Discussion Paper / Articulo de Reflexión - Tipo 2

A model of multilayer tiered architecture for big data

Sonia Ordóñez Salinas / sordonez@udistrital.edu.co

Alba Consuelo Nieto Lemus / acnietol@udistrital.edu.co

Universidad Distrital Francisco José de Caldas, Bogotá-Colombia

ABSTRACT Until recently, the issue of analytical data was related to Data Warehouse, but due to the necessity of analyzing new types of unstructured data, both repetitive and non-repetitive, Big Data arises. Although this subject has been widely studied, there is not available a reference architecture for Big Data systems involved with the processing of large volumes of raw data, aggregated and non-aggregated. There are not complete proposals for managing the lifecycle of data or standardized terminology, even less a methodology supporting the design and development of that architecture. There are architectures in small-scale, industrial and product-oriented, which limit their scope to solutions for a company or group of companies, focused on technology but omitting the functionality. This paper explores the requirements for the formulation of an architectural model that supports the analysis and management of data: structured, repetitive and non-repetitive unstructured; there are some architectural proposals –industrial or technological type– to propose a logical model of multi-layered tiered architecture, which aims to respond to the requirements covering both Data Warehouse and Big Data.

KEYWORDS Big data; data warehouse; multi-layered tiered architecture; repetitive structured data; non-repetitive unstructured data, hadoop, mapreduce, noSql.

Un modelo de arquitectura multicapas escalonado para *Big Data*

RESUMEN Debido a la necesidad del análisis para los nuevos tipos de datos no estructurados, repetitivos y no repetitivos, surge Big Data. Aunque el tema ha sido extensamente difundido, no hay disponible una arquitectura de referencia para sistemas Big Data que incorpore el tratamiento de grandes volúmenes de datos en bruto, agregados y no agregados ni propuestas completas para manejar el ciclo de vida de los datos o una terminología estandarizada en ésta área, menos una metodología que soporte el diseño y desarrollo de dicha arquitectura. Solo hay arquitecturas de pequeña escala, de tipo industrial, orientadas al producto, que se reducen al alcance de la solución de una compañía o grupo de compañías, que se enfocan en la tecnología, pero omiten el punto de vista funcional. El artículo explora los requerimientos para la formulación de un modelo arquitectural que soporte la analítica y la gestión de datos estructurados y no estructurados, repetitivos y no repetitivos, y contempla algunas propuestas arquitecturales de tipo industrial o tecnológicas, para al final proponer un modelo lógico de arquitectura multicapas escalonado, que pretende dar respuesta a los requerimientos que cubran, tanto a Data Warehouse, como a Big Data.

PALABRAS CLAVE Big data; data warehouse; arquitectura multicapas escalonada; datos no estructurados repetitivos; datos no estructurados no repetitivos; hadoop; mapreduce; noSql.

Um modelo de arquitetura em camadas empilhadas para *Big Data*

RESUMO A questão da analítica de dados foi relacionada com o Data Warehouse, mas devido à necessidade de uma análise de novos tipos de dados não estruturados, repetitivos e não repetitivos, surge a Big Data. Embora o tema tenha sido amplamente difundido, não existe uma arquitetura de referência para os sistemas Big Data que incorpore o processamento de grandes volumes de dados brutos, agregados e não agregados; nem propostas completas para a gestão do ciclo de vida dos dados, nem uma terminologia padronizada nesta área, e menos uma metodologia que suporte a concepção e desenvolvimento de dita arquitetura. O que existe são arquiteturas em pequena escala, de tipo industrial, orientadas ao produto, limitadas ao alcance da solução de uma empresa ou grupo de empresas, focadas na tecnologia, mas que omitem o ponto de vista funcional. Este artigo explora os requisitos para a formulação de um modelo de arquitetura que possa suportar a analítica e a gestão de dados estruturados e não estruturados, repetitivos e não repetitivos. Dessa exploração contemplam-se algumas propostas arquiteturais de tipo industrial ou tecnológicas, eu propor um modelo lógico de arquitetura em camadas empilhadas, que visa responder às exigências que abrangem tanto Data Warehouse como Big Data.

PALAVRAS-CHAVE *Big Data*; *Data Warehouse*; arquitetura em camadas empilhadas; dados estruturados; dados não estruturados repetitivos; *Hadoop*; MapReduce; NoSql.

Received / Recepción: Mayo 24, 2016 - Accepted / Aceptación: Junio 5, 2016 *doi. 10.18046/syt.v14i37.2240*

I. Introduction

In order to extract aggregated value from new data sources and types, it is necessary that organizations adopt architectures and technologies oriented to the use of *big data* features, widely referenced as the 7 V and 3 C (Bedi, Jindal, & Gautam, 2014; Chandarana & Vijayalakshmi, 2014; Lomotey & Deters, 2014). This entails a new *framework* that includes the definition of strategies for the capture and storage of data; treatment methods considering the features of the different data types, viz. structured, semi-structured, repetitive non-structured, and non-repetitive non-structured; alternative processing methods, data analysis and visualization; handling of scalability; and new practices of maintenance and management of the lifecycle of the system. For this reason, architects must think clearly as to how to handle efficiently all these aspects.

Traditional *data warehouse* and *Business Intelligence* [BI] systems have started to fail, not only due to the large growth of data volume and its types coming from several sources, but also in the need to process, analyze, and present results in real time. Consequently, some methodological alternatives for their development, together with new architectural models that support these challenges, are being pursued.

As an answer to these new requirements, *data warehouse* proposals have arisen, which extend the classic multi-dimensional model to include the treatment of some new data types. Alternatively, some *big data* solutions have appeared that not only entail the analytics perspective, but also look for answers to some of the 7 V and 3 C features, relevant to *big data*. However, the latter neither consider the natural lifecycle of the data nor a methodological and architectonic *framework* that allows – in a systematic way – the solution to typical BI problems. Also, the consideration of some integration between *big data* and *data warehouse* is not present.

In this study, we propose a multilayer-tiered architecture that, besides considering the data nature, allows one to consider the data sources, the frequency of arrival, the storage and processing given the analysis complexity and the opportunity in the information use; the features of the final users; and the possibility to reuse both the data and the analysis algorithms. Additionally, the proposed architecture takes into account the integration of *big data* and *data warehouse*.

In the Section II, we include the challenges in the treatment and storage of the big information volumes. In Section III, we present a review of the *data warehouse* vs. *big data* architectures, the specific architectures proposed for *big data*,

I. Introducción

Para extraer valor agregado de las nuevas fuentes y tipos de datos, es necesario que las organizaciones adopten arquitecturas y tecnologías orientadas al manejo de las características de *Big Data*, referenciadas ampliamente como las 7 V y 3 C (Bedi, Jindal, & Gautam, 2014; Chandarana & Vijayalakshmi, 2014; Lomotey & Deters, 2014). Esto implica un nuevo marco de trabajo que incluye la definición de: estrategias para la captura y el almacenamiento de datos; métodos de tratamiento que consideren las particularidades de los diferentes tipos de datos –estructurados, semi-estructurados, no estructurados repetitivos y no estructurados no repetitivos–; nuevos modelos de representación de datos; métodos alternativos de procesamiento, análisis y visualización de datos; manejo de la escalabilidad; y nuevas prácticas de mantenimiento y gestión del ciclo de vida del sistema. Es por esto que los arquitectos deben pensar claramente cómo manejar eficientemente los aspectos mencionados.

Los sistemas tradicionales de *Data Warehouse* y Business Intelligence [*Data Warehouse*/BI] se han quedado cortos frente, no sólo frente al crecimiento desmedido del volumen de datos de diferentes tipos, provenientes de diversas fuentes, sino también frente a la necesidad de procesar, analizar y presentar resultados en tiempo real. Es así que se buscan alternativas metodológicas para su desarrollo y nuevos modelos de arquitectura que soporten tales retos.

Como respuesta a estos nuevos requerimientos surgen, por un lado, propuestas de *Data Warehouse* que extienden el modelo multidimensional clásico para incluir el tratamiento de algunos nuevos tipos de datos, y por otro, aparecen soluciones de *Big Data* que no sólo contemplan la perspectiva de la analítica, sino que también dan respuesta a algunas de las características 7 V y las 3 C propias de *Big Data*, pero sin considerar el ciclo de vida natural de los datos ni un marco metodológico y arquitectónico que permita de manera sistemática resolver los problemas típicos del BI, y menos aún que considere alguna especie de integración entre *Big Data* y *Data Warehouse*.

En este artículo se propone una *arquitectura multicapas escalonada* que, además de considerar la naturaleza de los datos, permite tener en cuenta las fuentes de proveniencia de los datos, la frecuencia de llegada, el almacenamiento y procesamiento según la complejidad del análisis y la oportunidad de uso de la información, las características de los usuarios finales, así como la posibilidad de la reutilización, tanto de los datos, como de los algoritmos de análisis. La arquitectura propuesta considera adicionalmente la integración de *Big Data* y *Data Warehouse*.

En la segunda sección de este artículo se incluyen los desafíos en el tratamiento y almacenamiento de los grandes volúmenes de información; en la sección 3 se presenta en una revisión de las arquitecturas de *Data Warehouse* vs *Big Data*, las propuestas de arquitecturas específicas para *Big Data*, las herramientas tecnológicas usadas para la adquisición de datos, el almacenamiento, análisis e interfaces, entre otros; en la sección 4 se presenta el modelo conceptual de la arquitectura multicapas escalonada propuesta; y por último, en la sección 5 se incluyen las conclusiones y el trabajo futuro.

II. Desafíos en el tratamiento y almacenamiento de grandes volúmenes de datos

La inteligencia de negocios [BI] está soportada en los *Data Warehouse* que aparecen alrededor de los años ochenta y se basan en el almacenamiento y la integración y consolidación de datos estructurados provenientes de fuentes, generalmente, transaccionales. De otro lado, el término *Big Data* aparece hacia 1997, cuando investigadores de la NASA (Cox & Ellsworth, 1997, p.1) afirmaron que en la visualización científica se debía trabajar con "conjuntos de datos generalmente muy grandes, sobrecargando la capacidad de la memoria principal, del disco local y del disco remoto".

Si bien existen arquitecturas y metodologías ampliamente utilizadas, como las de Kimball, Thorthwaite, Becker, & Mundy (2008), Todman (2001) e Inmon (2005), para el desarrollo de proyectos de *Data Warehouse*, éstas se han quedado cortas a la hora de prever el crecimiento exponencial y la naturaleza cambiante de los datos, pues se requieren grandes esfuerzos para modificar o incluir nuevos requerimientos. Por el contrario, y dado lo nuevo de *Big Data*, aún no hay propuestas estandarizadas para su desarrollo; se presume que será necesario, además de resolver los mismos problemas y desafíos que presentan las arquitecturas y metodologías para la construcción de *Data Warehouse*, considerar en el ciclo de vida del desarrollo los nuevos tipos y características de los datos. Un hecho que motiva el presente análisis, tiene que ver con la forma aislada y no extensible del tratamiento de los datos, tanto en *Data Warehouse*, como en *Big Data*, y por ello la necesidad de proponer un nuevo marco de arquitectura que considere su integración y que a su vez resuelva las limitaciones del uno y del otro.

A. Nuevos tipos de datos

Con el desarrollo de Internet, las comunicaciones inalámbricas y la masificación de dispositivos móviles, se presenta un crecimiento exponencial del volumen de datos y con ello la aparición de nuevas fuentes y tipos de datos. Hoy en día una solución sistematizada que pretenda sacar valor a partir del análisis de los datos debe considerar, no solo las fuentes de datos tradicionales, como bases de datos transaccionales y hojas de cálculo, sino también datos originados por aparatos electrónicos como sensores, cámaras, escáneres y, en general, todo dispositivo que permita controlar las cosas –Internet de las cosas– (Nam, Choi, Ok, & Yeom, 2014).

