Webdam

Foundations of Web Data Management

Technical Description

ERC Advanced Grant
7th Framework Programme

Principal Investigator: Serge Abiteboul, http://www-rocq.inria.fr/~abitebou

Host institution: INRIA Saclay–Île-de-France

Date of this document: 2008

1 Extended synopsis of the project proposal

Abstract: We propose to develop a formal model for Web data management. This model will open new horizons for the development of the Web in a well-principled way, enhancing its functionality, performance, and reliability. Specifically, the goal is to develop a universally accepted formal framework for describing complex and flexible interacting Web applications featuring notably data exchange, sharing, integration, querying and updating. We also propose to develop formal foundations that will enable peers to concurrently reason about global data management activities, cooperate in solving specific tasks and support services with desired quality of service. Although the proposal addresses fundamental issues, its goal is to serve as the basis for ground-breaking future software development for Web data management.

One can see in computers and the network, a revolution comparable to that of writing. Writing freed us from the need to memorize information [**?**]. The new technology is freeing us (to some extent) from the need to reason about information and will enable us to focus our energy to imagination and creativity. The Web with data deployed on millions of machines flourishes at the core of this new revolution.

The Web was first seen as a universal access to an ocean of documents. It is progressing slowly (by Web-time measure) towards a world-wide repository of knowledge, i.e., towards the Semantic Web [**?**]. We are also observing the birth of Web 2.0 [**?**], that stresses social networks and community building. Then came proposals for Web 3.0 up to Web Googol.0. These initiatives, totally lacking scientific grounding, have "invented" concepts already studied in research labs and sometimes even present in software products. However, in spite of their obvious weaknesses, these proposals highlight the fact that the Web is meant to support functionalities that are beyond a simplistic query/answer paradigm. Indeed, with the Web as the basis of human sharing of information and human interaction,

Web data management is becoming the cornerstone of human activities.

In spite of its importance, the management of data on the Web suffers from a number of weaknesses that can be blamed to its recent birth and its too rapid growth. The goal of the proposal is to develop a mathematical model for Web data management and formal foundations to reason about Web applications. Indeed, we propose to develop the analog of the formal model that has successfully underlied relational database systems. Building on it, we propose to develop the necessary reasoning capabilities for controlling data exchange, sharing, integration, querying and updating, in a Web environment.

Time to stop hacking the Web For data management, the turn of the century has seen the maturing of the relational database industry (with notably Oracle, IBM DB2 and MS SQL Server) and of the corresponding scientific field, at the border of logic and complexity theory. Since the 60's,

industry and academic research have progressed in tandem, with strong interactions that have been instrumental in their respective successes. Today, with the preeminence of the Web, new challenges are facing the data management arena. New ideas are emerging and transferred into products very rapidly. The scientific community has difficulties following the pace and to a large extend, it is lagging behind. We claim that the quick and dirty solutions, often used today, cannot provide long-term solutions and do not scale to the Web of the future. We argue that, perhaps even more than for relational systems, academia has an essential role to play in shaping the new field. Indeed, the project we propose tackles key technological issues that present the severe risk of slowing down Web data management. Solutions even designed by fantastic hackers are bound to fail due to the complexity of the problem and its scale. We therefore propose to develop a formal model that is needed for developing efficient, powerful and reliable software for Web data management.

Limiting the scope: semantics A main issue for the Web is semantics. For instance, it is simply not possible to use some newly discovered resource without understanding first its semantics. The study of Web semantics leads to fascinating challenges of an AI nature such as knowledge representation, knowledge extraction from text, or reasoning with knowledge. Such issues will not be central here. We are focusing instead on the data layer that will help support semantic layers. To illustrate the distinction, for instance, we are not concerned here with how a system analyses a Web service to understand how to use it (an AI problem) but rather, with how it will actually take the best advantage of the service to obtain data (a DB problem). Of course, the boundary between the two domains is fuzzy and the author of the proposal has worked on a number of projects at their frontier. However, the techniques they require are rather different, which justifies limiting the scope of the proposal to the data management side. Clearly, semantics will still be very present in the sense that the data management layer should be powerful enough to satisfy the needs of semantic layers. So, for instance, the formal model will include the means to describe static properties of the data (e.g., with tree automata) as well as behavioral properties (e.g., with temporal logic formulas).

