The Data Organization

Database Development Life Cycle

By Rainer Schoenrank
Data Warehouse Consultant

December 2019
Biography

Rainer Schoenrank is the senior data warehouse consultant for The Data Organization. He has degrees in physics from the University of Victoria and computer science from the University of Victoria and California State University Hayward. He has built data warehouses for clients such as Pacific Bell, Genentech, GE Leasing, SGI, PPFA, Brobeck, BofA, Clorox, Leapfrog and Intuitive Surgical. He can be reached at rschoenrank@computer.org.
# Table of Contents

1. **INTRODUCTION** ......................................................................................................................... 4  
2. **AUTOMATION METHODOLOGIES** ...................................................................................... 5  
   2.1 Software Development Life Cycle ......................................................................................... 7  
   2.2 Database Development Life Cycle ....................................................................................... 8  
3. **THE DATABASE DEVELOPMENT LIFE CYCLE** .................................................................... 9  
   3.1 Scope – Business Data Vocabulary .................................................................................... 9  
   3.2 Enterprise – Semantic Data Model .................................................................................. 10  
   3.3 System – Logical Data Model ............................................................................................ 10  
   3.4 Technology – Database Design ......................................................................................... 10  
   3.5 Component – DDL Scripts .................................................................................................. 10  
4. **THE INTERFACE BETWEEN THE DDLC AND THE SDLC** ............................................. 11  
   4.1 Scope ................................................................................................................................. 11  
   4.2 Enterprise – Bus Matrix ...................................................................................................... 12  
   4.3 System – Business Object to Logical Table Mapping .......................................................... 12  
   4.4 Technology – Database Interface Specification ............................................................... 12  
   4.5 Component – Interface Stored Procedure Files ............................................................... 13  
5. **OBSERVATIONS** ....................................................................................................................... 14  
6. **REFERENCES** .......................................................................................................................... 15
1. INTRODUCTION

When attempting to automate an enterprise’s processing and record keeping, the Enterprise Architecture (TOGAF) identifies two components that need to be automated, the Application Architecture and the Data Architecture. The Application Architecture identifies the processes that will be automated (applications) and the relationships between these applications. The Data Architecture identifies the data structures that will store the data collected by the applications and the relationships between these data.

For the purpose of implementing an enterprise automation, the enterprise architecture ends up being a list of processes (application architecture) and a list of data (data architecture). To achieve the goal of automation, we need to have techniques to convert these lists into executable programs and functioning data storage locations.
2. AUTOMATION METHODOLOGIES

A guide for the automation is in the Function and Data columns of the Zachman Enterprise Framework. Much criticism of the Zachman Framework is centered on the fact that nobody actually implements the entire framework and the framework provides no guidance for automating business processes. This is not true. The framework has columns for the data architecture and the application architecture which describe the deliverables and the dependencies of creating the automation required by the Enterprise Architecture.

The Function (application architecture) and Data (data architecture) columns of the Zachman Framework are shown in Figure 1.

<table>
<thead>
<tr>
<th>scope</th>
<th>DATA</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>Business Data Vocabulary</td>
<td>List of Business Processes</td>
</tr>
<tr>
<td>enterprise</td>
<td>Semantic / Conceptual Data Model</td>
<td>Business Process Model</td>
</tr>
<tr>
<td>system</td>
<td>Logical Data Model</td>
<td>Application Architecture</td>
</tr>
<tr>
<td>component</td>
<td>Data Definition Language (DDL) files</td>
<td>Source Code Program files</td>
</tr>
</tbody>
</table>

Figure 1 – Enterprise Architecture Automation Development
The design and specification of the column deliverables is done during the scope, enterprise and system tasks. During these tasks any change in the understanding of the business process and data affects the entire model of the business. Once the design and specification are complete, the technology and components level are the tasks required for the implementation of the deliverables.

The task of the scope level is to document the list of items in the column that are important at the enterprise level, such as the sales process for function and the customer for data.

At the enterprise level, the task is to document the organization and relationship between the list of the scope items and show how they are related to each other creating a model of what the automation is expected to do.

At the system level, we document the details of the enterprise level model thus completing the logical design specification.

