ce3268ed30eebdf761b63a8392108dec2085d19dbde2baa5ed56a
root@alex-docker:~#

```

Figura 96. Docker run. Alejandro Reina

Una vez ejecutado el comando, nuestro contenedor ya está en marcha, si volvemos a ejecutar el comando “docker ps” podremos comprobarlo, además de otro tipo de información como el tiempo que fue creado, su estado o la imagen en la que está basada como vemos en la siguiente imagen:

```

root@alex-docker:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS             PORTS              NAMES
a59b6e86128c      bigdata            "/bin/sh -c '/usr/..." 14 seconds ago
Up 12 seconds      0.0.0.0:4040->4040/tcp, 0.0.0.0:7070->7070/tcp, 0.0.0.0
:8080->8080/tcp, 0.0.0.0:10002->10002/tcp, 0.0.0.0:16010->16010/tcp  competent_
mccarthy

```

Figura 97. Docker run. Alejandro Reina

En este momento el contenedor está corriendo en background, si queremos unir la consola al flujo de dicho contenedor, y comprobar que todo ha ido correctamente, utilizaremos el comando “docker attach idContenedor” como vemos en la siguiente imagen:

```

root@alex-docker:~# docker attach a59b
Zeppelin start [ OK ]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.4.1.jar
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4
j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Retrieving hadoop conf dir...

```

Figura 98. Docker attach. Alejandro Reina

Cuando finalice la ejecución del script, se nos quedara abierta la consola, y ya que tenemos el flujo del contenedor al haber utilizado docker attach, por lo que podemos ejecutar el comando jps para comprobar que todos los servicios están en ejecución, o hacer alguna operación que necesitemos, como puede ser copiar archivos mediante ssh u otras.

```
A new Kylin instance is started by . To stop it, run 'kylin.sh stop'
Check the log at /usr/local/kylin/logs/kylin.log
Web UI is at http://<hostname>:7070/kylin
root@a59b6e86128c:/# jps
646 RunJar
1016 ZeppelinServer
15387 Jps
15243 RunJar
587 HMaster
root@a59b6e86128c:/#
```

Figura 99. Ejecución jps dentro del flujo del contenedor. Alejandro Reina

Independientemente de unirnos al flujo de nuestro contenedor o no, ya tendremos listo para usar dichos servicios desde cualquier máquina que tenga acceso a nuestro host, en mi caso para simplificar esto, accederé desde el propio host, pero podremos acceder desde cualquier máquina que tenga acceso al host, en la imagen vemos el webUI de HBase, aunque podríamos utilizar el notebook Zeppelin para desarrollar los ETLs en Spark, montar el cubo en Kylin o monitorizar otros servicios como HBase, Hive o Spark.

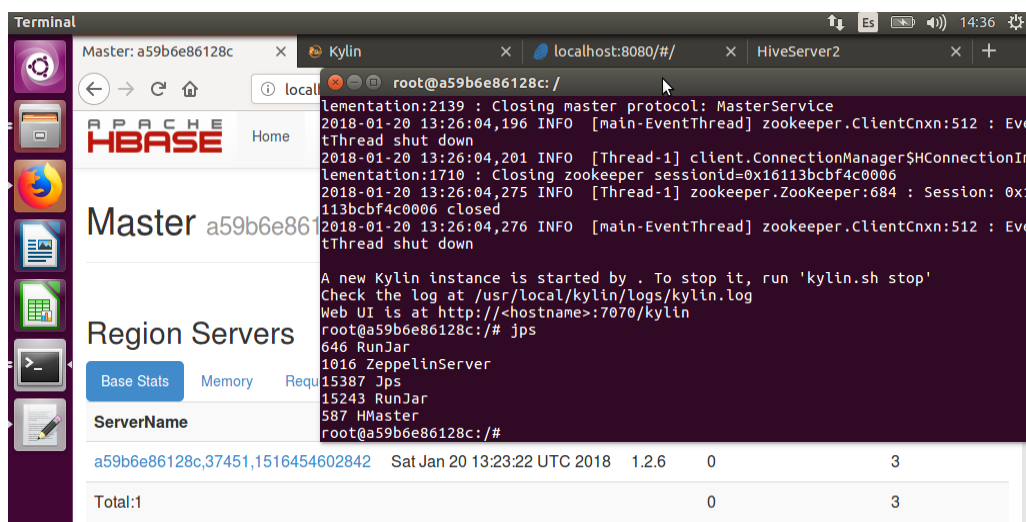


Figura 100. Acceso desde el host al container. Alejandro Reina

5.2.9. Configuración adicional: Python

Aunque Python no forma parte del script final, ya que nos hemos decantado por Scala como lenguaje de programación sobre Spark para nuestro propósito concreto, como veremos más adelante, si indicare los pasos que tuve que llevar a cabo para su instalación, y poder ejecutar pruebas con Python, ya que este no se descarta de utilizar en un futuro, para implementar machine learning.

En primer lugar, descargamos Python como vemos en la siguiente imagen:

```

root@alex-bi:/# wget https://repo.continuum.io/archive/Anaconda3-4.2.0-Linux-x86_64.sh
--2017-10-24 11:48:12-- https://repo.continuum.io/archive/Anaconda3-4.2.0-Linux-x86_64.sh
Resolviendo repo.continuum.io (repo.continuum.io)... 104.16.19.10, 104.16.18.10, 2400:cb
00:2048:1::6810:120a, ...
Conectando con repo.continuum.io (repo.continuum.io)[104.16.19.10]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 478051940 (456M) [application/x-sh]
Grabando a: "Anaconda3-4.2.0-Linux-x86_64.sh"

Anaconda3-4.2.0-Lin 100%[=====>] 455,91M  9,56MB/s  in 46s

2017-10-24 11:48:58 (9,96 MB/s) - "Anaconda3-4.2.0-Linux-x86_64.sh" guardado [478051940/
478051940]

root@alex-bi:/# █

```

Figura 101. Descarga de Python. Alejandro Reina

Una vez descargado, no es necesario establecer el Path, ya que la instalación lo hace de manera automática, si no fuese así, podemos añadirlo a nuestro fichero bashrc como vemos en la siguiente imagen:

```

GNU nano 2.5.3 Archivo: /root/.bashrc

. ~/.bash_aliases
fi

# enable programmable completion features (you don't
# this, if it's already enabled in /etc/bash.bashrc a
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix;
# . /etc/bash_completion
#fi

# added by Anaconda3 4.2.0 installer
export PATH="/usr/local/anaconda3/bin:$PATH"

```

Figura 102. Path Python. Alejandro Reina

Ahora haremos unos cambios en la configuración de Spark, para poder ejecutar PySpark (Spark con Python), para ello, añadiremos en el fichero `“./Spark/conf/spark-env.sh”` las siguientes líneas:

```

GNU nano 2.5.3 Archivo: spark-env.sh

# - SPARK_HISTORY_OPTS, to set config properties only for the history server (e$
# - SPARK_SHUFFLE_OPTS, to set config properties only for the external shuffle $
# - SPARK_DAEMON_JAVA_OPTS, to set config properties for all daemons (e.g. "-Dx$
# - SPARK_PUBLIC_DNS, to set the public dns name of the master or workers

# Generic options for the daemons used in the standalone deploy mode
# - SPARK_CONF_DIR      Alternate conf dir. (Default: ${SPARK_HOME}/conf)
# - SPARK_LOG_DIR       Where log files are stored. (Default: ${SPARK_HOME}/lo$
# - SPARK_PID_DIR       Where the pid file is stored. (Default: /tmp)
# - SPARK_IDENT_STRING  A string representing this instance of spark. (Default:$
# - SPARK_NICENESS      The scheduling priority for daemons. (Default: 0)
# - SPARK_NO_DAEMONIZE  Run the proposed command in the foreground. It will not$

JAVA_HOME=/usr/lib/jvm/java-8-oracle

PYSPARK_PYTHON=/usr/local/anaconda3/bin/python

```

Figura 103. Spark-env.sh Python. Alejandro Reina

Este paso para Scala no era necesario ya que Spark utiliza Scala por defecto. Una vez hecho esto, haremos los cambios en el notebook Zeppelin, para poder ejecutar PySpark, para ello modificaremos el fichero “./zeppelin/conf/zeppelin-env.sh”

```

GNU nano 2.5.3 Archivo: zeppelin-env.sh
# export HBASE_CONF_DIR= # (optional) Alternatively, configurati$
#### ZeppelinHub connection configuration ####
# export ZEPPELINHUB_API_ADDRESS # Refers to the address of the $
# export ZEPPELINHUB_API_TOKEN # Refers to the Zeppelin instan$
# export ZEPPELINHUB_USER_KEY # Optional, when using Zeppelin$
#### Zeppelin impersonation configuration
# export ZEPPELIN_IMPERSONATE_CMD # Optional, when user want to run inter$
# export ZEPPELIN_IMPERSONATE_SPARK_PROXY_USER #Optional, by default is true; $
export SPARK_HOME=/usr/local/spark
export PYSARK_PYTHON=/usr/local/spark
export pyspark_driver_python=/usr/local/anaconda3/bin/python
    
```

Figura 104. Zeppelin-env.sh. Alejandro Reina

Por último, de manera similar a como creamos el intérprete para Hive, modificaremos el intérprete de Spark que viene por defecto con Zeppelin para que pueda ejecutar PySpark, para ello modificaremos la siguiente línea:

```

zeppelin.pyspark.python python3
    
```

Figura 105. Zeppelin PySpark interpreter. Alejandro Reina

Una vez hecho podremos utilizar PySpark usando la cabecera %pyspark en cualquier ventana de ejecución de nuestro notebook.

5.3. Diseño de pruebas de rendimiento. Scala vs Python

En este punto se va a tratar las pruebas de rendimiento de Scala vs Python, para ello se han desarrollado cinco métodos en Scala y Python, estos métodos son exactamente iguales (consulta SQL, declaración de variables, y operaciones sobre los DataFrames) a excepción del lenguaje utilizado para implementarlos. Se ejecutarán sobre un fichero de 43MB, con los siguientes datos:

	Predetermina	Predetermina	Predeterminado		Predetermina	Predeterminado	Predetermina	Predetermina	Predeterminado
1	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
2	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom	
3	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom	
4	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom	
5	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom	
6	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom	

Figura 106. Datos Ejemplo. Alejandro Reina

En el siguiente punto podemos ver un par de métodos tanto en Scala como en Python ejemplo de las pruebas realizadas.

5.3.1. Metodos Scala

```

%spark

def metodo1(self: DataFrame, top_count: Int): DataFrame =
{
    //Inicializacion de variables
    var top_count_aux= top_count
    var df = sqlContext.emptyDataFrame
    var decimal_to_euro= udf((num:Int) => num%.2f)
    if(top_count == 0)
    {
        top_count_aux = 10
    }
    if(self != None)
    {
        df = (self
            .groupBy("CustomerID")
            .agg(sum("Total").alias("Importe Total"))
            .sort(desc("Importe Total"))
            .limit(top_count_aux))
    }
    return df
}

// Metodo 2
def metodo2(self: DataFrame, top_count:Int): DataFrame =
{
    var decimal_to_euro= udf((num:Int) => num%.2f)
    var top_count_aux = top_count
    var gb_clients = sqlContext.emptyDataFrame

    if(top_count == 0)
    {
        top_count_aux = 10
    }

    if(self != None)
    {
        gb_clients = self
            .groupBy("CustomerID")
            .agg(sum("Total").alias("Importe Total"))
            .sort(desc("Importe Total"))
            .cache()
    }

    var gb_clients_top10 = gb_clients.limit(top_count_aux)
    var nottop10_total = gb_clients.agg(sum("Importe
Total")).head().getDouble(0) - gb_clients_top10.agg(sum("Importe
Total")).head().getDouble(0)
    var df_nottop10_data = List(("Resto de clientes", nottop10_total))
    var df_nottop10 = df_nottop10_data.toDF("CustomerID", "Importe Total")

```

```
        var df_view = (gb_clients_top10//.withColumn('Posicion',
monotonically_increasing_id() + 1)
        .select("CustomerID", "Importe Total"))
        df_view = df_view.union(df_nottop10)

        return df_view
}
```

5.3.2. Metodos Python

```
import pyspark

from pyspark.sql import SparkSession

from pyspark.sql.functions import (col, concat, count, countDistinct,
current_timestamp, date_format, desc, floor, lag, lit,
monotonically_increasing_id, sum,udf, unix_timestamp)

from pyspark.sql.types import FloatType, StringType

from pyspark.sql.window import Window

#Metodo 1

def get_top_clientes_por_importe_total(self, top_count=10):

    decimal_to_euro = udf(

        lambda num: '{0:,.2f} â,-'.format(num), StringType())

    df = None

    if(top_count is None):

        top_count = 10

    if(self.app.get_invoice_lines_positive() is not None):

        df = (self.app.get_invoice_lines_positive())

            .groupBy('cliente')

            .agg(sum('Total').alias('Importe Total'))

            .sort(desc('Importe Total'))

            .limit(top_count)

