
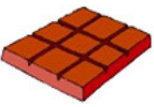






# The strict faceted classification model

Travis Wilson, Facetmap

## Summary of principles

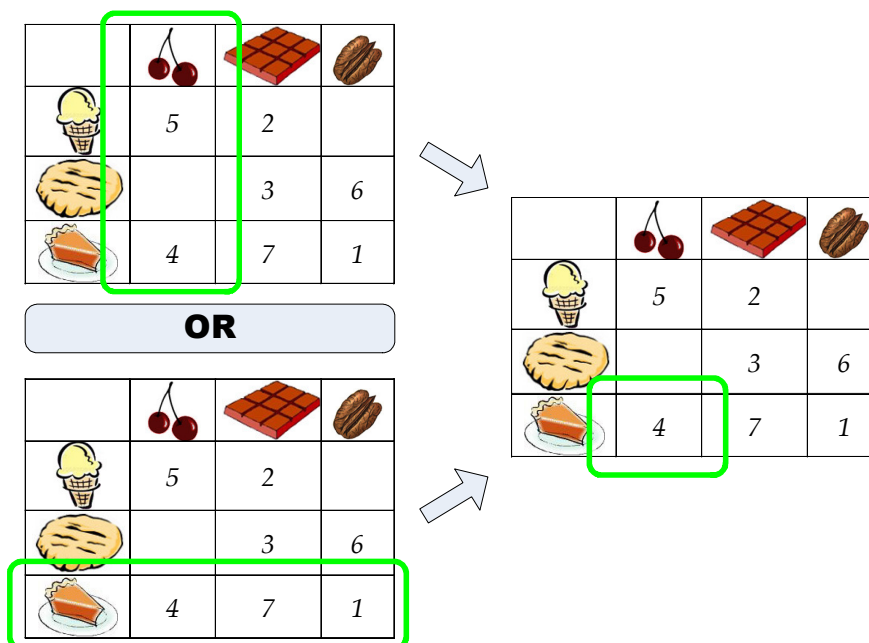
Faceted classification, at its core, implies orthogonality – that every facet axis exists at right angles to every other facet axis. That’s why a faceted classification is sometimes represented with a chart. This set of desserts has been classified by their confection types and, orthogonally, by their flavors:

Confection Type Flavor	 <b>Cherry</b>	 <b>Chocolate</b>	 <b>Pecan</b>
 <b>Ice Cream</b>	5	2	
 <b>Cookie</b>		3	6
 <b>Pie</b>	4	7	1

## Menu

1. Pecan Pie
2. Chocolate Ice Cream
3. Chocolate Cookie
4. Cherry Pie
5. Cherry Ice Cream
6. Pecan Cookie
7. Chocolate Pie

One application of this is flexible browsing; the browsing user can filter by row first or column first, and still end up with the same result:



If you're using faceted classification to organize your data, then perhaps you're familiar with this situation: you want to assign something two different headings from the same facet. For example, in the dessert scheme above, you want to assign both “Chocolate” and “Pecan” headings to a chocolate-pecan pie.