Las nuevas fuentes generan nuevos tipos de datos, con características muy particulares que dan lugar a la siguiente clasificación:

- Datos Estructurados [DE]: son los datos susceptibles de ser representados mediante estructuras predefinidas (vectores, grafos, tablas, entre otros) y pueden ser gestionados a través de algoritmia matemática computacional. Dado que el comportamiento y la estructura se pueden generalizar, los sistemas tradicionales de bases de datos y bodegas de datos se desarrollaron alrededor de este hecho.

the technological tools used for data gathering, and the storage, analysis, and interfaces, among others. Section IV presents the conceptual model of the multilayered tiered architecture, and Section V concludes the article.

II. Challenges in the treatment and storage of large data volumes

Business intelligence is supported on the data warehouses, which appeared around the 1980s and are based on the storage and integration/consolidation of structured data coming from transactional sources. On the other hand, the term *big data* appeared in 1997, when NASA researchers (Cox & Ellsworth, 1997) declared that in scientific visualization, work should be performed with "extremely large datasets, overheating the capacity of the main memory, local and remote disks".

Even though there are architectures and methodologies that are widely used, such as those in Kimball, Ross, Thorthwaite, Becker, and Mundy (2008), Todman (2001), and Inmon (2005), for the development of *data warehouse* projects, these proposals are now failing, given the fact of the exponential growth and changing nature of the data, since considerable efforts to modify or include new requirements are needed. In contrast, and given the newness of *big data*, there are no standardized proposals for its development. The most probable scenario would be, besides the resolution of the issues in the architectures and methodologies for the *data warehouse* construction, to take into account the new types and features of the data within the development lifecycle. One fact that motivates our analysis is the isolated and non-extensible form of data treatment, both in data warehouse and *big data*. Therefore, the need to propose a new architectural frame considering this integration and solving the limitations of both is a necessity that we attack.

A. New types of data

With the development of the Internet, wireless communications and the widespread growth of the mobile devices, an exponential growth of data volume is present. For this reason, a modern systematized solution that intends to achieve value from the analysis of data should consider, not only traditional data sources (such as transactional databases or spreadsheets), but also data from electronic devices such as sensors, cameras, scanners, and devices capable to control other things (the Internet of Things: Nam, Choi, Ok, & Yeom, 2014).

The new sources generate new data types with very peculiar features; this generates the following classification:

- Structured Data [SD]: these are the data susceptible to be represented with pre-defined structures (i.e. vectors, graphs, tables, etc.) and can be managed through computational mathematic algorithms. Given the fact that the behavior and the structure can be generalized, traditional database systems and data warehouses were developed around this fact.
- Semi-structured Data [SSD]: data with a less rigid structure than SD, generally demarcated with labels indicating the start and end of content. This content is as flexible as required. Generally, these data are represented by markup languages like XML or GML, widely used for information exchange; languages for the management of web pages like HTML and RDF are also in this category. Within this category, data from social networks and emails can be categorized. An email has a defined structure: destination, recipient, subject, content, and attachments. From here, the first two are clearly structured, whilst the latter two are not (as they can be as wide as required).
- Non-Structured Repetitive Data [NSRD]: data with their structure not defined. They occur many times in a time lapse, and some data with similar structure can be found. They are, generally, massive, and not all have a value for the analyses. For this reason, samples or portions can be used. Given the nature of their structure and the frequency of their appearance, algorithms for their treatment and analysis are susceptible to repetition and re-use. Within this category, we find data coming from electronic sensors, whose objective is the analog analysis of a signal with pre-defined algorithms; geo-referenced data, vital signs, seismic movements, positioning of cosmic elements, chemical and biological processes, etc.
- Non-Structured Non-Repetitive Data [NSNRD]: characterized by their different structure, which implies that treatment algorithms are not re-usable. To predict their structure is a complex task. Even though Inmon and Strauss (2008) locate within this category elements of textual nature (Inmon & Linstedt, 2014), i.e., those that require proper techniques of natural language processing and computational linguistics (Manning, 1999), from our perspective, in this category – besides textual information – images, dialog, video, and complex chains are suitable to be classified.

- Datos Semi-Estructurados [DSE]: Son datos con una estructura menos rígida que la anterior, generalmente se demarcan por etiquetas que indican el comienzo y final de un contenido, el contenido es tan flexible como se requiera. Normalmente se representan mediante lenguajes de marcado como XML y GML, entre otros, que han sido ampliamente utilizados para intercambio de información; y lenguajes de gestión de contenido de páginas Web, como HTML y RDF, entre otros. Dentro de esta categoría, además de los datos provenientes de las páginas Web, se pueden incluir los provenientes de aplicaciones de redes sociales y correos electrónicos. Un correo electrónico cuenta con una estructura definida: destinatario, remitente, asunto, contenido y adjuntos; mientras que los dos primeros son claramente estructurados, el contenido y el adjunto no son estructurados, pues son tan amplios y variados como se quiera.
- Datos No Estructurados Repetitivos [DNE-DR]: Son aquellos cuya estructura no está predefinida; ocurren muchas veces en el tiempo; se pueden encontrar algunos datos con estructura similar; generalmente son masivos; no todos tienen un valor para los análisis, por lo que se pueden utilizar muestras o porciones de estos. Por la naturaleza de su estructura y frecuencia de aparición, los algoritmos para su tratamiento y análisis son susceptibles de repetición y reutilización. Dentro de esta categoría caben los datos provenientes de sensores electrónicos, cuyo objetivo es el análisis analógico de la señal, para lo cual hay algoritmos definidos; datos geo-referenciales, signos vitales, movimientos sísmicos, posicionamiento de elementos cósmicos, datos de los procesos biológicos y químicos, entre otros.
- Datos No Estructurados No Repetitivos [DNE-DNR]: Se caracterizan porque tienen diferente estructura, lo que implica que los algoritmos de tratamiento no son reutilizables y el solo hecho de predecir su estructura ya es una tarea compleja. Si bien Inmon y Strauss (2008) ubican dentro de esta categoría los elementos de naturaleza textual (Inmon & Linstedt, 2014), es decir, que requieran de técnicas propias del procesamiento de lenguaje natural y de lingüística computacional (Manning, 1999), desde nuestra perspectiva, en esta categoría, además de la información textual, caben los análisis de: imágenes, diálogo, contenido de video y cadenas.

B. Requerimientos para los nuevos tipos de datos

Con la necesidad de gestionar, analizar, clasificar y agregar los nuevos tipos de datos, surgen nuevos requerimientos como:

- Los *Data Warehouse* tradicionales se caracterizaban por almacenar únicamente la información "útil", es decir aquella requerida para los análisis. Hoy en día, la posibilidad de interactuar con dispositivos (Internet de las Cosas) y con personas a través de la Internet, genera un crecimiento exponencial de la información. La disminución en los costos y el aumento de capacidad de los dispositivos de almacenamiento, permiten guardar grandes volúmenes de información que puede ser o no valiosa para el análisis inmediato o futuro.

- En los *Data Warehouse* clásicos los procesos de extracción y cargue de datos son periódicos y programados, mientras que hoy, la frecuencia de llegada de los datos es variable, puede ser periódica o irregular y con flujos controlados o ráfagas de información. Dado que *Big Data* se ha concebido para recibir toda la información que llegue y en cualquier frecuencia, es necesario destinar la cantidad de memoria, almacenamiento y procesamiento que se requiera.
- Los *Data Warehouse* clásicos se crearon bajo la premisa de guardar información histórica, "útil", no volátil y relacionada lógicamente. Hoy es necesario considerar la longevidad, la frecuencia y la oportunidad de uso de los datos: los generados más recientemente serán usados con mayor frecuencia y en tiempo real y, en la medida en que envejecen, su frecuencia de uso puede decrecer, pero no se pueden descartar, ya que para los análisis históricos es necesario contar con ellos en cualquier momento (Inmon, 2014; Kimball, 2012).
- Si bien los *Data Warehouse* tradicionales tenían como propósito integrar los datos a través del modelo multidimensional, con la aparición de los datos no estructurados repetitivos (DNE-DR) el primer problema que se encuentra es agrupar los datos dentro de un mismo contexto, independientemente del tipo de dato: por ejemplo, agrupar imágenes con diálogos, encontrar la estructura que mejor represente a todos los datos y hacer los algoritmos que permitan integrar, transformar y analizar.
- La filosofía de los *Data Warehouse* tradicionales sólo considera datos estructurados repetitivos (DE-DR), por lo que se facilitan, no sólo los procesos de Extracción, Transformación y Cargue (ETL), sino también el análisis. Con los datos no estructurados y no repetitivos (DNE-DNR), además de identificar el contexto y la estructura que represente los datos, se puede requerir de un algoritmo particular para cada dato, lo que dificulta los procesos ETL y aumenta la complejidad en el análisis.
- La integración de los datos se constituye como la mayor diferencia entre *Data Warehouse* y *Big Data*, pues para el primero de ellos su objetivo y filosofía radica en integrar para tener una visión global de la organización, mientras que en *Big Data* la integración no es el fin último. Para *Big Data*, algunos datos no estructurados y no susceptibles de integración deben mantenerse en bruto, lo que permite, no solo la oportunidad de acceso, sino la posibilidad de ser procesados por los expertos en ciencias de los datos.

III. Arquitectura de data werehouse Vs. big data

Los *Data Warehouse* tradicionales surgen con el objetivo de integrar y agregar datos estructurados para contar con información histórica que soporte el procesamiento analítico en línea, y apoyar la inteligencia de negocios (Inmon & Strauss,

B. Requirements for new data types

In order to manage, analyze, classify, and aggregate the new data types, new requirements arise:

- Traditional data warehouses were characterized by only storing the "useful" information, i.e., that related to the analyses. Nowadays, the possibility to interact with devices (e.g. the Internet of Things) and with people through the Internet, generates an exponential growth of information. The reduction in the costs and the increase in the storage capacity of the devices allows for the handling of large information volumes that can or cannot be useful for immediate or future analysis.
- In classic data warehouses, the data extraction and loading processes are periodic and programmed, whilst today, the input data frequency is variable; it might be periodic or irregular and with controlled flows or information bursts. Given the fact that *big data* was conceived to receive all the incoming information at any frequency, it is important to designate the memory, storage, and processing capabilities as required.
- Classic data warehouses were created to store historic, "useful", non-volatile, and logically related information. Today, it is indispensable to consider the longevity, frequency, and opportunity of using the data: the most recent data will be used more frequently and in real time; when they become "old", their usage frequency might be reduced, but they cannot be discarded. Because of historical analysis, its use at any time is required (Kimball, 2012; Inmon, 2014).
- Even though traditional data warehouses had as their main purpose the integration of the data through the multi-dimensional model, with the appearance of non-structured repetitive data [NSRD], the first issue is the gathering of the data within the same context, regardless of the data type. For instance, group images with dialogs find the structure that best represents all the data and create the algorithms that allow integration, transformation, and analysis.
- Traditional data warehouse philosophy only considers repetitive structured data [RSD]; hence, not only the extraction, transformation, and load [ETL] processes, but also the analyses are facilitated. With non-structured non-repetitive [NSNR] data, besides the identification of the context and the structure representing the data, a particular algorithm for each piece of data

might be required, which increases the difficulty of the ETL processes and the analysis complexity.

- Integration of the data is the most important difference between the data warehouse and the *big data* approaches, since, in the former, its objective and philosophy rely on integration to obtain a global view of the organization; while in the latter, integration is not the ultimate goal. For *big data*, some non-structured and non-susceptible data must be maintained "on raw"; this allows not only the access opportunity, but also the possibility to be processed by experts in data sciences.

III. Data warehouse Vs. big data architecture

Traditional *data warehouses* arose with the objective to integrate and aggregate structured data to obtain historic information that supports online analytic processing and the business intelligence and decision making (Inmon & Strauss, 2008; Kimball, 2011). The architecture of these warehouses consists of a layered model with the ETL processes, and storage in predefined multi-dimensional structures as a star (Todman, 2001; Kimball et al., 2008); and flakes (Todman, 2001; Inmon, 2005); and data analysis processes supported ad hoc queries, control boards, trends and patterns reports, among others.

