The Urano System: Integrating Academic Data from Heterogeneous Sources to Generate Reports for Institutional Administration

Bruno Tomazela[1], Caetano Traina Junior[1], Cristina Dutra de Aguiar Ciferri[1]

[1]Departamento de Ciências de Computação – Universidade de São Paulo (USP)
Caixa Postal 668 – 13.560-970 – São Carlos – SP – Brasil

{tomazela, caetano, cdac}@icmc.usp.br

Abstract. *Most academic data generated by the staff of research centers are available in digital media, but are usually scattered among a variety of sources. Therefore, the production of administrative reports for the management of such centers and to foster research frequently requires staff members to prepare reports containing essentially the same information but organized in different formats. This leads to inconsistency, incomplete data reporting and frustration on the part of research and management staff. In this paper we describe the Urano System, which was developed to gather academic data from different sources and integrate them into a central database, enabling reports to be generated automatically. Urano collects data from the University of Sao Paulo corporate systems and from the Lattes Database, which is managed by the National Council for Scientific and Technological Development. This paper identifies the main requirements of the system and describes the techniques applied to meet them. Urano is currently being used in the Institute of Mathematics and Computer Sciences to generate reports for institutional administration. It has been providing a huge improvement on the productivity of the Institute, as previous reports were generated manually. Furthermore, Urano has been incorporated to the Tycho System, and is also being deployed to integrate academic data from the 39 Institutes of USP. Urano can be accessed through the URLs http://urano.icmc.usp.br and http://www.sistemas.usp.br/tycho/RelatorioDeptoUnidade?codmnu=00.*

1. Introduction

Generating administrative reports is a boring task for lecturers and research institute managers. On the one hand, lecturers become frustrated by the need to prepare a variety of reports containing basically the same information but organized in different formats. On the other hand, managers of research institutes are disappointed with the reports thus produced, which are usually incomplete and inconsistent because they are prepared manually and by different people.

In view of the need to adopt modern and transparent institutional measures, decision-making based on administrative reports is a practice that has been adopted with increasing frequency. For instance, based on its annual departmental reports, a research institute manager can analyze the number of publications of each department and set down new goals and guidelines. In fact, decision-making based on consolidated reports ensures greater credibility, flexibility and responsiveness to organizations, enabling them to react rapidly and positively to changing business situations.

Aiming to generate reports for the management and decision-making processes, we are developing the Urano System at the ICMC (Institute of Mathematics and Computer Sciences) of USP (University of Sao Paulo). The purpose of the system is to gather academic data from distinct sources and compile them in a central database, enabling integrated and consistent reports to be generated automatically. Urano extracts data from pre-existing sources available in digital media. Thus, when institutional reports must be produced, the research staff is not required to report data, except when the required data is not available in any source.

Urano works with two different types of data sources. The first is the university's corporate systems that are responsible for providing transactional data relating to undergraduate (the Júpiter System) and postgraduate (the Fênix System) educational activities (http://www.sistemas.usp.br). The second source is the curricula vitae of Brazilian researchers that are available in the Lattes Database (http://lattes.cnpq.br). This database is managed by the National Council for Scientific and Technological Development (CNPq) and is widely used in Brazil to provide support to national and state agencies in granting financing. It must be noted, however, that Urano was developed to allow integrating a variety of data sources, so other systems are likely to be integrated in the future. Urano can also be adapted to other environments as well.

The development of an academic data integration system aimed at generating reports for institutional administration involves several challenges. The main requirements for such system can be summarized as follows:

- The system should automatically integrate data from several sources, whose architecture, data modeling and objective are probably heterogeneous.
- The system should not require users to report data if they are already available in an existing source.
- The system should continue generating reports even in the absence of an available source.
- Reports contain data about specific periods. Therefore, the system should not generate reports with conflicting data even though the sources are updated asynchronously.
- The system is not warned about which data from what sources are updated, but should generate consistent and updated reports.
- The system should identify inconsistencies in data from different sources and integrate the data correctly, without, however, correcting those sources.
- It is important that the sources be warned when inconsistencies are detected, but the update of their data is optional and the system should not depend on it.

This paper describes the techniques employed in the development of Urano to meet the aforementioned requirements. We specially detail how the state-of-the-art research in data integration, referred to as entity resolution (e.g., Kalashnikov and Mehrotra, 2006; Bhattacharya and Getoor, 2007; Chen *et al*., 2007; Benjelloun, *et al*., 2009), was adapted to meet the specific requirements imposed by Urano.

This paper is organized as follows. Section 2 reviews related work, while Section 3 describes the proposed Urano System. Sections 4 to 6 detail the techniques employed to solve the core problem faced by Urano: integrating heterogeneous data. Section 7 shows experimental results, and Section 8 concludes the paper.