    return df
```

```
#Metodo 2
def get_distribucion_de_clientes_por_importe(self, top_count=10):
    decimal_to_euro = udf(
        lambda num: '{0:,.2f} â¬'.format(num), StringType())
    df_view = None
    if top_count is None:
        top_count = 10
    if(self.app.get_invoice_lines_positive() is not None):
        gb_clients = (self.app.get_invoice_lines_positive()
            .groupBy('cliente')
            .agg(sum('Total').alias('Importe Total'))
            .sort(desc('Importe Total'))
            .cache())
        gb_clients_top10 = gb_clients.limit(top_count)
        nottop10_total = gb_clients.agg(sum('Importe Total')).collect()[0][
            0] - gb_clients_top10.agg(sum('Importe
Total')).collect()[0][0]
        df_nottop10_data = [('Resto de clientes', nottop10_total)]
        df_nottop10 = self.sparkSQL.createDataFrame(
            df_nottop10_data, schema=gb_clients_top10.schema)
        df_view = (gb_clients_top10.select('Cliente', 'Importe Total'))
        df_view = df_view.union(df_nottop10)
    return df_view
```

5.3.3. Resultados de las pruebas

Estos métodos se ejecutaron 100 veces cada uno, para comprobar su rendimiento, se han separado en dos gráficos para poder visualizar mejor las diferencias entre ambos métodos, ya que algunos métodos su ejecución es menor a 1 segundo y otros superan los 8 segundos, obteniendo los siguientes resultados:

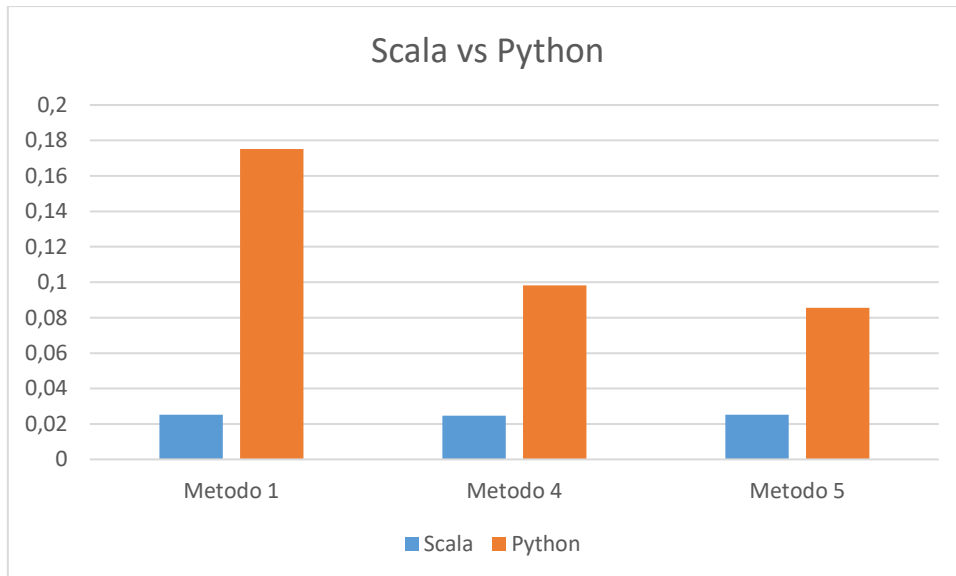


Figura 107. Scala vs Python - Tiempos. Alejandro Reina

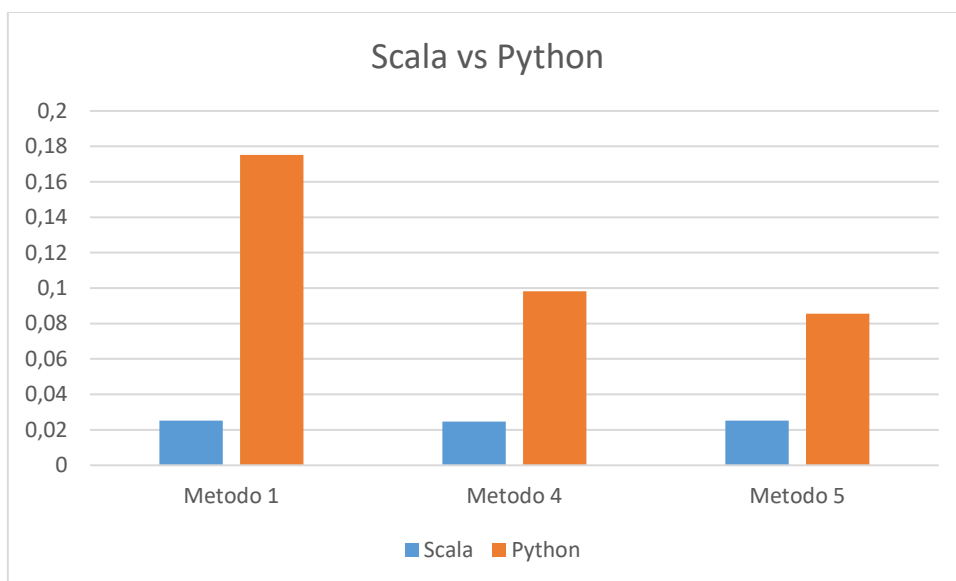


Figura 108. Scala vs Python - Tiempos. Alejandro Reina

5.3.4. Conclusiones

Como hemos podido comprobar en las pruebas anteriores, Scala es un 30% más rápido que Python, por lo que debido a la naturaleza de un datawarehouse donde vamos a realizar muchas operaciones sobre los datos (cálculo de medidas, joins, entre otros), y si lo que buscamos es una eficiencia máxima, el lenguaje a utilizar sería Scala.

A nivel personal, y una vez salvado la curva de aprendizaje inicial, no encuentro diferencia entre Python y Scala a la hora de diseñar un sistema BI, donde van a predominar una serie de funciones, por lo que para un sistema BI donde buscamos el mejor rendimiento elegiría Scala, sin embargo, si nuestro objetivo fuese diseñar un algoritmo machine learning sobre los datos,

nos decantaríamos por Python, debido a que dispone de muchas más librerías a día de hoy para machine learning, por lo cual disminuiría notablemente la dificultad de desarrollo frente a Scala, además de que no es un objetivo primordial el rendimiento(sin olvidarnos del rendimiento tampoco) que obtendríamos con Scala, si no obtener y descubrir información de esos datos.

5.4. Diseño del Datamart

5.4.1. Exploración de los datos

La primera fase antes de comenzar un análisis, pasa por definir los KPI (Key Performance Indicator), es decir, que queremos saber y medir, o preguntas analíticas a las cuales queremos contestar con nuestro sistema, en nuestro caso, al no tener preguntas analíticas predefinidas, haremos una exploración de los datos, para conocer qué información podemos obtener de ellos. En este caso, estudiaremos los dataset de la ciudad de San Francisco que encontramos en la página www.datasf.org (70), aunque disponemos de dataset de muchas áreas distintas, como puede ser salud y servicios sociales, transporte, energía, entre otros, hemos elegido el área de seguridad pública, por ser unos dataset que contienen mucha más información que los de otras áreas.

5.4.1.1. Police Department Incident

Uno de los dataset que voy a analizar es “Police Department Incident”, pesa 450MB y contiene 2,14M de filas y 13 columnas, y contiene información de los delitos producidos desde el 1/1/2013 hasta dos semanas antes de la publicación del dataset, el cual es actualizado diariamente. Otra consideración es que los datos a partir del 19 de Junio del 2015, no pueden ser comparados con los informes oficiales del departamento de policía de la ciudad de San Francisco(SFPD), ya que han implantado un nuevo sistema con una redistribución de los distritos, y los conjuntos de datos representados en el dataset provienen del sistema previo en proceso de ser retirado, aunque en nuestro caso de estudio al no compararlos con los datos oficiales de la policía, no nos afectara. (71)

IncidentNo	Category	Descript	DayOfWe	Date	Time	PdDistrict	Resolution
170902594	WEAPON LAWS	POSS OF PROHIBITED WEAPON	Sunday	11/05/2017	00:17	TARAVAL	ARREST, BOOKED
170902594	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Sunday	11/05/2017	00:17	TARAVAL	ARREST, BOOKED
170902613	RECOVERED VEHICLE	RECOVERED VEHICLE - STOLEN OUTSIDE SF	Sunday	11/05/2017	00:29	CENTRAL	ARREST, BOOKED
170902663	ROBBERY	ROBBERY ON THE STREET, STRONGARM	Sunday	11/05/2017	01:20	CENTRAL	NONE
170902679	ASSAULT	BATTERY	Sunday	11/05/2017	01:11	RICHMOND	NONE

Address	X	Y	Location	PdId
OCEAN AV / DORADO TR	-122.46120852061303	37.72495024876397	(37.72495024876397°, -122.46120852061303°)	17090259412120
OCEAN AV / DORADO TR	-122.46120852061303	37.72495024876397	(37.72495024876397°, -122.46120852061303°)	17090259465010
LEAVENWORTH ST / SUTTER ST	-122.41518693817171	37.788386679078926	(37.788386679078926°, -122.41518693817171°)	17090261307055
900 Block of POST ST	-122.41682519978644	37.787277154617044	(37.787277154617044°, -122.41682519978644°)	17090266303014
5300 Block of GEARY BL	-122.476719025527	37.780579794316736	(37.780579794316736°, -122.476719025527°)	17090267904134

Figura 109. Police_Department_Incidents.csv - data.sfgov.org

A continuación, incluyo una descripción de los datos que contiene cada uno de los campos:

IncidntNum es el identificador del delito, en algunos casos un mismo caso aparece varias veces en el dataset, lo que implica que una persona ha cometido más de un delito al mismo tiempo, por ejemplo, robar un coche y cometer un atraco.

Categoria se utiliza para clasificar el tipo de incidente ocurrido, por ejemplo, robo, asesinato, accidente de coche entre otros.

Descripcion incluye una breve descripción del incidente, este campo permitiría aplicando técnicas de minería de texto obtener y hacer estudios más concretos sobre el tipo de incidente.

DayOfWeek indica en que día de la semana se ha producido el incidente.

Date indica la fecha en la que se produce el incidente.

Time indica la hora en formato HH:mm de cuando se produce el incidente.

PdDistrict distrito donde se produce el incidente según la repartición de distritos de la ciudad de San Francisco.

Resolution nos indica información relativa a como ha finalizado el incidente, si el caso se ha cerrado con detención del implicado, o si sigue abierto entre otros.

Address nos indica en qué dirección se ha registrado el incidente.

X Coordenada GPS X

Y Coordenada GPS Y

Location coordenadas GPS X e Y

PdID este campo es un identificador, no nos aporta ninguna información directamente, es un campo que se utiliza como identificador único del incidente para operaciones de insertar o actualizar en una base de datos y poder hacer estudios sobre delitos independientes y no sobre ocurrencias delictivas (IncidntNum si podía repetirse haciendo referencia a que una misma persona cometía más de un delito al mismo tiempo).

5.4.1.2. Fire Department Calls for Service

Otro dataset que vamos a analizar es “Fire department calls for service”, que son los datos recogidos de las llamadas al departamento de bomberos de la ciudad de san francisco, este fichero pesa 1,7GB con 4,5M de filas y 34 columnas. Este dataset tiene información relativa al incidente, así como el tiempo de respuesta de las distintas unidades que intervienen en el incidente.

Call Num:	Unit :	Incident Num:	Call Type :	Call Date :	Watch Date :	Received DtTm ↓ :	Entry DtTm :
173324393	B06	17139513	Alarms	11/28/2017	11/28/2017	2017 Nov 28 11:57:51 PM	2017 Nov 28 11:59:05 PM
173324393	E11	17139513	Alarms	11/28/2017	11/28/2017	2017 Nov 28 11:57:51 PM	2017 Nov 28 11:59:05 PM
173324393	T11	17139513	Alarms	11/28/2017	11/28/2017	2017 Nov 28 11:57:51 PM	2017 Nov 28 11:59:05 PM
173324384	62	17139512	Medical Incident	11/28/2017	11/28/2017	2017 Nov 28 11:53:20 PM	2017 Nov 28 11:56:04 PM
173324352	50	17139511	Medical Incident	11/28/2017	11/28/2017	2017 Nov 28 11:40:29 PM	2017 Nov 28 11:41:18 PM

Dispatch DtTm :	Response DtTm :	On Scene DtTm :	Transport DtTm :	Hospital DtTm :	Call Final Disposition :
2017 Nov 28 11:59:51 PM	2017 Nov 29 12:01:37 AM	2017 Nov 29 12:03:12 AM			Fire
2017 Nov 28 11:59:51 PM	2017 Nov 29 12:00:58 AM	2017 Nov 29 12:02:31 AM			Fire
2017 Nov 28 11:59:51 PM	2017 Nov 29 12:02:00 AM	2017 Nov 29 12:02:42 AM			Fire
2017 Nov 28 11:56:22 PM	2017 Nov 28 11:56:34 PM	2017 Nov 29 12:04:59 AM	2017 Nov 29 12:16:22 AM	2017 Nov 29 12:23:00 AM	Code 2 Transport
2017 Nov 28 11:41:37 PM	2017 Nov 28 11:41:46 PM	2017 Nov 28 11:45:10 PM	2017 Nov 29 12:12:26 AM	2017 Nov 29 12:23:03 AM	Code 2 Transport

Available DtTm :	Address :	City :	Zipcode of Incide :	Battal :	Station A :	Box :	Original Prior :
2017 Nov 29 12:07:31 AM	200 Block of 30TH ST	San Francisco	94131	B06	11	5574	3
2017 Nov 29 12:09:03 AM	200 Block of 30TH ST	San Francisco	94131	B06	11	5574	3
2017 Nov 29 12:07:45 AM	200 Block of 30TH ST	San Francisco	94131	B06	11	5574	3
2017 Nov 29 12:45:47 AM	300 Block of BAY ST	San Francisco	94133	B01	28	1425	2
2017 Nov 29 12:52:13 AM	200 Block of 6TH ST	San Francisco	94103	B03	01	2252	3

Prior :	Final Prior :	ALS U :	Call Type Group :	Number of Alar :	Unit Type :	Unit sequence in call dispa :
3	3		Alarm	1	CHIEF	3
3	3	✓	Alarm	1	ENGINE	1
3	3		Alarm	1	TRUCK	2
2	2	✓	Potentially Life-Threatening	1	MEDIC	1
3	3	✓	Potentially Life-Threatening	1	MEDIC	1

Fire Prevention Distr :	Supervisor Distr :	Neighborhoods - Analysis Boundar :	Location :	RowID :
6	8	Glen Park	(37.74213368194098°, -122.42501817475811°)	173324393-B06
6	8	Glen Park	(37.74213368194098°, -122.42501817475811°)	173324393-E11
6	8	Glen Park	(37.74213368194098°, -122.42501817475811°)	173324393-T11
1	3	North Beach	(37.80577596711131°, -122.41278788719522°)	173324384-62
3	6	South of Market	(37.779167421896304°, -122.40634642563238°)	173324352-50

Figura 110. Fire departement calls for service.csv - data.sfgov.org

Al igual que con el anterior fichero, incluiremos una descripción de los campos:

Información que podemos obtener del dataset “Fire department calls for service”

CallNumberID: Numero único de 9 dígitos asignado por el DEM (departamento de llamadas de emergencias 911).

UnitId: Identificador único de la unidad a la que se le traspasa la llamada por el DEM.

IncidentNumber: Numero único de 8 dígitos asignado por el DEM para este “fire incident”.

Call Type: Tipo de incidente.

Call Date: Fecha de la llamada recibida en el 911. Este campo solo incluye el parámetro fecha, sin hora.

Watch Date: Fecha cuando la llamada es vista. Igual que el campo anterior, solo incluye fecha.

Received DtTm: Fecha y hora en que la llamada ha sido recibida por el departamento 911.

Entry DtTm: Fecha y hora en que el operador del 911 envía la información de la llamada al sistema.

Dispatch DtTm: Fecha y hora en que el operador envía a la unidad la información de la llamada.

Response DtTm: Fecha y hora en que la unidad confirma el envío y registra que la unidad está en camino.

On Scene DtTm: Fecha y hora en el que la unidad llega a la escena del incidente.

Transport DtTm: Si la unidad es una ambulancia, fecha y hora en que la unidad comienza el transporte al hospital.

Hospital DtTm: Si la unidad es una ambulancia, fecha y hora en que la unidad llega al hospital.

Call Final disposition: Disposición de la llamada. Por ejemplo, puede tener el campo TH2, que indica Transporte al hospital o FIR resuelto por el departamento de bomberos.

Available DtTm: Fecha y hora en el que la unidad ya no está asignada a la llamada, y está disponible para otro servicio.

Address: Dirección donde se produce el incidente.

City: Ciudad donde se produce el incidente.

Zipcode of Incident: Código postal donde se produce el incidente.

Batallion: Distrito de respuesta de emergencias (hay 9 distritos).

Station Area: Estación de bomberos más cercana de respuesta asociada a la dirección donde se ha producido el incidente.

Box: "Caja de incendios" asociada a la dirección del incidente. Una caja de incendios es la unidad más pequeña usada para dividir la ciudad.

Original Priority: Prioridad original asignada al caso.

Priority: Prioridad.

Final priority: Prioridad final asignada al caso.

ALS Unit: Este campo indica si la unidad llevaba el equipo necesario para el soporte de vidas, y si incluía un paramédico.

Call type group: Tipo de llamada (hay 4 tipos, Fuego, alarma, potencial amenaza a la vida y no potencial amenaza a la vida).

Number of Alarms: Numero de alarmas asociadas al incidente (entre 1 y 5)

Unit Type: Tipo de unidad asignada al incidente

Unit Sequence in call dispatch: Número que Indica el orden en el que la unidad ha sido asignada a la llamada, pudiendo inferir información de si el departamento DEM asigna correctamente la llamada a una unidad que pueda atenderla.

Fire Prevention District: Oficina del distrito de prevención de incendios asociada a la dirección.

Supervisor District: Supervisor del distrito

Neighborhoods – Analysis Boundaries: Barrio donde se produce el incidente

Location: Ubicación del incidente en coordenadas GPS X e Y (generalizadas a mitad de calle, intersecciones, o ubicación de la caja más cercana, para proteger la privacidad de la persona.

RowId: Combinación del Número único de llamada y la id de la unidad.

5.4.2. Definición de preguntas analíticas

Una vez hecho una exploración de los datos que disponemos vamos a definir unos KPI o preguntas analíticas a las cuales queremos dar respuesta con nuestro sistema, cabe destacar que esta fase suele ir antes de la exploración de los datos, ya que generalmente la organización cuyos datos van a ser analizados, nos definirá dichos KPI, o cual es la información que necesitan extraer de los datos, en este proyecto, no existe dicha organización por lo que definiré una vez visto los datos que disponemos, unas preguntas analíticas que nos puedan aportar información interesante.

5.4.2.1. KPI Police Department Incident

- ¿Cuáles son los incidentes más frecuentes? ¿Se producen por igual en todos los barrios?
- ¿Cuántos incidentes se saldan con la detención de alguien? ¿En cuántos no ocurre nada?
- ¿Cuándo es más frecuente en un día que ocurra un incidente?
- ¿Cuáles son los meses del año donde es más frecuente que ocurra un incidente?
- ¿Todos los tipos de robo ocurren por igual o su frecuencia se altera en el tiempo?

5.4.2.2. KPI Fire Department Calls for Service

- ¿Se producen los mismos incidentes por igual en todos los vecindarios? ¿Todos los tipos de incidentes tienen la misma proporción?
- ¿Cuál es el tiempo medio de respuesta por tipo de llamada?
- ¿Cuáles son los tipos de llamada que se producen y su frecuencia?
- ¿Todos los batallones de bomberos responden el mismo número de llamadas? ¿Hay batallones especializados en tipos específicos de llamadas?

5.4.3. Diagrama de jerarquías

En este punto del diseño, vamos a definir un diagrama de jerarquías, seleccionando y definiendo la granularidad del hecho que representará nuestro modelo, para ello definiremos dos diagramas, uno para cada fichero, que será nuestro Datamart.

5.4.3.1. Police Department Incident

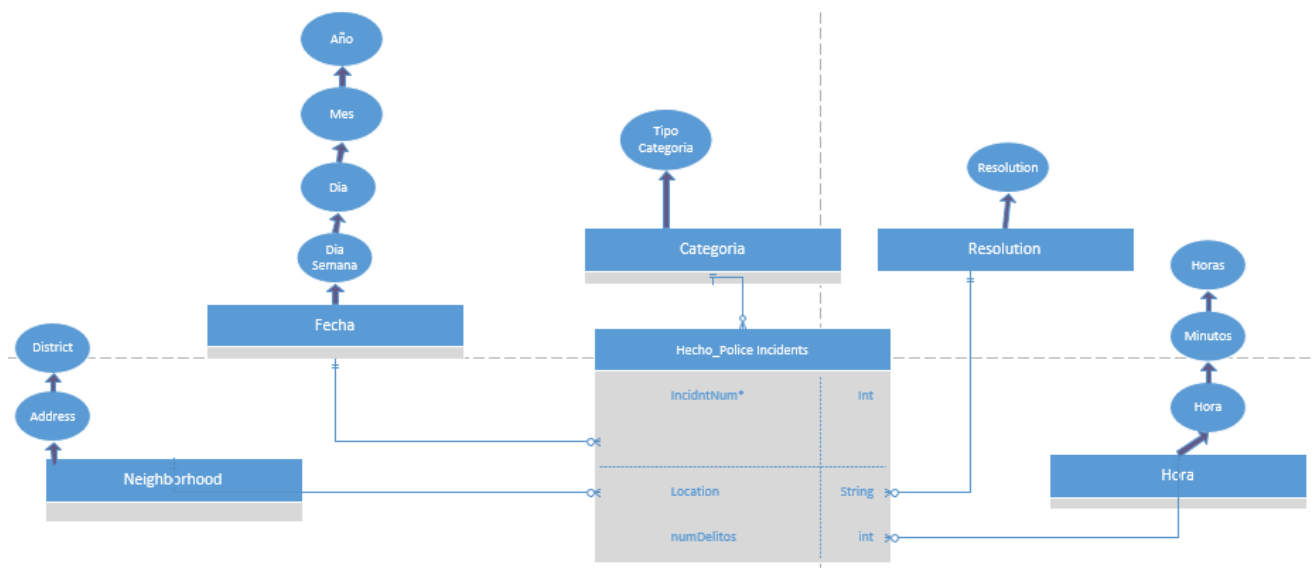


Figura 111. Diagrama de jerarquías Police Incidents – Alejandro Reina Reina

5.4.3.2. Fire Department Calls for Service

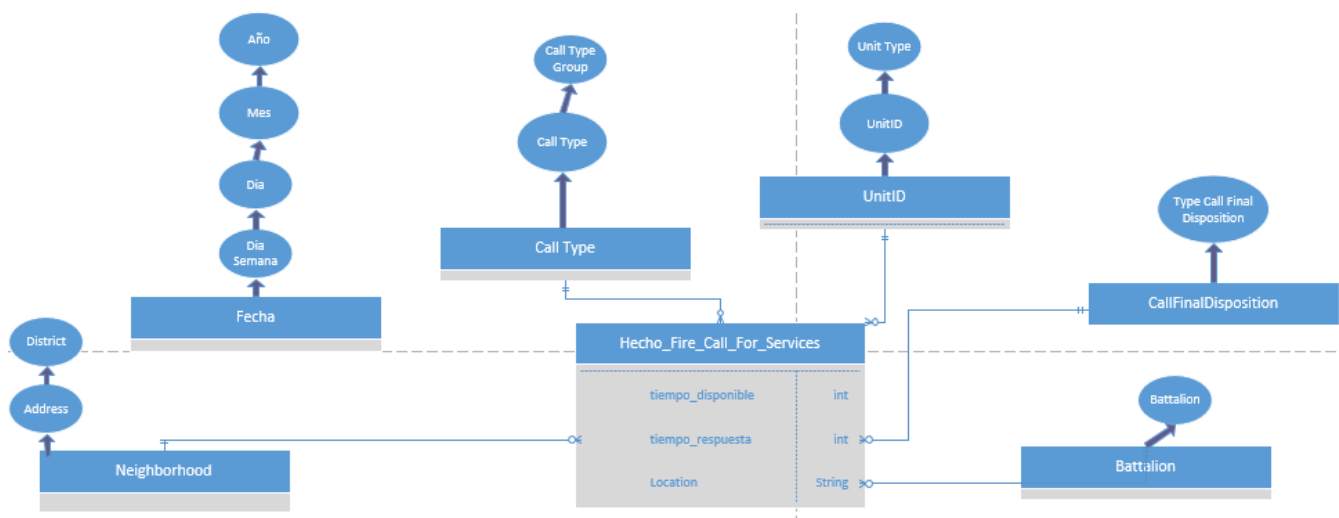


Figura 112. Diagrama de jerarquías Police Incidents – Alejandro Reina Reina

5.4.4. Esquema estrella

Una vez diseñado el diseño conceptual o diagrama de jerarquías, usare MySQL Workbench para modelar el esquema estrella basado en el esquema multidimensional de Ralph Kimball (6) donde se reflejarán los atributos de las tablas, así como las relaciones de las dimensiones con el hecho, además de definir el modelado, nos servirá para incluir esta información en el repositorio de metadatos del datawarehouse.

5.4.4.1. Police Department Incident

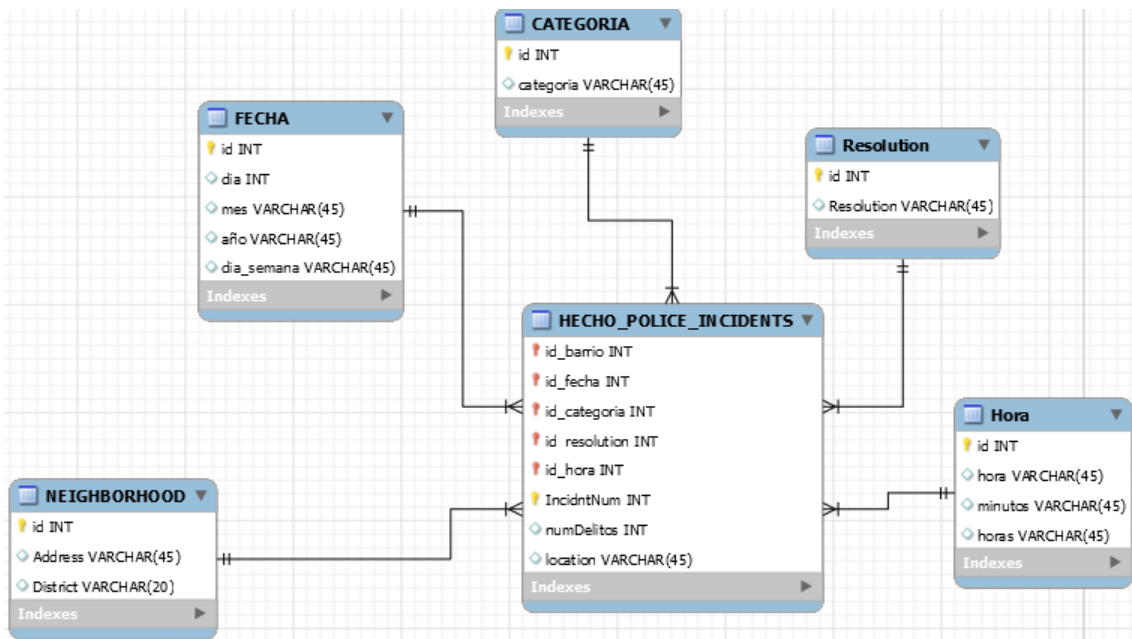


Figura 113. Diagrama estrella Police Incidents – Alejandro Reina Reina

5.4.4.2. Fire Department Calls for Service

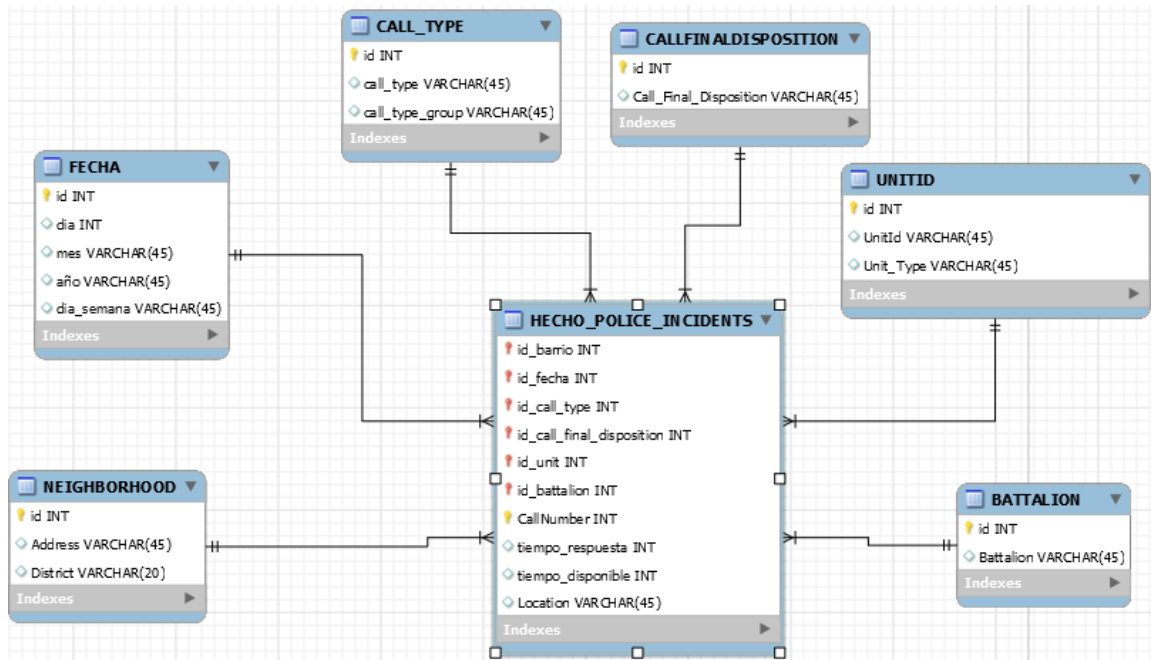


Figura 114. Diagrama estrella Fire Departments Calls for Service – Alejandro Reina Reina

Estos modelos creados, nos van a servir como repositorio de metadatos de nuestro datawarehouse, además de poder ver de un vistazo el esquema a seguir y situarnos de cara al diseño de ETL.

5.5. Diseño de ETL

En esta parte del proyecto, vamos a hablar del diseño de ETL, es decir del proceso de Extracción, transformación y carga de datos en nuestro datawarehouse.

A grandes rasgos, esta parte del diseño tendrá una parte donde cargaremos nuestras fuentes de datos, los ficheros “police_department_incidents” y “fire_department_call_for_service”, y una vez cargados los procesaremos, transformando los datos que necesitemos transformar y calculando las medidas, como por ejemplo el tiempo de respuesta de las patrullas de bomberos, y preparando los datos para crear las dimensiones y las tablas de hechos, y finalmente estas serán volcadas sobre una BD, en nuestro caso será Hive.

En primer lugar, vamos a definir dos métodos genéricos para lectura de ficheros y para volcar un DataFrame en una tabla Hive.

5.5.1. Lectura de ficheros

Este método lee un fichero en formato csv, y devuelve una variable de tipo dataframe, con los datos del fichero leído.

5.5.1.1. Código

```
//Metodo de lectura de ficheros
def read_csv_Police_Department_Incidents():DataFrame = df
{
  //lectura de ficheros
  var data = sqlContext.read.format("com.databricks.spark.csv")
    //El fichero tiene cabecera con el nombre de los campos
    .option("header","true")
    //Inferir el esquema y tipo de datos de manera automatica
    .option("inferSchema","true")
    //Delimitador entre los campos
    .option("delimiter",",")
    //Ruta del fichero
    .load("/home/alex/hbase/hfiles/testdf/Police_Department_Incidents.csv")
}
```

5.5.2. Guardar un DataFrame en una tabla Hive

Este método recibirá por parámetros un DataFrame y un nombre de tabla, y creará una tabla Hive con el contenido del DataFrame y el nombre pasado por parámetros.

5.5.2.1. Código

```
/*Metodo que dado un DataFrame y un nombre de tabla
crea una tabla hive basada en el dataframe*/
def create_hive_table(data: DataFrame, nameTable: String)
{
  //Definimos las sentencias SQL a ejecutar a ejecutar
  val drop = "drop table if exists " + nameTable
  val create = "create table " + nameTable + " as select * from tempTable"
  //Creamos HiveContext
  val hc = new org.apache.spark.sql.hive.HiveContext(sc)
  //Creamos tabla temporal que volcaremos en Hive
  data.registerTempTable("tempTable")
  //Ejecutamos sentencias SQL
```

```
    hc.sql(drop)
    hc.sql(create)
}
```

5.5.3. Diseño de la dimension Horas

A continuación, definiremos dos dimensiones genéricas, la dimension fecha y la dimension hora, el objetivo de hacerlo así, es poder reutilizar dichas dimensiones en otros datamarts o datawarehouses, y aunque en el caso particular del proyecto, la dimension hora no la reutilizaremos, si haremos uso de buenas prácticas, permitiendo la escalabilidad del proyecto en un futuro.

5.5.3.1. Código

```
def dimension_Horas()
{
    //Definicion de variables
    var listaHoras = List[String]()
    var hora = ""
    var minutos = ""
    var aux = ""
    var dfHoras = sqlContext.emptyDataFrame
    //Bucle que crea una lista con todas las horas y minutos de un dia
    for (i<- 0 to 23)
    {
        for(j<-0 to 59)
        {
            if(i < 10)
                hora = "0" + i.toString()
            else
                hora = i.toString()
            if(j < 10)
                minutos = "0" + j.toString()
            else
                minutos = j.toString
            aux = hora + ":" + minutos
        }
    }
}
```

```

        listaHoras = listaHoras ::: List(aux)
    }
}

//Convertimos la lista a dataframe, y transformamos para obtener los campos
hora, minutos e id

dfHoras = listaHoras.toDF

dfHoras = dfHoras.withColumnRenamed("value", "horas")

dfHoras = dfHoras.withColumn("id", monotonically_increasing_id())
                    .withColumn("hora", hour(dfHoras("horas")))
                    .withColumn("minutos", minute(dfHoras("horas")))

//Crear Tabla Hive

create_hive_table(dfHoras, "DIM_HORAS")
}

```

5.5.3.2. Visualización

San Francisco Took 0 sec. Last updated by admin at January 21 2018, 11:56:00 PM.

