Multiple entities

1 Entity Decks

Entity Decks will be developed as a container which contains combined data of all defined singular decks.

As a child of entity Decks you have already included the multiple (select one of more) entity Deck (see Developing a ship design process).

The entity Decks can contain one or more Deck entities, each containing the same parameters and relations. However, the user can provide different input values for every Deck.

During the computation, the user will be asked the number of Deck entities he/she wants to include. If you include the Nr parameter in the "container" entity Decks, this will be the parameter which determines the number of Deck entities that will be placed, because Nr contains an **@NRINST** attribute in the *Data*, as explained in Some handy attributes.

- Add the following parameters in the **Knowledge Browser**:

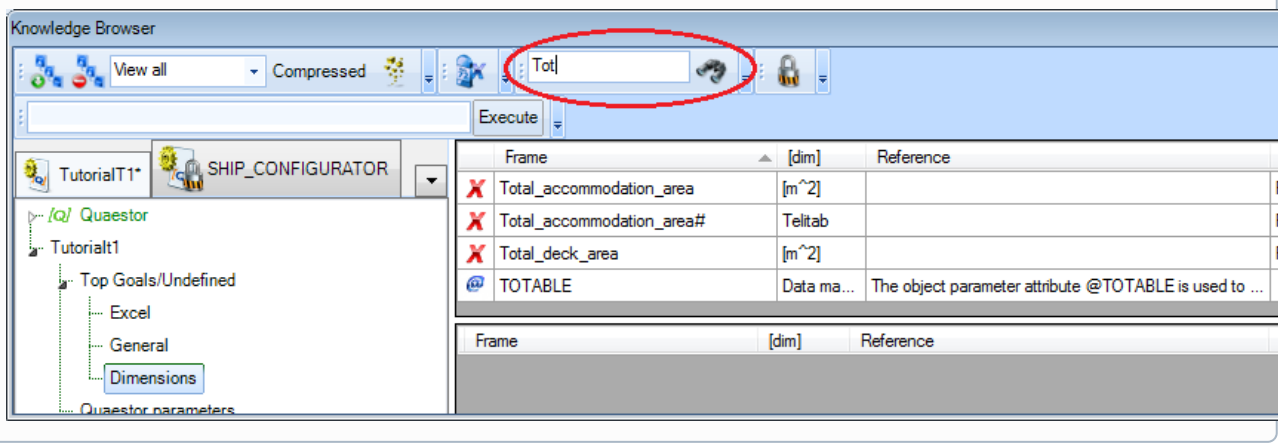
Parameter name	*Dimension*	*Determined by*	*Reference*	*In Class*
Total_deck_area	[m^2]	*USR: User or system/equation*	Total deck area	Dimensions
Deck_data#	[Telitab]	*USL: User or system/function*	Table of deck data	Dimensions
Total_accommodation_area	[m^2]	*USR: User or system/equation*	Total accommodation deck area	Dimensions
Total_accommodation_area#	[Telitab]	*USL: User or system/function*	Table of all accommodation deck data	Dimensions

Please note that "#" behind a parameter name automatically indicates that it is of type Telitab, which mean that this parameter can contain Text, a List or a Table. For more detailed information about a TeLiTab see TeLiTab. It is used here to create a Table.

- Include the following parameters in entity Decks: Nr, Total_deck_area, Deck_data#, Total_accommodation_area, Total_accommodation_area#, QEntityRef.

Tip

If you forgot in which class you put a parameter, use the seach box at the top of the Knowledge Browser to find it for you.



Relations for these parameters will be included at a later stage, because these will be clearer to you when you first have developed the entity Deck.