2. Related Work

Two surveys about heterogeneous data integration were published in (Halevy *et al*., 2006) and (Halevy *et al*., 2005). They focus on the academic and industrial areas, respectively. An important problem in data integration is to identify occurrences of different references to the same real-world entity. Entity resolution algorithms are aimed at automatically detecting references to the same entity and grouping them into clusters (e.g., Kalashnikov and Mehrotra, 2006; Bhattacharya and Getoor, 2007; Chen *et al*., 2007; Benjelloun, *et al*., 2009). Urano's integration uses the main idea of the G-Swoosh algorithm (Benjelloun, *et al*., 2009) as the basis to identify clusters of entities: it does pairwise comparisons. Benjelloun et al. (2009) argue that pairwise comparison is used frequently in practice, at least in the commercial world, and guarantees practical values, such as easier coding and efficiency.

Several integration systems are based on the use of mediators (McCann *et al*., 2005; Sattler *et al*., 2005; Kiani and Shiri, 2007). Integration is also an important topic in data warehousing environments (Inmon, 2005; Torlone, 2008). However, most existing works are aimed at developing generic solutions to the problems identified in (Halevy *et al*., 2006). Conversely, our paper describes the experience of developing a system that integrates academic data from pre-existing sources, such as the corporate systems of research institutes and the lecturers' curricula. An example of specialization is the fact that Urano is not warned about what data from the sources are updated. Another example is the fact that Urano also generates integrated entities from the clusters of entities. Furthermore, Urano manages data provenance, focusing on incremental integration. Data provenance allows the results of previous integrations to be used in the aid of new integrations.

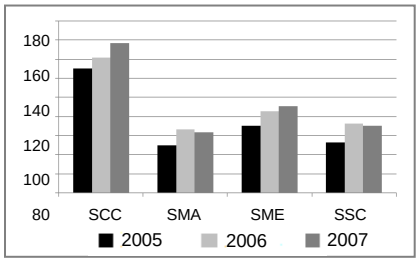
In (Dell'Aquila *et al*., 2007), it is described a data warehouse of a university composed of a set of data marts. However, the description covers only the system's architecture, without detailing any integration aspects. Conversely, our paper identifies the main requirements to be met in the development of a system that integrates academic data, examines how these requirements are satisfied by Urano, and reports the results obtained in terms of system's performance and quality.

3. The Urano System

Urano integrates the academic data of an institute's lecturers. Using the integrated data as its basis, it offers functionalities relating to the generation of administrative reports. Figure 1 shows examples of reports that can be generated: number of undergraduate and postgraduate supervisions in 2007 (Figure 1a); number of publications by year by academic department from 2005 to 2007 (Figure 1b); and list of journal publications in 2007 (Figure 1c). Urano's architecture is shown in Figure 2, and described in Sections 3.1 to 3.4. The core subsystem, UranoIntegrate, is detailed in Sections 4 to 6.

Type of Supervising	Number in 2007
Ph.D.	32
Undergraduate Studies	64
MSc	55
Others	7

(a)



(b)

...

Ali Tahzibi; Federico Rodriguez Hertz; Maria Alejandra Rodriguez Hertz; Raul Ures - A criterion for ergodicity of non uniformly hyperbolic diffeomorphisms - Electronic Research Announcements of the American Mathematical Society - 14 - 74 - 81 - 2007.

Alneu de Andrade Lopes; Roberto Pinho; Fernando Vieira Paulovich; Rosane Minghim - Visual Text Mining Using Association Rules - Computers & Graphics - 31 - 316 - 326 - 2007.

Ana Carolina Lorena; Andre Carlos Ponce de Leon Ferreira de Carvalho - Design of Directed Acyclic Graph Multiclass Structures - Neural Network World - 17 - 657 - 674 - 2007.

...

(c)

Figure 1. Examples of reports

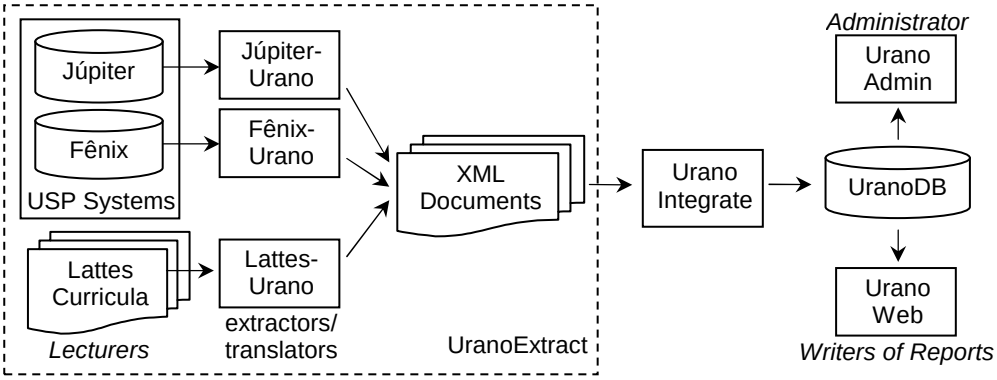


Figure 2. Architecture of the Urano System

3.1. The UranoDB Database

UranoDB is the database used by Urano to store integrated academic data. The integration is done in two levels: among lecturers and among heterogeneous sources. With regard to the first level, redundant data appearing in more than one curriculum are stored only once in UranoDB. On the second level, integrated data are stored in a single relational database, regardless of the format of the original sources.