A unifying model An underlying philosophy of the proposal is that we need an unified formal model for noncentralized data management. So, we are concerned here with all distributed applications that *possibly* manage huge volumes of data with *possibly* large number of autonomous and heterogeneous computers, typically without centralized authority. We believe that the foundational work that is needed should be applicable to such a wide range of applications. The prime motivation for using a single model across all Web applications is that it will enable all kinds of Web applications to exchange data. Another positive aspect of genericity, is that it will force us to stay away from being too directly influenced by particular technological biases, and focus on rich mathematical notions. The Webdam proposal is thus extremely ambitious because it targets the world-wide acceptance of a unique model for data management on the Web.

Note that we did not use the term "distributed data management" because for many it comes with limitations to small numbers of data servers, often with a centralized authority. We were also reluctant to use the term "peer-to-peer" because for many its meaning is restricted to very large numbers of machines and high churn rates. In Webdam, we address distributed and peer-to-peer data management scenario, even if each such context may come equipped with its specific optimization strategies and its specific issues. We prefer the term "Web data management" to stress that the Web

is the testbed for this new technology. Observe that we are neither excluding smaller scale devices (such as RFIDs) nor smaller scale networks (such as home networks).

Since distributed databases have been around for more than twenty years, one may wonder at this point whether some such foundations already exist. It turns out that this is not the case. While he taught distributed databases at Stanford, the principal investigator realized how informal, imprecise and incomplete was the material on the topic. For teaching relational databases, one can rely on a clean and solid formal model [**?**, **?**]. But when distribution is considered, the rare existing books, e.g. [**?**], propose ad hoc extensions, developed for the purpose of explaining particular aspects and specific techniques. A formal model is missing and to develop one is a main goal of Webdam. Indeed, as a by-product, Webdam should bring improvements to course materials for distributed databases, and provide the basis for new courses on Web data management.

The first goal of the proposal is thus to develop a mathematical model for Web data management. We next discuss the second goal, that is, to develop formal foundations to reason about Web applications.

Automatic reasoning In many Web applications, notably industrial applications, we are interested in achieving some desired quality of service level. For instance, we may want to allow several users working on the same data collection without disrupting each other (concurrency control) or to support recovery from peer failures. Other examples of a less classical database nature involve guaranteeing full awareness of changes occurring in sites of interest (monitoring) or checking that data only come from trusted sites (provenance). The current style of development of Web applications makes it infeasible to obtain such guarantees. Techniques developed for the relational model are not directly applicable. For instance, concurrently control techniques are typically assuming a central authority which rarely exists in a Web context. Two aspects are making the problem even more challenging: (i) the application design must be extremely flexible to adapt to the intrinsic anarchist style of the Web and (ii) there is typically no database administrator in Web applications and often not even any computer specialist to design and maintain the application. For instance, most cooperative Web sites (e.g., based on Wordpress or similar systems) integrating data from many sources (e.g., RSS feeds), are developed by non computer experts. Because of (i), we cannot impose any a-priori design such as ACID for relational transactions. And, because of (ii), we have to favor approaches that do not require actually writing any complex code, with application developers specifying declaratively requirements, and the code being generated from them. As a consequence of the two, to still be manageable, the system has to be able to reason about an application, analyze the current or future run and possibly adapt based on the results of the analysis. Automatic reasoning (taken here in a very large sense) is unavoidable in this setting to support all activities from querying and optimization, to updates, tuning, recovery, monitoring, etc. We will precise further on, the kind of reasoning that we mean.

High-risk, high gain What are the challenges and chances for achieving such an ambitious goal with Webdam?

Not enough theoretical There is a risk to stay too close to the current Web technology. If we are too influenced by technological details of a non fundamental nature, we will obtain a too complex model, biased by the current technology and its limitations. As a consequence, it will be impossible to lay the appropriate formal foundations. In recent applied works, the principal investigator could observe the limits of ad hoc approaches and the crucial need for solid foundations for Web data management. He is thus much aware of this pitfall. His experience and that of a number of talented theoreticians who will be associated to the endeavor, should allow avoiding it.

Too theoretical Of course, there is the somewhat opposite risk of developing beautiful theoretical techniques with little impact outside of academic circles. The record of the principal investigator is a strong indication that this will not be the case. Indeed, he has for the last ten years always been involved with works with transfer to industry. In particular, his work around semistructured data has had important impact notably on standards for XML query languages. Foundational work here is not a goal for its own sake (which would already have been a fair motivation) but is meant as a sound basis for future software development.

