

Tree Graph Views for a Distributed Pervasive Environment

Tuyêt Trâm Dang Ngoc¹ and Nicolas Travers²

¹ ETIS Laboratory - University of Cergy-Pontoise, France.

`Tuyet-Tram.Dang-Ngoc@u-cergy.fr`

² PRISM Laboratory - University of Versailles, France.

`Nicolas.Travers@prism.uvsq.fr`

Abstract. The pervasive Internet and the massive deployment of sensor devices have lead to a huge heterogeneous distributed system connecting millions of data sources and customers together [Fra01]. On the one hand, mediation systems [BGL⁺99,DNJT05] using XML as an exchange language have been proposed to federate data accross distributed heterogeneous data sources. On the other hand, work [MSFC02,AML05,BGS01,NDK⁺03] have been done to integrate data from sensors. The challenge is now to integrate data coming from both "classical" data (DBMS, Web sites, XML files) and "dynamic" data (sensors) in the context of an ad-hoc network, and finally, to adapt queries and result to match the client profile. We propose to use the TGV model [TDNL06,TDNL07a] as a mobile agent to query sources across devices (sources and terminal) in the context of a rescue coordination system. This work is integrated in the PADAWAN project.

Keywords: XQuery evaluation, Tree Graph View (TGV), Pervasive environment, Rescue Coordination

1 Introduction

The pervasive Internet and the massive deployment of sensor devices have lead to a huge heterogeneous distributed system connecting millions of data sources and customers together.

On one side, data sources are heterogeneous as they can be of different types (relational, text, XML, streaming value, etc.) and can have different update frequencies (from "never" for some text document to "always" for sensors value) and their autonomy (from non-manageable obfuscated black box that just provide values to a full access management on a DBMS). On the other side, there can be different profiles of clients: access permission, terminal capabilities, user preferences, etc.

To deal with distributed heterogeneous and autonomous data sources, mediation systems have been widely studied [Wie92,MFK01,NGT98,BGL⁺99,DNJT05]. Such mediation systems [Wie92] provide a uniform interface to a multitude data

sources using mediators and wrappers to handle respectively distributivity and heterogeneity.

XML [BPSM98], has become the preferred format to represent semi-structured data [Abi97] and an effective way to define any type of data that can be represented as a tree.

Moreover, XQuery [W3C05] has proved to be an expressive and powerful query language to query XML data both on structure and content, and to make transformation on the data. In addition, its query functionalities come from both the database community (filtering, join, selection, aggregation), and the text community (supporting and defining function as text search).

TGV (Tree Graph View) [Tra06,TDNL06,TDNL07a] is a Tree Pattern-based model (such as TPQ [CJLP03] and GTP [AYCLS01]) to model XQuery queries. This model is suitable to our needs since :

- it supports the complexity of the full untyped-XQuery specification: relational and set operator, aggregation, ordering, nested reconstruction, conditional predicate, etc.
- it is designed for a mediation context accessing to distributed autonomous and heterogeneous data sources: its structure identifies data collections and dependencies between them. An annotation model in layers allows to annotate any piece of information (location of the source(s), cost model, etc) that can be useful a evaluation time. Finally, transformation rules have been defined to optimize and evaluate the TGV to produce the result.

The rest of this paper is organized as follows. To start off with, we motivate the need for a mobile semi-structured model in a pervasive environment in Section 2 and we express issues and related works in Section 3. Further, we recall the TGV model and functionalities in Section 4 and show how it is suitable to our context. We then present some extensions to the TGV model that would make it more suitable to a pervasive environment (Section 5). In the end, we conclude in Section 6 and present future directions of our work.

2 Context

In the global context of our work, different types of data sources and terminal client are dissiminated all over a network consisted of traditional IP routing and addressing (e.g. the Internet), and ad-hoc routing scheme.

2.1 Motivational Scenario

The application scenario is the deployment of a Rescue Coordination Center after or during a disaster (fire, earthquake, flood, etc.) A truck (Figure ??) carries the PADAWAN proxy, some access points (wire, wireless WI-FI, SINK, etc.) and an Internet Access.

