

# camos Develop Developer training

Basics ORM

**camos.**

## Prerequisites

- Carconfigurator on the state 3. day modeler training according to training documents
- DSN „DataCarconfigurator“ on database „OfferData.mdb“
- **Note:**
  - If you have edited the chapter „Basics SQL“ before this exercise, some methods and Wasele that will be created in this chapter are already present

## Prerequisites

- **Contents**
  - Form element ORM-table
    - Supplier administration
  - Access to values of the ORM-table
    - WinSelectionGet()
  - Reading and writing via ORM
    - Read and edit supplier
  - Additional exercise ORMWrite and ORMDelete
    - New supplier and delete supplier

## Training targets

- **After these exercises you should...**
  - Name the advantages of the use of ORM
  - Define an ORM
  - Display data from the database on the form
  - Read in data from the database to the application

## Explanation of terms

- **What does object-relational mapping mean?**
  - **ObjectRelational Mapping** describes the mapping of object-oriented data on relational data
  - Programming is object-oriented, but saving the data is mostly carried out in relational databases
  - ORM in camos.Develop maps features of objects on table columns of a relational database
  - SQL-statements are generated automatically during runtime


# ORM

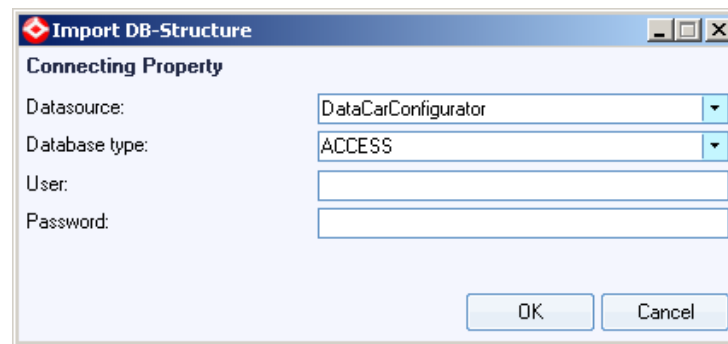
- **Usage**
  - ORMRead()
    - reads in values from the DB to the ORM-features
  - ORMWrite()
    - writes the values of the ORM-features to the DB
  - ORMDelete()
    - deletes values from the DB
  - The form element ORM-table
    - displays contents of the DB

## Exercise: Form element ORM-table

- **Target**
  - Administration environment in which the suppliers/car dealerships are maintained
- **Note**
  - Suppliers are deposited in the table „Supplier“ of the Access database „OfferData.mdb“
- **Form element ORM-table**
  - Via the ORM-table the contents of any table columns of a database can be displayed

## Exercise: Form element ORM-table

- **Create ORM-alias**
  - Click on the icon  in the toolbar in order to open the ORM-dialog
  - Create a new ORM and allocate the alias „DataCar“
- **Import database in ORM-alias**
  - Context menu item „Import DB-Structure“

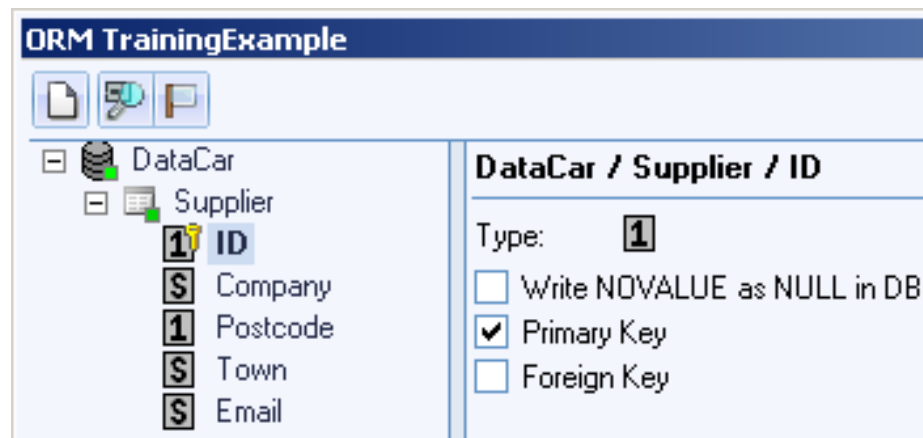


- Select the table „Supplier“ in the dialog „Preselection“



## Exercise: Form element ORM-table

- **Determine primary key column**
  - With Access databases the information which column is the primary key cannot be read out
    - Set the property „Primary Key“ on the column „ID“