All or nothing? With top quality researchers and the ambition to succeed, Webdam will serve as the catalyst for excellent European research on the topic. Even if Webdam does not reach its full goal of providing a comprehensive, universally accepted framework for Web data management, it should provide substantial progress towards this goal.

An opportunity for Europe Does Europe have a chance to succeed in this strategic field that will surely be very competitive? In the early 80ths (when the PI finished his PhD), Europe was almost absent from data management research. For data management systems, Europe has regularly closed the gap with the US and now hosts international quality groups in many countries, one of them being the Gemo team. Database theory developed in Europe starting from a few pioneers, e.g., Paredaens from Belgium and the PI in France. One can now say that the center of gravity has shifted from the US to Europe in this area. ACM PODS has had 4 European Program Chairs since 2000 and the past 7 winners of the Best Paper Awards had one European co-author. Success can be achieved only by driving the development of sophisticated mathematical models with concrete technological goals. With a strong presence in both database theory and data management systems, Webdam and more generally Europe are ideally placed to carry out such a program.

We will see in the next section how this can be achieved using a methodology based on the following principles:

1. developing mathematical foundations combining techniques from databases and verification,
2. building on the results and experience already achieved,
3. bringing together local talents and researchers from an already existing network of collaborators, that will be extended during the project,
4. relying on simplicity and carefully chosen limitations in the model of computation we use,

5. getting inspiration from real Web applications and validating results with prototypes.

To conclude this section, we want to stress that:

for obvious economical, social and political reasons, the management of the data of the Web is of strategic importance.

We therefore strongly believe that it is essential to fund ambitious projects in this area.

We next discuss the state of the art and the objectives, considering in turns the two main facets, namely formal model and reasoning. We then present the main traits of the proposal, briefly sketching the expected scientific contributions and the milestones. More technical details on some of these aspects may be found in a CIDR vision paper, written with Neoklis Polyzotis [**?**]. Finally we discuss resources.

2 State-of-the-art and objectives

2.1 A first scientific shift in data management: trees, functions

With the Web, we have shifted from the management of very structured data (i.e. relations) in centralized databases to semistructured data (from text to trees and graphs) in autonomous distributed systems. Even if most Web data still comes from relational databases, they are typically viewed on the Web as trees, XML or HTML. A main aspect of the information is its lack of regularity coming from the heterogeneity of the Web and from the multiplicity of sources and authors. It is also most importantly distributed, and dynamic. Another essential aspect of modern data management is its scale in terms both of quantity of data (e.g., see the billions of pages indexed by Google) and of number of peers (e.g. millions of peers for P2P music sharing). In Webdam, we will build on the standards of the Web, namely XML [**?**] and Web services [**?**]. These choices are somewhat not negotiable on the Web. They happen to also be scientifically sound. Indeed, these standards may be abstracted in a clean and elegant manner using trees and functions.

XML XML, the Web standard for data exchange, is based on unranked, labeled, ordered trees. Depending on the needs, we will sometimes see the trees as unordered to bring the model closer to standard logics that are set-oriented as well as to lower the complexity. Labeled trees are much more appropriate to the Web than relational structures that are too rigid and constraining. A most natural tool for such trees is tree automata [**?**] that have strong connections with monadic second-order logic. Standard query languages for XML, XPATH and XQuery may be seen (with some stretch of imagination) as logics for such trees. Theoretical aspects of XML have been actively investigated during the last few years, with tree automata and monadic second-order logic playing central roles. The results that were obtained are important for us and they will serve as basis for our investigation. They miss the target of being a formal model for Web data management - our primary concern - for two main reasons. First, they do not directly address distribution. Second, they are tree-based and

not graph oriented, as more essential for the Web. Finally, they address static issues and miss the dynamic nature and interactivity of Web needs. For instance, tree automata are a fantastic tool for static centralized trees, whereas we have to manage evolving distributed graphs.

