TGV: an Efficient Model for XQuery Evaluation within an Interoperable System

Nicolas Travers, Tuyet-Tram Dang-Ngoc, Tianxiao Liu

TGV: an Efficient Model for XQuery Evaluation within an Interoperable System

Nicolas Travers#
PRiSM Laboratory#
University of Versailles#
France#
Nicolas.Travers@prism.uvsq.fr

Tuyêt-Trâm Dang-Ngoc, Tianxiao Liu#
ETIS Laboratory#
University of Cergy-Pontoise#
France#
Tuyet-Tram.Dang-Ngoc@u-cergy.fr, Tianxiao.Liu@u-cergy.fr

Abstract: This paper presents a generic model called TGV for efficient evaluation of XQuery in a heterogeneous distributed system.

XQuery is a rich and so a complex language that allows users to express a large scale of queries over XML documents. This expressiveness makes difficulties to obtain an exclusive internal representation within a system. To this purpose, models based on Tree Patterns have been proposed: TPQ [ACLS01], generalized by the GTP [CJLP03]. However, they do not capture well all the expressiveness of XQuery, cannot handle mediation problems, and do not support extensible optimization and sources information.

We present a tree pattern-based model called TGV that (1) integrates the whole functionalities of XQuery (2) uses an intuitive representation that provides a global visualization of the request in a mediation context (3) provides a framework for extensible optimization using a rules definition model (4) take into account all the knowledge useful for the query evaluation (cost model, accuracy, etc.)

This work has been implemented in the XLive system [DJT05] based on a full-XML architecture.

Introduction

Integrating efficiently data from the many and heterogeneous data sources on the Internet is a crucial need for enterprises for their information systems.

To integrate data coming from distributed and heterogeneous sources under a single query interface, the famous mediators/wrappers architecture [W92] is generally used. In this architecture, the user issues a request to the mediator which sends in turn part of the request to wrappers associated to data sources. The result is then sent back to the mediator, which integrates the result accordingly.

XML is a popular language for representing business data and for exchanging information between business partners and applications. XQuery is a rich and complex language for querying XML that can produce very precise, fine-grained result joining databases powerful expression and document search functionalities.

In order to guarantee interoperability between the different elements of a mediator/wrappers architecture, it is now obvious to consider XML for data representation and XQuery as the query language.

#

But from the XQuery queried by the user to the XML result provided by the# mediation system, a complex evaluation process has to be completed. Several# issues emerge as well from the point of view of the distributed environment as of# the XQuery evaluation. First point has been widely studied in lot of papers and# project [BGLRM99, RS97, DG03] using various query languages. 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). However, the complexity of the XQuery language makes its# evaluation very difficult. To alleviate this problem, most of the systems support# only a limited subset of the XQuery language.##

The XQuery expressiveness makes difficulties to obtain an exclusive internal# representation within a system. To this purpose, models based on Tree Patterns# have been proposed: in particular TPQ [ACLS01], generalized by the GTP [CJLP03].# However, GTPs do#not capture well all the expressiveness of XQuery, cannot handle# mediation problems, and do not support extensible optimization.##

Another challenge is to provide an extensible optimization framework for efficient# query evaluation. Search strategies for optimizing requests have been studied in# Exodus [CDGH90], Starburst [W96], Volcano [GM93] and OPT++ [KD99]. The key# idea of an extensible optimizer is to generate a query optimizer from rules for# transforming plans into alternative plans. Selecting and ordering such rules for a# query execution often rely on cost information as introduced with expected cost# factor in [CDGH90] and [GM93]. However, all these works apply only on relational# or object context. As far as we know, nothing on rule-based optimizer has#been# done in semi-structured context on tree pattern matching queries.##

Finally, both for previous optimization consideration and to consider sources# specificities, it is important to take into account any piece of information that can# influence the evaluation processing. Such information can be cost models,# constraints on data sources (security, availability, and preferences), limited# functional capabilities [RS97], flexible results [CPS03], etc.##

We present a tree pattern-based model called TGV that:##

- integrates the whole functionalities of XQuery##
- uses an intuitive representation that provides a global visualization of the# request in a mediation context##
- provides a framework for extensible optimization using a rules definition# model##
- Takes into account all knowledge useful for the query evaluation (cost# model, accuracy, etc.)##

