Aalborg Universitet

Functional specification and architecture of EIAO DW, R2.2

Pedersen, Torben Bach; Thomsen, Christian

Publication date:
2008

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Pedersen, T. B., & Thomsen, C. (2008). Functional specification and architecture of EIAO DW, R2.2: Deliverable 6.1.1.1-3. European Internet Accessibility Observatory.

EIAO
European Internet Accessibility Observatory
Project no.: IST-2003-004526- STREP

Functional specification and architecture of EIAO DW, R2.2

Deliverable Number: 6.1.1.1-3

Version:	1.0.1
Date:	2008-05-13
Author:	Torben Bach Pedersen and Christian Thomsen
Dissemination Level:	Project
Status:	FINAL

This document consists of 18 pages plus this cover

Abstract

In this document, the functional specification and architecture for the EIAO DW is presented. EIAO DW is a data warehouse that holds results from the European Internet Accessibility Observatory (EIAO) project. These results are mainly on the accessibility to disabled users of web resources that are automatically crawled and evaluated by other parts of the EIAO Observatory. However, the results also include statistics about technologies used by and linked to from the tested web resources.

Version Control

Version	*Status*	*Date*	*Change*	*Author*
1.0	FINAL	2008-04-30		Christian Thomsen, Torben Bach Pedersen
1.0.1	FINAL	2008-05-13	Minor updates based on comments from Nils Ulltveit-Moe.	Christian Thomsen

1.Table of Contents

1 Introduction

1.1 Brief project description

The goal of the European Internet Accessibility Observatory (EIAO) project is to contribute to better e-accessibility for all citizens and to increase the use of standards for online resources. By providing frequently updated data on web accessibility metrics and deviations from standards of assessed European websites, the EIAO will:

- Provide quantitative background for policy-making and targeted actions to improve web accessibility. This may be of special importance for the discussion going on about legislation (anti-discrimination laws) for equal rights related to web accessibility.
- Promote e-accessibility by raising awareness and encouraging competition through benchmarking of accessibility metrics and deviations from standards. Provide background for the discussion of standards and the development of new standards.
- Contribute to the Lisbon strategy of making Europe the Worlds most competitive knowledge based economy within 2010 by reducing deviations from standards.

All kinds of website providers can be assessed, but the focus should be on public services, governmental and local authorities, and private service providers like banks, public transport etc.

To meet the objective, a technical basis is developed in the project. This basis consists of:

- A set of web accessibility metrics.
- An Internet robot for automatic and frequent data collection on web accessibility and deviations from web standards (the WAI guidelines)
- A data warehouse providing online access to collected accessibility data

The collection of web accessibility metrics and the tools for automated data collection and dissemination will be continuously improved by feedback from end users (e.g. policy makers, governmental agencies, associations of disabled people, etc.) and user testing to sharpen the relevance of the automatically collected data.

1.2 Scope of this document

This document specifies the functionality and architecture of the data warehouse (named EIAO DW) developed in the EIAO project. Thus, the document provides descriptions of what can be done with EIAO DW functionality as well as descriptions of how the system is built. As part of this, the document specifies the conceptual data model for the EIAO DW. The EIAO DW is an integral part of the EIAO Observatory supporting the UWEM produced by the WAB cluster.

It should be noted that this is the third version of this deliverable (D6.1.1.1). The previous versions (D6.1.1.1-1 and D6.1.1.1-2) focused on release 1 and release 2 of EIAO DW, respectively.

1.3 Related work and readers instructions

This document is related to the deliverables briefly described described below.

- D3.3.1 gives the specification of the final set of web accessibility metrics (WAM) that specifies which issues to evaluate. This will, of course, influence what to store in the data warehouse.
- D5.2.2.1-2 is a working and documented implementation of the EIAO crawler for release 2.2.
- D6.1.2.1-3 defines the user interface to the data warehouse and the available reports.
- D6.3.3.1-3 is a working and documented implementation of release 2.2 of the EIAO DW.

D6.3.3.1-3 consists of a schema design (at the logical and physical levels) and an implementation of the EIAO DW. The present specifications are thus used as input to D6.3.3.1-3. In particular D6.3.3.1-3 follows the architecture specified here, give physical and logical schemas for the conceptual data model specified here, and implements the functionality specified here. The present document can be read without prior knowledge to any of the related deliverables.