- Release the ORM-alias and the table „Supplier“

## Exercise: Form element ORM-table

- Create the form „SupplierAdministration“ in „start“
  - Create an ORM-table
    - ORM-alias: „DataCar“
  - Create the following numerical ORM-columns
    - ORM-column: ID, disable „Visible“
    - ORM-column: Postcode
  - Create the following ORM-columns with the data type String
    - ORM-column: Company
    - ORM-column: Town
    - ORM-column: Email
- Name and order of the columns can be defined as you like



## Exercise: Form element ORM-table

- **Create a pushbutton**

- Text: „Close“
- Selection trigger:

```
WinClose(WinGetHandle());
```

- **Preparation for the database connect**

- Create features ODBCHnd and ORMHnd (numerical)
- Create method DBConnect() and call it in new():

```
IF DBConnect() THEN  
    WinOpen('MainForm');  
ENDIF;
```

## Exercise: Form element ORM-table

- The method **DBConnect()** establishes the connection to the DB and then to the ORM:

```
# Establish connection to the database
ODBCHnd := SQLConnect('DSN=DataCarConfigurator');
# With unsuccessful Connect ->
# Display error message and return 0
IF ODBCHnd THEN
    # Establish connection to the ORM
    ORMHnd := ORMConnect(ODBCHnd, 'DataCar');
    RETURN 1;
ELSE
    WinMessage('ERROR', GetLastError());
    RETURN 0;
ENDIF;
```

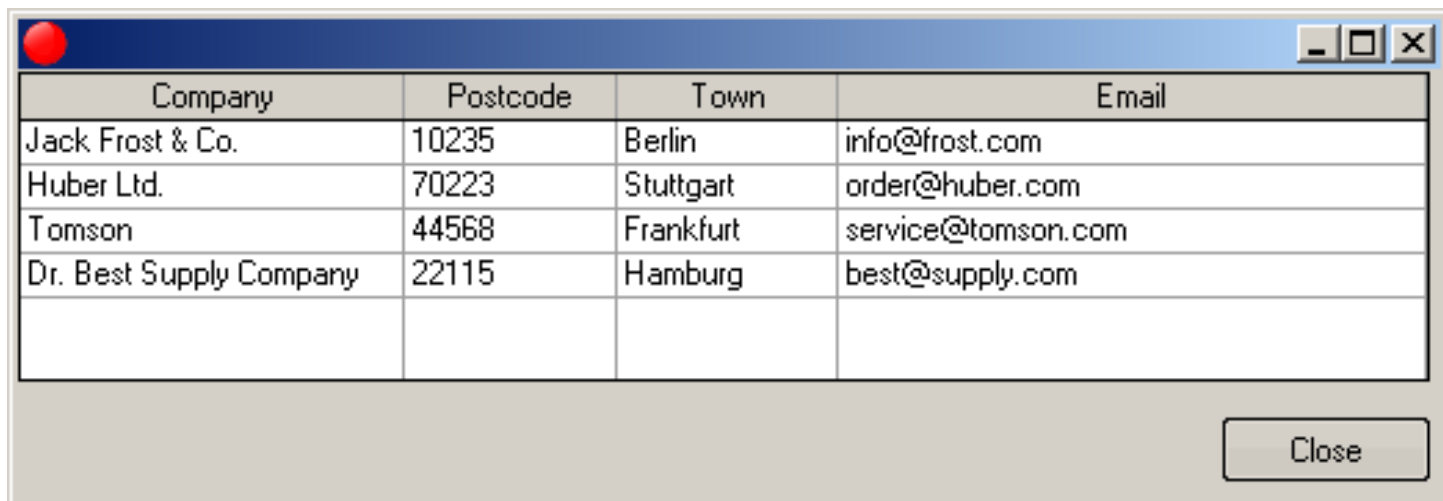
## Exercise: Form element ORM-table

- **Extend the menu „Administration“**

- Create menu title „Extras“
- Create numerical feature „AdministrationHnd“
- Create menu trigger „Administration car dealer“

```
AdministrationHnd := WinOpen('SupplierAdministration');  
WinStartModal(AadministrationHnd);
```

- Start the application and test the ORM-table!