The database is structured to store data from different categories. A data **category** is a high level concept that has a well-defined meaning in the reports. UranoDB stores data related to the following categories: (i) *lecturers*, and their departments and personal information; (ii) *papers*, such as papers in journals, proceedings, books and book chapters; (iii) *projects*, and their sponsors; (iv) *supervisions*, as of undergraduate and postgraduate students; (v) participation on *examination boards*; (vi) participation in *conferences* and *workshops*; and (vii) *courses*, such as undergraduate and postgraduate studies.

Each instance of a category is called an **entity**. With regard to *paper in journal*, the (attribute, value) pairs of Figure 3 exemplify an entity of this category. Note that an entity may be composed of other entities, such as authors (e.g., *lecturers*). UranoDB stores entities at their lowest level of granularity (i.e., detail).

paper-in-journal entity =	{(author, {(name, Bueno, R.), (order-of-citation, 1)}, (author, {(name, Traina, A.J.M.), (order-of-citation, 2)}, (author, {(name, Traina Jr, C.), (order-of-citation, 3)}, (title, Genetic Algorithms for Approximate Similarity Queries), (journal, Data and Knowledge Engineering), (volume, 62), (number, 3), (initial-page, 459), (final-page, 482), (year, 2007)}

Figure 3. Example of an entity of the category paper in journal

UranoDB also contains data relating to two other functionalities provided by Urano, which are also treated as categories: *report composition* and *log management*. To support the first functionality, UranoDB stores data that allow for the immediate generation of reports or the storage of report definitions, enabling reports to be generated at some other time. For instance, Urano allows for the definition of reports to be generated quarterly or to be available for on demand generation on personal web pages. As for the log management functionality, UranoDB stores data on inconsistencies found in the integration process (e.g., slightly different titles in similar papers).

3.2. The Administrative Subsystem

The administrative subsystem – UranoAdmin – ensures security, allowing for the submission of individual curricula and the generation of administrative reports. There are three classes of users with different permissions. *Administrators* are responsible for inserting, removing and updating the personal information of any Urano user. Writers of reports, hereinafter referred to as *writers*, are responsible for generating the reports used in institutional management. Lastly, only academic data from *lecturers* are stored in UranoDB and therefore listed in the reports.

3.3. The Extractor Subsystem

The extractor subsystem – UranoExtract – extracts data from the sources and translates them into the XML (eXtensible Markup Language), which is the internal format manipulated by Urano.

A **source** is a database of an autonomous, heterogeneous and distributed application containing academic data. As sources store data in different formats, there is a specific extractor/translator module for each source. Sources may be classified as internal or external. Internal sources are databases of the university corporate systems responsible for providing transactional data. At USP, the Júpiter and Fênix systems store data on the undergraduate and postgraduate educational activities, respectively. The external sources are the lecturers' Lattes curricula. Each Lattes curriculum contains data about all the academic activities of a specific lecturer. In the extraction process, UranoExtract considers each Lattes curriculum as a source.

The distinction between internal and external sources is based on the reliability of the data from these sources. Internal sources contain consolidated data that is assumed to be highly reliable. In contrast, Lattes curricula contain data prepared manually by different people, thus presenting typing mistakes, incomplete data, ambiguity and lack of standardization. Therefore, data from internal sources are assumed to be more reliable than data from external sources. This statement is used when the same entity is extracted from both an internal and an external source. However, internal sources usually do not provide all data required by administrative reports; so external sources provide complementary data to compose these reports.

For each category of academic data, each extractor/translator module processes the sources to extract entities of this category and to transform these entities into p-entities. A **p-entity** is an entity enriched with its provenance data, such as the source that provides the entity and its data of extraction. While an entity represents an instance obtained from a source, a p-entity represents an instance stored in UranoDB. A p-entity is defined by the triple **p-entity = <entity, source, date of extraction>**.

Before being stored in UranoDB, extracted entities that are transformed into p-entities by the UranoExtract subsystem are stored as a set of XML documents. There is an XML document for each category.

Note that Urano does not receive warning about what data are updated from the sources. To deal with this requirement, the system supports two extraction policies: by schedule and on demand. The first policy allows for the definition of specific dates and times to execute the extraction. For example, data from the Lattes curricula are extracted once a day. The on demand policy allows *administrators* to determine when an extraction should be carried out. For instance, *administrators* usually ask for data to be extracted from the Júpiter and Fênix systems during periods of course registration.

3.4. The Report Subsystem

The report subsystem – UranoWeb – generates reports. It offers a web interface through which integrated data stored in UranoDB can be transformed into useful information permanently available, independent of the sources.

Reports can differ from each other, in order to support the variety of formats and different data required for administrative reports. They may contain atomic or aggregate data. Atomic data involves the attributes of a category, such as the attributes *title* and *year* of the category *paper in journal*. Aggregate data correspond to the statistical functions applied to the fields of reports, such as counting and average.