```
%hive
select * from dim_horas;
```

FINISHED

dim_horas.horas	dim_horas.id	dim_horas.hora	dim_horas.minutos
00:00	0	0	0
00:01	1	0	1
00:02	2	0	2
00:03	3	0	3
00:04	4	0	4

Figura 115. Datos dimension Horas – Alejandro Reina Reina

5.5.4. Diseño de la dimensión Fecha

Este método, lo utilizaremos para crear todas las fechas comprendidas entre dos rangos de fechas, en nuestro caso, crearemos todas las fechas comprendidas entre 2000 y 2018, para la realización de este método, me he ayudado de la librería JodaTime (72), que provee de una función que calcula el número de días naturales entre dos fechas, de una función que dada una fecha le suma un número de días, y funciones de conversión de fecha a string, aunque dispone de una amplia gama de funciones para trabajar con fechas.

5.5.4.1. Código

```
def dimension_Fechas()
{
  //Rango de fechas
  var start = new LocalDate(2000,1,1)
  var end = new LocalDate(2019,1,1)
  //Calculo del numero de dias entre dos fechas
  val numberOfDays = Days.daysBetween(start, end).getDays()
  //Generamos todas las fechas existentes entre esas dos fechas
  var fechas = for (f<- 0 to numberOfDays) yield
start.plusDays(f).toString("yyyy/MM/dd")
  //Creamos una lista con las fechas
  var listaFechas = fechas.toList
  var dfFechas = sqlContext.emptyDataFrame
  //Creamos dataframe a partir de las fechas y calculamos otros campos de
nuestro interes
  dfFechas = listaFechas.toDF
  dfFechas = dfFechas.withColumnRenamed("value","date")
  dfFechas = dfFechas.withColumn("year",year(to_date(dfFechas("date"),
"yyyy/MM/dd")))
    .withColumn("month",month(to_date(dfFechas("date"),
"yyyy/MM/dd")))
    .withColumn("day",dayofmonth(to_date(dfFechas("date"),
"yyyy/MM/dd" )))
    .withColumn("dayofweek", date_format(to_date(dfFechas ("date")
, "yyyy/MM/dd"), "EEEE"))
    .withColumn("id", monotonically_increasing_id())
  //Creamos tabla Hive
  create_hive_table(dfFechas,"DIM_FECHAS")
}
```

5.5.4.2. Visualización

The screenshot shows a data visualization interface for San Francisco. At the top, there's a header with the name 'San Francisco' and various icons. Below that, a Hive query is displayed: `%hive select * from dim_fechas`. The query has been executed, as indicated by the 'FINISHED' status. Below the query, a table of data is shown with the following columns: `dim_fechas.date`, `dim_fechas.year`, `dim_fechas.month`, `dim_fechas.day`, `dim_fechas.dayofweek`, and `dim_fechas.id`. The data rows represent dates from 2000/01/01 to 2000/01/06, with corresponding year, month, day, day of the week, and a unique ID.

dim_fechas.date	dim_fechas.year	dim_fechas.month	dim_fechas.day	dim_fechas.dayofweek	dim_fechas.id
2000/01/01	2000	1	1	Saturday	0
2000/01/02	2000	1	2	Sunday	1
2000/01/03	2000	1	3	Monday	2
2000/01/04	2000	1	4	Tuesday	3
2000/01/05	2000	1	5	Wednesday	4
2000/01/06	2000	1	6	Thursday	5

Figura 116. Datos dimension Fechas – Alejandro Reina Reina

5.5.5. Diseño de dimensiones y tabla de hechos

Por simplicidad, hare referencia a algunas de las dimensiones del Datamart “*fire_department_calls_for_service*”, permitiéndonos entender el concepto y el desarrollo de las dimensiones y a la tabla de hechos de este Datamart, pero se aplicará por igual los mismos conceptos para el resto de dimensiones y tablas de hechos definidas anteriormente.

5.5.5.1. Diseño de la dimension Location

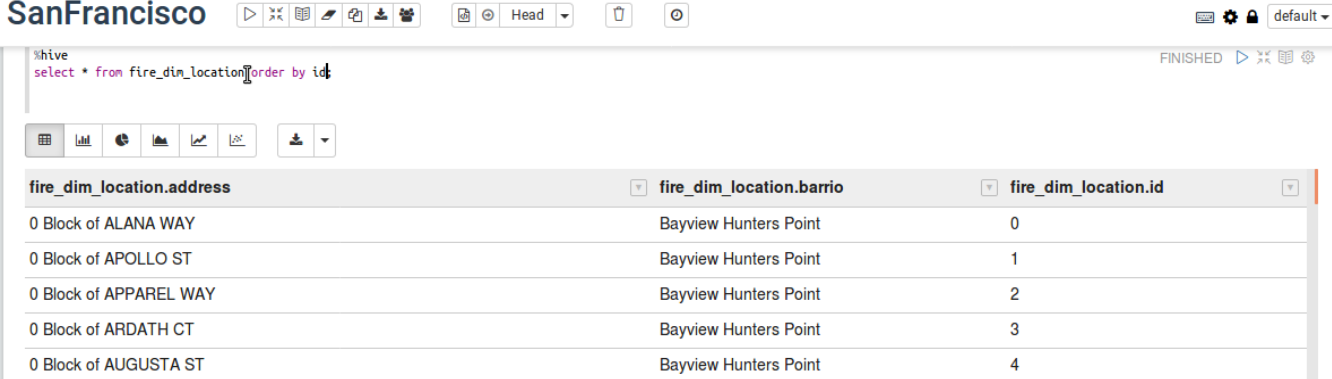
Con este método, leeremos del fichero los datos que nos interesan para la creación de la dimension que pertenece a la localización donde se ha pedido la asistencia o ha ocurrido el incidente, teniendo aproximadamente 30.000 tuplas.

5.5.5.1.1. Código

```
def dimension_Localizacion(data: DataFrame)//:DataFrame = df
{
  var dim = sqlContext.emptyDataFrame
  //Seleccionamos las columnas de nuestro interes
  dim = data.select("Address", "Neighborhoods - Analysis Boundaries")
    //eliminamos duplicados renombramos campos, y ordenamos
    .distinct()
    .withColumnRenamed("Neighborhoods - Analysis Boundaries",
"barrio")
    .orderBy("barrio", "Address")
  //Añadimos el id
    .withColumn("id",monotonically_increasing_id())
```

```
//Creacion de tabla Hive
create_hive_table(dim,"FIRE_DIM_LOCATION")
}
```

5.5.5.1.2. Visualización



The screenshot shows a data visualization interface for San Francisco. The query executed is `select * from fire_dim_location order by id`. The resulting table is as follows:

fire_dim_location.address	fire_dim_location.barrio	fire_dim_location.id
0 Block of ALANA WAY	Bayview Hunters Point	0
0 Block of APOLLO ST	Bayview Hunters Point	1
0 Block of APPAREL WAY	Bayview Hunters Point	2
0 Block of ARDATH CT	Bayview Hunters Point	3
0 Block of AUGUSTA ST	Bayview Hunters Point	4

Figura 117. Datos dimension `fire_dim_location` – Alejandro Reina Reina

5.5.5.2. Diseño de la tabla de hechos

Este es el método más complejo, ya que debemos de realizar una serie de operaciones para cada una de las dimensiones y de manera ordenada, dado el campo del fichero que defina nuestra dimension, haremos un lookup en la tabla Hive de esa dimension y devolveremos el id de dicha tabla, que será nuestra clave primaria y clave ajena de la tabla de hechos con dicha dimension. Y además calcularemos las medidas para cada una de las filas de nuestra tabla de hechos.

5.5.5.2.1. Código

```
//Tabla de Hechos
def fire_factTable(data:DataFrame)
{
  //Limpiamos campos que no vamos a utilizar para liberar espacio
  var factTable = data.drop("Neighborhoods - Analysis Boundaries")
                      .drop("UnitType")
                      .drop("Call Type Group")
                      .drop("Unit Type")
  val hc = new org.apache.spark.sql.hive.HiveContext(sc)
  //Dimension Localizacion
  //Cargamos datos de la dimension
  var dimension_localizacion = hc.sql("select id, Address from
fire_dim_location")
  //Relacionamos tabla de hechos con la dimension
  factTable = factTable.join(dimension_localizacion,
factTable.col("Address") === dimension_localizacion.col("Address"))
  //Limpiamos y renombramos campos
  factTable = factTable.drop("Address")
                      .withColumnRenamed("id", "idAddress")

  //Dimension Unidad
  //Cargamos Datos
```

```

var dimension_unidad = hc.sql("select id, unitID from fire_dim_unit")
//Relacionamos tabla de hechos con la dimension
factTable = factTable.join(dimension_unidad, factTable.col("Unit ID") ===
dimension_unidad.col("UnitID"))
//Limpiamos y renombramos campos
factTable = factTable.drop("Unit ID")
                        .drop("UnitID")
                        .withColumnRenamed("id", "idLocation")
//Dimension tipo de llamada
//Cargamos datos
var dimension_callType = hc.sql("select id, callType from
fire_dim_calltype")
//Relacionamos tabla de hechos con la dimension
factTable = factTable.join(dimension_callType, factTable.col("Call Type")
=== dimension_callType("callType"))
//Limpiamos y renombramos campos
factTable = factTable.drop("Call Type")
                        .drop("callType")
                        .withColumnRenamed("id", "idCallType")
//Dimension Disposicion final de la llamada
//Cargamos datos
var dimension_callFinalDisposition = hc.sql("select id,
callFinalDisposition from fire_dim_callfinaldisposition")
//Relacionamos tabla de hechos con la dimension
factTable = factTable.join(dimension_callFinalDisposition,
factTable.col("Call Final Disposition") ===
dimension_callFinalDisposition("callFinalDisposition"))
//Limpiamos y renombramos campos
factTable = factTable.drop("Call Final Disposition")
                        .drop("callFinalDisposition")
                        .withColumnRenamed("id", "callFinalDisposition")
//Dimension Battalion
//Cargamos datos
var dimension_battalion = hc.sql("select id, battalion from
fire_dim_battalion")
//Relacionamos tabla de hechos con la dimension
factTable = factTable.join(dimension_battalion, factTable.col("Battalion")
=== dimension_battalion("battalion"))
//Limpiamos y renombramos campos
factTable = factTable.drop("Battalion")
                        .drop("battalion")
                        .withColumnRenamed("id", "idBattalion")
//Dimension Fecha
//Cargamos datos
var dimension_fecha = hc.sql("select id,date from dim_fechas")
//Relacionamos tabla de hechos con la dimension
factTable = factTable.join(dimension_fecha,
unix_timestamp(factTable.col("Call Date"),"dd/MM/yyyy") ===
unix_timestamp(dimension_fecha.col("date"),"yyyy/MM/dd"))
//Limpiamos y renombramos campos
factTable = factTable.withColumnRenamed("id","idFecha")

```



```

        .drop("Date")
        .drop("date")

//Medidas
//Tiempo entre que recibe la llamada y acude a la escena en segundos
factTable = factTable.withColumn("tiempoRespuesta",
    //Omitir calculo cuando no hayan acudido a la escena
    when(factTable.col("On Scene DtTm").isNotNull,
        unix_timestamp(factTable("On Scene DtTm"),
"MM/dd/yyyy hh:mm:ss aa") -
        unix_timestamp(factTable("Received DtTm"),
"MM/dd/yyyy hh:mm:ss aa")))
    //Tiempo en que la unidad vuelve a estar disponible en segundos
    .withColumn("tiempoDisponible",
        unix_timestamp(factTable("Available DtTm"),
"MM/dd/yyyy hh:mm:ss aa") -
        unix_timestamp(factTable("Dispatch DtTm"),
"MM/dd/yyyy hh:mm:ss aa"))
//Limpieza de campos
factTable = factTable.drop("Received DtTm")
        .drop("On Scene DtTm")
        .drop("Available DtTm")
        .drop("Dispatch DtTm")
        .drop("Response DtTm")
        .drop("Call Date")

