

High-level introduction to Relational Document Management

Project	Semantica DMS
Category	Whitepaper
Status	Final
Version	1.0
Date	1 augustus 2009

Table of Contents

Introduction	3
What's relational document management?	3
Problems with hierarchies	3
Properties for structure	4
Properties for workflow	6
Properties for security	6



Introduction

Are you still using hierarchical databases? Probably not. You are most certainly using relational databases to store and relate information. Why? Because they are flexible and easy to use. So why are you still using hierarchical structures to store documents?

Allow us to introduce you to the next phase in efficiency: relational document management. And Semantica DMS is the first fully property-based system to implement this new way of working!

You are used to store files in folders in a cabinet. Historically the same way of organizing documents was introduced to the computer. This way of structuring was very convenient for computers, but less so for humans. If you ever looked for a document somewhere in a computer folder hierarchy, you know there must be an easier way. There finally is.

What's relational document management?

Relational document management refers to relationships between documents. These relationships are needed to structure large amounts of data. But instead of using physical structures, like hierarchies of folders and files, the relationships are maintained through values of properties. So why is this better?

It's easy to classify documents, e.g. an invoice and an employment contract will have very different properties. Having established the document type, you implicitly decided what properties need to be assigned. If it's an invoice, there should be some way of identifying the supplier. If you have a purchase order, or a contract, there too will be a supplier identification. Wouldn't it be easy to automatically relate these documents on supplier information?

Problems with hierarchies

The figures below show both hierarchal and property based structures. Some systems use the hierarchy as a direct metaphor, others try to obfuscate the hierarchy, but the fact is that most storage structures to date are based on hierarchies to store information.

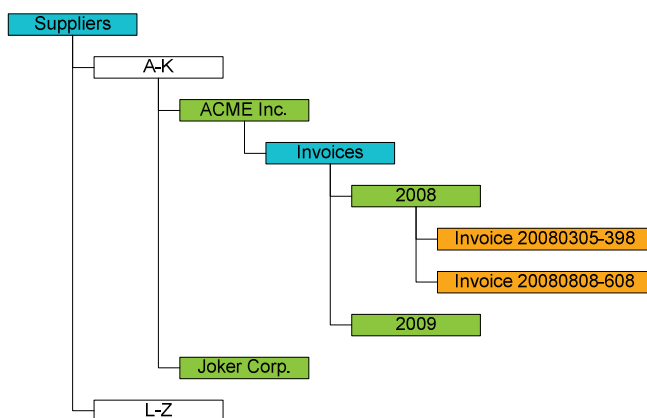


Fig. 1 - Commonly used hierarchical structure

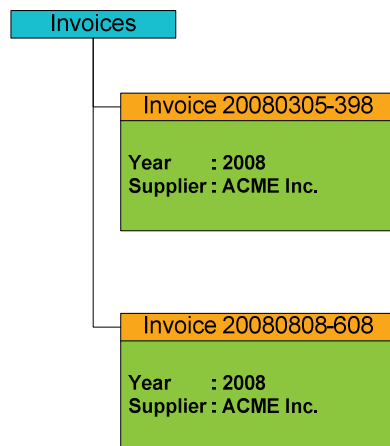


Fig. 2 - Semantica structure based on type and properties

Besides the obvious difference in complexity between hierarchy and property based structures, there are some interesting problems with hierarchies that need our attention.

Whether you use all depicted levels in the hierarchy or less, the problem stays the same; you are bound to the fixed limits of your predefined structure. When things change, it may be very hard, even impossible, to change your document storage structure too. This way you get stuck with an old structure in a new environment. Does this sound good to you?

Another problem is the number of levels humans can comprehend. You may not realize this when things are set up, but humans are usually able to work efficiently in a three level structure, maximum. Most users prefer to use two. But two levels dictate large amounts of data to be sifted through. Also, you would have to choose a far from optimal structure. If you go for 'Supplier-Year' levels, all documents, no matter what sort, are stored per year. Wanting to know how many invoices you had with a value above some amount might pose a challenge. The same problems are present in any other two-level structure that contains complex information, like documents.

Properties for structure

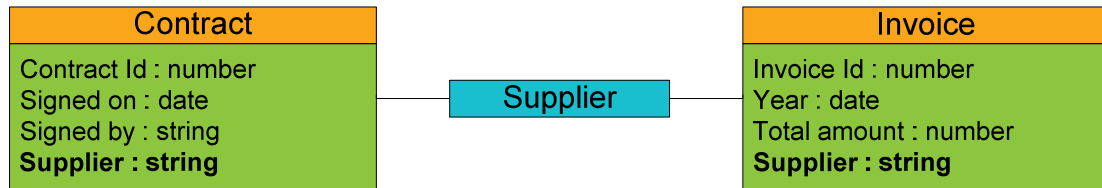
It's commonly known that most of today's databases are based on the relational model to store and retrieve data. This model is based on properties and their values to relate information.

