SQL Developer, Low Code and APEX Applications

Martin B. Nielsen
MBNDATA
Denmark

Keywords:
APEX, SQL Developer, Low Code, Data model, Process Model

Introduction

This session shows how to make the most of the SQL Developer when creating data models and database object, by automating part of the DDL creation (Low code). It also dives into the SQL Developer data modeler "reporting scheme" which enables us to re-use the artefacts from the model in our APEX applications.

SQL Developer model information can be exported into a reporting database schema. This information can then be providing input to the wizard generation of APEX pages, it can control navigation, provide help texts, domains and much more.

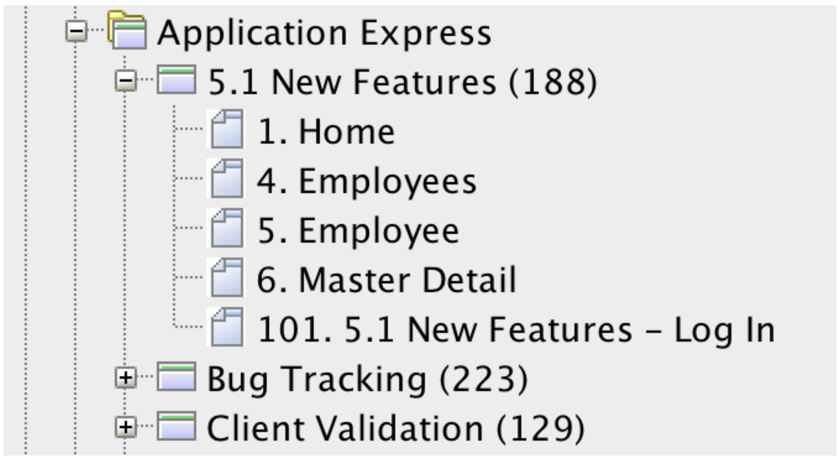
The session demos several examples of how this technique was used in a large Danish APEX project. It will also demo how easy it is to create data and process models in SQL Developer and how JavaScript (transformations) can be used to easily extend our models, make faster data models and automatically generate scripts for triggers and other database objects.

SQL Developer and the Datamodeller

The Oracle SQL developer is Oracle's tools for maintaining databases and developing SQL and PL/SQL code. The tool is actually quite extensive and contains many integrated "modules" for handling various task that we as developers need when developing APEX applications.

Apart from the standard DBA, SQL and PL/SQL development capabilities, the SQL Developer contains a browser for our APEX applications, REST integration and many other useful tools.

The Browser gives us a tree structure overview of our APEX applications and have some interactions, which will actually change the application. The capabilities can be used to get an overview of the application and useful for DBA's to enable / disable access to the application without actually logging into APEX:

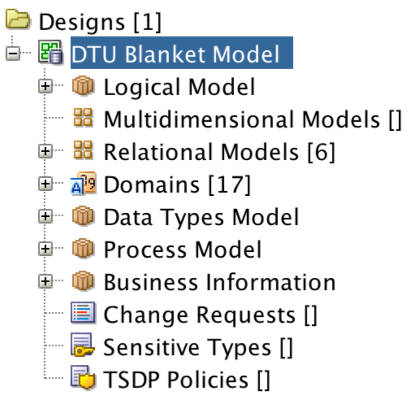


For more details about the APEX applications, I recommend looking into the APEX_xxx views which are part of the APEX data dictionary.

Historically the Data Modeller was a separate product, which could be purchased from Oracle. Over time, this has been absorbed into the SQL Developer, thereby allowing free use of the product. Many people who has "converted" to SQL Developer, from TOAD and other products are often not aware that their development tool actually contains a complete designer tool (I will not say CASE – which is not a popular word any more).

The Datamodeller is well hidden among the many menus, but can by activated and used as a visual design tool for creating process and data models. The datamodel can already be useful to generate DDL for creating our database tables, indexes and constraints.

In this session I will go into details about how the Data Modeller artifacts (our visual design models) can be put to even further use, hence provide a "Low code" development environment for our applications.