On the other hand, *big data* and the new generation of data warehouses do not have predefined models, are not supported by client-server architectures, and must support horizontal scaling. New data warehouses and *big data* provide the solution to the large data collection management through the MapReduce programming model (Díaz, 2011), which allows the parallelization of the process to gather partial results; all of this based on distributed file systems such as the *Hadoop* Distributed File System [HDFS].

A. Data Warehouse architecture

Models conceived for data warehouse are implemented over relational databases [Relational Online Analytical Processing – ROLAP] and are managed through the Structured Query Language [SQL (Maiorescu, 2010). Less frequent are the implementations under Multidimensional Online Analytical Processing [MOLAP]. Despite the fact that traditional data warehouses manage large volumes of information, their architecture has been supported in client-server models that only can be scaled in a vertical way. This entails major technological and economic efforts both in their development and in maintenance.

2008; Kimball, 2011) y la toma de decisiones. La arquitectura de éstas bodegas, generalmente, consiste en un modelo de capas que comprende: los procesos de Extracción, Transformación y Carga (ETL); el almacenamiento en estructuras multidimensionales predefinidas, como estrella (Kimball et al., 2008; Todman, 2001) y copo de nieve (Todman, 2001; Inmon, 2005); y procesos de análisis de datos soportado en consultas ad-hoc, tableros de control, reportes de tendencias y patrones, entre otros.

Por el contario, *Big Data* y la nueva generación de *Data Warehouse* no cuentan con modelos predefinidos, no se apoyan sobre arquitecturas cliente-servidor y deben soportar el escalonamiento horizontal. Los nuevos *Data Warehouse* y *Big Data* dan solución a la gestión de grandes colecciones de datos mediante el modelo de programación MapReduce (Díaz, 2011) que permite paralelizar el proceso para luego reunir los resultados parciales; todo ello soportado en sistemas distribuidos de archivos como *Hadoop* Distributed File System [HDFS].

A. Arquitectura para Data Warehouse

Los modelos concebidos para *Data Warehouse* son implementados sobre bases de datos relacionales [Relational Online Analytical Processing - ROLAP] y gestionados a través de lenguaje Structured Query Language [SQL] (Maiorescu, 2010). Aun cuando menos frecuente, también se encuentran implementaciones bajo esquemas multidimensionales [Multidimensional Online Analytical Processing, MOLAP]. A pesar de que los *Data Warehouse* tradicionales gestionan grandes cantidades de información, su arquitectura se ha soportado en modelos cliente-servidor que solamente pueden ser escalados de manera vertical, lo que acarrea grandes esfuerzos tecnológicos y económicos, tanto en su desarrollo, como en su mantenimiento.

La construcción del *Data Warehouse*, así como la exploración de la información residente en la bodega, incluye los procesos críticos de ETL que deben considerar gran cantidad de datos duplicados, posibles inconsistencia de datos, alto riesgo en la calidad de los mismos y la utilización de diferentes herramientas para los distintos procesos (ETL + exploración), lo cual puede conllevar a una metadata no compartida, resultados inconsistentes, rígidos modelos de datos relacionales o multidimensionales, y con ello la falta de flexibilidad para hacer análisis y cambios (Muntean & Surcel, 2013).

Para solucionar los retos a los que se enfrentan los *Data Warehouse*, se propone la utilización de la memoria extensiva. Dicho almacenamiento puede generar ganancia en el acceso a los datos, disminuir el tiempo de respuesta y, dependiendo de su arquitectura, permitir integrar distintas fuentes y eliminar redundancia. A partir del almacenamiento en memoria se encuentran propuestas para los procesos ETL, para el almacenamiento y la estructuración. Dentro de éstas vale la pena citar algunas como: los procesos ETL y el almacenamiento y estructura de datos, descritos a continuación.

Procesos ETL

A diferencia de las arquitecturas tradicionales donde los proceso ETL son en batch, programados y periódicos, YiChuan y Yao (2012) proponen, adicional a dichos procesos ETL en lote,

un subsistema de ETL en tiempo real que directa y automáticamente detecta los cambios en la fuente de datos y los carga dentro del área de almacenamiento, mediante herramientas de Change Data Capture [CDC]: cuando el sistema identifica que se cumplen ciertas condiciones, los datos son cargados en batch dentro de la bodega. La parte almacenada se puede dividir entonces, en área en tiempo real y área estática; las consultas se hacen sobre el almacenamiento en tiempo real y los datos estáticos equivalen al *Data Warehouse*, en donde se realizan las consultas históricas.

Dentro de las propuestas que remplacen totalmente la carga en lotes, está por ejemplo la de Agrawal (2009) quien introduce el componente middleware, conocido como el motor de análisis de flujo. Este motor realiza una exploración detallada de los datos entrantes antes de que puedan ser integrados en la bodega, para así identificar los posibles patrones y los valores atípicos.

Almacenamiento y estructura de datos

Adicional a las estructuras clásicas multidimensionales, se han propuesto otras que organizan la información en memoria de forma diferente al tradicional modelo relacional o que utilizan bases de datos NoSql. Se pueden citar por ejemplo: Microsoft PowerPivot (Muntean & Surcel, 2013), que organiza, analiza y almacena los datos en forma tabular, como si fuese una hoja de cálculo, en donde los datos son representados como celdas de un arreglo organizado en filas y columnas; QLinkView, originalmente llamada QuikView [Quality Understanding, Interaction, Knowledge], que carga y almacena todos los datos en un modelo de arreglos de tipo asociativo de datos [AQL Associative Query Logic] que permite realizar las uniones (joins) y los cálculos en tiempo real (Plattner, 2009; Schaffner, Bog, Krüger, & Zeier, 2009).

B. Arquitectura para Big Data

Big Data ha motivado el surgimiento de nuevas tecnologías para el almacenamiento, como *Hadoop* (Cuzzocrea, 2014) y bases de datos NoSQL (Vaish, 2013), que además de soportar estructuras dinámicas y datos no estructurados, permiten un escalonamiento horizontal y garantizan la velocidad, la variabilidad y el volumen de datos. Sin embargo, más allá del aspecto tecnológico, el manejo de las características de *Big Data* requiere un nuevo marco de trabajo que incluya la definición de estrategias para la captura y el almacenamiento de datos, métodos de tratamiento que consideren las particularidades de los diferentes tipos de datos (estructurados, semi-estructurados, no estructurados repetitivos y no estructurados no repetitivos), nuevos modelos de representación, métodos alternativos de procesamiento, análisis y visualización de datos, manejo de la escalabilidad, nuevas prácticas de mantenimiento y gestión del ciclo de vida del sistema. Es por esto que se debe pensar claramente cómo manejar eficientemente las 7V y las 3 C, a lo largo de toda la arquitectura del sistema.

Pese a que el tema ha sido ampliamente difundido, no hay disponible una arquitectura de referencia que gestione todo el ciclo de vida de los datos en *Big Data*; tampoco propuestas completas en la literatura (Bedi et al., 2014), ni una terminología

The construction of the data warehouse and the exploration of information in the hold include ETL critical processes, where a large volume of duplicated data has to be considered, together with a possible inconsistency of the data, high risk in their quality, and the use of different tools for the processes (ETL + exploration). This might entail non-shared metadata, inconsistent results, and rigid relational/multidimensional data models, causing a lack of flexibility for analysis and changes (Muntean & Surcel, 2013).

In order to solve the challenges of data warehouses, we propose the use of extensive memory. This storage can increase the efficiency in data access, reduce the response time, and – depending on the data architecture – allow the integration of several sources and eliminate redundancy. From the in-memory storage, we found proposals for the ETL processes related to storage and structuration. Within these, we cited as important the ETL processes and the data storage and structure, described in the following sections.

ETL processes

In contrast to traditional architectures, where the ETL processes are batched, programmed, and periodic, YiChuan and Yao (2012) propose – in addition to those batched ETL processes – an ETL subsystem in real time that directly and automatically detects the changes in the data source, loading them in the storage area through Change Data Capture [CDC] tools. This is, when the system identifies that certain conditions are satisfied, the data are loaded in batch inside the warehouse. The stored part can be split into real time and static areas; the queries are made in the real-time storage and the static data equal to the data warehouse, where the historic queries are performed.

Within the proposals to replace the load in batches totally, Agrawal (2009) introduces the middleware component, best known as the flow analysis engine. This engine performs a detailed exploration of the entering data before delivering them in the warehouse, to identify possible patterns and atypical values.

Storage and data structure

In addition to the multidimensional structures, other ones that organize the information in memory different to the traditional model and use NoSql databases are proposed. Examples of these are Microsoft PowerPivot (Muntean & Surcel, 2013), which organizes, analyzes, and stores the data in a tabular way, like a spreadsheet, where the data are represented as cells of an organized array in rows and columns; QLinkView, originally called QuikView [Quality Unders-

tanding, Interaction, Knowledge], which loads and stores all the data in a data-associative array model [AQL, Associative Query Logic] that allows joints and real time calculations to be performed (Plattner, 2009; Schaffner, Bog, Krüger, & Zeier, 2009).

B. Big data architecture

Big data has motivated new storage technologies like *Hadoop* (Cuzzocrea, 2014) and NoSql databases (Vaish, 2013). These support both dynamic structures and non-structured data, allow horizontal scaling, and guarantee data speed, variability, and volume. Nevertheless, beyond the technologic aspect, the handling of *big data* features requires a new *framework* that includes the definition of strategies for the data capture and storage, treatment methods considering the particularities of the different data types (see Section II.A), new representation models, alternative processing methods, data analysis and visualization, scalability, new management practices, and management of the system life-cycle. For this reason, clear and efficient management of the 7 V and 3 C is relevant through all the system architecture.

Although the topic has been widely studied, there is no reference architecture that manages all the data lifecycle in *big data*; nor any full proposals in the literature (Bedi et al., 2014), nor a standardized terminology in this area (Demchenko, Laat, & Membrey, 2014). What we have available are small scale architectures that are industrial type and product-oriented, reduced to the reach the solution for a company or group of companies; also, focused on the technology but omitting the functional point of view. For example, the HP Reference Architecture for MapR M5 (Hewlett Packard, 2013), which, in other words, is a document describing the *Hadoop* modules incorporated in MapR as its implementation over a Hewlett-Packard data warehouse, rather than a real reference (Bedi et al., 2014).

C. Revision of big data architectures

Other studies present several approaches of architectures from conceptual models to reference *frameworks*, defining the model components and their associated tools. We briefly describe some of these architectural models.

Conceptual model for big data components

Given the specification, classification, and relation of the associated requirements with 5 of the 7 V (volume, velocity, variety, veracity, and value) of *big data*, Bedi et al. (2014) propose a reference architecture. This proposal is based on layers (see **Figure 1**) that allow the acquisition, cleaning, integration, modeling, and data analysis, besides components

estandarizada en ésta área (Demchenko, Laat, & Membrey, 2014). Lo que hay son arquitecturas de pequeña escala, de tipo industrial, orientadas al producto, que se reducen al alcance de la solución de una compañía o grupo de compañías, y que se enfocan en la tecnología, pero omiten el punto de vista funcional. Tal es el caso de HP Reference Architecture for MapR M5 (Hewlett Packard, 2013), que más bien es un documento que describe los módulos de *Hadoop* incorporados en MapR y su implementación sobre un *Data Warehouse* de Hewlett-Packard, que un verdadero marco de referencia (Bedi et al., 2014).

C. Revisión de arquitecturas para Big Data

La literatura presenta diferentes aproximaciones de arquitecturas que van, desde modelos conceptuales, hasta marcos de referencia, que definen los componentes del modelo y las herramientas asociadas. A continuación se describen brevemente algunos de estos modelos de arquitectura.

