Towards the managment of time in data-intensive Web sites

Paolo Atzeni and Pierluigi Del Nostro

Dipartimento di Informatica e Automazione — Università Roma Tre
{atzeni,pdn}@dia.uniroma3.it

Abstract. The adoption of a logical model for temporal, data-intensive Web sites is proposed together with a methodology for the development. The model allows the definition of page-schemes with temporal aspects (which could be related to the page as a whole or to individual components of it). The design process follows a development that starts with a traditional E-R scheme; the various steps lead to a temporal E-R scheme, to a navigation scheme and finally to a T-ADM scheme. A tool associated with the methodology has been developed: it automatically generates both the relational database (with the temporal features needed) supporting the site and the actual Web pages, which can be dynamic (JSP) or static (plain HTML), or a combination thereof.

1 Introduction

The systematic development of Web sites has attracted the interest of the database community as soon as it was realized that the Web could be used as a suitable means for the publication of useful information of interest for community of users (Atzeni et al. [1], Ceri et al. [2], Fernández et al. [3]). Specific attention has been devoted to *data-intensive* sites, where the information of interest has both a somehow regular structure and a possibly significant volume; here the information can be profitably stored as data in a database and the sites can be generated (statically or dynamically) by means of suitable expressions (that is, queries) over them (Merialdo et al. [4]). In this setting, the usefulness of high-level models for the intensional description of Web sites has been advocated by various authors, including Atzeni et al [1, 4] and Ceri et al. [2], which both propose logical models in a sort of traditional database sense and a *model-based development* for data intensive Web sites.

When accessing a Web site, users would often get significant benefit from the availability of time-related information, in various forms: from the history of data in pages to the date of last update of a page (or the date the content of a page was last validated), from the access to previous versions of a page to the navigation over a site at a specific past date (with links coherent with respect to this date). As common experience tells, various aspects of a Web site often change over time: (i) the actual content of data (for example, in a University Web site, the instructor for a course); (ii) the types of data offered (at some point we

could decide to publish not only the instructor, but also the teaching assistants, TAs, for a course); (iii) the hypertext structure (we could have the instructors in a list for all courses and the TAs only in separate detail pages, and then change, in order to have also the TAs in the summary page); (iv) the presentation. Indeed, most current sites do handle very little time-related information, with past versions not available and histories difficult to reconstruct, even when there is past data. Clearly, these issues correspond to cases that occur often, with similar needs, and that could be properly handled by specific techniques for the support to time-related features. Therefore, we have here requirements that are analogous to those that led to the development of techniques for the effective support to the management of time in databases by means of *temporal database* (see Jensen and Snodgrass [5] for a recent survey and Snodgrass [6] for a textbook discussion).

It is well known that in temporal databases there are various dimensions along which time can be considered. Beside *user-defined time* (the semantics of which is "known only to the user", and therefore is not explicitly handled), there are *valid time* ("the time a fact was true in reality") and *transaction time* ("the time the fact was stored in the database"). In order to highlight the specific aspects of interest for Web sites, let us concentrate on valid time, even if transaction time could have some specific, additional facets.

In a Web site, the motivation for valid time is similar to the one in temporal databases: we are often interested in describing not only snapshots of the world, but also histories about its facts. However, there is a difference: in temporal databases the interest is in storing histories and in being able to answer queries about both snapshots and histories, whereas in Web sites the challenge is on how histories are offered to site visitors, who browse and do not query. Therefore, this is a design issue, to be dealt with by referring to the requirements we have for the site. The natural (and not expensive) redundancy common in Web sites could even suggest to have a coexistence of snapshots and histories.

This paper is aimed at giving a contribution to the claim that the management of time in Web sites can be effectively supported by leveraging on the experiences made in the database field, and precisely by the combination of the two areas we have briefly mentioned: temporal databases on the one hand and model-based development of Web sites on the other. In particular, attention is devoted to models and design: models in order to have a means to describe temporal features and design methods to support the developer in his/her decisions on which are the temporal features of interest to the Web site user.

The paper extends the experiences in the Araneus project [1, 4, 7, 8] where models, methods and a tool for the development of data-intensive Web sites were developed. Indeed, we propose a logical model for temporal Web sites, a design methodology for them and a tool to support the process (whose features have been recently demonstrated and sketched in a short paper, Atzeni and Del Nostro [9]).

The rest of the paper is organized as follows. Section 2 is devoted to a brief review of the aspects of the Araneus approach that are needed as a background.

