

Multidimensional Key, Version 1.0

URL

<http://www.wiley.com/compbooks/poole/patterns/MDKey.pdf>

Contributor

John Poole (jpoole@alum.poly.edu)

Structural Classification

Micro pattern

Usage Category

Structural

Intent

Provides a standard structural pattern by which the unique identities of members of a multidimensional structure (i.e., cube, dimension, level, or hierarchy) can be obtained.

Also Known As

- Primary key
- Unique key
- Member identifier
- Member name

Motivation

When using CWM to model various multidimensional structures (i.e., cube, dimension, level, and hierarchy), there is a need for a meta data component that assists in the identification of the elements (i.e., cells or members) described by (and conceptually stored by) those multidimensional structures. The well-known concept of a *primary key* or *unique identifier* can be applied here in terms of a specific usage of the UniqueKey metaclass defined by CWM.

Applicability

Use this pattern wherever a multidimensional structure must be capable of revealing a unique identifier of the instance values that the structure describes. This pattern has two variants, one which applies to server-side meta data only, and is not exposed by any client-side projection of the CWM metamodel. Instead, the presence of this pattern is exposed on the client-side by an application-specific tag or stereotype.

Projection

The Multidimensional Key pattern is based on a sub-graph of the CWM metamodel consisting of the metaclasses

- org.omg.cwm.foundation.keysindexes.UniqueKey
- org.omg.cwm.objectmodel.core.Class
- org.omg.cwm.objectmodel.core.Attribute
- org.omg.cwmobjectmodel.core.Stereotype
- javax.olap.clientsidemetadata.Cube
- javax.olap.clientsidemetadata.Dimension
- javax.olap.clientsidemetadata.MemberSelection
- javax.olap.clientsidemetadata.Level
- javax.olap.clientsidemetadata.Hierarchy
- javax.olap.clientsidemetadata.LevelBasedHierarchy
- javax.olap.clientsidemetadata.ValueBasedHierarchy

And the associations

- org.omg.cwm.objectmodel.core.ElementOwnership
- org.omg.cwm.objectmodel.core.ClassifierFeature
- org.omg.cwm.objectmodel.core.StereotypedElement
- org.omg.cwm.foundation.keysindexes.UniqueFeature
- org.omg.cwm.analysis.olap.DimensionOwnsMemberSelections
- org.omg.cwm.analysis.olap.DimensionOwnsMemberSelections
- org.omg.cwm.analysis.olap.DimensionOwnsHierarchies

This sub-graph is illustrated in the diagram below:

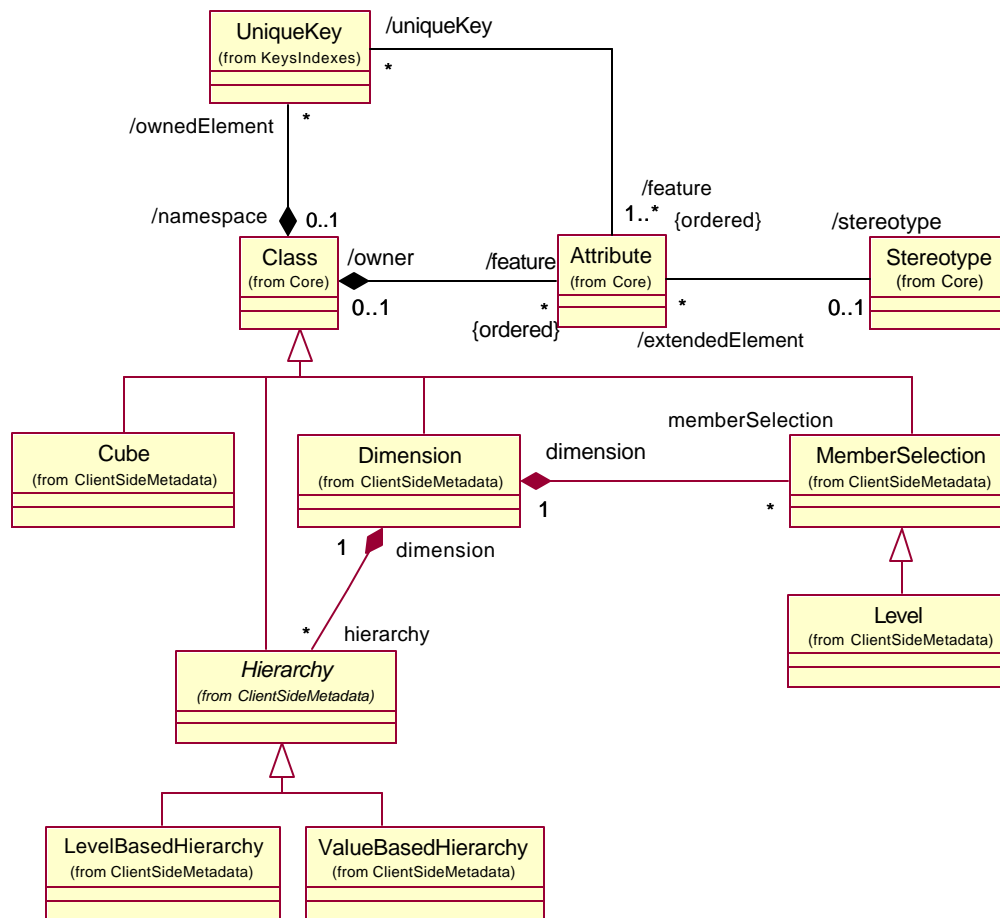


Figure 1: Multidimensional Key Projection

Restriction(s)

Restrictions on instances of the projection are as follows:

- The only allowable instances of Class are instances of the following subclasses or descendants: Cube, Dimension, Level, LevelBasedHierarchy, ValueBasedHierarchy.

This restriction is expressed formally by the following OCL constraint:

```

context Class
  inv: self.oclIsKindOf( Cube )
  inv: self.oclIsKindOf( Dimension )
  inv: self.oclIsKindOf( Level )
  inv: self.oclIsKindOf( LevelBasedHierarchy )
  inv: self.oclIsKindOf( ValueBasedHierarchy )

```

- In the case of client-side meta data, instances of Stereotype must be specified on client-side realizations of the projection. They are optional, but recommended, for use in server-side realizations of the pattern.
- Instances of UniqueKey are not to be included in any client-side realizations of the projection, but must be included on the server-side.

Usage

- The name of the Class instance is user-defined, and not prescribed by the pattern.
- The name of each Attribute comprising a multidimensional key is user-defined, and not prescribed by the pattern.
- The name of each UniqueKey is user-defined, and not prescribed by the pattern.
- The name of Stereotype is user defined, and not prescribed by the pattern.

Parameters

| M2 Parameter | M1 Value | Comments |
|--|-----------------|--------------------|
| Attribute.ModelElement::name | User-defined | Value is arbitrary |
| Class.ModelElement::name | User-defined | Value is arbitrary |
| Stereotype.ModelElement::name | User-defined | Value is arbitrary |
| Stereotype.baseClass | "Attribute" | Value is fixed |
| UniqueKey.ModelElement::name | User-defined | Value is arbitrary |
| Cube.ModelElement::name | User-defined | Value is arbitrary |
| Dimension.ModelElement::name | User-defined | Value is arbitrary |
| Level.ModelElement::name | User-defined | Value is arbitrary |
| LevelBasedHierarchy.ModelElement::name | User-defined | Value is arbitrary |
| ValueBasedHierarchy.ModelElement::name | User-defined | Value is arbitrary |

Commentary

The names of the various elements participating in any realization of the Multi-dimensional Key pattern are user-defined. That is, any software process driven by this pattern must be capable of discerning the pattern based strictly on structure, without any dependence on specific element names. This ensures that model element names are always meaningful the model user.

Consequences

TBD

Known Uses

TBD

Related Patterns

Local Stereotype, Version 1.0, describes how instances of the Stereotype class are situated in realizations of this pattern.

Sample Solution

The following diagrams illustrate two possible realizations of the Multidimensional Key pattern. In the first diagram, a server-side realization of the pattern is shown. In the second diagram, the client-side equivalent of the server-side realization of the pattern is shown. In both diagrams, those elements comprising the pattern realization specifically are highlighted in blue. This perhaps makes it more apparent why patterns like Multidimensional Key are classified as *structural micro-patterns* (i.e., they are used to define relatively fine-grained components of otherwise larger meta data structures).

