SEE ALSO:

bits of information in a physical world

CHRISTIAN SWINEHART

ETHEREAL DATA/CORPOREAL INTERFACE

Of all the changes in thinking ushered in by the digital age, one of the subtlest – and yet most pervasive – is the realization that information is a fundamentally different entity than the medium it is embedded in. Only a century ago it was impossible to send a message of any kind without physically conveying it. Whether by book, letter, or messenger, the content could not reach its destination without a material substrate literally carrying it through space. While a Dualist view of information as a spirit separate from the corpus of its medium might have made for entertaining salon banter, it was not until the arrival of electrical (and later electronic) communication that one could actually 'experience' content divorced from physical form. Needless to say, we are all Cartesians now.

The omnipresence of pure information processors in today's world has led to the aggregation and composition of data in ways that are so impractical as to be impossible if one attempted to construct them out of atoms rather than bits. While certain concepts of computation were achievable using the clockwork mechanisms of the time – Charles Babbage's Difference Engine being a particularly heroic example – data storage and analysis demand a degree of saved state and speed of information access that places them firmly out of reach.

Yet the steam age which preceded our own provides us with many of the fundamental metaphors for dealing with data. And while it requires the existence of the computer – free from the constraints of physicality – to conceive of a spreadsheet or relational database, it is equally impossible to imagine such things being invented absent their Victorian precursors: the ledger and card catalog.

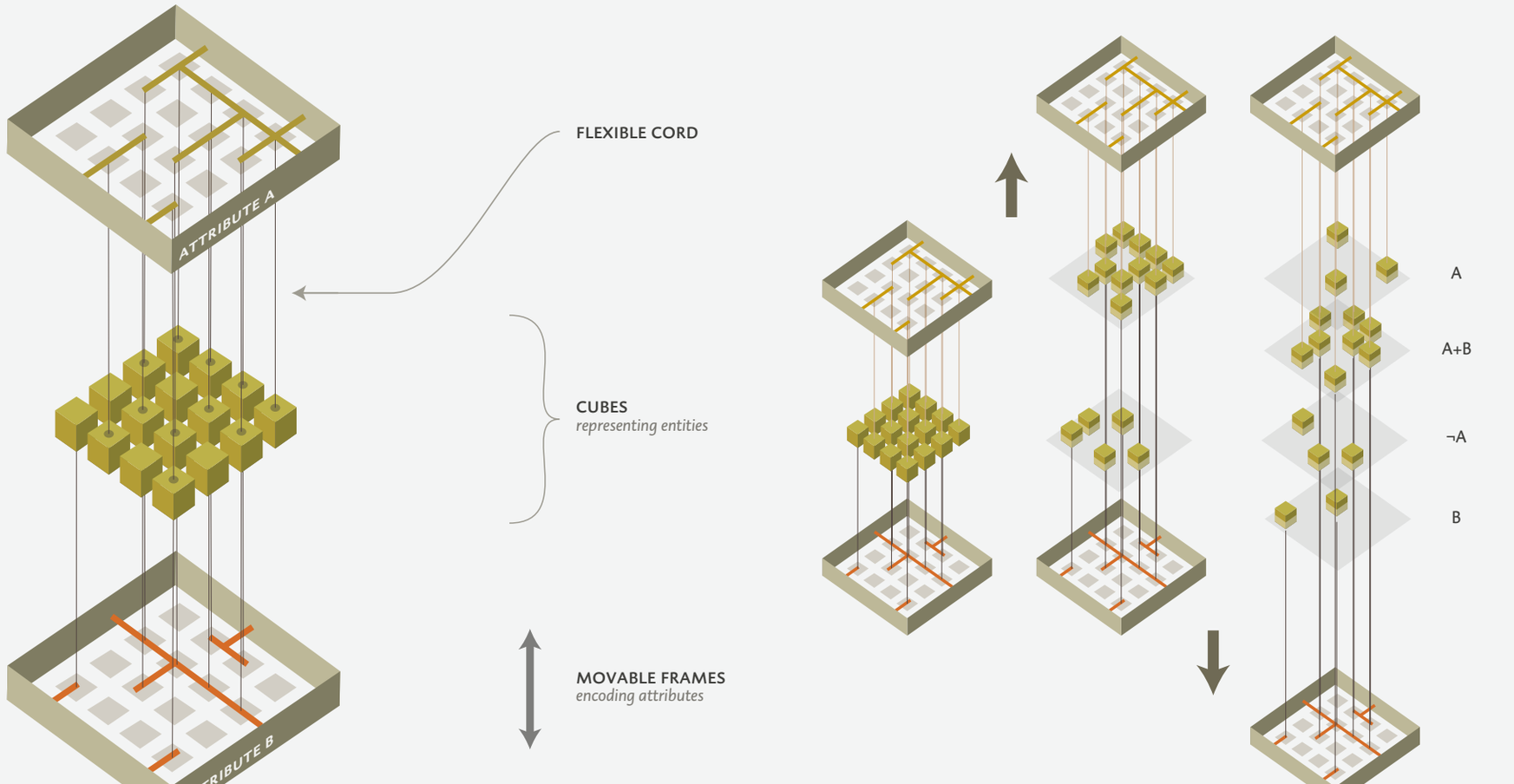
The fundamental unit which characterizes the modern approach to data is the link (or 'cross reference' in steam age nomenclature), either in the explicit form of documents hyperlinking to one another, or through the implicit linkage of a shared attribute among entities in a database. Interestingly, links have never taken physical form – even in atom-based filing systems they exist purely through the potential to follow the advice to 'see also' at the foot of an index card. In their modern form, links can be used to conjure up a new document or new assemblage of entities within the database. But the item itself is still wholly intangible.