The playground: the Web and Web services One of the key features of the Web are Web services, that is the possibility to activate a computation on a distant site and obtain data from it or more generally interact with that data (e.g. apply updates). In some sense, HTML, the hypergraph structure of the Web, and Web search engines, may be seen as very limited forms of read-only services and Web publication as primitive write services. Web services are now giving birth to a very active area of software development. Data exchanges between peers are captured by calls to complex Web services taking into account the *logic* of these data exchanges typically specified intentionally using logical formulas. Like XML coming with a family of "standards" (XML schema, XQuery, Xpointer, etc.), Web services come with a family of "standards", in particular, WSDL for specifying the signatures of services, BPEL for specifying their sequencing, UDDI for publishing/discovering services. In this proposal, the focus is on services for managing information, e.g., query/update services or data monitoring, that is on data intensive services rather than on services performing intensive computations as in scientific grids. More precisely, we see a Web service here as a function that just happens to be supported elsewhere, with some side effect there and eventually returning a piece or a flow of data.

We have already worked for several years on a formal model based on XML and Web services that will serve as a first step towards the Webdam model.

ActiveXML ActiveXML [**?**] is simply XML with embedded Web services, i.e., with intentional information. So the fundamental idea is to place views, i.e. possibly external information, at the center of the picture. The notion of documents with function calls is in the spirit of object-oriented languages. The combination of trees and functions opens an array of new questions that are central to Web data management, some we started addressing, for instance:

1. Querying intentional data: evaluating queries when part of the data is elsewhere.
2. Incremental maintenance: maintaining a materialized view when some functions bring in continuously data.
3. Casting: choosing which functions to call to force such a tree to match a desired type.
4. Distribution: distributing a tree between several peers in the style of Ldap to distribute both the information and the processing load.

Two essential lessons we have learned from the ActiveXML project, are the following: (i) function calls should be asynchronous and (ii) data exchange should be based on data streams. But most importantly we have acquired experience in developing Web applications and on the various facets of the problem.

The ActiveXML model is more complex than the relational model because it encompasses in a nutshell most of the various themes of 20 years of database research, most notably: deductive databases [**?**], object databases [**?**] and active databases [**?**]. This together with the simplicity of the definition and experiments with real Web applications are, we believe, indications that we are on the right track. Another indication is that independent proposals seem to reach similar shores from different angles. For instance, starting from XQuery, accesses to several sources are considered in [**?**]. Although the approach may seem very different, similar issues are encountered.

In its current state, the ActiveXML model is too basic, say like the core relational model of the early days, before dependency and concurrency control theory matured. As previously mentioned, the setting is much more complex so more work is needed to achieve the goal of a general model for Web data management with the proper accompanying theory. Building on the results already obtained with ActiveXML, with the experience in designing applications with that language, we are now ready for a crucial part of the work in Webdam: developing a simple and solid mathematical model for distributed data management and building the accompanying theory to obtain the desired formal foundations. We will sketch in the next section some directions to improve the model and questions to solve about this model.

2.2 A second shift in data management: deduction

Let us reconsider the origin of the success of the relational model. A relational database system is essentially a First-Order Logic machine placed on everyone's desk to manage data. More precisely, a non-specialist can specify some needs, declaratively, in first-order logic terms. The system then compiles such a "logical query" into an "algebraic query plan" that is optimized then evaluated. Relational systems therefore perform automatic reasoning to handle queries and views, e.g., to rewrite queries into equivalent ones to optimize them. Reasoning is also present in many other aspects of relational systems, e.g., dependencies (logical formulas over the data that the system should enforce), transactions and concurrency control, triggers (i.e. active rules). In relational systems, such reasoning is in some sense "hard-wired". For instance, several transactions are allowed to access/update simultaneously the same database. The system is in charge of verifying that they do not interact improperly. To do that, the system assumes some laws governing the interactions and implements an algorithm (some reasoning) to check that these laws are not violated. The laws are decided in advance (e.g., ACID properties) and the reasoning is encoded in algorithms, the correctness of which has been proved in advance (e.g., 2phase-locking).

Similarly, we want an intelligent interface between a human being and the network that nowadays stores the data we use. On the Web, logic is needed for the declarative specification of the application, but it is also needed to describe the laws governing applications (the equivalent to the "hard-wired" logic of relational systems) since much more freedom is desired and Web data management cannot live with inflexible preconceived laws. The distribution also brings fundamental differences. The reasoning is now performed by different, typically heterogeneous and autonomous systems. This entails that reasoning must become a first class citizen in the sense that one peer may have to delegate some reasoning task to another one and that they may have to exchange (partial) proofs. Also, reasoning will typically be much less hard-wired than in relational systems, since we need to support complex interactions between peers governed by application-dependent laws that

are not known in advance. As a consequence, the system must rely on more sophisticated reasoning for instance to optimize queries in a distributed setting (between autonomous peers) or perform change control in a very dynamic distributed environment.