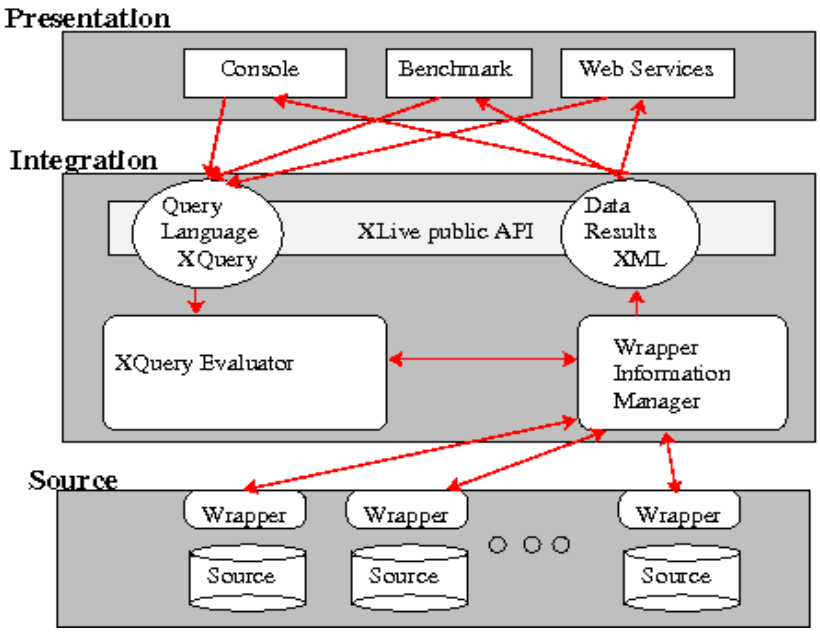
The next section focused on the middleware objective and briefly describes the# XLive system architecture and the query evaluation process. Then, we present the# TGV model designed for#manipulating efficiently any query in a distributed# heterogeneous environment. We give an overview of the rules definition applying# on the TGV and introduce generic annotation. In conclusion, we focus on typical# applications for the TGV and we give results and perspectives.##

#

Evaluating XQuery in a Distributed Heterogeneous# Environment##

XLive Mediator System##

The XLive [DJT05] prototype is designed to be a light mediation system with high# modularity and extension capabilities. It is a running research vehicle designed for# assessing the integration system at every stage of the process starting from sources# extraction to the user interface, including query parsing and modeling,# optimization and evaluation, and also benchmarking. As most mediation systems# [W92], XLive is composed of three layers: Presentation, Integration and Sources.##



#

Figure 1: XLive Architecture##

The integration system is shown in Figure 1. On the Source Layer, there are# multiple heterogeneous data sources (relational/object/XML data sources, web# services, files, etc.) These sources are queryable by the XLive system#*via*#wrappers.# The Wrapper is a component for accessing a specific Source for querying and# retrieving result. As sources have specific access methods, the role of the wrapper# is to translate the wrapper specificity to a common access method.##

The Integration Layer is the heart of the mediation system as it process XQuery# requests according to sources and send by the result in XML form. The Wrapper# Information Manager is for integrating information about wrapped sources. It# provides to the mediator sources metadata, capabilities, and costs statistics on# source data. The XQuery Evaluator parses the XQuery query, make logical plans# then physical plans by using optimization rules, choose the best execution plan,# and then evaluates the execution plan by querying relevant sources and merging# results The XLive system is provide a public API that process an XQuery query and# evaluates it on sources, and then return the result as an XML document.##

Some clients using the XLive public API have been implemented on the Presentation# Layer: The Console is a graphical interface for managing and querying XLive. The# Web service provides services for managing/querying XLive from the Web. And# finally, the#Benchmark is designed to test the XLive mediator and other sources in# order to make comparisons.##

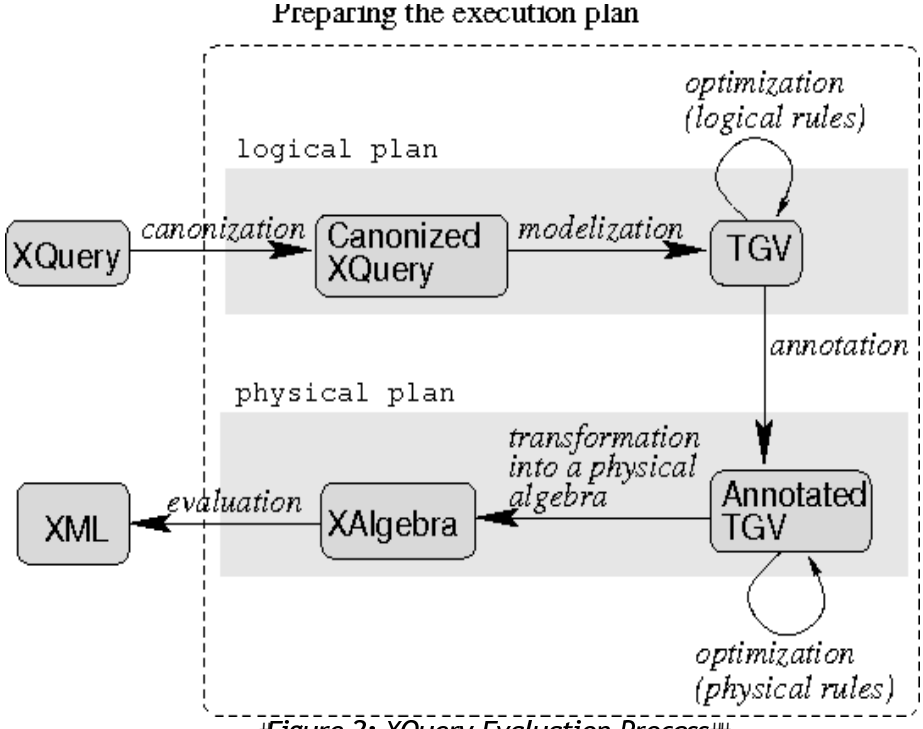
The rest of the paper will concentrate exclusively on the XQuery evaluation# process. More details on the XLive architecture can be found on [DG03,DJT05].##