Then, Section 3 illustrates the temporal extensions for the models we use in our process and Section 4 the methodology with the associated tool, with the help of an example. Finally, in Section 5 we briefly sketch possible future developments.

2 The Araneus models and methodology

The Araneus approach (Merialdo et al. [4]) focuses on data-intensive Web sites and proposes a design process (with an associated tool) that leads to a completely automatic generation of the site extracting data from a database. The design process is composed of several steps each of which identifies a specific aspect in the design of a Web site. Models are used to represent the intensional features of the sites from various points of view:

1. the Entity Relationship (ER) model is used to describe the data of interest at the conceptual level (then, a translation to a logical model can be performed in a standard way, and is indeed handled in a transparent way by the associated tool);
2. a "navigational" variant of the ER model (initially called NCM and then N-ER) is used to describe a conceptual scheme for the site. The main constructs in this model are the major nodes, called *macroentities*, representing atomic units of information, which often consolidate concepts from the ER model (one or more entities/relationships), and navigation paths, expressed as *directed relationships*;
3. a logical scheme for the site is defined using the Araneus Data Model (ADM), in terms of *page schemes*, which represent common features of pages of the same "type" with possibly nested attributes, whose values can come from usual domains (text, numbers, images) or be links to other pages.

The design methodology (sketched in Figure 1, see Atzeni et al. [7]), supported by a tool called Homer (Merialdo et al. [4]), starts with conceptual data design, which results in the definition of an ER scheme, and then proceeds with the specification of the navigation features, macroentities and directed relationships (that is, a N-ER scheme). The third step is the description of the actual structure of pages (and links) in terms of our logical model, ADM.

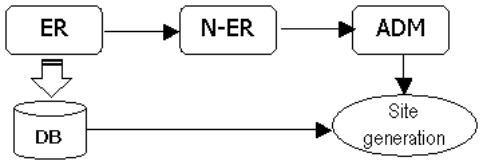

Fig. 1. The Araneus design process

Three simple schemes for the Web site of a University department, to be used in the sequel for comments, are shown in Figures 2, 3, and 4, respectively.

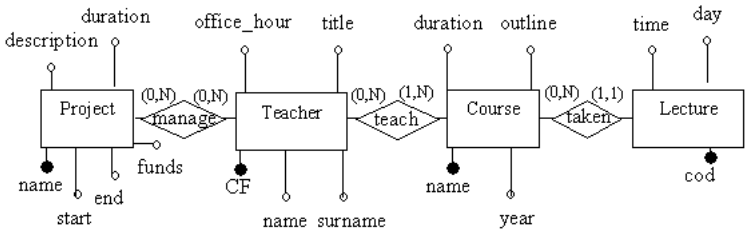

Fig. 2. The example ER schema

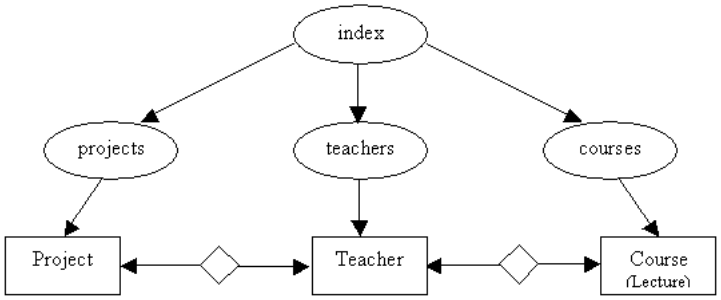

Fig. 3. The example N-ER schema

A fourth step is the specification of the presentation aspects, which are not relevant here. In the end, since all the descriptions are handled by the tool and the various steps from one model to the other can be seen as algebraic transformations, the tool is able to generate, in an automatic way, the actual code for pages, for example in JSP or in plain HTML, with access to a relational database built in a natural way from the ER scheme.

3 Models for the management of temporal aspects of Web sites

We believe that there is a need for the representation of temporal aspects at various levels during the design process, and therefore in each of our models, by means of features that are coherent with the focus of the phase of the development process the model is used in.

We propose a development process that follows the same path as we discussed in Section 2, with some extensions. and the tool we are implementing supports all phases as well. We start with brief comments on the models we use for describing our data, which follow known extensions from the temporal database literature, and then illustrate the conceptual and logical hypertext models.