2 Entity Deck

- Create a new class: Mass calculation under Top goals/Undefined. This class will be used later on.
- Add the following parameters in the **Knowledge Browser**:

Parameter name	*Dimension*	*Determined by*	*Reference*	*In Class*

B	[m]	*VR: User only*	Width	Dimensions
Area	[m^2]	*USR: User or system /equation*	Area	Dimensions
Deck_function$	[Str]	*VR: User only*	Define function of deck Accommodation<EQ> Cargo deck<EQ> RoRo<EQ> Tanktop<EQ> Other<EQ>	General
L	[m]	*VR: User only*	Length	Dimensions
Weight_area_factor	[t/m^2]	*USR: User or system /equation*	Weight factor per area	Mass calculation
X_aft	[m]	*VR: User only*	Aft deck position in X (longitudinal) direction	Dimensions
X_front	[m]	*VR: User only*	Front deck position in X (longitudinal) direction	Dimensions
X_aft_plane_ID	[ID]	*VR: User only*	Define aft (longitudinal) position of deck by selecting a transverse reference plane	Dimensions
X_front_plane_ID	[ID]	*VR: User only*	Define front (longitudinal) position of deck by selecting a transverse reference plane	Dimensions
Z_plane_ID	[ID]	*VR: User only*	Define Z (vertical) position of deck by selecting a horizontal reference plane	Dimensions

- Include the following parameters in entity `Deck`: `Name$`, `Area`, `B`, `Deck_function$`, `L`, `Weight_area_factor`, `X_aft`, `X_aft_plane_ID`, `X_front`, `X_front_plane_ID`, `Z`, and `Z_plane_ID`.
- Add the following relations (an explanation will follow):

```
B = ENTITY#(12).Boa
```
(12 is the value of `QEntityId` of entity `MainDimensions`)

```
Area = L*B
```

```
L = X_front - X_aft
```

- Set the attribute @SHOW on `QEntityData`

All parameters in entity `Deck` should be in **list view** and not in t**able view**, because all values are single values. So, parameters `Z` and `Name$` in entity `Deck` are automatically placed in the **table view** because you have set a **@MULTVAL** attribute on these parameters earlier.

- Localize ("instantiate") these parameters in entity `Deck` and remove the **@MULTVAL** attribute locally.

A second option would be to set **@NOMULTVAL** on parameter `QEntityData` of entity `Deck`. Now all **@MULTVAL** attributes within this entity will be ignored.

The user (ship designer) has to indicate the starting position and end position of a deck in longitudinal direction of the ship, with the parameters `X_aft` and `X_front`. This determines the length `L` of the ship.

Furthermore, it is assumed that the width `B` of a deck is equal to the width over all **Boa** of entity `MainDimensions`. To assume rectangular decks the area is calculated by `L*B`.

You might wonder why there is a `Weight_factor_area` parameter. This is explained later on, but the main reason is that this parameter is a property of the `Deck` and as such should be part of the `Deck` entity. However, hereafter you will discover that the input for this value should not be given in this entity, but as part of the `Mass calculation` entity. In order not to show the parameter in this entity:

- Localize ("instantiate") the parameter `Weight_factor_area`
- Set the **@HIDE** attribute on it.

We will come back to this last parameter in Mass calculation.

How to connect the start and end position to the reference planes will be discussed next.

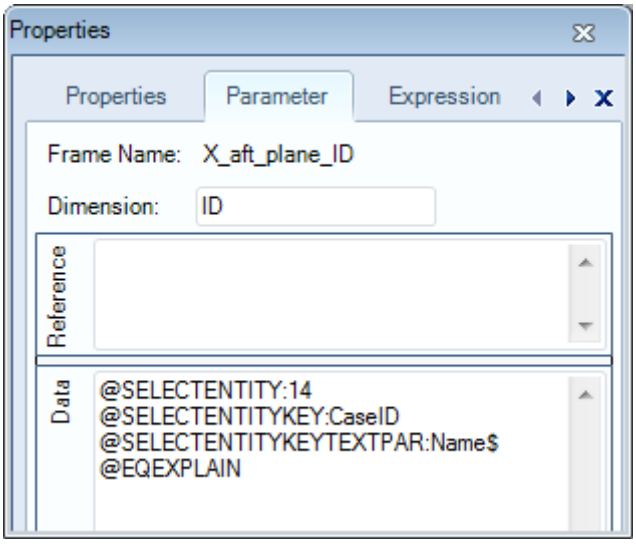
3 Create a selection list

Now, you will create a selection list from available data to position a deck with respect to specified reference planes.

