A*DAX: A Platform for Cross-domain Data Linking, Sharing and Analytics

Narayanan Amudha, Gim Guan Chua, Eric Siew Khuan Foo, Shen Tat Goh, Shuqiao Guo, Paul Min Chim Lim, Mun-Thye Mak, Muhammad Cassim Mahmud Munshi, See-Kiong Ng, Wee Siong Ng, Huayu Wu

Institute for Infocomm Research, A*STAR, Singapore
{naraa, ggchua, foosk, stgoh, guosq, limmc, mtmak, mcmmunshi, sknq, wsng, huwu}@i2r.a-star.edu.sg

Abstract. We introduce the A*STAR Data Analytics and Exchange Platform ("A*DAX"), which is the backbone data platform for different programs and projects under the Urban Systems Initiative launched by the Agency for Science, Technology and Research in Singapore. The A*DAX aims to provide a centralized system for public and private sectors to manage and share data; meanwhile, it also provides basic data analytics and visualization functions for authorized parties to consume data. A*DAX is also a channel for developers to develop innovative applications based on real data to improve urban services.
In this paper, we focus on presenting the platform components that address challenges in data integration and processing. In particular, the A*DAX platform needs to dynamically fuse heterogeneous data from unpredictable sources, which makes traditional data integration mechanisms hard to be applied. Also, the platform needs to process data with different dynamics (i.e., database vs. data stream) in large scale. In our design, we use a semantic approach to handle data fusion and integration problems, and propose a hybrid architecture to process static and dynamic data to answer queries at the same time. Other issues about the A*DAX platform, e.g., security and privacy, are not covered in this paper.

1 Introduction

1.1 Background

Cities around the world are growing at an extraordinary pace, with the current population of 1.7 billion urban dwellers expected to grow to 2.2 billion by 2020. This translates to an increase of 1 million people living in cities every week - particularly cities in emerging regions like Asia and Africa. The challenge of all cities, regardless of its stage of development, is to be able to grow sustainably, create job opportunities for its dwellers and ensure social wellness through efficient provision of infrastructure and services.

The Agency for Science, Technology and Research (A*STAR) of Singapore launched the Urban Systems Initiative in 2012, which is a five-year multi-program

initiative to address the new technological needs of the rapidly urbanizing world. It aims to enable the development of solutions for complex urban challenges in collaboration with the relevant government agencies and industries to enhance the competitiveness of Singapore in the construction of "smart city".

Under the Urban Systems Initiative, four inter-related programs have been launched to address these challenges:

Integrated Urban Planning To develop an integrated platform for quantitative and evidence-based urban planning.

Sense & Sense-abilities To develop a unified platform to "sense" and "make-sense" of the living environment in real-time.

Complex Systems To develop complex system theories and models to unravel the complexity of city dynamics.

City Logistics & Supply Chain To develop city logistics platforms based on complex systems approach and data analytics for addressing freight traffic congestion.

The four programs identify research problems, design solutions and implement infrastructures in each domain. Meanwhile, they are also dependent on each other for resource reuse, research achievement sharing and decision making.

Data is essential to all the programs under the initiative. In each domain, both problem identification and solution validation are supported by data, which are contributed by the government agencies, industrial partners and social media. Furthermore, another objective of the Urban Systems Initiative is to encourage companies, research institutions and individual developers to fuse data from different domains and come out with new applications or services for better social good.

To achieve the goal, large amount of social, economic, geographic, business and educational data that are collected by the public and private sectors need to be effectively merged and stored, efficiently accessed for use, and securely shared among programs and third-party developers. Motivated by this, the A*STAR Data Analytics and Exchange Platform ("A*DAX") is built. The A*DAX is a scalable and open standards based platform for data management, sharing and analytics. It is the backbone for the various projects under the Urban Systems Initiative.

1.2 Challenges

Designing the A*DAX platform faces difficult challenges in system, networking, security, etc. In this paper, we focus on the challenges in data management. The first challenge we need to resolve is integrating unpredictable source data. Traditional data integration [3] assumes the deterministic set of input database schemas, and applies either Local-As-View (LAV) or Global-As-View (GAV) approach to logically link up the source data and provide a unique view to data users. However, in our scenario, ideally more and more parties will dynamically join the A*DAX platform for data sharing. It is not possible to determine a

permanent unique view across source data. Thus, it is difficult to use the existing approach for schema mapping.

Another challenge is the combination of static data and dynamic data. Based on the current tenants of the A*DAX platform, some data are one-time loaded, while some data are continuously streaming into the system for storing. The platform needs an efficient solution to archive and to query two types of source data, so that the data are well consumed by different users.