XQuery Evaluation Process##

XQuery is a rich and complex language. Its powerful expression capabilities provide# a large range of queries over XML documents. However the richness of the language# makes the XQuery processing very difficult so it is necessary to#reduce the XQuery# domain to study (but not restrict the XQuery domain that must be recognized and# evaluated!). For that purpose, we use query equivalent syntax using some# transformation rules. These transformation rules keep the semantic of queries and# make them more convenient to manipulate. Set of equivalents queries are then# reduced to one unique canonical query.##

XQuery canonization has been introduced by [CJLP03] and we have extended# canonization rules in order to support the whole XQuery semantic (except types# support). Canonization rules and proof are described in [T06].##

Figure 2 describes the evaluation process: (1) XQuery is canonized into a canonical# form of XQuery (2) then the canonized XQuery is modeled in the internal structure# TGV which can (3) be restructured into equivalent structures using equivalence# rules. (4) Then the TGV is annotated with information for evaluation such as the# data sources location, cost models information, sources functional capabilities,# etc. The optimal annotated TGV#is then selected and (5) the logical TGV is# transformed into an execution plan using a physical algebra. We have chosen the# XAlgebra [DG03], an extension of the relational algebra to XML. (6) Finally, the# execution plan is evaluated and produces an XML result.#



#Figure 2: XQuery Evaluation Process##

The whole process is implemented in the XLive [DJT05] system and validates all use-cases defined by the W3C that do not imply strong typing consideration (our system recognize 8 of the 9 categories of XQuery).

Tree Graph View (TGV)

TGV

Although canonization reduces the XQuery queries expression to a canonical form, XQuery modeling is a difficult exercise since XQuery provides a large set of functionalities. Moreover, we also choose to integrated mediation requirements (data localization on sources, heterogeneous sources capabilities).

We propose the TGV (Tree Graph View) model implemented in the XLive mediator for XQuery processing. It is based on Tree Patterns [ACLS01, CJLP03]. The TGV model extends the Tree Pattern representation in order to make it intuitive, has full XQuery support and has support to be manipulated, optimized and then evaluated.

Each of elements in the TGV model has been defined formally using Abstract Data Type (sees formalization in [T06] and has a graphical representation.

TreePatterns

A TreePattern selects nodes based on their structural characteristics. TreePatterns are made of Node defined by a label and NodeLink that represents relations between the Nodes. These relationship include XPath axis characteristics (child, descendant, self, etc.), but also additional information as mandatory/optional. A Pattern Tree is illustrated on Figure 3.

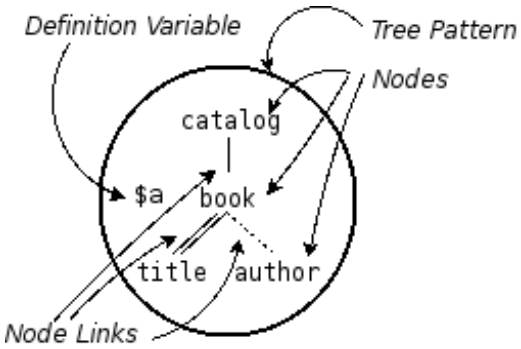


Figure 3: Example of a Node and of a NodeLink in a TreePattern

To represent all XQuery#richness, we must introduce specific Tree Patterns to# model each characteristic of XQuery:##

- Source Tree Pattern is a#*Tree Pattern*#defined by a targeted document or set# of documents and a root path.##
- Intermediate Tree Pattern is a#*Tree Pattern*#defined on a#previous one that# specializes the domain on a specific Node.##
- Return Tree Pattern is a#*Tree Pattern*#that defines the result construction of# an XML document. Each node coincides with a tag on the XML result set, or a# contents text.##
- Aggregate Tree Pattern is a#*Tree Pattern*#that builds a temporary result set.# It represents nested queries and aggregated functions on a set of trees.##

