Product Model Versioning

Maxim Noah Khailo

March 9, 2016

1 Purpose

The product model described in "The Enlightened, Post-Modern Product Model" article went a long way describing a way to manage change based on context. This article takes the design and pushes it further to include versioning.

Because not only does the product change in context, but also changes in time.

2 Forces

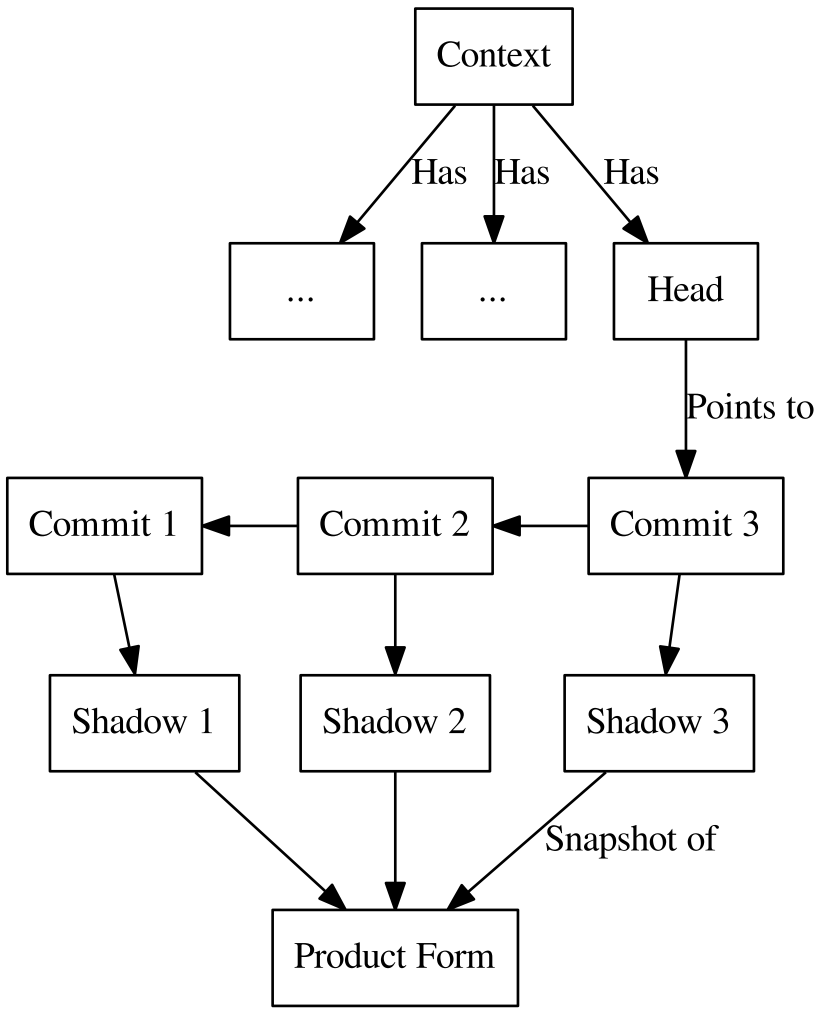
We have to balance several problems.

- Provide Change in context and time.
- Handle various UI paradigms for showing differences and undo.
- Optimize data for storefront while maintaining control in PIM.

3 The Shadow Change Log

The Product Form is composed of named attributes. Each attribute has a type, and a set of versions. A shadow points to specific versions of attributes on a form.

We can represent versioning a product as a log of product shadows. Here we introduce the concept of a product commit, which points to a shadow. Since a shadow is a "snapshot" of a product form, we can represent all history to changes of a product within a specific context over time.



Context
Has
Has
Has
...
...
Head
Points to
Commit 1
Commit 2
Commit 3
Shadow 1
Shadow 2
Shadow 3
Snapshot of
Product Form

4 Changes to Product Form and Shadow

The changes to the product model are that the named attributes have a set of versions of the attribute. Before we used named keys but why not just set the key name to a hash of the content?

Here is the old model...

```
//Old Model Product
{
    id: 3,
    name: {
        type: "string",
        x: "Red Shoe",
        y: "Big Red Shoe",
    },
}
//Old Shadow
{
    productId: 3,
    name : "y"
}
```

Instead of "x" and "y", we compute the SHA-1 hash of the content and take the first x values of the hash. Lets say we take the first 7.

```
//New Model Product
{
    id: 3,
    name: {
        type: "string":
        "d5ae001": "Red Shoe",
        "2981c9f": "Big Red Shoe",
    },
}
//New Shadow
{
    productId: 3,
    name : "2981c9f"
}
```

Notice here, all a shadow is is a pointer to versions of attributes in the product form. If we use the git revision control system as an analogy, the product shadow is similar a git tree object and the product form is a collection of attributes, where each attribute has a set of versions of that attribute.

5 The Product Commit Object

Since each shadow is a "snapshot" of a form, an ordered list of shadows that point to the same form could represent a history of changes of that product.

```
//Commit Object
{
    id: 3,
    shadowId: 5,
    productId: 4,
    previousCommitId: 2,
    reasonId: 10
}
```

5.1 Keeping Track of the Latest Shadow in a Context

To keep track of the set of latest shadows, we need another object that simply points to the latest commit. We will call the Head Object

```
//Head Object
{
    id: 3,
    contextId: 1,
    shadowId: 5,
    productId: 4,
    commitId: 3
}
```

We can also store the *shadowId* and *productId* here for convenience.

6 Discussion

There are some pros and cons to this design.

6.1 Pros

- The Form has all the versions of each attribute together. This should compress well.

- The Shadow is decoupled from the context. We can share a shadow into a context by creating a Head Object.
- The commit log is a tree. We can write merge algorithms later if we wanted to.
- We can use contexts the same way people uses branches in git. An admin can have their own context where they stage commits. Once they are approved The head object in the stores context can point to the correct commit.
- We can have contexts based on events. For example a context to stage commits for Easter and migrate the store to easter mode, then migrate back to normal mode trivially.
- Using contexts and head objects, we can trivially snapshot the store or provide custom catalogs.

6.2 Cons

- PIM UI, which is shadow aware now, will need to be commit aware.
- Is a full git type system in the DB a good idea? Could a form get too big?
- The type of attribute is on the attribute and not the version. This means shadows cannot change attribute type. Should we move type to be on the version?