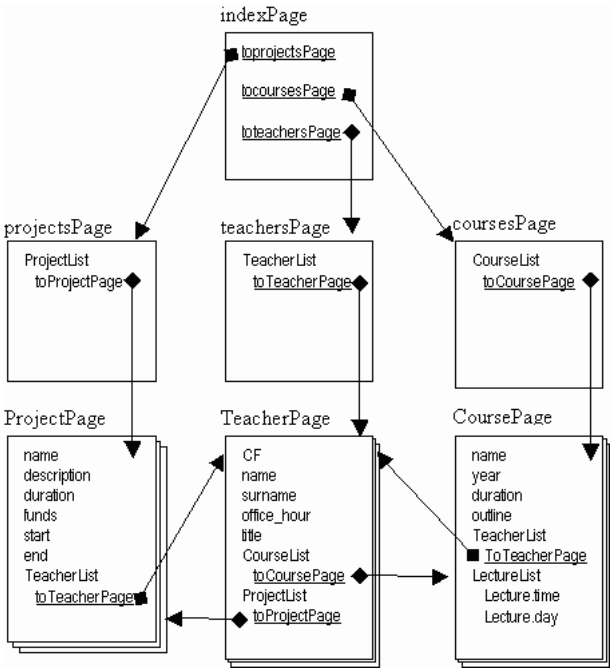

Fig. 4. The example ADM schema

3.1 Models for the representation of data

The temporal extension for the conceptual data model refers to standard proposals in the literature for temporal E-R models (see Gregersen and Jensen [10] for a survey). In the data conceptual design phase, temporal features are added to the ER scheme, by indicating which are the entities, relationships and attributes for which the temporal evolution is of interest. The temporality can be specified either for an entity (or a relationship) as a whole or for single attributes; if an entity is temporal, then it is not allowed (as not needed) to assign the temporality property to its individual attributes. By specifying an entity as temporal, the designer indicates interest in maintaining a history about its life-cycle, and especially about its creation instant and its possible deletion instant, as well as about changes for the values of all and each its attributes.

The database used to handle the data for our Web sites is relational, as in the Araneus approach, with temporal features added to it (if using our tool the temporal features are generated automatically and the developer need not have any specific competence). If an entity is defined as temporal, when a change is applied to one or more of its attributes (at the same time), the system stores the tuple as a whole. If one single attribute is marked as temporal, its history is maintained independently from the other attributes.

3.2 Temporal aspects of Web sites at a conceptual level

The two models we use to describe the structure of a Web site refer to different levels: the N-ER model considers concepts whereas ADM refers to the actual structure of pages. The same disticntion applies to their temporal extensions.

The temporal N-ER model allows the specification of whether versions have to be managed for the concepts (macroentities and directed relationships) of interest for the site, and how.

More precisely, it allows the definition of the temporality feature for each of its concepts (macroentities, direct relationships, and attributes). A concept can be defined as temporal if its origin in the ER scheme has at least a temporal component, but not necessarily vice versa. For example, a macroentity can be defined as temporal if it involves temporal elements from the T-ER model; however, we could have macroentities that are not defined as temporal even if they involve temporal elements, for example because the temporality is not relevant within the macroentity itself (indeed, Web sites often have redundancy, so an attribute or an entity of the ER scheme could contribute to various macroentities, and, even if temporal, it need not be temporal in all those macroentities).

For each temporal element, a major facet is relevant here: which version(s) are of interest from the conceptual point of view? Currently, we consider this as a choice from a set of alternatives, such as (i) the last version with a timestamp; (ii) versions at a given granularity (to be specified by the designer); (iii) all versions.

3.3 Temporal aspects of Web sites at a logical level

The logical design of a temporal Web site has the goal of refining the description specified by a temporal N-ER scheme, by introducing all the details needed at the page level: how concepts are organized in pages and how the versioning of temporal elements is actually implemented. The temporal extension of ADM (hereinafter *T-ADM*) includes all the features of ADM (and so allows for the specification of the actual organization of attributes in pages and the links between them), and those of the higher level models (the possibility of distinguishing between temporal and non-temporal page schemes, and for each page, the distinction between temporal and non-temporal attributes; since the model is nested, this distinction is allowed at various levels in nesting, with some technical limitations), and some additional details, on which we concentrate. A major choice here is the implementation of the versioning requirement specified at the conceptual level. Out of the three cases (i)-(iii) mentioned at the end of the previous section, the non-trivial ones are the second and the third, which offer the same alternatives, as follows:

- A first possibility is that the various versions are included together in the same page, each annotated with the respective validity interval.
- A second alternative is to separate the "current value" from the previous versions, correlated by means of links. The current value could be associated

with the date of the last change, whereas the versions could have validity intervals.
- A completely different organization is the "time-based selective navigation," where, for a page, the user selects the instant of interest and sees the corresponding version (and then can navigate over pages as of that date).