Figure 4 gives example of different types of TreePatterns.##

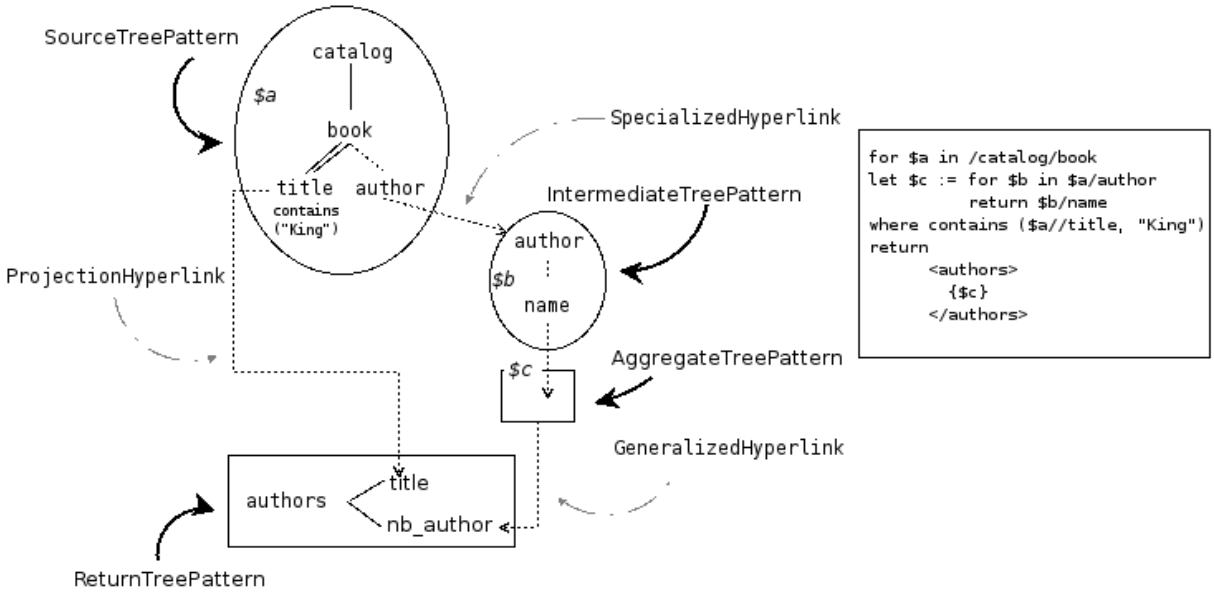


Figure 4: TreePatterns 1 Four types of tree Patterns: SourceTreePattern, IntermediateTreePattern,# ReturnTreePattern and AggregateTreePattern; and three types of Hyperlink: ProjectionHyperLink,# SpecializedHyperLink, and GeneralizedHyperLink##

Hyperlinks##

Other links named Hyperlinks (see figure 5 and 6) have been defined#to represent# additional relation between nodes belonging to different tree pattern:##

- Association Hyperlinks for restriction purposes on some sets of elements.##
  - Join Hyperlinks are made to connect two#*Nodes*#with a#*Constraint*#that# represents a join constraint.##
  - Constraint Hyperlinks connect two# *Constraints*# with a Boolean# connector associated to a# *return clause*, in order to preserve# constraint association (with and/or operators) and the treatment# declaration level.#
- Directional Hyperlinks to transform set of trees into a new one.##

- Projection Hyperlink is a# *Node*-to-Node# *Directional Hyperlink*. It# contains a mandatory or optional status. It represents a value# projection of the given node into the projected one.##
- Specialized Hyperlink is a# *Node*# to# *Tree Pattern*# *Directional Hyperlink*.# It contains a mandatory or optional status. It represents the# specialization of a Node, by specifying a new TreePattern which root# is the given node.##
- Generalized Hyperlink is a# *Tree Pattern*# to# *Node*# *Directional*# *Hyperlink*. It contains a mandatory or optional status. It represents a# TreePattern generalization result set, which result is projected into# the given node.##
- Set Hyperlink is a set of# *Tree Patterns*# to# *Node*# under# *Constraint*# *Directional Hyperlink*. The Constraint possible values are: Union,# Intersect or Difference. It represents a set operation between few# TreePatterns on a single Node.##
- IfThenElse Hyperlink is a set of# *Elements*# to# *Node*# under# *Constraint*# *Directional Hyperlink*. Elements can be a Node or an# *AggregateTreePattern*, and the constraint is a Predicate or a# Function. It represents a conditional expression which result is# deduced by the constraint status.##