The investigations that follow attempt to provide the link with a measure of real-world heft, expressing its presence, composition, and scope in terms of mass, light, and volume.

LITERAL SET THEORY

A database is an assemblage of individual entities which, unlike a list or catalog, has no inherent meaningful ordering. Instead, any order which occurs is transient and emergent, deriving from the qualities of the contained data themselves. For while the entities are not organized, they are each tagged with one or more attributes. Thus there are as many *potential* groupings as there are values for these attributes, as entities with like values are segregated into ad hoc categories.

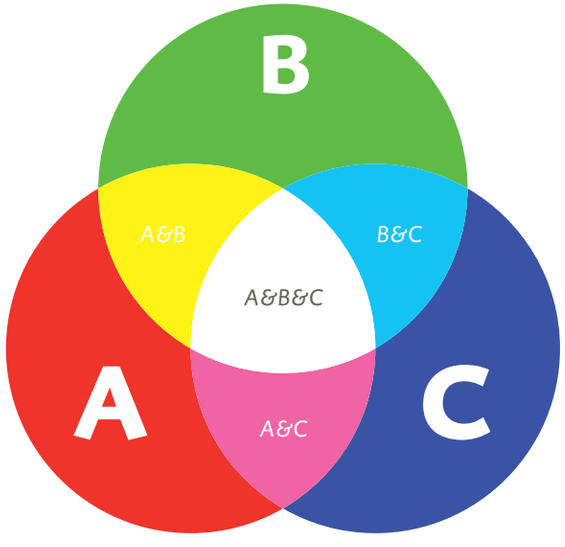
But even if this structure is only apparent through the epiphenomenon of a sorted set of entities, the connections are there even when the database is inactive. In a physical implementation of a database, entities could be represented as cubes selectively tied to one or more frames representing attributes that the entities might possess. By examining the frames, the pattern of connections emerges and by raising or lowering the frame the set of connected entities will rise or fall away from their dissimilar peers. A second attribute can be incorporated by pulling an additional frame in the opposite direction. This causes a subset of both of the groups formed by the first frame to separate, segregating all possible combinations of the two attributes in horizontal layers.



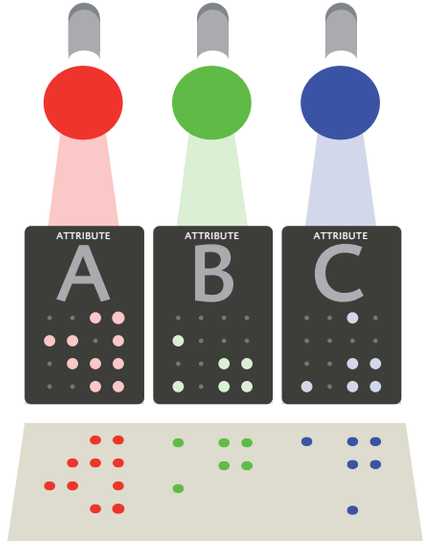
OPTICAL CORRELATION DETECTION

Spatial models of group membership quickly become unworkable as more and more attribute frames are added. An alternative way to visualize the membership of entities in multiple groups is to allow the physics of color to do the heavy lifting. Shining primary colored lights onto a surface results in either a white spot if all three are projected, or a more saturated hue if fewer are present.