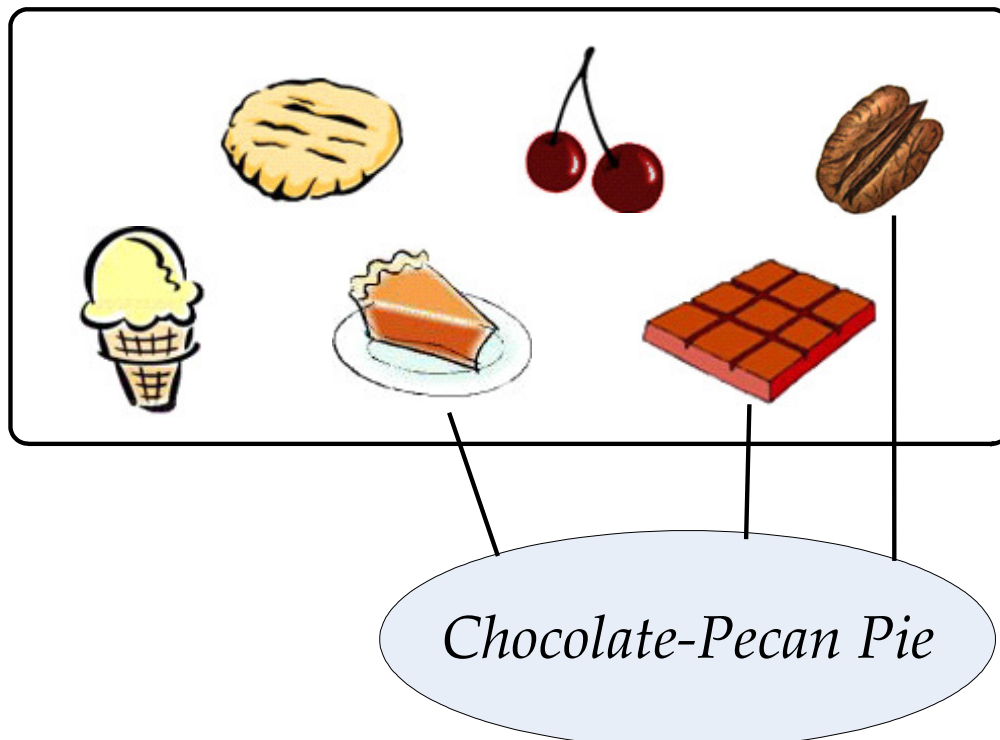
This innocent-looking scenario cuts open a surprisingly wide swath of information theory. It turns out that, given the facets in the example, *a strict faceted classification model forbids you to assign both those headings*, and with good reason. This is counterintuitive, controversial, and if you subscribe to S.R. Ranganathan's original facet theory, heretical.

Yet a faceted classification is most effective when built upon that restriction. This session explores how to use the restriction constructively, whether the faceted classification model really fits the user's intent, and what the alternatives are. We'll critically examine facet structures, especially in light of such new metadata schemes as tagging. Along the way we'll discover:

- the real nature of the faceted classification model
- the similarities to and differences from free-form tagging
- orthogonality
- binary facets
- recent suggested enhancements to the faceted classification model
- new technology that leverages the strict model to sort tags into facets
- how to make your classification as usable as possible

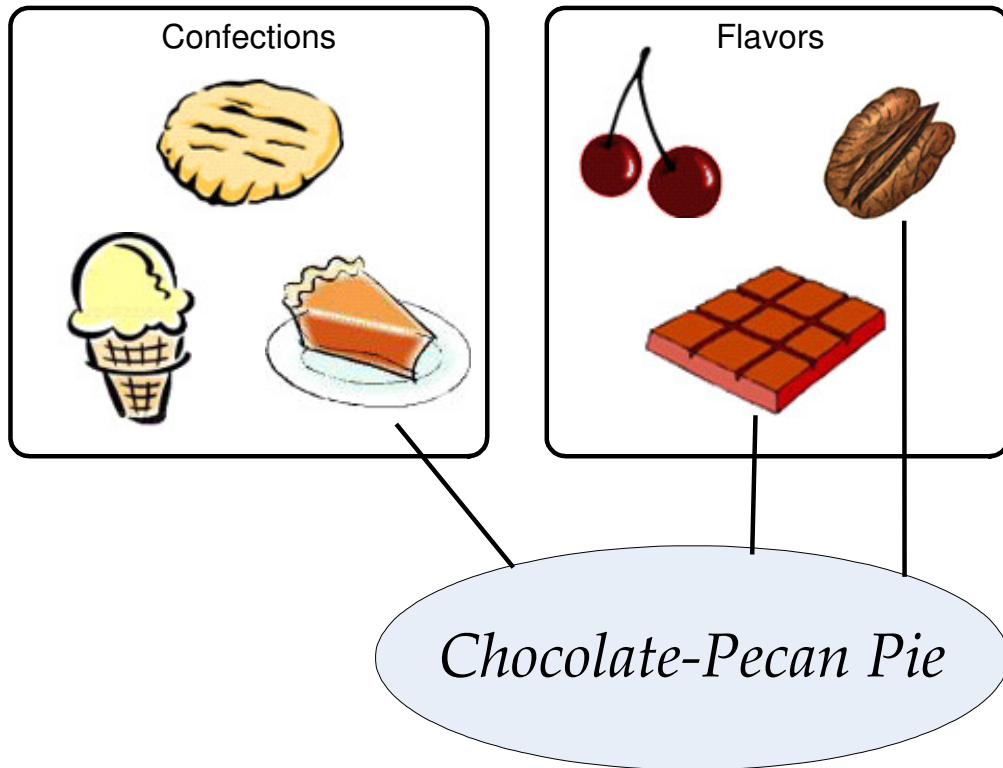
Over the last couple of years, tagging has emerged as a flexible method of assigning headings to resources, regardless of other headings on that resource. Its applications sharply illuminate its differences from faceted classification; in fact tagging is the closer of the two to Ranganathan's canons of facet theory.