//Crear Tabla Hive
create_hive_table(factTable,"FIRE_FACTTABLE")
}

```

5.5.5.2.2. Visualización

Incident Number	Location	idAddress	idUnit	idCallType	callFinalDisposition	idBattalion	idFecha	tiempoRespuesta	tiempoDisponible
31637	(37.7773216006661...	609885356117	1159641169921	146028888064	103079215104	42949672960	105	813	2031
31637	(37.7773216006661...	609885356117	1159641169921	146028888065	103079215104	42949672960	105	813	2031
31637	(37.7773216006661...	609885356117	1159641169921	146028888066	103079215104	42949672960	105	813	2031
31637	(37.7773216006661...	609885356117	1159641169921	146028888067	103079215104	42949672960	105	813	2031
31649	(37.7706078154142...	438086664245	687194767361	34359738368	94489280512	42949672960	105	647	902
31649	(37.7706078154142...	438086664245	687194767361	34359738369	94489280512	42949672960	105	647	902
31542	(37.7775342405103...	1692217114669	687194767361	34359738368	94489280512	42949672960	105	339	469
31542	(37.7775342405103...	1692217114669	687194767361	34359738369	94489280512	42949672960	105	339	469
31404	(37.7742331365027...	472446402666	652835028992	206158430208	94489280512	42949672960	105	295	249
31404	(37.7742331365027...	472446402666	652835028992	206158430209	94489280512	42949672960	105	295	249

Figura 118. Datos dimension fire_facttable – Alejandro Reina Reina

5.6. Diseño del cubo MOLAP

Una vez montado los ETLs, donde hemos volcado los datos en nuestra tabla Hive, es momento de diseñar el cubo, o lo que es lo mismo nuestro servidor de consulta. Kylin lo que va a hacer es sacrificar espacio en pro de rendimiento. En Kylin crearemos dicho cubo multidimensional el cual será contra el que hagamos las consultas. Kylin, según como optimizemos el cubo, montará todas las combinaciones posibles entre dimensiones y tabla de hechos, con el objetivo de que cuando una consulta se realice sobre Kylin, no tenga que ser calculada sobre una BD,

directamente devolverá el resultado, es decir, Kylin nos precalculará las consultas y las almacenara, como resultado, cualquier consulta que le hagamos, será devuelta.

Por simplicidad, solo hare referencia a la creación del cubo de uno de los datamarts, pero montar el cubo sobre el otro datamarts, será exactamente igual, bajo los mismos conceptos, pero seleccionando sus dimensiones y tablas propias como se especificaron en otras fases del diseño.

El diseño del cubo Kylin estará compuesto de tres partes fundamentalmente, carga de datos de Hive en Kylin, diseño en Kylin del modelo multidimensional y diseño del cubo MOLAP. Cabe destacar, que el manual en la página oficial del diseño del modelo y el cubo no está actualizado del todo, han incluido algunos cambios por lo que el diseño del cubo en Kylin que se detalla a continuación corresponde a la versión 2.2.0.

5.6.1. Carga de las tablas de Hive

El primer paso para esto, es acceder a Kylin, y una vez estamos en Kylin crearemos un nuevo proyecto, haciendo clic en el icono de “+” y tras rellenar los datos haremos clic en “submit” como vemos en la siguiente imagen:

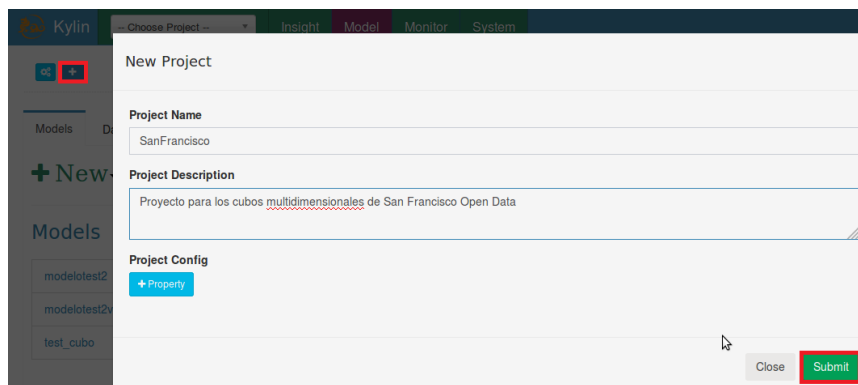


Figura 119. Nuevo proyecto Kylin – Alejandro Reina Reina

Una vez creado el proyecto, haremos clic en Datasources, para cargar las tablas Hive creadas previamente en la fase de diseño de ETL.

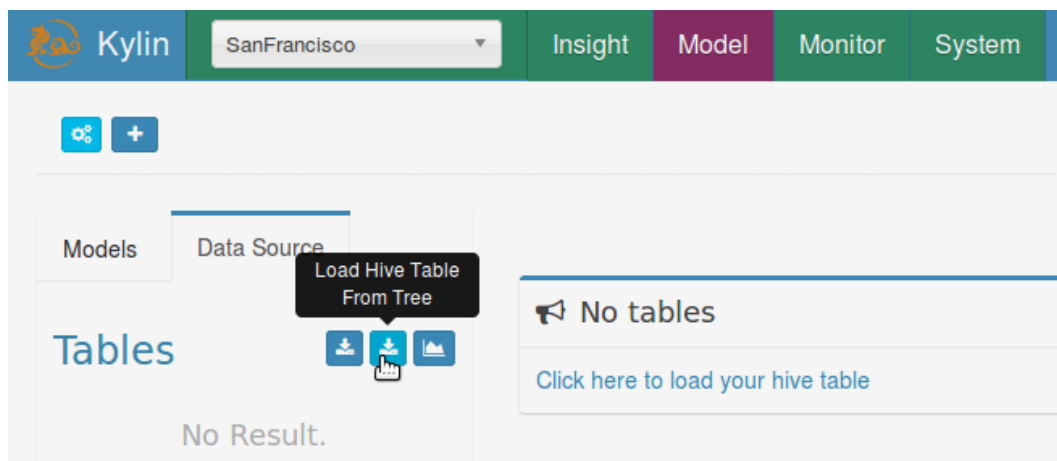


Figura 120. Cargar Tablas Hive – Alejandro Reina Reina

Una vez hagamos clic, en “Load Hive Table From Tree”, leerá todas las tablas que tenemos en Hive, seleccionaremos las tabla que nos interesen, en nuestro caso, vamos a seleccionar todas las pertenecientes al Datamart “police_department” como vemos en la imagen.

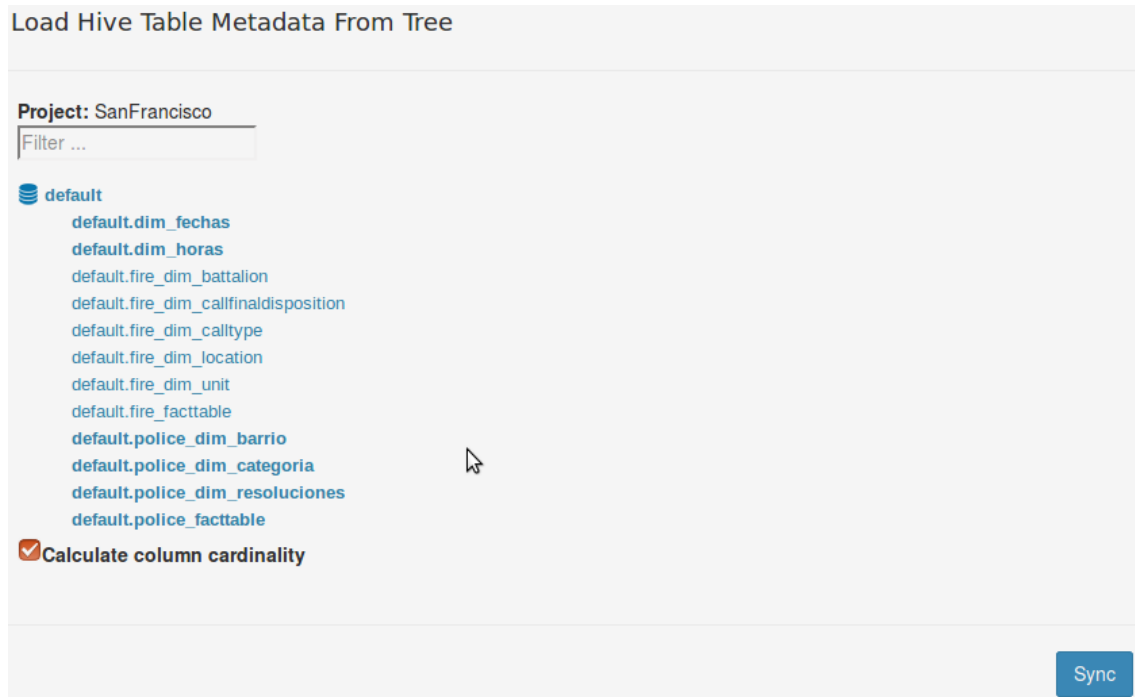


Figura 121. Seleccionar Tablas Hive – Alejandro Reina Reina

Si todo ha ido correctamente y tenemos bien integrado Kylin con Hive, nos mostrara la siguiente ventana, en caso contrario nos indicara un mensaje de error, pero de estos mensajes de error hay poca o ninguna información, si falla en este punto se debe a un error en la integración de Kylin con Hive a pesar que en el arranque no haya saltado ningún fallo, por lo que los pasos descritos en el diseño de la arquitectura para integrar Kylin son muy importantes.

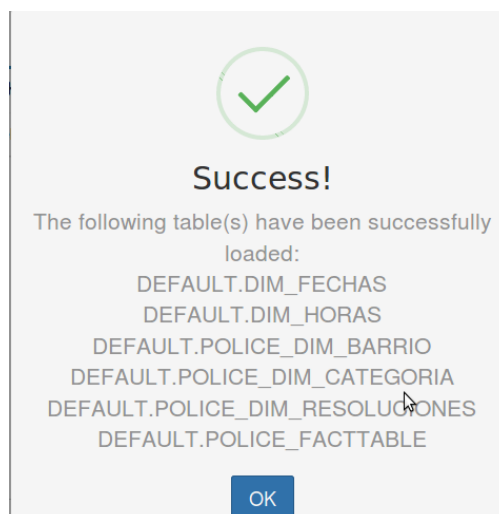


Figura 122. Tablas cargadas correctamente en Kylin – Alejandro Reina Reina

Una vez cargadas, podemos consultar los metadatos de las tablas, en la propia interfaz de Kylin, recargarlas si han sido modificadas o borrarlas del repositorio de Kylin para que no sean cargadas nuevamente.

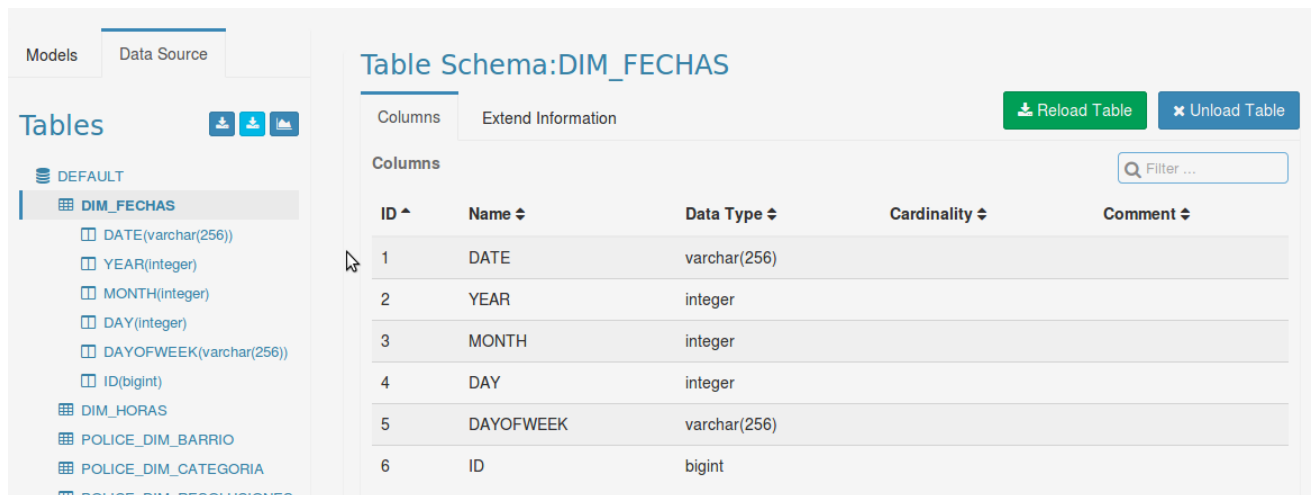


Figura 123. Tablas cargadas correctamente en Kylin – Alejandro Reina Reina

5.6.2. Diseño del modelo Multidimensional en Kylin

El primer paso para poder crear nuestro cubo, es diseñar el modelo de nuestro cubo, para ello, haremos clic en New Model y añadiremos un nuevo modelo como vemos en la siguiente imagen.

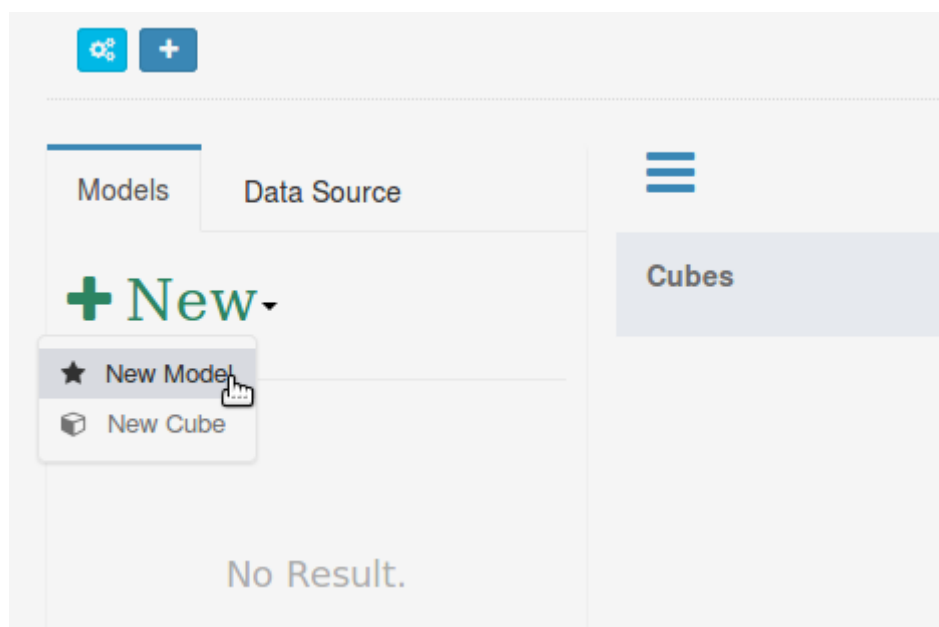


Figura 124. New Model Kylin – Alejandro Reina Reina

Una vez creamos el modelo, deberemos de rellenar una serie de campos para poder crear nuestro modelo, básicamente lo que hacemos a través de esta interfaz es crear un fichero en formato JSON y SQL el cual Kylin leerá para consultar dicho cubo, los campos y la finalidad de dichos campos los iré detallando a continuación:

En primer lugar, insertaremos los datos de nuestro modelo a crear, nombre y descripción del modelo, como vemos en la siguiente imagen:

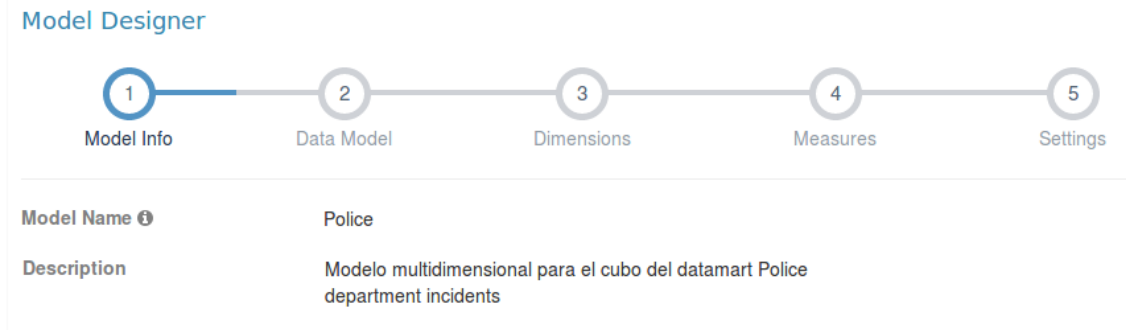


Figura 125. Model Info Kylin – Alejandro Reina Reina

En el siguiente paso, le diremos a Kylin cuál es nuestra tabla de hechos, y cuáles son nuestras dimensiones o lookup tables, y las columnas por las cuales estas tablas se relacionan.

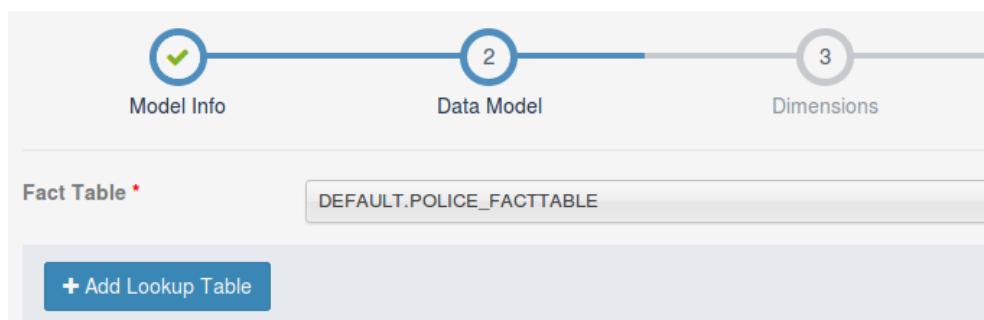


Figura 126. Añadir Tabla de hechos y lookup tables – Alejandro Reina Reina

Una vez seleccionada nuestra tabla de hechos, seleccionaremos “Add Lookup Table” para añadir nuestras dimensiones, al hacer clic, veremos una ventana similar a la siguiente donde rellenaremos que tablas relacionamos y porque campo se relacionan:

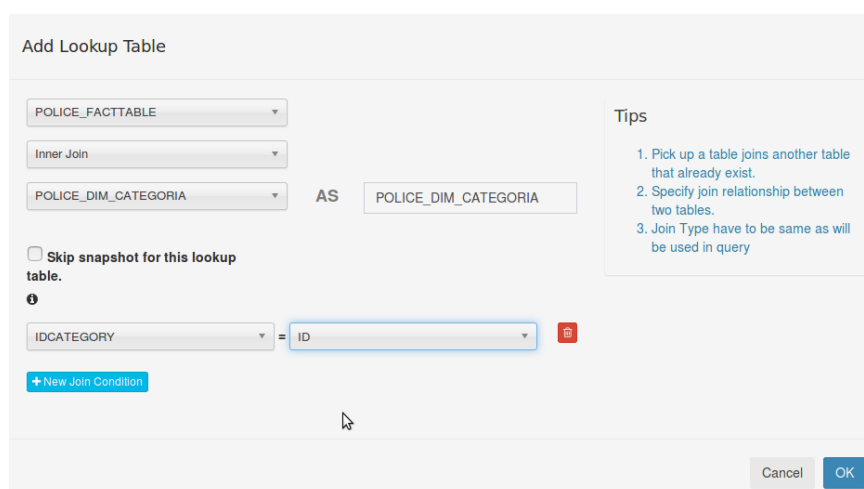


Figura 127. Añadir lookup tables – Alejandro Reina Reina

Quedando finalmente este paso como en la siguiente imagen:

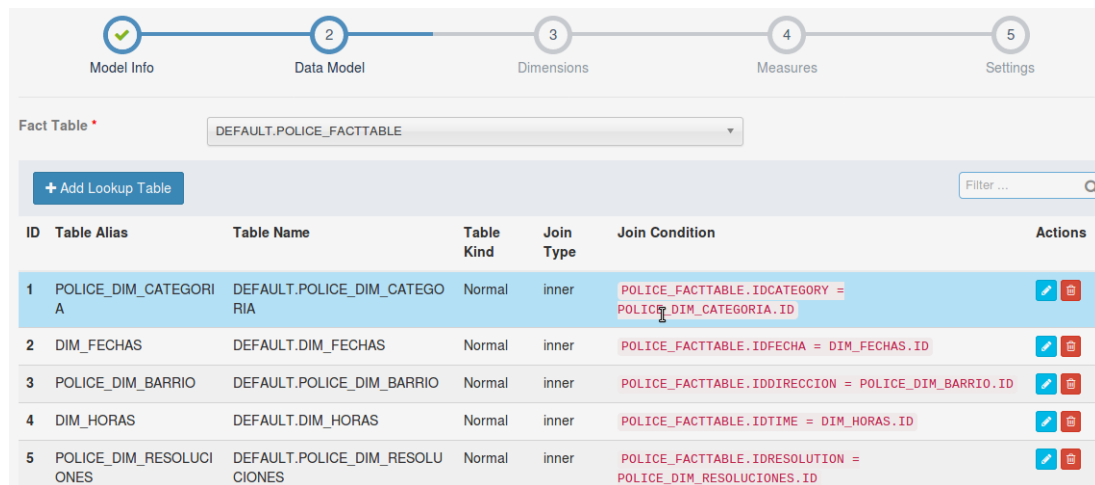


Figura 128. Paso Data Model – Alejandro Reina Reina

Una vez tenemos las tablas relacionadas en nuestro modelo de datos, le indicaremos a Kylin cuáles son las columnas que forman nuestra dimensión, es decir, los campos de las jerarquías de nuestra dimensión, como vemos a continuación:

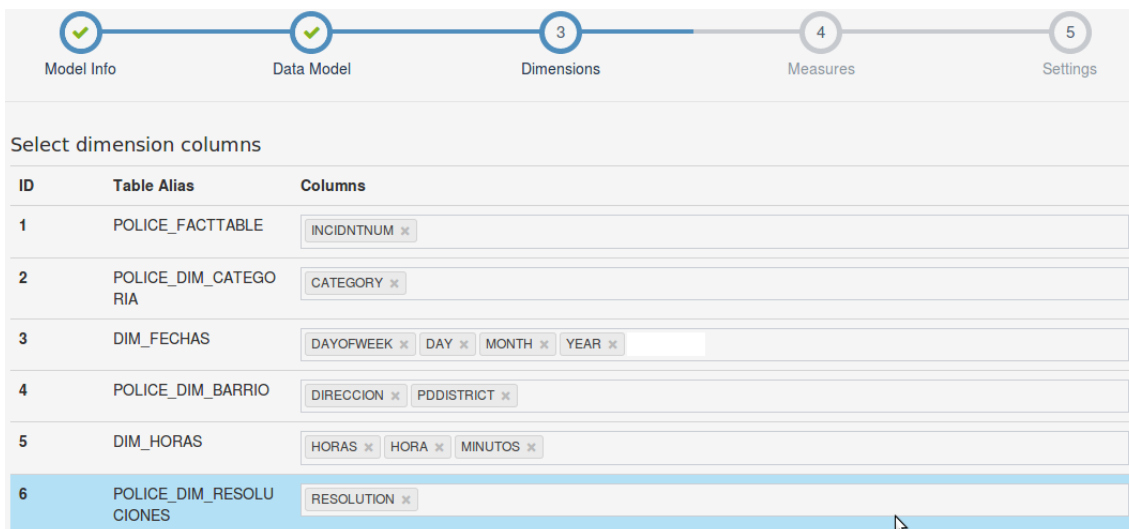


Figura 129. Seleccionamos columnas de dimension – Alejandro Reina Reina

El siguiente paso, es seleccionar los campos que son nuestras medidas en la tabla de hechos como vemos en la siguiente imagen:

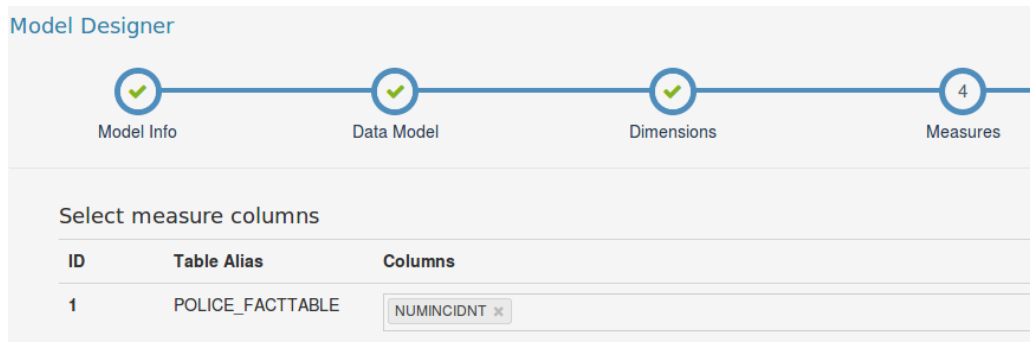


Figura 130. Seleccionar medidas – Alejandro Reina Reina

El último paso, es para particionar el cubo, donde le indicaremos cual es el campo que indica o aumenta nuestra tabla de hechos, generalmente será con respecto al tiempo, es decir si queremos particionar la tabla de hechos por meses o años, seleccionaremos la columna fecha correspondiente seleccionando su formato, y filtrando por la fecha de los campos o estudio que queramos realizar, pero en nuestro caso, analizaremos todos los datos al completo por lo que lo dejaremos en blanco este paso.

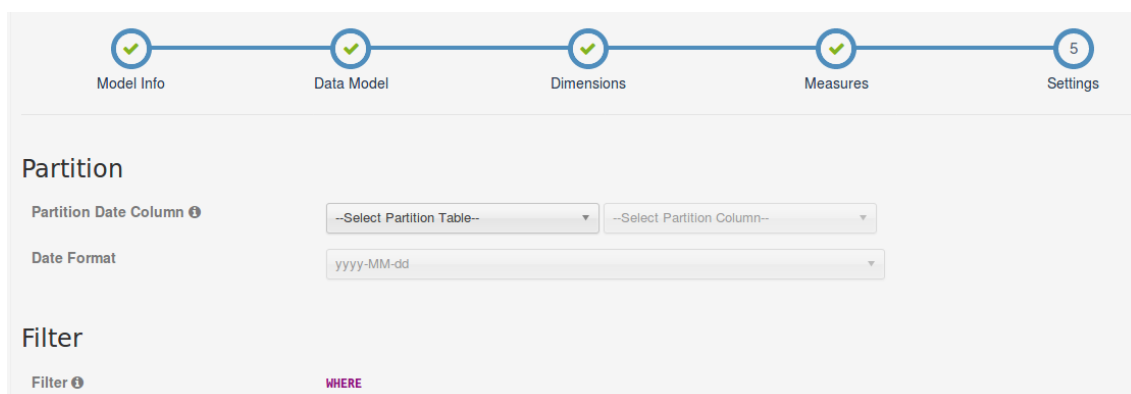


Figura 131. Partición de tabla – Alejandro Reina Reina

Si hemos configurado bien cada uno de los pasos, al final haremos clic en save cube, y tras confirmar esta acción, nos mostrara una ventana como la siguiente. Nuevamente no encontraremos mucha información sobre los errores producidos en este punto, pero si tenemos algún fallo en este punto, hay altas probabilidades de que se deba a un mal diseño en los pasos anteriores o algún campo incorrecto:

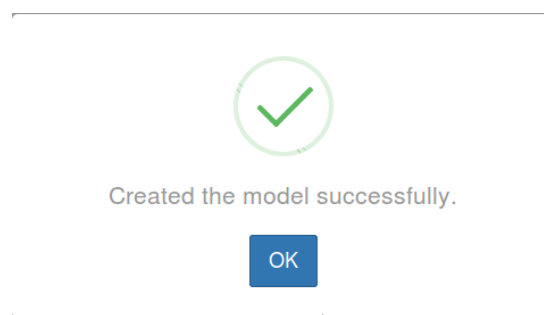


Figura 132. Modelo creado correctamente – Alejandro Reina Reina

Una vez creado el modelo, podemos editarlo o ver algunos datos que puedan ser de nuestro interés:

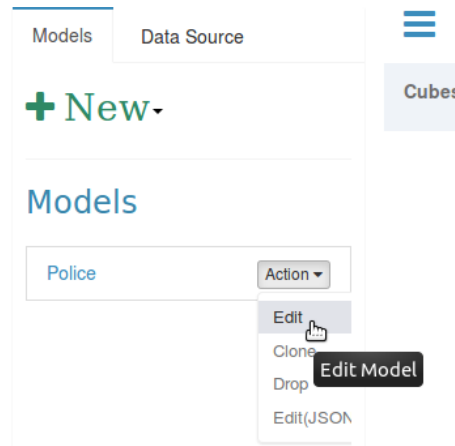


Figura 133. Editar Modelo – Alejandro Reina Reina

Si hacemos clic en el nombre, podemos ver el código en formato JSON del modelo o las relaciones de nuestro modelo multidimensional como vemos en la siguiente imagen:



Figura 134. Visualización del modelo – Alejandro Reina Reina

5.6.3. Diseño del cubo en Kylin

Esta fase del diseño del cubo es la más delicada de las fases del diseño del cubo, ya que debemos de tener cuidado a la hora de crearlo y optimizarlo, como explicare más adelante.

Para crear el cubo, haremos clic en new cube como vemos en la siguiente imagen:

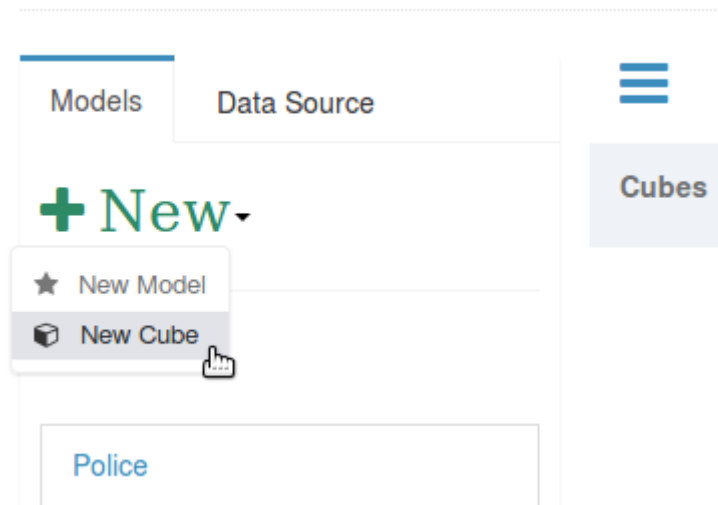


Figura 135. Nuevo Cubo – Alejandro Reina Reina

La creación del cubo también tiene una serie de pasos para su creación que detallare uno a uno.

En el primer punto, debemos seleccionar el modelo en el cual se basará nuestro cubo, nombre y descripción del mismo, y adicionalmente podremos configurar para que nos notifique vía Email si ha ocurrido alguno de los eventos que le indiquemos como fallo en la construcción, o construcción correcta entre otros, aunque esto último no lo configuraremos, en la siguiente imagen podemos ver cómo queda este primer paso:

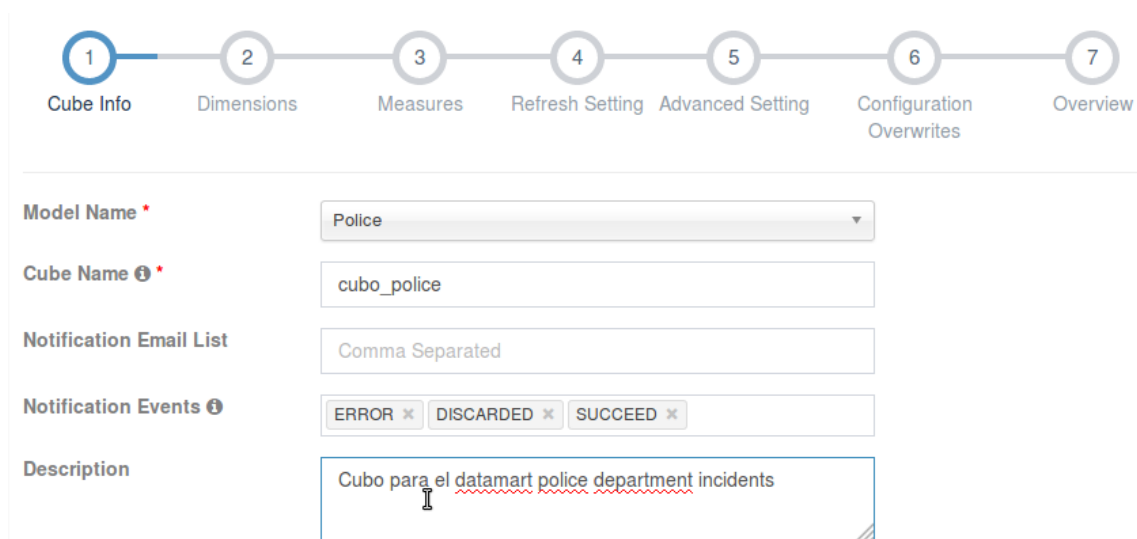


Figura 136. Info del cubo – Alejandro Reina Reina

En el siguiente paso, indicaremos si son dimensiones normales o dimensiones derivadas de otros campos, por defecto, en la tabla de hechos estas dimensiones solo podrán ser dimensiones normales, pero en las lookup tables si pueden ser derivadas, vemos en las siguientes imágenes como añadir las dimensiones, , en nuestro caso todas serán normales y para ello primero haremos clic en “Add dimensions” y a continuación, seleccionaremos los campos que conforman nuestras jerarquías, indicándole si es normal o derivada, en la página oficial de Kylin, podemos

encontrar más información sobre dimensiones normales y derivadas y como optimizar la construcción de cubos. (73)

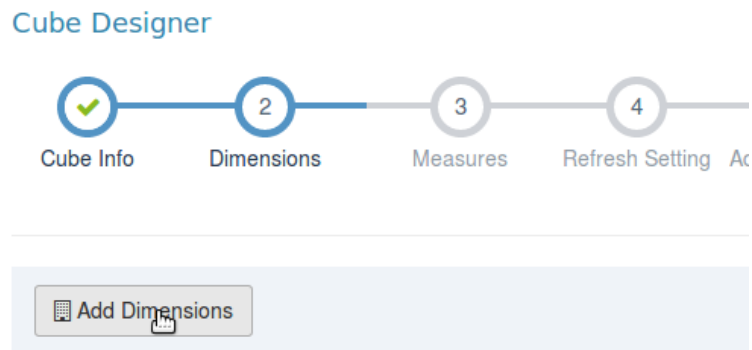


Figura 137. Add Dimensions – Alejandro Reina Reina

Una vez le demos a añadir dimensiones, seleccionaremos aquellos campos que sean nuestras dimensiones, en la siguiente imagen vemos la tabla de hechos, y como no podemos indicarle si es normal o derivada.

Auto Generate Dimensions ⓘ This is a helper for you to batch generate dimensions.
 Visit [derived column](#) for more about derived column.

POLICE_FACTTABLE [FactTable]

<input type="checkbox"/> Select All	Name	Columns
<input checked="" type="checkbox"/>	INCIDENTNUM	INCIDENTNUM
<input type="checkbox"/>	IDCATEGORY	IDCATEGORY

Figura 138. Add dimensions en la tabla de hechos – Alejandro Reina Reina

En la siguiente imagen vemos la selección de las dimensiones de las otras tablas.

Auto Generate Dimensions ⓘ This is a helper for you to batch generate dimensions.
 Visit [derived column](#) for more about derived column.

<input checked="" type="checkbox"/>	RESOLUTION	RESOLUTION	<input checked="" type="radio"/> Normal	<input type="radio"/> Derived
<input type="checkbox"/>	ID	ID	<input type="radio"/> Normal	<input type="radio"/> Derived

POLICE_DIM_BARRIO [LookupTable]

<input type="checkbox"/> Select All	Name	Columns		
<input checked="" type="checkbox"/>	DIRECCION	DIRECCION	<input checked="" type="radio"/> Normal	<input type="radio"/> Derived
<input checked="" type="checkbox"/>	PDDISTRICT	PDDISTRICT	<input checked="" type="radio"/> Normal	<input type="radio"/> Derived
<input type="checkbox"/>	ID	ID	<input type="radio"/> Normal	<input type="radio"/> Derived

Figura 139. Add dimensions en lookup tables – Alejandro Reina Reina

Finalmente, este paso nos quedara algo similar a la siguiente imagen:

ID	Name	Table Alias	Type	Column
1	CATEGORY	POLICE_DIM_CATEGORIA	normal	CATEGORY
2	YEAR	DIM_FECHAS	normal	YEAR
3	MONTH	DIM_FECHAS	normal	MONTH
4	DAY	DIM_FECHAS	normal	DAY
5	DAYOFWEEK	DIM_FECHAS	normal	DAYOFWEEK