Using these gateway access points, the PADAWAN proxy can reach three types of networks:

- *Wireless ad-hoc Sensor Network [Toh01] (ad-hoc WSN)*: Sensors are deployed (eg. from helicopter or embedded in the rescue team equipment) in the monitoring area to form a WSN. The entry point is a *sink* (a node with large resources that collects statistics from nodes in its coverage and generally remains in a static location).
- *private network and/or ad-hoc network*: used by the rescue team (firemen, officers, doctors, emergency unit)
- *the Internet*: to access databases, web site, directories, etc.

Using such a system, depending of their profile and access rights, not only the members of the rescue team, but the experts and the press can access to an integrated view on all sources related to the disaster and query it.

2.2 PADAWAN

The rescue scenario relies on the PADAWAN (a Proxy for All Devices Accessing the World And Neighborhood) infrastructure. The PADAWAN infrastructure is seen as a graph where links are networks links (radio or wire), and nodes are either data sources, client terminals or the PADAWAN proxy itself.

Data Sources We define by **source**, any device that provides data: DBMS, sensors, web sites, RSS feeds, legacy applications, directories, files, etc. We consider different kinds of (non-exclusive) heterogeneity:

- *data type*: sources can have a relational, text-based, semi-structured, unstructured model. The query language can differ : SQL, XQuery, OQL, contains function, http request, etc. and so the result format : tuple, XML, objects, text lines, text documents, etc.
- *autonomy*: except for some databases managed by the PADAWAN administrators themselves, the major number of sources are autonomous and just communicate what their owner want them to communicate. They mainly just provide data to reply to requests more or less supported, and rarely give their internal information (cost model, data statistics, schema, etc.)
- *update frequency*: some data sources can be considered as static or semi-static sources where data do not change frequently, eg. Web pages, LDAP directories, some databases. Other data sources are considered as dynamic, as their information update very frequently or on each request (RSS feed, sensors measure).

Clients All kind of clients can access to PADAWAN to query the sources: from a simple pager to a complex application manipulating huge volume of sources. Available data and results are processed using specific views based on client profile. These views are computed depending on:

- *user preference*: eg. the firemen officer want to know about the temperature measured by each sensors, the press just needs to access the average temperature of the site.

- *user access rights*: eg. the rescue team has access to the personal medical information of injured people, others don't.
- *client terminal capabilities*: eg. the cell-phone used by the firefighter does not have the same display capabilities as the laptop computer of the rescue coordination officer, so large volume of information, images and video is not applicable in every case.

The PADAWAN Proxy The core of the system is the PADAWAN proxy (that is located in the rescue truck). This proxy is a mediator system collecting and requesting all available data coming from deployed sensors, and embedded systems used by the rescuers received via the access points, and also from databases and RSS feeds from the Internet.

The Figure 1 shows a query Q and its XQuery representation. We suppose that, for example, information on building occupation is stored in a relational DBMS located on an Internet site, and that sensors are deployed over an ad-hoc sensor network reachable by a sink access point.

Let Q be the query that "list every buildings occupied by more than 100 inhabitants, and for each, get the district and the list of maximum temperature measured by the sensors located in the same district."

| XQuery Request | XML Result |
|--|--|
| <pre> for \$a in //buildings/building where \$a/description/inhabitant > 100 return <districtMonitoring> <location> {\$a/district} </location> <temperatures> { for \$b in //sensor where \$b/deploymentArea/district = \$a/district return <temperature> {\$b/max_temp} </temperature>} </temperatures> </districtMonitoring> </pre> | <pre> <districtMonitoring> <location> Yellow Lake </location> <temperatures> <temperature> 14 </temperature> </temperatures> </districtMonitoring> <districtMonitoring> <location> Green Valley </location> <temperatures> <temperature> 163 </temperature> <temperature> 25 </temperature> <temperature> 43 </temperature> </temperatures> </districtMonitoring> </pre> |

Fig. 1. Query, XQuery request for Query Q and XML result

3 Issues and Related Works

Mediation systems [BGL⁺99, DNJT05] based on mediator/wrappers architecture [Wie92] using XML as an exchange language have been proposed to federate data across distributed heterogeneous data sources. The heterogeneity of the data type is handled by *wrappers* that act as "translators" from the source native query language and result to the common query language and model used by the mediator. The mediator decomposes the user into subqueries sent to wrappers, and recomposes the final result, and thus, manages the distributed aspect of the system.