1.3 Contribution and Organization

In this paper, we first depict the A*DAX platform with respect to different components for different functions. Then we focus on the several components that resolve challenging problems such as data fusion and static and dynamic data processing. We propose a metadata-level semantic graph to guide data fusion, and we show how the semantic graph is incorporated with new data source merging. For data processing across both static datasets and dynamic data streams, we adapt the λ-architecture and manage both view and materialized view for users to query the underlying data. This design benefits both the data owners and data users in access control and query issuing.

The rest of the paper is organized as below. In Section 2 we describe the A*DAX platform in a high level. In Section 3 we present how the platform adaptively manage and fuse source data. In Section 4, the data processing architecture is introduced. We introduce the access control mechanism in the A*DAX platform in Section 5. Finally, we conclude the paper in Section 6.

2 Platform Overview

Fig. 1 shows the logical architecture of the A*DAX platform. Basically, there are three layers for the platform, namely Storage Layer, Master Layer and Application Layer. The Storage Layer maintains several database management systems for different types of data. The relational database management system (RDBMS) is used for storing structured data. There is also an XML extension to the RDBMS to handle semi-structured data. The Geospatial DB is used for managing GIS data. To serve the urban planning, map-based GIS data processing and visualization is a main part of the platform. The Text DB archives text-based documents. It supports keyword-based search and retrieval. The platform also incorporates a Hadoop cluster to store large-scale datasets and provide parallel data processing. All the data storages are inter-connected and communicate with the View Manager in the Master Layer for data access.

The Master Layer contains all the major components of the platform. The Data Semantics Manager summarizes the semantics of the metadata of all data sources. It constructs a semantic graph, which can be dynamically extended as new data sources arriving, to link up all data sources and merge identical entities. It also guides the view generation to provide the interface for users to use the data from different sources holistically. The View Manager creates both materialized

and non-materialized views on top of the data storage. We require that users can only query the data store via provided views. There are two advantages of this restriction. First, the data can be better protected. The data providers will authorize and supervise the creation of views on their data. Thus they can choose what data can be showed to users. Second, it eases the management of pools of data. On one hand, the users do not need to worry about different query formats to different source data, as all queries will be interpreted against views rather than raw datasets; on the other hand, access control policies can be enforced on top of views so that it is easy to check whether a user query is legitimate to access the information it is interested in. The Query Manager accepts user queries, validates them and then passes them to the View Manager. The View Manager will do query parsing and transformation and forward sub-queries to different data storages for processing. More details about the Data Semantics Manager and View Manager will be discussed in the next two sections.

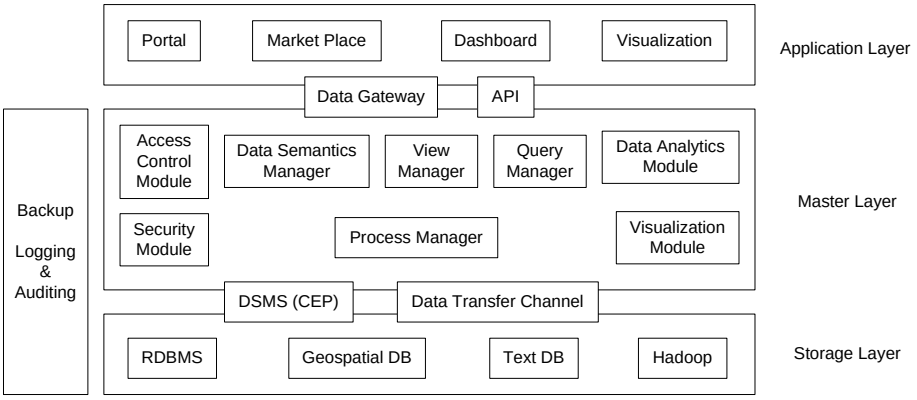


Fig. 1. Logical Architecture of the A*DAX platform

The Master Layer also contains the Data Analytics Module which provides classic data mining tools, e.g., for regression and classification, to users. The Visualization Module can visualize data and data analytics results to users. There are also Access Control Module and Security Module to ensure the security and privacy of the system. All the operations will be logged and audited.

Finally, the Application Layer provides user interfaces to upload data to the A*DAX platform, or to consume data. The platform also provides APIs for developers to use the data (by authorization) to develop innovative applications.