6	RESOLUTION	POLICE_DIM_RESOLUCIONES	normal	RESOLUTION
7	DIRECCION	POLICE_DIM_BARRIO	normal	DIRECCION
8	PDDISTRICT	POLICE_DIM_BARRIO	normal	PDDISTRICT
9	HORA	DIM_HORAS	normal	HORA
10	MINUTOS	DIM_HORAS	normal	MINUTOS

Figura 140. Add dimensions resumen – Alejandro Reina Reina

En el siguiente paso le indicaremos las medidas, por defecto, siempre aparecerá la medida count que nos mostrara el número de filas que tenemos, pero de manera adicional, podemos añadir nuestras medidas y la manera de tratar esta medida de manera agregada, es decir, cuando queramos visualizar esta medida para un conjunto de datos, por ejemplo el número de incidentes será un count de las tuplas, mientras que el tiempo de respuesta será la media(AVG) de dicha medida. Aclarado este punto, veremos cómo añadir estas medidas en el cubo:

Name	Expression	Parameters	Return Type	Actions
COUNT	COUNT	Value:1, Type:constant	bigint	

Figura 141. Añadir medidas al cubo – Alejandro Reina Reina

En el siguiente paso, podemos configurar el cubo para que cargue los datos de manera incremental, es decir, una vez creamos un cubo, dicho cubo solo dispone los datos que hasta el momento estaban cargados en las tablas Hive en el momento de generar dicho cubo, si queremos refrescar este cubo con los últimos datos, deberíamos de volver a crear un cubo. Con esta opción evitamos volver a crear dicho cubo, creando un job que cuando tenga X datos los cargue automáticamente, por ejemplo, podríamos indicarle que cargue automáticamente los datos del último día, que estarán en nuestra tabla Hive.

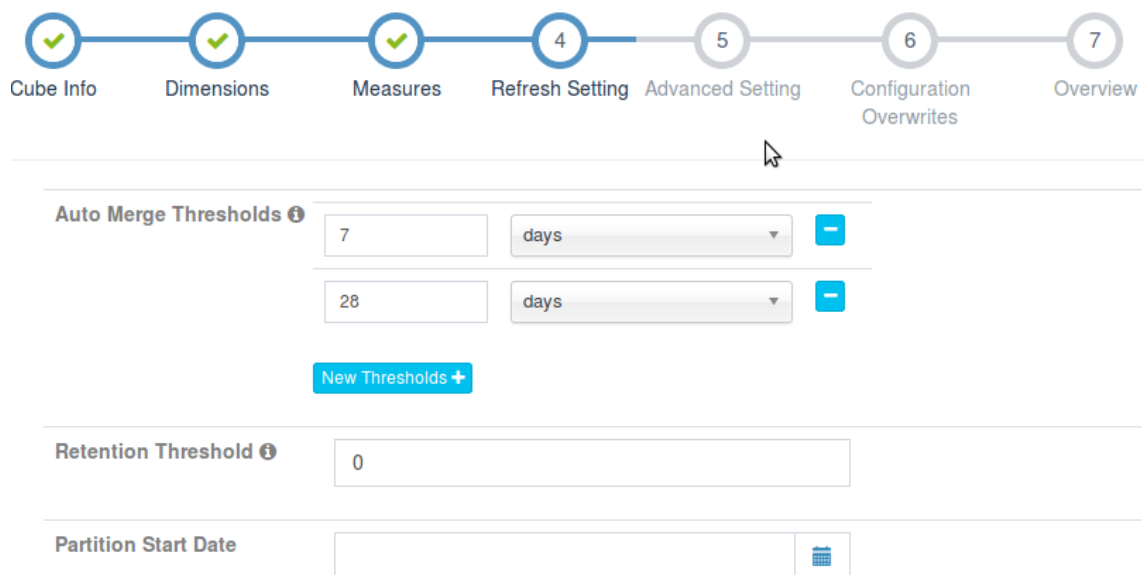


Figura 142. Cubo incremental – Alejandro Reina Reina

El siguiente paso es un paso muy importante en cuanto a optimización del cubo se refiere y el más delicado a la hora de poder compilar el cubo. En este step, se introducen una serie de reglas para la creación del cubo. Debemos de tener en cuenta que Kylin genera todas las posibles combinaciones entre las dimensiones, es decir si tenemos 20 dimensiones, tendríamos 2^{20} combinaciones (aproximadamente 1 millón de combinaciones distintas), multiplicado por la cantidad de datos, es algo a tener en cuenta y con lo que debemos de llevar cuidado. Para la optimización del cubo en este paso tenemos tres conceptos a destacar de las dimensiones o, mejor dicho, reglas que se aplicaran en la construcción del cubo.:

- **Mandatory dimensions:** Esta regla es establecida con la intención de eliminar combinaciones, pretende hacer una “poda combinada”, por ejemplo, si una dimension se especifica como “Mandatory” u obligada, entonces todas las combinaciones sin dicha dimension se eliminarán en la construcción del cubo.
- **Hierarchy dimensions:** Esta regla nos permite crear jerarquías en una dimension, por ejemplo, si establecemos que A, B y C tienen una relación de jerarquía, entonces solo se mantendrán las combinaciones con A, AB o ABC.
- **Joins dimension:** Esta es una última regla introducida, y establece que, si dos o más dimensiones son conjuntas, entonces cualquier cuboide valido no contendrá ninguna de estas dimensiones o las contendrá todas. En otras palabras, estas dimensiones siempre

estarán juntas. Esto es útil para cuando el cubo que queremos diseñar, sabemos que algunas de las dimensiones siempre se consultaran juntas. También es muy potente si sabemos que algunas de las dimensiones serán menos probables de usar, por ejemplo, si tenemos 20 dimensiones, como en el ejemplo comentado anteriormente, y sabemos que 10 de esas dimensiones son menos consultadas, podemos “unirlas” o establecer la regla de joins, reduciremos el número de cuboides o combinaciones a 2^{11} . En versiones anteriores se conocía como “grupo de agregación”.

Con estas tres reglas podremos calcular y optimizar la construcción del cubo y la generación de combinaciones que se crearan, reduciendo significativamente los gastos generales de computación y almacenamiento, especialmente cuando el cubo sirve para un Dashboard concreto, que reproducirá una serie de consultas SQL que solo requieren algunos cuboides específicos.

En nuestro caso, debido a las limitaciones de almacenamiento, se intentará reducir considerablemente el número de combinaciones, por lo que las jerarquías dentro de una dimension se agregaran, quedando de la siguiente manera:

The screenshot shows a configuration interface for a data cube. It is divided into several sections:

- Includes:** A list of dimension columns: POLICE_DIM_CATEGORIA.CATEGORY, DIM_FECHAS.YEAR, DIM_FECHAS.MONTH, DIM_FECHAS.DAY, DIM_FECHAS.DAYOFWEEK, POLICE_DIM_RESOLUCIONES.RESOLUTION, POLICE_DIM_BARRIO.DIRECCION, POLICE_DIM_BARRIO.PDDISTRICT, DIM_HORAS.HORA, and DIM_HORAS.MINUTOS.
- Mandatory Dimensions:** A dropdown menu with the text "Select Column...".
- Hierarchy Dimensions:** A button labeled "New Hierarchy+".
- Joint Dimensions:** Three groups of dimension columns, each with a minus sign button:
 - Group 1: DIM_FECHAS.YEAR, DIM_FECHAS.MONTH, DIM_FECHAS.DAY, DIM_FECHAS.DAYOFWEEK.
 - Group 2: POLICE_DIM_BARRIO.PDDISTRICT, POLICE_DIM_BARRIO.DIRECCION.
 - Group 3: DIM_HORAS.HORA, DIM_HORAS.MINUTOS.
- Bottom:** A button labeled "New Joint+".

Figura 143. Grupos de agregación de las dimensiones del cubo – Alejandro Reina Reina

De manera adicional, en este paso, aun podríamos optimizar más la construcción del cubo añadiendo diccionarios a las columnas que conforman nuestras dimensiones, como vemos en la siguiente imagen, aunque el diseño de diccionarios requeriría un estudio más profundo tanto de Kylín como de optimización y no lo tratare en este proyecto.

Rowkeys ?

ID	Column	Encoding	Length	Shard By
1	POLICE_FACTTABLE.IN...	dict	Coli	false by default
2	POLICE_DIM_CATEGO ...	dict	Coli	false by default
3	DIM_FECHAS.DATE	dict	Coli	false by default
4	DIM_FECHAS.YEAR	dict	Coli	false by default
5	DIM_FECHAS.MONTH	dict	Coli	false by default
6	DIM_FECHAS.DAY	dict	Coli	false by default
7	DIM_FECHAS.DAYOFW ...	dict	Coli	false by default

Figura 144. Grupos de agregación de las dimensiones del cubo – Alejandro Reina Reina

Por último, en este último paso, podremos indicar el motor con el cual se van a realizar los cálculos (MapReduce o AGC y RDD de Spark), y diccionarios avanzados, aunque no entraremos en este punto y lo dejaremos por defecto, pero con Spark mejoraríamos el tiempo de computo a la hora de crear el cubo, en el caso particular de este proyecto, no pudo realizarse ya que requiere de otro servicio, YARN(negociador de recursos) (74).

Cube Engine ?

Engine Type :

Advanced Dictionaries ?

Column	Builder Class	Reuse	Actions
--------	---------------	-------	---------

[+ Dictionaries](#)

Advanced ColumnFamily ?

Figura 145. Motor de construcción del cubo y otras opciones – Alejandro Reina Reina

En el siguiente paso, podremos sobrescribir propiedades del cubo de Kylin, en nuestro caso lo dejaremos por defecto, y en el último paso veremos un resumen de nuestro cubo, como vemos en la siguiente imagen:

Model Name	Police
Cube Name	CuboPolice
Fact Table	DEFAULT.POLICE_FACTTABLE
Lookup Table	5
Dimensions	12
Measures	2

Description

Cubo multidimensional para el datamart police_department_incidents

Figura 146. Resumen de construcción del cubo – Alejandro Reina Reina

Si el esquema de nuestro cubo es correcto, veremos el siguiente mensaje.

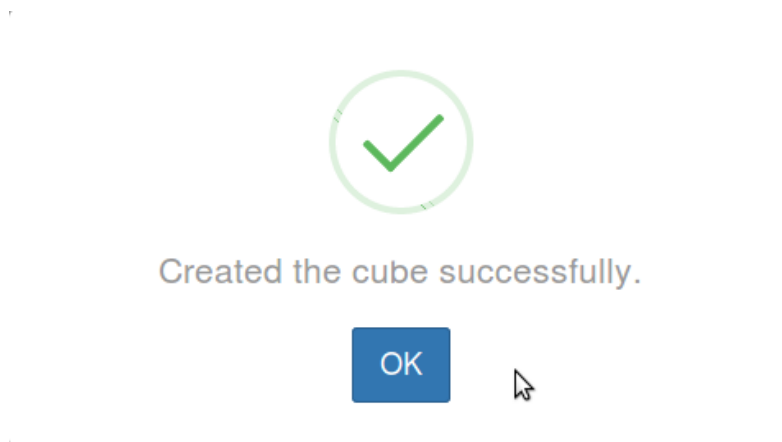


Figura 147. Resumen de construcción del cubo – Alejandro Reina Reina

Pero una vez creado el cubo, podremos verlo en la interfaz que nos provee Kylin, aunque este estará desactivado.

Cubes								
Name	Status	Cube Size	Source Records	Last Build Time	Owner	Create Time	Actions	Admins
CuboPolice	DISABLED	0.00 KB	0		ADMIN	2018-01-22 20:08:32 GMT+8	Action	Action

Figura 148. Visor del cubo – Alejandro Reina Reina

Para activar el cubo y poder consultarlo, primero debemos de compilarlo, en unas pruebas iniciales sobre este mismo Datamart que estoy detallando en el ejemplo de construcción del cubo, en una tabla de hechos de 2000 filas, con 11 dimensiones (en Kylin las jerarquías también son dimensiones) y sin optimización, dicho cubo ocupaba 8,5 GB, por lo que debemos de tener presente lo comentado anteriormente para un buen diseño del cubo y para poder almacenarlo.

5.6.4. Compilar cubo

Una vez creado el modelo multidimensional y el modelo de los cuboides, debemos de compilar el cubo, donde basándose en los modelos anteriormente, conectara con Hive para el motor relacional, y con HBase para almacenar dichas combinaciones físicamente.

Para ello haremos clic en el panel de acciones y en build como vemos a continuación:

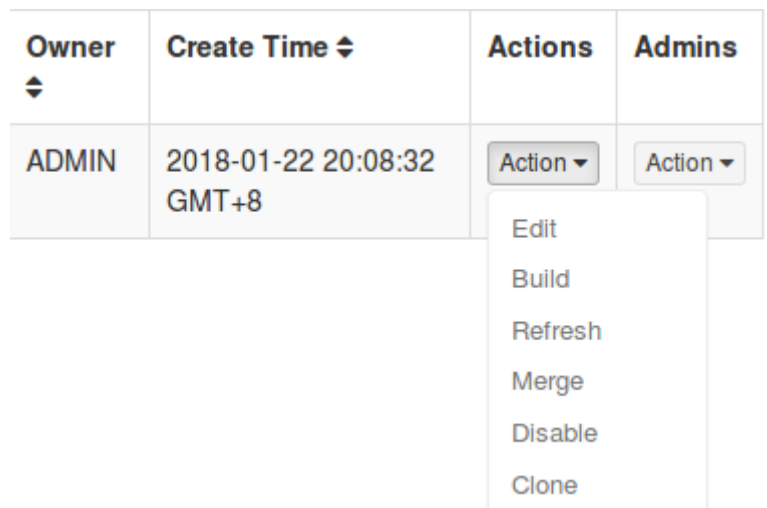


Figura 149. Acciones del cubo– Alejandro Reina Reina

Una vez le demos clic en Build, se iniciará el proceso de compilación, en la pestaña superior de Monitor podremos ver el job ejecutándose, así como el tiempo transcurrido o que ha necesitado para llevar a cabo:

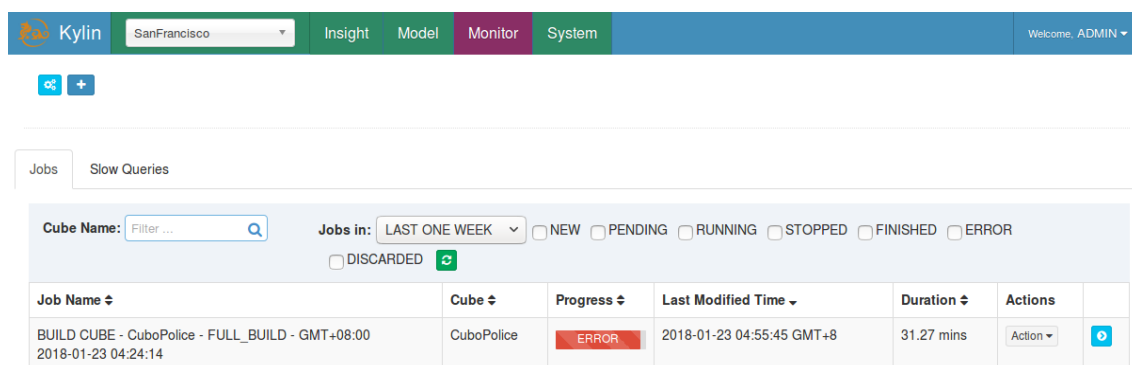


Figura 150. Progreso de compilación del cubo – Alejandro Reina Reina

En mi caso aparece un error, ya que ha fallado en el último step de la compilación, aunque esto no impide que se cree el cubo, el motivo de este error es porque no ha podido eliminar las tablas intermedias de Hive que crea Kylin para apoyarse en la construcción del cubo como vemos a continuación:

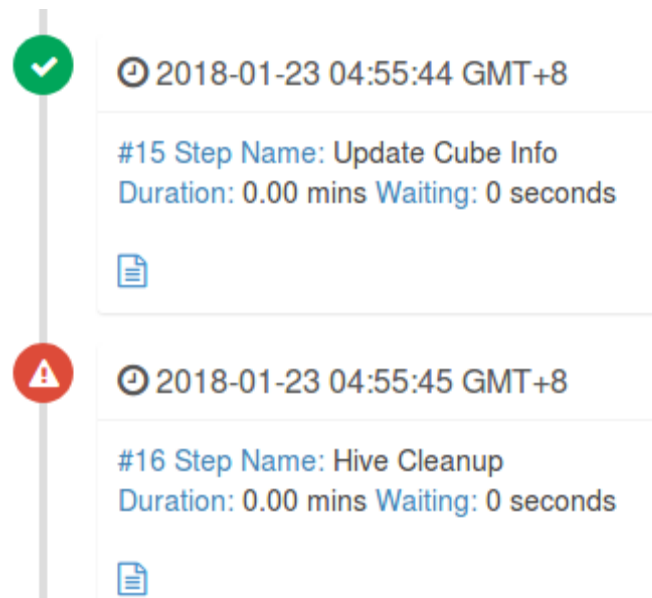


Figura 151. Error en el step Hive Cleanup – Alejandro Reina Reina

Este error, es un bug de Kylin, catalogado como Kylin-2833 – Storage cleanup job (75), que puede ser solucionando ejecutando manualmente el job que comprueba y elimina aquellas tablas que no son necesarias una vez construido el cubo. (76) Aunque no es necesario ejecutar este job, si es recomendable, ya que liberamos espacio de datos que ya no necesitaremos. Una vez compilado, el cubo se activará automáticamente, si no es así, haremos clic en Action->enable para poder consultar el cubo. Cabe destacar también que, si en la construcción del cubo tenemos algún fallo, la información respecto al fallo es prácticamente nula, debemos de consultar a la mailing list de Kylin, o a través del log intentar inferir cual ha sido nuestro error. Si falla el job, deberemos eliminarlo desde la pestaña de Monitor, y luego purgarlo, haciendo clic en Action->Purge, eliminando datos intermedios y permitiéndonos volver a modificar el modelo de construcción del cubo para volver a compilarlo.

6. Visualización

En este apartado vamos a ver como consultar el cubo una vez compilado y habilitado, mediante Kylin y Zeppelin, para poder obtener una visualización de los datos que nos permita obtener información de ellos.

Cabe destacar que esta no es la manera de visualizar la información en un proyecto real, ya que requiere conocimientos en MySQL para ejecutar las sentencias, en proyectos reales y de cara a que los usuarios finales de estos sistemas no disponen de dichos conocimientos informáticos para realizar la consulta, se crea un Dashboard en un servidor web, utilizando herramientas como Tableau, Pentaho, PowerBI, entre otras muchas, aunque independientemente de la que utilicemos no afectara en gran medida al sistema, ya que el sistema ya está montado y funcional para consultas, y estas herramientas lo único que proporcionan es una interfaz drag and drop que traduce a MySQL el cual consultaría el cubo, y permite configurar un tablero o Dashboard con una serie de consultas predefinidas que responden a las preguntas analíticas definidas anteriormente, pero por motivos de tiempo y la gran problemática de integración de Kylin debido a la falta de documentación, es necesario acotar el proyecto en este punto, aunque será una futura ampliación del servidor analítico.

6.1. Consulta del cubo con Kylin

Por tanto, si quisiésemos consultar estas preguntas desde Kylin, desde la pestaña “Insight” podríamos realizar consultas o sentencias SQL directamente, y Kylin automáticamente se apoyará en el cubo para devolver el resultado precalculado, como vemos en la imagen:

Query String » Start Time: 2018-01-28 00:03:13 GMT+8 Duration: 0.50s Rerun Save