There are different types of report, according to the category of academic data. For instance, the attributes in a report of the *paper* are different from the attributes in a report of the category *project*. The attributes exhibited are determined dynamically according to the data stored in UranoDB for each category. More specifically, the definition of the report is based on a web page that contains specific options for the required type of report. These options allow *writers* and *lecturers* to specify filters, select the attributes of the category to be included in the report, determine the ordering, and specify aggregation functions. Furthermore, users can add special fields to the report, such as the link to the DBLP page of the lecturers. Reports can be generated and saved in different formats such as PDF, HTML and Microsoft Excel format.

4. Detailing the UranoIntegrate Subsystem

The integration subsystem – UranoIntegrate – integrates p-entities, taking into account that different sources may provide complementary, redundant and/or inconsistent data, and stores them in UranoDB. Figure 4 shows the architecture of UranoIntegrate, while Figure 5 shows integration examples that will be used hereafter to illustrate the functionalities of UranoIntegrate. Figures 5a and 5d show p-entities of the category *paper in journal* obtained from two different curricula on dates 2009/03/25 and 2009/03/26, respectively.

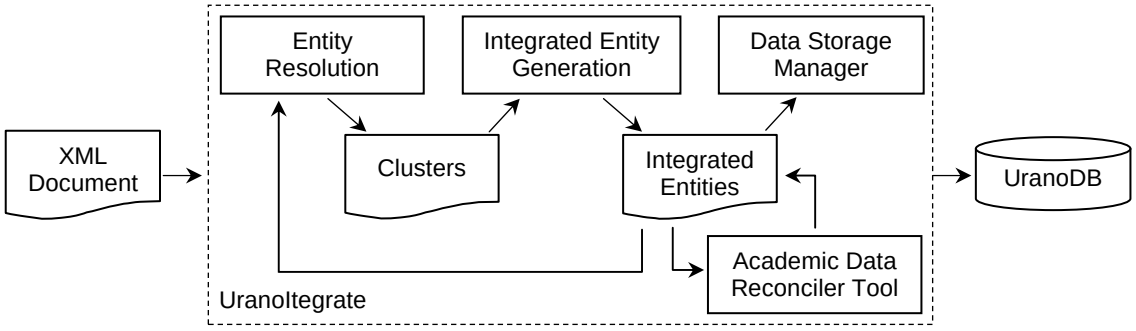


Figure 4. Architecture of the UranoIntegrate Subsystem

Execution on date 2009/03/25

p-entity$_1$ = <{(title, Integration Tests), (journal, SIGMOD Record), (year, 2000), (volume, v1), …}, Lecturer$_1$, 2009/03/25>
p-entity$_2$ = <{(title, Example of Integration), (journal, VLDB Journal), (year, 2004), (volume, v3), … }, Lecturer$_1$, 2009/03/25>
p-entity$_3$ = <{(title, Integration Test), (journal, SIGMOD), (year, 2000), (volume, v1), … }, Lecturer$_2$, 2009/03/25> **(a)**

cluster$_1$ = {p-entity$_1$, p-entity$_3$}
cluster$_2$ = {p-entity$_2$} **(b)**

integrated-p-entity$_1$ = p-entity$_1$ → integrated-p-entity$_1$ is associated to cluster$_1$
integrated-p-entity$_2$ = p-entity$_2$ → integrated-p-entity$_2$ is associated to cluster$_2$ **(c)**

Execution on date 2009/03/26

p-entity$_4$ = <{(title, Integration Tests), (journal, SIGMOD Record), (year, 2000), (volume, v1), …}, Lecturer$_1$, 2009/03/26>
p-entity$_5$ = <{(title, Integration Test), (journal, SIGMOD Record), (year, 2000), (volume, v1), … }, Lecturer$_2$, 2009/03/26>
p-entity$_6$ = <{(title, A New Paper), (journal, TKDE), (year, 2003), (volume, v7), … }, Lecturer$_2$, 2009/03/26> **(d)**

cluster$_1$ = {p-entity$_1$, p-entity$_5$}
~~cluster$_2$ = {p-entity$_2$}~~
cluster$_3$ = {p-entity$_6$} **(e)**

integrated-p-entity$_1$ = p-entity$_1$ → integrated-p-entity$_1$ remains associated to cluster$_1$
integrated-p-entity$_3$ = p-entity$_6$ → integrated-p-entity$_3$ is associated to cluster$_3$ **(f)**

Figure 5. Example of two consecutive executions

For each category of academic data (i.e., for each XML document), the Entity Resolution module processes the p-entities and groups them in clusters of similar p-entities. A **cluster** is a subset of p-entities of a same category, all of them evaluated as similar. In Figure 5b, p-entity$_1$ and p-entity$_3$ are associated to cluster$_1$, as they are

classified as similar. Conversely, p-entity$_2$ represents a different paper and is associated to cluster$_2$. The Entity Resolution module is detailed in Section 5.

The clusters are then analyzed by the Integrated Entity Generation module, which generates automatically an integrated p-entity that best represents each cluster. Therefore, each integrated p-entity is associated to its cluster. In Figure 5c, the integrated p-entities of cluster$_1$ and cluster$_2$ are p-entity$_1$ and p-entity$_2$, respectively. The Integrated Entity Generation module is detailed in Section 6.

