
IBIMA
Publishing
mobile

Journal of Software & Systems Development

Vol. 2012 (2012), Article ID 840273, 73 minipages.
DOI:10.5171/2012.840273
www.ibimapublishing.com

OLAP Services on Cloud Architecture

Author

Cătălin Strîmbei

Al.I.Cuza University, Faculty of Economics and Business Administration, Iaşi, Romania

Abstract

Although OLAP technology seemed to be mature enough so that little real or disruptive changes encountered in the last decade. Still the emergence and the wide spreading of some technologies like Web Services, Cloud Computing and Big Data we believe that will have the potential to paradigmatically change the OLAP technological landscape. One of the most important consequences could be a more widely open access to web analytical technologies. In the current paper we have tried to (re)evaluate the OLAP Web Services viability in the context of the Cloud based architectures.

Keywords: OLAP, Data Web Services, Data Warehousing, SQL, Service Data Objects, Cloud Computing, APEX, Google Docs, Amazon Web Services.

Introduction

With the advent of the BigData wave it became clearly that OLAP-area will not be avoided by this kind of technological turbulence. In fact, the current technological tendency of cloud computing with all its direct and indirect consequences is the main source of the technological and architectural shift of OLAP context.

Considerations on Data Integration and Data Warehousing Architecture

Generically, every OLAP application requires a certain level of data organizing. That's why the OLAP technology is often considered as part of a DataWarehousing architecture. The current "cloud democracy" for data access changed this

architectural picture so that one can perform analytical analysis on Web provided data, as Lean Y. et al. (2008) mentioned. Consequently, there are many visual analytic tools based on web technologies and standards to make easier this kind of job for end-users. Here is the first “gap” that we want to outline:

- Web provided data for analytics are not quite appropriately structured for OLAP processing, and

- The traditional and well-known OLAP tools are not quite ready to smoothly integrate these kind of data, otherwise than a pre-integration phase that accumulates them into a pre-defined Data Warehouse.

The primary architecture that someone can think about as a valid solution to fill this gap will have a Data Warehouse as basis, and looking for solutions to allow web access, as in Figure 1. Consequently the web standards and technologies will be used, among others, as an alternative for data interchange, but most of the weight will still remain on proprietary technologies and tools for Data Warehouse.

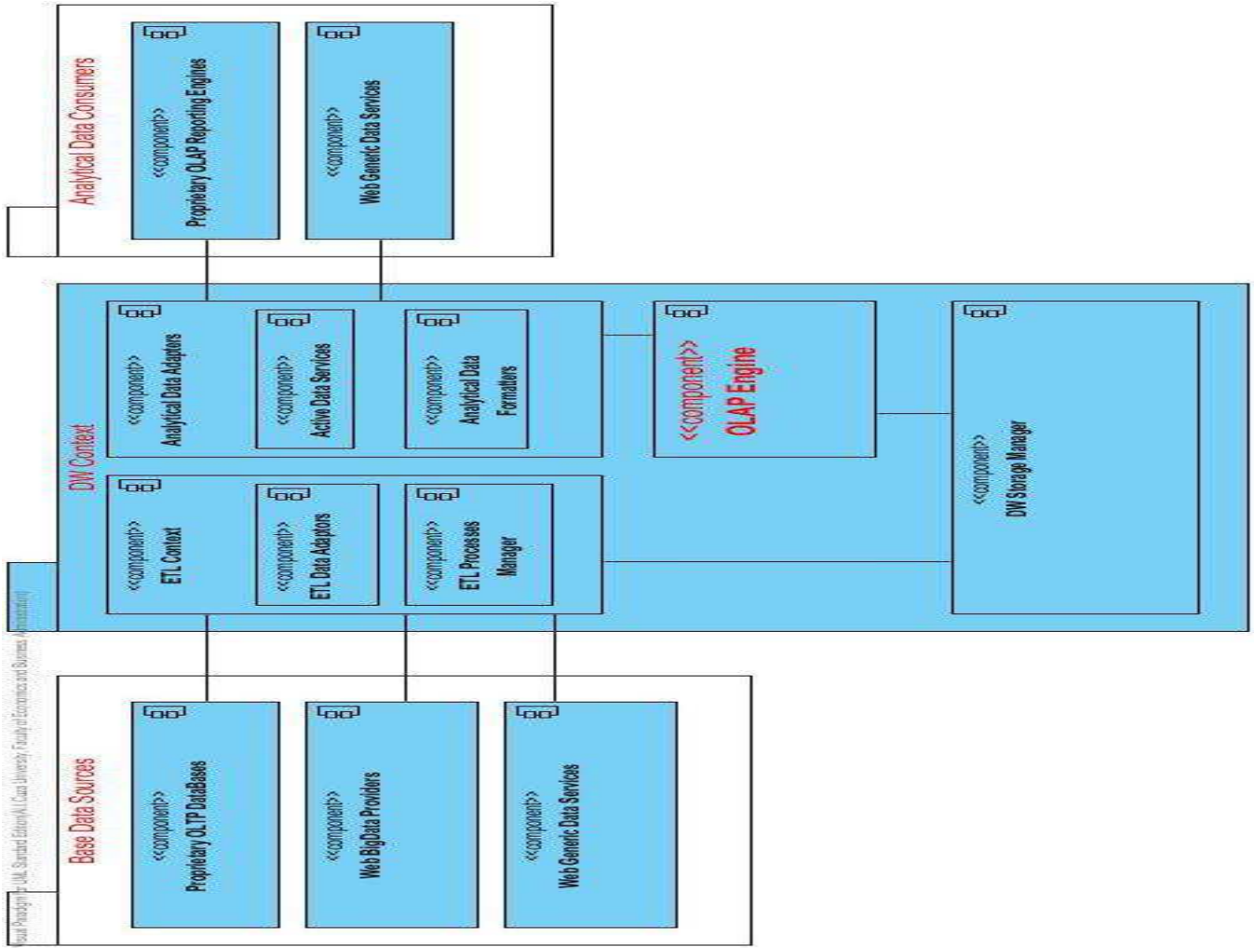

Fig 1. Base Architecture for Datawarehouse and OLAP

The Data Warehouse specific tools for data entry/acquisition form the main gate into the analytical context where the analytical content is structured, stored and prepared to be delivered to the “analytical consumers”.