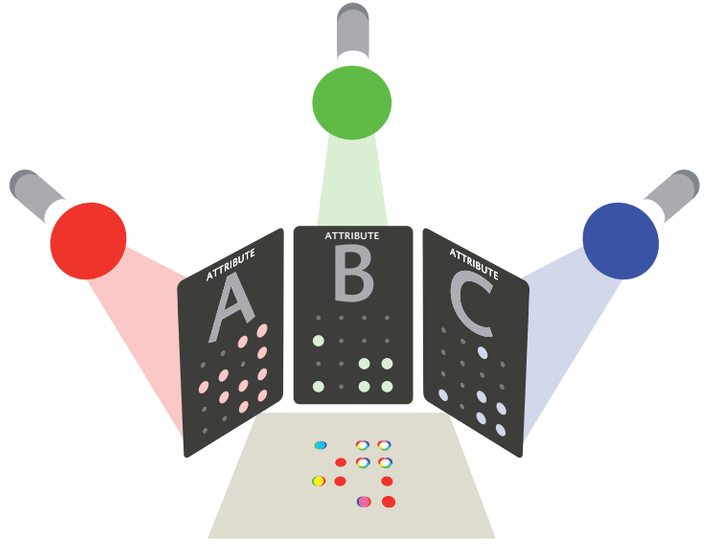
Rather that representing entities as boxes, in this system they are locations in a grid on a projection screen. Attributes are cards instead of frames, with holes in positions corresponding to particular entities which possess that attribute (allowing light to pass) and no holes for entities not sharing that attribute. When multiple primary colored lights are projected through these cards, onto a single target, the correlations between attributes can be seen through the color mixtures produced.



ADDITIVE INTERFERENCE WITH COLOR
If each primary color encodes an attribute, elements sharing two attributes will mix to form a secondary color while elements with all three will be pure white.



CARDS REPRESENT ATTRIBUTES
Elements in the database are encoded by position on the card. Those possessing a given attribute have a hole allowing light to pass. Elements lacking it are opaque.



CORRELATED ATTRIBUTES MIX COLORS
When light is shown through the cards in an convergent pattern, elements with more than one attribute will allow multiple colors to illuminate their address on the screen.

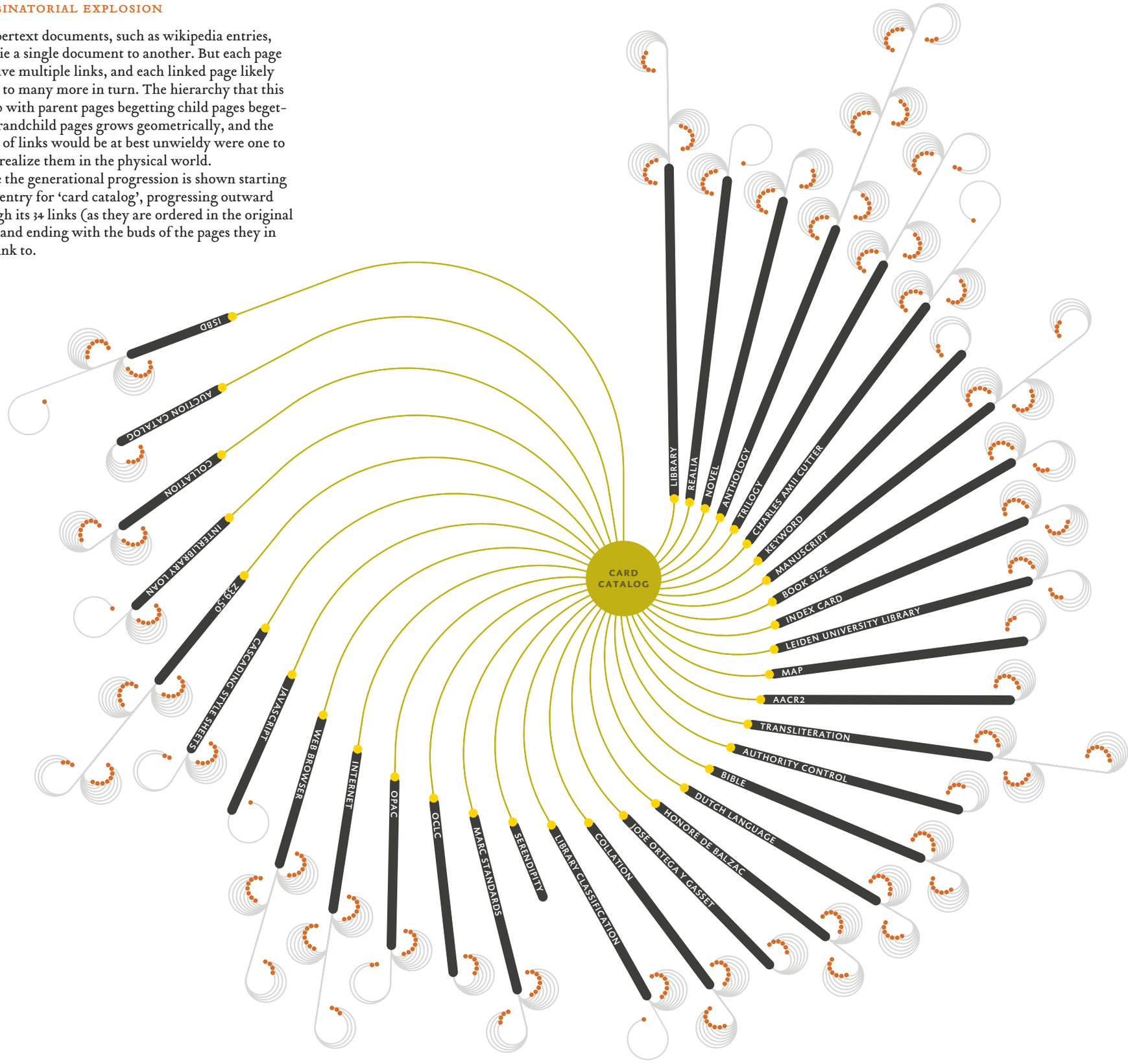
PARENT PAGE
CHILD PAGES
GRANDCHILD PAGES

