DitaExchange

WHITEPAPER

Structuring Content for the Future:

How to Bring the Database Revolution to Content Management

The database revolution that began in the 1960s changed our relationship to data, making it reusable for different target audiences, contexts and situations. It also made it ready for flexible delivery to different apps and across many different platforms and media. This change had a profound impact on almost all organizations and companies around the globe and on the way data was used by the people working there. It led to huge productivity gains, cost savings and increased the overall wealth of the world.

Why did this not happen to content? How can we bring this revolution to content and win the same kind of enormous improvements that the database revolution brought us? And another, closely related question: How can we get our content ready for bot technologies, AI, ML, deep learning, and robotic process automation?

Thanks to the database revolution, data is ready for these new technologies. So, what are we waiting for? The proprietary Information Mapping method, and later the Open DITA standard, were significant stepping stones on the road to re-use and delivery across a variety of platforms. And now the Information 4.0 Consortium, founded in 2017, defines content in six different dimensions: molecular, dynamic, offered, ubiquitous, spontaneous and profiled.

How can we get our content ready for bot technologies, AI, ML, deep learning, and robotic process automation?

Let us start out with an overview of the database revolution and then turn to answering this question. The key to bringing the same revolution to content that the database revolution brought to data is Structured Content Management (SCM). It provides the necessary foundation to get content ready for the future of technology.

But first we need to define some of the key terms used.

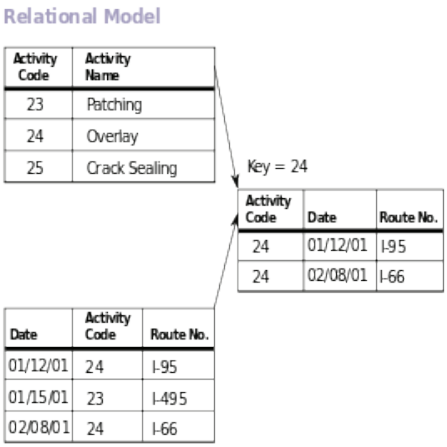
Definitions

Data
For this whitepaper, we will use a variation of the Merriam-Webster definition for data: "information in digital (numerical, character-based, and binary) form that can be transmitted or processed."[1]

Content
Let us use the following working definition for content: Content is text that may contain other media objects like images, audio or videos and potentially rich formatting and/or sematic tagging. A content object can have one or more metadata fields attached to it.

Relational database
A relational database is a digital database whose organization is based on the relational model of data, as proposed by E. F. Codd in 1970. The various software systems used to maintain relational databases are known as a Relational Database Management System (RDBMS). Virtually all relational database systems use Structured Query Language (SQL) as the language for querying and maintaining the database.[2]

Figure 1. The Relational Database Model

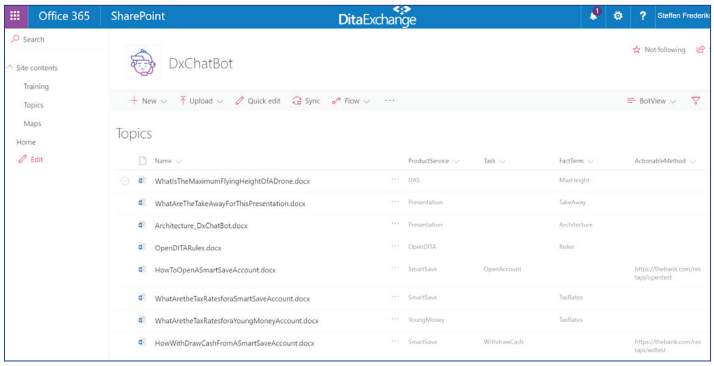


Data is logically organized in database tables (relations) where each row is a record, with multiple columns/fields/attributes. To be well-formed, the fields must be atomic – meaning they cannot in a meaningful way be subdivided into multiple fields.

With relational algebra (joining and splitting tables) and a query language like SQL, you can quickly get granular data or define lengthy reports, input forms or views.