Company	Postcode	Town	Email
Jack Frost & Co.	10235	Berlin	info@frost.com
Huber Ltd.	70223	Stuttgart	order@huber.com
Tomson	44568	Frankfurt	service@tomson.com
Dr. Best Supply Company	22115	Hamburg	best@supply.com

Close

## Exercise: Form element ORM-table

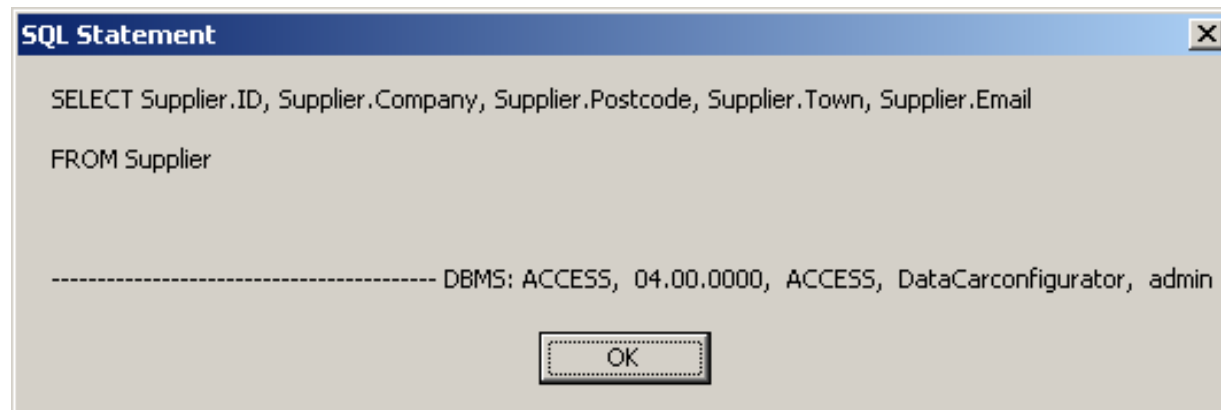
- **Exit DB-Connect**

- Create the method Delete() in „start“ in order to disconnect the database connection with exiting the application:

`SQLDisconnect ( ODBCInd ) ;`

- **Tip for the ORM-table**

- The SQL-statement that was generated from the ORM-table can be queried via WinWidgetGet(,SQLStatement') or via the key combination Ctrl+Alt+Shift+F2



## Exercise: Access to selection

- **Target of the exercise**
  - The information for the dealer that is marked in the ORM-table has to be displayed in a form
  - This data is edited in the next chapter
- **Problem**
  - In contrast to the form element DB-table no Selected feature can be deposited on ORM-table columns
  - > How can I get the values of the marked table line?

## Exercise: Access to selection

- **Solution**
  - Via WinSelectionGet() the values of the currently marked line in the ORM-table can be read out
- **Attention frequent error source**
  - WinSelectionGet() only works with ORM-columns that are either primary key columns or columns on which the option „Selection access“ was enabled in the form editor!



## Exercise: Access to selection

- **New method „DetermineDealerID()“:**
  - String variable Sel[], return type numerical
  - Disable side effects (optional)

```
Sel[] := WinSelectionGet(AdministrationHnd, 'ORM_Table', 'Supplier.ID');  
RETURN String2Num(Sel[1]);
```

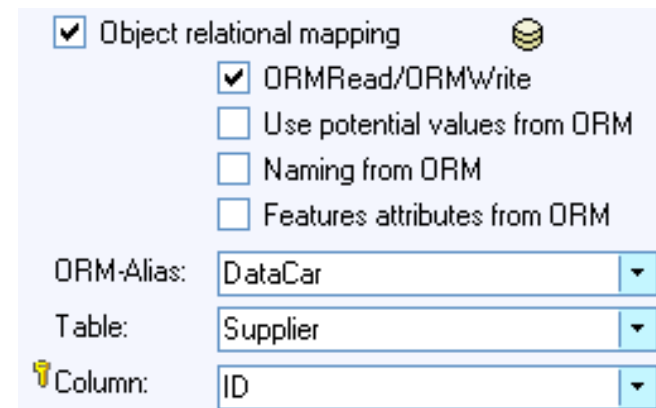
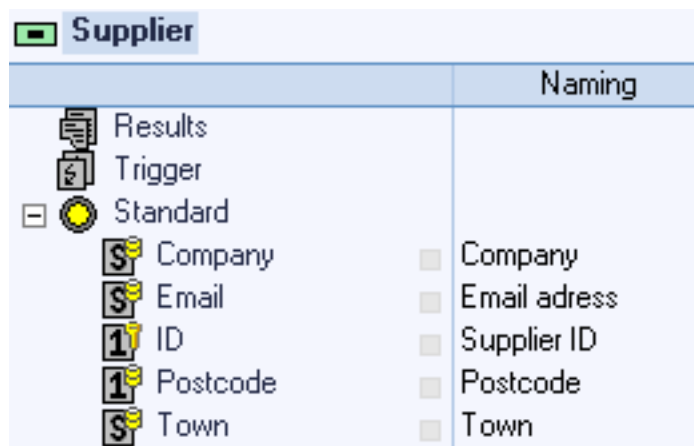
- The method returns the ID of the marked dealer
  - The last parameter specifies the name of the DB-column whose value has to be read out. Several columns can also be specified as list, e.g. {,Supplier.ID', ,Supplier.City'}
  - With multiple selection the returned list contains the queried columns per flagged line, e.g. {2, Stuttgart, 8, Hamburg}