At the technology level, we begin the implementation phase by taking into account the computational environment being used by the enterprise and extending the design specification with the requirements of the computational environment.

At the component level, we actually produce programs and database that will automate the enterprise.

The process done for the function column is commonly known as the Software Development Life Cycle. In parallel to function column is the data column which outlines the deliverables of the Database Development Life Cycle although the column has never previously been described as an integrated process.

We can see two dependencies between the two columns. In the implementation phase of the SDLC:

- The database specification must be complete before the functional specification can begin
- The database must be implemented before the source code functions can be programmed.

There are more interactions between the two methodologies that will be discussed in Chapter 4.
2.1 Software Development Life Cycle

Progressing through the rows of Figure 1 from top to bottom, in the Function column, one can trace-out the Systems Development Life Cycle (SDLC) which is a de facto standard across the Business Automation Industry. The SDLC has been used since the early 1970’s (Royce) to implement business processes and create executable code.

The SDLC moves from a list of named business processes in the scope level through decomposition and detailed description to the list of deliverables. Each level is a more detailed description of the business process until the description is in a programming language in the component level. The final deliverables are the leaves of a tree describing the business processes at various levels of detail. The SDLC has no way of determining what is a complete set of branches at each level, so there is no limit on the scope of the project.

The SDLC specifies what you need to know and the dependencies between the deliverables, not how to achieve the final deliverables. The project management techniques, not the SDLC, determine the processes, sequence, timing and resources needed to achieve the project goals.
2.2 Database Development Life Cycle

In the database development life cycle, each level is a separate task with its own inputs and deliverables that are the input to the next level. This process is shown in the DDLC process flow diagram.

The development of a database development life cycle has been overshadowed by arguments about
- Codd’s relational data model (theory) which contains no relation update operations
- Database analysis techniques (conceptual modeling) Simsion, Data Modeling, Description or Design,
- Database design techniques (data modeling) Simsion,
- the structure of the database given that the purpose of the database was not explicitly specified. (database type)
- the data definition and query language (database tools) (SQL, NoSQL, XML, etc.)
- the best physical implementation of the Relational Data Model (ORACLE, SQL Server, Netezza, NoSQL, Azure, etc.) without identifying the purpose of the database.
3. THE DATABASE DEVELOPMENT LIFE CYCLE

The terminology around the database development life cycle is confusing. When describing data structures, the different levels use different names to describe similar concepts at a different level of abstraction and from different points of view.

Each level in the DDLC uses the same concept label but in each level, it has a new meaning, e.g., table can mean:

- relation (scope),
- entity (enterprise),
- logical table (system),
- physical table (technology)

3.1 Scope – Business Data Vocabulary

The list of the business data vocabulary (the DDLC input) has always been ambiguous with multiple meanings for each business data term. The enterprise has a problem with this ambiguity because in the database (the DDLC deliverable), each table and column have a single meaning that is fixed across all the enterprise applications.

The key problem and accomplishment of the DDLC are the disambiguation of the business data vocabulary, because:

- There are many words used to identify a single concept.
- A single word has many meanings depending on where in the business it is used.
- Automating the business functions requires that a concept have a single label and meaning because computers do not handle ambiguity.

The level deliverable is the dictionary that holds the business data vocabulary, the definitions of the business data terms and the thesaurus of the relationships between the business data terms.
3.2 Enterprise – Semantic Data Model

The level deliverable is the document that holds the conceptual data model. The project stakeholders will have agreed to the conceptual entity labels and definitions. The business data dictionary will hold the relationships between the conceptual entities and the business data terms.

3.3 System – Logical Data Model

The level deliverable is a project within a CASE tool such as Visible Analyst. The set of logical tables comprise the Enterprise Logical Data Model (ELDM). The logical table definitions will have been done using logical attributes defined by logical data types and the report generated by Visible Analyst will look like the data dictionary.

3.4 Technology – Database Design

The level deliverable is a physical database project within a CASE tool such as Visible Analyst. The set of physical tables comprise the Enterprise Physical Data Model. The physical table definitions will have been done using columns and the physical data types associated with the chosen target DBMS.