2 Data model for EIAO DW

In this chapter, we present the conceptual model for the EIAO DW. This is a central part of the specification since it shows what data will be available in the DW and how this data is organised. Further, this serves as a high-level specification of the *output* of the system. The *input*, on the other hand is to be extracted from the crawler's RDF repository and the URL repository. The data from the crawler is stored as RDF (mainly EARL) in a triplestore and the Extract-Transform-Load (ETL) tool for the EIAO DW will read triples from this. The RDF schema for the RDF input data is available from http://svn.eiao.net/robacc/RDFSchema/. The URL repository holds informations about the crawled sites and URLs and is described in D5.2.2.1-2. The ETL also extracts from the URL repository. Thus, the ETL performs a mapping between the RDF repository and URL repository on the one side and the conceptual model described below.

The descriptions of the conceptual model are on purpose kept rather short, but an exhaustive description of all classes and attributes in the conceptual model can be found in Appendix A to Deliverable Number: 6.1.1.1-3.

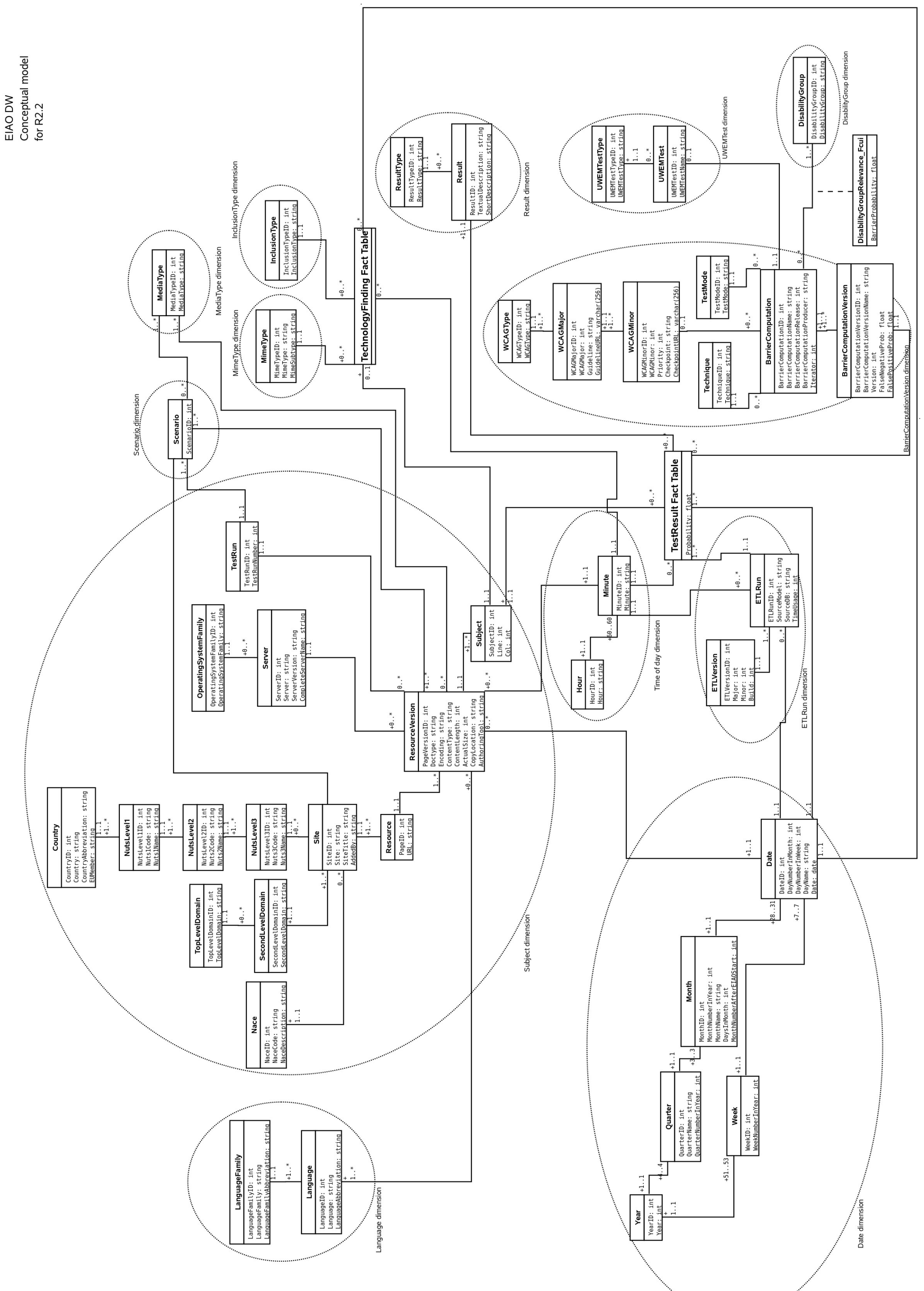

Figure 2.1: The conceptual model for EIAO DW R2.2

The conceptual model is shown in UML notation in Figure 2.1. Classes and their attributes and associations are shown. The dotted ellipses are strictly speaking not part of the conceptual model, but are included for ease. They show how the classes are grouped together as *dimensions* in the logical model (see D6.3.3.1-3). In the ellipses, the hierarchy within each dimension is represented such that higher levels in the hierarchy are drawn above lower levels in the hierarchy. For example, in the Date dimension it is seen that days roll up into months.

The conceptual model consists of 43 classes and 1 association class that together have 126 attributes. In the logical model this results in 13 dimensions and 2 fact tables. In the following, we briefly describe the purpose of each of the dimensions and fact tables. The individual attributes are not described here in the main document, but in another document titled “Appendix A to Deliverable Number: 6.1.1.1-3”.

The *Subject dimension* is used to represent information about specific, tested subjects in web resources (i.e., (X)HTML and CSS files). This can, for example, be an `h1` element on line 7, column 5 in the web page `www.example.org/webpage.html`. The dimension also holds contextual informations such as location of the providing organisation and the page's server.

The *Scenario dimension* is used to represent information about how the different web resources belong together. An HTML page may depend on another HTML file which uses a frame. Both files are then tested in the same *scenario*. These scenarios are represented by the Scenario dimension.

The *MediaType dimension* can be used to represent different kinds of media types supported by CSS documents. (Note that this functionality is not used by the current CSS WAM which only supports the media type screen.)