This kind of tools is often known using the ETL acronym, from Extract-Transform-Load syntagma and they are considered extremely important due to their potential data integration role into any data consolidation strategy that is specific to the DataWarehouse systems, as Adzic J. et al. (2007), Simitisis A. et al. (2007) and Giordano A.D. (2011) outlined. The ETL tools have, at least, two main kind of capabilities:

- Orchestration of heterogeneous data processing;

- Definition of a data integration model to be referenced by data processing.

Consequently, the ETL specific tools not only mediate the access to the providers of base data sources , but they have to make compatible with the heterogeneous data sources to an denominated or common data model from where OLAP tools could make analytical assumptions like financial diagnoses as those from the paper of Pavaloia V.D. (2009).

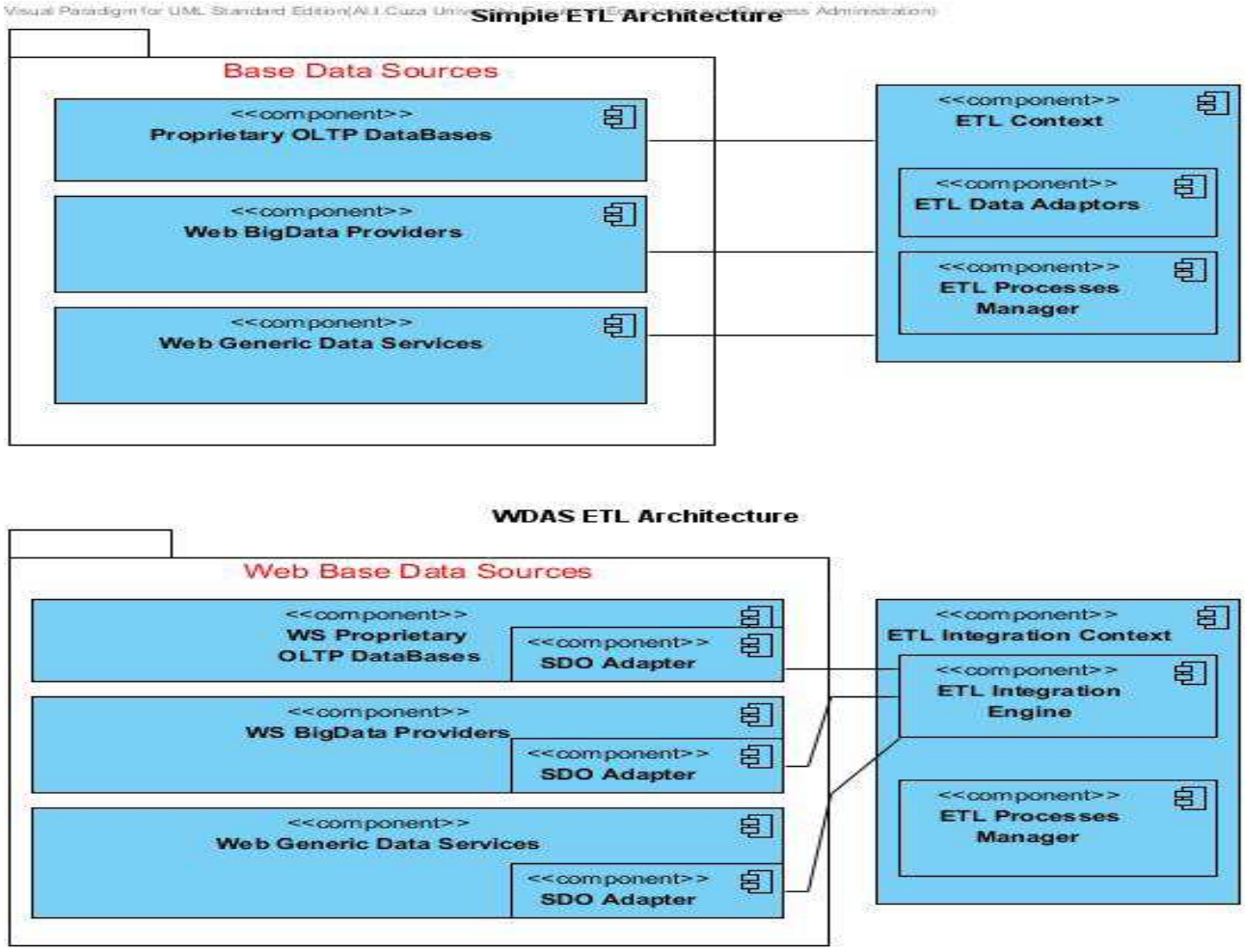

Fig 2. The Comparison of ETL Architectures

We have to say, before anything else, that, these days, the data interchange have become increasingly more like data integration, or, in other words, the data interoperability become increasingly more data integration.

Today, there are plenty of data access technologies and “data drivers”. Beyond the advantage of the richness of data sources, there is at least one big issue concerning this situation: the ETL tools must be extremely complex to be able to integrate the heterogeneous data flows (data access channels) and data formats.

Therefore, the web accessibility of business data sources becomes one of the most prominent trends of these days regarding the evolution of data intensive business information

systems as those described by Dospinescu O. and Perca M. (2011). The main strategy to benefit from the openness of the interoperability of web context consists in using the web services based technologies as a data interchange agent.