Additional features allow for the emphasization of recent changes (on a page or on pages reachable via a link).

The above features are expressed in T-ADM by means of a set of constructs, which we now briefly illustrate.

LAST MODIFIED This is a special, predefined attribute used to represent the date/time (at the granularity of interest) of the last change applied to a temporal element. This is a rather obvious, and widely used technique, but here we want to have it as a first class construct offered by the model (and managed automatically by the support tool) and also we think it should be left to the site designer to decide which are the pages and/or attributes it should be actually used for, in order to be properly informative but to avoid overloading.

VALIDITY INTERVAL This is another standard attribute that can be associated with any temporal element.

TIME POINT SELECTOR This is a major feature of the model, as it is the basis for the time-based selective navigation. It can be associated with pages and with links within them, in such a way that navigation can proceed with reference to the same time instant; essentially, in this way the user is offered the site with the information valid at the selected instant.

TARGET CHANGED This feature is used to highlight a link when the destination is a page that includes temporal information which has recently (according to a suitable metric: one day, one week, or whatever the designer chooses) changed. This property can be used in association with LAST MODIFIED to add the time the modification has been applied. The TARGET CHANGED feature is illustrated in Figure 5: a Department page (source) has a list of links to teacher pages (target). In a teacher page the office hours have been modified. When the user visits the department page he is informed which teacher pages have been modified (and when) so he can follows the link to check what is new. The example refers to

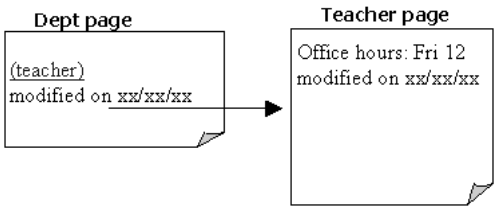

Fig. 5. The TARGET CHANGED feature

just one source and one target page, but things may become more interesting when we consider non-trivial hypertextual structures: this gives the opportunity to propagate this kind of information through a path that leads to the modified data (see Figure 6). When a new lecture is introduced, then both the teacher and the department page are informed (and highlight the change) so the user can easily know which are the site portions with modified data.

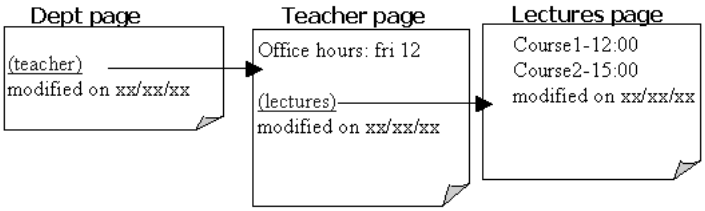

Fig. 6. The TARGET CHANGED feature along a path

REVISION LIST This feature allows for the specification that all versions of a temporal element are shown in the same page as a list of revisions.

LINK TO VERSIONS This is a special type of link that has as a target a VERSION STRUCTURE (to be illustrated shortly), handling the versions of a temporal element called. It is used when the designer chooses to have just the last version in the main page and the others held in other pages.

VERSION STRUCTURE These are "patterns" for pages and page schemes, used to organize the different versions of a temporal element and referred to by the LINK TO VERSIONS attribute. There are various forms for this construct involving one or more pages:

- SIMPLE VERSION STRUCTURE: a single page presenting all the versions for the temporal element with timestamps.

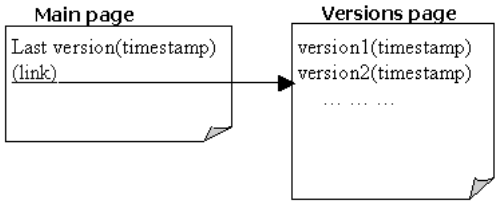

Fig. 7. The SIMPLE VERSION STRUCTURE pattern

– LIST VERSION STRUCTURE: an "index" page with a list of links labelled with the validity intervals that point to pages showing the particular versions and include links back to the index.

