

# Information/Content Architectures

– John Maxwell, January 2007

## What is Information Architecture?

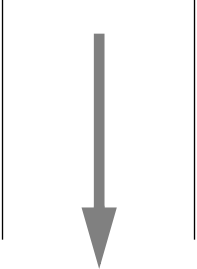
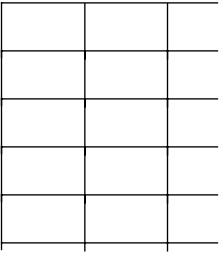
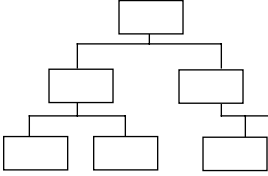
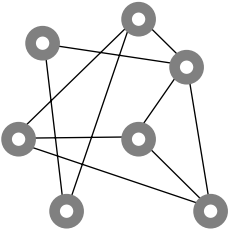
At least two versions:

1. Based on the assumption that *information is a product of the user's experience*, information architecture is seen as the large-scale organization of a publication: what information exists where, in what form, and how does one get to it?
2. Based on the assumption that *information can be represented abstractly*, information architecture is seen as the organization of content: what does the information look like, what shape is it, and what opportunities for access does it afford?

Perhaps there should be two terms: *publication* architecture and *content* architecture, with information architecture encompassing both. These two views of information are *not one and the same, nor is one dependent on the other*.

# The *Shape* of Information

## Four Basic Models:

Linear	Tabular	Hierarchical	Web
			
Examples: scrolls narratives "begats"	Examples: accounting, ledgers phone book catalogues	Examples: org. charts most books grammar	Examples: the Web your friends your mind (?)

## Documents vs. Data

### Data tends to be tabular

"Data processing" generally pertains to the management of tabular data: names, purchases, accounts, taxes, records, and so on.

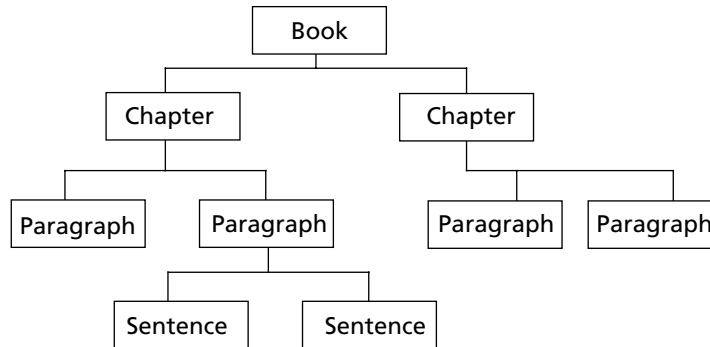
The earliest forms of writing (think cuneiform in clay) were not linear, but tabular: *Nebuchadnezzar has 100 cows; Urduk has 125*; etc. Similarly, the earliest uses of computers were for tabular data: taxes, artillery shell trigonometry tables, etc.

Data tends to be atomistic; you combine it to compose larger structures.

*Data*—etymologically, *the givens*—is actually the plural of *datum*, but nobody ever says datum (except on tide tables).

## Documents tend to be hierarchies

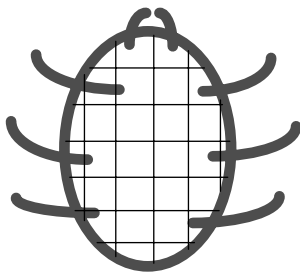
What we commonly think of as “documents” (*and what exactly are those, anyway?*) are at first glance linear, but when *analyzed* and *interpreted*, reveal a hierarchical structure.



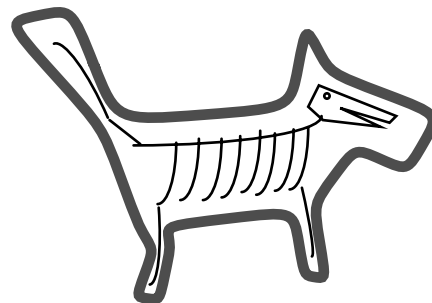
The hierarchical model of a document is not inherent or essential. It is *generated* in the process of design (during authoring) and/or interpretation (during rendering and reading), by a human being, a computer, or both.

Documents are in some sense holistic. Unlike data, the structure can be decomposed from the whole.