Many work [MSFC02, AML05, BGS01, NDK⁺03] have been done to integrate data from sensors. The first type of approach considers the sensor network as a

virtual [MSFC02] or materialized [BGS01] relational table. The second type of approach as in the IrisNet [NDK⁺03], considers the web as a "huge" XML document, using DNS extension to locate nodes of the XML document. However, these works are not designed to be integrated with other DBMS in a heterogeneous environment, using queries with complex functionalities.

The challenge is how to evaluate XQuery across the graph composed of heterogeneous sources, heterogeneous clients and heterogeneous infrastructure.

We propose to use the TGV model to deal with this problem. We recall the TGV basis in the next section, and show how it is suitable to our needs.

4 Tree Graph View (TGV)

TGV (Tree Graph View) [Tra06,TDNL06,TDNL07a] is a Tree Pattern-based model (such as TPQ [CJLP03] and GTP [AYCLS01]) designed to represent XQuery request and its evaluation. The TGV supports all the functionalities of untyped-XQuery, uses an intuitive representation compliant with mediation issues, and provides a support for optimization and information.

4.1 TGV example

This TGV representation of the XQuery Q (Figure 1) is shown on Figure 2 (a). The tree patterns of the two data collections of the query are shown in circles. The intermediate and final result construction are represented within boxes. Dependencies are shown by hyperlink lines binding the patterns or the pattern nodes: join on the two source tree patterns and projection from one node or tree pattern to another. The nested `temperature` in `temperatures` is also supported (the \$t box).

Fig. 2. (a) TGV of the Query Q - (b) TGV Annotation Layers View

This type of representation is suitable to a mediation system as subpart of the requests can be identified and also dependencies between them. The TGV model has been fully formalized using Abstract Data Types in [Tra06].

4.2 Annotation

Set of elements of the TGV can be annotated for (a) any *granularity* of information and (b) any *type* of information (cost models and statistics, location, constraint, accuracy, security, rule tracability).

Using annotation, a TGV can be viewed on any type of annotation that has been defined on it. On Figure 2 (b), the original logical TGV ④ has three annotated views:

- ① *location annotation*: each location of the execution of subparts of the TGV is reported on the associate set of TGV elements on the TGV. *In our example, the information on buildings can be retrieved from a source Source1 that is a DBMS accessible from the Internet. The information on sensors are retrieved from Source2 and Source3 that are located on two SINKS respectively located on the Yellow Lake and on the Green Valley. The other parts of the TGV are evaluated by the mediator, located on the PADAWAN proxy.*
- ② *time cost annotation [LDNL07]*: the time cost execution are annotated on subparts of the TGV. The time costs are evaluated using cost models.
- ③ *evaluation annotation*: This annotation layer is used to evaluate the TGV. The evaluation annotations are intermediate or final results that have been evaluated on subparts of the TGV. The evaluation process is described in the subsection 4.4 ("evaluation").

The annotation specification [TDNL06,LDNL07] are generic enough to annotate any subpart of the TGV, including money cost, energy cost (battery for sensors), accuracy, etc.

4.3 Transformations

In [TDN07], a pattern-based language for extensible rules has been defined for TGV. The subpart of the TGV that match the rule condition pattern is transformed into another pattern by applying transformation rules. Annotations can also be considered in the rule condition. For instance, during the optimization phase, cost annotation can be used to generate better plan: the rule condition can express that if the estimated cardinality of left side of an bind-join is much lesser than the cardinality of the right side, than the rule will invert each side of the bind-join for better performance during the evaluation.

4.4 Evaluation

A particular category of transformation rules is the evaluation rules category, which evaluate subparts of TGV using evaluation annotation. A TGV with empty evaluation annotation is considered as an execution plan. Matching sources fill the evaluation annotation with data matching the recognized patterns. Then, using iteratively evaluation rules, the annotations are propagated in the TGV. At the end of the evaluation, process, the whole TGV is annotated with the result of the query. The Figure ?? shows the evaluation process steps. Starting from the TGV Query on Figure 2 with empty evaluation annotations, the Source Tree Patterns are annotated (7a) with matching information retrieved from the appropriate sources (using location annotation). Then the transformation rule matching the join hyperlink apply (7b), then the aggregation rule (7c) and finally, the projection rule annotates the whole TGV with the evaluation annotation containing the final result (7d) that can be returned as an XML document (Figure 1 (left column)).