We already explored such data integration strategy in a SOA based architecture, see Strimbei C. (2012), the architecture that guided our experimental work to transform a “traditional” object-relational database into a veritable (full-fledged) WDAS (as Web Data Access Services). Our implementation was based on SDO initiative (Service Data Objects), explored by Antonioletti M. et al. (2006), Portier B. and Budinsky F. (2004), and Resende, L (2007), initiative usefully to define a core data interchange format with the advantage of being platform-agnostic.

Taking into account these considerations, the ETL architectural pictures in Figure 1 could evolve into a different architecture partially illustrated in Figure 2.

Considerations on OLAP Engines and Cloud Computing

The main ability of the OLAP engine component from figure 1 is to resolve the OLAP specific queries on data stored in the accessible data warehouses (DW). Consequentely, the components from the ETL context have to integrate data flows, but with respect to the data model which dictates the composition of data structures from the Data Warehouse.

The OLAP tools use specific data models that are very different in nature than those data models used by the operational business

information systems. The OLAP queries are built on top of those data models that support concepts like dimensions, measures, variables, hierarchies, and formulas discussed by Thomsen E. (2002).

The concepts that reflect the OLAP principles, although their nomination varies among different analytical products, imply a very special data type system that must support specific Business Intelligence (BI) analytical functions. The output exposed to the end user is very often like a dashboard, but the underground data comes from the data structures defined by the specific OLAP type system. The most common name for a data repository designed with this kind of data type systems is the multidimensional database (MD).

There are different incarnations or implementations for OLAP specific data types. They could be classified into three major categories:

- The relational based category covers those OLAP models that intend to have a very solid and formal background that is founded on those mathematical bases specific to relational model, as Varga M. (2002) argued; another very solid argument of this kind of MD databases is the SQL support extended with a specific class of operators that allow OLAP queries presented by Celko J. (2006);

- The object oriented category covers those OLAP models that are grounded on some consideration regarding the inability of relational based OLAP tools to properly represents the

semantics of some very important OLAP concepts like multidimensional projections, dimension inheritance, derived measures etc., presented by Pardillo J. (2008);

- The XML based category of OLAP tools comes from the fact that a very large amount of business operational data is stored and/or flow as XML documents. Therefore there is a need to analytically process these kinds of data in the same way like the relational data are analyzed multi-dimensionally, as Hümmer W. et al. (2003) and Park P. et al. (2005) argued.

Consequently, the intimate nature of multidimensional data type systems, reflected by these categories, will fundamentally determine the storage systems that will persist the OLAP specific data.

There is a difference between the fact of being reachable “from the web” and actually being “on the web”. The cloud computing paradigm proves to be a “disruptive” shift for building web-ready applications. The main advantages of using this kind of web applications deployed on cloud come from:

- The availability levels that could be ensured by the cloud provider,

- The scalability and security levels, and

- The cost control of operating and maintenance depending on the pricing schema provided by the cloud provider.

The main issues or architectural differences that need to be investigated refer to the modularization levels and techniques to be approached in the design of business applications. In this regard, we already proposed a BI-modularized architecture for cloud computing, see Strimbei C. (2010). Our modularization approach is based on these guiding principles:

- Business Intelligence as a Service Oriented Platform to include distinct and autonomous (from the deployment point of view) architectural components as services;

- Cloud hosting based services, as a combination of storage for inner BI data structures (Data Warehouse storage in figure 1), processing power for BI (ETL), data transformation rules (like

OLAP engine in figure 1), and BI Reporting visualization portal (like OLAP reporting engine in figure 1);

- Client oriented customization workflow, to define the entire data flow chain from the base data sources to the BI reports delivered to the end-users.

In our vision, this approach will have to enable:

- A some degree of flexibility in the architectural style: from a very focused and very centralized cloud computing context (where all services are hosted by the same cloud platform) to a very modularized architecture where any component ready to be deployed as service could be resident into a different cloud context but still interoperable with the rest of the BI system;

- A comfortable level of openness concerning service interoperability, data interchange protocol, workflow design, definition and monitoring tools. This openness is based on the richness of open-source initiatives due to the tools like: JasperServerAnalytics, Eclipse BIRT OLAP, SpagoBI, PentahoBI and others (online) like: PaloBI. Saiku, DataBrewery, GoogleFusionTables&GoogleAnalytics.

This way, one can store the DataWarehouse collections privately, but could still use an OLAP SaaS in order to manage its OLAP queries, or one can use a DBaaS to store its DataWarehouse repository, and, in the same time, could provide its own private OLAP SaaS to run its OLAP queries and reports.

Discussion on OLAP and Web Service Technology

An OLAP Framework for Cloud Architecture

Our framework proposal is grounded on the modularization approach previously discussed. Such design and deploy framework for OLAP applications as cloud-based web services (which we could call as OLAPaaS) involves, at least, three specific issues to argue:

- Specific data interchange;
- OLAP specific data collections to store;
- OLAP specific queries to call and to deliver their output.

Our opinion is that although XML based (with XSD specific data type system) OLAP data processing models (like XCube) may not be superior to the OLAP-SQL Extensions or to the semantic expressiveness of the Object-Oriented OLAP models and languages, XML remains critical from the perspective of Web Services interoperability. There are two points where XML-interoperability is critical for our proposed BI architecture:

- The connection of the external web data services with the ETL integration engine where the simple form of the SDO standard could be used (see figure 2);

- The integration of OLAP engine with the OLAP data sources that could be:

- A cloud based DataWarehouse that can be considered somehow internal to BI system;

- Any external data source which is compatible with OLAP requirements, meaning that depending on the architectural style used, a customized OLAP application system may not have a private DataWarehouse but rather imports its OLAP data from external service providers.

The most simplified architecture for BI-OLAP web services may assume that OLAP services consume only external data sources that are formatted consistently with OLAP requirements. A more sophisticated architectural style assumes a closer integration with the base DataWarehouse for OLAP queries. There are several possibilities to deploy a such DataWarehouse on the

public or private cloud using DBaaS (database as a services) platforms that support the traditional relational data model (SQL99), the object-relational data model (SQL3, SQL2008, SQL2011) or the object oriented data model (using NoSQL or pure object oriented technologies like Cache).