3 Data Fusion

The main purpose of the A*DAX is to provide a platform for different agencies to share their data. Despite the uniqueness of the data owned by each agency,

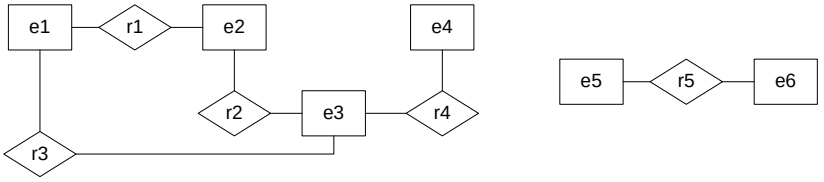
there are still many overlapped attributes among different data sources. These overlapped attributes will link up different datasets and then give opportunities for users to mesh them and come out with new insight. We call the process of linking up different data sources by their common attributes as *data fusion*.

As mentioned in Section 1, traditional data integration techniques are not proper for the data fusion in the A*DAX platform. The reason is that the number of data sources keeps increasing as more sectors join the platform. We need a more dynamic approach to extend the data fusion by keeping the existing fused data framework unchanged. In our platform, we propose to use a metadata-level semantic graph to reflect the linking between data sources.

The semantic graph is similar to the ER (Entity-Relationship) diagram [1] which models a relational database at conceptual level. There are two types of nodes in the semantic graph, i.e., entity nodes and relationship nodes. An entity node models an entity class of a data source. If two datasets from different sources have any common attributes, the related entity classes in the two datasets will be linked by a relationship node in the semantic graph, and the relationship node contains the information of the common attributes from the two datasets. Once a new dataset is added to the data sharing platform, it will be linked to the existing nodes in the semantic graph by common attributes, and keep the existing graph unchanged. Similarly, if a dataset is removed from the platform, only the relevant entity nodes and corresponding relationship nodes in the semantic graph are removed, without affecting other nodes.

In fact, we do not really require the owner of each dataset to identify entity class and to decompose their data. When a data owner tries to upload data to the system, he/she only needs to go through the schema of the existing datasets through the semantic graph, and tell the system which attributes in his/her dataset overlap with the attributes in the existing data store. Then the new dataset can be linked to existing ones in the semantic graph. In other words, each dataset can be treated as a whole as an entity class in the semantic graph. Fig. 2 shows an example semantic graph for the data sources from different government agencies.

In this example, let us assume that the Immigration & Checkpoints Authority (ICA) shares citizen data, the Inland Revenue Authority (IRAS) shares personal income tax data, the Housing Development Board (HDB) shares residential data and the Urban Redevelopment Authority (URA) shares land planning data in the A*DAX platform. There are many common attributes between different datasets, and the semantic graph can be constructed as shown. There is an entity table and a relationship table describing the semantic graph. In the entity table, the name of each entity dataset and its database location, owner and upload time are recorded. As a result, given an entity in the semantic graph that is queried by a user, from the entity table we can locate the physical storage of the dataset and then route the relevant sub-query (composed by the View Manager, as described later) to the data storage for processing. The relationship table tells how two entities are linked. For example, for r1, the NRIC attribute from the ICA citizen data can be matched to the ID attribute in the IRAS data.



Entity

ID	name	location	onwer	timestamp
e1	citizen	RDB.citizen	ICA	2008-01-01
e2	tax	RDB.tax	IRAS	2009-01-01
e3	housing	RDB.hdb	HDB	2009-01-01
e4	landuse	RDB.land	URA	2007-01-01
...	...	...	...	...

Relationship

ID	E1	E2	Linkable Attributes
r1	e1	e2	E1.NRIC ~ E2.ID
r2	e2	e3	E1.ID ~ E2.owner
r3	e1	e3	E1.NRIC ~ E2.onwer
r4	e3	e4	E1.postcode ~ E2.pc
...	...	...	...

Fig. 2. Logical Architecture of the A*DAX platform

The semantic graph and the description tables are managed by the Data Semantics Manager in the platform. It also communicates with the View Manager for creating views across different data sources, as well as the Query Manager for accepting new queries and finally translating the queries into sub-queries issued to different data sources.

4 Data Processing

As mentioned, the A*DAX platform can be used for sharing heterogenous data, e.g., data with different representation type and data with different dynamics. We need to resolve this data heterogeneity during data processing.

4.1 Mixed-type Data Processing

In the A*DAX platform, we support three types of data, i.e., structured data, semi-structured (XML) data and unstructured (text) data. As such, we support both structured query and unstructured query to the mixed-type data storage. Structured query can be issued to the structured and semi-structured datasets, via SQL and XQuery query languages (though we do not require users to master all query languages as discussed later). Unstructured query, i.e., keyword query can be used to either select relevant records in the structured and semi-structured databases, or retrieve relevant text documents.