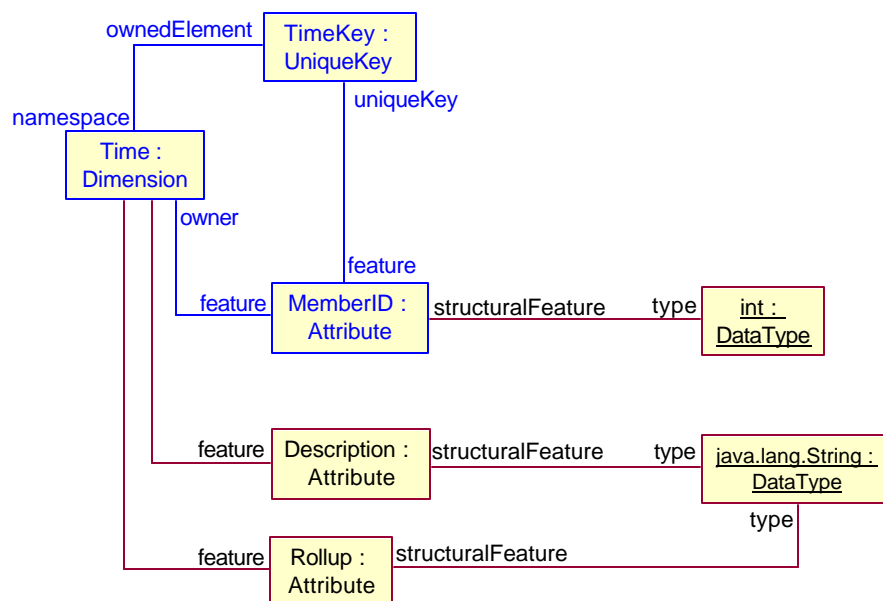


Figure 2: Simple Server-Side Dimension

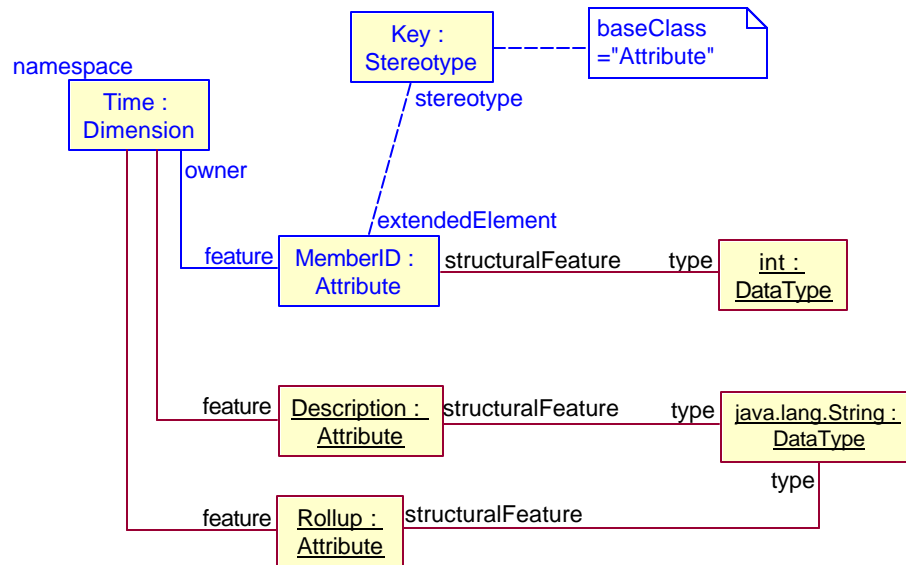


Figure 3: Equivalent Client-Side Dimension

The next example illustrates a simple model in which a Dimension contains multiple Levels. In this case, the Dimension is considered to be the union of its Levels. Therefore, any key structure associated with the Levels must be mapped to the key structure of the Dimension. This is accomplished on the server-side using CWM Classifier and Feature Maps. On the client-side, the equivalent realization has a `<<Key>>` stereotype denoting the key attributes, but no explicit mapping structure. Rather, the mapping is implicit in the semantics of the `<<Key>>` stereotype:

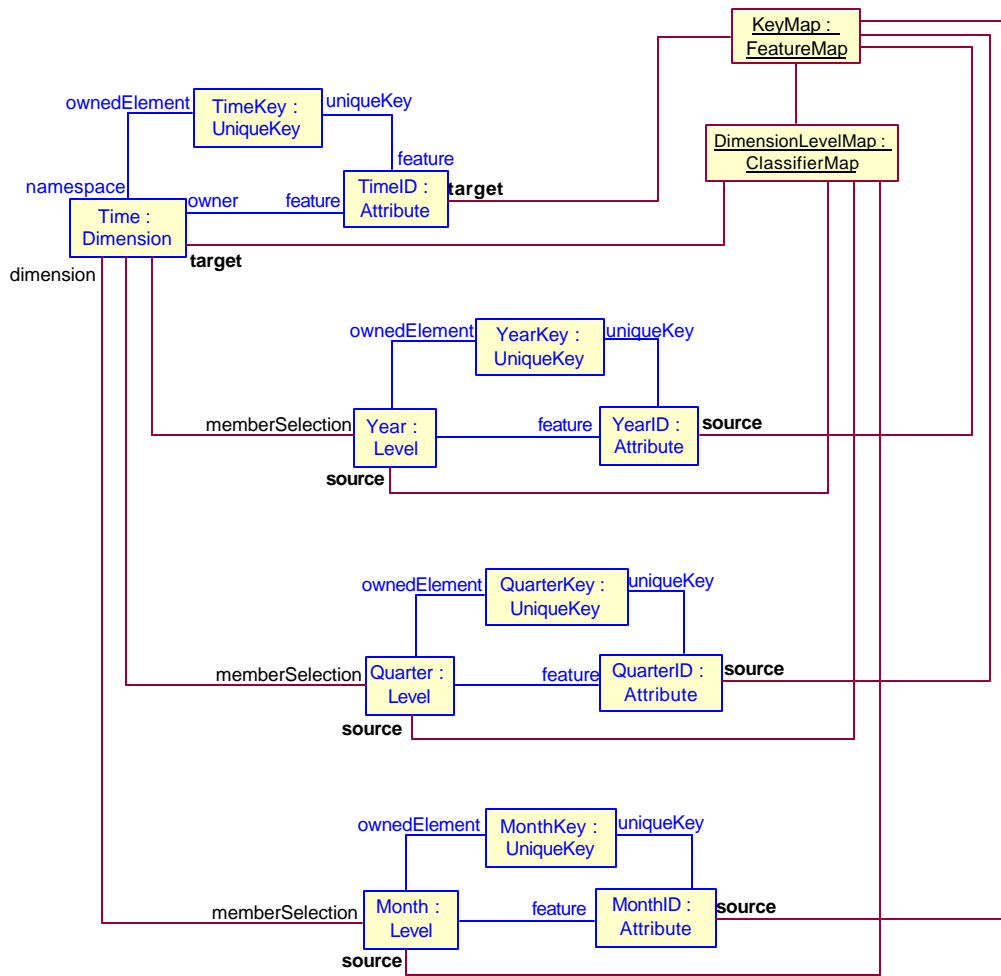


Figure 4: Server-Side Dimension with multiple, mapped Levels

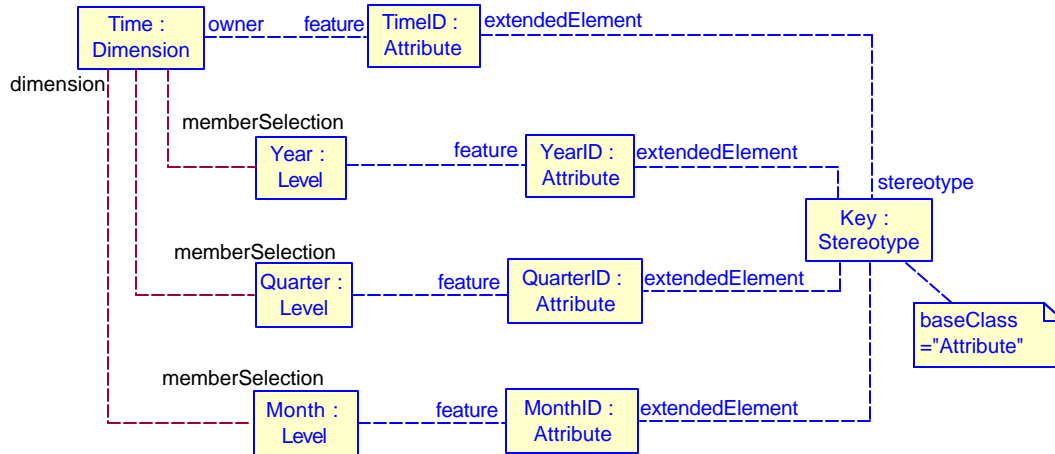


Figure 5: Equivalent Client-Side Dimension with multiple Levels