Entity Attribute Value Database vs. strict Relational Model Ecommerce question

It is safe to say that the EAV/CR database model is bad. That said,

Question: **What database model, technique, or pattern should be used to deal with "classes" of attributes describing e-commerce products which can be changed at run time?**

In a good E-commerce database, you will store classes of options (like TV resolution then have a resolution for each TV, but the next product may not be a TV and not have "TV resolution"). How do you store them, search efficiently, and allow your users to setup product types with variable fields describing their products? If the search engine finds that customers typically search for TVs based on console depth, you could add console depth to your fields, then add a single depth for each tv product type at run time.

There is a nice common feature among good e-commerce apps where they show a set of products, then have "drill down" side menus where you can see "TV Resolution" as a header, and the top five most common TV Resolutions for the found set. You click one and it only shows TVs of that resolution, allowing you to further drill down by selecting other categories on the side menu. These options would be the dynamic product attributes added at run time.

Further discussion:

So long story short, **are there any links out on the Internet or model descriptions that could "academically" fix the following setup?** I thank Noel Kennedy for suggesting a category table, but the need may be greater than that. I describe it a different way below, trying to highlight the significance. I may need a viewpoint correction to solve the problem, or I may need to go deeper in to the EAV/CR.

Love the positive response to the EAV/CR model. My fellow developers all say what Jeffrey Kemp touched on below: "new entities must be modeled and designed by a professional" (taken out of context, read his response below). The problem is:

- entities add and remove attributes weekly
  (search keywords dictate future attributes)
- new entities arrive weekly
  (products are assembled from parts)
- old entities go away weekly
  (archived, less popular, seasonal)

The customer wants to add attributes to the products for two reasons:

- department / keyword search / comparison chart between like products
- consumer product configuration before checkout

The attributes must have significance, not just a keyword search. If they want to compare all cakes that have a "whipped cream frosting", they can click cakes, click birthday theme, click whipped cream frosting, then check all cakes that are interesting knowing they all have whipped cream frosting. This is not specific to cakes, just an example.

sql database design-patterns entity-attribute-value key-value

edited May 18 '09 at 16:54

asked May 15 '09 at 20:48
Dr. Zim
7,858 16 75 148

8 this is gonna get rough – JohnIdol May 15 '09 at 20:51

Why can't you just have a 'category' table with a foreign key refering to itself? – Noel Kennedy May 15 '09 at 21:31

20 It's not safe, nor precise, to say that the EAV database model is bad, because it is well suited to some applications. – spencer7593 May 15 '09 at 23:16

What if you decorate various objects with various properties, inheriting from a parent like in Entity Framework 4? How does it persist those objects? – Dr. Zim Jul 27 '11 at 2:26

Just back to point to this excellent article about one consultant's experience with a system based on an *extreme* version of EAV. Read it! simple-talk.com/opinion/opinion-pieces/bad-carma – Jeffrey Kemp Jun 13 '13 at 1:51

9 Answers

There's a few general pros and cons I can think of, there are situations where one is better than the other:

Option 1, EAV Model:

- Pro: less time to design and develop a simple application
- Pro: new entities easy to add (might even be added by users?)
- Pro: "generic" interface components
- Con: complex code required to validate simple data types
- Con: much more complex SQL for simple reports
- Con: complex reports can become almost impossible
- Con: poor performance for large data sets

Option 2, Modelling each entity separately:

- Con: more time required to gather requirements and design
- Con: new entities must be modelled and designed by a professional
- Con: custom interface components for each entity
- Pro: data type constraints and validation simple to implement
- Pro: SQL is easy to write, easy to understand and debug
- Pro: even the most complex reports are relatively simple
- Pro: best performance for large data sets

Option 3, Combination (model entities "properly", but add "extensions" for custom attributes for some/all entities)

- Pro/Con: more time required to gather requirements and design than option 1 but perhaps not as much as option 2 *
- Con: new entities must be modelled and designed by a professional
- Pro: new attributes might be easily added later on
- Con: complex code required to validate simple data types (for the custom attributes)
- Con: custom interface components still required, but generic interface components may be possible for the custom attributes
- Con: SQL becomes complex as soon as any custom attribute is included in a report
- Con: good performance generally, unless you start need to search by or report by the custom attributes