For a long time complex information, like a document, was considered "unstructured". Although this might seem true from a distance, a closer look reveals a lot of structural data is present in documents, especially corporate documents like reports, invoices and contracts. Semantica is the first document management system to use this structure to its full potential.

Properties like "author", "creation date" and "document name" are well known and almost universal to all documents, and as they can be determined by a computer system we call them "system properties". But there are more properties that are of

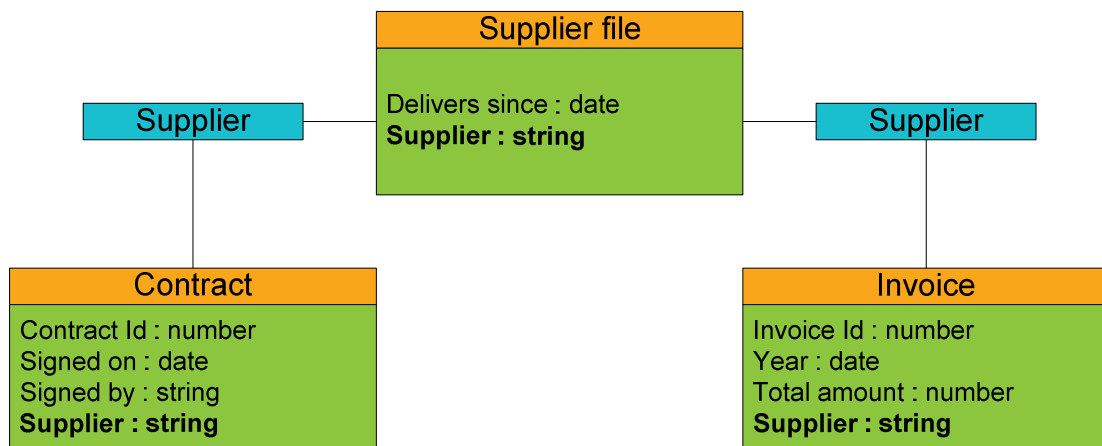
interest; we call them "document properties", and their exact number and meaning depend on the sort of document, or "document type".

To take this to a structural level we use this property information to link documents together. First of all an administrator defines the relationship between two document types by determining the appropriate property types. On an invoice and a contract we will be interested in something like "supplier name" to link the two together. Then, when documents enter the system, the property values can be used to relate information about contracts from supplier 'A' to the invoices from this supplier.



To make more use of this way of relational structure, there is no need for a real-life document type to hold the key. You might introduce the abstract concept of a supplier file to the system.

It's very well possible this document type may only serve a purpose as a container, like the paper file itself is not part of the contents. The "supplier file" document type also would have a property type like "supplier name". Linking this piece of information to invoices and contracts is simple, but reveals the strength of relational document management: flexibility.



Because there is no technical limit to the amount of relationships you can define, and more importantly because the relationship is present in the documents property values, when things change, you simply change the way you relate the documents.

This resembles the way relational databases link information together, hence the name "relational document management".

Properties for workflow

Being fully property-based, the use of properties isn't restricted to linking information; we also put it to work as part of our workflow processes. Traditional systems separate flow and contents where we combine them in a concept called "process property".

With workflow you want some piece of work to be signaled, or assigned, to a worker. The flexible notion of "work lists" provides just that: all the documents that need to be attended to. The work lists show all documents that comply with queries based on properties. For example the work list "Fill in address" may be based on a query like "show all documents of type Contract with Address is empty". This uses document properties to determine the contents of a work list. But you may have a process called "Check address", used by a senior worker, that uses a dedicated process property like "Address checked". This kind of property isn't deducible from the document's contents; it is part of your organization's workflow process.

Properties for security

Last but not least Semantica offers a unique fine-grained security mechanism based on properties. Most systems grant you access to some type of information, like contracts, based on your rights on the structural container. We use our knowledge on properties to expand on this insufficient security concept. Do you want to restrict access to invoices with amounts above \$10,000? That's simple, no need for a separate structural element, just use this definition when administering your system. And when things change? You've probably guessed by now; Semantica incorporates your changes, effortlessly and safely.

Want to get up-to-date on relational document management? Check www.semantica.com and try Semantica before you buy.

Do you have any questions or remarks regarding this subject? We value your input, so please leave a message at the website or contact us at dms@semantica.com.