BRANDROCK

SOFTWARE

Using Metadata Documents to Design, Develop & Maintain Databases

By Peter Brand of BrandRock Software

21 August 2008

Copyright Notice & Disclaimer

This edition was published August 2008.

WHO WE ARE

BrandRock Software is a sole proprietorship company operating out of Cape Town, South Africa. Headed up by Peter Brand, a veteran Software Developer since 1981 who has had international working experience in United States of America (USA), United Kingdom (UK), New Zealand and South Africa.

BrandRock develops software tools for developers, and software applications for commercial enterprises.

CONTENTS OF THIS PAPER

This paper discusses how by concentrating on the metadata level to design, document and deploy databases, expensive and disastrous software development mistakes can be avoided . It shows how the metadata level can be used to improve the software development life cycle, providing the ability to analyze, document and control the database development process.

Introduction

The heart of any application that stores data, is the back-end database or databases that it uses. As a result, a significant proportion of the development team's efforts are spent working with the database: designing and understanding its structure, coding software to interface to it, populating the database with static data, and a host of other related tasks. Failure or oversight in any one of those tasks can prove expensive or even disastrous to the application and the enterprise.

Database Development Life-cycle

State of play

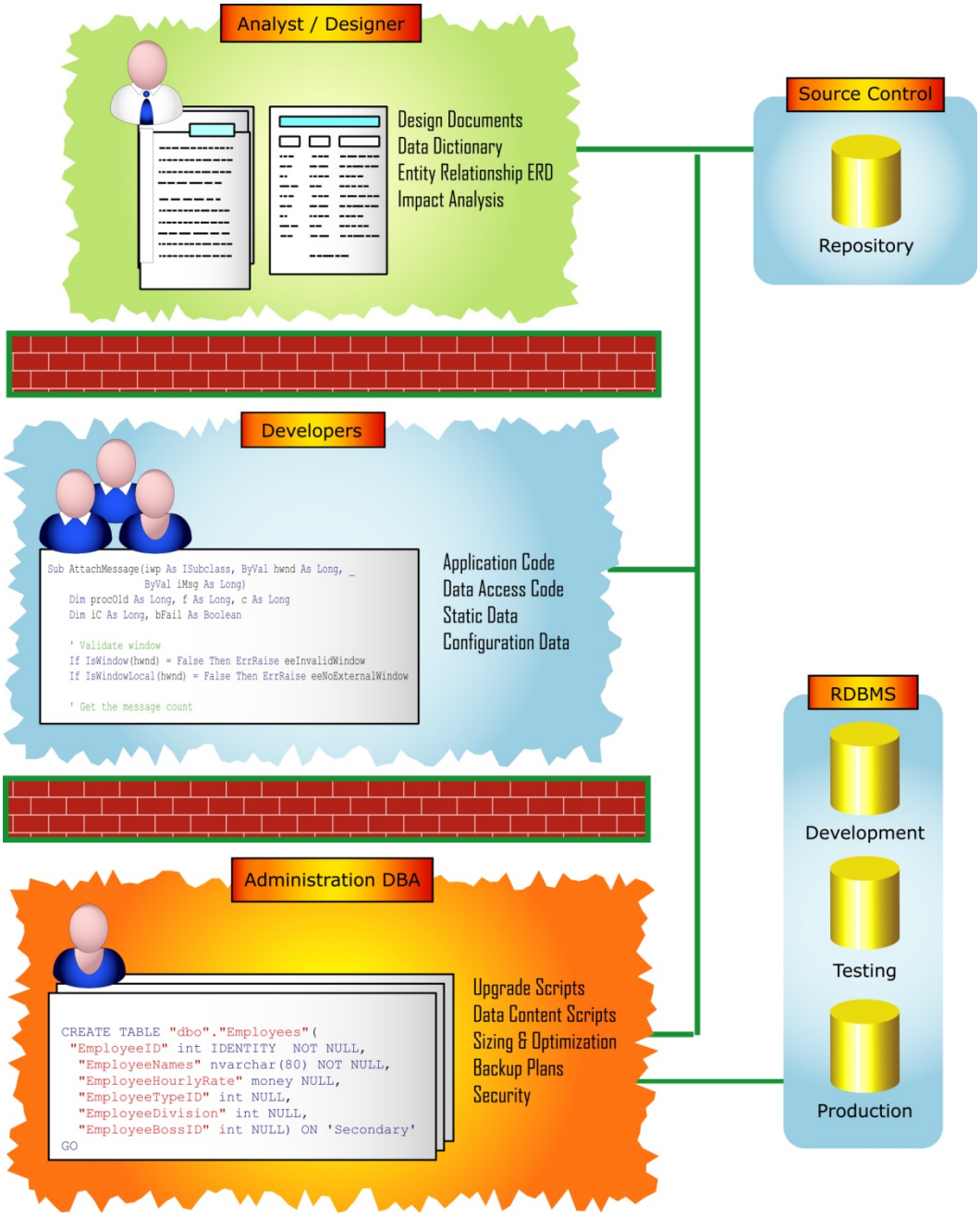
Typically, development teams are organized into islands of responsibility with limited or restricted flows of information between them.