## Exercise: ORMRead

- **Target: Read in supplier data**
  - Via the determined ID of the marked supplier the further information (Name, City etc.) can now be read in
- **Procedure via ORM**
  - The statements (Insert, Update, Delete) are generated automatically due to the link between feature and DB-table column
- **Advantages**
  - SQL-statements do not have to be formulated by yourself
  - ORM-functions can read and write complete object structures with one function call

## Exercise: ORMRead

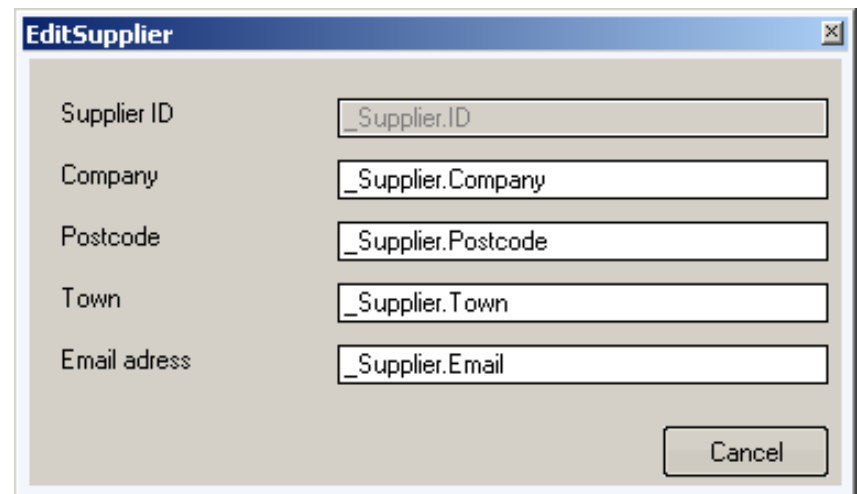
- **Create ORM-features**
  - ORM-features are required in order to establish the link between features and database columns
- **Create the object class „Supplier“ under the root**
  - Create ORM-features via D&D of the columns from the ORM-dialog to the structure tree
    - Maintain the namings of the features



## Exercise: ORMRead

- Create the component `_Supplier` in „start“
- Create the form „EditSupplier“
  - Create editlines and naming labels for the features Company, ZIP-code, City and Email
  - Readonly or disabled editline for feature ID
- Pushbutton „Cancel“

```
WinClose(WinGetHandle()) ;
```



The screenshot shows a Windows-style dialog box titled "EditSupplier". It contains five rows of labels and text input fields. The labels are "Supplier ID", "Company", "Postcode", "Town", and "Email adress". The corresponding text fields contain the text "\_Supplier.ID", "\_Supplier.Company", "\_Supplier.Postcode", "\_Supplier.Town", and "\_Supplier.Email". A "Cancel" button is located in the bottom right corner of the dialog box.

## Exercise: ORMRead

- **Create the method OpenDealer()**
  - Parameter: SelID\_ (numerical)

```
_Supplier := 1;  
_Supplier.ID := SelID_  
# Read in data of the dealer with the transferred ID  
ORMRead('S', _Supplier);  
# Open form „EditSupplier“ modal  
WinStartModal(WinOpen('EditSupplier'));  
# Delete supplier object  
_Supplier := NOVALUE;
```
  - SelID\_ contains the ID of the dealer that has to be read
  - ORMRead() reads the data of the dealer to the ORM-features in the object \_Supplier

## Exercise: ORMRead

- **Open supplier via double-click**
  - Deposit the following procedure code in the Double click trigger of the ORM-table in the form „SupplierAdministration“:

```
OpenDealer ( DetermineDealerID ( ) ) ;
```

- **Test the opening of the marked dealer**

The screenshot displays a software interface with a table of suppliers and a modal form for editing a selected supplier.