## Data vs Documents: an analogy



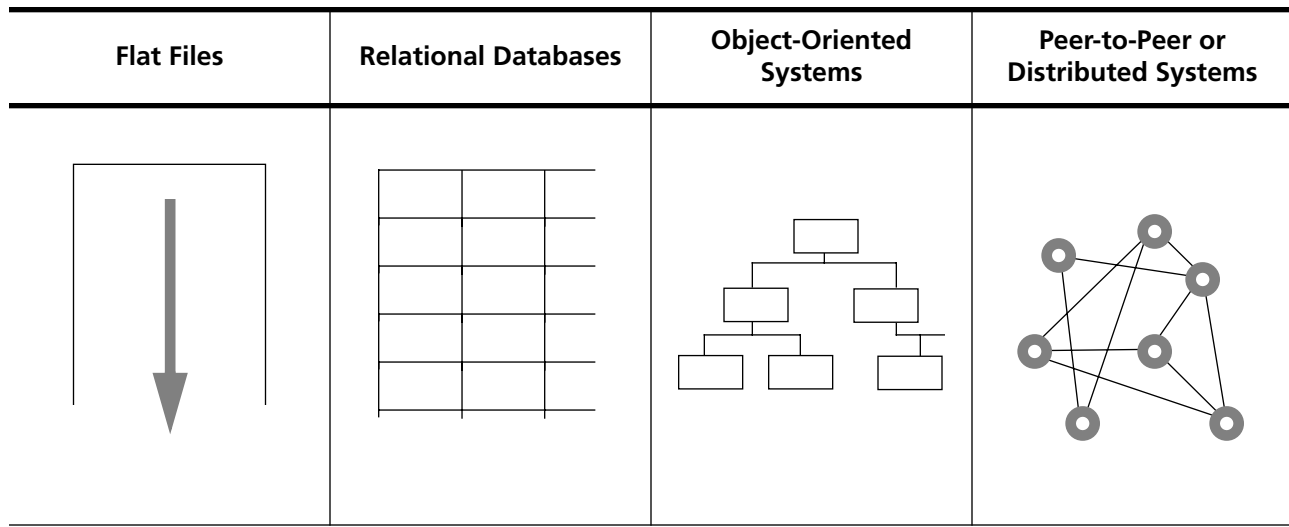
**Data: Exoskeleton**  
Structure outside,  
Squishy bits inside.  
Look into the structure  
to find the meat.



**Document: Endoskeleton**  
Structure inside,  
Squishy bits outside.  
Look into the meat  
to find the structure.

# Content Architecture Models: An Overview

An oversimplification:



## Flat Files

The most simplest, common digital content format: text files, HTML, XML, images, audio, video; files of all kinds. The basic model of abstraction is that 1 file = 1 information object (a page, a diagram, a movie, etc.)

Flat files are essentially *sequential-access*, and are sometimes human-readable (HTML, ASCII, XML)

## Relational Databases (RDBMS)

Relational DBs are increasingly common for online content management. Relational Database Management Systems (RDBMS) such as *Oracle*, *Sybase*, *MySQL*, *MS Access*, *MS SQL Server* use multiple linked (“relational”) *tables*, which contain *records*, which contain *fields*.

SQL—*Structured Query Language*—is a cross-platform standard for talking to a relational DB; searching, asking for reports, adding or manipulating data, etc. Much web-DB programming involves writing “SQL statements” that pull the appropriate information out of the DB.

Relational DBs are multi-dimensional, composed of tables connected to tables. This is a very good way to represent *many-to-many* relationships ('authors' to 'titles' for instance) without redundancy. They are, in effect, *random access*, and are not in themselves human-readable. RDBMSs are a very mature technology, and much of the development work goes into making them as fast as possible. This is the technology behind most search engines and web-based content management systems.

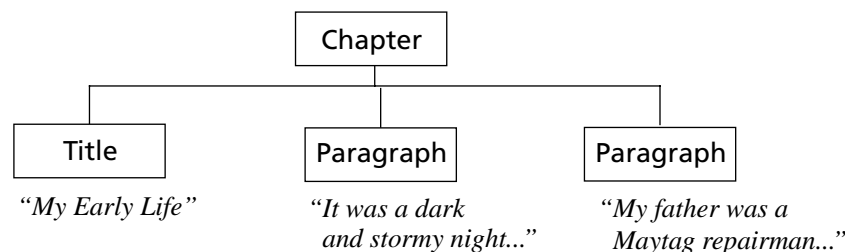
## Object-Oriented (OODBMS, parsed XML)

Object-Oriented systems work at a higher level of abstraction than relational DBs. A common OO model is a hierarchy or tree structure, and we use a "family tree" metaphor for talking about them: *parent, child, siblings*. The depth of objects on the tree is not limited (branches of branches of...), which means OO systems handle arbitrary complexity better than RDBMS.

*Object-orientation*, or the *object paradigm* is an abstract information modelling concept. An *object* combines properties (data) with behaviours (actions or functionality). As such, *object modeling* is a complex science of systems management.

