www.FUZION.org

Atomic Data Model

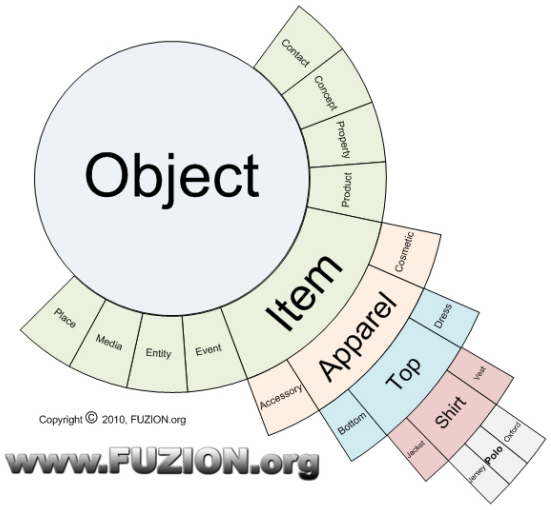
Atomic Data makes search engine dominance possible

Online retail is not the same as brick and mortar retail. When a brick and mortar store launches online they fall into this biggest trap. Take an apparel shop… when you first walk in you find a men's department and a ladies department. The store is physically trying to demographically segment you.

If you create a data model that matches this, you will end up with the first <xml> node being <gender> which is a highly limiting path to follow for a search engine even though it may make the most sense for a human being. You would then add data for teams, sports, colors, sizes, variants, materials of manufacture, and many other "parameters" for this data. To avoid 3rd normal database limitation, you would start to peel this data out into separate tables… one for colors… one for teams…one for sports. Then you would need to create many-to-many crosslink tables. Over time, your table count just gets larger and larger as new needs arise.

```
<Root-Object>
  <Item>
    <Toy/>
    <Instrument/>
    <Gear/>
    <Novelty/>
    <Office-Supply/>
    <Vehicle/>
    <Apparel>
      <Suit/>
      <Accessory/>
      <Bottom/>
      <Dress/>
      <Top>
        <Shirt>
          <Jersey/>
          <Oxford/>
          <Polo/>
        </Shirt>
        <Sweater/>
        <Sweatshirt/>
        <Jacket/>
        <Vest/>
      </Top>
      <Footwear/>
      <Headwear/>
      <Underwear/>
      <Gloves/>
      <Jewelry/>
      <Tattoo/>
    </Apparel>
    <Part/>
    <Food/>
  </Item>
  <Event/>
  <Entity/>
  <Media/>
  <Place/>
</Root-Object>
```

The Root Object Classification



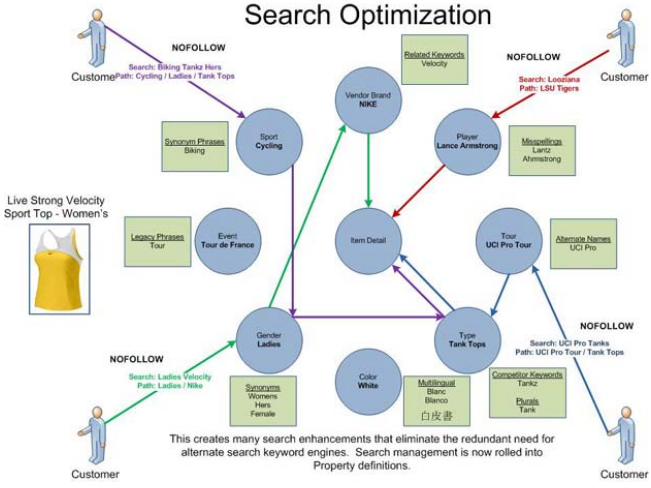
There is certain data that "hangs" off each sub-classification. In this example the Item class stores who the manufacturer is (because most items have manufacturers). The Apparel class contains the style information (because style is global to all apparel objects), whereas the Shirt class contains collar styles, sleeve variants, etc.

By localizing this information to class levels, once I define a "field" for the Apparel class, all future objects that inherit from that class will inherit that field. Any objects that do not inherit from the Apparel class will not have the field at all.

Note how different this is from a traditional 3rd normal representation of data where we would have fields like "color1" and "color2" and "color3" simply to leave enough fields available just in case we might need them for a particular product application.

Maximum Flexibility for Customer Paths

Now that our data is structured with infinite flexibility while still retaining a core hierarchy (for default navigation purposes), when a customer walks into our store, we can simply ask Google "how they sent them" to us… and what keywords they used. Now when the customer enters our "store" we can toss all of the inventory up into the air and literally rebuild our store to match the words they used in the order they used them. Now they can enter as "ladies yellow tank top" and we structure our product data in terms of gender first, color next and product class third… but we also can welcome customers that ask for "white womens Nike shirt" which we do by scanning for aliases of class nodes, parent classes, and other permutations of the item for maximum comfort to the customer and higher conversion rates on sales.



1132 Edgewood Avenue South · Jacksonville · Florida · 32205
Producer@FUZION.org 904-638-2455

FUZION.org