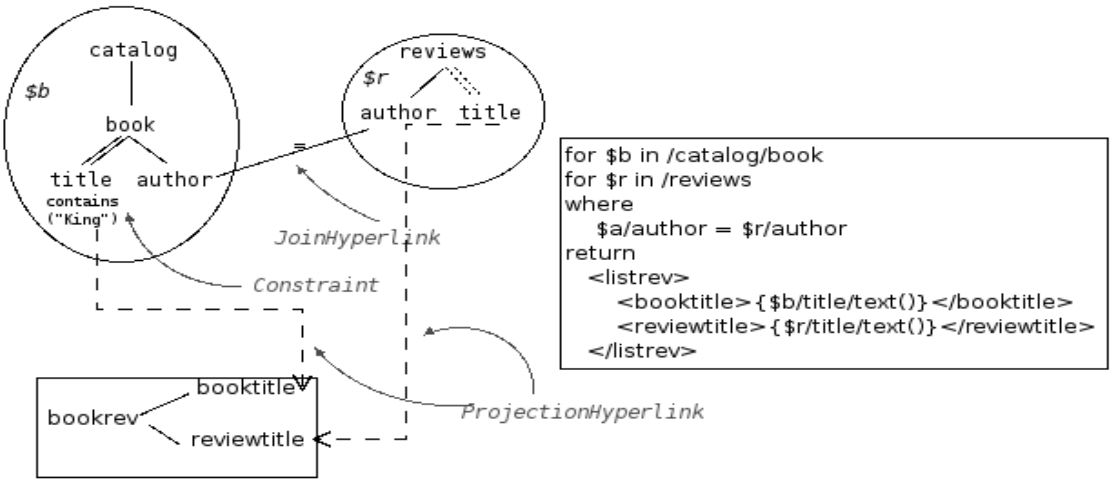


Figure 5: Example of a Constraint and of a Join HyperLink#

A Tree Graph View is a combination of the presented previous element that# composes an XQuery query. Examples of TGV have been seen in Figures 3, 4, 5, 6# and 7.##

Tree patterns for source document, return document and intermediate structure# are shown in the figure. On tree pattern, label, constraints on value and# ascendant-descendant links are represented (simple line for father-son link, double# link otherwise), and a variable is bound. Between trees, hyperlinks representing# aggregation, join, projection or condition are represented.##

For more descriptions on TGV, see [DGT04, T06].##

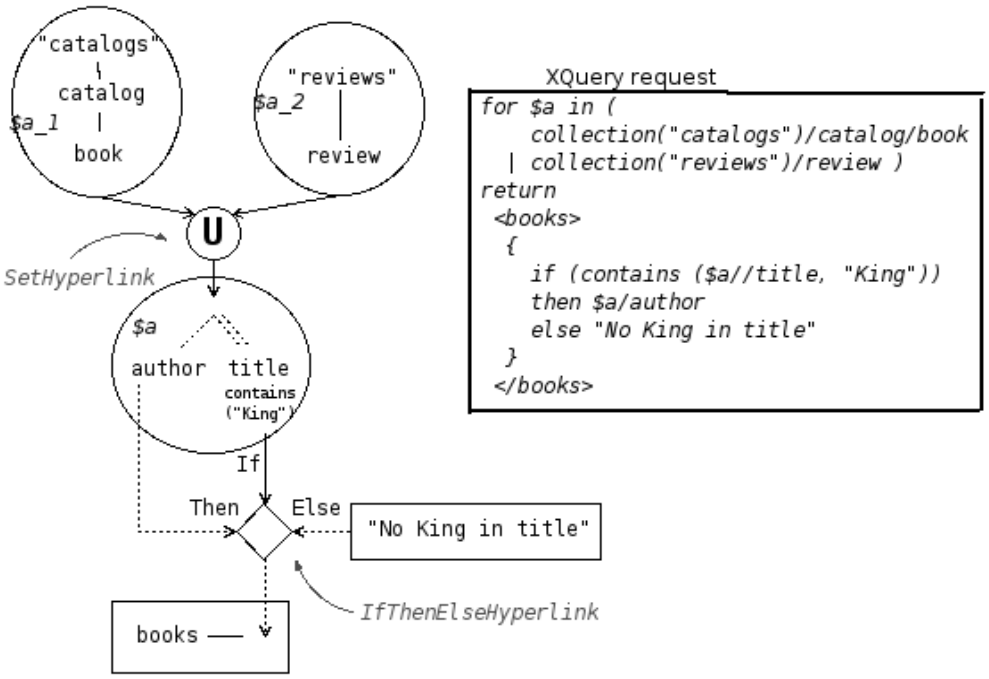


Figure 6: Two examples of Hyperlinks: A SetHyperLink (union) and an IfThenElseHyperlink