The way to do this will be explained by determining the aft position of a deck.

- Add the following attributes to parameter `X_aft_plane_ID:`
- **@SELECTENTITY**`:14` (QEntityID of `Transverse planes`).The entity `Transverse planes` is used to create a selection list from with the other attributes.

- **@SELECTENTITYKEY**:`CaseID`. The parameter `CaseID` is the parameter (of entity `Transverse planes`) whose values must be listed for selection.
- **@SELECTENTITYKEYTEXTPAR:**`Name$`. The parameter `Name$` defines the case description that must be displayed in the selection list.
- **@EQEXPLAIN**. This results in the display of description rather than value, e.g. in a combobox.



Note that you may have a different value than 14 here**,** because it depends on the sequence of creating entities in the tree!

By including the attributes as described above, the user can select a reference plane from a drop down list, containing the names of all defined transverse reference planes. The result of the selection is a value of parameter `CaseID`, but the value of parameter `Name$` is shown to the user instead.

The value of parameter `X_aft` should be the value of `X` from the selected transverse plane.

- Add the following relation:

```
X_aft = ENTITY#(14).X.X_aft_plane_ID
```

This means the following: entity `Transverse planes` (in this example 14) contains a table of transverse planes, in which each column (case) represents a transverse plane. When the user has selected the second name from the table, the value of `X_aft_ID` will be 2 (although `Name$` was presented to the user). So, the value of `X_aft` will become the second "**X**" value from the table within entity `Transverse planes` (`X_aft = ENTITY#(14).X.2`)

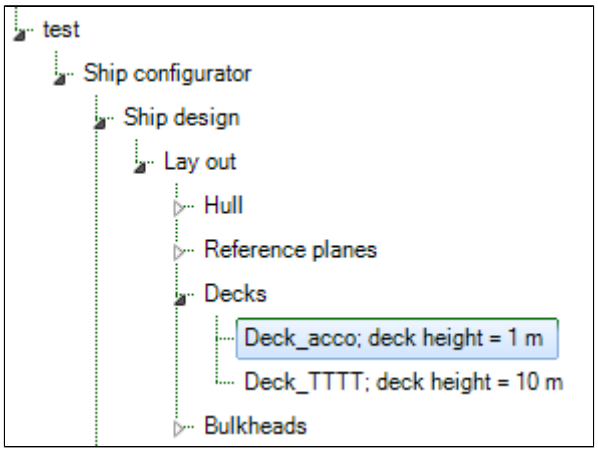
- Include exactly the same attributes for parameter `X_front_plane_ID` as you did for `X_aft_plane_ID`.
- Add the following relation:

```
X_front = ENTITY#(14).X.X_front_plane_ID
```

- Do the same for `Z_plane_ID.` Here, you have to refer to the `QEntityID` of entity `Horizontal planes`, in our example 15.
- Create the following relation:

```
Z = ENTITY#(15).Z.Z_plane_ID
```

4 Provide a name for a multiple entity



As shown in the calculation result above, each `Deck` entity node name contains the name and height of a deck. This can be accomplished by using the attribute **@OBJECTTITLE** in parameter **QEntityData** of entity **Deck**. You can provide a flexible string, for example:

```
@OBJECTTITLE:"Deck_" + Name$ + "; deck height = " + STR$(Z) + " m"
```

Here, everything between quotes will be presented as text. The value of a string parameter like `Name$` will also be shown as text. And, if you also want to present the value of a parameter which is not of the string type, you first have to convert it to a string, for example **STR$(Z)**.

- Add the **@OBJECTTITLE** attribute to `QEntityData`. For this, double click on the parameter value. The content will open in a larger editor window.

5 Combine data from child entities

You now will create a table with a subset of parameter values of all defined decks.