Structured Content Management
Structured Content Management (SCM) solutions enable organizations to store, organize and construct their content so that content topics or molecules are easily re-usable, machine readable components. They can be used to build documents and reports or to feed mobile apps, portals, chatbots or other applications. The topics are normally stored in a repository that resembles a relational database. Content types define the "relations" – similar to database tables – and each topic/molecule is provided with rich metadata.

Figure 2. Repository View of Topics with Metadata



With relational databases, a typical input form often links to multiple data tables. The user filling in data does not need to worry about this.

This is the same for structured content authoring. The authoring "view" – or template – can be a compound document, combining multiple topics and data from other sources, and the author does not have to know or care about this.

[1] *"Data." Merriam-Webster. Accessed August 16, 2017.*
[2] *Relational database." Wikipedia. Accessed August 16, 2017.*

What was the Database Revolution?

In terms of the basic science, the database revolution started in the 1960s and 70s with the first notions of DBMS solutions.[3] But in terms of having a major, world-wide impact on how data was used in organizations, we should look at the period from 1980 to 2000, where relational databases and SQL really dominated.[4]

But for the following example, let us look at the situation in a typical major manufacturing enterprise in 1960 ("Before") and again in 2000 ("After") and highlight the changes in the way customer data was captured, maintained and used.

Data Activity Before and After the Database Revolution

Creation

Before

Customer data was captured and filed independently in many places, for example in Marketing, Sales, Shipping, Invoicing, and in A/R. A new customer could easily appear "on file" in five different places and people in one place did not necessarily know how other people solved this filing need. Each department would register a certain set of data about a customer, but the "datasets" would normally be different from place to place.

After

Customer data can be captured in many different departments, but they are only captured once. As in the before situation, some departments might need to add extra data elements about customers, but each individual data element is only captured once, in one place. The fields or data elements are defined globally across the company, so a "Surname" means the same to everyone.

Storage

Before

In most companies, data was stored locally in the department, listed in a typed document or on index cards, but in some advanced companies, data could be stored electronically on magnetic tape or similar devices. Data redundancy was a sure thing!

After

Logically, data is stored once, in one place. No data redundancies.

The logical structure (tables/relations, fields) is defined and maintained by a data architect, together with the database administrator. "Normalization" of the structure ensures no redundancy, no inconsistencies and that the structure is flexible enough to support even new, unknown use cases.

Physically, data can be stored in many places to ensure recovery or even to speed-up processing. But this is handled automatically by the DBMS/RDBMS, so it should not concern users at all.

Maintenance

Before

There was initial confusion when one department got new data, changed its files and then the change would trickle down over time to other departments. 3-4 months with inconsistent and incorrect customer data was normal.

After

Thanks to the "normalized" data architecture, maintenance is simple. Data changes are captured once and then immediately available for everyone.

[3] Carl Olofson, "The database revolution." IBM Data Magazine (February 7, 2011).
[4] E. F. Codd, "A relational model of data for large shared data banks." Communications of the ACM 377 (1970).

Usage

Before

From the storage example above, it should be apparent that structure and usage was one and the same thing. The typed list or the index card was the "output format" or the "report." If you wanted another look and feel, you would have to get the typewriter ready and start all over again.

Even in the best cases, where data was stored on magnetic tape, the storage format defined the one possible output type. If you wanted a report sorting customers by state instead of by name that would often also mean starting from scratch, with a new tape and new software.

Very often, the use cases of the various departments were different and the data stored only supported the local use case. Overall, there is no, or very minimal, reuse of data.

After

Based on the logical database architecture, you can define as many database views, input forms, and reports as needed, if the data is there.

In this way, you can support:

- multiple use cases
- different target audiences
- different applications
- different media
- different platforms

You can also support emerging technologies:

- bot-based automation
- chatbots
- AI
- ML
- deep learning
- big data

This enables you to get data to the right person, at the right place, in the right format, on the right platform – just in time.

Overall, there is a massive reuse of data.

Management

Before

There was no centralized management of data at all. Data management was entirely local, looking for one use case only.

After

Together with the database administrator, the database architect is a key person in the management of the corporate database.

Evaluation

When you consider the evidence above, this was such an obvious improvement – with countless benefits.