Again, a minimal architecture for OLAP Web Services will just deliver the results of OLAP queries as plain textual data to other Web components that will transform them in some user oriented visual formats (like JChart, BIRT Chart or GoogleChart). In this context, can be elaborated a XML/SDO like specific format in order to support OLAP data interchange.

The Potential of OLAP Web Services

When they were invented, OLAP, as ONLINE Analytical Processing, assumed a class of software functions available throughout an enterprise network system. The OLAP tools have prevailed over time in the context of Data Warehousing-DW, Decision Support Systems-DSS oand Data Mining DM. The ONLINE term has kept its relevance, but, today, its contextual meaning has changed. Today, ONLINE suppose mainly the web and cloud-based technologies, therefore one can observe the emergence of new technologies like Web Warehousing, as Lean Y. (2008) mentioned. Likewise, some other analytical tools or technologies were re-branded as Web Analytics, but we will prefer the OLAP-Web Services term (with OLAP-WS acronym) to better outline the predictable technological context.

As one can easily concludes from the framework shortly described above, there are (at least) two quite different approaches concerning developing and deploying of OLAP as Web Services.

This first approach assumes a Business Intelligence complex surrounding a very integrated OLAP component, alongside with ETL, DW and other highly sophisticated analytical tools like data mining. This kind of architecture has some distinctive features:

- Existence of a comprehensive and extensible ETL Layer;

- Storage of analytical data in a technologically complex system like Data Warehouse;

- The main and singular data provider for analytical tools (OLAP) is the heavy data warehouse system;

OLAP components, although they may be accessible from external sites, they are not considerate as fully-external-accessible resources, because the OLAP data model and the query engine are dependent on data collections and formats from Data Warehouse.

Therefore, the OLAP components could be wrapped as Web Services in order to be invoked with analytical requests on already established (but not customizable) DW data structures. The most flexible and functional approach of a such architecture is to invoke these OLAP-Web Services with dynamic queries as parameters (compiled against static DW data structures) in order

to get dynamic structured result sets which could then be valued by some business user-oriented tools like Google Docs Spreadsheets with specific diagrams or chart gadgets.

OLAP Components as Autonomous Cloud-Based Web Services

This second approach assumes that OLAP Web Services will be released from the Data Warehousing dependency. Among the distinctive features of this kind of architecture could be outlined:

The OLAP-WS admin-user will have to provide data flows (which are already dimensional or could be easily formatted this way) to the original analytical data collections that will be entirely stored externally (maybe on another DBaaS cloud platform).

The OLAP-WS will have to register these data-flows as analytical data sources with their URLs for access and with their specific meta-data. The OLAP components will have to use an internal registry which will not store the actual analytical data but their meta-data.

The OLAP-WS will provide mainly (as their defining feature) the analytical-query-processing service. It will not access (by default) or will define internally a Data Warehouse infrastructure, except maybe a data storage system for temporary processed data sets from external data flows.

The users' visualization tools of OLAP dataset results will be external, but they could be integrated with original OLAP-WS so that the concrete output of the query invocation to be an URL to

an external web document that will host and visually model the OLAP resultsets.

The OLAP-WS will not be dependent on any existing DW infrastructure, but will be dependent on:

- A dimensional-data-access protocol (XML based) to interpret the inputs from the registered data flows;

- A special extension of the query specific OLAP language (like SQL-OLAP extensions) that is necessary to mention the registered data flows as a special kind of dimensional data views;

- An analytical XML data format needed to export/output the SQL-OLAP query results.

Experimental Study

In order to explore the possibility to develop such Web Services project, using the already presented OLAP framework, we made a short experimental study.

Architectural Issues Concerning OLAP Autonomous Web Services - OLAP AWS

There are some important issues that come in the way of developing a feasible architectural background needed to deploy

the necessary components to support the publishing of OLAP Web Services.

The **generic functional work flow**, covering the activities ranging from the necessary customization to the delivering analytical result-set, assumes a sequence of activities and among them the most representative ones could be:

1. An OLAP WS-operation will be invoked in order to insert meta-data necessary to describe the necessary URIs into the specific registry of the OLAP component. These meta-data describes how to locate original data sources in order to initiate the analytical/dimensional data flows, and to describe the structural types necessary to parse the SQL-OLAP queries.

2. (Optional) An OLAP WS-operation will be invoked in order to register the specific OLAP consumers that will receive the analytical result-set that will be delivered by the OLAP component.

3. An OLAP WS-operation will be invoked in order to specify the custom OLAP queries on the registered data flows.

4. Some internal OLAP processing will take place in order to parse remitted OLAP queries and then to bind the data links to registered data flows specified by these queries.

5. Some internal OLAP processing will take place in order to initiate the bounded data flows and to process the OLAP computations from the OLAP query.

6. Finally, some internal OLAP processing will take place in order to format and deliver OLAP result set to WS-invoker or to registered WS-consumer.

In order to describe the external dimensional data collections we distinguished (at least) two possibilities:

- Using plain XSD complex data typing infrastructure (metadata = XSD data types);

- Using an associated XML document to describe meta-data through its ordinary elements (metadata = a collection of self-contained XML tags).

The SDO standard, presented by Resende L. (2007) and Strimbei C. (2012), could be used to get data from external data sources and to export analytical data resultsets produced through OLAP query processing. Therefore XSD schema describing SDO objects must be included in WSDL document under the types subtag of the definitions main tag. Also, the SDO collection representing analytical resultset will form the content of the body tag of the envelop from the Web Service response document.