3.5 Component – DDL Scripts

The level deliverables are generated from the physical database project from a CASE tool such as Visible Analyst. The tool has options for making the output scripts complete and ready to use.
4. THE INTERFACE BETWEEN THE DDLC AND THE SDLC

When automating business processes, a key requirement is how the business functions interface with the database. Figure 1 will be enhanced to show and document the interface column as seen in the diagram below.

<table>
<thead>
<tr>
<th>scope</th>
<th>DATA</th>
<th>INTERFACE</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Business Data Vocabulary</td>
<td></td>
<td>List of Business Processes</td>
</tr>
<tr>
<td>enterprise</td>
<td>Semantic / Conceptual Data Model</td>
<td><strong>Bus Matrix</strong></td>
<td>Business Process Model</td>
</tr>
<tr>
<td>system</td>
<td>Logical Data Model</td>
<td>Business Object to Logical Tables Mapping</td>
<td>Application Architecture</td>
</tr>
<tr>
<td>technology</td>
<td>Database Design / Database Specification</td>
<td><strong>Database Interface Specification</strong></td>
<td>Application Design / Functional Specification</td>
</tr>
<tr>
<td>component</td>
<td>Data Definition (DDL) files</td>
<td>Interface Stored Procedure Files</td>
<td>Source Code Program files</td>
</tr>
</tbody>
</table>

**Figure 2 – Architectural Interface**

At each level of the methodologies there is a concept that connects the DDLC to the SDLC.

4.1 Scope

There is no interface between the business data vocabulary and the list of business processes at this level.
4.2 Enterprise – Bus Matrix

The relationships between the conceptual data model and the business processes are shown in the bus matrix diagram.

4.3 System – Business Object to Logical Table Mapping

There is a relationship between every business data object used in the OLTP application function and the logical tables in the ELDM. The business data object is a view of the tables in the ELDM. The business data object view is the specification for the business data object stored procedure interface for the DBMS database.

4.4 Technology – Database Interface Specification

There is a choice of where in the OLTP application program the database interface is located. The recommended interface is stored procedures located in the DBMS database because this is less effort than creating the same interface in each of the business functions required for the application.
4.5 Component – Interface Stored Procedure Files

There need to be many stored procedure interface files for each table implemented in the DBMS database. Each table needs the following stored procedures as a minimum:

- Insert (create) a new table row
- Select (read) an existing table row
- Update (change) an existing table row
- Delete an existing table row
- Undelete a previously deleted table row
- Read a table cursor (read a set of table rows). This can be many stored procedures depending on the number of rows in the cursor, the sort order of the cursor, and the column on which the retrieval is done.
- Each of the business data objects identified in section 4.3 will have its own set of interface stored procedures.
5. OBSERVATIONS

At each level of Figure 2, there are interesting observations to be made:

**Scope** – The components of automating an enterprise:
- The use of natural language to label the business data and functions,
- the organization of the business functions (organization chart)
- the organization of the business data structures into an ELDM

make each enterprise uniquely different from every other enterprise. There is too much choice in deciding the labels and type of the data structures to enable multiple enterprises to use the same ELDM.

**Enterprise** – as shown in the bus matrix, every automated business function uses a database that is a subset of the ELDM.

**System** – the ELDM represents a graph of the data structures and every automated business function (OLTP application) can be described as a path or view in the ELDM. The business function can be implemented as a sequence of table operations.

The number of views in the ELDM is the factorial of the number of logical attributes. For an ELDM with 120 logical attributes (12 tables with 10 attributes each) the number of business functions is over 1.7 trillion. So how do you decide which functions need to be implemented?

The minimum estimate for the automation project is the length of time it takes to create the five basic database operations on each of the tables in the ELDM.

**Technology** – there is sufficient information in the physical database specification that the database table interface stored procedures can be generated.

**Component** - the user interface programs for the database table record maintenance can be generated from the physical database. This was done by Wall Data, Inc., in 1996 in its product Salsa for the Desktop.
6. REFERENCES


Simsion, Graeme C., Data Modeling Theory and Practice, Technics Publications, LLC, Bradley Beach, New Jersey, 2007