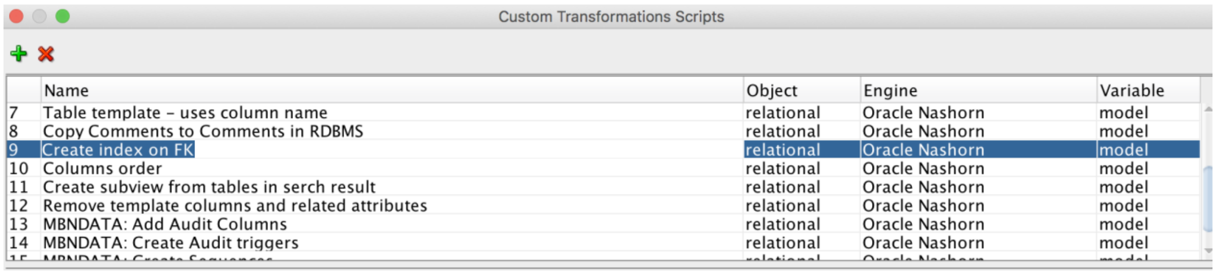


Low Code – generating code from SQL developer Datamodeller

As developers we tend to repeat out trusted development methods that we use form project to project. We can also be asked to implement and honor certain development standards during the project development. This could be anything from naming standard to index best practices (like always indexing foreign keys), having certain triggers to supply auditing information etc.

The best way to solve these tasks are by automation. The less we code (or copy/paste), the fewer errors we introduce. The Datamodeller itself has great tools for enforcing naming standard, prefixes etc. so we can focus on adding that extra functionality that our current project needs. The main tool for that, in the Data Modeller, is "Transformations".

Transformation are snippets of JavaScript code that we write once, and save inside the Data Modeller.


Custom Transformations Scripts

	Name	Object	Engine	Variable
7	Table template – uses column name	relational	Oracle Nashorn	model
8	Copy Comments to Comments in RDBMS	relational	Oracle Nashorn	model
9	Create index on FK	relational	Oracle Nashorn	model
10	Columns order	relational	Oracle Nashorn	model
11	Create subview from tables in serch result	relational	Oracle Nashorn	model
12	Remove template columns and related attributes	relational	Oracle Nashorn	model
13	MBNDATA: Add Audit Columns	relational	Oracle Nashorn	model
14	MBNDATA: Create Audit triggers	relational	Oracle Nashorn	model
15	MBNDATA: Create Sequences	relational	Oracle Nashorn	model

Transformations can change the objects in out models, write to files and much more. There are a number of pre-supplied transformation from Oracle. Ex. "Create index on FK", which will walk through the current datamodel and add indexes on all un-indexed foreign keys.

```
function getIndex(tab,cols){
 keys = tab.getKeys();
 for (var i = 0; i < keys.length; i++) {
  index = keys[i];
  if(!(index.isPK() || index.isUnique()) && !index.isFK() &&
index.isIndexForColumns(cols)){
     return index
    }
 }
 return null;
}

tables = model.getTableSet().toArray();
for (var t = 0; t<tables.length;t++){
 table = tables[t];
 indexes = table.getKeys();
 for (var i = 0; i < indexes.length; i++) {
    index = indexes[i];
    if(index.isFK()){
     columns = index.getColumns();
     if(columns.length>0){
       newIndex = getIndex(table,columns);
       if(newIndex==null){
         newIndex = table.createIndex()
         table.setDirty(true);
         for (var k = 0; k < columns.length; k++){
           newIndex.add(columns[k]);
         }
       }
     }
   }
 }
}
```

This snippet is making it quite fast for us to make sure we minimize locking issues and improve performance on our tables. The indexes are only implemented in the visual model, not the database (for that we need to generate DDL from the model).

The customer Transformations can also be used for generating other objects such as sequences, triggers, PL/SQL packages etc. This can be achieved by traversing the model and creating new files based on the tables in the model.