**Supplier Table:**

Company	Postcode	Town	Email
Jack Frost & Co.	10235	Berlin	info@frost.com
Huber Ltd.	70223	Stuttgart	order@huber.com
Tomson	44568		
Dr. Best Supply Company	22115		

**Supplier Administration Form (Modal):**

Supplier ID: 2

Company: Huber Ltd.

Postcode: 70223

Town: Stuttgart

Email adress: order@huber.com

Buttons: Cancel

## Exercise: ORMWrite

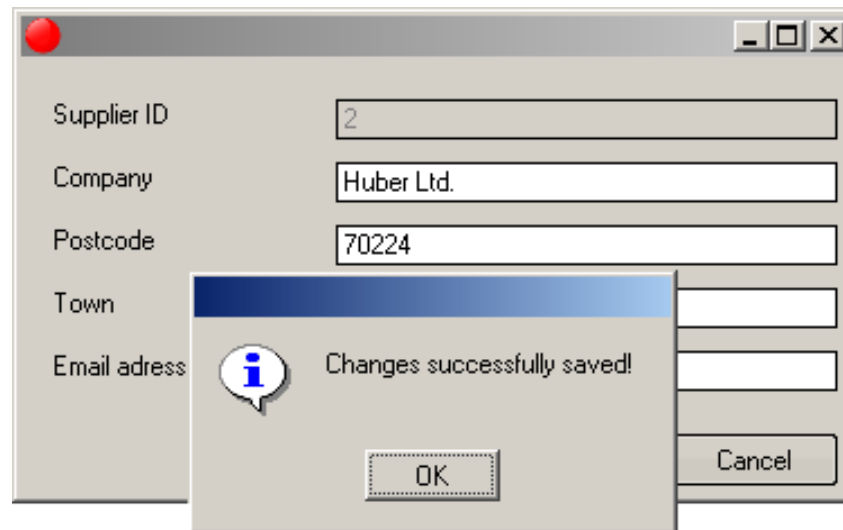
- **Target**
  - Editing the read data of the dealer and writing it back to the database
  - Using ORMWrite() which generates the matching UPDATE statement analog to ORMRead()

- **Pushbutton „Save“ in form „EditSupplier“**

```
IF ORMWrite('S', _Supplier) THEN
    WinMessage('INFO', 'Changes successfully saved!');
    WinClose(WinGetHandle());
ELSE
    WinMessage('ERROR', GetLastError());
ENDIF;
```

## Exercise: ORMWrite

- **Test the saving of the changes**
  - Saving is successful, but change is not immediately visible in the ORM-table
  - Add update to the method `OpenDealer()`:  
`WinRefresh(AdministrationHnd, 'ORM_Table');`





## Additional exercise: ORMWrite

- **Target**
  - It should also be possible to create new dealers in the administration environment
- **Note**
  - 90% of this request is already supported by the implemented opening and editing of a dealer
- **Problem**
  - Determining a new ID, because an INSERT is not possible without primary key value(s)!
    - Dealer IDs are consecutive. The new ID has to be calculated on the basis of all present supplier IDs

## Additional exercise: ORMWrite

- **Procedure for determining the next dealer ID**
  - Read in all present suppliers
  - Write IDs of the read-in suppliers to list
  - Determine highest value of this list and add 1
- **Note:**
  - Normally a GUID is used as primary key
  - In this case the above mentioned problem would not exist, because a new unique ID can be generated via CreateGUID()
- **Create a list component of the class „Supplier“ in „start“**
  - `_AllSuppliers[]`

## Additional exercise: ORMWrite

- Create the method **GenerateDealerID()**
  - Create local variables i, iMax und IDList[] (num)

```
# Read in all suppliers
IF ORMRead('S', _AllSuppliers[]) THEN
    iMax := MaxIndex(_AllSuppliers[]);
    FOR i := 1 TO iMax DO
        # Write all supplier IDs to the list IDList[]
        IDList[i] := _AllSuppliers[i].ID;
    ENDFOR;
    # Delete supplier objects
    _AllSuppliers[] := NOVALUE;
    # Determine highest value of the list + add 1
    RETURN Max(IDList[]) + 1;
ELSE
    WinMessage('ERROR', GetLastError());
    RETURN NOVALUE;
ENDIF;
```