The *MimeType dimension* is used to represent different MIME types (“formats”) used in web resources. This could, for example, be to represent PDF files, JavaScript code and so on.

The *InclusionType dimension* represents how a specific resource may be used by another resource. For example, a web page can include a JavaScript directly in the web page's own file or it can reference another file holding the JavaScript. These different inclusion techniques are represented by the InclusionType dimension.

The *Result dimension* represents different outcomes of accessibility assessments. A result is either “Passed” or a description of what failed.

The *UWEMTest dimension* represents the different UWEM 1.2 tests used in the EIAO Observatory.

The *BarrierComputationVersion dimension* represents the barrier computation versions supported by the EIAO Observatory. They are defined in D3.3.1.

The *ETLRun dimension* contains meta data describing when and how the data was loaded into the DW. This dimension makes it easier to search for bugs and gain statistics about load operations. The ETLRun dimension is, thus, only intended for internal usage in the project.

The *Time dimension* represents different times during the day. It does not represent times on specific dates. So, for example, it can represent the time “3:45”, but not “3:45 2006-11-11”.

The *Date dimension* represents specific dates such as “2006-12-24”.

The *Language dimension* represents natural languages used by web pages.

The *TestResult fact table* (strictly speaking, it is a *class* in the conceptual model, but in the logical model, it will be a *fact table*) represents the execution of a specific barrier computation version on a specific subject at a specific date and time that gave a specific result which was loaded into the DW during a specfic ETL run. Thus TestResult is used to represent accessibility data for the tested web resources.

The *TechnologyFinding fact table* (again, this is strictly speaking a *class*) represents that a specific subject uses (either by referencing it or embedding it directly) a monitored technique (such as JavaScript or PDF) with a specific MIME type and when this result was found. Thus TechnologyFinding is used to represent which technologies the Observatory has found in the tested web resources.

3 Functionality

In this chapter, we describe the functionality supported by EIAO DW.

EIAO DW is delivered in more main releases, called R1, R1bis, R2, and R2.2. R1 and R1bis (which is like R1 but with added functionality, mainly a graphical user interface) have already been delivered as well as R2 which added many new features. The current R2.2 release improves R2 with UWEM 1.2 support, bug fixes and improved performance

3.1 Functionality supported

In this section, we give summary descriptions of the main functionalities supported by EIAO DW release 2.2. The section is split into subsections that group related functionality descriptions.