The **`QEntity()`** expression collects parameters of all child entities.

- Create the following relation in entity `Decks`:

```
Deck_data# = QEntity(@Name$, @Deck_function$, @Z, @X_aft, @X_front, @Area)
```

The total accommodation area will be shown separately. For this, parameter "`Total_accommodation_area#`, which is a Telitab (as its name ends with #). It should only contain data of decks for which `Deck_function$` returns `Accommodation`. Use the **QUERY#** function, which returns a Telitab subset on the basis of a set of search criteria.

- Create the following relation in entity `Decks`:

```
Total_accommodation_area# = QUERY#(Deck_data#, "NullString", "Accommodation":"Deck_function$")
```

Next sums of the datasets will be made using the **SUM** function.

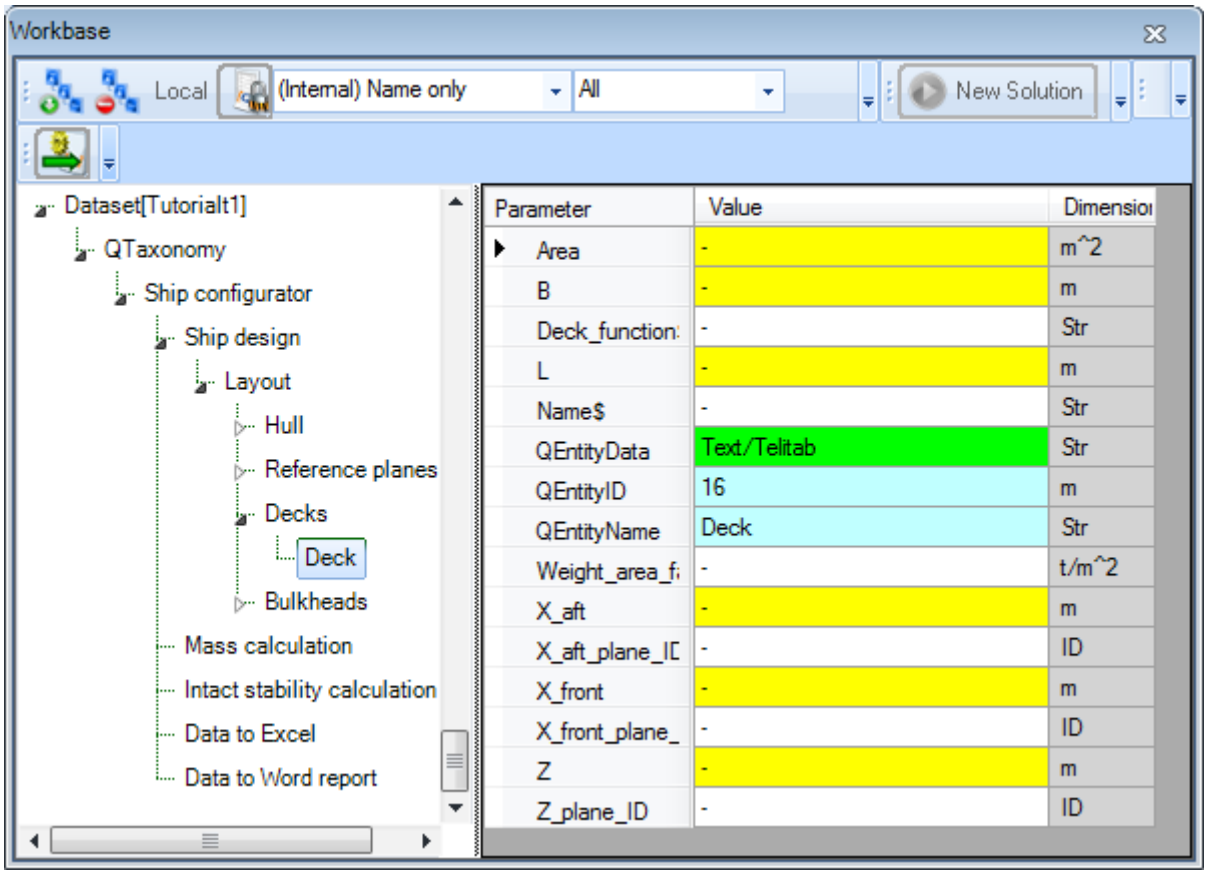
- Create the following relations:

```
Total_deck_area = SUM(Deck_data#, 1, "Area")

Total_accommodation_area = INCASE(Total_accommodation_area# = "0" + Qcrlf, THEN, 0, ELSE, SUM
(Total_accommodation_area#, 1, "Area"))
```

