ERW: Entities and Relationships on the Web

Sebastiano Vigna
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico 39/41
I-20135 Milano, Italy
vigna@acm.org

ABSTRACT

ERW is an innovative system for handling complex databases using a web browser. It uses the most recent standards endorsed by the W3C to offer to the user a sophisticated environment, similar to a dedicated client. Moreover, the user interface is generated in a completely automatic way starting from a conceptual description of the database by means of an XML-based language for entity-relationship schemata.

Keywords

web database access, entity-relationship schemata, XML, conceptual modelling

1. INTRODUCTION

ERW is a system for managing complex (and possibly large) databases using a web browser. Its original purpose was allowing easy maintenance of web site data organised in a database in a heterogeneous environment, where little is known about client machines except that they will have a web browser.

ERW is based on a formal description of the database in a variant of the common entity-relationship paradigm [2]. An XML-based language, ERL (Entity-Relationship Language), is used to describe formally all conceptual aspects of the database.

From the ERL-based description, `ERtool`, a Java™ preprocessing tool *reifies* [5] the database, that is, it transforms the (abstract) description based on entity and relationship types in a set of tables that suitably implement that description. Moreover, it generates SGML documentation about the reification process, which allows one to understand exactly how the process was carried out. And, last but not least, it generates a set of configuration files written in PHP, a very powerful scripting language designed for dynamic page generation. These files, when fed into the provided run-time PHP environment, generate a set of forms that allows easy editing of the database.

Which is the difference with other web-based database administration tools? The point is that ERW knows the abstract structure of the database, and thus can offer a much more sophisticated interface to the user. In particular, the user never sees an SQL table: rather, it is presented with intuitive relationship-based operations such as "associate this entity to this entity". The run-time environment (using the configuration files produced starting from the ERL file) will modify the database tables correspondingly.

Just to make a comparison, a tool like PHPmyAdmin is to ERW as a disassembler is to a compiler/run-time environment. You can of course use PHPmyAdmin to do raw administration of the SQL tables generated by ERW, but the knowledge that ERW has of the abstract structure of the entity-relationship schema gives it the possibility of interpreting the same tables in a deeper way. On the other hand, ERW can just administrate a database generated from an ERL description, and thus it is less general.

A tool that adheres to the same design principles of ERW, but with different purposes, is WebML [1]. WebML specifies abstractly a database and a web site based on it, allowing automatic generation of editing forms and online pages. Indeed, the database specification language of WebML could be easily mapped to a subset of ERL.

There are two main differences between ERW and WebML. The first is in focus: ERW does not generate web sites, but just sophisticated user interfaces accessible with a browser. On the other hand, the database support of ERW is much wider than that of WebML, and includes relations with attributes, SQL-expressions based default

attribute values, static and dynamic enumerative types, weak entities, subtyping and so on.

2. FEATURES

1. ERW is entirely based on international standards and open-source tools. It is free software, downloadable from http://erw.dsi.unimi.it/, and it has been used for more than one year at the Computer Science Dept. of the Università degli Studi di Milano. It can use a large number of DBMS, and is completely platform independent.

2. ERW maintains *referential* and *logical* integrity of a database. That is to say, not only editing cannot create dangling references, but also cardinality constraints will be automatically enforced; for instance, if every document must have an author, then it will be impossible to insert a new document without associating to it an author.

3. ERW allows any number of users to access the database concurrently, with user, group and element-based authorisation.

4. ERW uses the W3C DOM [3] to offer a rich and intuitive graphical interface. This interface comes at no cost once the database abstract structure has been defined in ERL. All forms produced by ERW validate under the Transitional HTML 4.01 DTD [6].

There are, of course, also some drawbacks. ERW makes a number of assumptions on the structure of the database and on the inner workings of an entity-relationship schema that prevents you from doing certain things:

1. ERW is based on *local editing*: each entity type has a corresponding editing form, and from that form you can edit entities of that type and adjacent relations, but you cannot edit other components of the entity-relationship schema (you can, of course, from their respective forms).

2. ERW handles binary relations only. If you really need relation of greater arity, you will have to factor them. The main reason for this limitation is that it is very difficult to device a generic user interface for *n*-ary relations that will adapt to every situation.

3. STATELESS EDITING

The main ingredient in the design of ERW web interface is *stateless editing*. HTTP is a stateless protocol, and, as such, is not prone to support stateful interaction with a server (this is usually achieved with a mix of cookies and server sessions imposed over the protocol). ERW is designed to let the user edit freely a database recording *no* information server-side. All state related to the modification made by the user is stored client-side, so that the connection to the database is truly stateless.