Modelo conceptual de componentes para Big Data

A partir de la especificación, clasificación y relación de los requerimientos asociados con 5 de las 7 V (volumen, velocidad, variedad, veracidad y valor) de *Big Data*, Bedi et al., (2014) proponen una arquitectura de referencia. Dicha arquitectura se basa en capas (ver **Figura 1**) que permiten la adquisición, limpieza, integración, modelamiento y análisis de datos, y componentes para: identificación de datos, extracción de datos, flujo de datos, extracción de información, manejo de la calidad de datos, integración de datos, análisis de datos y distribución de datos; además de componentes transversales para el almacenamiento de datos, manejo de la metadata, manejo del ciclo de vida y seguridad. Adicionalmente, el modelo propuesto describe el papel que desempeñan algunas tecnologías como *Hadoop*, MapReduce y bases de datos NoSQL en la implementación del modelo de referencia. Este modelo, a pesar de identificar claramente cada una de las tareas y los procesos en las diferentes capas de la arquitectura, así como la necesidad de contar con un componente que permita gestionar el ciclo de vida de los datos, no prevé almacenamientos según la naturaleza y el ciclo de vida de los datos.

Modelo de componentes para un ecosistema Big Data

El conjunto complejo de herramientas técnicas y componentes de infraestructura construido alrededor de los datos y de su uso es lo que se denomina Ecosistema de *Big Data*. Demchenko et al., (2014) proponen un marco de arquitectura llamado *Big Data*AF (*Big Data* Architecture Framework) que considera los principales elementos del ecosistema: el manejo de las 5´Vs (volumen, variedad, velocidad, valor y veracidad), además de otros aspectos como los nuevos modelos de datos; origen, ciclo de vida y evolución de los datos; análisis en tiempo real, análisis de flujo de datos, máquinas de aprendizaje, herramientas e infraestructura (almacenamiento, redes, infraestructura en la nube, integración); fuente, destino y estructura de los datos. El *framework Big Data*AF define cinco componentes principales de la arquitectura que incluyen los principales aspectos de *Big Data*:

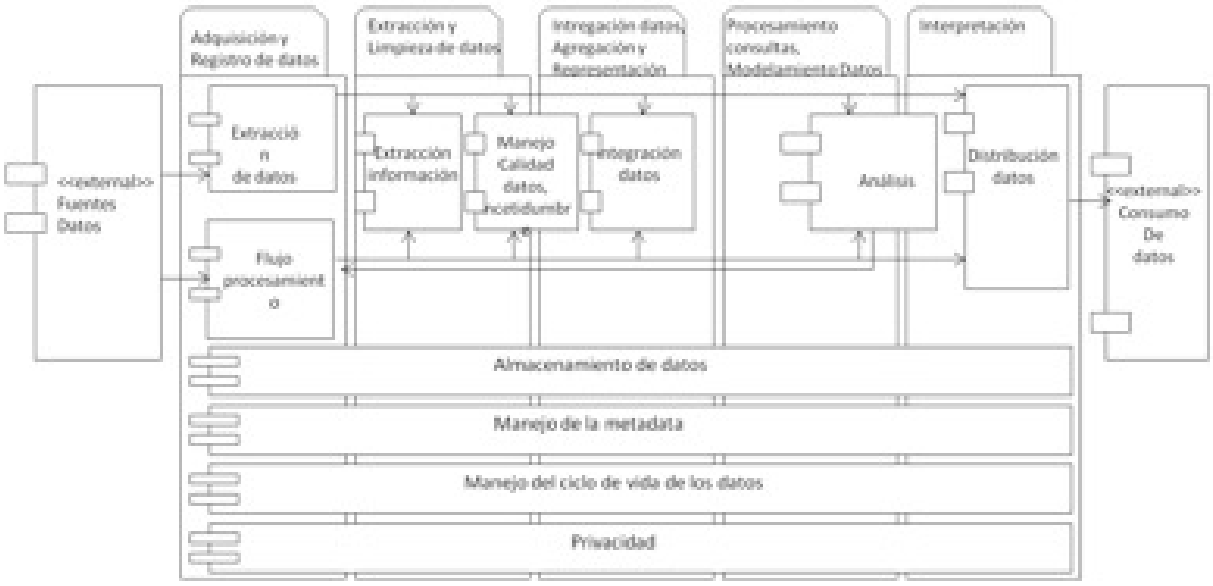

Figure 1. Conceptual components model for a big data architecture /
Modelo conceptual de componentes para una arquitectura de Big Data (Bedi et al., 2014)

- modelos, estructuras y tipos, tanto para datos estructurados, como no estructurados;
- manejo del ciclo de vida de los datos (proveniencia, almacenamiento, transformación);
- herramientas de análisis y presentación de los datos;
- infraestructura (servicios de cluster, *Hadoop* y herramientas relacionadas, herramientas de análisis, servidores de bases de datos, procesamiento paralelo); y
- seguridad (en los datos, en el movimiento de los datos, en el ambiente de exploración y procesamiento).

Este modelo, aún cuando se presenta como una propuesta altamente escalable, considera el ciclo de vida de los datos de una manera genérica, sin tener en cuenta la longevidad, la frecuencia de llegada y la oportunidad de uso e integración de datos de diferente naturaleza.

Modelo de arquitectura empresarial para Big Data

Oracle (2015) propone un modelo de arquitectura desde el punto de vista empresarial que se construye a partir de los requerimientos a los que se enfrentan los arquitectos para definir la visión de la arquitectura: de dónde provienen los datos; cuál es la tecnología apropiada para el almacenamiento; cuál es la estrategia más práctica de procesamiento; cómo maximizar la velocidad de las consultas ad-hoc, la transformación de los datos y el procesamiento analítico; cómo minimizar la latencia entre los diferentes componentes del sistema; dónde y cómo hacer análisis de datos (al ingresar los datos, al almacenarlos, en una bodega de datos, en la nube); y dónde proveer seguridad sobre los datos. Las respuestas a estas preguntas incluyen condiciones de *Data Warehouse* (redes, servidores, almacenamiento, sensores, memoria), modelos de persistencia (relacional, sistema de archivos *Hadoop* Distributed File System (HDFS), NoSQL, *Data Warehouse*) y técnicas de procesamiento (en tiempo real, distribuido, paralelo), entre otras.

for: data identification, extraction, and flow; information extraction; handling data quality; data integration, analysis, and distribution. Besides the transverse components for data storage, metadata handling, and lifecycle and security handling. Additionally, this model describes the role that some technologies like *Hadoop*, MapReduce, and NoSQL databases have in its implementation. This proposal, regardless of the clear identification of each one of the tasks and processes in the different layers of the architecture, and besides the need to consider a component that allows the management the lifecycle of the products, does not provide storage given the nature and lifecycle of the data.

Components model for a big data ecosystem

The complex set of technical tools and infrastructure components built within the data and its use is called the *big data* ecosystem. Demchenko et al. (2014) propose an architecture frame called *Big Data*AF [*Big Data* Architecture Framework] that considers the main elements of the ecosystem: the handle of the 5V and other aspects and the new data models; origin, lifecycle, and evolution of the data; real time analysis, data flow analysis, learning machines, tools and infrastructure (i.e. storage, networks, cloud infrastructure, integration); source, destination; and data structure. The *Big Data*AF *framework* defines five main components of the architecture, which include the main *big data* aspects:

- models, structures, and types, both for structured and non-structured data;
- handle of the data lifecycle (origin, storage, transformation);

- tools for the analysis and presentation of the data;
- infrastructure (i.e. cluster services, *Hadoop*, and related tools, analysis tools, database servers, parallel processing); and
- security (in data movement and in the exploration and processing environment).

This model, even though a highly scalable proposal, considers the data lifecycle in a generic way without considering the longevity, the arrival frequency, and the use and integration opportunity of data of different nature.

Corporate architecture model for big data

Oracle (2015) proposes an architectural model from the corporate point of view, built through the requirements the architects face to define the vision of the architecture: where the data comes from; which is the appropriate technology for the storage; which is the most practical processing strategy; how to maximize the speed of the ad hoc queries, of the data transformation, and the analytic process; how to minimize the latency between the system components; where and how to perform data analysis (e.g. in the moment they are entered, stored, in a warehouse, in the cloud); and where to provide data security. The answers to those questions include data warehouse conditions (networks, servers, storage, sensors, memory), persistence models (relational, HDFS, NoSQL, data warehouse), and processing techniques (real time, distributed, parallel) among others.

The architectonic vision presented in Oracle (2015) is aligned with the corporate architecture model of the organization, where the business processes, the data quality and access, security, the available infrastructure, and the governmental practices of the data are considered. The logical architecture model includes the data deposit layers, the exploration and discovery laboratories, and staging progression layers (intermediate storage zones) to protect the raw data and allow access to the consumers needing it.

The platform of this proposal is defined by an ecosystem of tools to acquire, organize, analyze, and visualize the data. It provides proprietary tools, but also allows one to integrate free software solutions (e.g. Apache, *Hadoop*, MapReduce, Cloudera, etc.) and the proprietary solutions of rivals. Even though the different data types and aspects of business management are considered, it is a proposal that is highly focused on a set of products more than in the methodology and management of the lifecycle of *big data* projects.

La visión arquitectónica presentada en Oracle (2015) está alineada con el modelo de arquitectura empresarial de la organización, en el que se consideran los procesos de negocio, la calidad y el acceso a los datos, la seguridad, la infraestructura disponible y las prácticas de gobernanza de los datos. El modelo lógico de arquitectura incluye la capa de depósito de datos, los laboratorios de exploración y descubrimiento, y una progresión de capas de staging (zonas intermedias de almacenamiento de datos) para proteger los datos en bruto y permitir a los consumidores acceder a los datos que necesiten.

La plataforma de ésta propuesta está conformada por un ecosistema de herramientas para adquirir los datos, organizarlos, analizarlos y visualizarlos; provee herramientas propietarias, pero también permite integrar soluciones de software libre (Apache, *Hadoop*, MapReduce, Cloudera, entre otras) y soluciones propietarias de terceros (competidores). Aunque se consideran los diferentes tipos de datos y aspectos de gestión empresarial, es una propuesta altamente enfocada en un conjunto de productos, más que en la metodología y gestión del ciclo de vida de proyectos *Big Data*.

Revisión de tecnologías que soportan la arquitectura de Big Data

Dentro de las arquitecturas propuestas para el manejo Big-Data, Chandarana y Vijayalakshmi (2014) hacen una recopilación de algunas de las herramientas más utilizadas, dentro de cuales incluyen: Apache *Hadoop*, Apache Drill y Apache Storm. Además de las anteriores, vale pena relacionar otras dos: Apache Spark y Apache Hive.

Apache Hadoop

Hadoop es un proyecto de código abierto de la fundación Apache. Es una aplicación diseñada para soportar el almacenamiento y procesamiento de grandes conjuntos de datos a través de múltiples sistemas en clúster. Es ampliamente utilizada, tanto para almacenamiento como para procesamiento de grandes cantidades de información, y ha sido ampliamente probada en el campo de *Big Data*. Dentro de las investigaciones se pueden citar los trabajos de: Manikandan y Ravi (2014), Nandimath, Banerjee, Patil, Kakade, y Vaidya (2013), Katal, Wazid, y Goudar (2013); Pal y Agrawal (2014), y Zhang, Hildebrand, y Tewari (2014). Algunos de los componentes del ecosistema de *Hadoop* (Welcome to..., n.d) son:

Hadoop Distributed File System HDFS

Es un sistema de archivos que proporciona un alto rendimiento en el acceso a los datos de una aplicación. Además de ser escalable, ofrece alta disponibilidad y tolerancia a fallos mediante la replicación (HDFS..., 2013). El sistema se organiza en clusters; un cluster consta de varios nodos que almacenan los datos. En cada cluster hay dos tipos de archivos: el namenode y datanode. Sólo hay un namenode por cluster el cual contienen la metadata de los directorios y los archivos del cluster; es el encargado del controlar el acceso a los archivos por parte de los clientes del sistema HDFS. El datanode, almacena el contenido de un archivo en forma de bloques. Cada bloque está replicado en otros nodos (datanodes), para asegurar la toleran-

cia a fallos. Cuando un cliente requiere acceder a los datos de un cluster envía una petición al namenode, el cual lista los datanodes donde están los bloques solicitados (ver **Figura 2**).