Deduction and reasoning have been around in relational databases, e.g. with deductive databases. The novel shift is that deduction is now used within a much wider spectrum of functionalities.

Reasoning about applications is also particularly essential when one wants to compose Web services to support some complex task (orchestration) or when several services cooperate towards a particular goal (choreography). One may want to verify that a particular contract is enforced, some quality of service guaranteed. Current Web service technology is way too limited in terms of functionalities. Tools for verifying properties of Web services, e.g. their composability, are too primitive. These issues are not particular to data management, but as we will see, the fact that the focus is on data management, does bring fundamental differences.

We briefly consider next relevant existing technologies.

Web knowledge representation languages such as RDF [**?**], SPARQL [**?**] or Owl [**?**] are essential components of the Web, in particular, for data integration. They adress issues that somewhat complement the core issues we consider. They often ignore distribution and when they provide rich means for describing changes and dynamic behavior, little is known on the automatic verification of dynamic properties. Since the developments of knowledge representation models and systems are important for the semantic Web, we will have to consider their interaction with the techniques for data management we envision. However, research on Web knowledge representation is not at the core of Webdam.

In databases, reasoning about data and queries has a long history. In particular, the beautiful theory of dependencies has emerged and query equivalence and optimization have been studied extensively. Most results were obtained for the relational model even if recent developments have been concerned with tree data and XML. Results in this areas are relevant, e.g. the chase, a general technique used in many contexts for inferring data properties. Unfortunately, most of the results deal with query processing (most of the time for monotone queries) and ignore data changes, an essential aspect in our setting.

Information retrieval and search technology are also relevant for the Web, because of the number of available resources and the difficulty to choose between them. For that aspect, we will rely on results obtained by the ERC Advanced Grant Proposal SeCo on Search Computing. We are considering collaborations between the two projects.

For reasoning about Web applications, three other computer science areas are very relevant: model checking [**?**], automata [**?**] and temporal logic [**?**]. These areas are now very mature with impressive achievements. In particular, the automatic verification of temporal properties of (even distributed) programs has made enormous progress. Unfortunately, works in these directions rarely take data into consideration. Automatic program analysis is typically limited to finite state systems whereas the presence of data immediately leads to infinite states. Researchers in these areas are aware of the limitations and have considered recently incorporating data. For instance, there has been some recent work on integrating some aspects of data into automata, logics, and model check-

ing, see, e.g., [**?**, **?**, **?**, **?**]. One can in general consider extensions of model checking techniques to infinite state systems. In particular, symbolic analysis of infinite systems based on rewriting techniques analyzed with automata techniques have already proved successful in a number of areas: communication systems, parameterized verification, program analysis, timed systems, security protocols, XML and Web services. This is the topic of a Dagstuhl seminar [**?**] and we intend to follow this line of work. There are also interesting recent works on the verification of Web services with data (so infinite state), e.g., [**?**, **?**]. We will study more systematically verification techniques adapted to our context that includes data.

Another technology is very relevant, notably because it has data at its core, namely, datalog and deductive databases [**?**, **?**]. Observe that, because of distribution, the specification of data on the Web, so issues such as query evaluation, are recursive by nature: Data in Peer 1 refers to data in Peer 2, that refers to data in Peer 1. So, we need a logic that handles recursion and datalog is one that scales to large volumes of data.

Datalog Datalog has been very popular in the 80's. In essence, datalog offers the possibility to define recursive views, i.e., associate names to recursive query statements. Unfortunately, datalog found no real application that the simple transitive closure of relations from SQL could not solve. Thus, even if sometimes silently present in relational systems via recursive CREATE VIEW statements, datalog has not been a success. The situation is very different now because of distribution. Indeed, we believe that the Web is the killer application for datalog. To motivate this claim, we next mention three recent works all relevant here that have in common their reliance on datalog:

- Trees: In [**?**], Gottlob et al specify the extraction of data from trees in datalog. Here what is useful is the notion of patterns in datalog and in particular that of tree patterns.
- Graph: In [**?**], Hellerstein et al compute rooting tables over the Internet with datalog. Here, what is useful is the recursion in datalog that permits navigating in a graph by traversing new peers.
- Trees and graphs: In [**?**], Abiteboul et al use datalog to study the diagnosis problem in a network of independent components. The "unfolding" of a computation (in the distributed Petri net sense) is captured as a tree that is distributed between different peers.

