- We need new tables and relationships for every new use case if we choose this structure.

So it can work, but I don't feel it's flexible enough to handle real-world use-cases.

Using a JSON column

Instead of using separate tables, we can add a JSON column to the broadcasts table, something like this:

```
{
  "form_ids": [1,2,3],
  "tag_ids": [12,4]
}
```

We can store a JSON array like this in a column. The data structure above describes a filter:

- Forms with the ID of 1,2,3
- Tags with the ID of 12,4

What are the advantages of this solution?

- Easy to implement.
- Easy to extend.
- Fast implementation. It can be an advantage in many projects and a disadvantage in others.
- Later, it's easy to convert it to separate tables if you change your mind.

What are the drawbacks of this solution?

- No data integrity. What if you delete the tag with the ID of 4? You need extra code to guarantee that none of the filters reference it anymore. However, in this situation, it's only partly true. Using `broadcast_form` and `broadcast_tag` pivot tables gives you data integrity. But we know that we need the same filters for sequences later. It means we are forced to use polymorph relationships, and instead of `broadcast_id` we have