Just as there are RDBMSs, there are OODBMSs, though they are not so common. Notably, XML files are "parsed" into an object system called the *Document Object Model* (DOM). This makes highly flexible OO management available for Web publishing.

A very simple document-object example:



In the above example, there are four objects. The parent object is of type *Chapter*. There are three sibling objects, of types *Title* and *Paragraph*. These objects have content (data) and might have a set of behaviours attached to them. For instance, *Title* objects should be displayed centred and in 18pt Frutiger bold. *Paragraph* objects should be displayed in 12pt italic Garamond, and all but the first in a series should start with an indent. Each object in a sense “knows” what it should do.

## **P2P and Distributed (WWW, Kazaa, BitTorrent, etc.)**

Peer-to-Peer (P2P) and Distributed information models are all the rage online. The WWW partially works this way: one page points to another, and another, and so on. There is no top-down structure (excepting Google). P2P file-sharing models are even more fluid: where any given file is changes depending on who’s connected.

Making sense of P2P information only works when you actually use it. If we were to track a person’s surfing on the web, the resulting collection of navigation paths and pages *would be a tree* (an OO model). However, that particular tree structure only exists *in the moment*, or over the time the person spent going through those pages. Another person surfs a different structure. The actual location of things is an arbitrary web of information, and it changes from minute-to-minute. Access in a P2P system is *associative*, rather than sequential or random.

### ***Nota bene...***

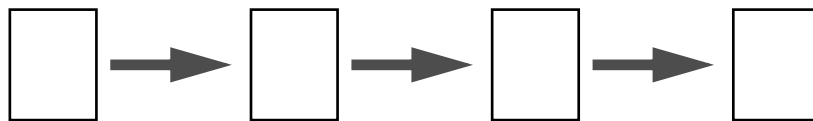
There are few hard boundaries between these models: narratives are sometimes hierarchical; flat files are sometimes made to work like RDBMS; RDBMS sometimes emulate OODBMS; flat XML files become trees of objects when “parsed”, and so on. The models are useful as *abstractions*, or mental scaffolding, when thinking about information architecture—and of course for building technology—but at the bottom level, it’s all just bits.

## How All This Translates to “Publication Architecture” ...

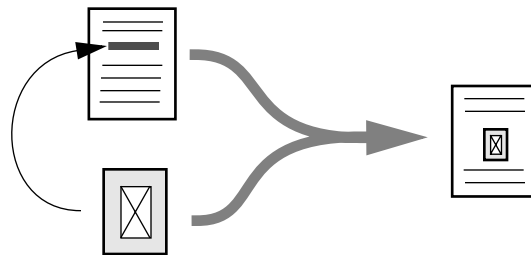
These four basic models are also used in organizing websites—that is, doing *publication architecture*. Here are some examples of how the theory works in practice.

### Static Files

If each file is a page, then this works more or less like a book: you go from page to page to page. Because we’re not bound by physical constraints, this page-turning can be quite complex and even seemingly non-linear. But basically, we’re turning pages. This is the simple, basic WWW model.



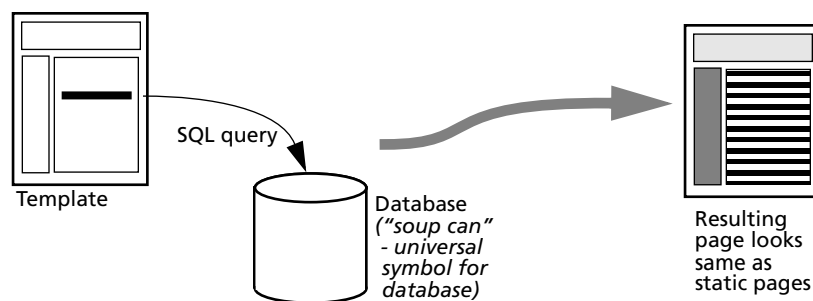
The other common linear model is *inclusion*: I have an HTML file, and I have a graphic file. I include a reference to the graphic in the HTML, and the graphic is inserted, sequentially, on the page.



### RDBMS and SQL Queries

The common model in using databases is to construct standard page *templates*, then fill the template components (over and over again) with either static info (logos, boilerplate text) or dynamic content pulled from

the database via SQL. The webserver software is responsible for assembling these components when a page is requested by a browser.