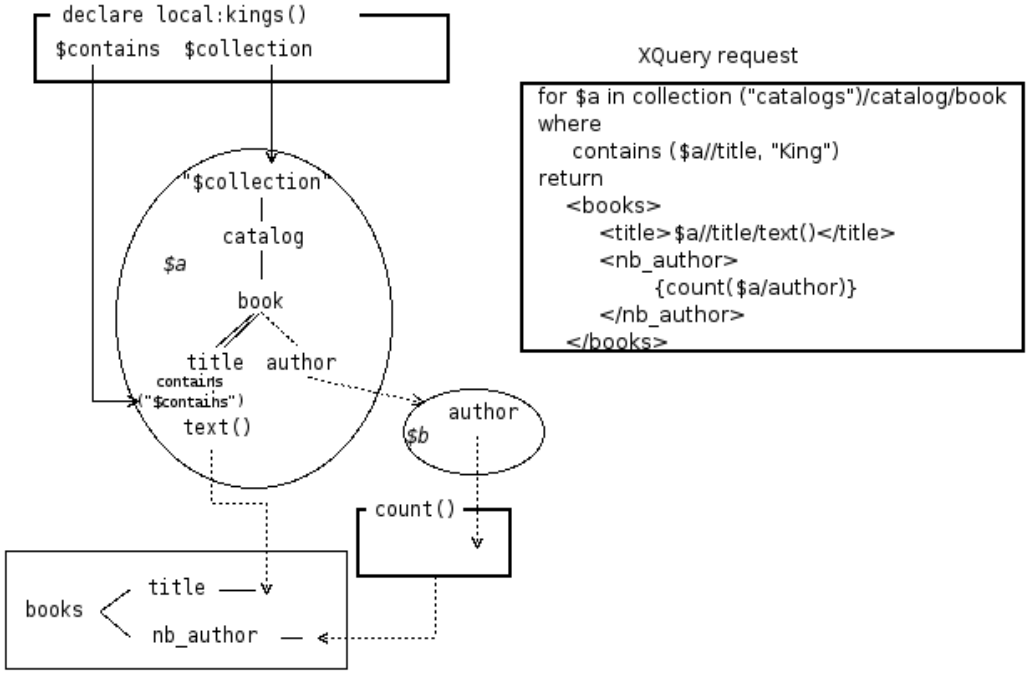


Figure 7: Declaring and applying functions

Rules definition for extensible optimization

To optimize the XQuery evaluation, a search-strategy is needed. This strategy relies on rule valuation with a coefficient of improvement [CDGH90]. This coefficient is directed by information such as cost model. We define TGV transformation rules that transforms a given execution plan into an equivalent

execution plan (i.e. that give the same result when evaluated). Transformation# rules can be logical (i.e. based on the algebraic properties of the operators),# physical (i.e. based on the hardware, system and data statistics, formulas ad# coefficients) or user-defined (i.e. defined arbitrarily by the user or the# administrator according to its knowledge of some particular specificities on data# and sources, or expected results).##

Rule Definition Language##

We define the rules definition language. A rule consists in a rule condition followed# by a rule conclusion. Operations applying on the rule condition to result in the rule# conclusion are called a rule transformation.##

#FOR [Variables]#
IF [Conditions]#
THEN [Modifications]#

Table 1: Rule Language#

In table 1, we propose a#rule definition language that contains three clauses:#*FOR*,# *IF, THEN*. Each one defines a rule property:##

- **FOR**: Entry type which defines search space on rules concerned elements.# Type can be:#*Node, NodeLink, TreePattern, Hyperlink, and Constraint*.##
- **IF**#*(for rules conditions)*: Rule applying conditions on elements defined into# the#*FOR*#clause. ADT operations are used to define conditions, and temporary# variables are used to facilitate manipulations. Each temporary variable will# be declared into the form:#$*var*:= ....##
- **THEN**#*(for rules conclusion)*: Transformations to apply if all#*IF*#clause# operations are verified. We can use variables defined into the previous# clause to modify values. Transforming operations are creation, destruction# and updates.##

Rule Patterns##

Like#TGV mappings on XML documents with Tree Patterns, we represent rules in# intuitive manners in order to build a pattern to map visually on TGV# representations. This rule modeling is called Rule Pattern Model (RP Model). The RP# Model consists in two patterns:#condition pattern and conclusion pattern. Part(s) of# the TGV that match the condition pattern must be replaced by the conclusion# pattern.##

Figure 8 shows a logical rule example: a constraint of one member of a join# hyperlink also applies to the other member. Figure 9 shows another user-defined# rule example: a descendant optional NodeLink can be replace by the full path with# optional child link.##