```

1 select idcategory, count(*) from police_facttable
2 group by idcategory
3
    
```

Status: Success Project: SanFrancisco **Cubes: CUBE[name=police_cube]**

Results (34) Visualization Export

IDCATEGORY	EXPR\$1
0	28
549755813888	2
558345748480	2
1391569403904	1
300647710720	320
1400159338496	17
326417514496	4

Figura 152. Error en el step Hive Cleanup – Alejandro Reina Reina

Donde podemos observar como utiliza el cubo internamente, lo que aporta un mayor rendimiento en ejecutar dichas consultas, ya que al haber sido precalculadas en el cubo, no es necesario el gasto computacional de realizar dicha consulta, además Kylin proporciona una serie de gráficos predefinidos donde podemos jugar con la visualización y seleccionar un gráfico que nos permita entender y visualizar los datos para obtener información de ellos.

6.2. Consulta del cubo con Zeppelin

También podemos consultar dicho cubo es a través del notebook Zeppelin, el cual también nos provee de un intérprete que conecta con Kylin para acceder a este a través de una apiREST, y realizar las consultas, para ello, deberemos configurar el intérprete de Zeppelin para Kylin, para que conecte con nuestro servidor de consulta, para ello, iremos a intérpretes como explique ya anteriormente, y buscaremos el intérprete de Kylin, quedando la configuración como vemos en la siguiente imagen:

Properties	
name	value
kylin.api.password	KYLIN
kylin.api.url	http://localhost:7070/kylin/api/query
kylin.api.user	ADMIN
kylin.query.ispartial	true
kylin.query.limit	0
kylin.query.offset	0
kylin.query.project	SanFrancisco
zeppelin.interpreter.localRepo	/usr/local/zeppelin/local-repo/2D21A2N61
zeppelin.interpreter.output.limit	102400

Figura 153. Propiedades del intérprete Kylin en Zeppelin – Alejandro Reina Reina

Destacando los campos de password y user que será los que hayamos configurado en Kylin, en mi caso mantengo dicha configuración por defecto, y los campos “Kylin.query.limit” que indica el límite del tamaño de la Query a devolver, en mi caso ha sido puesto a cero para eliminar dicho limite, y el campo “Kylin.query.project” que indica el proyecto donde está el cubo a consultar.

Una vez configurado, desde nuestro propio proyecto en Kylin, podemos realizar consultas utilizando el cubo, indicando desde el notebook Zeppelin que ejecute el código utilizando el intérprete de Kylin, como vemos a continuación:

IDCATEGORY	EXPRES1
1391569403904	1
300647710720	320
1400159338496	17
326417514496	4
601295421440	203

Figura 154. Error en el step Hive Cleanup – Alejandro Reina Reina

6.3. Herramientas de cuadros de mando con Kylin

Aunque las anteriores formas de consultar el cubo son totalmente validas, para ejecutar una consulta requiere conocimientos en SQL y conocer el sistema para realizar correctamente dichas consultas, por eso, se monta una herramienta de Dashboard como MicroStrategy, PowerBI, Pentaho, Tableau entre otros.

6.3.1. Driver ODBC

Para poder conectar estas herramientas, en la web oficial de Kylin nos indican que una de las cosas que debemos utilizar para conectar con Kylin, es el driver ODBC(Open DataBase Connectivity) que ellos mismos proporcionan (77). Cabe destacar que es distinto driver dependiendo de si nuestro procesador es a 32 o 64 bits, en mi caso utilizare el driver ODBC 64 bits. Tras instalarlo en Windows, instalación típica de Windows, abriremos el programa instalado “Orígenes de datos ODBC”, donde deberemos configurar la conexión con nuestro servidor Kylin para poder realizar las consultas.

En primer lugar, tenemos que crear la conexión DSN (Data Source Name), para ello, haremos clic en la pestaña DSN de sistema y haremos clic en agregar como vemos a continuación:

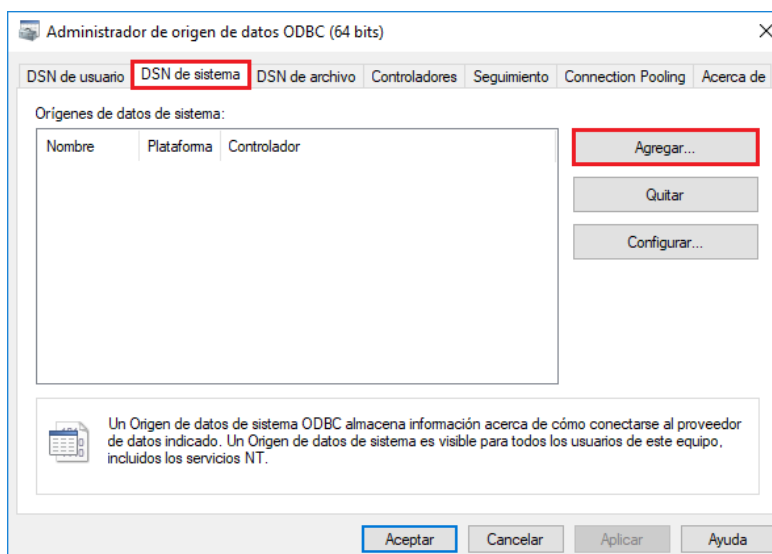


Figura 155. Agregar DSN de sistema – Alejandro Reina Reina

Después tenemos que seleccionar el driver ODBC de Kylin, como vemos en la imagen:

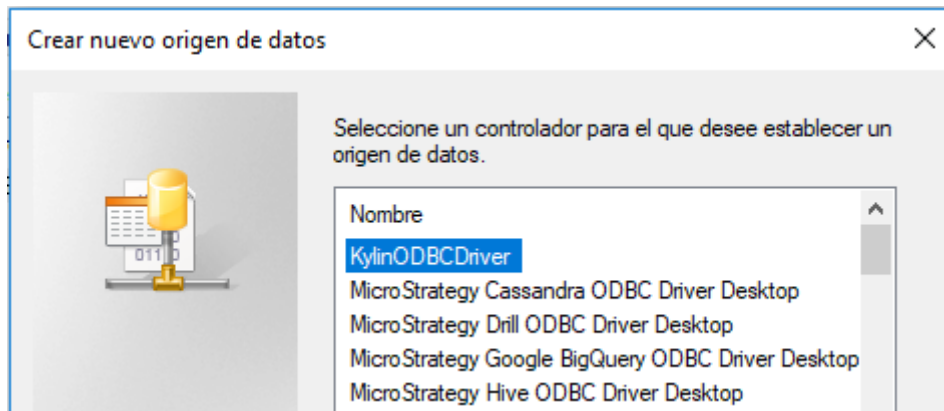


Figura 156. Seleccionar driver ODBC de Kylin – Alejandro Reina Reina

Y a continuación insertaremos los datos de nuestro servidor para poder realizar la conexión, como vemos a continuación:

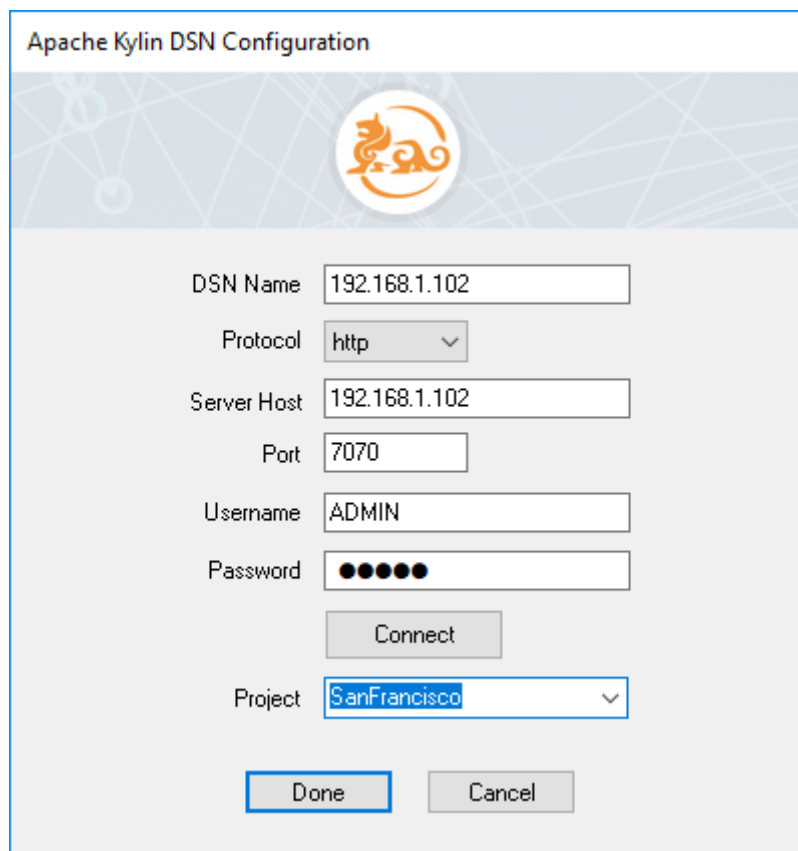


Figura 157. Apache Kylin DSN Configuration – Alejandro Reina Reina

En mi caso, tanto en Server Host como DSN Name es la IP de mi máquina virtual, indicándole el puerto de conexión, usuario y contraseña, tras hacer clic en “Connect”, si la conexión ha ido correctamente podremos indicar el proyecto a conectar, y finalmente Done.

6.3.2. Conectar PowerBI a Kylin

Después de configurar el driver ODBC para Kylin, podemos utilizar distintas herramientas de consulta, como mencione anteriormente. En mi caso, utilizare PowerBI, aunque es una herramienta de pago, nos permite utilizarla siempre y cuando no publiquemos el Dashboard, por lo que, en futuras ampliaciones, se utilizara Pentaho, que es open source, aunque esto también implica que no haya tanta documentación y más difícil su integración, y ya que diseñar un Dashboard no es objetivo de este proyecto, para probar el servidor de consulta utilizaremos PowerBI.

Tras instalarlo y ejecutarlo, haremos clic en *Get Data->Others*, donde seleccionaremos que queremos utilizar un driver ODBC y a continuación en “connect” como podemos ver en la siguiente imagen:

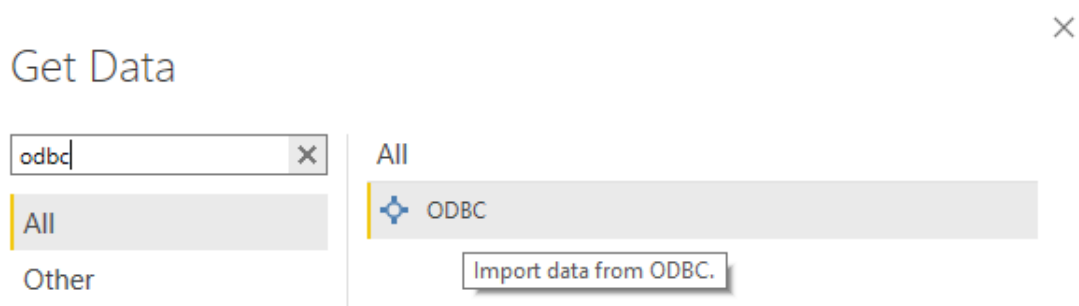


Figura 158. Seleccionar driver ODBC en PowerBI – Alejandro Reina Reina

Ahora seleccionaremos la conexión DSN que creamos anteriormente:

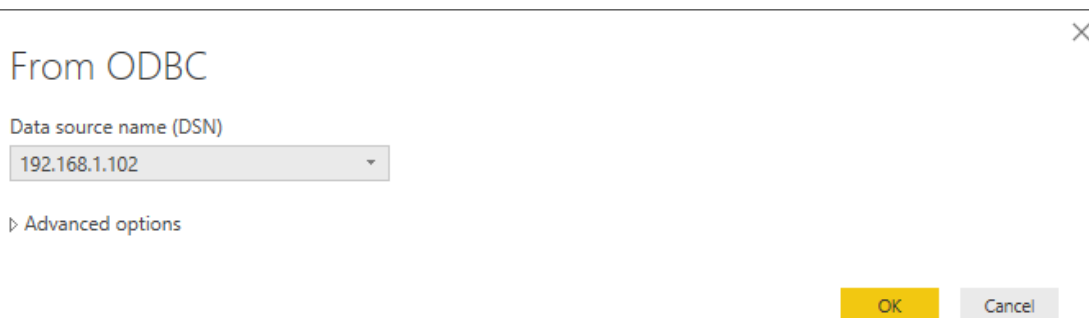


Figura 159. Seleccionar DSN en PowerBI – Alejandro Reina Reina

El siguiente paso es seleccionar aquellas tablas que queramos utilizar para la creación de nuestro Dashboard, y haremos clic en “load” como podemos ver a continuación:

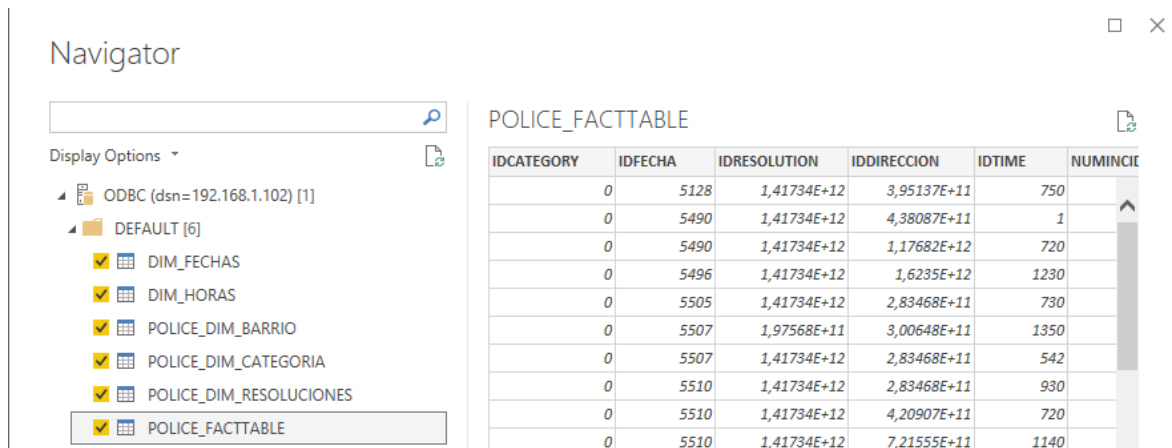


Figura 160. Seleccionar tablas para cargar en PowerBI – Alejandro Reina Reina

Una vez cargadas las tablas, necesitamos hacer otro paso para poder empezar a consultar el cubo correctamente, esto es debido a que la definición de estas tablas no es tan rica semánticamente como si utilizásemos un servidor de consulta sobre MySQL, por lo que deberemos definir en PowerBI las relaciones de nuestro modelo multidimensional, que creamos en el punto 5.4.4, para realizar esta tarea, haremos clic en el icono “Manage Relationships” y en “new” para crear dichas relaciones, donde seleccionaremos las tablas y las columnas que relacionan estas tablas, como vemos en la siguiente imagen:

Edit relationship

Select tables and columns that are related.

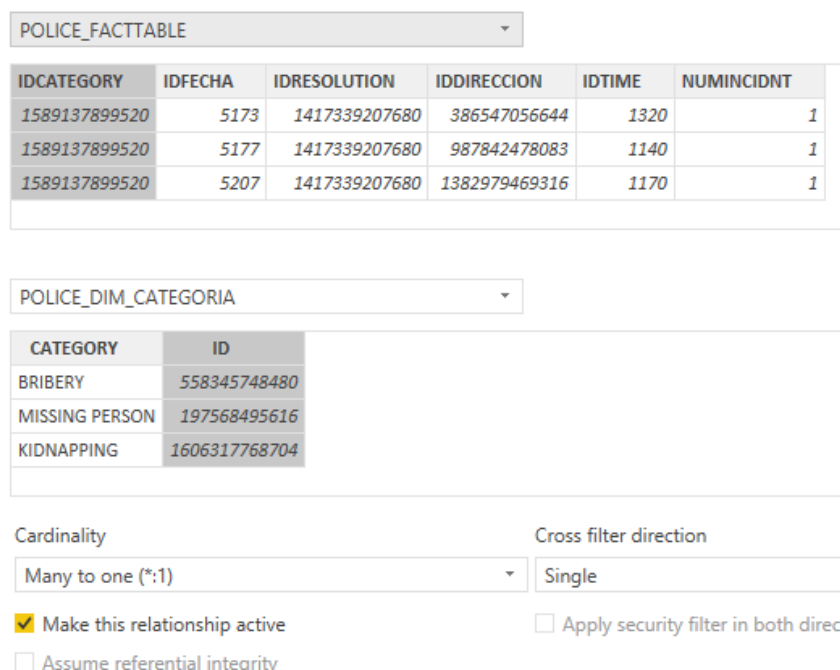


Figura 161. Nueva relación en PowerBI – Alejandro Reina Reina

Finalmente, una vez tengamos todas las relaciones creadas, quedara como en la siguiente imagen:

Manage relationships

Active	From: Table (Column)	To: Table (Column)
<input checked="" type="checkbox"/>	POLICE_FACTTABLE (IDCATEGORY)	POLICE_DIM_CATEGORIA (ID)
<input checked="" type="checkbox"/>	POLICE_FACTTABLE (IDDIRECCION)	POLICE_DIM_BARRIO (ID)
<input checked="" type="checkbox"/>	POLICE_FACTTABLE (IDFECHA)	DIM_FECHAS (ID)
<input checked="" type="checkbox"/>	POLICE_FACTTABLE (IDRESOLUCION)	POLICE_DIM_RESOLUCIONES (ID)
<input checked="" type="checkbox"/>	POLICE_FACTTABLE (IDTIME)	DIM_HORAS (ID)

Figura 162. Relaciones de tablas en PowerBI – Alejandro Reina Reina

Una vez creada estas relaciones podremos utilizar ya PowerBI para realizar consultas al cubo o crear un Dashboard, a continuación, muestro el resultado de una consulta con PowerBI utilizando el cubo Kylin.

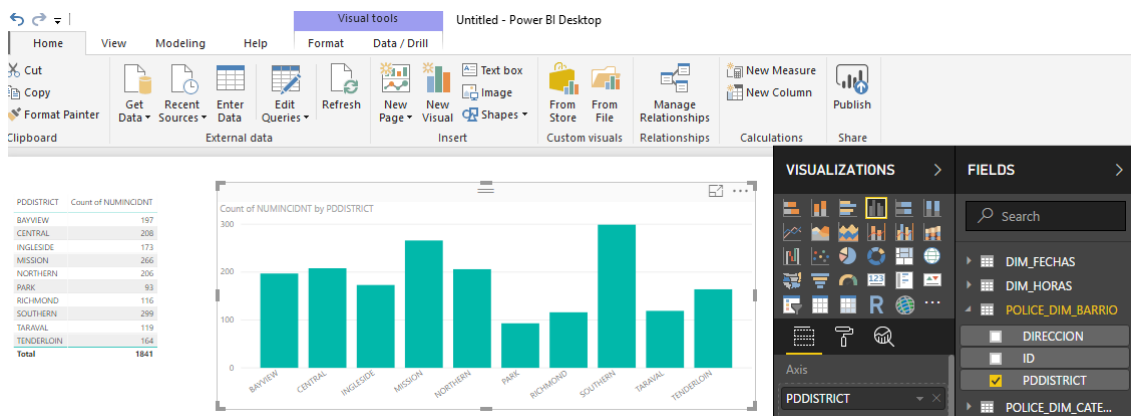


Figura 163. Consulta con PowerBI a Kylin – Alejandro Reina Reina

Finalmente podemos jugar con los gráficos, y explorar los datos, siempre con el objetivo de obtener información, y respondiendo a las preguntas analíticas definidas anteriormente en el punto 5.4.2, podremos obtener los siguientes gráficos de los distintos cubos.

6.4. Police Department Incident

En el siguiente punto, me centrare en las preguntas analíticas definidas anteriormente para el Datamart Police Department Incident.

6.4.1. Incidentes por Barrio

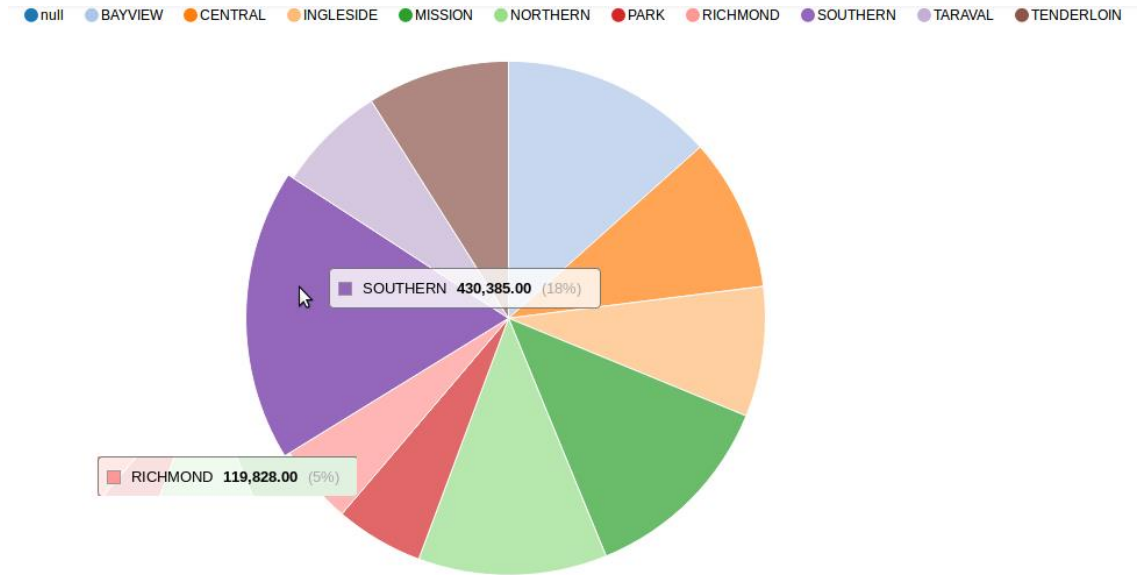


Figura 164. Grafica Número de incidentes por barrio – Alejandro Reina Reina

En la siguiente grafica podemos ver como se distribuye la frecuencia de incidentes por barrio, donde el barrio donde más incidentes se produce es Southern con 430.385 incidentes, un 18% de los incidentes ocurridos en San Francisco. En el otro extremo tenemos el barrio Richmond con 119.828 incidentes producidos en los últimos 20 años, un 5% de los incidentes.

6.4.2. Incidentes más frecuentes

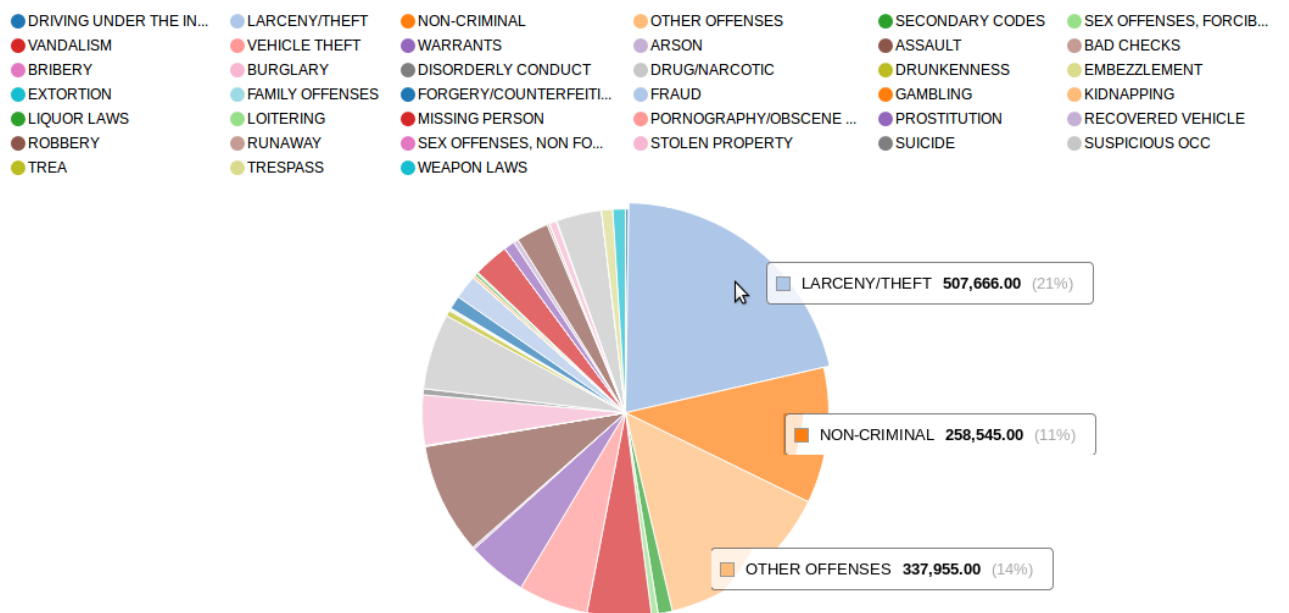


Figura 165. Gráfica Incidentes más frecuentes – Alejandro Reina Reina

En esta grafica vemos la proporción de los incidentes según su tipo o categoría, donde podemos ver que el robo/hurto es el tipo de incidente que se produce con más frecuencia, siendo un 21% de los incidentes ocurridos de este tipo, seguido de otras categorías, como incidentes no criminales u otras ofensas.

6.4.3. ¿Cuántos robos se saldan con la detención de alguien? ¿En cuántos no ocurre nada?

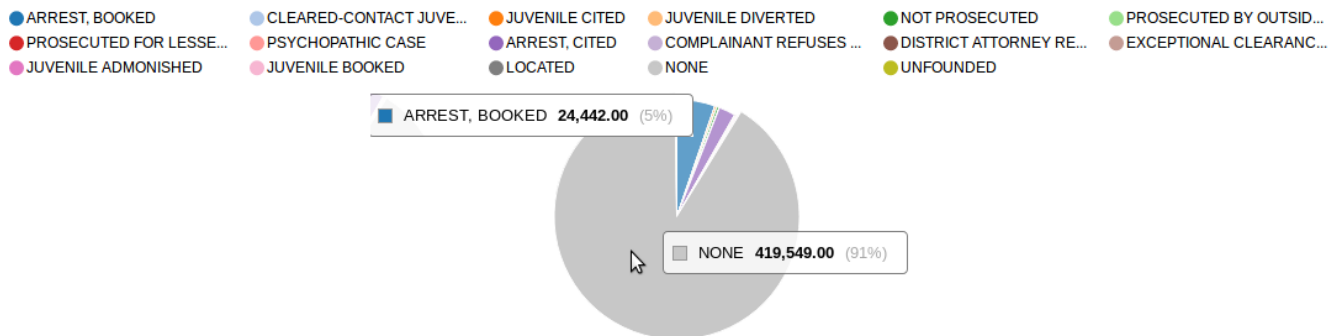


Figura 166. Gráfica Resoluciones de Robos/hurto – Alejandro Reina Reina

Explorando un poco más los datos, nos encontramos que, si analizamos los incidentes de tipo robos, el 91% de estos incidentes se quedan abiertos o no producen ninguna detención, siendo solo un 5% de estos incidentes que se saldan con el arresto y amonestación del implicado, y un 4% que se salda con arrestado/citado. En este punto nos damos cuenta que deberían de tomarse medidas para cambiar estos números ya que en un porcentaje muy alto de los robos no ocurre nada.

6.4.4. ¿Cuándo es más frecuente en un día que ocurra un incidente?

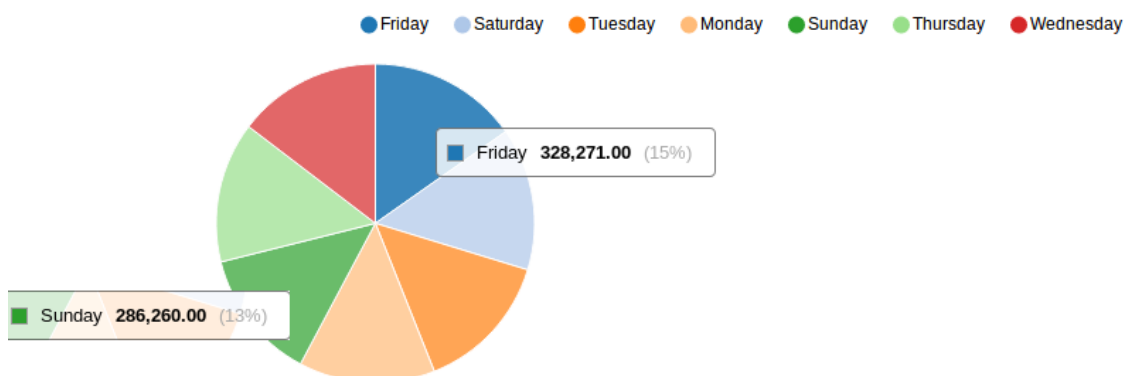


Figura 167. Gráfica Incidentes por día de la semana – Alejandro Reina Reina

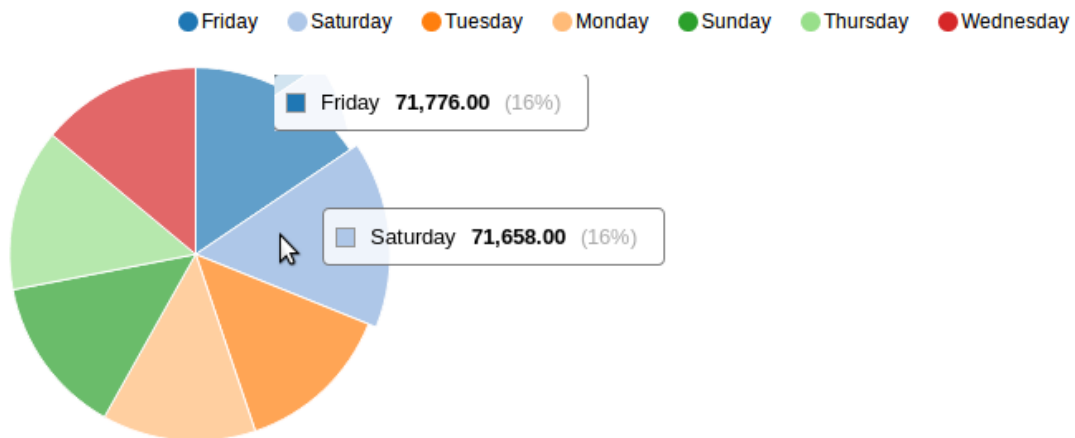


Figura 168. Gráfica Frecuencia de incidentes por día de la semana – Alejandro Reina Reina

En estas graficas vemos cuando son los días que más incidentes se producen siendo el viernes y el sábado ligeramente superior a los incidentes ocurridos en otros días con un 15% de los incidentes producidos, siendo el domingo el día que menos incidentes se producen con un 13% de los incidentes ocurridos. Si exploramos un poco más en profundidad y filtramos por asaltos, vemos que esta proporción se mantiene, aumenta ligeramente en el % de incidentes ocurridos por día de la semana subiendo a un 16% de los robos ocurridos sábado y otro 16% de robos en sábado. De nuevo el domingo es el día donde ligeramente se producen menos incidentes de este tipo.

6.4.5. Meses del año que ocurren con mayor frecuencia un incidente

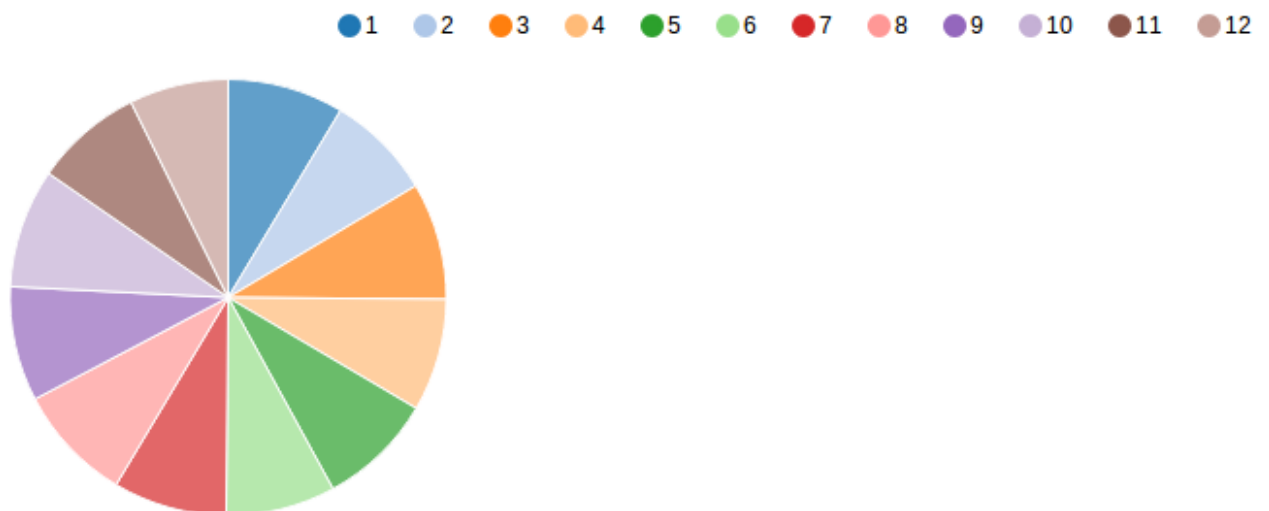


Figura 169. Gráfica Frecuencia de incidentes por meses – Alejandro Reina Reina

Cuando observamos los incidentes producidos por meses, nos damos cuenta de que no hay ningún mes donde estos incidentes se produzcan con mayor frecuencia, agosto y septiembre son ligeramente superiores, pero no de manera significativa con respecto a los otros meses.

6.4.6. ¿Todos los tipos de robos ocurren por igual o su frecuencia se altera en el tiempo?

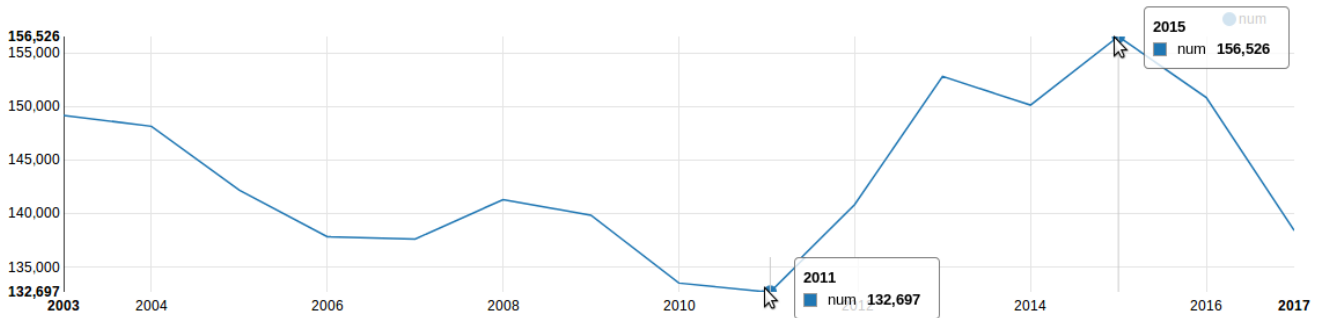


Figura 170. Gráfica Incidentes por año – Alejandro Reina Reina

Con esta pregunta queremos ver que ocurren con la frecuencia de los incidentes a lo largo del tiempo, donde vemos que en el año 2011 fue el año que menos incidentes se produjeron con 132.697 incidentes, siendo en 2015 el año donde más incidentes se producen registrados por la policía de San Francisco, siendo la cifra de 156.526 incidentes, en los últimos años, esta cifra está reduciéndose, aunque con los datos que tenemos no podemos inferir si se debe a medidas que se han tomado por parte de la ciudad de San Francisco, o a una mera casualidad, aun así la media aproximada de los incidentes ocurridos por año sería cerca de 140.000 incidentes.