Another critical issue concerns how to represent Dimensional Data Links into OLAP query language so that the query engine to easily and correctly parse and then bind these references to the actual registered data flows. Taking into consideration the SQL-OLAP query language, we can extend it in order to mention these references:

1. As already configured external data sources, their meta-data being stored before into OLAP-WS configuration repository;

2. As special formatted URLs to the Dimensional Web Services, meaning that those Web Services will actually provide the already modelled data collection as data dimensions and will interchange them using a SDO-like standard;

3. By means of some sub-queries that will invoke Data Web Services which will provide analytical data that are not already formatted dimensionally. These sub-queries will have the responsibility to process these raw data taking into consideration the specific dimensional requirements as defined by Thomsen E. (2002).

/* Listing 1: Templates for SQL-OLAP subclauses to invoke external dimensional data collections */

(1) Already registered

SELECT ...

FROM external_schema1.dimension_x … external_schema1.dimension_y

GROUP BY CUBE ...

(2) Special formatted URLs

SELECT ...

FROM dimension_x@data_ws_uri_1 …
dimension_y@data_ws_uri_2

GROUP BY CUBE ...

(3) Subqueries to adapt source data to the dimensional schema

SELECT ...

FROM

(SELECT a, b, c FROM data_col@data_ws_uri_1) as dimension_x

(SELECT c, d, e FROM data_col@data_ws_uri_2) as dimension_y

GROUP BY CUBE ...

The possibilities (2) and (3) from the previous listing assume that OLAP query engine has the ability to produce, through some pre-compiling process, the dimensional meta-data needed further in the OLAP query processing (see the bellow paragraphs). The @ symbol is used in order to mark the external (not registered) WS data sources. In this respect one could use even a more advanced conventional SQL-sub-clauses to introduce an XML formatted URL or message instead of the actual alias of the registered analytical data source.

The computing model of the autonomous OLAP Web Services implies some distinct characteristics:

- It will dynamically and transparently invoke external-data-links into the OLAP declarative queries.

- It will have temporary buffer the original data sets to feed OLAP engine processing.

- It will have temporary buffer to store the analytical resultset to support active SDO collections in the context of large amount of analytical data expecting to be delivered to the OLAP WS consumer.

These characteristics could cause some serious performance issues concerning:

- Query compilation time which is dependent on the internal repository performance and the internal computing power necessary to parse SQL-OLAP text;

- Query execution time dependent on:

  - The data-flow access time, in fact it is about the data storage performance of the analytical data provider and about the network through-output;

  - Processing algorithms to manage data fetched with external data flows.

Also this OLAP-AWS computing model is fully compatible with the cloud computing approach. Consequently, this computing

model has also the potential not only to outsource the computing resources, but also to capitalize a new and grid-based computing architecture.

Feasibility of OLAP-AWS Architecture

The logical and physical architecture of our autonomous OLAP Web Services consists in that conceptual and concrete combination of software components, together with their residence context, that will address to the issues outlined in the previous section.

In the following, we will briefly expose what we consider to be a feasible solution resulted from our architectural analysis.

At logical level, we differentiated the following types of components that will deliver the (relatively) autonomous functionality that could be easily integrated in a standardized way as a Web Services Infrastructure:

- Analytical data providers;
- Dimensional data adaptors and wrappers;
- (optional) dimensional data storage (Data Warehousing);
- OLAP autonomous engine;
- OLAP data consumers.

To prove the feasibility of the previously proposed logical architecture, we have used a tripartite cloud infrastructure based on:

- ***Amazon Cloud*** that hosts their own Web Services accessible through Product Advertising API;

- ***Oracle (demo) Cloud*** that hosts their own APEX Platform which in its turn will host our OLAP-Web Service prototype;

- ***Google Cloud*** that hosts their own Google Docs Apps/Drive, which is quite suitable for business oriented end users working on a daily basis with web and cloud-based data sources (analytical data sources in this case).

Our experimental intention was to perform some analytical queries on data from Amazon Product Catalog, data that could be acquired by invocation of the Amazon Web Service accordingly with the Amazon Product Advertising API. The Amazon Web Service could be invoke with a classical SOAP request, as the one listed in the following script.

<!-- Listing : Amazon Product Advertising API Request -->

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://webservices.amazon.com/AWSECommerceService/2011-08-01">