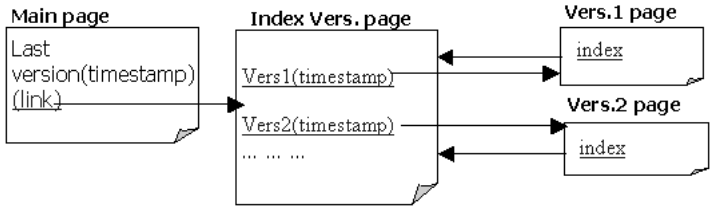

Fig. 8. The LIST VERSION STRUCTURE pattern

– CHAIN VERSION STRUCTURE: this is a list of pages each of which refers to a specific version. It is possible to scan versions in chronological order, by means of the "previous" and "next" links available in each page.

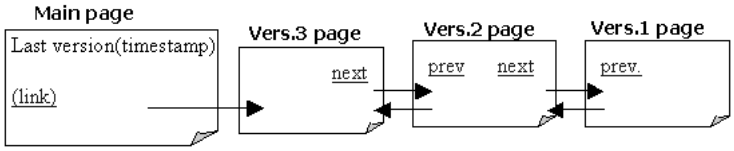

Fig. 9. The CHAIN VERSION STRUCTURE pattern

– SUMMARY VERSION STRUCTURE: similar to the previous case but the navigation between versions is not chronological. Each version page has a list of links that works as an index to all versions.

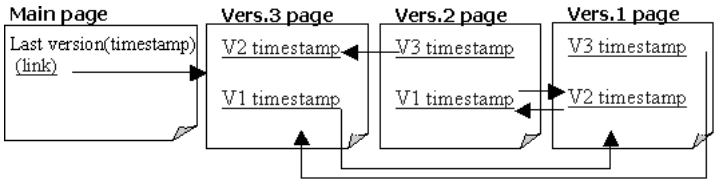

Fig. 10. The SUMMARY VERSION STRUCTURE pattern

4 The T-Araneus methodology and tool

Let us now exemplify the design process by referring to the example shown in Section 2 which, despite being small, allows us to describe the main issues in the methodology. We also sketch how the tool we are implementing supports the process itself. Rather than showing a complete example, where it would be heavy to include all the temporal features, we refer to a non-temporal example, and comment on some of its temporal extensions.

The first step is to add temporal features to the snapshot ER schema. Let us assume that the requirements specify that we need: (i) to know the state (with all attribute values) of the entity *Project* when a change is applied to one or more attribute values; and (ii) to keep track of the modifications on the *office hour* and *title* attributes for the *Teacher* entity. The first point means that the whole *Project* entity needs to be temporal, whereas for the second point, indeed, the designer has to set the temporality only for the attributes *office hour* and *title* in the *Teacher* entity.

In Figure 11 we illustrate the portion of interest of the resulting T-ER schema: the elements tagged with **T** are those choosen as temporal.

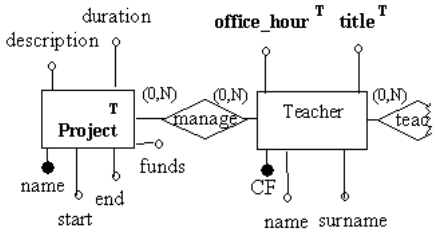

Fig. 11. The example T-ER schema

With respect to the conceptual design of the navigation, we have already shown in Section 2 the overall N-ER scheme. Let us concentrate here on the temporal features: at this point it is possible to choose how to manage versions for each macroentity and attribute. We have defined the *Project* entity as temporal in the T-ER model so the temporal database will handle the modifications but we don't want the site to show all versions so we choose here to have only the last version with a timestamp in the *Project* macroentity. It will be possible in the future to change this choice and add the projects version handling in the site simply changing this property. For the temporal attributes in *Teacher* we want all the versions to be managed. In Figure 12 the temporal N-ER scheme is shown with explicit indication of the version management choices (with the following codes: AV: all versions; LV: last value; LV_TS: last value with timestamp; VG: versions at a given granularity).

Let us then consider the logical design. As we have specified in Section 2, a standard ADM scheme can be automatically generated as an algebraic transfor-

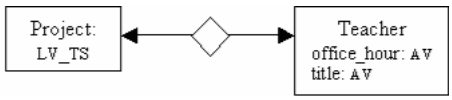

Fig. 12. Temporal features in the N-ER model

mation based on the conceptual models (it can then be restructured if needed). During this automatic generation, for each temporal element in a page scheme, it is possible to specify how to present versions starting from the choices made in the N-ER scheme. On the basis of the requirements, we could decide to handle versions for *CoursePage* by means of a CHAIN VERSION STRUCTURE. For the *TeacherPage* page scheme we could choose to handle versions in different ways for the various attributes: all versions in the main page for the *title* attribute and just the last version in the main page for the *office hour* attribute with an associated SIMPLE VERSION STRUCTURE page presenting all versions.