A similar evaluation approach has been developed in Mutant Query Plan [PMT03].

5 TGV in a Pervasive Environment

5.1 TGV Mobile Agent

The TGV evaluation by transformation rules on annotations is very well adapted to a mobile agent platform. The TGV Mobile Agent move between nodes across the graph. Each node of the graph:

- applies evaluation rules on the matched TGV, to fill evaluation annotations
- reads location annotations to route the TGV to the next concerned node

Query processing using Mobile Agent -but in relational context- has been developed in the work of [MHMM05]. Using the TGV model, mobile agent would be able to query on semi-structured distributed data, and thus be used in a heterogeneous environment.

5.2 Views on TGV

A client profile is represented by a view. A view is a request, and so can be represented by a TGV. Each client has a view constructed from the user preference, the user access permission and the client terminal capabilities. The profile is modeled as a view (a TGV request) that is applied to the client request (another TGV request). Such TGV merging has been developed in [DNGT04] via mapping.

5.3 TGV* (TGV Star)

When a source connects to the PADAWAN proxy, the associated wrapper send the source description to the proxy. The source description is also a TGV with as many annotation views as different types of information provided. As this TGV is a TGV where all paths of supported tree patterns (as in dataguide [GW97]) are annotated, this TGV is called TGV* (TGV star). All TGV* are sent by wrappers to the PADAWAN proxy and merged to make a big TGV* on the proxy. This TGV* is then used by the proxy to annotate TGV user request with information about location, cost, etc.

6 Conclusion

In this article, we have presented the TGV as a model suitable for distributed evaluation of an XQuery request over a pervasive environment, using a rescue coordination scenario.

- as a representation [TDNL07b] of the full-untyped XQuery specification, the TGV inherits of all the power of the XQuery language (relational and set operator, aggregation, ordering, nested reconstruction, conditional predicate, etc.)
- its tree pattern based structure and dependencies, make it suitable to a multi-sources context
- its annotation layers view is designed for a distributed environment
- merging rules make views on request easy to evaluate, and thus, client profile easy to consider
- its evaluation rules by using annotation and transformation make the distributed TGV evaluation feasible on autonomous nodes. Thus, with routing consideration, moving TGV to mobile agent can be done.
- its extensible optimisation rules, allows us to process TGV efficiently.

Future Works In this paper, we consider that sources description (metadata) are centralized on the PADAWAN proxy as a kind of a Yellow Page service. The scenario context of the coordination of rescue teams legitimates this approach.

In a more general context, it would be interesting to distribute the metadata all over the network. This raise the problem of maintaining such a distributed index of metadata. We are currently studying whether a P2P DHT approach is suitable to our needs, and how to distribute the TGV* in this way.

If we choose to distribute the data sources description, there won't be a central PADAWAN proxy anymore, and any node could then be considered as a PADAWAN proxy for other nodes, with more or less capabilities. This approach will transform the whole PADAWAN architecture into a P2P network that will be applied in the context where no critical coordination is needed.

Reflexion about routing TGV as a mobile agent would then have to be done. To optimize queries execution, nodes can also cache data and source description from TGV they forward to other nodes.

Acknowledgement

This work is done as part a of the PADAWAN project supported by the ANR.

References

- [Abi97] S. Abiteboul. Querying Semistructured Data. In *Proceeding of the 6th International Conference on Database Theory*, Delphi, Greece, 1997.
- [AML05] D.J. Abadi, S. Madden, and W. Lindner. Reed: Robust, efficient filtering and event detection in sensor networks. In *VLDB*, pages 769–780, 2005.
- [AYCLS01] S. Amer-Yahia, S. Cho, L.V.S. Lakshmanan, and D. Srivastava. Minimization of Tree Pattern Queries. In *SIGMOD Conference*, 2001.
- [BGL⁺99] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, and P. Velikhov. XML-Based Information Mediation with MIX. In *ACM-SIGMOD*, Philadelphia, USA, 1999.