We will study datalog and extensions (in particular, with negation) in the Web setting. We will use the language (and extensions) both for describing and for reasoning about Web computations.

3 Methodology

The methodology we adopt is based on a number of principles that we develop in this section. We also mention problems that we will address. Clearly, five years is a long time frame and we expect more issues to be raised during the project.

A model for Web data management We will first develop a mathematical model for Web scale data management. We use the term model here in a very broad sense, encompassing issues such as the syntax of the data and typing, but also the specification of some processing on it (data transformations, queries, updates) as well as of behavioral properties.

The model should capture data management in typical Web applications (see examples further). In a nutshell, the setting consists in a number of autonomous systems communicating by activating services on other peers and exchanging information. Locally a peer evaluates queries and updates, communicates with other peers, and as a result its state evolves in time. Observe that we typically assume no central authority and no global knowledge. Each peer reasons and decides independently of the other peers, even though they may exchange information. Critical issues such as avoiding deadlocks or not disclosing information to peers without proper credentials have to be resolved based on local processing only.

A major aspect of Webdam is that the mathematical model will be driven by the specific domain of interest. So, in the forthcoming model, we should be able to describe complex tasks such as:

1. query processing and optimization, the most common tasks for data management, including locating the data of interest, and the management and use of (distributed) access structures such as distributed hash tables and replication.
2. change management including the management of updates, versions and the specification of policies for concurrency control.
3. data integration and in particular, dynamic data integration in the style of mashups (Web applications that combine data from more than one sources into a single integrated tool).
4. surveillance with functionalities such as complex monitoring subscriptions.
5. efficient data exchange and integration based on complex mapping rules between different peers in a static or dynamic (source changing) environment.
6. diffusion/communication of information based on structured or unstructured (gossiping) networks.
7. choreography of the activities of several services collaborating to achieve some particular data (intensive) task.

We already mentioned Active XML as a starting point. Part of the work will consist in studying logics and algebras for Web data management going beyond ActiveXML that has been previously mentioned. In particular, experience with that language have stressed its limitations in terms of control. Recent work with Segoufin and Vianu has introduced the concept of *guarding* formulas to control the activation of services. Such features in the spirit of active rules in databases or standard rules in production systems seem quite promising. Indeed, they correspond quite closely to features found in mashup systems. So the challenge will be to integrate such sophisticated notions of control and simultaneously study restrictions of the model so that reasoning about applications remains feasible.

Reasoning about Web applications We will also develop the necessary tools to reason about Web applications, e.g. verify temporal properties of a system. We will consider both static and dynamic properties, for example:

1. To optimize queries (similarly for updates), we need to automatically verify equivalence. This is typically a static property although in a Web context, one tends to blur the separation between query optimization (compile time) and processing (runtime) and are often led to modify an execution plan at runtime.
2. One may wish to determine (at compile time) whether a process is guaranteed to terminate, or whether it will always satisfy certain desired properties (no product is ever shipped before payment has been received).
3. One may want to determine (at runtime) why certain situation has been reached (diagnosis) and how to recover from an undesired state (error recovery). This has to be done automatically since the systems are typically self administered (self healing).
4. One may want to analyze the log of some run (possibly distributed between several machines) and detect a posteriori, e.g., misuses of the system or rooms for better taking advantage of resources.

Observe that one cannot separate the static part of the processing (query optimization) from the dynamic part. For instance, a mail order system obeying a certain workflow may involve many calls to databases (for stock management and billing). And one may want, for optimization reasons, to "pack" several calls to the databases. The separate optimization of each call would simply miss the point.

To attack the problem, we will combine techniques from databases and verification, using logic and automata as the cornerstone. We place as a compulsory requirement that we want to be able to reason in absence of any centralized authority in a context possibly involving a large number of peers and huge volumes of data. A measure of success for the modeling task is the range of Web management activities we will be able to describe (data exchange, surveillance, error recovery, etc.). For reasoning, they are first the nature of the reasoning, both static and on-line analysis, from optimization, to tuning, error diagnosis and recovery, automatic administration, etc.

In contrast with theories developed in the relational context, we will take into account the dynamic nature of the information. In contract with the standard works on verification, we will place information at the center of the picture so that we do not only talk about control but primarily about data (exchange, transformation, query, update, etc.)