The integrated p-entities are then stored in UranoDB by the Data Storage Manager module.

Before the execution of Urano on date 2009/03/25, UranoDB is empty. With regard to the next execution, Urano does not receive warnings about what data are updated from the sources. Therefore, it automatically identifies these updates and populates UranoDB as expected.

Using the clusters generated in the previews execution (Figure 5c), the Entity Resolution module analyzes the p-entities of the new execution (Figure 5d) and identifies that the following actions were carried out: (i) p-entity$_4$ is the same of the first execution; (ii) p-entity$_3$ was updated to p-entity$_5$; (iii) p-entity$_2$ was removed; and (iv) p-entity$_6$ was filled in the Lecturer$_2$'s curricula. The Entity Resolution module then processes the p-entities of Figure 5d, associating them to existing clusters (e.g., cluster$_1$), removing clusters (e.g., cluster$_2$) and creating new clusters (e.g., cluster$_3$), as shown in Figure 5e. The Integrated Entity Generation module then automatically generates the new integrated p-entities (Figure 5f). Finally, the Data Storage Manager module stores integrated-p-entity$_1$ and integrated-p-entity$_3$ in UranoDB and removes integrated-p-entity$_2$ from UranoDB.

The integration process may be optionally assisted by the Academic Data Reconciler tool, which aims at generating an integrated p-entity that is validated by the user. The tool: (i) compares p-entities from the same cluster; (ii) exhibits side-by-side p-entities and integrated p-entities so that users can see the differences between similar p-entities; and (iii) reconciles p-entities, allowing users to copy data from the p-entities to the integrated p-entity. The tool also allows users to move p-entities from one cluster to another, aiming at improving the precision of the generated clusters.

5. The Entity Resolution Module

5.1 Entity Similarity Detection

The Entity Resolution module generates groups of similar p-entities, using a similarity function based on an adaptation of the edit distance. Given two p-entities, the module compares each attribute value of the first p-entity with each corresponding attribute value of the second p-entity by dividing the two strings in words and comparing them word by word. Different weights are assigned to each word in a string according to its grammatical class. The comparison result of each word is then combined to give the strings similarity degree. If the combined similarity degree is above a pre-defined threshold, the strings are considered similar. Two p-entities are grouped in the same cluster when all the similarity degrees of their compared attributes are above the threshold. Otherwise, they are assigned to different clusters.

Since similarity functions are expensive, the Entity Resolution module uses the following approach to improve performance. It compares only a subset of the p-entities' attributes, instead of all their attributes. This subset is composed of one or more attributes that tend to identify each p-entity unambiguously. Therefore, the Entity Resolution module can improve performance without compromising the quality of the integration process. As for Urano, this subset of attributes was defined during the project of UranoDB, through an analysis of the characteristics of the instances of each academic data category. For example, for the category *paper in journal*, only the attributes *title*, *journal* and *year* are analyzed, instead of all the attributes of Figure 3. In Figure 5b, p-entity$_1$ and p-entity$_3$ are evaluated as similar, despite the slight differences in their attributes *title* and *journal*.

A method applied by the Entity Resolution module to string comparison based on similarity is a pre-processing that normalizes the strings by removing consecutive blank spaces, eliminating accents and cedillas, and transforming all the letters of the string to lowercase. String normalization aims at improving the integration quality, as it increases the probability of two strings to be classified as similar.

5.2 Automatic Detection of what Data are Updated from the Sources

The Entity Resolution module tackles the problem of identifying automatically what data are updated from the sources by using provenance data of p-entities. Let E_1 and E_2 be two executions carried out on dates D_1 and D_2, respectively. Let G_1 be a cluster generated by E_1, which contains the p-entities $pe_1 1$, …, $pe_1 n$ and is used as input to E_2. Also, consider that the following p-entities were extracted from source S and are currently being processed: $pe_2 1$, ..., $pe_2 m$. The automatic detection is performed as follows:

- $pe_2 j$ ($1 \leq j \leq m$) is the **same p-entity** $pe_1 i$ ($1 \leq i \leq n$) if they are from the same source, have the same values for all its attributes, and $D_1 < D_2$. In Figure 5, p-entity$_4$ is the same as p-entity$_1$;
- $pe_2 j$ ($1 \leq j \leq m$) is a **new p-entity** if it is different from all the p-entities $pe_1 i$ ($1 \leq i \leq n$) obtained from S, and $D_1 < D_2$. In Figure 5, p-entity$_6$ is a new p-entity; and
- $pe_1 i$ ($1 \leq i \leq n$) is a **removed p-entity** if it was obtained from S, is different from all the p-entities $pe_2 j$ ($1 \leq j \leq m$), and $D_1 < D_2$. In Figure 5, p-entity$_2$ is a removed p-entity.

An updated p-entity is treated as a removed p-entity followed by a new p-entity. In Figure 5, the Entity Resolution module detects that p-entity$_3$ is a removed p-entity, while p-entity$_5$ is a new p-entity.