4.1.1 Structured Query Processing

Although there are standard structured query languages for different types of data, i.e., SQL for relational data and XQuery/XPath for XML data, we cannot

prompt the platform users to specify queries in different languages to search different parts of data store. This is because users may be blind to the underlying data structures, and do not know the type of the data they are interested in. To solve this problem, we leverage on the views, which are a set of type-unified virtual representation of underlying data. Views are driven by user queries, and created under the supervision of owners of involved datasets. Views are the only data interface presented to users to let them know what data they can query. The internal relationship between a view and involved datasets in either relational format or XML format will be handled by the system. A user only need to issue SQL queries to the view, and the system will translate the query into sub-queries, in either SQL or XQuery to query involved physical datasets.

Continuing with the example data in Fig. 2, if a user would like to know the details of particular house owners, he/she may raise a request to the A*DAX. After getting approvals from the ICA and the URA, a view with citizen's particulars and their housing information will be created. The user can just issue SQL queries to the view, and the system will internally join the two datasets to process queries.

There are also materialized views, which will be discussed later. All views are managed by the View Manager.

4.1.2 Keyword Query Processing

Keyword queries can be issued to structured/semi-structured datasets, or text document store. If the keyword query is to search structured or semi-structured data, the query engine will return data records that contain query keywords and are ranked based on interpreted query intention. Both structured data and semi-structured data will be semantically indexed and searched by our algorithms. The detailed semantics-based keyword search algorithms are omitted in this paper. They can be found in [2].

If the keyword query is issued to the text database, an inverted list based document retrieval will be executed. Relevant text documents will be returned. We follow the existing document retrieval techniques, and omit the details in this paper.

4.2 Mixed-dynamics Data Processing

4.2.1 Data Storage

Data shared in the A*DAX platform can be either uploaded in batch to the system, or streamed into the system in real-time. For batch data, depending on the data size, data characteristics and the data owner's preference, the system will either store the data into databases, or the Hadoop Distributed File System. Basically, if the data size is manageable and the data will be used for OLTP-like query processing, the data will be stored in database. On the other hand, if the data is too large and only for BI-like analytical purpose, the data will be stored in the Hadoop system and processed by MapReduce.

For streaming data, a data stream management system (DSMS) will receive the data and perform complex event processing (CEP) over the data to answer continuous queries. Meanwhile, the data will also be archived in databases, unless the data owner denies this operation. The system will keep a size threshold for each data stream, and the database archive for each stream will be periodically transferred to the Hadoop system to release database space and ensure database query performance.

4.2.2 Query Processing

In this part, we talk about how the A*DAX processes queries over both static data and dynamic data. We adapt the λ-architecture [4] to design the query processing module for both static data and dynamic data in the A*DAX platform. Fig. 3 shows the architecture in the A*DAX for query processing.

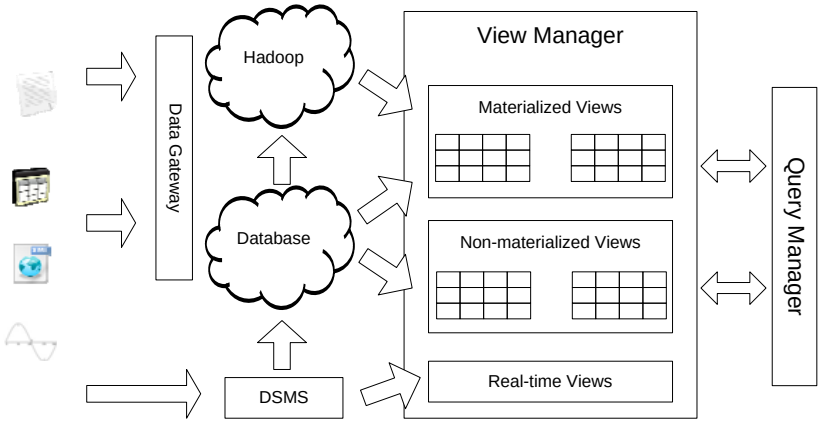


Fig. 3. Query processing over static and dynamic data in A*DAX

As shown in Fig. 3, we consider three layers of data storage. The DSMS system temporarily store the most recent set of streaming data in its memory, in order to process continuous queries registered in the system. Then it pushes all data to the database system. The database system is expected to execute SQL queries on-the-fly. Periodically, the database system will further archive old data in the Hadoop Distributed File System.