The problem we will be addressing are complex. Two main guiding principles for the methodology will make the entire research program feasible:

1. Accept severe limitations: The limitations of the relational model were indispensable to achieve its success. We will similarly adopt severe limitations. The crux will be to still be able to capture the essence of Web data management.

2. Keep it simple: Model simplicity is always essential in mathematics and is a main factor of success in software development, and this most particularly in the context of the Web (because of the distribution, the scale, the heterogeneity and autonomy of the participants).

To illustrate the style of work that we will conduct, we next mention some first results in that spirit.

Some recent works around ActiveXML [**?**] trace the border of decidability of reasoning for a specific model for Web data management. Temporal properties are specified in a temporal logic based on Linear Temporal Logic [**?**], that allows specifying statements such as, *eventually* the system will reach the state *delivered* or *mailorder-aborted*. But we are specifically interested in statements that mention explicitly data and distribution such as: for each mailorder with a particular ID, we will eventually reach a state, where there will be a receipt with the same ID at the financial service of some department or a reject mail will have be sent to the appropriate customer. In [**?**], we use a temporal logic that integrates LTL and XML tree pattern formulas. We established the boundaries of decidability and the complexity of automatic verification for this temporal logic. From a positive angle, this work demonstrates that one can isolate models where desirable dynamic properties can be decided. From a negative one, the complexity is very high. So, why are we convinced such an approach is feasible in practice? Because the tasks we are interested in are fairly simple: fetch data, apply simple transformation rules to obtain new data or update existing data. Very strong limitations (to start, as classical in databases, abandoning Turing completeness) are acceptable for the ability to reason about programs and in particular optimize them.

Workflow and business artifacts An important aspect of the work will be to also develop a better understanding of the flow of control in these applications. The connections with workflows will in particular have to be investigated. A workflow is a reliably repeatable pattern of activity enabled by a systematic organization of resources, defined roles and information flows, into a work process that can be documented and learned [**?**]. We have witnessed a mixed success of workflow systems in industry. They are successful at an atomic level with local applications described by workflow systems more and more routinely turned into actual Web services. But the approach is much less successful in environments where one has to perform the choreography of many Web services. In particular, the workflow approach seems to be poorly adapted to the needs of business applications designers who think more in terms of rules and constraints than in terms of too constraining workflows. A new approach based on "business artifacts" has been proposed [**?**]. A business artifact can be seen as a document capturing the information that is produced during some business activity. The state of the process is obtained from the content of the artifact and a change of state corresponds to an update (typically an insertion) to the artifact. This is also the spirit of the approach we will follow, with the data and its evolution seen as the core of Web applications and not as the by-product of a complex workflow. In that sense, the model we will develop may also be seen as a theory of business artifacts. The philosophy here is that we want to unify the approaches based on data with those based on control, and thus be able to reason explicitly with applications involving both data and control.

Datalog As already mentioned, we expect datalog to play an important role in Webdam. So we will study datalog and extensions (in particular, with negation) in a Web setting. By considering datalog in a distributed context, new issues arise. For instance, simply detecting the termination of a datalog query evaluation that is trivial in a centralized context, becomes nontrivial and costly. We illustrate next three issues where datalog can be useful in our context that we will investigate:

1. Query evaluation. We will study the evaluation of datalog queries in a Web setting, with the extensional data and the intentional definitions distributed over the network. Difficulties in this context (compared to a centralized one) are that the processing may involve a large number of peers (possibly an unbounded number as in queries to retrieve music over the Internet) and that we have to optimize a datalog program that is distributed between many peers with no one having a global picture of the program.
2. Data evolution. Techniques have been developed for the incremental maintenance of datalog queries. These techniques should be adapted to our context in particular for handling monitoring functionalities in a Web setting.
3. Run analysis. One can imagine some computation going on the Web with the participants logging in journals partial distributed traces of the execution. At any time, one may want to analyze the traces, e.g. to detect patterns of usage of the system (surveillance), or explain some unexpected behavior (diagnosis). Datalog is quite appropriate to do so.

We expect more applications of datalog to come up during the project.

In spite of its simplicity and limitations, it is not easy to reason in general with datalog (e.g., containment is undecidable for datalog), and even less with extensions such as datalog with negation that we will have to consider. However, datalog is a natural setting for defining even more restricted models where reasoning is possible.