- library
- realia
- novel
- anthology
- trilogy
- charles ammi cutter
- keyword
- manuscript
- book size
- index card
- leiden university library
- map
- AACR2
- transliteration
- authority control
- bible
- dutch language
- honoré de balzac
- josé ortega y gasset
- collation
- library classification
- serendipity
- MARC standards
- OCLC
- OPAC
- internet
- web browser
- javascript
- cascading style sheets
- z39.50
- interlibrary loan
- collation
- auction catalog
- ISBD

COMBINATORIAL EXPLOSION

In hypertext documents, such as wikipedia entries, links tie a single document to another. But each page can have multiple links, and each linked page likely points to many more in turn. The hierarchy that this sets up with parent pages begetting child pages begetting grandchild pages grows geometrically, and the tangle of links would be at best unwieldy were one to try to realize them in the physical world.

Here the generational progression is shown starting at the entry for 'card catalog', progressing outward through its 34 links (as they are ordered in the original text), and ending with the buds of the pages they in turn link to.



```
    return CardTree('Library_catalog',finalLinks)

# def fancyInsensitiveSort(aList):
#     realNames = dict([item.lower,item] for item in aList)
#
#     lcList = [item.lower for item in aList]
#     lcList.sort()
#
#     sortedList = [realNames[lcName] for lcName in lcList]
#     return sortedList

class CardTree:
    def insensitiveSort(aList):
        realNames = {}
        for item in aList:
            lcName = item.lower()
            if lcName[:4] == "the ": lcName = lcName[4:]
            realNames[lcName] = item
        lcList = realNames.keys()
        lcList.sort()
        sortedList = []
        for item in lcList:
            sortedList.append(realNames[item])
        #little hack - move the numbers to the back of the stack
        lastNumber = -1
        for item in sortedList:
            if item[0] in "0123456789":
                lastNumber += 1
            else:
                break
        if lastNumber >= 0:
            sortedList = sortedList[lastNumber+1:] + sortedList[:lastNumber+1]
        return sortedList

def spoolCards(nodeNames):
    #The cards are 432 x 306px (4.25" x 6")
    DPI = 72
    pageRect = CGRectMake(0, 0, 13*DPI, 19*DPI)
    cardRect = CGRectMake(0, 0, 6*DPI, 4.25*DPI)
    book = CGPDFContextCreateWithFilename("catalog spool.pdf", pageRect)
    vPadding = (pageRect.size.height - 4*cardRect.size.height)/2.0
    hPadding = (pageRect.size.width - 2*cardRect.size.width)/2.0
    for i in range(0,len(nodeNames),8):
        nextNodes = nodeNames[i:min(i+8,len(nodeNames))]
        print "Paginating", nextNodes
        book.beginPage(pageRect)
        #draw the crop marks
        for i in range(3):
            book.moveToPoint(hPadding+i*cardRect.size.width,vPadding/3)
            book.addLineToPoint(hPadding+i*cardRect.size.width, pageRect.size.height-
        for i in range(5):
            book.moveToPoint(5,vPadding+i*cardRect.size.height)
            book.addLineToPoint(pageRect.size.width-5,vPadding+i*cardRect.size.heigh
        book.setLineWidth(0.75)
        book.setRGBStrokeColor(0,0,0,1)
        book.strokePath()

        book.setRGBFillColor(1,1,1,1)
        book.fillRect(CGRectMake(hPadding-10, vPadding-10, pageRect.size.width-2*hPa
dding+20))
        #do the left column
        for j in range(min(4,len(nextNodes))):
            cardName = nextNodes.pop(0)
            card = CGPDFDocumentCreateWithProvider(CGDataProviderCreateWithFilename(
            cardRect.origin.x = hPadding
            cardRect.origin.y = vPadding + j*cardRect.size.height
            book.drawPDFDocument(cardRect,card,1)
        #do the right column
        for j in range(min(4,len(nextNodes))):
```