But even this came with a change management challenge. Users had to work in new ways, dictated by a central administrative instance, and at least partially defined by the needs of other parts of the organization. If you had grown fond of your own index card file – where you could add your own notes and secret signs – then this was a challenge.

But for the organizations, it was a no-brainer. The benefits were huge and they outweighed the change management issues.

The Content Reuse Revolution: What is It Like?

We have already defined the meaning of relational databases and SCM, but here we present the analogy between the two.

Table 1. Relational Databases and SCM

Relational Databases	SCM
Table/Relation (defines the fields of a record)	Content type (defines the type of information in the content module/chunk)
Field/Column	Metadata field
Record (table row)	Topic/content molecule
Atomicity of fields, a field cannot in a meaningful way be subdivided	Topic/molecule is the smallest possible unit of content that makes sense standalone, when seen together with the metadata
Database	Repository
View	View
Report	Compound document
Input form (possibly combining multiple tables)	Template (possibly combining multiple topics)
Database administrator	Repository manager
Data architect	Content or information architect

As you can see, the analogy is quite robust – and this becomes even more evident in the following.

Content Before and After Structured Content Management

Creation

Before

Monolithic documents – sometimes with hundreds of pages – containing partially identical content were written or rewritten independently in many places in the organization. In some cases, cut-and-paste played an important role in the writing process, both from other documents and from other data sources.

A certain piece of content (a topic or molecule) could easily appear over time in five different documents and on websites, portal pages and mobile apps. People in one place did not necessarily know how other people used this content.

After

Content topics/molecules can be captured in many different departments, but they are only captured once.

As in the before-situation, some departments might need to add extra topics, but each individual topic is captured once, in one place.

The content types and metadata fields are defined globally across the company; so, for example, a TASK topic in an SOP is easily recognizable.

Storage

Before

The monolithic documents are stored on disc, locally, or on a file share. In advanced cases, maybe they are placed on a document management system, and maybe even with some metadata on the mega-document level.

Versions re-written in HTML might sit on web servers somewhere – or live in mobile apps.

Content redundancy was a sure thing and inconsistencies (even before considering maintenance) were very likely.

After

Logically, content topics/molecules with rich metadata are stored once, in one place. No content redundancies.

The logical structure (content types/metadata fields) is defined and maintained by a content architect, together with the repository administrator. "Normalization" of the repository ensures no redundancy, no inconsistencies and that the structure is flexible enough to support even new, unknown use cases.

Physically, topics could be stored in many places to ensure recovery or even to speed-up processing. But this is handled automatically by the repository management system, so it should not concern users at all.

Maintenance

Before

Users would change a piece of content that was included (and maybe by cut-n-paste) in multiple documents. You can imagine the initial confusion when one department got the new content, changed its documents and then the change would over time trickle down to other departments. 3-4 months with inconsistent and incorrect content was normal.

After

Thanks to the "normalized" content architecture, maintenance is simple. Content/topic changes are captured once and then immediately available for everyone.

Content references (the smart replacement for cut-and-paste), even to external sources, are automatically refreshed.

Usage

Before

From the storage section above, it should be apparent that structure, i.e. a monolithic document in PDF or Word format, and usage was one and the same thing. Each document was built to support exactly one use case and that was it.

Very often, the use cases of the various departments were different and the document stored only supported the local use case.

After

Based on the logical repository content architecture, you can define as many documents, templates (with pre-configured reused content and data), web pages and reports as needed, if the topics are there. In this way, you can support:

- multiple use cases
- different target audiences
- different applications
- different media
- different platforms

You can also support new technologies:

- bot-based automation
- chatbots
- AI
- ML
- deep learning
- big data

This will get the right content molecule to the right person, at the right place, in the right format, on the right platform – just in time.

Management

Before

As you might have guessed by now, there was no centralized management of content at all. None, or at least very close to none. Content management was entirely local, looking out for one use case only.

After

Together with the repository administrator, the content architect is a key person in the management of the corporate content repository.

Evaluation

Did you notice that you could reuse 80% of the sections from the description of the database revolution before and after? Structured content management can bring the same revolution to content that the database revolution brought to data.

But, why would it not?