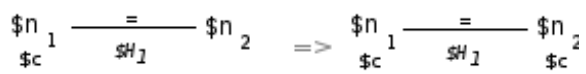


Figure 8: A Rule example defined with RP Model##

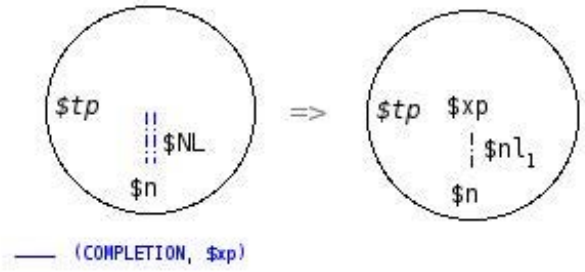
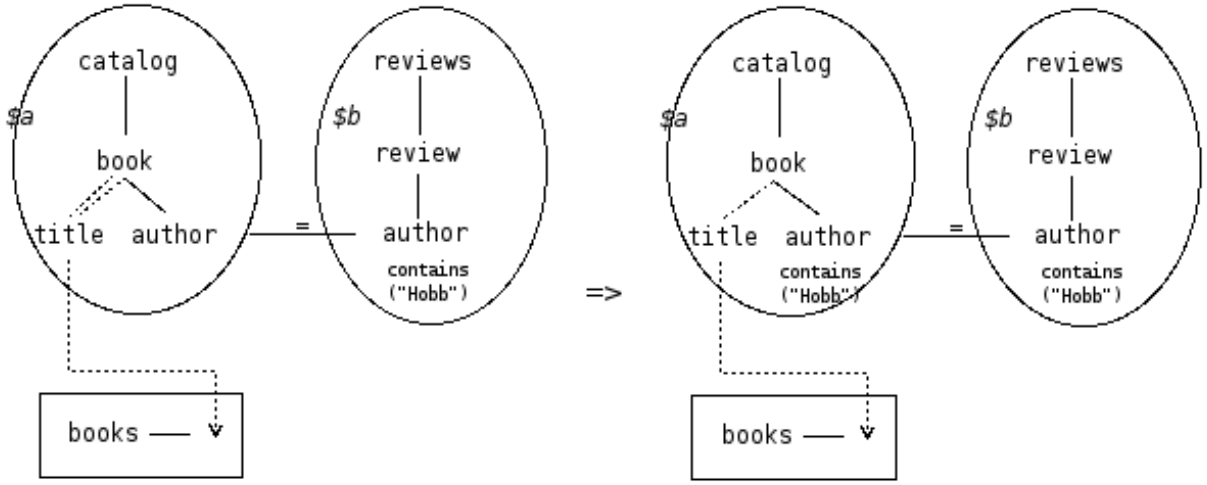


Figure 9: Another rule example defined with RP Model##

#

Figure#10 describes the transformation applied to the TGV on the left sides using# the two previous rules. We obtain the TGV on the right side.##



#

Figure 10: A Rule with RP Model##

Annotation##

The more information we have on data, on which the request applies, the#more# efficiently we can define and use optimization rules on TGV.##

Generic Annotation Model##

We define a generic annotation model to annotate the TGV for (a) any granularity# of information and (b) any type of information (cost models and statistics,# constraint, accuracy, security, rule traceability).##

1. Granularity: Indeed, in a distributed heterogeneous environment,# information as cost can be more or less available according to sources. Some# manageable sources can provide precise information (statistics, algorithms)# for each operator and data it handles. At the other extreme, very# autonomous sources (e.g. web site) do not give any piece of statistics# information, and we just can estimate a global behavior of the system.##
2. Type: is for characterizing information such as cost that can be for example# time cost, resource cost, price cost, energy cost, etc. or other information# as accuracy, constraints as presented in the work of weighted tree pattern# [DT00], fuzzy [DGA00] or flexible [CPS03] queries.##

#

Our annotation model is generic and allows any type of information. We define an annotation frame as follow:

Annotation frame = 1 set of TGV elements + 1 annotation

The set of TGV elements indicates the annotation granularity. A TGV element can be either a node name or a father-son link or a predicate on node or a hyperlink. In order to visually represent the annotation on the TGV we use different colors. Each annotation frame is associated to a unique color, and so every element of an annotation frame has the same color.

Information annotated can be of any form: formulas, equations system, files, string, etc. They are given with the associated color near the TGV model

(Note: In this black&white paper, we cannot represent visually the annotated TGV. But just imagine the TGV on previous figures where all elements (links, labels, predicates) have a color defined dependently on the annotation frame they belong to).

In the XLive system, we have mainly concentrated on cost model annotation. To estimate cost of execution plan generated by previous transformations, and to allow physical rules to use physical information, information as statistics of sources, data cardinalities, network speed, etc., are provided to the optimizer. They are annotated on the TGV.

Conclusion

The TGV model [DGT04, T06] extends the Tree Pattern representation in order to make it intuitive, has full XQuery support and has support to be manipulated, optimized and then evaluated. The representation is intuitive and acts as a template for the data source, as QBE [Z77] did for querying relational data.

XQuery query is modeled with TGV that can directly generate execution plan that is used to evaluate the XQuery query. In fact, specific XQuery characteristics have been integrated into the TGV model by canonization or specific elements.

The TGV logical structure allows us to cover 8 of 9 use cases of the W3C. The use-cases category not covered by our TGV model is the STRONG category that queries type information. We are not supporting it as we don't yet support typing in our XLive system. Adding typing to the TGV model needs some works using XQuery/XPath typing consideration [GKPS05] on validated XML document.

Since the TGV model is not bound specifically to a specific language (thought for XQuery), it can be applied to any un-typed queries in any language on structured or semi-structured data (OQL, OEM-QL, etc.).

Since the TGV model is generic to model queries, the rules patterns design is also applicable for any query in a distributed and heterogeneous environment for an extensible optimizer. Finally, the generic annotation model can be used to annotate any information that is useful during the evaluation process.

The work presented in this paper privileges interoperability between the components of a middleware such as a mediator. It provides a model, rules definitions and information annotation that can be used in any exchanges between business applications.