* *I'm not sure if Option 3 would necessarily save any time in the design phase.*

Personally I would lean toward option 2, and avoid EAV wherever possible. However, for some scenarios the users need the flexibility that comes with EAV; but this comes with a great cost.

edited Jun 26 '13 at 23:03

answered May 18 '09 at 6:29
Jeffrey Kemp
30k 7 43 87

What if you had a single table with indexes for text values 1-n, then in C# (in ram) map what you want to what you need. It would still work like an EAV, but the "matches" would be domain models. Sort of like a serialization, but you could use SQL selects on indexed text fields. No multiple selects per record. All the "cost" happens in RAM. – Dr. Zim Jun 15 '10 at 3:06

1 @Zim, that sounds pretty much like option 3. Each row has 1-n extra "generic" columns, and the data stored in them is interpreted at the application level. You get the performance benefit of having all the data for one record in one place. The metadata about those columns needs to be stored somewhere, however, and this is where the cost creeps in. Sure, we can cache the metadata in ram, but it still costs more than having the domain modelled directly in application code. Certainly better than a fullfledged EAV model though! – Jeffrey Kemp Jun 15 '10 at 11:52

+10000 Great answer. Nowadays people skimp on database design and requirement gathering. They'd rather write a hundred times more lines of code, that take the time to make a good design. – user1598390 Aug 17 '14 at 3:56

> It is safe to say that the EAV/CR database model is bad.

No, it's not. It's just that they're an inefficient usage of relational databases. A purely key/value store works great with this model.

Now, to your real question: How to store various attributes and keep them searchable?

Just use EAV. In your case it would be a single extra table. index it on both attribute name and value, most RDBMs would use prefix-compression to on the attribute name repetitions, making it really fast and compact.

EAV/CR gets ugly when you use it to replace 'real' fields. As with every tool, overusing it is 'bad', and gives it a bad image.

answered May 15 '09 at 21:44
Javier
36.7k 5 51 95

1 Great pragmatic answer! – MikeSchinkel Dec 4 '10 at 10:21

I'm surprised nobody mentioned NoSQL databases.

I've never practiced NoSQL in a production context (just tested MongoDB and was impressed) but the whole point of NoSQL is being able to save items with varying attributes in the same "document".

answered Apr 5 '11 at 23:10
Lucas T
350 4 11

Consider that writes to MongoDB require *database-level locking*, and what that means for concurrent production traffic. – Bill Karwin Mar 13 '13 at 21:22

Consider that the lock duration is in the order of microseconds. – Hello World Jan 2 at 16:06

```
// At this point, I'd like to take a moment to speak to you about the Magento/~~Adobe PSD~~
~~format~~.
// Magento/~~PSD~~ is not a good ecommerce platform/~~format~~. Magento/~~PSD~~ is not even a bad
ecommerce platform/~~format~~. Calling it such would be an
// insult to other bad ecommerce platform/~~formats~~, such as Zencart or OsCommerce. No,
Magento/~~PSD~~ is an abysmal ecommerce platform/~~format~~. Having
// worked on this code for several weeks now, my hate for Magento/~~PSD~~ has grown to a
raging fire
// that burns with the fierce passion of a million suns.
```

http://code.google.com/p/xee/source/browse/trunk/XeePhotoshopLoader.m?spec=svn28&r=11#107

The internal models are wacky at best, like someone put the schema into a boggle game, sealed that and put it in a paint shacker...

Real world: I'm working on a midware fulfilment app and here are one the queries to get address information.

```
CREATE OR REPLACE VIEW sales_flat_addresses AS
SELECT sales_order_entity.parent_id AS order_id,
       sales_order_entity.entity_id,

CONCAT(CONCAT(UCASE(MID(sales_order_entity_varchar.value,1,1)),MID(sales_order_entity_varcha
 "Address") as type,
       GROUP_CONCAT(
         CONCAT( eav_attribute.attribute_code," ::::: ", sales_order_entity_varchar.value
)
         ORDER BY sales_order_entity_varchar.value DESC
         SEPARATOR '!!!!!'
       ) as data
  FROM sales_order_entity
       INNER JOIN sales_order_entity_varchar ON sales_order_entity_varchar.entity_id =
sales_order_entity.entity_id
       INNER JOIN eav_attribute ON eav_attribute.attribute_id =
sales_order_entity_varchar.attribute_id
   AND sales_order_entity.entity_type_id =12
 GROUP BY sales_order_entity.entity_id
 ORDER BY eav_attribute.attribute_code = 'address_type'
```

