Master/Detail the kbmMW way

for kbmMW v. 1.xx

Master/detail relations are often used in business applications.

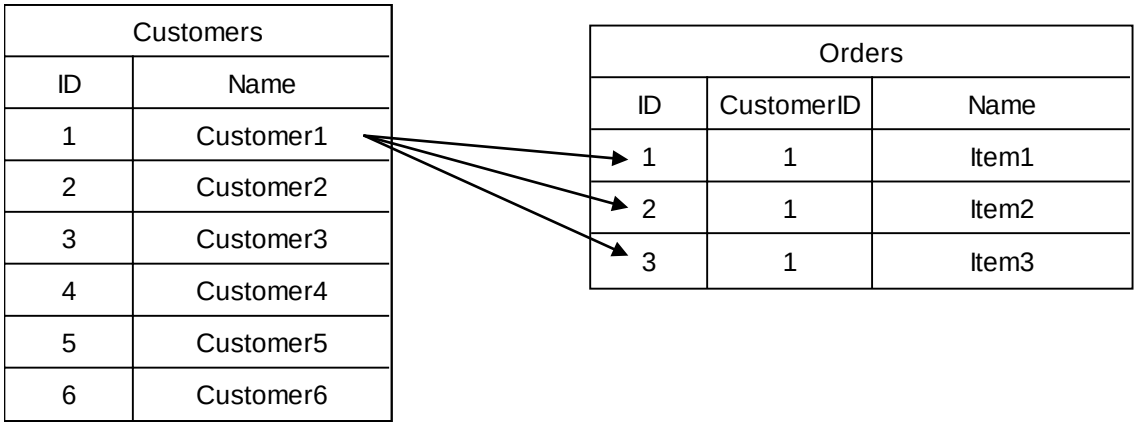
An example of it can be a customer/orders M/D relationship where you have a table of customers and another table of ordered items and you need to show which ordered items have been ordered by a particular customer.

Customers	
ID	Name
1	Customer1
2	Customer2
3	Customer3
4	Customer4
5	Customer5
6	Customer6

Orders		
ID	CustomerID	Name
1	1	Item1
2	1	Item2
3	1	Item3
4	2	Item4
5	3	Item5
6	3	Item6
7	4	Item7
8	4	Item8
9	5	Item9
10	6	Item10

This type of relation is easily setup in kbmMW.
There are two primary ways of doing a master/detail relation with kbmMW – full detail or partial detail.

Partial detail



This is the way that resembles standard M/D setups the most. When the current master record, the database will be queried for a new set of detail records matching exactly that master record.
If used on the client, it means that each time the master record is changed (eg. moving to another record in a grid etc.), the client will have to re-query the application server for detail records matching the new master record.

Advantages:

- You will usually always have up to date detail information.
- Only the absolutely needed detail records are transferred which is good if its not the norm for the application to need detail records for most of the master records.

Disadvantages:

- It require a call to the application server or database each time a navigation happens on the master table. This can be helped by enabling caching on the detail table, in which case the data will be retrieved from the cache if available.

How to set a M/D relationship up this way?

Create a master query component. Eg. qMaster which could be a TkbmMWClientQuery on the client side or any other TkbmMWxxxxQuery on the server side.

Create a detail query component. Eg. qDetail which again could be either a TkbmMWClientQuery or TkbmMWxxxxQuery depending on if it's a client or server side M/D.

Set the following properties:

- qMaster.Query.Text:='select * from customers';
- qDetail.Query.Text:='select * from orders where CustomerID=:CustomerID';
- Define the CustomerID parameter on qDetail to be an ftInteger type and parameter type ptInput via the Params property.
- Add a TDatasource, eg. dsMaster, and connect its Dataset property to qMaster.
- qDetail.MasterSource:=dsMaster;
- qDetail.MasterFields:='ID'; // This is the unique key field on the master table for which detail records should be searched. More fields can be used by separating then with ; (semicolon)
- qDetail.DetailFields:='CustomerID'; // This is the field on the detail table that should be searched to find detail records matching a specific master record. Again more fields can be specified by separating the field names with ;.
- qDetail.RequeryDetails:=true; // This controls that the detail data will be refetched each time the master record changes.

Full detail

Customers	
ID	Name
1	Customer1
2	Customer2
3	Customer3
4	Customer4
5	Customer5
6	Customer6

Orders		
ID	CustomerID	Name
1	1	Item1
2	1	Item2
3	1	Item3
4	2	Item4
5	3	Item5
6	3	Item6
7	4	Item7
8	4	Item8
9	5	Item9
10	6	Item10

This is an alternative way of doing M/D. Instead of querying only for the detail records matching the current master record, the developer is responsible for once and for all query for all detail records matching all master records. kbmMW will then internally automatically filter out the ones that should not be visible for the current master record.

Advantages:

- After the all the detail records have been fetched, browsing through the master table is very fast. No additional requests for data are needed from the application server or database.
- It allows for briefcase master/detail relations. That means that you can store the detail table in a local file, disconnect from the network and your master/detail relation will still work.

Disadvantages:

- All detail records matching all master records needs to be fetched on open.
- You may not at all times have the latest copy of a detail record.

How to set a M/D relationship up this way?

Create a master query component. Eg. qMaster which could be a TkbmMWClientQuery on the client side or any other TkbmMWxxxxQuery on the server side.

Create a detail query component. Eg. qDetail which again could be either a TkbmMWClientQuery or TkbmMWxxxxQuery depending on if it's a client or server side M/D.

Set the following properties:

- qMaster.Query.Text:='select * from customers';
- qDetail.Query.Text:='select * from orders';
- Add a TDatasource, eg. dsMaster, and connect its Dataset property to qMaster.
- qDetail.MasterSource:=dsMaster;
- qDetail.MasterFields:='ID'; // This is the unique key field on the master table for which detail records should be searched. More fields can be used by separating then with ; (semicolon)
- qDetail.DetailFields:='CustomerID'; // This is the field on the detail table that should be searched to find detail records matching a specific master record. Again more fields can be specified by separating the field names with ;.
- qDetail.RequeryDetails:=false; // This controls that the detail data will be filtered from the contents in the detail table each time the master record changes.

This concludes the whitepaper about M/D the kbmMW way.

Kim Madsen
Components4Developers