Hadoop YARN [Yet Another Resource Negotiator]

Es un módulo dedicado a la planificación de tareas y gestión de los recursos, que aparece a partir de la versión 2.0 de *Hadoop*. Separa las funcionalidades: gestión de recursos y monitoreo, en demonios separados, y permite que diferentes tipos de aplicaciones de MapReduce se puedan ejecutar en el cluster.

Hadoop Map Reduce

Es una implementación del modelo de programación MapReduce para procesar en paralelo de grandes conjuntos de datos, consta de dos funciones: la función Map, para transformar un conjunto de datos en parejas clave-valor, y la función Reduce, para combinar los valores en un mismo resultado (ver **Figura 3**).

Apache Pig

Es una plataforma para analizar grandes conjuntos de datos. Incluye un lenguaje de alto, que permite expresar programas de análisis, y la infraestructura para la evaluación de los mismos. La infraestructura de Pig consiste de un compilador que produce secuencias de programas MapReduce. En este nivel se incluye el lenguaje PigLatin que se caracteriza por permitir programar en paralelo, optimizar la ejecución en forma automática y crear funciones propias. PigLatin, crea estructuras tipo SQL, de manera que, en lugar de escribir aplicaciones separadas de MapReduce, se pueda crear un script de PigLatin el cual es automáticamente paralelizado y distribuido a través de un cluster.

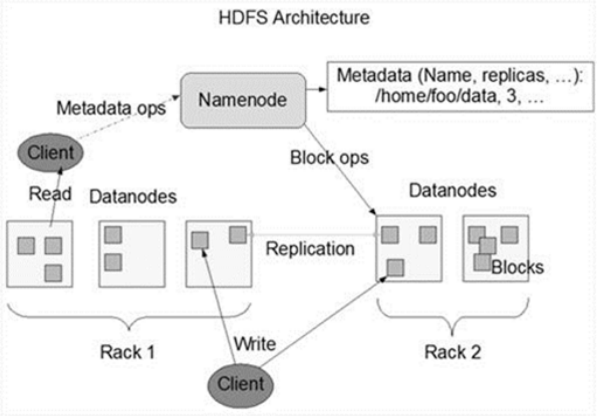

Figure 2. Hadoop Distributed File System /
Sistema distribución de archivos de Hadoop - HDFs (HDFS, 2013)

Apache Drill

Apache Drill fue inspirado por Dremel de Google; es un sistema distribuido para el análisis interactivo de grandes conjuntos de datos. Permite distribuir los datos (estructurados, semi-estructurados y anidados) en miles de servidores; el objetivo de Drill es responder a las consultas ad-hoc en un modo de baja latencia y alta velocidad. El núcleo de Apache Drill es el servicio Drillbit (Architecture..., n.d), responsable de aceptar las peticiones del cliente, el procesamiento de las consultas y

Review of technologies supporting the big data architecture

Within the proposed architectures for *big data* handling, Chandarana and Vijayalakshmi (2014) made a collection of some of the more used tools, including Apache *Hadoop*, Drill, and Storm. Besides of these, it is important to mention another two: Apache Spark and Hive.

Apache Hadoop

Hadoop is an open-source project of the Apache foundation. It is an application that is designed to support the storage and processing of large datasets through multiple systems in clusters. It is widely used, both for storage and processing of large information amounts, and has been widely tested in the *big data* field, e.g., Manikandan and Ravi (2014); Nandimath, Banerjee, Patil, Kakade, and Vaidya (2013); Katal, Wazid, and Goudar (2013); Pal and Agrawal (2014); and Zhang, Hildebrand, and Tewari (2014).

Hadoop Distributed File System [HDFS]

This is a file system that provides high performance in the access to the data of an application. Besides being scalable, it offers high availability and fault tolerance using the replication (HDFS, 2013). The system is organized into clusters where one consists of several nodes storing the data. On each cluster, there are two file types: namenode and datanode. There is only one namenode per cluster, which contains the metadata of the directories and files of the cluster. It is also in charge of controlling the access to the files from the HDFS system clients. The datanode stores the content of a file in the form of a block. Each block is replicated in another node to ensure fault tolerance. When a client requires access to the data in a cluster, it sends a request to the namenode, which lists the datanodes where the requested blocks are available (**Figure 2**).

Hadoop YARN [Yet Another Resource Negotiator]

This is a module dedicated to tasks planning and resource management that appears from the version 2.0 of *Hadoop*. It splits the resource management and monitoring functionalities in separated daemons, allowing that different MapReduce application types can be executed in the cluster.

Hadoop Map Reduce

This is an implementation of the MapReduce programming model designed to process in parallel large data amounts. It has two functions: the Map function to transform a set of data in key-value pairs, and the reduce function to combine the values in the same result (**Figure 3**).

Apache Pig

This is a platform to analyze large datasets. It includes a high-level language that allows the expression of analysis programs and the infrastructure for the evaluation of the same. Its infrastructure consists of a compiler that produces MapReduce program sequences. In this level, the PigLatin language is included, which is characterized by allowing parallel programs, by the optimization in the automatic execution, and the creation of proper functions. PigLatin creates SQL structures; hence, instead of writing MapReduce separated applications, a script in PigLatin is created and automatically parallelized and distributed in a cluster.

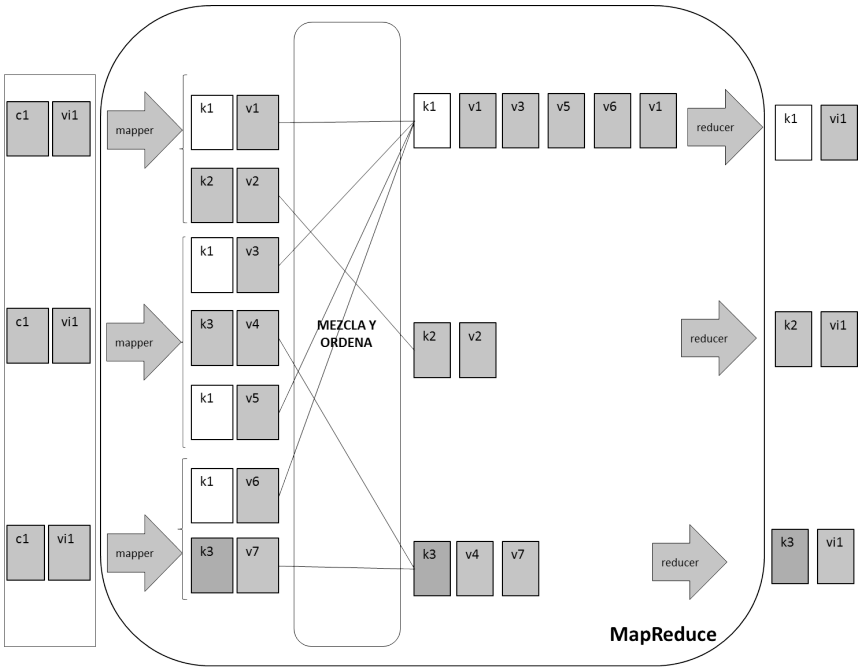

Figure 3. Map Reduce model scheme / Esquema del modelo Map Reduce (Díaz-Zorita, 2011)

Apache Drill

Apache Drill was inspired by Dremel of Google. It is a distributed system for the interactive analysis of large datasets. It allows the distribution of the data (structured, semi-structured, and nested) into thousands of servers; its objective is to respond to ad hoc queries in low-latency and high-speed modes. The core of Apache Drill is the Drillbit service (Architecture..., n.d), in charge of accepting the client requests, processing the consults, and returning the results to the client. Drillbit does not work under the client-server model; it can be installed and executed in all the necessary nodes in a *Hadoop* cluster to form a distributed environment. Despite the fact that Drill can operate in a *Hadoop* cluster environment, it is not linked to *Hadoop* and can be executed on any distributed cluster environment.

Apache Storm

Apache Storm is a free and real-time system that can be integrated with databases. Its processing is based on data flows distributed in relation to the necessities. Given the literature (Marz, n.d), it is fault-tolerant and scalable. It guarantees the processing of more than one million tuples per second per node.

Storm has the possibility to perform real time analyses, online machine learning and distributed Remote Procedure Call [RPC] from different sources. Through an API, the tuple flows can be managed, where each tuple is a named list of values. Storm handles three abstractions: spouts, bots, and topologies. The first are those that gather the data in the data source; the bots process these data flows.

devolver los resultados al cliente. Drillbit no trabaja bajo el concepto cliente-servidor, se puede instalar y ejecutar en todos los nodos necesarios en un cluster *Hadoop* para formar un entorno distribuido. Aunque Drill puede trabajar en un entorno de cluster *Hadoop*, no está ligado a *Hadoop* y se puede ejecutar en cualquier entorno de clúster distribuido.

Apache Storm

Apache Storm es un sistema libre y en tiempo real que puede integrarse con bases de datos. Su procesamiento se basa en flujos de datos que son distribuidos de acuerdo con las necesidades. Según la literatura (Marz, n.d), es tolerante a fallos y escalable. Garantiza procesar más de un millón de tuplas por segundo por nodo.

Storm cuenta con la posibilidad de realizar análisis en tiempo real, aprendizaje de máquina en línea y comunicación entre procesos (Remote Procedure Call, RPC) distribuida, a partir de diferentes fuentes. A través de un API se pueden gestionar los flujos de tuplas, donde cada tupla es una lista nombrada de valores. Storm maneja tres abstracciones: grifos (spouts), rayos (bots) y topologías. Los spouts son los encargados de recoger los datos en la fuente de datos y los bots los encargados de procesar los flujos de datos.

Apache Spark

Surge como una alternativa a la tecnología *Hadoop*, pero a diferencia de Haddop, que utiliza la secuencia Map-Shuffle-Reduce para el procesamiento de los datos, utiliza el *framework* Tez que incluye un grafo acíclico dirigido para la secuencia de ejecución, el cual permite incluir secuencias como Map -Shuffle - Reduce - Shuffle – Reduce. Trabaja principalmente en memoria y permite definir con prioridades los datos que deben residir en memoria.

Apache HIVE

Apache Hive (n.d) es un software que tiene origen en Facebook. Fue construido como parte de *Hadoop*, pero ahora funciona de forma independiente bajo la Fundación Apache. Permite la gestión del almacenamiento distribuido de grandes volúmenes de datos y con variados formatos. Puede gestionar el almacenamiento de datos a través de archivos HDFS y bases como Hbase. Utiliza el lenguaje de consulta llamado HiveQL que ejecuta las consultas vía MapReduce para posibilitar el análisis de datos. Permite que el mismo usuario defina cuándo aplicar las funciones Map y Reduce. Incluye funciones personalizados para escalares [UDF], agregaciones [UDAF] y tablas [UDTF]. Hive no está diseñado para cargas de trabajo OLTP y no ofrece consultas en tiempo real o actualizaciones a nivel de fila. Se utiliza para trabajos por lotes de grandes conjuntos de datos.