The second relation has a condition (the **`INCASE()`** function). If **`Total_accommodation_area#`** is an empty table (which is possible if the user does not create decks with `"Deck_function$" = "Accommodation"`) then the total accommodation deck area is 0 [m^2]. Because the content of an empty table in Quaestor will be: "0" + Qcrlf (in which Qcrlf is a Carriage return-line feed string constant) this should be the value to test the parameter against.

- Add the `QEntityRef` parameter to `Decks`
- Assign the value "`Combined data of all decks`" to `QEntityRef`
- Set the @SHOW attribute on `QEntityData`



6 Entity `Bulkheads`

As mentioned in Developing a ship design process, this ship configurator uses a different entity structure for defining (transverse) bulkheads in comparison with defining decks. Of course the same entity structure could be used, but it is more instructive to present (and develop) a different approach.

The development of the `Bulkheads` entity is comparable to the `Decks` entity. However, in the previous paragraphs a multiple entity was used to enable the user to define one or more decks. Here an entity is developed where the user can create one table to define one or more transverse bulkheads instead of several `Deck` entities. Contrary to `Deck` (child of `Decks`), which is multiple, the child entity `Bulkheads` (child of `Bulkheads`) is singular.

- Create the following parameters in the **Knowledge Browser**:

Parameter name	*Dimension*	*Determined by*	*Reference*	*In Class*
`H`	[m]	*USR: User or system /equation*	Height	`Dimensions`
`X_plane_ID`	[ID]	*VR: User only*	Define X position of bulkhead by selecting a transverse reference plane	`Dimensions`
`Z_bottom`	[m]	*VR: User only*	Bottom position bulkhead in Z (vertical) direction	`Dimensions`
`Z_top`	[m]	*VR: User only*	Top position bulkhead in Z (vertical) direction	`Dimensions`
`Z_bottom_plane_ID`	[ID]	*VR: User only*	Define bottom Z (vertical) position of bulkhead by selecting a horizontal reference plane	`Dimensions`
`Z_top_plane_ID`	[ID]	*VR: User only*	Define top Z (vertical) position of bulkhead by selecting a horizontal reference plane	`Dimensions`

- Include the following parameters in last child entity `Bulkheads: Nr, Name$, Area, B, H, Weight_area_factor, X, X_plane_ID, Z_bottom, Z_bottom_plane_ID, Z_top`, and `Z_top_plane_ID`.

With exception of parameter `Nr` and the `QEntity*` parameters all parameters must be placed within the **table view**.

- Localize ("instantiate") the relevant parameters and set the attributes **@MULTVAL**.
- Set attribute **@SHOW** on `QEntityData`

Again, a selection list must be created from data in the reference entities to, in this case, position a bulkhead. In section 3 it *was* explained how to achieve this for decks.