This approach has a number of advantages: in particular, it adheres to the *principle of least surprise*: web users are by now accustomed to the idea that closing abruptly a web page (or even killing the browser), even in the middle of a multi-page form submission, will result in no changes being recorded server-side.

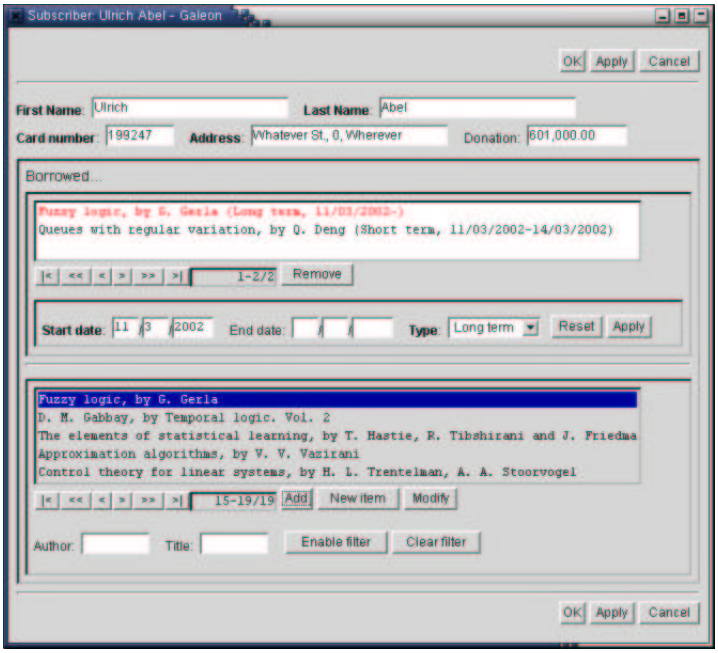

Figure 1: An example of ERW form.

Indeed, any web form is an example of stateless editing: input controls store client-side the current choice of the user. The situation, however, is completely different when we take into consideration relationship editing. In Figure 1, the user has added a relationship with a book (the relationship is evidenced in red).

The user, before submitting any information to the server, should be able to add other relationships, delete old or new ones, and edit their attributes, *without storing state on the server*. This requires a completely different approach, as the state of input controls is not sufficient to store this information.

ERW uses ECMAScript [4] scripting and the W3C DOM [3] to alter the appearance of the form, showing how relationships are added or deleted, and how their attributes are modified. *All these data are stored in the ECMAScript state of the browser*. Suitable ECMAScript code interacts with the database presenting the user with a modified view, which however does not really exist server-side. When the user has finished with editing, and submits the form, the entire ECMAScript state of the form is suitably serialised and sent to the server. The PHP run-time environment on the server locks the part of the database that is affected by the changes, checks that no integrity constraints are violated, and performs the changes. If anything goes wrong, the user is presented again with the form that was submitted: using the serialised client state, the server is indeed able to completely rebuild the ECMAScript state of the submitted form.

4. SIMULATING REMOTE PROCEDURE CALLS

ECMAScript provides no direct way to connect to a database. Usually, whenever the user has to choose among a set of elements the entire set is packed in a suitable HTML input element (e.g., `SELECT`) and sent to the browser.

This approach, however, makes it impossible to edit large databases. ERW aims at the greatest generality, and thus implements on top of HTTP a simulated remote procedure call (RPC) mechanism using a hidden frame.

Every form contains an invisible IFRAME element that is used to interact with the server. Whenever the user interface code needs some data from the database (for instance, because the user is browsing a set of entities or relationships) the location of the IFRAME element is set to a particular URL that will load into the IFRAME element some ECMAScript code; in turn, this code manipulates via the DOM the user interface, shows these data to the user, and updates the ECMAScript state. The effect is much like an RPC.

6. AN EXAMPLE

Just to give the feel of how an ERL file looks, consider the following simplified ER diagram (without attributes) that represents the loans of a library:

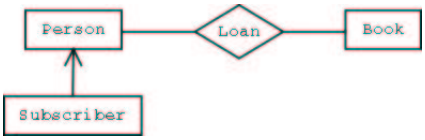

Figure 2: A simple ER diagram.