<soapenv:Header/>

<soapenv:Body>

<ns:ItemSearch>

<ns:AWSAccessKeyId>?</ns:AWSAccessKeyId>

<ns:Timestamp>?</ns:Timestamp>

<ns:AssociateTag>?</ns:AssociateTag>

<ns:Signature>?</ns:Signature>

<ns:Shared>

<ns:SearchIndex>?</ns:SearchIndex>

</ns:Shared>

<ns:Request>

<ns:Title>?</ns:Title>

</ns:Request>

</ns:ItemSearch>

</soapenv:Body>

</soapenv:Envelope>

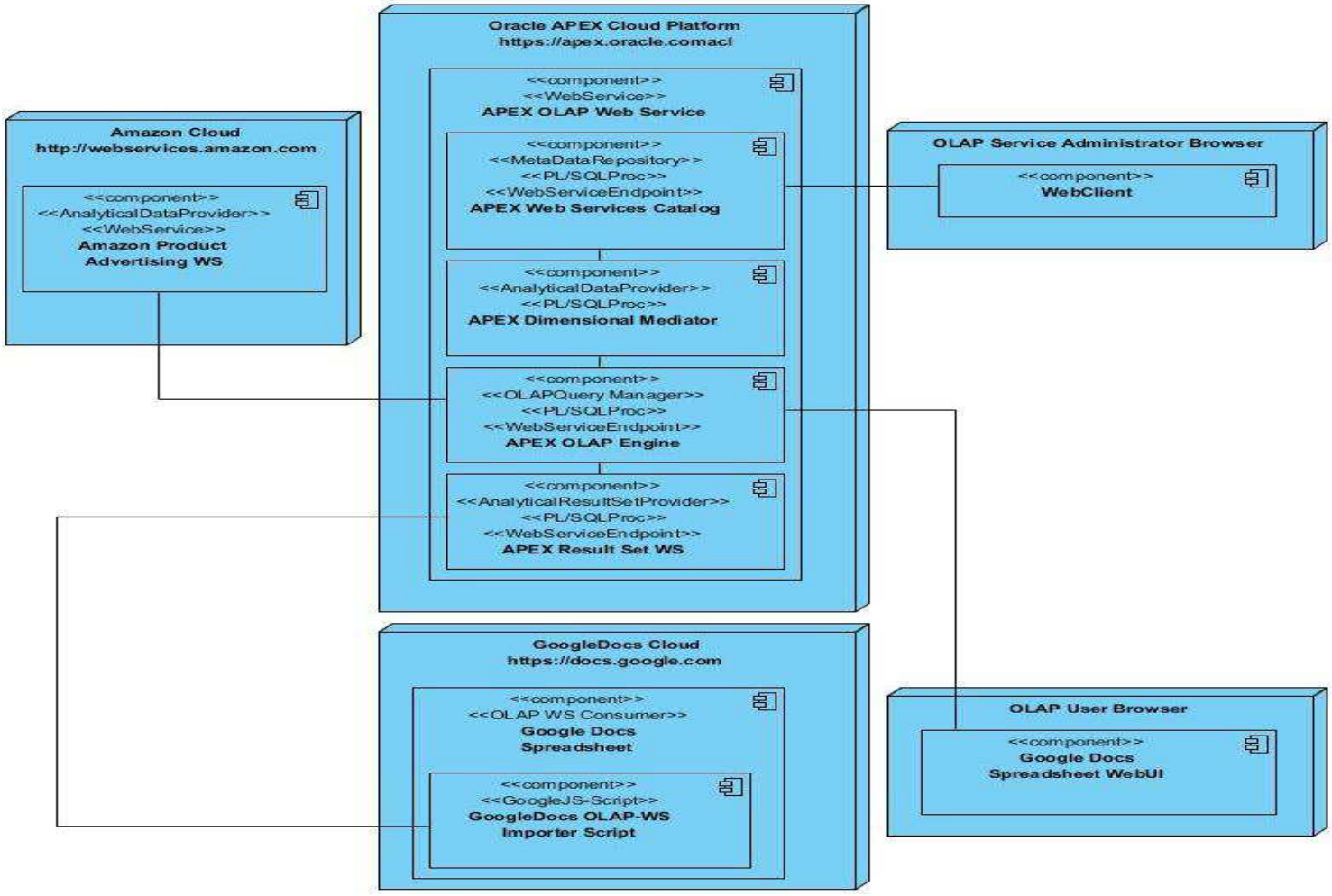

Fig 3. OLAP WS Proposed Architecture

The response will contain a collection of items as the following template suggests.

<!-- Listing 3: Amazon Product Advertising API Response -->

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:aws="http://webservices.amazon.com/AWSFault/2005-15-09">

<soapenv:Body>

<TotalResults>?</TotalResults>

<TotalPages>?</TotalPages>

<Item>

<ASIN>?</ASIN>

<ItemAttributes>

<Manufacturer>?</Manufacturer>

<ProductGroup>?</ProductGroup>

<Title>?</Title>

</ItemAttributes>

</Item>

<Item>?</Item>

<Item>?</Item>

<Item>?</Item>
… … …

</soapenv:Body>

</soapenv:Envelope>

Therefore our OLAP infrastructure count on the Amazon Web Service infrastructure, but does not intent to change or to deploy a specific functionality to the Amazon-Cloud (internal) platform. It will just reuse the Amazon Product Advertising Web Service

that could be invoked after a minimal subscription procedure. In fact, the Amazon platform, through this Web Service, will act as a Data Provider for the dimensional mediator component of our infrastructure, see figure 3.

The main platform that will host the concrete OLAP-WS implementation is **Oracle Application Express**. The ORACLE company offers an experimental cloud context at [https://apex.oracle.com] site, and, also, offers commercial hosting on their official public cloud site [https://cloud.oracle.com/mycloud/f?p=service:database:0:::::]. Oracle APEX may not be the only option to potentially support OLAP Web Services, but has some unique advantages that will make it very convenient for our approach:

- It has an extreme integration with Oracle database context (SQL or PL/SQL).

- It owns a specific PL/SQL libraries that allow direct integration of database components with the Web access protocols, including those used by Web Services infrastructure;

- It has a very capable web application development environment running on top of APEX platform as the web applications that will produce.

Also, the integration of the APEX web applications with Oracle Databases has another major advantage: the possibility to use the Oracle (11GR2) XML DB native web services infrastructure so that any PL/SQL procedure could be published as a full-

featured Web Service, or even an Oracle schema could be queried using a Web Service request.