Otras herramientas

Adicional a las mencionadas se encuentran otras herramientas como:

- SQoop (Apache Sqoop, 2016), aplicación que a través de comandos permite transferir datos entre bases relacionales y *Hadoop*; soporta cargas incrementales a partir de una tabla o de una consulta Sql. SQoop, se convierte en un proyecto Apache a partir de 2012. Dentro de los motores que utilizan SQood, están Sql Server de Microsoft, Couchbase, Habase, e Hive, entre otros.
- Cloudera Impala (Apache Impala, n.d), motor de consultas para el procesamiento masivo en paralelo [Massively Parallel Processing, MPP] de datos almacenados en cluster mediante *Hadoop*. Ejecuta las consultas a baja latencia sin necesidad de transformar ni migrar los datos almacenados en HDFS o Hbase, bajo el esquema MapReduce.
- Apache Thrift (n.d), *framework* para servicios de intercambio entre lenguajes, entre los que se incluyen C ++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C #, Cacao, JavaScript, Node.js, Smalltalk, OCaml, Delphi. Dicho *framework* serializa y trasporta e invoca objetos remotos.
- ZooKeeper (Apache Zookeeper, n.d), servicio centralizado para aplicaciones distribuidas que mantiene la información de configuración y denominación; proporciona sincronización distribuida y prestación de servicios de grupo.

Bases de datos NoSQL

Las bases de datos NoSQL se utilizan para la gestión del almacenamiento en un sistema de *Big Data*. Permiten no sólo el almacenamiento de datos estructurados y no estructurados en disco, sino también el uso extensivo de la memoria para soportar operaciones de búsqueda sobre grandes volúmenes de datos, como es el caso de Facebook, Google y Yahoo. El término NoSQL se refiere a un conjunto de sistemas de gestión de datos basados en estructuras no relacionales, que no garantizan las propiedades transaccionales, conocidas como las propiedades

Apache Spark

This became an alternative to the *Hadoop* technology; but, instead using the Map-Shuffle-Reduce sequence used by the latter, Spark uses the Tez *framework*, which includes an acyclic-directed graph for the execution sequence. This *framework* allows the inclusion of sequences like Map-Shuffle-Reduce-Shuffle-Reduce. Its operation is mainly performed in memory and allows the definition of properties with the data that must be in the memory.

Apache HIVE

Apache Hive (n.d) is software with origins in Facebook. It was built as a part of *Hadoop*, but now it is independent under the Apache Foundation. It allows the management of the distributed storage of large data volumes with several formats. HIVE can manage the data storage through HDFS and databases like Hbase. It uses the query language called HiveQL, which executes the queries via MapReduce to provide the data analysis. It lets the same user define when to apply the Map and Reduce functions, and includes customized functions for scalars [UDF], aggregations [UDAF], and tables [UDTF]. Hive is not designed for OLTP load charges and does not offer real-time consults or row-level updates; it is used to work with batches of large datasets.

Other tools

In addition of the aforementioned tools, there are also:

- SQoop (Apache Sqoop, 2016), an application that uses commands, allowing the transmission of data between relational databases and *Hadoop*; it supports incremental loads from a table or a SQL query. SQoop is now an Apache project and the engines using it are SQL server of Microsofy, Couchbase, Habase, and Hive.
- Cloudera Impala (Apache Impala, n.d), query engine for a Massively Parallel Processing [MPP] of data stored in clusters through *Hadoop*. It executes the queries with low latency and without any need to transform or migrate the data stored in HDFS or Hbase, under the MapReduce scheme.
- Apache Thrift (n.d), a service *framework* for the exchange between languages like C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C #, Cacao, JavaScript, Node.js, Smalltalk, OCaml, and Delphi. This *framework* serializes, transports, and invokes remote objects.
- ZooKeeper (Apache Zookeeper, n.d), a centralized service for distributed applications that maintains the configuration and denomination information. It provides distributed synchronization and group services.

NoSQL databases

NoSQL databases are used for storage management in a *big data* system. They allow not only the storage of structured and non-structured data in the disk, but also the extensive use of memory to support searching operations over large data volumes, like Facebook, Google, and Yahoo. The NoSQL term entails a set of data management systems based on non-relational structures, implying that they do not guarantee transactional properties, known as ACID (Atomicity, Consistency, Isolation, Durability), since they are not required in all application scenarios, guaranteeing horizontal distribution. These databases are based on two theoretical foundations:

- The CAP theorem that defines a distributed system can only provide two of the following properties simultaneously: consistency (C), availability (A), or partition tolerance (P) (Brewer, 2012).
- The BASE [Basic Availability, Soft state, Eventual consistence] theorem – derived from the CAP – considers the features of the system: basically available, changing state, and temporary inconsistent but time consistent (Vaish, 2013).

Within the NoSQL systems, there are several data structures to store the information. Given the computational structure used by the information storage in the NoSQL databases, they can be classified as oriented to documents, to columns, to key-value, and to graphs, as we describe in the following sections.

NoSQL databases oriented to documents

They allow one to store, recover, and manage data grouped in the most natural and logical manner, without the restrictions that the relational models impose. In a data model oriented to documents, each register is stored as a document in a flexible structure that stores information non-obeying any scheme (Vaish, 2013). The documents are independent units with an easy distribution into multiple servers and good performance, since the relational data are read in disk continuously. The application logic is easier to write because there is no need for translations between the objects and SQL query languages: an object in the application layer is a document.

The documents are encapsulate and codified in a semi-structured way in a standard format like XML, JSON , BSON (Gudivada, Rao, & Raghavan, 2014). The data are typically accessed with the HTTP protocol using the RESTful API. Some NoSQL database implementations are oriented to documents are MongoDB, CouchDB, Apache Cassandra, and BaseX.

ACID (Atomicity, Consistency, Isolation, Durability) debido a que no son requeridas en todos los escenarios de aplicación, y que garantizan un escalonamiento horizontal. Se basan principalmente en dos fundamentos teóricos:

- El teorema CAP, que define que un sistema distribuido puede proveer únicamente dos de las siguientes propiedades simultáneamente: consistencia (C), disponibilidad (A) o tolerancia a la partición (P) (Brewer, 2012).
- El teorema BASE [Basic Availability, Soft state, Eventual consistence], derivado del teorema CAP que hace referencia a las características del sistema: básicamente disponible, estado cambiante y temporalmente inconsistente, pero consistente con el tiempo (Vaish, 2013).

Dentro de los sistemas NoSQL existen diferentes estructuras de datos para almacenar la información. Según la estructura computacional utilizada para el almacenamiento de la información las bases NoSQL se clasifican en bases: orientadas a documentos, a columnas, a clave-valor y a grafos, como se describe a continuación.

Bases NoSQL orientadas a documentos

Permiten almacenar, recuperar y gestionar datos que son agrupados de la manera más natural y lógica, sin las restricciones que imponen los modelos de datos relacionales. En un modelo de datos orientado a documentos cada registro se almacena como un documento, siendo éste una estructura flexible que almacena información que no obedece a esquema alguno (Vaish, 2013). Los documentos son unidades independientes de fácil distribución en múltiples servidores y de gran desempeño, dado que los datos relacionados se leen de forma contigua del disco. La lógica de la aplicación es más fácil de escribir, pues no hay que hacer traducción entre el lenguaje de objetos y el de consulta SQL: un objeto en la capa de aplicación es justamente un documento.

Los documentos se encapsulan y codifican de forma semi-estructurada en un formato estándar como XML, JSON , BSON (Gudivada, Rao, & Raghavan, 2014). Los datos se acceden típicamente con el protocolo HTTP usando el Api de RESTful . Algunas implementaciones de bases de datos NoSQL orientadas a documentos son MongoDB, CouchDB, Apache Cassandra, BaseX.

Bases NoSQL orientadas a columnas

Una familia de columnas (column family) es una colección de filas; cada fila puede contener diferente número de columnas. Las llaves de cada fila (rows key) son únicas dentro de una familia de columnas, pero pueden ser reutilizadas en otras familias, permitiendo así almacenar datos no relacionados, con la misma llave, en diferentes familias de columnas. Un espacio de llaves (keyspace) es un contenedor de datos; es similar al concepto de esquema en una base de datos relacional, y se utiliza para agrupar familias de columnas. La replicación de datos se específica a nivel del espacio de llaves (keyspace) así, datos replicados residen en keyspaces separados (Vaish, 2013; Gudivada et al., 2014). Algunas implementaciones de bases de datos NoSQL orientadas a familias de columnas son Google BigTable, Apache Cassandra, Apache HBase, Hypertable y Cloudata.

Bases NoSql orientadas a clave-valor

Se caracterizan por almacenar los datos como parejas clave-valor, en las cuales la funcionalidad y el desempeño varían de acuerdo con la implementación. Son similares a las bases de datos orientadas a documentos y se usan especialmente para trabajar datos en memoria mediante el uso de estructuras map, arreglos asociativos o tablas hash, aunque también manejan almacenamiento persistente (Vaish, 2013; Gudivada et al., 2014). Dentro de esta categoría se encuentran: Apache Acumulo, CouchDB, Amazon Dynamo, Apache Casandra y Redis, entre otros.

Bases NoSql orientadas a grafos

Estas bases de datos utilizan los grafos como estructura lógica de representación de los datos. Se caracterizan por que cada nodo representa un objeto. Los nodos se pueden relacionar mediante conectores dirigidos y un nodo puede contener múltiples relaciones. Generalmente, tanto los objetos, como las relaciones, permiten ser caracterizados por propiedades. Dentro de esta clasificación se puede mencionar a: NeO4j, AllegroGraph y FlockDB.

IV. Propuesta de arquitectura multicapas escalonada para big data

En el presente artículo se han explorado las características de *Big Data* y de *Data Warehouse* alrededor de la arquitectura, las características y el ciclo de vida de los diferentes tipos de datos. Se presenta como un desafío, no sólo, establecer los límites entre las dos tecnologías, sino también anticipar el futuro de *Data Warehouse*, a la luz de los nuevos requerimientos de *Big Data*.

Desde el punto de vista del propósito, es claro que el fin último tanto de *Data Warehouse* como de *Big Data*, es el mismo, la exploración de datos con el objeto de identificar patrones, soportar la toma de decisiones, y generar estadísticas e indicadores de gestión. Lo que cambia son los límites, la naturaleza de los datos, el usuario a quien va dirigido y los procedimientos y herramientas para la adquisición, el almacenamiento y el análisis de los datos.

Data Warehouse se enfoca hacia datos estructurados (DE-DR) y en las nuevas propuestas se incluyen datos no estructurados-repetitivos (DNE-DR). Los datos requieren de un pre-procesamiento para entregarle al usuario final la información necesaria para que él pueda realizar sus propios análisis, independiente de las fuentes de datos, el tipo de almacenamiento, la arquitectura, las herramientas y los algoritmos utilizados para llegar a tal resultado. Dicha información se debe presentar a estos usuarios con el nivel de agregación y formato adecuados, lo que significa que la exploración que realiza el usuario final, no se hace sobre los datos en bruto, sino sobre los datos previamente tratados, lo que implica que el usuario final no debe requerir de conocimientos especializados en exploración de datos.

Por el contrario, *Big Data* parte de la exploración de los datos en bruto, dentro de los cuales están los de tipo no estructurados no repetitivos (DNE-DNR), que no son susceptibles de agregación ni de tratamientos sistemático, por lo que requieren usua-

NoSQL databases oriented to columns

A column family is a collection of rows; each file can have a different number of columns. The row keys are unique within a column family, but can be reused in other families, allowing for the storage of non-related data – with the same key – in different column families. A keyspace is a data container; it is similar to the scheme concept in a relational database and is used to group column families. The data replication is specified at keyspace level like this: replicated data are on separated keyspaces (Vaish, 2013; Gudivada et al., 2014). Some NoSQL databases implementations oriented to column families are Google BigTable, Apache Cassandra, Apache HBase, Hypertable, and Cloudata.

NoSql databases oriented to key-value

They are characterized by storing the data as key-value pairs, where the functionality and performance are modified given the implementation. These are similar to the databases oriented to documents and are specially used to work data in memory through the use of map structures, associative arrays or hash tables; even though they can also handle persistent storage (Vaish, 2013; Gudivada et al., 2014). Within this category, we can find Apache Acumulo, CouchDB, Amazon Dynamo, Apache Casandra, and Redis, among others.