Various tools and methodologies for database development are used to varying degrees of success. Designer uses one set of tools to design, and documents or data dictionary with varying levels of detail to support their design. Application code to interface to the database is maintained by hand by the development team. Deployment of structure changes and data content changes are manually documented and manually and repeatedly done on each instance of the database; development, testing and production by the administration team. Version control is carried out by storing various documents in the audit trail repository. Utilities have been created to compare databases to verify correct deployment and for analysis. All of this serves to get the job done, but it is limited, labor intensive and prone to error.

Database Management Systems (DBMS's) do not position themselves as tools for the complete database development life-cycle. They focus on providing the tools to enable a Database Administrator (DBA) to configure particular instances of a database. They do not provide documentation helpful for developers to understand the design or to generate data-access level code. They do not provide the ability to extend the database model beyond the proprietary properties that they provide. They are tools for a DBA, not for a developer or analyst.

Database model design tools on the other hand, distance themselves from any particular DBMS but then make themselves of little use to the DBA.

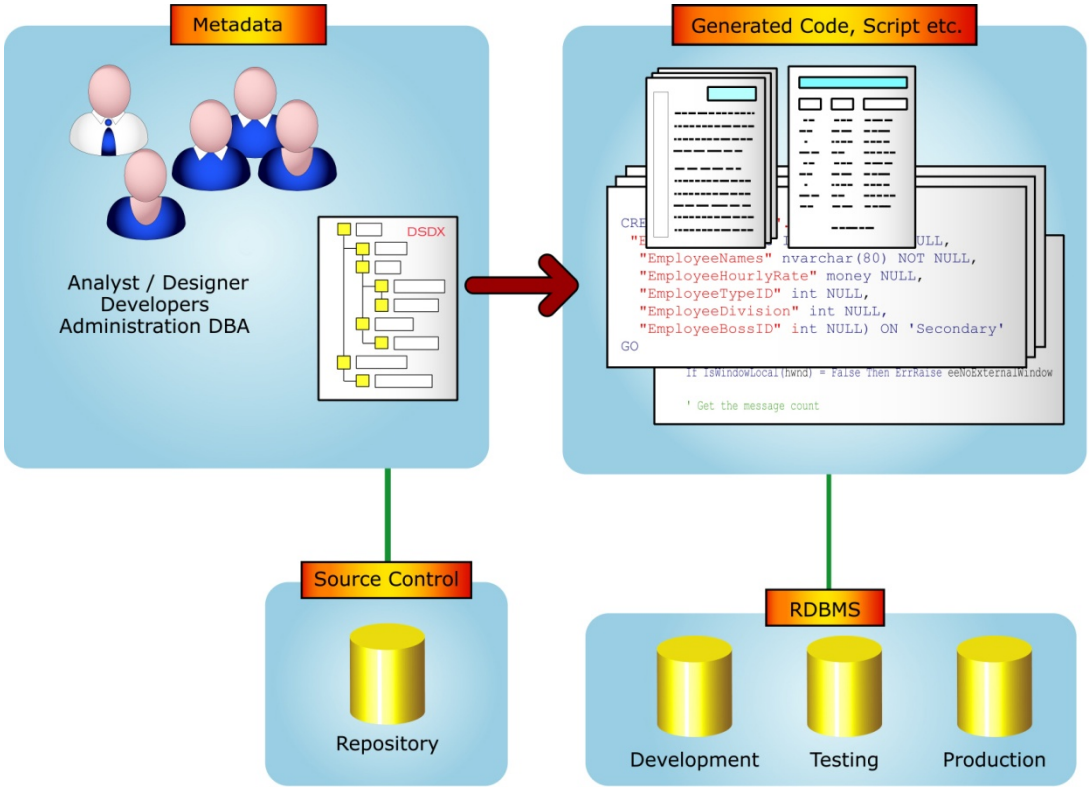
Software developers function on an island between the designers and the administrators. They have to interpret and understand the database design, and rely on the administrators to implement database changes in tandem with their software changes.



Ultimately, the "Database" is defined by a set of documents and scripts and by one or more instances on servers in the enterprise.

Rising to the Metadata level

To get some leverage in the process, an obvious method would be to have a single document that pulls together the needs of Designers, Developers and Administrators, covering all aspects of the database and its instances. Effectively pulling all team members onto to the same island, facilitating better communication and efficiency.