Note that, in Section 5.1, a p-entity is associated to a cluster using similarity. Conversely, two p-entities $pe_1 i$ ($1 \leq i \leq n$) and $pe_2 j$ ($1 \leq j \leq m$) are considered the same during the automatic detection of updates if all their corresponding attributes have the same values. Otherwise, the p-entities are considered different.

5.3 The EntityResolution Algorithm

Figure 6 shows the EntityResolution algorithm, which is executed once for each academic data category. The inputs of the algorithm are a set of sources, a set of clusters

and a set of p-entities of an academic data category. The first input contains only the sources that are being investigated in the current execution of the algorithm. The second input contains the clusters of the current category that have already been analyzed in previous executions. In the first execution of EntityResolution, *clusterSet* is empty. The last input contains the p-entities to be analyzed.

For each source in *sourceSet*, the first step of the algorithm consists of setting a flag *removed* for all the p-entities of this source that are already present in *clusterSet* (lines 01 to 03). This flag is used to identify automatically what data are updated from the sources, as described in Section 5.2.

```
The EntityResolution Algorithm (sourceSet, clusterSet, p-entitySet): set of clusters
01 for all source in sourceSet do
02   set all p-entities from source in clusterSet as removed
03 end for
04 for all p-entity in p-entitySet do
05   p-entity' ← look for p-entity in clusterSet
06   if p-entity' found then
07     unset entity' as removed
08   else
09     found ← false
10     targetCluster ← null
11     while not found and there is a cluster in clusterSet to analyze do
12       while not found and there is a p-entity'' in cluster to analyze do
13         if (p-entity is similar to p-entity'') then
14           targetCluster ← cluster
15           found ← true
16         end if
17       end while
18     end while
19     if not found then
20       targetCluster ← create new cluster in clusterSet
21     end if
22     insert p-entity into targetCluster
23   end if
24 end for
25 for all cluster in clusterSet do
26   remove p-entities marked as removed from cluster
27   if cluster is empty then
28     integratedP-entity ← get integrated p-entity of cluster
29     remove cluster
30     mark integratedP-entity to be removed from UranoDB
31   end if
32 end for
33 return clusterSet
```

Figure 6. The EntityResolution Algorithm

The algorithm then analyzes each p-entity in *entitySet*, as follows (lines 04 to 24). In line 05, it determines whether the p-entity is present in *clusterSet*. If this condition is satisfied, the p-entity is the same p-entity analyzed earlier. In this case, the flag *removed* of the p-entity is unset (line 07). Otherwise, the algorithm tries to find a cluster into which to include the p-entity (lines 08 to 24). To find a suitable cluster for the p-entity, EntityResolution analyzes each cluster (lines 11 to 18), determining by similarity which cluster will contain the p-entity, as described in Section 5.1. When the p-entity does not belong to any cluster, the algorithm creates a new cluster in *clusterSet*

to hold the p-entity (lines 19 to 21). The steps of lines 08 to 24 indicate that the p-entity under analysis was updated or is a new p-entity.

After analyzing all the p-entities, EntityResolution investigates the clusters (lines 25 to 32). For each cluster, the algorithm removes the p-entities whose flag *removed* remains set (line 26), and if the cluster is now empty, it also removes the cluster (line 29). Furthermore, the algorithm marks as removed the integrated p-entity that represents the removed cluster (line 30), specifying that this integrated p-entity should be removed from UranoDB.

6. The Integrated Entity Generation Module

The IntegratedEntityGeneration Algorithm (Figure 7) generates an integrated p-entity that best represents each cluster that has been modified by the EntityResolution algorithm. A modified cluster is a cluster that contains new p-entities or had some p-entities removed.

```
The IntegratedEntityGeneration Algorithm (clusterSet): set of integrated p-entities
01 integratedP-entities ← ∅
02 for all modified cluster in sourceSet do
03   if integratedP-Entity of cluster is marked as user-validated then
04     mark integratedP-Entity as automatic-validated
05   else
06     if there is exactly one p-entity in cluster then
07       integratedP-entity ← p-entity
08       mark integratedP-entity as automatic-validated
09     end if
10     if there are exactly two p-entities in cluster then
11       integratedP-entity ← select one p-entity from cluster
12       if both p-entities in cluster have the same values then
13         mark integratedP-entity as automatic-validated
14       else mark integratedP-entity as non-validated
15       end if
16     end if
17     if there are three or more p-entities in cluster then
18       integratedP-entity ← more frequent p-entity in cluster
19       if all p-entities in cluster have the same values then
20         mark integratedP-entity as automatic-validated
21       else mark integratedP-entity as non-validated
22       end if
23     end if
24     integratedP-entities ← integratedP-entities ∪ integratedP-entity
25   end if
26 end for
27 return integratedP-entities
```

Figure 7. The IntegratedEntityGeneration Algorithm

There are three different situations related to the generation of integrated p-entities. The first refers to clusters that contain only one p-entity. In this case, this p-entity is considered to be the integrated p-entity (lines 06 to 09). In situations where the cluster contains exactly two p-entities, the algorithm selects one of them to be the integrated p-entity (lines 10 to 16). The last situation considers clusters that contain more than three p-entities. In this case, all the p-entities of the cluster are compared to each other to determine the number of times that p-entities with the same values for all

its attributes appear. The integrated p-entity is one of the p-entities that appear most. If two or more groups of p-entities appear most, the algorithm selects one of these p-entities to be the integrated p-entity (lines 17 to 23).