The corresponding ERL file (with attributes) would look as follows:

```
<?xml version="1.0"?>

<!DOCTYPE erl PUBLIC "-//DSI//DTD ERL V1.0//EN" "erl.dtd">

<erl id="library" title="A Library">

&ERWauth;

<enum id="termlen" type="char" size="1">
        <enumval value="L" label="Long term"/>
        <enumval value="S" label="Short term"/>
</enum>

<ent id="person" label="Person">
        <attr id="fname" label="First Name" size="30" mand="true"/>
        <attr id="lname" label="Last Name" size="30" mand="true"/>
</ent>

<ent id="subscriber" label="Subscriber" is="person">
        <attr id="card" label="Card number" type="int" mand="true"/>
        <attr id="address" label="Address" mand="true"/>
        <attr id="donation" label="Donation" type="numeric" size="11" scale="2"/>
</ent>

<ent id="book" label="Book">
        <attr id="title" label="Title" mand="true"/>
        <attr id="author" label="Author" size="40" mand="true"/>
        <attr id="publisher" label="Publisher" mand="true"/>
        <attr id="isbn" label="ISBN Code"/>
        <attr id="year" label="Year" type="integer"/>
        <attr id="description" label="Description" type="text"/>
</ent>

<rel id="loan" label="Loan">
        <attr id="startdate" label="Start date" type="date" mand="true"
                default="CURRENT_DATE"/>
        <attr id="enddate" label="End date" type="date"/>
        <attr id="duration" label="Type" type="enum" enumref="termlen" mand="true"/>
        <leg entref="person" label="Borrowed..."/>
        <leg entref="book" label="Lent to..."/>
</rel>

</erl>
```

The example uses some of the more sophisticated features of ERW, such as defined enumerative types, subtyping, relations with attributes and default values for attributes. Compiling this file with `ERtool` is all one needs to get a complete web database administration system (for an example of the editing forms, see Figure 1). The generated SQL code would look as follows:

```
CREATE TABLE book (
   id int NOT NULL PRIMARY KEY,
   title varchar(255) NOT NULL,
   author varchar(40) NOT NULL,
   publisher varchar(255) NOT NULL,
   isbn varchar(255),
   year integer,
   description text
);

CREATE TABLE person (
   id int NOT NULL PRIMARY KEY,
   fname varchar(30) NOT NULL,
   lname varchar(30) NOT NULL
);

CREATE TABLE loan (
   id int NOT NULL PRIMARY KEY,
   id0_person int NOT NULL,
   id1_book int NOT NULL,
   startdate date NOT NULL DEFAULT CURRENT_DATE,
   enddate date,
   duration char(1) NOT NULL
);

CREATE TABLE subscriber (
   id int NOT NULL PRIMARY KEY,
   card int NOT NULL,
   address varchar(255) NOT NULL,
   donation numeric(11,2)
);

ALTER TABLE subscriber ADD FOREIGN KEY (id) REFERENCES person (id);
ALTER TABLE loan ADD FOREIGN KEY (id1_book) REFERENCES book (id);
ALTER TABLE loan ADD FOREIGN KEY (id0_person) REFERENCES person (id);
```

This example is of course intentionally simple: ERW can handle any number of entity and relationship types, attributes, and provides facilities for storing files server-side. At the URL http://erw.dsi.unimi.it/ you can access an on-line demo of ERW, and browse its documentation.

5. BROWSER STANDARD COMPLIANCE

ERW heavily relies on browser compliance with respect to a number of standards. Presently only Mozilla (and thus Gecko-based browsers, such as Netscape 6 or Galeon) and Microsoft® Internet Explorer (5.5 or greater) have a sufficiently compliant implementation of these standards to work properly. Other browsers suffer mainly from lack of DOM features.

6. ACKNOWLEDGEMENTS

The author would like to thank Paolo Boldi for his collaboration in the creation of `ERtool`. Roberto Posenato and Alberto Belussi gave useful advice on the first versions of ERW, and Adriano Gugole conducted extensive beta testing.

7. REFERENCES

1. Stefano Ceri, Piero Fraternali, Aldo Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. *Proc. Ninth World Wide Web Conference*, Amsterdam, May 2000.
2. P.P. Chen. The entity-relationship model: towards a unified view of data. *ACM Trans. on Database Systems*, 1(1):9–36, 1976.
3. Document Object Model (DOM) Level 2, Version 1.0. W3C Recommendation. 13 November, 2000. http://www.w3.org/DOM/DOMTR.
4. ECMAScript Language Specification. 3rd edition (December 1999). Standard ECMA-262. ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf.
5. Ramez Elmasri and Shamkant B. Navathe. Fundamentals of Database Systems, 3rd edition. Addison-Wesley, 2000.
6. HTML 4.01 Specification. W3C Recommendation, 24 December 1999. http://www.w3.org/TR/html4/.