Inspiration and validation The road map up to this point may sound abstract. But we will use real applications as inspiration for our work. We will also use the applications as testbeds to validate the model and the results through prototypes.

We conclude this section by briefly presenting three of the applications that we will consider in order to validate the results. They will illustrate the kinds of Web activities we are targeting. The first example is of an industrial nature, the second is about scientific data management, while the last one is about social networks. The common ground is that all three involve autonomous systems interacting by exchanging data.

Biological data management We focus on biological data but similar concerns may be found for different kinds of scientific data as well. There are several major biological databases (e.g. genetics, proteomics) and thousands of specialized databases. As a result of the evolution of domain knowledge, these databases change regularly; the few ontologies that govern their terminologies and notably the identifiers they use also change often. Although the volumes of data are not very large, the integration of these databases poses real challenges in particular because of dependencies

between them, their highly dynamic nature and often the presence of many inconsistencies. In this context, a particular database typically imports data from a number of independent sources, restructures it, integrates it with local data, exports some data itself. The main issues are related to the management of changes, propagation, synchronization, etc. The current data models that are used, typically ad hoc extensions of the relational model, poorly capture complex aspects such as data provenance or temporal information and bring very limited responses to the scientists expectations.

In this application, we will most particularly focus on change control aspects and the management of data evolution.

Supply chain Consider the supply chain of a computer manufacturer [**?**]. The manufacturing system processes continuous flows of orders and has to cope with issues such as distant suppliers. The different participants in the P2P system are the Web servers that are used by customers, the plants that put together the computers, the banks that process payments, dispatchers that assign mail orders to plants, warehouses that act as buffers between plants and suppliers, and finally the suppliers themselves. Each participant is represented by an autonomous peer of the system. The data exchanges between participants are intense. We want to be able to specify such an application, deploy and monitor it, detect, diagnose and recover from errors, enrich the application dynamically with complex business rules, discover and integrate new partners dynamically.

We are already using this example to test a P2P monitoring system we have developed [**?**].

In this application, we will most particularly test issues related to the flow of control and novel ideas around business artifacts.

Social Networking in P2P A social networking system such as Facebook is based on the management of information about its users stored in a central repository. It is rather straightforward to develop new applications using a simple API. However, these applications are rather limited in scope. We claim that there are two fundamental flows in this setting. The first one is that it is technically rather inefficient to centralize all the data and the control in a system that is bound to become a bottleneck or end up wasting enormous resources (the Facebook farm). More importantly, many users are reluctant to give full control over their data to a provider (Facebook) who can sell it to other businesses and worse, leave such control to third parties of unknown affiliations. It is feasible to develop similar systems where personal data is kept in full control (in some proxy database) by their owners. This becomes a problem of Web data management in the realm we are addressing.

A user interacts with a system with an interface in the style of mashups. A proxy handles her data and the interaction with the community. Note that with such an approach, a user can have her own data (e.g., phone number, list of trusted friends) shared between many systems (Myspace, GoogleMail, Flickr, etc.) rather than replicated and inconsistent on the private servers of these systems.

In this last application, we will more particularly test interactions among a large number of peers.

Each of the three application has its own specificities. The Social Network one involves a huge number of peers. The Scientific one may be quite demanding in terms of quantity of data. The Manufacturing one comes with high quality of service requirements. But beyond their specificities, all three belong to a wide range of applications that are based on data management in a distributed

context with autonomous machines. More generally, the technology we will develop will be applicable to a large number of fields, including e-commerce, e-government, information manufacturing systems, social networking systems, classical and new telecom services. In many such applications, the number of participants, the complexity of their interactions and the pressure for fast deployment and evolution make manual solutions infeasible. The clear needs for declarative specifications and for being able to automatically analyze and reason about applications are essential motivations for using the forthcoming Webdam technology.

Acknowledgments The Webdam proposal owes a lot to a number of colleagues, in particular, Omar Benjelloun, Ioana Manolescu, Tova Milo, Neoklis Polyzotis, Luc Segoufin and Victor Vianu. Also, this proposal has been influenced by the work in the ANR project Docflow with Anca Muscholl and Albert Benveniste, as well as by brainstorming around the FET-Open project proposal Fox lead by Luc Segoufin. We also wish to thank Marie-Hélène Pautrat for her help in preparing the Webdam proposal.