The integrated p-entities generated automatically by the algorithm are *automatic-validated* when the cluster contains only one p-entity (line 08), or when all the p-entities of the cluster have the same values for all their attributes (lines 13 and 20). Otherwise, the integrated p-entity is *non-validated* (lines 14 and 21). An integrated p-entity is *user-validated* when it is validated using the Academic Data Reconciler tool (Figure 4). When a cluster with a *user-validated* integrated p-entity is modified, the integrated p-entity becomes *automatic-validated*. However, no new integrated p-entity is generated for this cluster, respecting the earlier user's decision (lines 3 to 5).

7. Experiments

7.1. Implementation Aspects

The Lattes curricula of the lecturers are XML documents. The Júpiter and Fênix systems use the Sybase® Adaptive Server Enterprise 15 database management system (DBMS), while Urano uses the PostgreSQL® 8.2.3 DBMS. UranoExtract is implemented in Java, while UranoIntegrate and UranoAdmin are implemented in C++.

UranoWeb is a web application based on the request/reply mechanism of the client-server architecture. The infrastructure of the server side includes the Apache HTTP server, the PostgreSQL® DBMS and the PHP interpreter. On the client side, the development of UranoWeb is based on the recommendations of the W3C organization. The main technologies used here are CSS (Cascading Style Sheets), XHTML (eXtensible Hypertext Markup Language), JavaScript and the concepts of AJAX. UranoWeb can be accessed through the URL http://www.urano.icmc.usp.br.

7.2. Performance Results

Experiments were conducted in the Ubuntu 8.04 operating system running on an Intel Core2Duo 2.66 GHz with 4GB DDR2 800MHz of main memory and 500 GB of disk space (SATA 7200RPM). Table 1 indicates the sources responsible for providing the main categories of academic data stored in UranoDB spawning data from 1982 to present. The category *person* refers to any given Urano user.

Table 1. Sources of the main data categories

Category	Sources
Person	Lattes, Fênix
Paper	Lattes
Project	Lattes
Supervision	Lattes, Fênix
Examination Board	Lattes, Fênix
Conference	Lattes
Course	Júpiter, Fênix

In this section, we focus our performance tests on the category *paper*. Lecturers usually work on a collaborative environment, publishing their scientific papers jointly. Therefore, the integration results of the category *paper* are a good example of the Urano's integration process, as the same paper may be obtained from different curricula.

Before the first execution of Urano, UranoDB is empty. Table 2 shows the elapsed time (seconds) to process different categories of *paper* in the first execution of Urano. Data of these categories were obtained from 118 Lattes curricula. The term processing includes the following functionalities provided by Urano: extraction and integration (generation of clusters plus generation of integrated p-entities). There is no data translation because data from Lattes are stored in XML, and the XML internal format manipulated by Urano is based on the Lattes format. Table 2 also shows the number of p-entities obtained from the curricula and the number of clusters (i.e., the number of integrated p-entities) generated by the integration process. The total elapsed time to process all the categories is about 6 minutes. The time spent to process *papers in proceedings* was approximately 81.5% of that spent to process the remaining categories. There is a larger volume of papers in proceedings; therefore there are more p-entities of this category to extract and integrate.

Table 2. Results for the category paper (first execution)

Category	Extract (s)	Integration (s)	Total Elapsed Time (s)	# p-entities	# clusters
paper in journal	8.38	33.13	41.51	1642	1357
paper in proceedings	14.27	275.54	289.81	4561	3588
paper in journal (accepted for publication)	5.56	1.01	6.57	108	102
book chapter	4.58	0.63	5.21	215	184
book	4.33	0.14	4.47	97	90

After populating UranoDB, Urano processes daily to find out only those curricula that have been updated in the Lattes Database. Table 3 shows the extraction and integration results for the different categories of *paper* with regard to 55 executions of Urano (test set). The average results include days with and without curricula updates. The total elapsed time to process all the categories is about 54 seconds per day. The column *average # p-entities* lists the average number of updated p-entities during the test set. These p-entities do not necessarily generate new clusters or new integrated p-entities, as they may be associated to existing clusters.

Table 3. Average results for the category paper (55 days of execution)

Category	Extract (s)	Integration (s)	Total Elapsed Time (s)	average # p-entities	average # clusters
paper in journal	1.43	5.74	7.17	0.47	0.15
paper in proceedings	2.30	41.39	43.69	0.65	0.16
paper in journal (accepted for publication)	0.83	0.14	0.97	0.15	0.13
book chapter	0.91	0.24	1.15	0.04	0.04
book	0.95	0.12	1.06	0.02	0

For each category, Table 4 shows the number of clusters with one, two and three or more p-entities. These results reflect the number of integrated p-entities stored in UranoDB after executing the test set. The total percentage of clusters with one p-entity is 80% of the total number of clusters of all the categories. For these clusters, there is no need to further compare their p-entities during the generation of integrated p-entities. Therefore, 20% of the total numbers of clusters require additional processing to generate integrated p-entities. As for determining if two p-entities of a cluster are the same or not, we have used a hash function defined over all the entities' attributes values to improve comparison performance. Note that clusters with two or more p-entities are also used by Urano's log management functionality (Section 3.1) to store data on inconsistencies

found in the integration process. Urano allows users to generate reports containing such data to correct their curricula.