The T-ADM page scheme for the *TeacherPage* is illustrated in Figure 13. The *office hour* attribute is associated with the LAST MODIFIED information and a LINK TO VERSION that point to the SIMPLE VERSION STRUCTURE page scheme presenting all the versions each with a VALIDITY INTERVAL. The simple attribute *title* has instead been transformed into a REVISION LIST due to the designer choice to have all versions in the same page; again, a VALIDITY INTERVAL is associated with each version value.

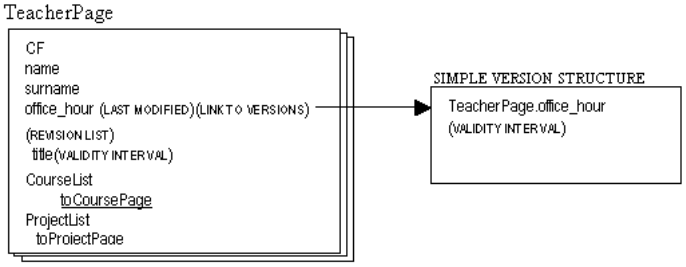

Fig. 13. A T-ADM page scheme

At the end of the design process, the tool can be used to generate the actual site, which can be static (that is, plain HTML) or dynamic (JSP); actually some of the features (such as the time point selector) are allowed only in the dynamic environment.

Currently, there is an implementation of the tool that is complete with respect to the development process even some of the features of the models are not available.

5 Future works

In this paper we have focused on one of the aspects of interest for the management of temporal evolution: the content. Other dimensions are obviously of interest for real-world, complex Web sites, and we plan to consider them in the near future. They include: (i) the presentation; (ii) the hypertext structure; (iii) the database structure. Among them, the most challenging is probably the last one: as we consider data intensive Web sites, the hypertext structure is obviously strongly related to the database structure so it could be very important to keep track of the schema evolution. If you change the ER schema (and, as a consequence, the underlying database schema), for example deleting an entity and a relationship, it can result in a change in the hypertext structure and/or the presentation. Essentially, this would be a variation of a maintenance problem, with the need to keep track of versions.

Another issue we are considering is the relationship between a temporal Web site and an associated Content Management System (CMS). Indeed, the management of the temporal database could be delegated to the CMS: updates to data would generate histories if the involved data is defined as temporal. This would completely support the management of the temporal data of interest if updates are allowed only via the CMS. We are working on a prototype for a "data-intensive" CMS to be used in conjunction with the T-Araneus tool.

References

1. Atzeni, P., Mecca, G., Merialdo, P.: To weave the Web. In: VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece, Morgan Kauffman, Los Altos (1997) 206–215
2. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kauffman, Los Altos (2002)
3. Fernandez, M., Florescu, D., Levy, A., Suciu, D.: Declarative specification of Web sites with Strudel. VLDB Journal **9** (2000) 38–55
4. Merialdo, P., Atzeni, P., Mecca, G.: Design and development of data-intensive web sites: The araneus approach. ACM Trans. Inter. Tech. **3** (2003) 49–92
5. Jensen, C., Snodgrass, R.: Temporal data management. IEEE Transactions on Knowledge and Data Engineering **11** (1999) 36–44
6. Snodgrass, R.: Developing Time-Oriented Database Applications in SQL. Morgan Kauffman, Los Altos (1999)
7. Atzeni, P., Merialdo, P., Mecca, G.: Data-intensive web sites: Design and maintenance. World Wide Web **4** (2001) 21–47
8. Merialdo, P., Atzeni, P., Magnante, M., Mecca, G., Pecorone, M.: Homer: a model-based case tool for data-intensive web sites. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA, ACM (2000) 586
9. Atzeni, P., Del Nostro, P.: T-Araneus: Management of temporal data-intensive Web sites. In: Int. Conf. on Extending Database Technology (EDBT 2004), Crete, Lecture Notes in Computer Science 2992. Springer-Verlag (2004) 862–864
10. Gregersen, H., Jensen, C.: Temporal entity-relationship models—a survey. IEEE Transactions on Knowledge and Data Engineering **11** (1999) 464–497