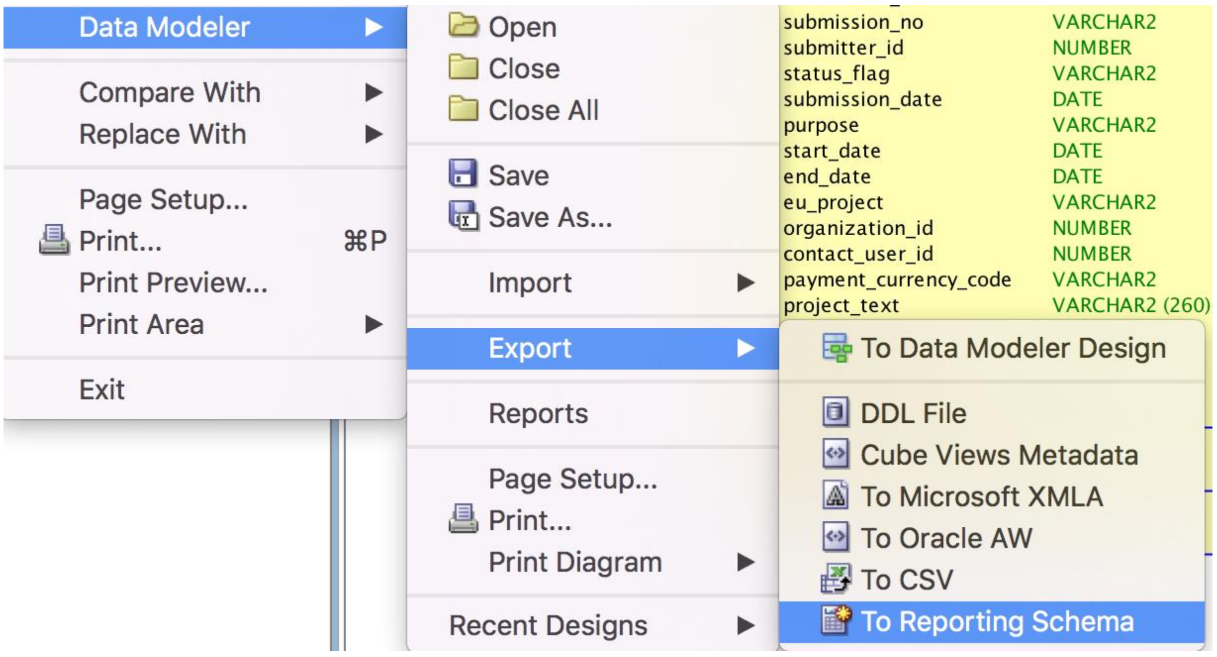
Here is an example of a custom transformation for creating sequences for all the tables in the model:

```
outFile                                  =                                  new
java.io.FileWriter("/Users/martin/SVN/DTU/XXSUB/ddl/create_sequences
_per_diem.sql");
out = new java.io.PrintWriter(outFile);
out.print("-- ------------------------------------------------");
out.println();
out.print("-- Title  : Sequences");
out.println();
out.print("-- Created: " + new Date() );
out.println();
out.print("-- ------------------------------------------------");
out.println();
out.println();

tables = model.getTableSet().toArray();

for (var t = 0; t<tables.length;t++)
{
  table = tables[t];
  tableName = table.getName();

  out.println("CREATE SEQUENCE " + tableName + "_S START WITH 1
INCREMENT BY 1 NOCACHE;");
}

out.close();
```

The SQL Developer reporting schema

The great strength of the Data Modeller is that any artifact we create in the model can be exported to a so called “Reposting scheme”. Collections of models (our projects) are saved as a “Designs” which are a file based storage (consisting of lots of files in a directory, many of them XML files).

This is not so useful for us as database developers, so in order to use the information to something exiting, we need to export this to a datamodel (the reposting scheme):



submission_no	VARCHAR2
submitter_id	NUMBER
status_flag	VARCHAR2
submission_date	DATE
purpose	VARCHAR2
start_date	DATE
end_date	DATE
eu_project	VARCHAR2
organization_id	NUMBER
contact_user_id	NUMBER
payment_currency_code	VARCHAR2
project_text	VARCHAR2 (260)

Once our designs are in the reporting scheme we can use them for our APEX applications.