6.5. Fire Department Calls for Service

Vamos a ver ahora de manera gráfica la respuesta a las preguntas analíticas sobre el fichero Fire_Department Calls for service, es decir, las llamadas al departamento de bomberos en los últimos 20 años, este fichero, es mucho mayor que el anterior, por lo que la información que tenemos es mayor y podemos hacer un mejor estudio.

6.5.1. ¿Se llama por igual de todos los vecindarios?

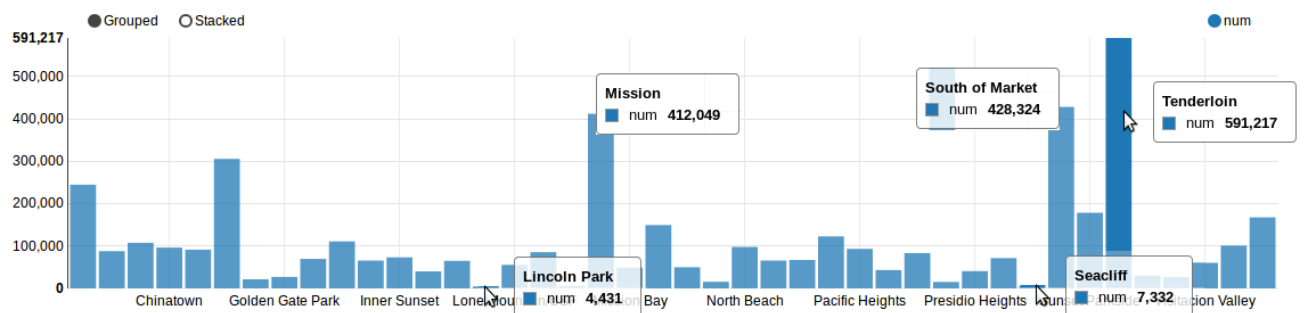


Figura 171. Gráfica Llamadas por barrio – Alejandro Reina Reina

Analizando las llamadas que se realizan desde los distintos barrios, vemos que hay una mayor participación o, dicho de otra manera, una mayor cantidad de incidentes en los barrios de Tenderloin con 591.217, South of Market y Mission. En el otro extremo, los barrios donde menos incidentes o llamadas se producen, son Lincoln Park y Seacliff, un 1% frente a los otros barrios, por lo que, si es una diferencia significativa a tener en cuenta, y podríamos investigar por qué ocurre esto.

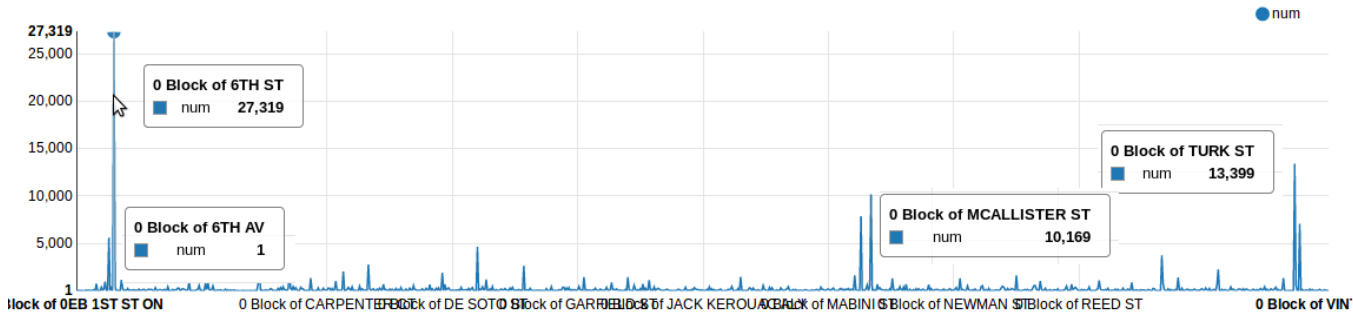


Figura 172. Gráfica Llamadas por dirección – Alejandro Reina Reina

Con el fin de forzar el sistema, la dimension localización se hizo acorde al número de bloques o direcciones que teníamos en el fichero, teniendo cerca de 30.000 direcciones distintas donde se ha producido alguna llamada al departamento de bomberos, y en esta grafica encontramos que la dirección “0 Block of 6TH ST” tiene un número muy superior de llamadas informando de incidentes. Ayudándonos en Google Maps, podemos ver una imagen de la zona que corresponde a esta dirección.

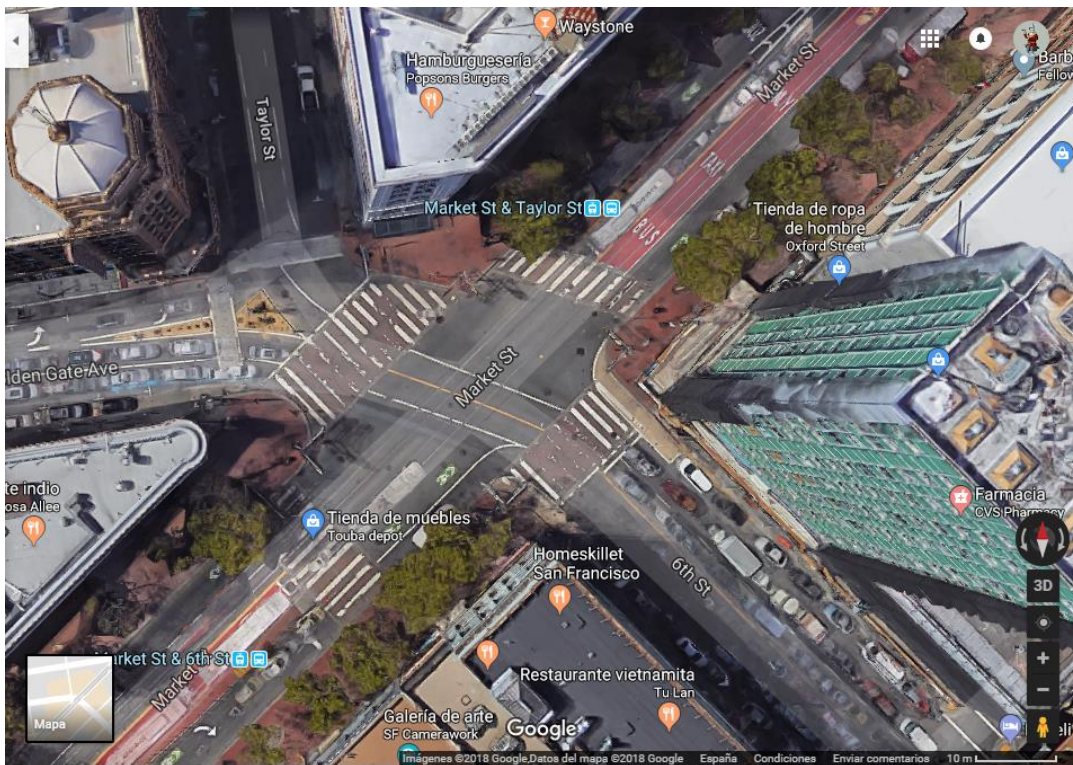


Figura 173. Gráfica Llamadas por dirección – Alejandro Reina Reina

Viendo la dirección donde más incidentes se producen vemos que no es ningún barrio marginal como pensaba al ver los datos, si no se corresponde con una zona de intersecciones entre distintas calles, rodeada de restaurantes y comercios, por lo cruzando información con el otro dataset, y considerando que el incidente que más se produce en San Francisco son los robos, tiene sentido que esta sea una de las zonas más conflictivas en cuanto a robos se refiere, aunque como veremos en la siguiente gráfica, esto tampoco puede ser cierto del todo, ya que la mayor cantidad de llamadas al departamento de bomberos, son por alarmas, incendios y incidentes médicos.

6.5.2. ¿Tipos de llamada tienen la misma proporción?

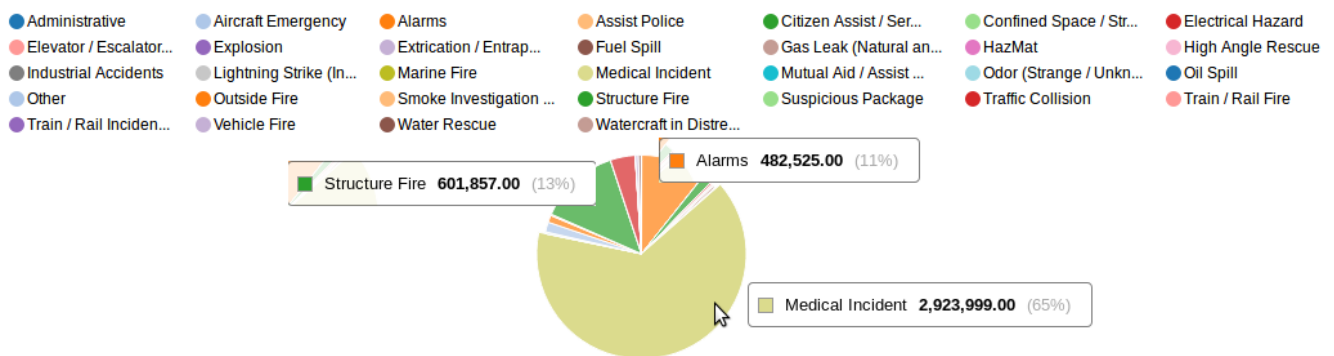


Figura 174. Gráfica Tipos de Llamada – Alejandro Reina Reina

Analizando los tipos de llamadas que se producen al departamento de bomberos, vemos que los tres tipos de llamada que más se producen son por incidentes médicos con un 65% de las llamadas, y casi 3 millones de llamadas de este tipo, seguido de alarmas e incendios, con un 11% y un 13% respectivamente.

6.5.3. ¿Tiempo medio de respuesta por tipo de llamada?

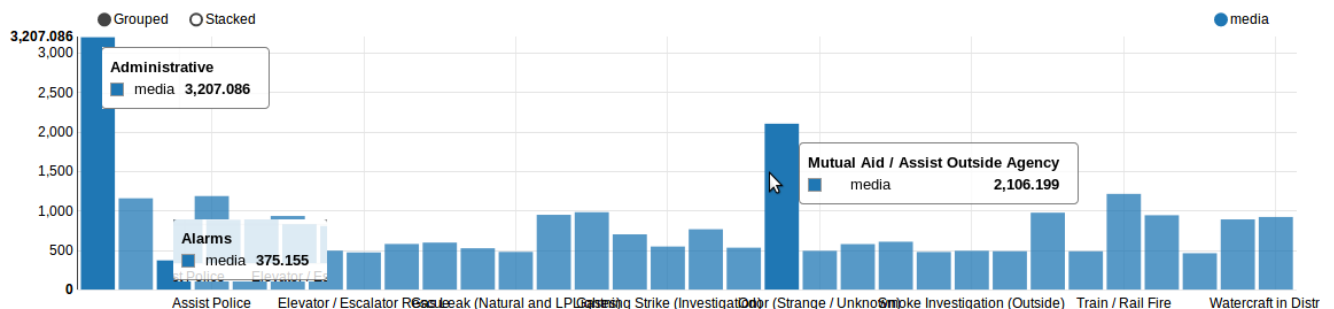


Figura 175. Gráfica Tiempo medio de Respuesta – Alejandro Reina Reina

Analizando el tiempo medio de respuesta de los batallones a la escena del crimen de aquellas llamadas que requieren asistencia, vemos que las llamadas en la categoría de administrativo, y en aquellas de ayuda mutua o ayuda a agencias externas son aquellas que más tiempo en llegar a la escena requieren. Siendo las alarmas las que menos tiempo requieren para situarse en la

escena con poco más de 6 minutos desde que la llamada es transferida al batallón hasta que el batallón llega a la escena.

6.5.4. Evolución de las llamadas por año

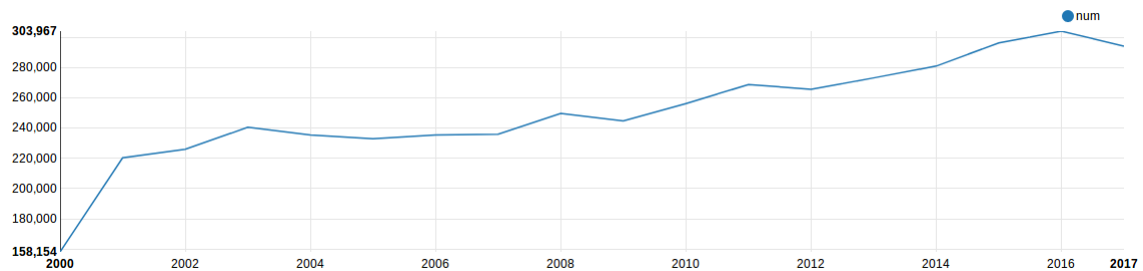


Figura 176. Gráfica Evolución de las llamadas por año – Alejandro Reina Reina

Con la siguiente grafica se pretende estudiar la evolución de las llamadas al departamento de bomberos, donde vemos que en los últimos 20 años ha ido creciendo, llegando a duplicarse el número de llamadas desde el 2000 al 2016 que es el año donde más llamadas se produjeron con 303.967 llamadas al departamento de bomberos.

6.5.5. Evolución de las llamadas por tipo y meses del año

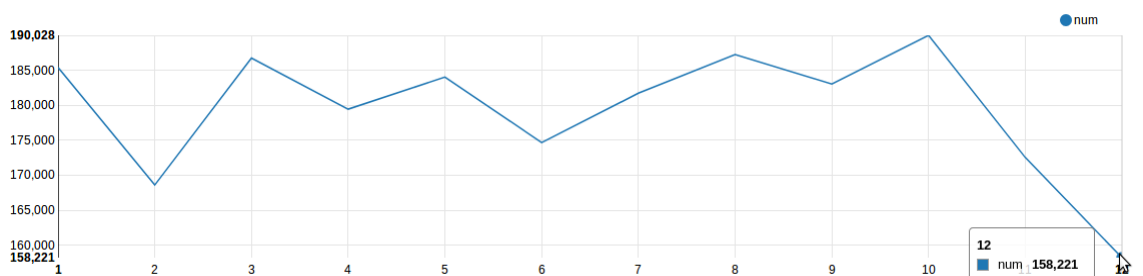


Figura 177. Gráfica Evolución de las llamadas por meses – Alejandro Reina Reina

Esta grafica me ha resultado también interesante, porque estudiamos la evolución de las llamadas según el mes del año, donde podemos ver que en el mes de diciembre es el mes que menos llamadas al departamento de bomberos se hacen siendo octubre el mes que mas llamadas se reciben.

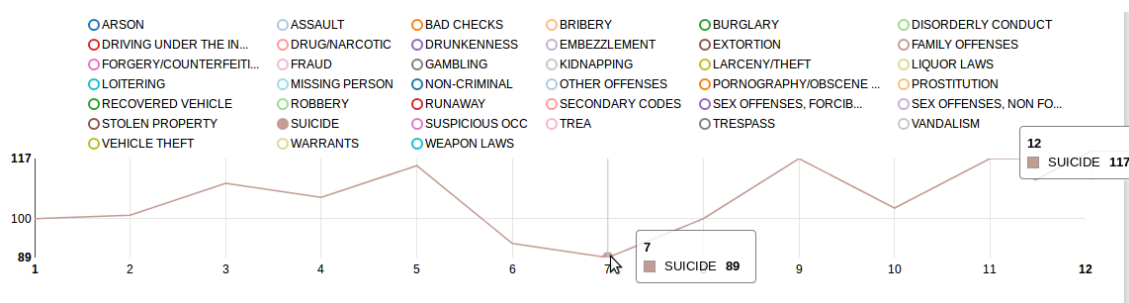


Figura 178. Gráfica Evolución de las llamadas de suicidio por meses – Alejandro Reina Reina

Siguiendo la evolución de las llamadas por meses, he añadido la evolución de estas llamadas por categoría de la llamada, y en general he encontrado algunos patrones, por ejemplo los meses de Mayo y Junio y a veces Febrero, son meses donde se producen menos incidentes, también debido al número de llamadas recibidas, pero a pesar de eso, vemos como hay algunas categorías donde en diciembre se aumenta a pesar de la disminución de las llamadas como es el caso de los suicidios producidos durante ese mes, o las ofensas familiares que también se ve incrementada en esas fechas.

En las siguientes dos graficas vamos a ver como el patrón se reduce en navidades, aunque esto se deberá probablemente a la reducción del número de llamadas, aunque cruzando datos con el dataset de la policía, se producen los mismos incidentes que en otros meses, las llamadas a este servicio en navidades se reducen, como vemos a continuación en las llamadas producidas al departamento de bomberos por asalto y casos de drogas y narcóticos.

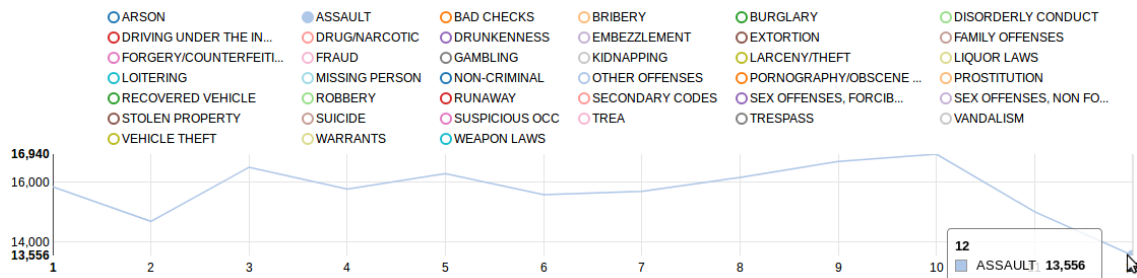


Figura 179. Gráfica Evolución de las llamadas de asalto por meses – Alejandro Reina Reina

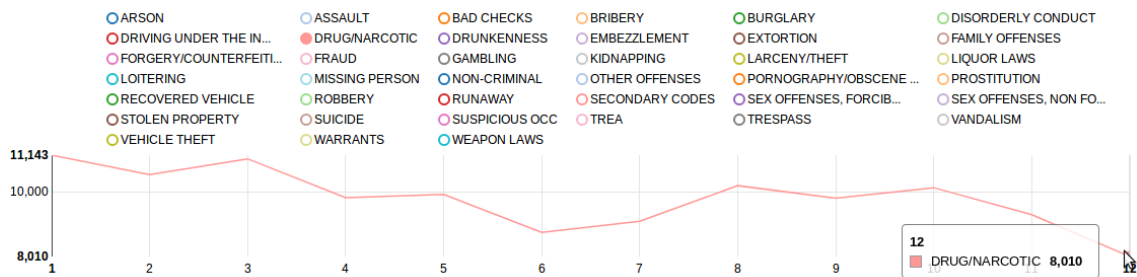


Figura 180. Gráfica Evolución de las llamadas de drogas y narcóticos por meses – Alejandro Reina Reina

6.5.6. ¿Los batallones de bomberos responden el mismo número de llamadas?

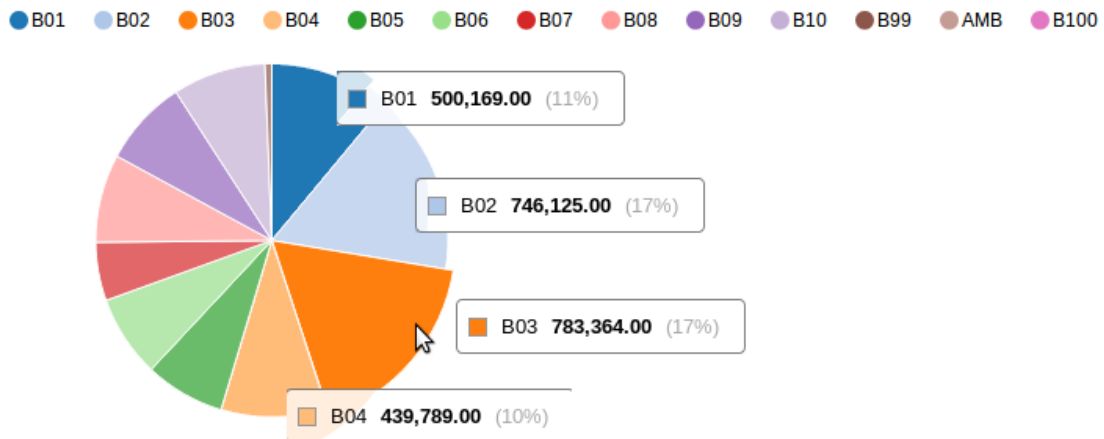


Figura 181. Gráfica Número de llamadas por batallones – Alejandro Reina Reina

Haciendo un estudio de las llamadas que reciben los distintos batallones de bomberos, vemos que los batallones 1, 2, 3 y 4 reciben prácticamente el 50% de las llamadas siendo el batallón 2 y 3 los que más reciben con un 17% de las llamadas cada uno, luego los batallones 1 y 4 reciben un 11% y un 10% de las llamadas respectivamente, cifra que va disminuyendo en el resto de batallones.

6.5.7. ¿Hay batallones especializados en tipos específicos de llamadas?

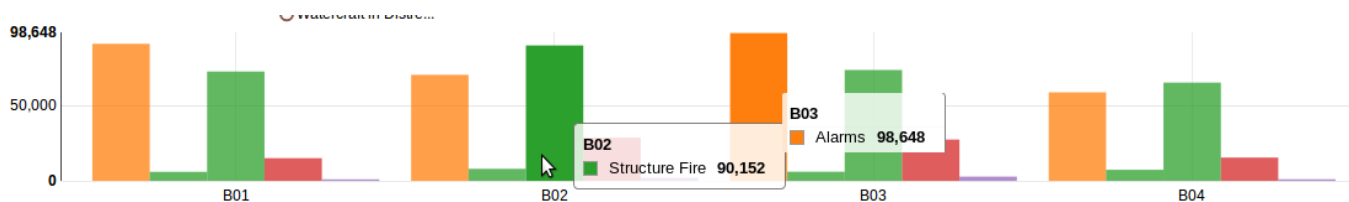


Figura 182. Gráfica Llamadas más frecuentes por batallones – Alejandro Reina Reina

Explorando los batallones que más llamadas reciben, vemos que no hay una gran especialización de los batallones por tipo de llamadas, aunque ligeramente y teniendo en cuenta que el batallón 2 y 3 son los que más llamadas reciben teniendo aproximadamente el mismo número, vemos que el batallón 3 responde a más alarmas que el batallón 2, y a la inversa ocurre con los incendios, el batallón 2 recibe más llamadas de este tipo, aunque esto no indica que sean especializados en tipos de llamadas, estos datos pueden tener que ver con la zona que atienden estos batallones, donde se produzcan más de un tipo que de otro.

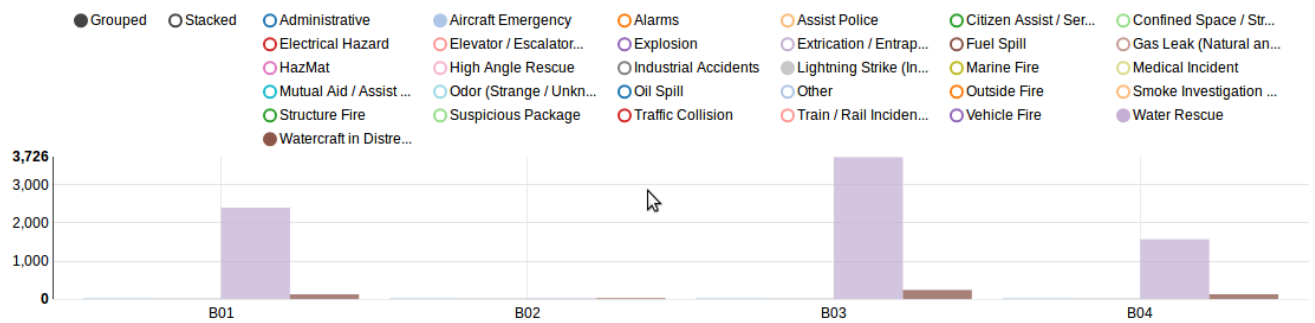


Figura 183. Gráfica Tipos de Llamadas por batallón – Alejandro Reina Reina

En la última grafica que tenemos, he ido explorando las llamadas que reciben los batallones por categorías, donde podemos ver que el batallón 2 no ha atendido ningún caso por caída de rayos, emergencias de aeronaves, embarcaciones en peligro o rescates de agua.

7. Conclusiones

Una vez finalizado el proyecto, es momento de sacar conclusiones. Dichas conclusiones voy a exponerlas desde mi punto de vista, y basado en mi experiencia con los sistemas que he visto.

El año pasado, en una asignatura (Gestión de la información), uno de los puntos de esa asignatura fue la población masiva de una BD de MySQL, donde intentamos introducir un millón de tuplas en dicha BD, esto fue un proceso muy muy lento, ya que yo y otro compañero dejamos toda la noche enchufado Pentaho para insertar estos datos en la BD, introduciendo en la noche aproximadamente unos 500.000 registros en dicha BD, aproximadamente introducíamos unas 200 tuplas por segundo, aunque esta velocidad puede ser mayor en otras BD, raramente pasaremos la inserción de 1000 tuplas por segundo.

En cuanto al sistema que he montado, fue capaz de leer, y procesar en la fase de ETL, un fichero de 2,5 millones de tuplas, creando sus respectivas dimensiones, y calculando las medidas en un tiempo de 6 minutos, es decir, en 6 minutos ya teníamos cargado todos los datos en el datawarehouse y pudiendo ser analizados, teniendo en cuenta los recursos con los que se ha desarrollado el proyecto (Linux virtualizado sobre Windows), donde forzando los recursos asignados a dicha máquina, se le pudo asignar 2 núcleos de un procesador i5 6200U, y 5 GB de RAM, el cual el propio sistema operativo Linux, consumía recursos (2GB de RAM es lo mínimo para ejecutar Linux 16.04), por lo que hay que tener en cuenta que todo este sistema al final ha corrido con aproximadamente con 3GB de RAM, por lo tanto si tenemos esto en cuenta, considero que la velocidad que aporta Spark y Hive es extraordinaria, de hecho, no esperaba esa velocidad de procesamiento en ningún momento, de hecho incluso esta carga de ETL, la realice por la noche esperando que sería un proceso de horas, pero me equivocaba.

También hay que tener en cuenta, que en este proyecto, y esto tiene que ver con futuras ampliaciones, no hemos comentado ni desarrollado nada del clúster con Hadoop, esto multiplica de manera lineal la escalabilidad y potencia de procesamiento, es decir, que si hubiese montado un clúster de 6 ordenadores con las características mencionadas anteriormente, habría durado 1 minuto en procesar e insertar estos datos y estar listos para ser procesados, algo que en un sistema BI tradicional sería impensable.

También me gustaría destacar Docker en el desarrollo de este TFG, ya que la funcionalidad que aporta en despliegues rápidos de servidores, así como la administración de recursos de dichos servidores, es algo totalmente nuevo que no se había visto ahora, como anécdota, hace unos días Windows se me actualizo, y a consecuencia de ello, no pude entrar a las máquinas de virtual box (aunque es algo que solucione días después), pero una vez instale un nuevo sistema operativo Linux sobre Virtual Box, en aproximadamente unos 20 minutos ya tenía descargado y configurado todos los servicios tal y como lo tenía antes, y ejecutando simplemente un script, esto es algo muy muy potente, ya que nos permite en entornos de producción volver a levantar un servidor desde cero en cuestión de minutos.

Si he de ser franco, a pesar de que utilizo mucho la palabra BigData a lo largo del proyecto, porque utilizo tecnología Big Data, hay que destacar que el tamaño de los datos tratados en este proyecto no son ni la B de Big Data, Big Data es mucho más, como ejemplos comentados

anteriormente sobre la cantidad de datos que se procesan por ejemplo la NASA, Facebook, Google, Amazon u otros muchos gigantes, de hecho, en los últimos años se han generado más datos que en toda la historia de la humanidad, algunos estudios hablan de que para 2020 tendremos cerca de 44 Zeta Bytes, para hacernos una idea de que es esto, 1 ZB es 1 billón de GB europeo o 1000 billones de GB americanos, una auténtica locura, otro ejemplo de esto, lo tenemos en un artículo del país, donde un coche autónomo genera 3,6 TB de datos por una hora de conducción autónoma, al año sería casi un Petabytes solo un coche autónomo, conduciendo unas 260 horas en dicho año. (77) (78)