NoSql databases oriented to graphs

These databases use graphs as a logical structure for data representation. They are characterized since each node represents an object. The nodes can be related through addressed connectors, and a node can have multiple relationships. Generally, both the objects and the relationships can be characterized by properties. In this classification, we can mention NeO4j, AllegroGraph, and FlockDB.

IV. Proposal of multilayered tiered architecture for big data

We have explored the features of *big data* and data warehouse around the architecture, features, and lifecycle of different data types. We present as a challenge not only the establishment of the limits between the two technologies, but also to "predict" the future of data warehouse from the requirements of *big data*.

From the point of view of the purpose, it is clear that the ultimate goal of both data warehouse and *big data* is the same, i.e., data exploration with the aim to identify patterns, supporting the decision-making process, and generating statistics and management signs. The only difference is related

to the limits, the nature of the data, the end user, and the procedures and tools for the data acquisition, storage, and analysis.

Data warehouse is focused on structured data, and in the new proposals, non-structured repetitive data are included. These data require pre-processing to deliver the end user the necessary information for performing further analyses, regarding the data sources, type of storage, architecture, tools, and algorithms used to achieve such results. This information must be presented to users with adequate aggregation and format, i.e., the exploration that the end user performs is not executed on the raw data, but on the previously treated data. This implies that the end user does not require specialized knowledge in data exploration.

On the other hand, *big data* starts from the exploration of the raw data, where non-structured non-repetitive data are present. These data are not susceptible to the aggregation or systemic treatments; therefore, they require specialized users (i.e. those knowing the technology and data sciences), who by using special tools, techniques, and algorithm patterns, can make identifications in order to conclude about the data.

With the rise of *big data*, it is possible to speculate that data warehouse has completed its lifecycle. However, as a result of our research, it is clear that data warehouse is a complement of *big data*. Consequently, both might be integrated in the same model. In **Figure 4**, we present a proposal of a multilayered tiered architecture that complements the different data sources, their nature, features as arrival frequency, longevity, and opportunity of data access. It might be possible to integrate both *big data* and data warehouse.

As **Figure 4** shows, the layers of the tiered architecture are similar to those in a classic data warehouse: data sources, load of data, storage, processing, and analysis. However, we extend the architecture to include the new *big data* requirements:

- The different data types are considered in an explicit way in the treatment of the sources and in the data loading process.
- The data load will be specialized in the information storage, regardless of the data type. The RSD are susceptible to pre-processing and storage under structures and standard algorithms, e.g., over relational databases (distributed or centralized). The NSRD and NSNRD must be stored raw and free of context, by using the NoSQL distributed databases and under different schemes.

rios especializados (conocedores de tecnología y de la ciencia de los datos), quienes mediante el uso de herramientas, técnicas y algoritmos especiales, puedan identificar patrones y sacar conclusiones acerca de los datos.

Con el surgimiento de *Big Data* se podría especular que *Data Warehouse* ha cumplido con su ciclo de vida. Sin embargo, como resultado de la investigación realizada, es claro que *Data Warehouse* es complemento de *Big Data*, por ende, los dos podrían integrase en un mismo modelo. En la **Figura 4** se muestra una propuesta de arquitectura multicapas escalonada que contempla las distintas fuentes de datos, su naturaleza, sus características de frecuencia de llegada, longevidad, y oportunidad de acceso a los datos, en el que sería posible integrar, tanto a *Big Data*, como a *Data Warehouse*.

Como se observa en la **Figura 4**, las capas de la arquitectura escalonada son similares a las de un *Data Warehouse* clásico: fuentes de datos, carga de datos, almacenamiento, procesamiento y análisis; sin embargo, la arquitectura se extiende para incluir los nuevos requerimientos de *Big Data*:

- Los diferentes tipos de datos (DE-DR, DNE-DR y DNE-DNR) se consideran de manera explícita en el tratamiento de las fuentes y en el proceso de carga de datos.
- La carga de datos se deberá especializar en el almacenamiento de información, según sean datos estructurados repetitivos (DE-DR) o no estructurados (DNE-DR y DNE-DNR). Los DE-DR son susceptibles de pre-procesamiento y almacenamiento bajo estructuras y algoritmia estandarizada, por ejemplo, sobre bases de datos relacionales (distribuidas o centralizadas). Los DNE-DR y DNE-DNR se deberán almacenar en bruto y libres de contexto, para ello se podrán utilizar bases de datos NoSql distribuidas y bajo diferentes esquemas.
- En la capa de procesamiento y almacenamiento los datos estructurados terminarán agregados de acuerdo con un modelo predefinido. Mientras que los DNE-DR y los DNE-DNR, en la medida de lo posible, deben sufrir un proceso de categorización, de acuerdo con el contexto, para filtrar los datos que deben almacenarse en el área de "Datos Contextualizados"; los demás permanecerán en el área de "Datos en Bruto". Del área de "Datos Contextualizados", a través de un proceso búsqueda de relaciones o patrones, escalarán al área de "Datos relacionados" (este escalonamiento no implica su eliminación del área de "Datos Contextualizados"). Los "Datos Relacionados" ya procesados que puedan adaptarse a estructuras o semi-estructuras predefinidas, y con algún tipo de indexación que permita sistematizar las consultas, podrán alimentar el área de "Datos Explorados". Por último, será posible que algunos de los datos explorados se logren integrar con la bodega de datos estructurados.

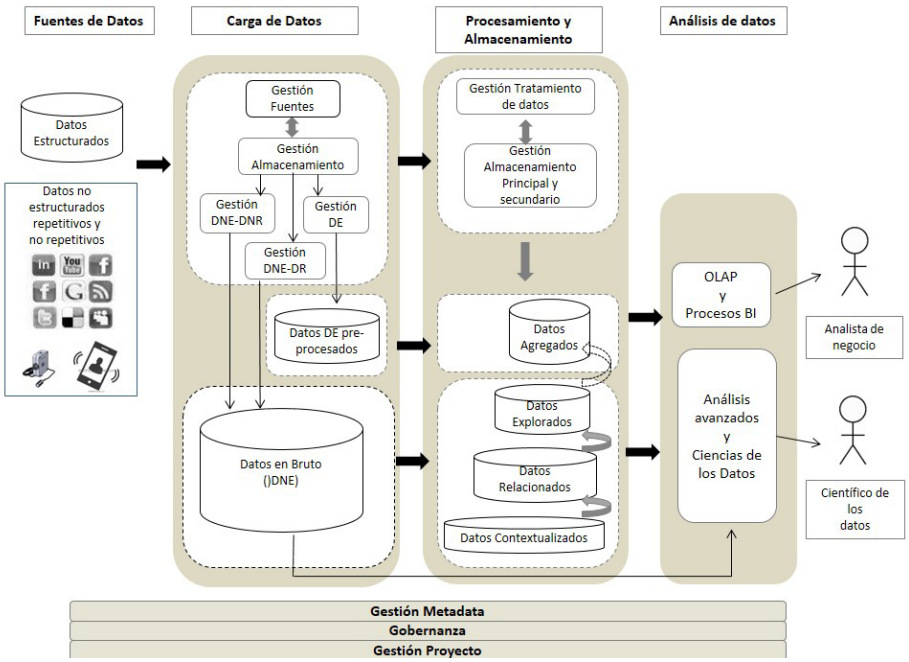

Figure 4. Multilayered tiered architecture for big data[6] / Arquitectura multicapas escalonada para Big Data[6]

- El escalonamiento de los datos a través de las diferentes áreas de almacenamiento permitirá, no solo gestionar la longevidad de los datos, sino la oportunidad de acceso para la toma de decisiones, sin desechar esfuerzos anteriores: en el área de "Datos Contextualizados" se ubican los datos más recientes que soportan análisis inmediatos; en el área de "Datos Relacionados" se encuentran datos de longevidad media, que ya han sido procesados; en el área de "Datos Explorados" residen datos históricos, que soportan los análisis estadísticos; y en área de "Datos Agrupados" se encuentran datos históricos que soportan análisis OLAP y la inteligencia de negocios. La gestión del escalamiento en las áreas de almacenamiento requiere de procesos distribuidos, soportados en tecnologías como *Hadoop* y los componentes asociados a su ecosistema.
- La arquitectura propuesta especializa a dos tipos de usuarios: los analistas de negocio, usuarios finales de soluciones OLAP e inteligencia de negocios, quienes trabajaran sobre el área de "Datos Agrupados"; y los científicos de los datos, quienes lo harán desde "Datos en Bruto" hasta "Datos explorados".
- Este modelo de arquitectura multinivel escalonada requerirá la existencia de un componente transversal que gestione la metadata en todos los procesos, y que le de cohesión y semántica a los datos mediante el uso de ontologías o bases de datos especializadas.
- Un proyecto de esta naturaleza resulta complejo por la cantidad de consideraciones que se deben tener en cuenta, por lo tanto, será necesario adoptar estrategias de gestión para su desarrollo y mantenimiento, así como

- In the processing and storing layer, the structured data will be aggregated given a pre-defined model. On the other hand, the NSRD and NSNRD must suffer a categorization process related to the context, to filter the data that must be stored in the "contextualized data" area. The remainder will be in the "raw data". From the first and through a relations/patterns searching process, we will move to the "related data" area (this movement does not imply the elimination of the "contextualized data" area). The "related data" that are already processed and capable to be adapted to predefined structures or semi-structures and with some type of indexation that allows the systematization of the queries, will be able to feed the "explored data" area. Finally, it will be possible that the explored data are joined with the warehouse of structured data.
- The moving of the data through the different storage areas will allow not only the management of the data longevity, but also the access opportunity for the decision-making process, without discarding previous efforts. That is, in the "contextualized data" areas, the most recent data that supports immediate analysis will be located. In the "related data" area, the data with medium longevity are present, and are already being processed. In the "explored data" area, historical data are present that support OLAP analysis and business intelligence. The scaling management in the storage areas requires distributed processes, supported in technologies like *Hadoop* and its associated components.
- The proposed architecture specializes two user types: the business analysts – end users of OLAP solutions and business intelligence that work in the "gathered data" area – and data scientists, who will work from "rad data" to "explored data".
- This multilayered tiered architecture will require the existence of a transverse component that manages the metadata in all the processes. This component should also provide cohesion and semantic to the data through the use of ontologies or specialized databases.

- A project of this nature is complex due to the number of considerations to take into account. Consequently, it will be necessary to adopt management strategies for its development and maintenance, besides governmental policies to define agreements and communication mechanisms between the different actors. These policies must consider change management, standards handling, restrictions control, and the adoption of best practices for its development. For this reason, we propose –besides the management of the 7 V and the 3 C – the management of the new 2 G.
- The implementation of the multilayered tiered architecture requires the integration of technologies like *Hadoop*, SQL and NoSQL databases, and – in general – several of the previously described tools.

V. Conclusions and future work

Given the nature and complexity of the analysis of the new sources and data types, it is more difficult to solve the topic of *big data* and data warehouse architecture. For this reason, we have proposed a multilayered tiered architecture that attempts to provide an answer to the presented requirements by recognizing the lifecycle, the data sources and the possibility of integration and reuse of the data in a systemic way. Besides this, the ETL processes and the analysis algorithms supported in a *framework* for the management of the metadata, the governability, and project management that considers the location, integration, longevity, use, availability, access, security, and control of any modification of the data and processes, are all required.

For future work, we propose to implement the proposed architectonic model by using some of the described technological tools. The *Hadoop* and NoSQL databases are very important for this task.

[1] MapR: Apache Hadoop distribution by Hewlett Packard.
[2] Dremel is a distributed system developed in Google for the interactive query of large datasets.
[3] JSON [JavaScript Object Notation] is a light, open, text-based standard that allows data exchange between a server and a web application.
[4] BSON [Binary JSON] is a data exchange format of JSON documents in binary representation.
[5] REST [Representational State Transfer] is an HTTP API that uses the four methods GET, POST, PUT, and DELETE to execute several operations.
[6] Images of non-structured repetitive and non-repetitive data were adapted from Carter (2013).

políticas de gobernanza para definir acuerdos y mecanismos de comunicación entre los diferentes actores. Dicha gobernanza deberá considerar la gestión del cambio, el manejo de estándares, el control de restricciones y la adopción de mejores prácticas para el desarrollo. Por lo anterior, se propone, además de gestionar las 7 V y las 3 C, gestionar las nuevas 2 G.

