

Relationship Building

Defining and Querying Complex Relationships
Between Your Content

Alec P. Mitchell

Existing Tools

- Archetypes Reference Engine
- External SQL database
- `zc.relationship` + `plone.relations`
- ???

Benefits of AT References

- Can relate any AT content to any other AT content.
- There are nice widgets available to manipulate relationships.
- Uses the familiar ZCatalog internally.
- Relationships can provide complex behaviors.

Disadvantages of AT References

- Requires mixin base classes. Essentially only works with AT content.
- References are intrinsic to their source, and generally defined as part of the content schema.
- General purpose ZCatalog not entirely optimized for relationship queries.
- API tends to be somewhat inconsistent, and test coverage is sub-optimal.

Benefits of an RDBMS-based Solution

- Can support arbitrarily complex relationships and queries.
- Can be performance tuned for specific use-cases.
- Can be used to relate anything to anything else.

Disadvantages of an RDBMS-based Solution

- Entirely DIY
- Requires an external database
- Need to create some 1-to-1 mapping between RDBMS keys and ZODB content.

What is `zc.relationship`

- A low-level ZODB index for querying relationships
 - Highly optimized for simple relationship queries across large data-sets.
 - Default configuration allows relationships between arbitrary persistent objects.
 - Index can be configured to index complex relationships, which may include non-ZODB objects.
 - Provides transitive searches.

What is plone.relationships

- A local utility built on `zc.relationship`, which is applicable to a wide variety of relationship models.
- A relationship class that models many-to-many content relationships.
- Some optional aspects of the relationship are also indexed:
 - Relationship Type
 - Relationship State
 - Relationship Context

What is plone.app.relationships

- A content-centric API for defining and querying relationships:

```
src = IRelationshipSource(ob)
src.createRelationship(target=ob2)
src.getTargets()
...
```

- A set of optional adapters and subscribers
 - DCWorkflow for relationships
 - “Holding” relationships
 - Relationships which are copied when their source is copied.

Relationship Sources

- `IRelationshipSource`
 - Create (`createRelationship`), supports multiple targets
 - Query (`getTargets`, `isLinked`, `getRelationshipChains`, `getRelationships`)
 - Modify (`deleteRelationship`, `getRelationships`)

Relationship Targets

- `IRelationshipTarget`
 - Same query methods and parameters as `IRelationshipSource` + `getSources`
- `ISymmetricRelation`
 - Query (`isLinked`, `getRelationships`, `getRelations`)
 - No transitivity, for now

Brief Code Example

```
>>> class IFriendship(IDCWorkflowableRelationship):
...     """A friendship"""
>>> source = IRelationshipSource(ob1)
>>> rel = source.createRelationship(ob2, relation='friend',
...                                 interfaces=(IFriendship,))
>>> list(source.getRelationships(relation='friend'))
[<Relationship 'friend' from (<Demo ob1>,) to (<Demo ob2>,)]
>>> list(rel.targets), list(rel.sources)
([<Demo ob2>], [<Demo ob1>])
>>> list(source.getTargets())
[<Demo ob2>]
>>> list(source.getRelationshipChains(target=ob2, maxDepth=5))
[(<Relationship 'friend' from (<Demo ob1>,) to (<Demo ob2>,)>,)]
>>> target = IRelationshipTarget(ob2)
>>> list(target.getSources(relation='friend'))
[<Demo ob1>]
>>> target.isLinked(source=ob1)
True
>>> list(ISymmetricRelation(ob1).getRelations())
[<Demo ob2>]
>>> list(ISymmetricRelation(ob2).getRelations())
[<Demo ob1>]
>>> wf_rel = IDCWorkflowRelation(rel)
>>> wf_rel.workflow_id = 'friend_workflow'
>>> wf_rel.doAction('approve')
>>> wf_rel.state
'approved'
```

What Can You Do With It?

- Model any non-container relationships you might need.
 - Social Networking (user→user, user→group/workspace)
 - User Favorites (user→content)
 - Placeless Content
 - Taxonomies or Complex Vocabularies
 - Related Content

What Have I Done With It?

- The Daily Reel (<http://www.thedailyreel.com>)
 - Social network for online video creators
 - Contacts (user relationships with approval)
 - Crews (discussion groups)
 - Videos which “belong” to crews and users (placeless content)
 - Favorite Content

Brief UI Demo

Shortcomings

- Relationships are inherently asymmetric, with sources and targets on unequal footing.
- Relationships are unordered
- Requires persistent sources/targets/contexts.
- Uses IntIds: fast but limits the number of indexed objects to `sys.maxint`.
- No user-space tools/UI, yet.

When to Use Something Else

- Efficient sorting on a large result set of relationships.
- Complex queries. Currently there's only '==' (for source/target, state, context)
- Non-ZODB objects
- Don't have time to make your own UI (stick with AT for now)

The Future?

- Support for explicit ordering of relationships
- API providing permission based filtering
- UI (schema field, formlib widget)
- Move general code downstream into `zc.relationship`

Thank You!

- Special thanks to
 - Balazs Ree
 - Gary Poster and Zope Corporation
 - Whit Morriss
 - Ramon Navarro Bosch
 - The Daily Reel
 - Vincenzo and the conference organizers
- And now we sprint!!