Should I use EAV model?

I'm am designing my database/domain for an eCommerce application and I'm having a hard time figuring out how to store products.

The website will sell a wide range of products, pens, thongs, tattoos, umbrellas, everything. Each of these product will share a few common attributes, height, width, length, weight, etc but some products have special data. For example, pens have different ink colors and tips/lids and brochures can have different types of folds. So far I have thought up some 20+ extra attributes, but these attributes may only apply to 1% of products on the website.

So I am wondering if it is appropriate to implement a EAV model to handle the extra data. Keeping in mind that when customers are viewing the site in the frontend, there will be a filtering sidebar like on eBay and carsales.com.au. (So keeping in mind there will be a fair bit of querying)

I don't think it's practical to implement Class Table inheritance as the system needs to remain flexible. This is because, down the track we may have more attributes in the future with new types of products.

The other thing I have considered is using a NoSQL database (probably MongoDB) however I have little experience with these types of databases, will it even solve my problem?

Review of options:

1. Single products entity with lots of columns
2. Separate attributes entity (EAV)
3. Switch to schema-less persistence

I'm in the process of building a prototype with an attributes entity to see how flexible it is, and testing the performance and how out of control the querying gets.

EDIT: I am, of course, open to any other solutions.

php database-design magento entity-attribute-value doctrine2

edited Sep 14 '12 at 5:43

asked Nov 1 '10 at 3:21
Cobby
2,581 1 14 32

6 Is there a particular reason you are hand-building an eCommerce app? It could be difficult, dangerous and perhaps a little unnecessary... – Jonathan Day Nov 1 '10 at 4:07

Just found this question which seems to be pretty much the same as what you're asking. – BenV Nov 1 '10 at 4:09

1 @BenV, while the question is similar, the answers are likely to be quite different, particularly relating to Magento. Magento did really struggle with EAV performance initially, but they have resolved that through careful caching (query, model, html-block and full-page) and optional denormalization. These are new developments since that question was asked, and worthy of re-asking the question in that context, IMHO. – Jonathan Day Nov 1 '10 at 4:28

The application I am making is very different from a typical eCommerce app. The only possible existing solution I have found in PHP is Magento, but it seems to be rather complicated and will be just as hard/time-consuming for me to learn it then to make my own from scratch. – Cobby Nov 1 '10 at 4:29

8 @Cobby, fair enough, you know your requirements better than anyone else, but with all due respect, if you feel that learning Magento is complicated, then writing your own eCommerce application is not something that I'd recommend. There are literally tens of thousands of Developer and Architect hours gone into applications like Magento, and a lot of battle scars along the way. When you're dealing with customers' or your clients' money, I'd rather have **someone else** made those mistakes in the past, not me... – Jonathan Day Nov 1 '10 at 4:46

3 Answers

Great question, but of course, there is no "one true way". As per @BenV, Magento does use the EAV model. My experience with it has been overwhelmingly positive, however it does trip up other users. Some considerations:

1. Performance. EAV requires complex, multi-table joins to populate your object with the relevant attributes. That does incur a performance hit. However, that can be mitigated through careful caching (at all levels through the stack, including query caching) and the selective use of denormalization. Magento does allow administrators to select a denormalized model for categories and products where the number of SKUs warrants it (generally in the thousands). That in turn requires Observers that trigger re-indexing (always good!) and updates to the "flat" denormalized tables when product data changes. That can also be scheduled or manually triggered with a prompt to the administrator.

2. 3rd Party User Complexity If you ever plan to make this application available to other users, many will find EAV too complex and you'll end up dealing with a lot of bleating and uninformed abuse on the user forums (ref Magento!!).

3. Future extensibility and plugin architecture. There is no doubt that the EAV model really comes into it's own when extensibility is a factor. It is very simple to add new attributes into the model while minimizing the risk of breaking existing ORM and controller code.

4. Changes in datatype EAV does make it a little harder to alter attribute datatypes. If your initial design calls for a particular attribute datatype that changes in future (say `int` to `varchar`), it means that you will have to migrate all the records for that attribute to the corresponding table that matches the new datatype. Of course, purists would suggest that you get the design right first time, but reality does intrude sometimes!

5. Manual product imports One thing that EAV makes almost impossible is importing products (or other entities) into the database using SQL and/or phpMyAdmin-style CSV/XML. You'll need to write an Importer module that accepts the structured data and passes it through the application's Model layer to persist it to the database. That does add to your complexity.

edited Nov 1 '10 at 4:23

answered Nov 1 '10 at 4:05
Jonathan Day
13.4k 3 48 100

Excellent answer. – timdev Nov 1 '10 at 4:32

I don't think performance will be a major concern, Doctrine 2 already has a query cache, plus I will implement my own level. Furthermore, it's important to remember that the attributes only have to main purposes: 1. To filter lists of products 2. Display on a product page. The data contained in the attributes is only metadata and isn't necessary for the application to work. I guess they are a bit like tags, in which case EAV seems practical. – Cobby Nov 1 '10 at 4:40

2 @Cobby Jonathan's point about caching stands. How do you generate totals for the filters? How to you efficiently serve pages for large numbers of customers? If you insist on building this yourself, at least do yourself a favor and review the changes that Magento made to their EAV system to allow for scale. – Joseph Mastey Nov 1 '10 at 15:06

Just playing devils advocate - My feelings exactly on EAV: weblogs.sqlteam.com/davidm/articles/12117.aspx#42216 I'm not the only one who feels the same: activecodeline.net/f-magento-eav – B00MER Mar 29 '11 at 19:24

1 An advantage of EAV model is that changes to schema can usually be acheived without changing the database schema. This is important because in MySQL ALTER TABLE statements lock the table to prevent all reads and writes during schema changes. Hence if you need to change schema on a large table this can result in locking issues and potentially outages while the ALTER TABLE runs. (Think very large and critical tables such as customers or products on an eCommerce site). – Jim OHalloran Jan 20 '12 at 0:37

The open source shopping cart Magento allows custom attributes for their products using an EAV design. You can check out their database schema here.

answered Nov 1 '10 at 3:52
BenV
4,084 3 16 48

1 Yes, I have already reviewed their database design, which is what I am basing my prototype off. My question isn't about implementation of EAV... I already know how to use it. My question IS about the architectural practicality of using EAV in a production environment. – Cobby Nov 1 '10 at 3:55

I would suggest you to look closer on Doctrine 2 ORM with OXM plugin for it (https://github.com/doctrine/oxm). It will solve your problem with different attributes. Of course you will be required to build indexes for searchable custom attributes, but I don't think it will be a problem :)

If you don't care about number of community members, then you can use MongoDB as well.

answered Mar 11 '11 at 10:48

Ivan Chepurnyi
6,966 1 27 39