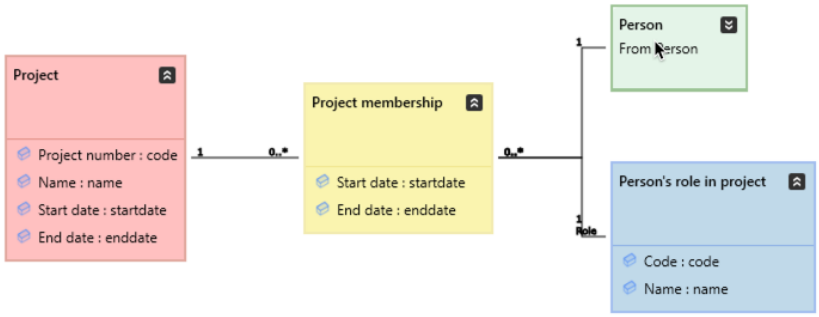
m:n Tables

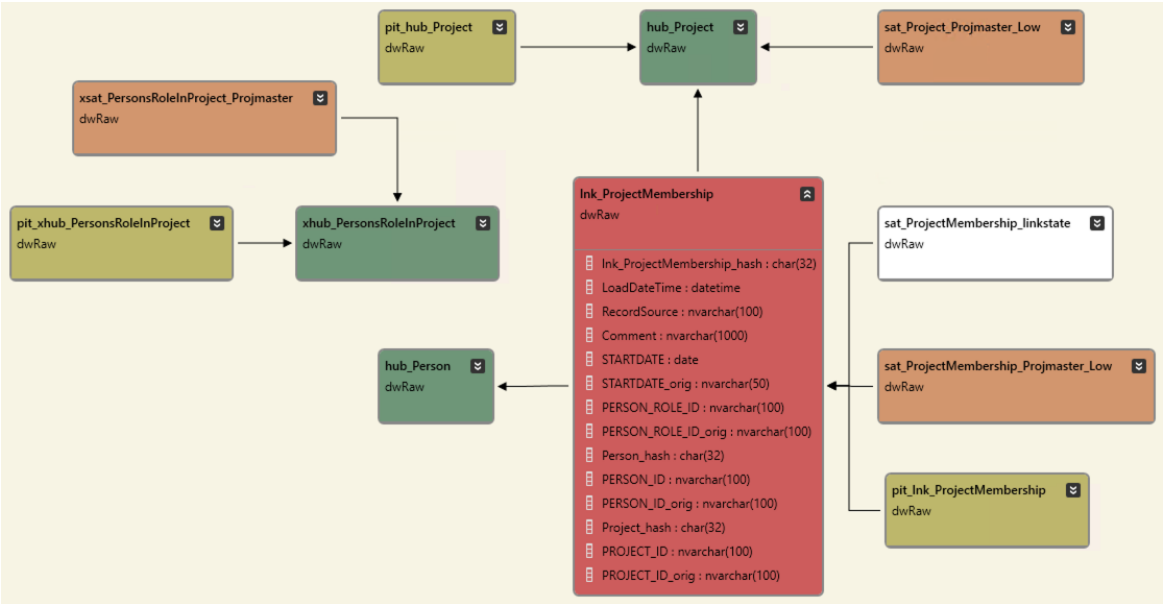
We define the typical many-to-many case of *Something* being associated with *Something else* in *Some role* at *Some specific time* (and variations thereof) as a design pattern called *Participation*, and it is typically encountered several times in any given model.



Example of Participation pattern

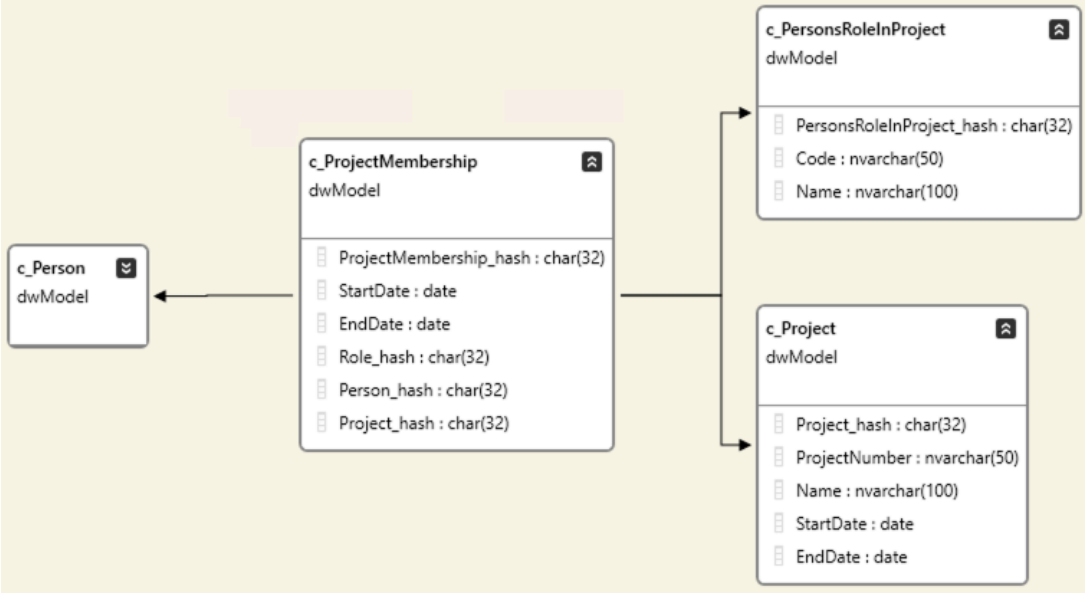
An m:n table is then an implementation of the Participation class that binds all its participants together (Project membership in the example above). Its key is defined as a non-business key containing the relevant participants plus any additional relevant attributes, so the class is represented by a link table (either with state tracking turned on or off) which directly implements the many-to-many case.

Turning on State tracking results in a state tracking satellite attaching itself to the link, and a procedure to populate it. With State tracking activated, Soft delete may additionally be applied in cases where retired link rows need to be hidden from the view.



Generated structure

In the example above, the reference to the “lookup table” hub representing *Person’s role in project* is a soft unhashed reference from PERSON_ROLE_ID that is hashed in the view layer.



View layer publishing the raw vault content

DSharp Studio does not allow link-to-link situations. Should a need arise later to reference the Project member class, this can be done by creating a new class that as its key has a the same, or a superset of, the Project member class key, after which the references in the view layer can be made using these key parts.

An alternative way to technically enable references to the Project member class is to change its key type to Business key after the fact and then refactor the existing data.

We are looking into enabling direct references between classes in the view layer while keeping the underlying table structure free from link-to-link references.

Also see *(Historized) Reference table*.