Table 4. Number of p-entities per cluster for the category paper

	# p-entities per cluster			
Category	One	Two	Three or more	Total
paper in journal	1144	169	51	1364
paper in proceedings	2804	639	152	3595
paper in journal (accepted for publication)	93	6	0	99
book chapter	160	22	4	186
book	83	7	0	90

To illustrate the precision of the integrated p-entities, we have compared manually some reports generated by Urano with the lectures' Lattes curricula. Table 5 shows the precision and recall for the reports that list paper publications in 2007. For instance, 93% of the *papers in journals* listed in Urano's report are correct. This number represents 88% of the total number of papers that should have been reported. Table 5 does not show the precision and recall of the integrated p-entities of the category *paper in journal accepted for publication* because the report refers to the year of 2007, and the p-entities of this category are now p-entities of the category *paper in journal*.

Table 5. Precision and recall of some reports

Category	Precision	Recall
paper in journal	93%	88%
paper in proceedings	90%	91%
book chapter	94%	100%
book	100%	100%

8. Conclusions

In this paper we described the Urano System, which was developed to gather academic data from different sources and consolidate them in a central database. Urano integrates academic data from pre-existing heterogeneous sources, in order to generate reports for purposes of institutional administration.

The main characteristics of Urano are as follows. It integrates data according to the main categories of academic information. It also supports two extraction policies (by schedule and on demand), allowing data from each source to be extracted according to the policy that best fits their characteristics. Another characteristic of Urano is that it extracts data from internal and external sources, without requiring data from these sources to be updated when it finds inconsistencies in the integration process. Urano stores data relating to errors inconsistencies in the integration process and, if appropriate, the sources can be updated based on reports containing those inconsistencies. Thus, the continuous use of Urano allows for the identification of inconsistencies among data and facilitates the update of inconsistent data, improving the quality and reliability of data. Finally, Urano employs concepts of the state-of-the-art research in data integration, providing an entity resolution technique that meets its requirements.

Urano is currently under use at the ICMC to generate reports for institutional administration. It has been providing a huge improvement on the productivity of the institute, since previously reports were generated manually. Furthermore, since Urano

stores integrated data in a central database and generates reports from these pre-stored data, reports can be generated quicker than before. Moreover, Urano has been incorporated to the Tycho System and is also being deployed to integrate academic data from the 39 Institutes of USP (http://www.sistemas.usp.br/tycho/RelatorioDeptoUnidade?codmnu=00).

We are currently improving Urano with data warehousing and data mining functionalities. Another project is the development of extractor/integrator modules for the corporate systems of other institutes. The structure of these modules is simple and depends only on the source and the XML formats. The modularity of the specific modules enables Urano to be expanded to integrate academic data from heterogeneous sources and to generate reports for institutional administration anywhere.

References

Benjelloun, O., et al. (2009) “Swoosh: a Generic Approach to Entity Resolution”. The VLDB Journal, 18(1): 255-276.

Bhattacharya, I. and Getoor, L. (2007) “Collective Entity Resolution in Relational Data”. ACM TKDD, 1: 1-36.

Chen, Z., et al. (2007) “Adaptive Graphical Approach to Entity Resolution”. In Proc. ACM IEEE JCDL, p. 204-213.

Dell’Aquila, C., et al. (2007) “An Academic Data Warehouse”. In Proc. 7th WSEAS AIC Conference, p. 24-26.

Halevy, A. Y., et al. (2005) “Enterprise Information Integration: Successes, Challenges and Controversies”. In Proc. ACM SIGMOD, p. 778-787.

Halevy, A. Y., et al. (2006) “Data Integration: The Teenage Years”. In Proc. 32nd VLDB, p.9-16.

Inmon, W.H. (2005) "Building the Data Warehouse", Wiley, 4th edition.

Kalashnikov, D. and Mehrotra, S. (2006) “Domain-Independent Data Cleaning via Analysis of Entity-Relationship Graph”, In ACM TODS, 31: 716-767.

Kiani, A. and Shiri, N. (2007) “A Generalized Model for Mediator Based Information Integration”. In Proc. IDEAS, p.1-5.

McCann, R., et al. (2005) “Mapping Maintenance for Data Integration Systems”. In Proc. 31st VLDB Conference, p. 1018-1029.

Sattler, K.-U., et al. (2005) “Concept-based Querying in Mediator Systems”. The VLDB Journal, 14: 97-111.

Torlone, R. (2008) “Two Approaches to the Integration of Heterogeneous Data Warehouses”. In Distributed and Parallel Databases, v.23, p. 69-97.