Exacts address information for an order, lazily

--

Summary: Only use Magento if:

1. You are being given large sacks of money
2. You must
3. Enjoy pain

edited Sep 28 '10 at 14:26 answered Sep 27 '10 at 21:15

Vee
491 8 8

This is an older post but I wish I had found this 3 months ago when I started a Magento project for a client. +1 for the boggle/paint-shaker analogy! – nameEqualsPNamePrubeGoldberg Jan 12 '11 at 21:52

1 Pretty interessting, magento seems like it's the king-of-the-road in terms of e-commerce systems. Maybe only it's marketing is very good – Herr Aug 17 '11 at 9:22

1 @Vee FYI, The link is broken.. – Mr_Green Oct 23 '13 at 8:32

Where performance is not a major requirement, as in an ETL type of application, EAV has another distinct advantage: differential saves.

I've implemented a number of applications where an over-arching requirement was the ability to see the history of a domain object from its first "version" to it's current state. If that domain object has a large number of attributes, that means each change requires a new row be inserted into it's corresponding table (not an update because the history would be lost, but an insert). Let's say this domain object is a Person, and I have 500k Persons to track with an average of 100+ changes over the Persons life-cycle to various attributes. Couple that with the fact that rare is the application that has only 1 major domain object and you'll quickly surmize that the size of the database would quickly grow out of control.

An easy solution is to save only the differential changes to the major domain objects rather than repeatedly saving redundant information.

All models change over time to reflect new business needs. Period. Using EAV is but one of the tools in our box to use; but it should never be automatically classified as "bad".

answered Jul 20 '11 at 13:04
Jerry Jasperson
79 1 1

I'm struggling with the same issue. It may be interesting for you to check out the following discussion on two existing ecommerce solutions: Magento (EAV) and Joomla (regular relational structure): https://forum.virtuemart.net/index.php?topic=58686.0

It seems, that Magento's EAV performance is a real showstopper.

That's why I'm leaning towards a normalized structure. To overcome the lack of flexibility I'm thinking about adding some separate data dictionary in the future (XML or separate DB tables) that could be edited, and based on that, application code for displaying and comparing product categories with new attributes set would be generated, together with SQL scripts.

Such architecture seems to be the sweetspot in this case - flexible and performant at the same time.

The problem could be frequent use of ALTER TABLE in live environment. I'm using Postgres, so its MVCC and transactional DDL will hopefully ease the pain.

answered Dec 28 '09 at 10:04
aaimnr
1,133 1 9 27

I still vote for modeling at the lowest-meaningful atomic-level for EAV. Let standards, technologies and applications that gear toward certain user community to decide content models, repetition needs of attributes, grains, etc.

answered May 7 '10 at 16:58
Amanda Xu
11 1

If it's just about the product catalog attributes and hence validation requirements for those attributes are rather limited, the only real downside to EAV is query performance and even that is only a problem when your query deals with multiple "things" (products) with attributes, the performance for the query "give me all attributes for the product with id 234" while not optimal is still plenty fast.

One solution is to use the SQL database / EAV model only for the admin / edit side of the product catalog and have some process that denormalizes the products into something that makes it searchable. Since you already have attributes and hence it's rather likely that you want faceting, this something could be Solr or ElasticSearch. This approach avoids basically all downsides to the EAV model and the added complexity is limited to serializing a complete product to JSON on update.

answered Sep 27 '14 at 18:08
bob
98 1 6

EAV has many drawbacks:

1. Performance degradation over time Once the amount of data in the application grows beyond a certain size, the retrieval and manipulation of that data is likely to become less and less efficient.
2. The SQL queries are very complex and difficult to write.
3. Data Integrity problems. You can't define foreign keys for all the fields needed.
4. You have to define and maintain your own metadata.

edited 2 days ago
herry
1,319 2 10 23

answered 2 days ago
Gabriel Voinea
1