3.1.1 Data

WAMstorage: WAM result storage
Description: The EIAO DW provides storage of WAM accessibility results (as defined in D3.3.1) for web pages, for use in analytical queries. This includes to store data from the B-WAMs (barrier computations) and calculate and/or store C-WAM values. Further, meta data from the so-called M-WAMs can be stored. The results are stored in an analysis-oriented DW format, not, e.g., in EARL format. In particular, the results are stored in a *dimensional* model which is specified conceptually in Chapter 2. The results may include both automatically and non-automatically assessed results and user feedback, as long as these results are reported in EARL format. All results are timestamped to allow the DW to monitor trends over time.
Links: The final WAMs are described in D3.3.1 D6.6.1.1.1-3 describes the web pages that are tested. Note, however, that EIAO DW is not limited to only support data for web pages listed in D6.6.1.1.1
Comment: The focus on analytical queries distinguishes the DW from, e.g., the RDF repository that also stores WAM results, but for operational purposes.

UpdateWAMspecs: WAM specifications may be updated
Description: The definition of a WAM may be updated, and the DW supports measurements with several versions of a WAM.
Links: D3.3.1 describes *barrier computation* versions which handle the concept of versions of WAMs.
Comment: This is represented by the BarrierComputationVersion class in the conceptual model presented previously.

3.1.2 ETL

Fullload: Support for full load
Description: The EIAO DW provides functionality for doing an initial, full load, transferring all the data available in the data sources to the (empty) DW.
Links:
Comment: A full load needs to be executed only when the DW is first started, or when very major restructuring of the DW database has been performed. The initial load is potentially very heavy and can take many hours to execute. The initial load can be initiated by the DW administrator.

Incload: Support for incremental load
Description: The EIAO DW provides functionality for doing an incremental load, that transfers *only* the data in the data sources that has *changed* or *added* since the last DW load into the DW. When the Extract-Transform-Load tool is started, it is specified which data sources to load data from.
Links: The procedure for how and when to start an incremental load are described in D5.2.2.1-2.
Comment: The incremental load for accessibility results for one web site should be able to load in a few minutes. The incremental load can be initiated by the ETL server described in D5.2.2.1-2 or the DW administrator. We note that a WAM measurement is never “updated”, a new measurement at a later date is considered a separate measurement.

Monthload: Monthly loads
Description: The DW is refreshed once a month using several incremental loads.
Links: Incremental load is described in “Incload” above. The crawling process and calls to start loading the DW are described in D5.2.2.1-2.
Comment: The load of the DW runs with the same frequency as the crawling process as described in D5.2.2.1-2. The incrementally loaded data sets will all be made available for reporting and querying at the same time, in particular once a month.

EARLRDFinput: The input accessibility data is in EARL/RDF format
Description: The main input data on accessibility results comes in EARL format, more precisely a precisely defined subset of EARL corresponding to the results delivered by the WAMs. Other input data such as meta-data, etc., comes in an RDF based format.
Links: D5.2.2.1-2 gives the RDF schema for the input data.
Comment:

NoBadData: Unreliable data is not loaded into the DW but logged
Description: Data from site surveys that are marked with eiao:unreliableWarning or eiao:notApplicable is not loaded into the DW. Instead information about such site surveys is written to a log file to allow for further inspection.
Links:
Comment: This ensures that only reliable data is inserted in the DW. Further, this makes the design and use of the DW easier.

Reposinputdata: The input accessibility data is read from the RDF repository
Description: The DW ETL program reads its input accessibility data from the WP5 RDF repository which stores the test results in EARL format. This is the case both when doing initial and incremental loads.
Links: The RDF repository is described in D5.2.2.1-2.
Comment: We need to be able to extract the accessibility results that have been added/changed in the RDF repository since the last incremental load "the delta") in a very efficient way. This is handled by the ETL server process described in D5.2.2.1-2.

URLinputdata: The input site data is read from the URL repository
Description: The DW ETL program reads its input data with information about URLs and sites from the WP5 URL repository which stores the data in a relational database.
Links: The URL repository is described in D5.2.2.1-2.
Comment:

Importextdata: Import of external data
Description: External accessibility test results not performed by EIAO crawler, that the providers want included in the EIAO DW may be stored in the EIAO DW and used in combination with the existing data.
Links:
Comment: The inclusion of external data has implications on the reporting functionality. The external data should be placed in the DW using the normal mechanism, i.e., the DW WAM plug-in interface and the ETL component. To do this, the data is inserted into the RDF repository and loaded from there.