The reposting schema consists of a number of tables, prefixed with DMRS_ and views prefixed by DBRV_x: Ex.:

```
DMRS_VIEW_ORDER_GROUPBY
DMRS_VIEW_CONTAINERS
DMRS_VIEW_COLUMNS
DMRS_VALUE_RANGES
DMRS_TRANSFORMATION_TASK_INFOS
DMRS_TRANSFORMATION_TASKS
DMRS_TRANSFORMATION_PACKAGES
DMRS_TRANSFORMATION_FLOWS
DMRS_TRANSFORMATIONS
DMRS_TELEPHONES
DMRS_TASK_PARAMS_ITEMS
DMRS_TASK_PARAMS
```

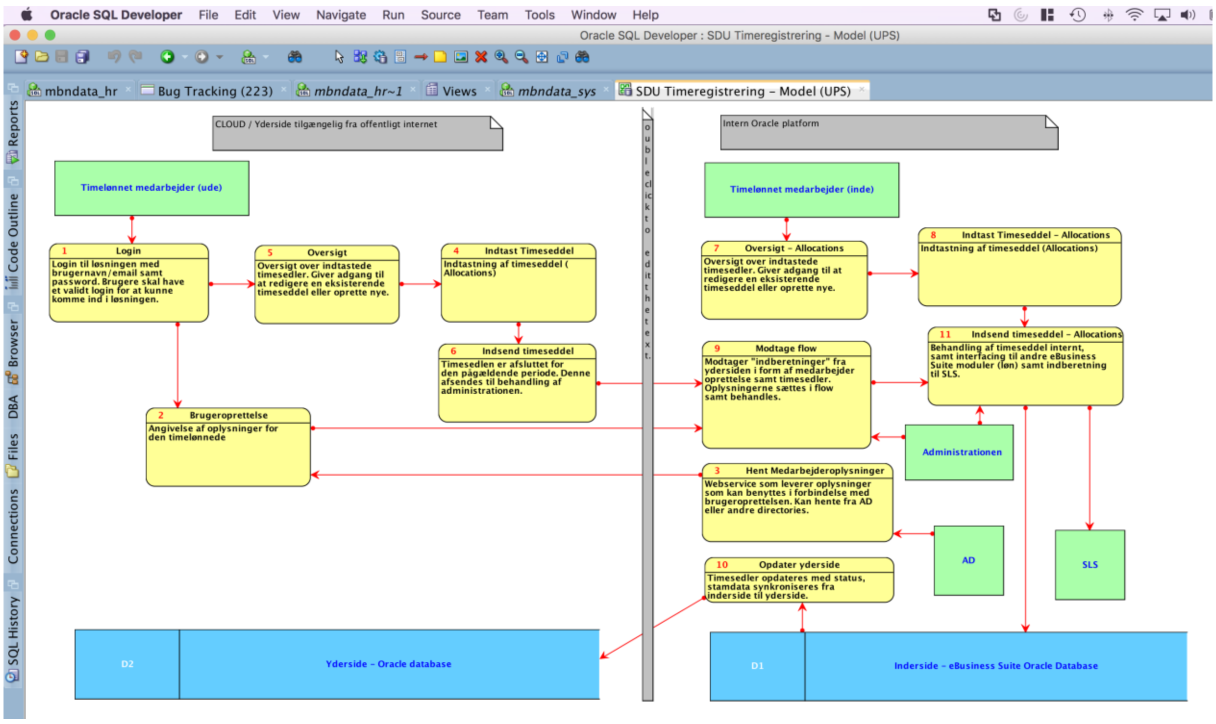
```
DMRS_TABLE_INCLUDE_SCRIPTS
DMRS_TABLE_CONSTRAINTS
DMRS_TABLEVIEWS
DMRS_TABLES
DMRS_STRUCT_TYPE_METHOD_PARS
DMRS_STRUCT_TYPE_METHODS
DMRS_STRUCT_TYPE_ATTRS
DMRS_STRUCTURED_TYPES
DMRS_SPATIAL_DIMENSIONS
DMRS_SPATIAL_COLUMN_DEFINITION
```

And

```
DMRV_DOMAINS
DMRV_RAGGED_HIER_LINK_ATTRS
DMRV_COLLECTION_TYPES
DMRV_MODEL_DISPLAYS
DMRV_STRUCTURED_TYPES
DMRV_STRUCT_TYPE_METHOD_PARS
DMRV_DOMAIN_AVT
DMRV_HIERARCHY_ROLLUP_LINKS
DMRV_RECORD_STRUCT_EXT_DATAS
DMRV_RES_PARTY_ELEMENTS
```

Integrating the process model with APEX

Now that our models are stored in database tables, we can start using the information in APEX applications. The process model contains our dataflow in the application. Each process can be a composite, consisting of many underlying processes.