Acknowledgment

The XLive is supported by the ACI SemWeb project. The cost model is also# supported by the ANR PADAWAN project and by XCalia S.A.#
The research prototype XLive system is Open Source software and can be# downloaded on:#

http://www.prism.uvsq.fr/~ntravers/xlive/

References#

[ACLS01]# S. Amer-Yahia, S. Cho, L. V. S. Lakshmanan, and D. Srivastava: Minimization of Tree# Pattern Queries.#*In#SIGMOD Conference*, 2001.#

[BGLRM99]# 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.#

[CPS03]# D. Calmès, H. Prade, and F. Sedes: Requêtes flexibles et données semi-structurées.# *In#LFA#*, pages 23-30, 2003.#

[CDGH90]# M. J. Carey, D. J. DeWitt, G. Graefe, D. M. Haight, J. E. Richardson, D. T. Schuh, E.# J. Shekita, and S. Vandenberg: The EXODUS Extensible DBMS Project: An Overview.# *In D. Maier and S. Zdonik, editor,#Readings on Object-Oriented Database Sys.*## *Morgan Kaufmann*, San Mateo, CA, 1990.#

[CJLP03]# Z. Chen, H. Jagadish, L. V. Laksmanan, and S. Paparizos: From Tree Patterns to# Generalized Tree#Patterns: On efficient Evaluation of XQuery.#*In#VLDB#*, pages 237-248, Germany, 2003.#

[DT00]# E. Damiani and L. Tanca:. Blind Queries to XML Data.#*In#Database and Expert System#* *Applications*, 2000.#

[DGA00]# E. Damiani, L. Tanca, and F.Arcelli: Fuzzy XML queries via context-based choice of# aggregation.#*Kybernetika#*, 36, 2000.#

[DGT04]# T. Dang-Ngoc, G. Gardarin, and N. Travers: Tree Graph View: On Efficient# Evaluation of XQuery in an XML Mediator.#*In#BDA*, pages 429-448, France, october# 2004.#

[DG03]# T.-T. Dang-Ngoc and G. Gardarin: Federating Heterogeneous Data Sources with XML.# *In#Proc. of IASTED IKS Conf*, 2003.#

[DJT05]# T.-T. Dang-Ngoc, C. Jamard, and N. Travers. XLive: An XML Light Integration Virtual# Engine.#*In#Proc. of BDA*, 2005.#

[GKPS05]# G. Gottlob, C. Koch,#R. Pichler, and L. Segoufin: The complexity of xpath query# evaluation and XML typing.#*ACM (JACM) archive#*,#52:284-335, 2005.#

[GM93]# G. Graefe and W. J. McKenna. The Volcano Optimizer Generator: Extensibility and# Efficient Search. In#ICDE#, pages 209-218,#1993.#

[KD99]# N. Kabra and D. J. DeWitt: OPT++ : an object-oriented implementation for# extensible database query optimization.#*VLDB Journal#*, 8(1):55§78, 1999.#

[RS97]# M. Roth and P. Schwartz. Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy# Data#Sources.#*In#Proceeding of the Conference on Very Large Data Base*, Athens,# Greece, 1997.#

[T06]# N. Travers:#Optimization Extensible dans un Médiateur de Données XML#, PhD thesis,# University of Versailles, 2006.#

[W3C05]# W3C: An XML Query Language (XQuery 1.0), 2005.#

[W96]# J. Widom: The Starburst Active Database Rule System.#*Knowledge and Data#* *Engineering*, 8(4):583-595, 1996.#

[W92]# G. Wiederhold: Mediators in the Architecture of Future Information Systems.# *Computer#*, 25(3):38-49, Mar. 1992.#

[Z77]# M. Zloof:#QBE : Query By Example, 1977.#

Biography#

Nicolas Travers, is currently in the third year of his PhD (the defense is planned for# december 2006) at University of Versailles in PRiSM laboratory. His PhD thesis# concerns extensible optimization of XQuery queries in a mediation context.##He# introduced the TGV model to represent XQuery queries and enable optimization# and evaluation. Its prototype is integrated in the XLive mediation system, an open-source software used in several research and industrial project.#He is supervised by# Prof Georges Gardarin and Dr Tuyêt-Trâm Dang-Ngoc.#

Dr. Tuyêt-Trâm Dang-Ngoc#is an Assistant Professor at University of Cergy-Pontoise# in the ETIS Laboratory. She obtained a Ph.D thesis in Computer Science on the# subject "Federating Semi-Structured Data with XML" in 2003 from the university or# Versailles. Her main research areas#are data integration and semi-structured data.#

Tianxiao Liu#is a Ph.D student in the ETIS Laboratory and an engineer at XCalia S.A# Data&Service intermediation.#His PhD topic concerns cost model in distributed# heterogeneous environment.#He is supervised by Prof Dominique Laurent and Dr# Tuyêt-Trâm Dang-Ngoc.#

#
#

#