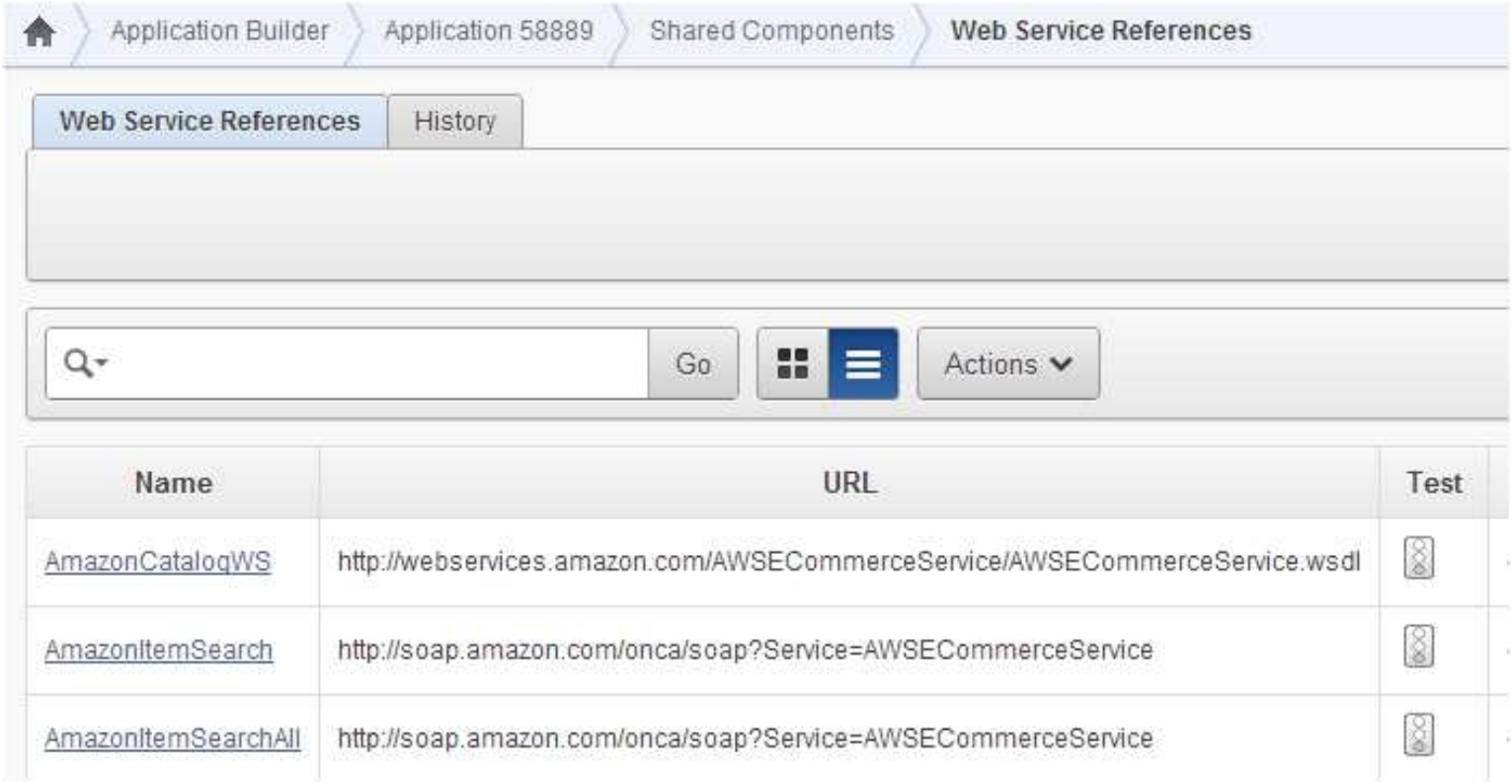

Fig 4. Oracle APEX Web Services Repository

finally deliver OLAP functionality, same as Web Services will have the following package of components (figure 3):

- Every external data provider will be registered as a *Web Service Reference* in *Shared Components* catalog (Dimensional Link Repository).

The APEX environment (or the APEX specific library system) provides the necessary APIs to invoke the original referenced Web Services and to convert the invocation result into a PL/SQL. Our specific OLAP web application assumes some special kind of components that will make accessible these components as some veritable object relational views, as in listing 4, ready to be invoked in any SQL-SELECT compatible phrases.

/* Listing 4: Template of SQL phrase to define OR Views Object

In order to access Web Service Result from SQL context */

```
CREATE WS_DIM_VIEW as

SELECT * FROM TABLE(ws_wrapper_fn(ws_ref))
/
```

- As we already mentioned, the analytical data provider may not "supply" their operational data as being already compatible with the desired dimensional schema; consequently, it may be necessary to create some special PL/SQL components that will refactor these virtual data collections (views) in order to

seamlessly integrate them into an analytical dimensional model, like star schema (Dimensional Data Adapter/Wrapper).

- Oracle native SQL-OLAP extensions are, in fact, the main reasons to ground this OLAP-WS architecture on the APEX infrastructure. Therefore our APEX application has a relatively simple component that will receive the original WS-SQL query from WS consumers, and they will make some simple parsing action, as the ones necessary to bind WS-Refs mentioned above to the internal OR views that will access them and will transform their resultsets into Oracle collections. Finally they will invoke the Oracle native SQL-OLAP engine and will manage the queries' result sets (using a very "delicate" buffering system) in the course of delivering analytical data to the next component (OLAP Query Manager).

- The last fundamental component of our APEX application is a PL/SQL procedure that will be published as Web Service endpoint and that will have the role to receive the analytical resultset from previous component, to convert these data into an XML document, and to deliver them to the analytical Web Service Consumer (WS Spec: endpoint for analytical resultset).

The last platform that we have mentioned in order to sustain our OLAP web-based edifice is the Google Docs (Cloud) Apps/Driver. As the final mediator in delivering the analytical data to the business user, Google Docs Platform has some valuable advantages:

- It is native Web oriented, so that any Google Doc is, in fact, a web resource with a specific URL associated.

- It has a rich scripting environment (based on Java Script) that include the libraries necessary to invoke and interpret from a Google Docs application any other web resource, like Web Services.

A very short snippet to invoke a OLAP-WS service could look like this:

Fig 5. Google Script to Get and Process an Analytical Resultset from an OLAP WS

Conclusion

In this workpaper we have tried to argue an open and web-based architecture where OLAP web services prove to be the cornerstone of the entire technological edifice. The main advantages of such components as OLAP web services consist in the openness and flexibility concerning interoperability and modularization on cloud computing platforms.

The Cloud-based Web Services in general, and, consequently, the OLAP Web Services also, go beyond the “universal” interoperability advantage by braking/cracking organisational information system boundaries. It is about to give more control to service users, in fact this is the story of the Internet paradigm. Ultimately, the next virtual enterprises or organizations will be

about freely combining web-services components into some very agile and ad-hoc business processes. This new business democracy involves at least two requirements: as much control as possible the end user services and, also, as much autonomy as possible to the concrete (deployed) service used.