Oracle SQL Developer File Edit View Navigate Run Source Team Tools Window Help
Oracle SQL Developer : SDU Timeregistrering - Model (UPS)
mbndata_hr
Bug Tracking (223)
mbndata_hr~1
Views
mbndata_sys
SDU Timeregistrering – Model (UPS)
Reports
Code Outline
Browser
DBA
Files
Connections
SQL History
CLOUD / Yderside tilgængelig fra offentligt internet
Intern Oracle platform
Timelønnet medarbejder (ude)
Timelønnet medarbejder (inde)
Administrationen
AD
SLS
D2
Yderside – Oracle database
D1
Inderside – eBusiness Suite Oracle Database
oubleclick to edit the text.

1 Login
Login til løsningen med brugernavn/email samt password. Brugere skal have et validt login for at kunne komme ind i løsningen.

5 Oversigt
Oversigt over indtastede timesedler. Giver adgang til at redigere en eksisterende timeseddel eller oprette nye.

4 Indtast Timeseddel
Indtastning af timeseddel (Allocations)

6 Indsend timeseddel
Timesedlen er afsluttet for den pågældende periode. Denne afsendes til behandling af administrationen.

2 Brugeroprettelse
Angivelse af oplysninger for den timelønnede

7 Oversigt – Allocations
Oversigt over indtastede timesedler. Giver adgang til at redigere en eksisterende timeseddel eller oprette nye.

8 Indtast Timeseddel – Allocations
Indtastning af timeseddel (Allocations)

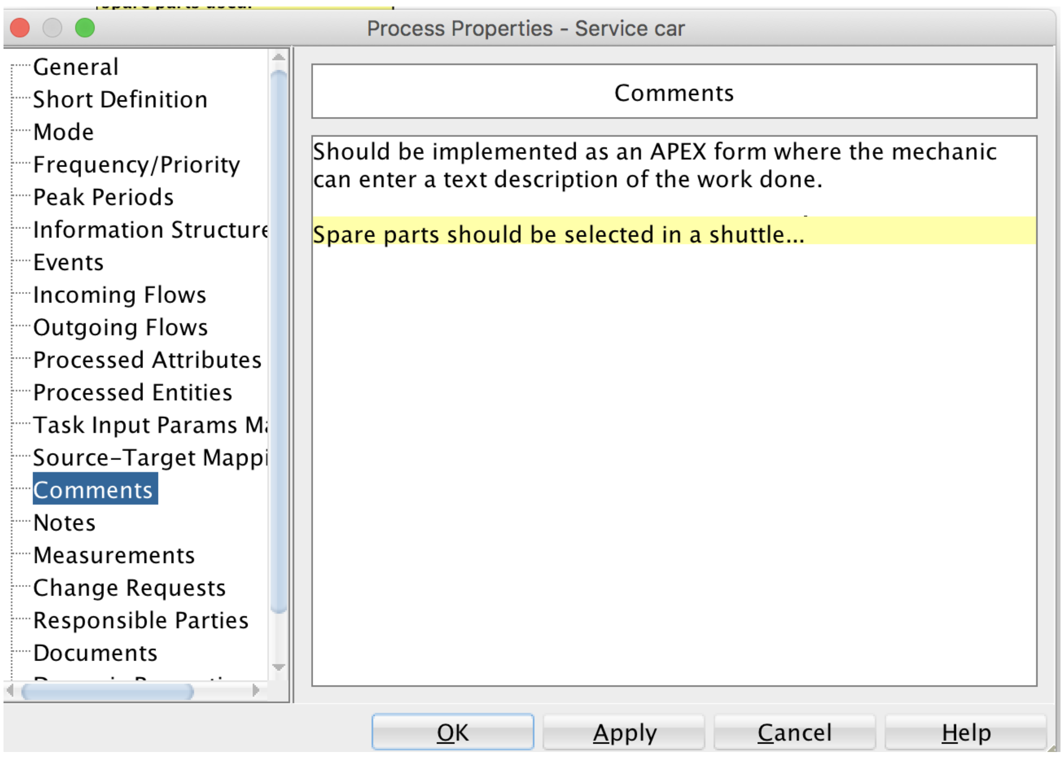
9 Modtage flow
Modtager "indberetninger" fra ydersiden i form af medarbejder oprettelse samt timesedler. Oplysningerne sættes i flow samt behandles.