- [BGS01] P. Bonnet, J. Gehrke, and P. Seshardi. Toward sensor database systems. In *Conference on Mobile Data Management*, 2001.
- [BPSM98] T. Bray, J. Paoli, and C. Sperberg-MacQueen. Extensible Markup Language (XML) 1.0 (W3C Recommendation), 1998.
- [CJLP03] Z. Chen, H.V. Jagadish, L.V.S. Laksmanan, and S. Pappas. From Tree Patterns to Generalized Tree Patterns: On efficient Evaluation of XQuery. In *Very Large Data Bases*, pages 237–248, Germany, 2003.
- [DNGT04] T.-T. Dang-Ngoc, G. Gardarin, and N. Travers. Tree graph view: On efficient evaluation of xquery in an xml mediator. In *Proc. of BDA*, 2004.
- [DNJT05] T.-T. Dang-Ngoc, C. Jamard, and N. Travers. XLive: An XML Light Integration Virtual Engine. In *Proc. of BDA*, 2005.
- [Fra01] M.J. Franklin. Challenges in ubiquitous data management. *Informatics*, pages 24–33, 2001.
- [GW97] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *VLDB*, 1997.
- [LDNL07] T. Liu, T.-T. Dang-Ngoc, and D. Laurent. Cost framework for a distributed semi-structured environment. In *Database Management and Application over Networks (DBMAN)*, 2007. (to appear).
- [MFK01] I. Manolescu, D. Florescu, and D. Kossmann. Answering XML Queries over Heterogeneous Data Sources. In *27th Intl Conf VLDB*, pages 241–250, Roma, Italy, 2001.
- [MHMM05] N. Marsit, Abdelkader Hameurlain, Zoubir Mameri, and Franck Morvan. Query processing in mobile environments: A survey and open problems. In *Distributed Frameworks for Multimedia Applications (DFMA)*, 2005.
- [MSFC02] S. Madden, R. Szewczyk, M.J. Franklin, and David E. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *WMCSA*, 2002.
- [NDK⁺03] S. Nath, A. Deshpande, Y. Ke, P.B. Gibbons, B. Karp, and S. Seshan. Irisnet: An architecture for internet-scale sensing services. In *VLDB*, 2003.
- [NGT98] H. Naacke, G. Gardarin, and A. Tomasic. Leveraging Mediator Cost Models with Heterogeneous Data Sources. In *ICDE*, pages 351–360, 1998.
- [PMT03] V. Papadimos, D. Maier, and K. Tufte. Distributed query processing and catalogs for peer-to-peer systems. In *CIDR*, 2003.
- [TDN07] N. Travers and T.-T. Dang-Ngoc. An extensible rule transformation model for xquery optimization. In *International Conference on Enterprise Information Systems (ICEIS)*, 2007. (to appear).
- [TDNL06] N. Travers, T.-T. Dang-Ngoc, and T. Liu. Tgv: an efficient model for xquery evaluation within an interoperable system. *International Journal of Interoperability in Business Information Systems (IBIS)*, 2, 2006.
- [TDNL07a] N. Travers, T.-T. Dang-Ngoc, and T. Liu. Tgv: a tree graph view for modelling untyped xquery. In *Database Systems for Advanced Applications (DASFAA)*, April 2007.
- [TDNL07b] N. Travers, T.-T. Dang-Ngoc, and T. Liu. Untyped xquery canonization. In *Workshop on Emerging Trends of Web Technologies and Applications (WebETrends)*, 2007. (to appear).
- [Toh01] C. K. Toh. *Ad Hoc Wireless Networks: Protocols and Systems*. Prentice Hall, NJ, USA, 2001.
- [Tra06] Nicolas Travers. *Optimisation Extensible dans un Médiateur de Données XML*. PhD thesis, University of Versailles, 2006.
- [W3C05] W3C. An XML Query Language (XQuery 1.0), 2005.
- [Wie92] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *Computer*, 25(3):38–49, March 1992.