## Additional exercise: ORMWrite

- **Create the method CreateDealer()**

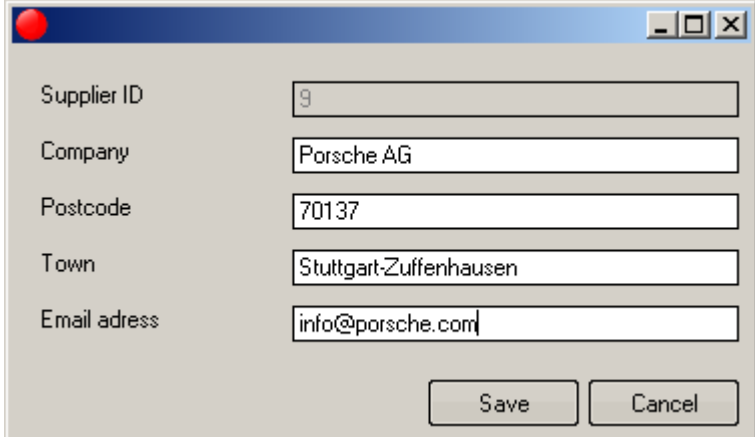
- Parameter NewID\_ (numerical)

```
# Instantiate new supplier object
_Supplier := 1;
# Assign the transferred ID to the feature ID
_Supplier.ID := NewID_;
WinStartModal(WinOpen('EditSupplier'));
# Reset component
_Supplier := NOVALUE;
# Update ORM-table
WinRefresh(AdministrationHnd, 'ORM_Table');
```

- **New pushbutton „New...”  
in the „SupplierAdministration”**

```
CreateDealer(GenerateDealerID());
```

- **Test the creating  
of a new car dealer!**



Supplier ID	9
Company	Porsche AG
Postcode	70137
Town	Stuttgart-Zuffenhausen
Email adress	info@porsche.com

Save Cancel

## Additional exercise: ORMDelete

- **Target**
  - It should be possible to delete a present supplier
- **Procedure**
  - Determine ID of the dealer that has to be deleted
  - Create supplier object and assign ID
  - Execute ORMDelete
- **New pushbutton „Delete...” in „SupplierAdministration”**

```
DeleteDealer(DetermineDealerID());
```

## Additional exercise: ORMDelete

- Create new method **DeleteDealer()**
  - Parameter SelID\_ (numerical)

```
IF SelID_ > 0 THEN
  IF WinMessage('QUESTION', 'Really delete?') = 1 THEN
    _Supplier := 1;
    _Supplier.ID := SelID_;
    #
    IF ORMDelete('S', _Supplier) THEN
      WinRefresh(AdministrationHnd, 'ORM_Table');
      WinMessage('INFO', 'Supplier successfully deleted!');
      _Supplier := NOVALUE;
    ELSE
      WinMessage('ERROR', GetLastError());
    ENDIF;
  ENDIF;
ELSE
  WinMessage('ERROR', 'No supplier marked!');
ENDIF;
```

## Tips: ORMRead/ORMWrite

- **Exclude individual objects from ORMRead / ORMWrite**
  - Properties „ORMRead Exclude“ or „ORMWrite Exclude“ in the component editor
  - During runtime via ORMReadExclude(Object, 0/1) or ORMWriteExclude(Object, 0/1)
- **Check for changed ORM-features**
  - ORMHasChanged() returns 1 if changed ORM-features were found
- **Extend Where-clause for ORMRead()**
  - ORMWhereClauseSet() adds a Where-clause to the Where-clause that was generated from ORMRead()

## Tips: Properties of the ORM-features

- **Further options can be set in the editor of an ORM-feature:**
- ORMRead/ORMWrite
  - Defines if a reading and writing access to the database column is permitted
- Value from ORM
  - In the properties of the ORM-dialog you can define value attributes for the values of a database column which are displayed instead of the value



## Tips: Properties of the ORM-features

- **In the properties of the ORM-dialog:**
- Naming from ORM
  - A text element can be defined as naming for each DB-column
  - If the option is enabled, the naming of the database column is used
- Feature attributes from ORM
  - Wasele attributes can be defined for each DB-column
  - If the option is enabled, the Wasele attributes that are defined in the ORM-dialog and their values are applied to the local feature and are available on the tab page Feature attributes