3.1.3 Operations

DWhours: The DW is operational from 9-16 CET on weekdays
Description: The DW is operational from 9-16 CET on weekdays only.
Links:
Comment: It is necessary to have regular down-time intervals, e.g., every night, where finalisation of loads, maintenance, upgrades, bug-fixing, etc. can be performed. The exact up-time can be discussed, but it is better to have a shorter up-time where things always work, than a claimed 24*7 up-time where the system is often not available anyway. This is especially true with respect to the trustworthiness of the EIAO project. The up-time can be improved later when more experiences have been gathered and it is known that the entire system works well.

DWhosting: The DW is hosted at AUC
Description: The DW is hosted at, and managed by, AUC.
Links:
Comment: The DW is developed at AAU and AUC.

DWexport: Export of DW data
Description: The DW provides a mechanism for exporting a selected subset of its data. The data should be exported in an XML-based format or in a PostgreSQL specific dump format.
Links:
Comment: This functionality was discussed with the project officer at the kickoff meeting. This can be used if external parties want to get some data for additional, specific analyses. This service should be provided asynchronously such that interested parties request a subset of the data. When a request has been accepted by an administrator, the data is exported and made available to the user who requested it. This is necesseary to avoid abuse of the system such as repeated dumps of large data sets which will make the system perform badly. The tool RelaXML developed during the project, provides an efficient and easy way to dump data to XML format.

3.1.4 Graphical User Interface

DW_web_inter: Web interface to DW
Description: The EIAO DW is accessed using a web interface running in a standard web browser. This provides only read access.
Links: The user interface is described in D6.1.2.1-3.
Comment: The only way for outside users to access the DW is through the web interface, no other client tools, etc., will be developed.

DW_web_acc: The DW web interface is accessible
Description: The DW web interface is accessible according to the accessibility standard WCAG 1.0 level AA.
Links: The user interface is described in D6.1.2.1-3.
Comment: Currently, the user interface is developed to conform with WCAG 1.0 level AA. Further, it is independent of type of browser used and specific fonts or font sizes.

DWrepstoproc: The DW provides stored procedures for standard reports
Description: The DW provides stored procedures that return the results for the standard reports offered by the GUI.
Links: D6.1.2.1-3 defines the needed reports offered by the GUI.
Comment:

3.1.5 Performance

Basicpreagg: Basic use of pre-computed aggregates
Description: The DW is be able to use pre-computed aggregates in a basic way to support faster queries.
Links:
Comment: Results for aggregations can be pre-computed once and then stored in a table. The stored procedures that provide results to the reports can then look up the results those results instead of calculating them from scratch. This gives significant performance improvements.

DWeffic: The DW is efficient
Description: The DW handles the data efficiently such that standard reports can be answered quickly (within 45 seconds).
Links: D6.1.2.1-3 defines the reports that are available from the GUI.
Comment: For each of the standard reports defined in D6.1.2.1-3, the DW should be able to find the relevant results within 45 seconds (in the current implementation most standard reports are answered in much shorter time). Note that this does *not* hold for custom queries since it is not possible to optimise the DW for any possible query.

DWscaling: The DW scales well
Description: The DW scales to handle data for 10,000 sites pr. month in 24 months. In particular, the DW can store such data amounts and the DW can still be incrementally loaded with data for 10,000 sites within 20 days and answer standard reports within 45 seconds.
Links:
Comment: The time needed to load data and to answer queries for data within one month is not significantly affected by the number of months for which the DW already holds data.

3.1.6 Reports supported by stored procedures

A number of reports are supported by stored procedures implemented in the DW.

Supporting the reports with stored procedures gives a good performance. Further, it gives flexibility. It is possible to make changes to the schema, e.g., introduce partitioning without affecting the GUI in any way. In this section the supported reports are described. Note that this is not a description of the implementation of these. Thus, names and detailed descriptions of the implemented procedures are not given, but are given in D6.3.3.1-3. Mathematical specifications of the reports are given in D3.3.1. The graphical presentations of the reports are described in D6.1.2.1-2.

In general, it holds for all the reports, that a test run identifier must be explicitly given. All reports either consider a specific test run or compare data from two specific test runs. For those that compare results, the DW will *not* support to do the actual comparisons. Instead the GUI must make two calls to the DW to get the data to compare and then perform the comparisons itself. This reduces the number of procedures to implement by a factor 2 and does nearly not cause extra work for the GUI.