- Use this method for the new parameters `X_plane_ID`, `Z_bottom_plane_ID`, and `Z_top_plane_ID`.
- Next create the following relations:

```
Area = B*H
```

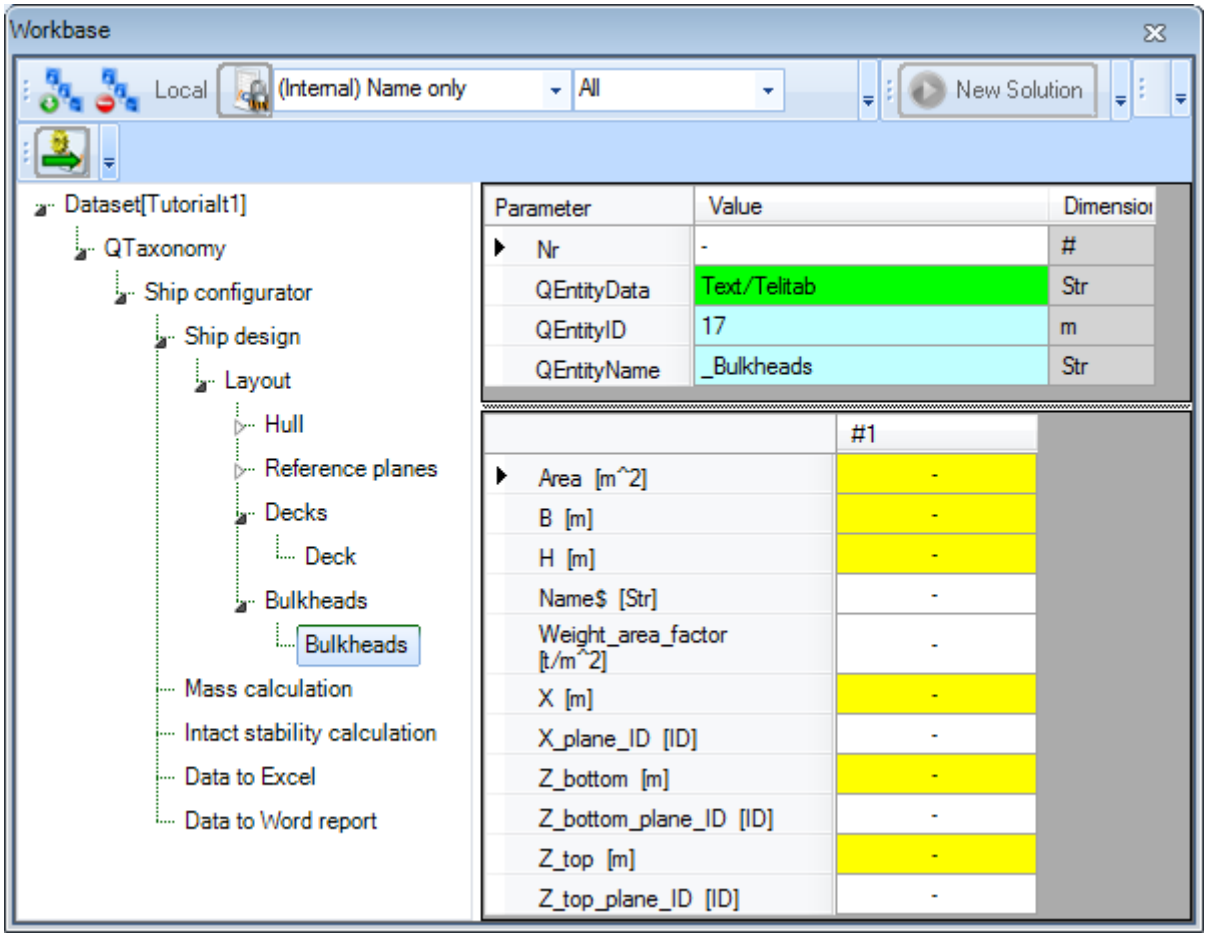
```
B = ENTITY#(12).Boa (12 is the value of QEntityId of entity MainDimensions)
```

```
H = Z_top - Z_bottom
```

```
X = ENTITY#(14).X.X_plane_ID (14 is the value of QEntityId of entity Transverse planes)
```

```
Z_bottom = ENTITY#(15).Z.Z_bottom_plane_ID (15 is the value of QEntityId of entity Horizontal planes)
```

```
Z_top = ENTITY#(15).Z.Z_top_plane_ID (15 is the value of QEntityId of entity Horizontal planes)
```



1. Back to content | << Previous | Next >>