- La implementación de la arquitectura multicapas escalonada, requiere la integración de tecnológicas como *Hadoop*, Bases Sql y NoSql, y en general, varias de las herramientas descritas previamente.

V. Conclusiones y trabajo futuro

Dada la naturaleza y complejidad de análisis de las nuevas fuentes y tipos de datos, cada vez es más imperioso resolver el tema de la arquitectura de *Data Warehouse* y *Big Data*, por lo que en este articulo se ha propuesto un modelo de arquitectura multicapas escalonada que intenta dar respuesta a los requerimiento planteados, reconociendo el ciclo de vida, las fuentes de datos y la posibilidad de que de manera sistemática se integren y reutilicen los datos, los procesos ETL y los algoritmos de análisis, todo ello apoyado en un marco de gestión de la metadata, la gobernanza y la gestión del proyecto que considere aspectos de ubicación, integración, longevidad, uso, disponibilidad, acceso, seguridad y, en general, del control de cualquier modificación sobre los datos y procesos.

Como trabajo futuro se plantea el desafio de implementar el modelo arquitectónico propuesto, utilizando algunas de las herramientas tecnológicas descritas, en lo que *Hadoop* y las bases NoSql juegan un papel preponderante.

[1] MapR: distribución de Apache Hadoop de Hewlett Packard.
[2] Dremel es un sistema distribuido desarrollado en Google para la consulta interactiva de grandes conjuntos de datos.
[3] JSON [JavaScript Object Notation] es un estándar ligero, abierto, basado en texto, que permite intercambiar datos entre un servidor y una aplicación web.
[4] BSON [Binary JSON] es un formato de intercambio de datos de documentos JSON en representación binaria.
[5] REST [Representational State Transfer] es un API HTTP que usa los cuatro métodos GET, POST, PUT y DELETE para ejecutar diferentes operaciones.
[6] Las imágenes de los datos no estructurados repetitivos y no repetitivos se adaptaron de Carter (2013).

References / Referencias

Agrawal, D. (2009). The reality of Real-Time Business Intelligence. En: M, Castellanos, U, Dayal. & T, Sellis. (Eds.), *Lecture Notes in Business Information Processing. Vol. 27. Business Intelligence for the Real-Time Enterprise* (pp. 75-88). Berlin Heidelberg : Germany : Springer-Verlag Berlin Heidelberg : Germany

Apache Hive TM. (n.d.). Retrieved from https://hive.apache.org/

Apache Impala. (n.d). Retrieved from: http://www.cloudera.com/products/apache-hadoop/impala.html

Apache Sqoop (2016, march 4). Retrieved from: http://sqoop.apache.org/

Apache SparkTM-Lightning-fast cluster computing. (n.d.). Retrieved from: http://spark.apache.org/

Apache Thrift - Home. (n.d.). Retrieved from https://thrift.apache.org/

Apache ZooKeeper - Home. (n.d.). Retrieved from https://zookeeper.apache.org/

Architecture - Apache Drill. (n.d.). Retrieved from http://drill.apache.org/architecture/

Bedi, P., Jindal, V., & Gautam, A. (2014). Beginning with big data simplified. In: Data Mining and Intelligent Computing (ICDMIC), 2014 International Conference on. IEEE. doi:10.1109/ICDMIC.2014.6954229

Brewer, E. (2012). CAP twelve years later: How the "rules" have changed. Computer. 45(2), 23-29.

Carter, S. (2013, Feb, 21). Social and BIG Data! #socbiz #ibmsocialbiz #bigdata #socialbusiness. Retrieved from: http://socialbusinesssandy.com/tag/big-data-2/page/14/

Chandarana, P. & Vijayalakshmi, M. (2014). Big data analytics frameworks. In Circuits, Systems, Communication and Information Technology Applications (CSCITA), 2014 international conference on (pp. 430-434. IEEE.

Cox, M. & Ellsworth, D. (1997). Application-controlled demand paging for out-of-core visualization [NASA Reports]. Retrieved from: http://www.nas.nasa.gov/assets/pdf/techreports/1997/nas-97-010.pdf

Cuzzocrea, A. (2014). Privacy and security of big data: current challenges and future research perspectives. In: Proceedings of the First International Workshop on Privacy and Security of *Big Data* (pp. 45-47). New York, NY: ACM. http://doi.acm.org/10.1145/2663715.2669614

Demchenko, Y., Laat, C. & Membrey, P. (2014). Defining architecture components of the big data ecosystem. In: Collaboration Technologies and Systems (CTS), 2014 International Conference on, 104-112. IEEE.

Díaz, Ma. (2011). Evaluación de la herramienta de código libre Apache Hadoop [thesis]. Universidad Carlos III de Madrid Escuela Politécnica Superior: Leganés, España.

Gudivada, V., Rao, D. & Raghavan, V. (2014). NoSQL systems for big data management. In: 2014 IEEE World Congress on Services (pp. 190-197). IEEE.

HDFS architecture guide. (2013, April 8). Retrieved from: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

Hewlett Packard. (2013). HP Reference Architecture for MapR M5 [technical white paper]. Retrieved from: https://www.mapr.com/sites/default/files/hp_reference_architecture_for_mapr_m5.pdf

Inmon, W. (2005). Building the data warehouse [4a ed.]. Indianapolis, IN: Wiley.

Inmon, W.,Strauss, D. & Neushloss, G. (2008). DW 2.0: The Architecture for the Next Generation of Data Warehousing. Burlington, MA: Morgan Kaufmann

Inmon. H. & Linstedt, D. (2014). Data architecture: A primer for the data scientist: big data, data warehouse and data vault. Waltham, MA: Morgan Kaufmann.

Katal, A., Wazid, M. & Goudar, R. (2013). Big data: Issues, challenges, tools and good practices. In: Contemporary Computing (IC3), 2013 Sixth International Conference on (pp. 404-409). IEEE.

Kimball, R. (2011). The evolving role of the enterprise data warehouse in the era of big data analytics [Kimball Group white paper]. Retrieved from: http://www.montage.co.nz/assets/Brochures/DataWarehouseBigDataAnalyticsKimball.pdf

Kimball, R. (2012). Newly emerging best practices for big data [Kimball Group, white paper]. Retrieved from: http://www.kimballgroup.com/wp-content/uploads/2012/09/Newly-Emerging-Best-Practices-for-Big-Data1.pdf

Kimball, R., Ross, M., Thorthwaite, W., Becker, B. & Mundy, J. (2008). The data warehouse lifecycle toolkit [2a ed.]. Indianapolis, IN: Wiley.

Lomotey, R. K., & Deters, R. (2014). Towards knowledge discovery in big data. In: Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on (pp. 181-191). IEEE.

MacDonald, A. (2015). Integrating SAP HANA and hadoop. Boston, MA: SAP Press.

Maiorescu, T. (2010). General Information on Business Intelligence and OLAP systems architecture. In: Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on (V.2, pp. 294-297). IEEE.

Manikandan, S. G., & Ravi, S. (2014). Big data analysis using Apache Hadoop. In: IT Convergence and Security (ICITCS), 2014 International Conference on. doi: 10.1109/ICITCS.2014.7021746

Manning, C. & Schütze. H. (1999). Foundations of statistical natural language processing. Cambridge, MA: The MIT.

Marz, N. (n.d). Storm, distributed and fault-tolerant realtime computation. Retrieved from: http://cloud.berkeley.edu/data/storm-berkeley.pdf

Muntean, M., & Surcel, T. (2013). Agile BI - The Future of BI. Informatica Económica, 17(3), 114–124.

Nam, T., Choi, K., Ok, C. & Yeom, K. (2014). Service composition framework for big data service. In: Future Internet of Things and Cloud (FiCloud), 2014 International Conference on (pp. 328-333). IEEE.

Nandimath, J., Banerjee, E., Patil, A., Kakade, P., & Vaidya, S. (2013). *Big Data* analysis using Apache Hadoop. In: 2013 IEEE 14th International Conference on Information Reuse & Integration (IRI) (pp. 700-703). IEEE.

Oracle Corp. (2015). An enterprise architect's guide to big data [Oracle enterprise architecture - white paper.]. Retrieved from: http://www.oracle.com/technetwork/topics/entarch/articles/oea-big-data-guide-1522052.pdf

Pal, A. & Agrawal, S. (2014). An experimental approach towards big data for analyzing memory utilization on a hadoop cluster using HDFS and MapReduce. In: Networks & Soft Computing (ICNSC), 2014 First International Conference on (pp. 442-447). IEEE.

Schaffner, J., Bog, A., Krüger, J., & Zeier, A. (2009). A hybrid row-column OLTP database architecture for operational reporting. In: M. Castellanos, U. Dayal, & T. Sellis (Eds.), Business intelligence for the real-time enterprise (pp. 61-74). Berlin Heidelberg, Germany: Springer.

Todman, C. (2001). Designing a data warehouse: Supporting customer relationship management. Nueva Jersey, NJ: Prentice Hall.

Vaish, G. (2013). Getting started with NoSQL. Birmingham UK: Packt.

Welcome to ApacheTM Hadoop®! (n.d.). Retrieved from: https://hadoop.apache.org/

YiChuan, S. & Yao, X. (2012). Research of Real-time Data Warehouse Storage Strategy Based on Multi-level Caches. Physics Procedia, 25, 2315–2321.

Zhang, R., Hildebrand, D., & Tewari, R. (2014). In unity there is strength: Showcasing a unified *Big Data* platform with MapReduce Over both object and file storage. In: *Big Data* (*Big Data*), 2014 IEEE International Conference on (pp. 960-966). IEEE.

CURRICULUM VITAE

Sonia Ordóñez Salinas Ph.D. in Systems, M.Sc. in Systems and Computing and Statistics of the Universidad Nacional de Colombia (Bogota) and Systems Engineer of the Universidad Distrital Francisco José de Caldas(Bogota). She has a widely experience in statistic models to transform text in graphs, recovery systems and databases, processing of natural language, data mining, and social networks. She currently is a full time professor in the Universidad Distrital Francisco José de Caldas and she is the leader of the GESDATOS research group, associated to the same university / Doctora en Sistemas, Magister en Sistemas y Computación, y Estadística, de la Universidad Nacional de Colombia (Bogotá);e Ingeniera de Sistemas de la Universidad Distrital Francisco José de Caldas (Bogotá). Cuenta con amplia experiencia en: métodos estadísticos para transformar texto en grafos; sistemas de recuperación y bases de datos; procesamiento de lenguaje natural, minería de datos y redes sociales. Trabaja actualmente como docente de planta en la Universidad Distrital Francisco José de Caldas y dirige el grupo de investigación GESDATOS, asociado a la misma universidad.

Alba Consuelo Nieto Lemus Systems engineer of the Universidad Nacional de Colombia (Bogota), M.Sc. in in Systems Engineering and Computing of the Universidad de los Andes (Bogota). She has a wide professional experience in software development, software architectures, data management and software quality management, both in the public and private sectors. She currently works as a full time professor of the Universidad Distrital Francisco José de Caldas and she is a member of the GESTADOS and ARQUISOFT research groups, associated to the same university / Ingeniera de Sistemas de la Universidad Nacional de Colombia (Bogotá); Magister en Ingeniería de Sistemas y Computación de la Universidad de Los Andes (Bogotá). Cuenta con una amplia experiencia profesional en desarrollo de software, arquitecturas de software, gestión de datos y gestión de la calidad de software, tanto en el sector privado como en el público. Trabaja actualmente como docente de planta en la Universidad Distrital Francisco José de Caldas y es integrante de los grupos de investigación GESDATOS y ARQUISOFT, asociados a la misma universidad.