Tags do not fit well into a chart model. The topography of tagging is more like this:



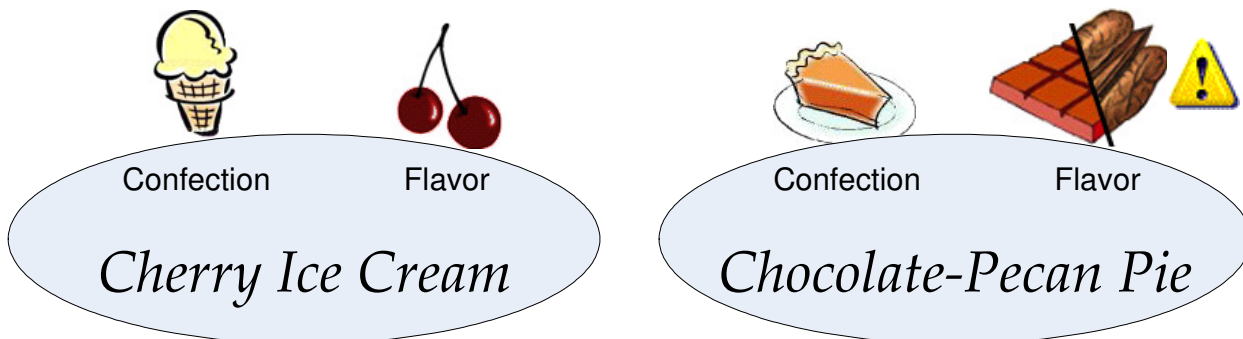
The resource can be connected to any tag, to indicate that it expresses the property that the tag represents. This connection can be made regardless of what other tags are already connected to the resource. So tags carry orthogonality to an extreme: *every* tag is orthogonal to every other tag! This can certainly be convenient, but it is different from a faceted classification in that it does not identify which headings are mutually exclusive.

The tagging situation is not really any different if we draw it this way:



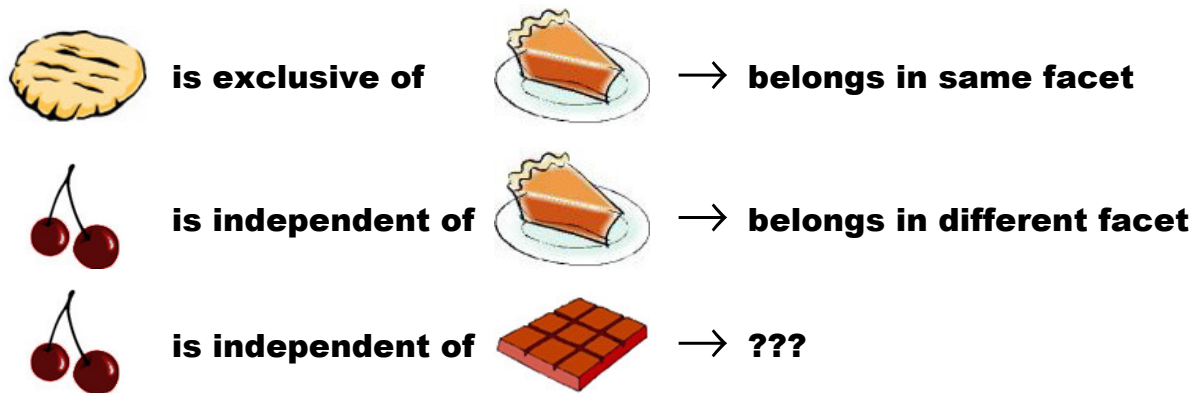
The separation of headings into two groups only enables us to present the headings to the user in a more organized fashion, for browsing or for classification. Grouped tags do not constitute a transition to faceted classification; nothing about the classification itself has changed, nothing different happens when you select one of the tags.