The DSMS offers a virtual real-time view to users to use the data streams. Users may issue continuous queries, in SQL format, to the real-time view, which will eventually register into the DSMS system for data filtering. There are also virtual views on top of the database system. As mentioned, such views may be constructed across different database instances, guided by the semantic graph in the Data Semantics Manager. Users may issue SQL query to search the datasets in the A*DAX platform via these views, and all queries will be eventually executed in the Storage Layer.

The materialized views are constructed based on approved analytical queries (such as aggregation queries) over the large-scale historical data (mainly) in the Hadoop system. In other words, materialized views store the result of analytical queries processed in the Hadoop system. Since data processing over large-scale data, e.g., using MapReduce in Hadoop is time consuming, we pre-cache processing results for certain frequently asked queries for users to perform efficient online analytical data processing and visualization. The data in the materialized views will be periodically updated as new data are archived in the Hadoop system. Note that the platform also allows users to program MapReduce jobs and process data on the Hadoop Distributed File System.

Let us assume the scenario that the HDB continuously sends house transaction data to the A*DAX, and another government agency would like to find out the highest transaction price for each month, after being approved to view such data. Assume the house transaction data are partially stored in both the Hadoop system and the database system, and a materialized view to summarize each month's transaction data stored in the Hadoop system was created. This query will be sent to all the three views. In the materialized view, historical summarized records can be easily retrieved. For the non-materialized view, the query will be executed against the database data, and return the result. During the data execution, the continuous query issued to the real-time view will monitor the most recent data, and compare the result with the database search result to find out the highest transaction price for the most recent month.

If the materialized view for the issued query does not exist, the query will be programmed as a MapReduce task and executed in the Hadoop system. In this case, the Hadoop execution will be long, and probably dominates the overall query processing time. Then the continuous query issued to the real-time view becomes significant, which ensures that no data is missing during the query processing.

Finally, the query results from all views will be merged and post-processed, if necessary, and returned to the user.

5 Access Control

The A*DAX platform involves a lot of sensitive data from public and private sectors in Singapore. The security and privacy of the system is crucial. The Security Module of the platform guarantees the system security using cryptographic techniques. In this section, we focus on privacy control.

Each data provider has a right to control the access to his/her data. Since the data in the platform are very sensitive and the platform users can be quite diverse with different levels of profiles, we do not follow the typical role-based access control model. Instead, we provide the channel and require each data user to get approval from the data owner before he/she can access the data. The data owners do not need to specify any access control policy on the data. They will evaluate each data access request in ad-hoc manner, and grant the access or partial access to their data based on data users' profiles and access purposes.

Because the A*DAX shares data among government agencies and big companies, rather than in individual level, the number of users is limited and the workload for ad-hoc access approval is not high.

The Access Control Module of the A*DAX platform maintains the access privilege of each user, and compiles it against the views to make decision that whether the user is allowed to access the queried attributes of each view.

For example, a user is querying the view contains citizen particulars and housing information, i.e., the view across the ICA and the HDB's data. Suppose the user is granted the full access to the citizen data by the ICA, and granted partial access to the housing data on which only the transaction price for each house is not accessible. If the user query does not involve the attribute of transaction price, the query will be processed. Otherwise, the query will be rejected.

6 Conclusion

In this paper, we introduced the A*DAX platform, a platform for cross-domain data linking, sharing and analytics for public and private sectors in Singapore, to improve urban planning and development. We described the general framework of the platform, and focused on the components that handle data storage, data fusion and query processing. We designed an adaptive semantics-based metadata to guide the system integrating data from unpredictable data sources. Furthermore, we designed a system architecture based on the λ-architecture to process heterogeneous data in the A*DAX data storage. In particular, our architecture provides virtual and materialized views between the data storage and the query engine. It offers convenience for users to issue queries in unique format, and it also provides a way for data owners to control the disclosure of their data to users.

Acknowledgement

This work was supported by the A*STAR SERC Grant No. 1224604057 and 1224200004.

References

1. P. P. Chen. The ER model: toward a unified view of data. In ACM Trans. Database Syst., 1(1):9-36, 1976.
2. T. N. Le, H. Wu, T. W. Ling, L. Li, J. Lu: From Structure-based to Semantics-based: Effective XML Keyword Search. In ER, pages 356-371, 2013.
3. M. Lenzerini. Data Integration: A Theoretical Perspective. In PODS, pages 233-246, 2002.
4. N. Marz and J. Warren. Big Data - Principles and best practices of scalable realtime data systems. MEAP Began, 2012.