The metadata document (DSDX in the diagram above) should contain the following information:

- Database and Version identification: version number, company & designer details, general information.
- Database Structure: all information necessary to create the database tables, indexes, foreign keys etc.
- Database Code: all non-structural information such as stored procedures, functions, views etc.
- Instance specifications: information for the instances of the database (Development, Test and Production) such as size, physical location etc.
- Database Security: all users, groups and their access permissions matrix to each of the instances above.
- Design Documentation: not only a data dictionary which describes columns, but documentation on all structures and procedures, design notes etc.
- Entity Relationship Diagrams: information to construct the diagrams, as many as desired for a range of audience, with different levels of detail, with overlapping information .
- Data Population SQL scripts: used to prepare instances - to populate new tables, new rows or new columns with data; to correct existing data; or to update static-content tables (also known as configuration, system, code, or look-up tables).

- Any other properties that the end-user may require (the document must be extensible).

From this metadata document would flow the following processes.

Design

- Design proposals can be constructed and reviewed, independent of the development environment.

- Ad-hoc queries for design consistency, where-used, impact analysis etc. are achieved by querying the metadata. This method can be used to find for example "all tables that have a column named xyz" (where-used) or "all tables that have a primary key that is not referenced by an integrity check" (design consistency).

- Customized, extensible reports on any facet of the database. Although DBMS's generally provide access to report on their metadata, this method allows complete reporting by including other information not part of the DBMS metadata. For examples Tables could be categorized by usage, or grouped into subsystems etc.

Development

- Program code for an n-tier application data-access layer can be automatically generated from the metadata. This process is made simpler because the metadata could contain extended properties required for the process.

- Information required for the User Interface (UI) layer could be automatically extracted from the metadata: form labels, help text, validation rules etc.

Administration

- By comparing two versions of the metadata, script required to effect the changes could be automatically generated. Huge savings in DBA time would be made, no change would be overlooked.

- An audit report could generated to show what changes were made between versions. The metadata document could also be stored and tracked with standard change control repositories used for application code.

- Depending on the capabilities of the relevant DBMS, parts of the metadata could be generated from an existing database catalog. This method could be used to create (reverse engineer) a starting-point metadata, or for verification against a master metadata, to ensure the database instance does not deviate from design.

Team Communication

- The metadata can be transmitted and collaboratively worked on, discussed and approved amongst a distributed development team, as a single complete document.

- The metadata can be shared for review by an expert on normalization, consistency, naming conventions etc.

- All team members would have accessible information as to the purpose and content of all parts of the database, with the ability to create personalized reports and views of the metadata to suit their needs.

Reduced Risk

- Deviations between design, documentation and implementation are reduced by having all information in one document, combined with an automated deployment of changes.
- Improved communication between team members reduces misunderstandings and improves efficiency.
- Standards for the correct manipulation of database objects when upgrading can be enforced. For example, tables where a column is to be deleted are always backed up to retain old data.
- Technical skill shortages can be alleviated by automatic script generation to process complex or unusual changes, and to implement agreed standards and best practices.

Building a Central Metadata Environment

Choice of Technology

An obvious choice for storing the database metadata would be an XML document. XML provides the structure and the flexibility to store all the metadata and allow for customized extensions required by the enterprise.

XPATH provides the ability to query the metadata for analysis (where used, impact analysis) and application user interface support (retrieving input validation rules, labels for forms, help text etc) .

XSLT can transform the metadata for reporting, SQL script generation and application code generation.

XSD is used to define the contents of the metadata, including custom properties added by the enterprise. It could be used by tools that edit the document.

At BrandRock, we built the Xeus™ Database Studio on the XML platform.

Centralizing the Metadata

Depending on the DBMS used, reverse engineering the database structure is possible either programmatically (using an API) or via DBMS reports or script. At BrandRock we have been able to reverse engineer Microsoft SQL Server 7, 2000 and 2005 databases into XML metadata successfully using Microsoft's API.

Merging design and data dictionary data into the metadata is probably the harder task, depending entirely how it is currently being stored, and how accessible it is.

Undoubtedly, this is where short-term effort needs to be invested to achieve the benefits of a centralized metadata.

Tools & Utilities

Standard XML editing tools could be used to edit the metadata, but would probably not be able to render Entity Relationship Diagrams. If the ERD's are stored in VRML, a browser could be used to render them. Alternatively a custom tool would have to be built to render the ERD from XML.

All reports, code and script generation can be build with XSLT transformations.

Generating SQL upgrade script is achieved by comparing two versions of the metadata, detecting differences and generating the SQL code required. This is somewhat complicated by having to ensure cross-dependent changes are done in the correct sequence (e.g. new column is added to the table before the index in which it is included).

Conclusion

BrandRock's Xeus™ Database Studio has a complete solution for the team to design, develop and deploy databases using an XML metadata document. Visit www.brandrock.co.za for more information and to download a free trial version.

Contacting BrandRock Software

Web site

http://www.BrandRock.co.za

Telephone

+27 73 767 1717 (mobile cell)

Postal address

PO Box 541, Betty's Bay, Overstrand, South Africa, 7141

Electronic mail

info@brandrock.co.za