In the GUI's aggregated reports, it is possible to do aggregation based on regions[1], sectors[2], and a combination of these. This means that there should be stored procedures that are given a NUTS code (or another region code to handle, e.g., all EU countries), a NACE code, or both. The stored procedures should return the average, margin of error for the average, maximum and minimum of the CWAM barrier probability results for the sites that are covered by the given conditions.

Further, there should be stored procedures that given the same inputs, return lists of technologies found on these sites including average number of occurrences, external[3] links and internal[4] links on the pages.

The GUI also offers reports that consider the barrier indicators of one site. To support these reports, the DW must provide a stored procedure that given a site name (as a web address) as input, returns the average, the margin of error for the average, maximum and minimum of the page barrier probabilities calculated by the CWAMs.

Finally, the DW should provide stored procedures that consider a single page. One such stored procedure should return the barrier indicator for the page scenario (that includes the (X)HTML and possibly a number of CSS files). Another should return a list of occurrences of technologies used/linked from the page.

D 6.3.3.1-3 lists the names of the stored procedures and describes the arguments of each stored procedure.

3.2 Functionality not supported

To avoid misunderstandings, we also briefly list functionality that is **not** supported by the EIAO DW.

– The page source (HTML, etc.) is not saved in the EIAO DW.

[1] We consider both *geographical* regions (based on NUTS) and *political* regions based on EU membership, EU candidate status, and EFTA membership.

[2] Represented by NACE codes.

[3] By an *external link* we understand a link from a resource on the domain *x* to a resource on a different domain *y*.

[4] By an *internal link* we understand a link from a domain *x* to a resource also on domain *x*.

- The actual EARL source for the results is not saved in the EIAO DW.

4 Architecture

In this chapter, we describe the architecture of EIAO DW. First, we give a short overview of the architecture and then we give more detailed descriptions.

4.1 Overview

Pictorially, the EIAO DW system can be represented as shown in Figure 4.1

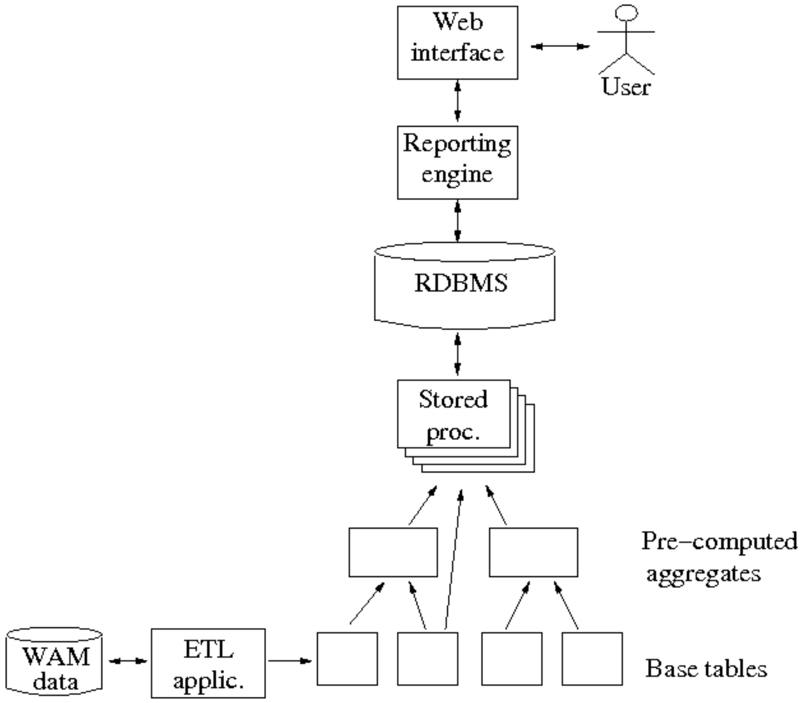

Figure 4.1

Figure 4.1 shows that the user interacts with a web interface. The communication with the user's web interface is done by a reporting engine that defines which reports are available and presents data to the user. The reporting engine invokes stored procedures in the relational database management system (RDBMS) or uses SQL directly. Then, data is extracted from the base tables and pre-computed aggregates (there is no difference between these kinds of tables from the RDBMS's point of view). The data is then sent to the web interface used by the user via the reporting engine. Further, the figure shows that WAM test result data is loaded into the database by an ETL tool. The ETL tool extracts data from a repository where the crawler stores its results.