A SQL example (this is what you pay database programmers \$100/hr for):

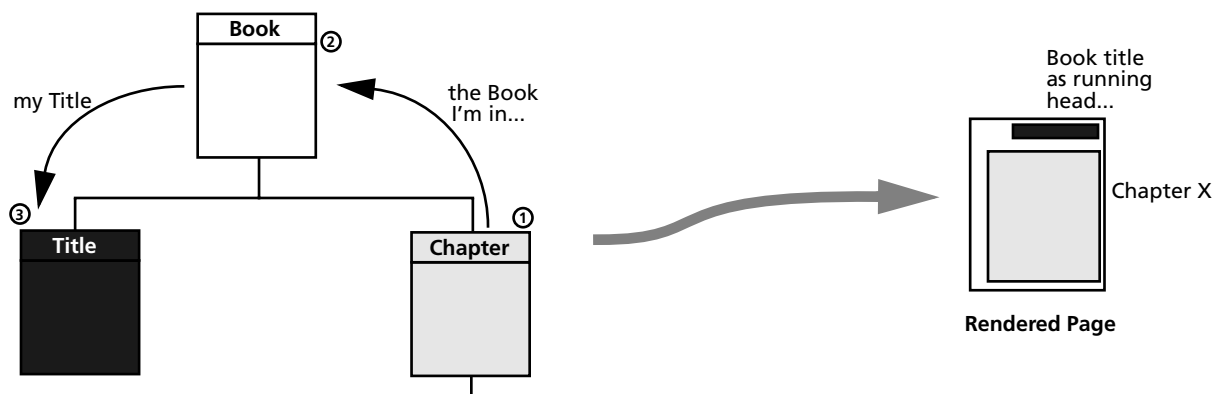
```
SELECT SoupFlavour
FROM CampbellsVarietiesTable
WHERE TYPE = 'Clear' AND (MEAT = 'Chicken' AND NOODLES > 10000);
```

People sometimes put large amounts of textual content (whole pages, even) into the fields in a relational database. Some people think this is an inelegant thing to do.

## OO: Context Driven

In an object-oriented system, any information object occupies a location in a larger context. So a *Chapter* object might be a child of a *Book* object, sibling of a *Diagram* object, parent of a *Paragraph* object. From any point in the tree, you can derive information about the nodes above,

below, and beside: *From this particular Chapter, what is the Title of the Book I am in?*



## “Hypertext” Considered Three Ways:

Hypertext is traditionally defined as “non-linear writing” (after Ted Nelson). But this definition only really speaks to the phenomena of hypertext as experienced by a reader; it doesn’t tell us much about information architecture.

Given our basic architecture models, the idea of hypertext can be considered in at least three different ways.

### Linear/flat file model:

A hypertext link is simply a *pointer* from one page (file) to another.

### RDBMS model:

A hypertext link is a *query* that references another chunk of data in the database.

## OO model:

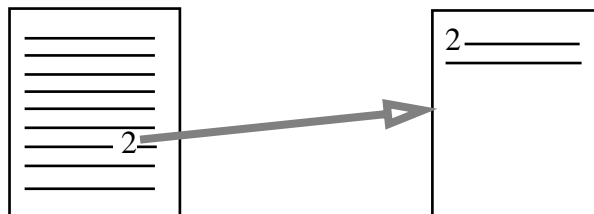
A hypertext link is a *reference* to another node in the tree, with a particular relationship to the starting node.

## An Example: the Lowly Footnote

The footnote is the most common and basic form of hypertext<sup>1</sup>. Here are three different ways of conceptualizing what a footnote is, from an information architectures perspective.

### Flat file model:

The body text we are looking at is *file1*, the footnote context is *file2*. The footnote's reference token (the little number or whatever) is a link that refers to *file2*. The relationship between the two is simply that of two files.



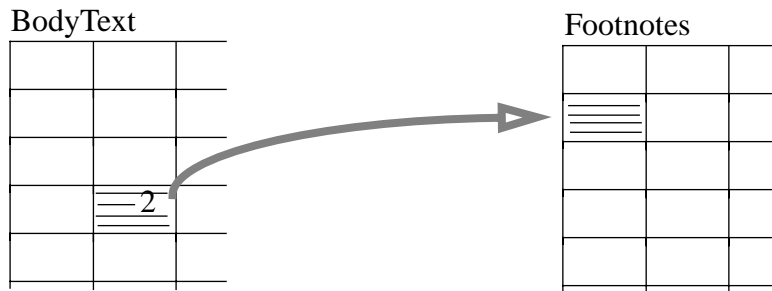
### RDBMS model:

The body text is a field in a particular table (the *BodyText* table, for instance). The footnote is a field in another table (the *Footnotes* table). The

---

1. This hypertext thing is not new.

footnote's reference token is a *query* that requests the contents of a particular field in the Footnotes table, given the footnote number. The relationship between the two is a relation between two tables.



### OO model:

The footnote is a child object of the body text object. The footnote's reference token is a call to display the child object. The footnote number "2" might refer to the second child object. The relationship between the two is one of ownership: the body text "owns" its footnotes.