The Change Management Challenge

Just as with the databases, SCM does bring with it change management challenges. The common perception of SCM includes:

- SCM is often linked to XML and special XML authoring tools
- SCM involves rewriting/redoing the entire bulk of legacy documents before anyone can get any results from it
- SCM is often linked to authors having to write "molecules" out of context
- SCM is often linked to adding a lot of extra work and complexity on the writer, without making it easier for her to solve the primary mission of getting a document done

Let us have a look at these, one by one.

SCM and XML

Handwritten scrolls would not be ideal for SCM. And XML, like JSON or HTML, is a great tool for creating machine-readable content that can contain semantic tagging. But this does not mean that a content creator would need to use an exotic XML editor.

Since 2007, Microsoft Word has in fact been using XML as the default document format (Office Open XML, ISO 29500:2008-2016, ECMA-376, Editions 1-5). When you write something using Word, you are creating XML without even knowing it. The same goes for some of the best XML editors, like oXygen XML Web Author. You do not need to be an XML expert to use it.

It may be true that SCM works best with XML, but you do not need to worry about it. You could use Microsoft Word or FrameMaker or similar tools. With SCM you do not need to worry about XML, just can keep using Microsoft Word.

> Structured content management can bring the same revolution to content that the database revolution brought to data.

SCM and Legacy Documents

Are all our 1.5 million legacy documents lost forever when we move to SCM?

A tempting answer to that question would be: Yes, to the extent they have little or no metadata and to the extent they contain unstructured text. They were lost right after you placed them in the file share.

But a more correct answer would be: No, your legacy content is not lost when you implement SCM. Good SCM solutions will enable you to easily migrate the valuable parts, or even reference the existing legacy content into new topics.

SCM and Writing Out of Context

You might believe that with SCM you would have to write single, stand-alone sentences with no option to see what you are doing in a meaningful context. But this is only true if you have a really bad SCM implementation. A good SCM topic must make sense to the writer, to the reviewer and to the approver. If it does not make sense, it is not a good SCM topic.

And there is an even more compelling response. The conversion from the monolithic document-like templates to the granular content topics or molecules can be handled automatically behind the scene. This would be just the same as when a database user would use an input form without worrying about how many separate tables were involved in the background. If writing into a complete document makes more sense for the writer, then the SCM solution can support it using a compound template for writing. Many topics could be hidden so that the writer does need not to think about them.

SCM and Extra Work

Any change can involve some extra work, but with SCM, the content architect can prepare a template for the writer that automatically fills in up to 50% of the needed content, maybe more. A good SCM implementation will mean less effort to get more done, not the opposite.

Preparing Content for the Future

SCM is not only about writing, reviewing, approving, managing and publishing PDF files. The molecular content stored in a database-like repository is the ideal food for:

- portal sites
- mobile apps
- chatbots
- AI/ML and deep learning applications

And it is also ideal for machine-generated content and for machine-to-machine communication.

The revolution has just started and it is going to be very interesting to follow it over the next few years

References

Codd, E. F. "A relational model of data for large shared data banks." Communications of the ACM 377 (1970).

"Data." Merriam-Webster. Accessed August 16, 2017. https://www.merriam-webster.com/dictionary/data.

Information 4.0 Consortium. 2017. Accessed August 18, 2017. http://information4zero.org/.

Olofson, Carl. "The database revolution." IBM Data Magazine, February 7, 2011.

"Relational database." Wikipedia. Accessed August 16, 2017. https://en.wikipedia.org/wiki/Relational_database.

About the Author

Steffen Frederiksen founded DitaExchange in 2007 and currently serves as the Chief Strategy Officer. In this role, he is responsible for aligning technology development and corporate strategy to enable DitaExchange to deliver sophisticated structured content management solutions that help organizations stay ahead of market needs, while keeping simplicity in mind for end users.

About DitaExchange

DitaExchange simplifies the way organizations create, manage, deliver and re-use important content with Dx4 – our structured authoring and content management solution built to run on the Microsoft SharePoint platform. By helping companies efficiently produce and maintain important information and documents quickly while also following compliance guidelines, employees spend less time duplicating content and repeating approval cycles while also improving overall compliance by improving consistency and quality.

Contact info@ditaexchange.com to learn more