A nivel personal estoy muy satisfecho con el trabajo realizado, y más aun con los conocimientos adquiridos, ya que como remarque al principio de esta memoria, era un mundo desconocido para mí, un mundo que tiene mi interés y a lo largo de mi carrera universitaria, no hemos aprendido nada sobre este mundo, por lo que este proyecto ha sido ideal para iniciar mi formación en lo que a tecnología y mundo Big Data se refiere, porque esto es solo la punta del iceberg.

En definitiva, de manera gráfica, lo que hemos montado, es uno de los servicios que ofrecen empresas como Amazon Web Services, o Azure, las cuales te permiten desplegar de manera rápida un servidor sobre el cual poder ejecutar y procesar tu información, entrenar algoritmos machine learning entre otros, en nuestro caso, estaríamos ofreciendo un servidor con todo lo necesario para realizar análisis sobre un gran conjunto de datos. Dicho de otra forma, hemos montado un servidor analítico similar a los que la NASA, Amazon Web Services o Azure disponen entre los servicios que ofrecen, a falta de otorgarle operatividad dentro de un clúster, pero solo un único servidor, cada uno de esos armarios pueden contener quizá 20 o 30 servidores en un único RACK



Figura 184. Clúster NASA – Alejandro Reina Reina

8. Posibles líneas de investigación futura.

Aunque este proyecto tiene muchas ramas de investigación futura, destacare las que a mi parecer son las más importantes.

8.1. Herramienta de visualización de datos

Por un lado, la inclusión de herramientas de creación de Dashboard como es Pentaho, Tableau, o PowerBI, las cuales facilitarían la creación de un servidor web desde el cual podríamos tener las consultas predefinidas listas para un usuario final no experto en dichas tecnologías, aportando potencia en la visualización y extracción de la información.

8.2. Clúster Hadoop

Otra de las líneas de investigación y desarrollo de este TFG, seria incorporar dicha tecnología dentro de un clúster Hadoop, donde cada uno de los integrantes de dicho clúster pudiese procesar y almacenar parte de la información aumentando la escalabilidad y potencia de cálculo como se ha mencionado a lo largo de este proyecto.

8.3. Machine learning

Otra de las ramas de investigación seria preparar el servidor para aprendizaje automático, machine learning, como mencionamos en las pruebas de Scala vs Python, pero no hay que olvidar la potencia que aportan estos sistemas de aprendizaje a la información que podemos extraer de los datos.

8.4. Ampliación del sistema BI

Otro punto donde podemos seguir investigando, es reunir y analizar más ficheros sobre esta ciudad, que están publicados en el portal, con el fin de hacer un análisis más exhaustivo, y obtener más información de esta ciudad.

Referencias

1. **Panteleeva, Olga Vladimirovna.** *Fundamentos de Probabilidad y Estadística.* Fundamentos de Probabilidad y Estadística.
2. **Alliance, The Software.** <http://data.bsa.org/>. [En línea] http://data.bsa.org/wp-content/uploads/2015/10/BSADataStudy_es.pdf.
3. **Microsoft.** youtube.com. [En línea] https://www.youtube.com/watch?v=_1y5jBESLPE&feature=youtu.be.
4. **McCarthy, Dan.** [En línea] Febrero de 2017. <http://www.itbriefcase.net/how-business-intelligence-has-evolved-over-the-last-few-decades>.
5. **Oracle.** oracle.com. [En línea] http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/317529_esa.pdf.
6. **Kimball, Ralph.** *The data warehouse toolkit: the definitive guide to dimensional modeling.* 2003.
7. **Inmon, Bill.** *Building the data warehouse.* 2006.
8. **Berzal, Fernando.** *Warehousing, El modelo multidimensional Data.*
9. **Abramson, Ian.** *Datawarehouse: the choice of Inmon versus Kimball.* 2002.
10. **Bernabeu, Dario.** dataprix.com. [En línea] Mayo de 2006. <http://www.dataprix.com/datawarehouse-manager>.
11. *International Journal of Applied Information Systems (IJ AIS) – ISSN : 2249-0868.* **Varios.** Mayo de 2017, Vols. 12 - No 2.
12. **Google.** *The Google File System.* 2003.
13. **Google.** *MapReduce: Simplified Data Processing on Large Clusters.* 2008.
14. **Laney, Doug.** [En línea] Febrero de 2001. <https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>.
15. **Mattmann, Chris.** informationweek.com. [En línea] Noviembre de 2013. <https://www.informationweek.com/big-data/hardware-architectures/how-nasa-manages-big-data-/d/d-id/899791>.
16. **Fragoso, Ricardo Barranco.** ibm.com. [En línea] Junio de 2006. <https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/>.
17. **Turck, Matt.** mattturck.com. [En línea] Abril de 2017. <http://mattturck.com/bigdata2017/>.
18. **Sevilla, Javier Lahoz.** icemd.com. [En línea] Noviembre de 2016. <https://www.icemd.com/digital-knowledge/articulos/se-busca-experto-arquitectura-big-data/>.

19. **Dixon, James.** jamesdixon.wordpress.com. [En línea] Octubre de 2010.
<https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/>.
20. **Dixon, James.** YouTube.com. [En línea] Agosto de 2010.
https://www.youtube.com/watch?v=tR_yLsr87Uk.
21. **Dixon, James.** YouTube.com. [En línea] Agosto de 2010.
<https://www.youtube.com/watch?v=1CG01JmKp2Y>.
22. **Dixon, James.** jamesdixon.wordpress.com. [En línea] Septiembre de 2014.
<https://jamesdixon.wordpress.com/2014/09/25/data-lakes-revisited/>.
23. **Dixon, James.** jamesdixon.wordpress.com. [En línea] Enero de 2015.
<https://jamesdixon.wordpress.com/?s=data+lake>.
24. **Leo-Revilla, Ángel.** momentotic.com. [En línea] Mayo de 2013.
<https://momentotic.com/2013/05/16/que-es-hadoop/>.
25. **Hadoop.** hadoop.apache.org. [En línea] <http://hadoop.apache.org/>.
26. **Hess, Ken.** datamation.com. [En línea] Febrero de 2016.
<https://www.datamation.com/data-center/hadoop-vs.-spark-the-new-age-of-big-data.html>.
27. **Hadoop.** wiki.apache.org/hadoop. [En línea] <https://wiki.apache.org/hadoop/PoweredBy>.
28. **Hadoop.** hadoop.apache.org. [En línea]
https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.
29. **Recuero, Paloma.** data-speaks.luca-d3.com/. [En línea] Septiembre de 2017. <http://data-speaks.luca-d3.com/2017/09/hadoop-por-dentro-ii-hdfs-y-mapreduce.html>.
30. **Fernandez, Jesus Santisteban.** cetatech.ceta-ciemat.es. [En línea] Marzo de 2015.
<https://cetatech.ceta-ciemat.es/2015/03/desplegando-un-cluster-distribuido-con-hadoop-2-6-0/>.
31. **Google.** *Bigtable: A Distributed Storage System for Structured Data.* 2006.
32. **Zaforas, Manuel.** paradigmadigital.com. [En línea] Marzo de 2016.
<https://www.paradigmadigital.com/dev/cassandra-la-dama-de-las-bases-de-datos-nosql/>.
33. **Amster, Ari.** [En línea] Enero de 2016. <https://www.qubole.com/blog/cassandra-vs-hadoop/>.
34. *Solving Big Data Challenges for Enterprise Application.* **Rabl, Tilmann, y otros.** Agosto de 2012.
35. **Stackchief.** stackchief.com. [En línea] Septiembre de 2017.
<https://www.stackchief.com/blog/Hadoop%20HBase%20vs%20Cassandra>.
36. **Morales, Aurelio.** mappinggis.com. [En línea] Diciembre de 2017.
<https://mappinggis.com/2014/07/mongodb-y-gis/>.

37. **RubenFa**. genbetadev.com. [En línea] Febrero de 2014.
<https://www.genbetadev.com/bases-de-datos/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no>.
38. **Facebook Data Infrastructure Team**. *Proceedings of the VLDB Endowment*. 2009. Vol. Volume 2 Issue 2.
39. **Parquet**. parquet.apache.org. [En línea] <http://parquet.apache.org>.
40. **Hive**. cwiki.apache.org. [En línea]
<https://cwiki.apache.org/confluence/display/Hive/DeveloperGuide#DeveloperGuide-HiveSerDe>.
41. **SQL, Spark**. spark.apache.org. [En línea] <https://spark.apache.org/sql/>.
42. **MLib, Spark**. spark.apache.org. [En línea] <https://spark.apache.org/mllib/>.
43. **GraphX, Spark**. spark.apache.org. [En línea] <https://spark.apache.org/graphx/>.
44. **Streaming, Spark**. spark.apache.org. [En línea]
<https://spark.apache.org/docs/latest/streaming-programming-guide.html>.
45. **Esteso, Mario Pérez**. geekytheory.com. [En línea] <https://geekytheory.com/apache-spark-que-es-y-como-funciona/>.
46. **Spark**. spark.apache.org. [En línea] <http://spark.apache.org/releases/spark-release-2-0-0.html#removals-behavior-changes-and-deprecations>.
47. **indatalabs.com**. [En línea] Abril de 2017. <https://indatalabs.com/blog/data-engineering/convert-spark-rdd-to-dataframe-dataset>.
48. **Luangsay, Souryigna**. bbvaopen4u.com. [En línea] Enero de 2015.
<https://bbvaopen4u.com/es/actualidad/spark-la-proxima-tecnologia-estrella-de-big-data>.
49. **OpenHub, Spark**. openhub.net. [En línea] <https://www.openhub.net/p/apache-spark>.
50. **Dezyre.com**. dezyre.com. [En línea] Febrero de 2016.
<https://www.dezyre.com/article/scala-vs-python-for-apache-spark/213>.
51. **JanBask**. janbasktraining.com. [En línea] octubre de 2017.
<https://www.janbasktraining.com/blog/scala-vs-python/>.
52. **Paredes, Javier García**. gruposolutio.com. [En línea] Julio de 2017.
<http://www.gruposolutio.com/blog/index.php/2017/07/21/lenguaje-data-science/>.
53. **Kylin**. kylin.apache.org. [En línea] <http://kylin.apache.org/>.
54. **Zeppelin**. zeppelin.apache.org. [En línea] <http://zeppelin.apache.org/>.
55. **hortonworks.com**. [En línea] <https://es.hortonworks.com/apache/zeppelin/>.
56. **Zeppelin**. zeppelin.apache.org. [En línea] 0.7.3.
<https://zeppelin.apache.org/docs/0.7.3/rest-api/rest-interpreter.html>.

57. **docker.com.** [En línea] <https://www.docker.com/>.
58. **Mula, Ángel Luis.** babel.es. [En línea] Febrero de 2017. <http://babel.es/es/blog/blog/febrero-2017/que-es-docker>.
59. **Microsoft.** docs.microsoft.com. [En línea] Septiembre de 2017. <https://docs.microsoft.com/es-es/dotnet/standard/containerized-lifecycle-architecture/what-is-docker>.
60. **Mula, Ángel Luis.** babel.es. [En línea] Febrero de 2017. <http://babel.es/es/blog/blog/febrero-2017/que-es-docker>.
61. **Scala.** scala-lang.org. [En línea] <http://www.scala-lang.org/files/archive/>.
62. **Spark.** spark.apache.org. [En línea] <http://spark.apache.org/downloads.html>.
63. **Spark.** spark.apache.org. [En línea] <https://spark.apache.org/docs/latest/>.
64. **HBase.** hbase.apache.org/. [En línea] <https://hbase.apache.org/>.
65. **Hive.** hive.apache.org. [En línea] <https://cwiki.apache.org/confluence/display/Hive/LanguageManual>.
66. **R, Nair.** apache-spark-user-list. [En línea] <http://apache-spark-user-list.1001560.n3.nabble.com/Save-hive-table-from-spark-in-hive-2-1-0-td30247.html#a30254>.
67. **Zeppelin.** zeppelin.apache.org. [En línea] <https://zeppelin.apache.org/download.html>.
68. **Kylin.** kylin.apache.org. [En línea] <http://kylin.apache.org/docs21/>.
69. **Docker, doc.** docker.com. [En línea] <https://docs.docker.com/engine/reference/builder/>.
70. **City of San Francisco. datasf.org.** [En línea] <https://datasf.org/opendata/>.
71. **City of San Francisco— data.sfgov.org.** [En línea] <https://data.sfgov.org/Public-Safety/Police-Department-Incidents/tmnf-yvry>.
72. **joda.org. joda.org.** [En línea] <http://www.joda.org/joda-time/>.
73. **Kylin. kylin.apache.org.** [En línea] kylin.apache.org/docs/howto_optimize_cubes.html.
74. **Hadoop. hadoop.apache.org.** [En línea] <https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html>.
75. **Kylin. kylin.apache.org.** [En línea] http://kylin.apache.org/docs21/release_notes.html.
76. **Kylin. kylin.apache.org.** [En línea] http://kylin.apache.org/docs21/howto/howto_cleanup_storage.html.
77. **Kylin. kylin.apache.org.** [En línea] <http://kylin.apache.org/docs21/tutorial/odbc.html>.
78. **elpais. elpais.com.** [En línea] https://retina.elpais.com/retina/2017/11/17/tendencias/1510920126_844738.html.

79. Universia. *universia.es*. [En línea]

<http://noticias.universia.es/cultura/noticia/2015/10/01/1131820/15-cifras-sorprendentes-big-data.html>.

80. Hive. *cwiki.apache.org*. [En línea]

<https://cwiki.apache.org/confluence/display/Hive/DeveloperGuide#DeveloperGuide-FileFormats>.