Strict faceted classification provides a different model. Instead of sorting the headings into intuitive groups, it sorts the orthogonal attributes of the resource itself. Originally, we said our desserts had two independent attributes – the “Confection” facet and the “Flavor” facet, and the each facet of each dessert resource was assigned a value thus:

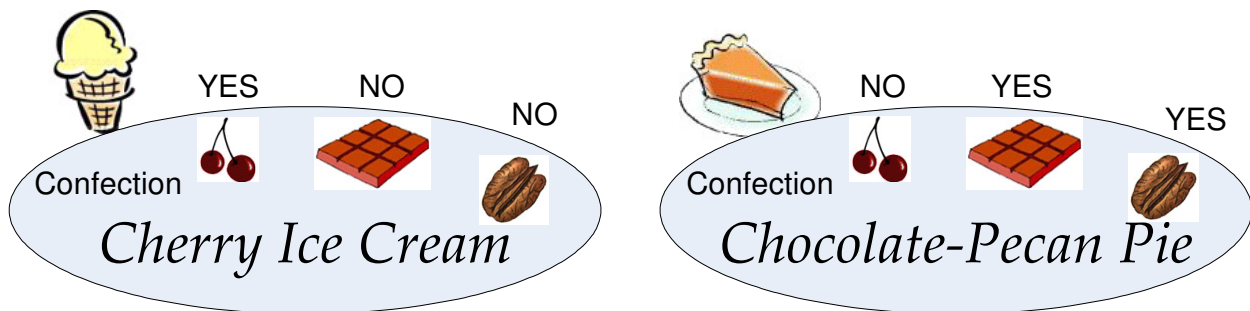


But this creates a huge headache! There is no Flavor heading defined for “Chocolate and Pecan”. One solution to the problem is to recast the whole classification into a tagging model, but we want to stay within the strict faceted classification model. So another solution would be to create hybrid headings within “Flavor”, but that has the same permutation-explosion problem that we hope to avoid by using faceted classification in the first place.

Consider instead the true meaning of the headings we’ve used. “Pie” means “the confection type of this resource is pie,” but “Chocolate” only means “this resource contains Chocolate.” The heading “Chocolate” only indicates the chocolateness of the dessert and says nothing about the cherriness or pecanness. Should it really be lumped into a facet with those other flavors?



In the same sense that a “Cherry” selection is independent of a “Pie” selection, a “Cherry” selection is also independent of a “Chocolate” selection. By strict facet definition, they are therefore in separate facets. Our desserts should actually look like this when faceted:



Now there are four facets. There are four separate questions that can be asked about a given dessert, for which the answer is independent of all the other answers – so four is the correct number of facets. Three of them are binary facets, having only “yes” and “no” headings (plus the usually implicit “undefined” heading).

For real-world situations where there are a lot of tags, that means a lot of facets. You will probably still want to group the facets for presentation to the user – but there is nothing inherently wrong with a lot of facets. Between this strict model and a loose-model classification that must anticipate an unknown number of tags on one facet of a resource, the strict model is still more efficient to implement in software.

Why would you use faceted classification at all, when tagging is available? There are several reasons. The right blend for a project, of course, depends on the nature of the project.

- **Clarity of information / well-defined orthogonality**
- **Easy to make more informative queries against the data**
- **Clearer structure for data mining**
- **Clearer directives for designing UI**
- **Better runtime performance**

This presentation is available in full, beginning April 2006, at

<http://facetmap.com/pub/>