4.2 Description of architecture

In the following, we give more detailed descriptions of the architecture of EIAO DW.

There are two basic ways to implement the data warehouse: Either as a ROLAP (Relational On Line Analytical Processing) system where relational database technologies are used to store the data or as a MOLAP (Multidimensional On Line Analytical Processing) system where specialised data structures are used [1]. With respect to scalability and the possibility of redefining facts, ROLAP system are generally preferable [1]. Both of these issues are important for the EIAO DW. Therefore, the data warehouse is based on a relational database and is thus a ROLAP system.

With respect to the schema used in the data warehouse, we use a star schema at the logical level. Thus, a star schema is what is seen from the outside of the data warehouse when, e.g., queries are written. This makes it easier to write queries and to browse the data in the data warehouse.

The Bus Architecture by Kimball [2] is used. This implies that the dimensions used in the data warehouse must conform and fact definitions are standardised. In this way, it is possible to build and extend the warehouse in small steps. Further, extra data can be added incrementally. In this way, it is also possible to add “external” data to the EIAO DW and use it in combination with the existing, “internal” data. This external data could, for example, be “high-level” statistical information that should be included in the EIAO DW for comparison purposes. To include such data, a number of new fact tables and possibly some dimension tables could be added. Other dimensions (for example Date and Time) could be shared (i.e., used) by all the fact tables. Note that this integration of external data is not used in release 2.2, but can be done later if needed.

At the physical schema level, other solutions than a star schema are applied. For example, horizontal partitioning may be used such that instead of having one fact table with data for many months (and thus extremely many rows), the system has many fact tables where each holds data for one month (and thus significantly fewer rows than an all-encompassing fact table). This kind of partitioning can, however, be done in the physical layer and be transparent to the user. The final schema design (logical and physical) is given in the documentation for D6.3.3.1-3.

Based on a survey of open source business intelligence tools [3] it has been decided that the EIAO DW will not use any existing open source OLAP server. Instead, specialised solutions for the purposes of this project are used. We believe this is an easier way to achieve the functionality and performance we require. Further, this reduces the software requirements to the server, as the only available open source OLAP server is Mondrian which requires Java and Tomcat.

The loading of data into the data warehouse is done by a specialised Extract-Transform-Load (ETL) tool. The reason for not using existing generalised open source ETL tools, is that they still not have reached an acceptable level of quality as described in [3]. Further, the input data (EARL and other RDF forms) is atypical and needs specialised handling.

Test result data from the crawler is given to the data warehouse as RDF triples. The data is extracted from the RDF repository. That repository is holding the EARL test reports from the performed tests.

During the release 2 and 2.2 development time, a tool for on-line integration has also been developed. With the tool, it is possible to do “near real-time data warehousing” such that data floats directly into the DW. This can be done by holding the newly added data in main memory, but such that both the new and historical data can be queried efficiently and in a way where the data all looks the same to the user that does not have to consider whether the data is in memory or in tables. At regular intervals the data held in memory is bulk loaded into the underlying DW tables. This happens transparently to the user. If the crawler later is extended to do “on-demand crawling” of web sites, the data can thus immediately be inserted into the DW and used from there.

As previously described, basic use of pre-computed aggregates (or “materialized views”) is to be used to improve the performance for most used queries. This is handled by the stored procedures that implement the CWAMs and the ETL tool in cooperation. The result of this is that the user can get answers to reports nearly immediately, also for reports that aggregate over very large data sets such as all tested domains in the European Union.

5 Platform

The software platforms used are PostgreSQL [4] version 8.3 running on a Linux platform on x86 processors. The ETL tool is written in Python 2.4.

6 Summary

We have specified the functional requirements and architecture for the EIAO DW, with a special focus on release 2 issues. The main challenge in the further development is the coordination with the WP3 and WP5 developments, given the physically separate development teams. This will be handled by frequent phone meetings and other interaction, as well as a project workshops if needed.

Bibliography

1: Pedersen, Torben Bach; Jensen, Christian S., Multidimensional Database Technology, IEEE Computer 34(12), 2001
2: Kimball, Ralph et. al., The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses, 1998
3: Thomsen, Christian; Pedersen, Torben Bach, "A Survey of Open Source Tools for Business Intelligence" in Proc. of DaWaK'05, 2005
4: PostgreSQL Global Development Group, PostgreSQL, 2006, www.postgresql.org