Concerning only OLAP Web Services, at least two opportunities could be foreshadow:

- One could produce analytic information (web analytics) on some aggregated e-commerce industry areas in order to sell analytical industry reports/studies. This kind of applications will be based on a predefined Dimensional-Adaptor-Plug-in-System that will take into account the main commercial or public data providers (Amazon, Wallmart, AppleStore,

GooglePlay, Google Webstore, etc.), in fact will be like a minimal ETL tool.

- One could build a highly customized OLAP infrastructure but with a completed outsourced ETL. This kind of services will register Private Dimensional Data Sources with the predefined format protocol (SDO dimensional data format) and could exploit cloud-ready services to deploy enterprise analytics so that one could buy a cloud-based-OLAP-computing model which could acquire more computing power than from the in-house servers.

Acknowledgement

This work was supported by the project "Post-Doctoral Studies in Economics: training program for elite researchers - SPODE" co-funded from the European Social Fund through the Development of Human Resources Operational Programme 2007-2013, contract no. POSDRU/89/1.5/S/61755.).

References

Antonioletti, M., Krause, A., Paton, N. W., Eisenberg, A., Laws, S., Malaika, S., Melton, J. & Pearson, D. (2006). "The WS-DAI Family of Specifications for Web Service Data Access and Integration," *SIGMON Record,* Vol.35, No.1, 2006; p.48-55.

Celko, J. (2006). "Joe Celko's Analytics and OLAP in SQL," *Morgan Kaufmann Publishers, by Elsevier Inc.*

Dospinescu, O. & Perca, M. (2011). "Technological Integration for Increasing the Contextual Level of Information," *Analele Stiintifice ale Univesitatii Al. I. Cuza din Iasi, Stiinte Economice,* vol. LVIII, 2011, ISSN: 0379-7864, pp. 571-581.

Giordano, A. D. (2011). "Data Integration Blueprint and Modeling: Techniques for a Scalable and Sustainable Architecture," *IBM Press.*

Hümmer, W., Bauer, A. & Harde, G. (2003). "XCube – XML for Data Warehouses," *Proceedings of DOLAP'03, November 7, New Orleans, Louisiana, USA.*

Kozielski, S. & Wrembel, R. (2009). Editors, "New Trends in Data Warehousing and Data Analysis," *Springer Science+Business Media, LLC.*

Neale, M. (2009). "Amazon AWS Signatures," [Online], Oct.2009, Available: http://matthewneale.net/2009/10/11/amazon-aws-signatures

Pardillo, J., Mazón, J. N. & Trujillo, J. (2008). "Bridging the Semantic Gap in OLAP Models: Platform-Independent Queries," Proceeding of the ACM 11th international workshop on Data warehousing and OLAP, pg. 89-96.

Park, B. K., Han, H. & Song, I. Y. (2005). "XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses," A Min Tjoa and J. Trujillo (Eds.): DaWaK 2005, LNCS 3589, pp. 32–42, 2005., *Springer-Verlag Berlin Heidelberg.*

Pavaloaia, V. D. (2009). "Web Based Application for SMEs Economic and Financial Diagnose," In Innovation and Knowledge Management in Twin Track Economies: Challenges & Solutions, VOLS 1-3, The 11th International-Business-Information-Management-Association Conference JAN 04-06, EGYPT, Published by Int Business Informat Management Assoc, ISBN 978-0-9821489-0-7

Portier, B. & Budinsky, F. (2004). "Introduction to Service Data Objects. Next-Generation Data Programming in the Java

Environment," *IBM developerWorks;* http://www.ibm.com/developerworks/java/library/j-sdo/

Resende, L. (2007). "Handling Heterogeneous Data Sources in a SOA Environment with Service Data Objects (SDO)," SIGMOD '07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of data, ACM, New York, NY, USA, 2007, pp. 895–897.

Strîmbei, C. (2010). 'Modern Architecture Proposal For Business Intelligence/Reporting Software Systems,' Proceedings of The 4th International Conference Globalization and Higher Education in Economics and Business Administration – GEBA 2010, Ed. Univ.Al.I.Cuza, Iasi.

Strimbei, C. (2012). 'Data Integration Architecture Using Web Service Data Objects,' INSODE 2012, *Elsevier Procedia Technology.*

Thomsen, E. (2002). OLAP Solutions: Building Multidimensional Information Systems, *Second Edition, John Wiley & Sons, Inc, New York, USA.*

Varga, M. (2002). "A Procedure of Conversion of Relational into Multidimensional Database Schema," *Journal of Computing and Information Technology* - CIT 10, 2002, 2, 69–84.

Wrembel, R. & Koncilia, C. (2007). Editors, Data Warehouses and OLAP: Concepts, Architectures and Solutions, *IRM Press* (an imprint of Idea Group Inc.): Adzic, J., Fiore, V., Sisto, L. Extraction, Transformation, and Loading Processes, pg. 88-110.

Yu, L., Huang, W., Wang, S. & Lai, K. K. (2008). "Web Warehouse – A New Web Information Fusion Tool for Web Mining," *Information Fusion* 9 (2008), p. 501–511.

***, "Google Apps Script Overview," [Online], Sept.2012, Available: https://developers.google.com/apps-script/overview

***, "Google Chart Gadgets," [Online], Sept.2012, Available: http://support.google.com/docs/bin/answer.py?hl=en&answer=99488

***, "Product Advertising API Signed Requests Sample Code - Java SOAP," [Online], Sept.2012, Available: http://aws.amazon.com/code/Product-Advertising-API/2479

***, "Using Native Oracle XML DB Web Services," [Online], Sept.2012, Available: http://docs.oracle.com/cd/B28359_01/appdev.111/b28369/xdb_web_services.htm