11 Indsend timeseddel – Allocations
Behandling af timeseddel internt, samt interfacing til andre eBusiness Suite moduler (løn) samt indberetning til SLS.

3 Hent Medarbejderoplysninger
Webservice som leverer oplysninger som kan benyttes i forbindelse med brugeroprettelsen. Kan hente fra AD eller andre directories.

10 Opdater yderside
Timesedler opdateres med status, stamdata synkroniseres fra inderside til yderside.

Processes can be annotated by designers with all relevant information the developer needs when the process should be coded (ex. As an APEX page).



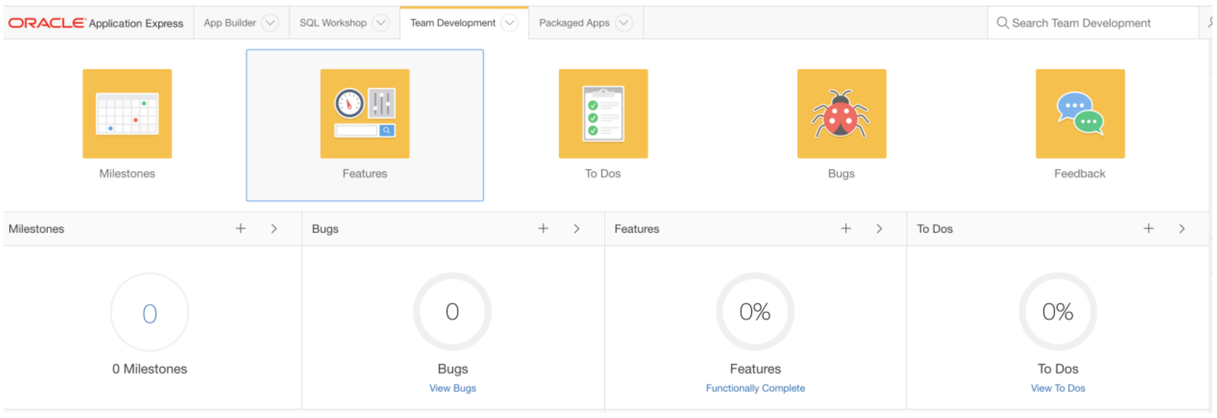
Should be implemented as an APEX form where the mechanic can enter a text description of the work done.

Spare parts should be selected in a shuttle...

Information from the process model can be used for various purposes in our application. I will demonstrate each of these usages with a live demo and code examples:

__Custom APEX Menu__
The hierarchical process model defined can be used to control the menu structure of our APEX applications. This can be achieved by creating a dynamic menu using a SQL statement, which looks in to the reporting Scheme.

__Integrating with APEX Team development__
The APEX team development module can be used to track the features which should be implemented in the new APEX application.
Processes can often be equal to “Features”, so why not create the features automatically based on the processes from out process models.



__Developer assistance__
When the developer starts working on the new feature, we can supply all the information the developer needs from the model. The information can be available in regions that are only displayed during the development. Ex. Using “Build Options”

Integrating the data model with APEX

The Data Models contains a wealth of information, which can likewise be integrated into our APEX applications. The use cases are ex.:

__User Interface Defaults__
The data model is full of good information about our columns and tables. If this information can be placed in the APEX Users Interface defaults, our applications can use this when new pages are generated (using the wizards).

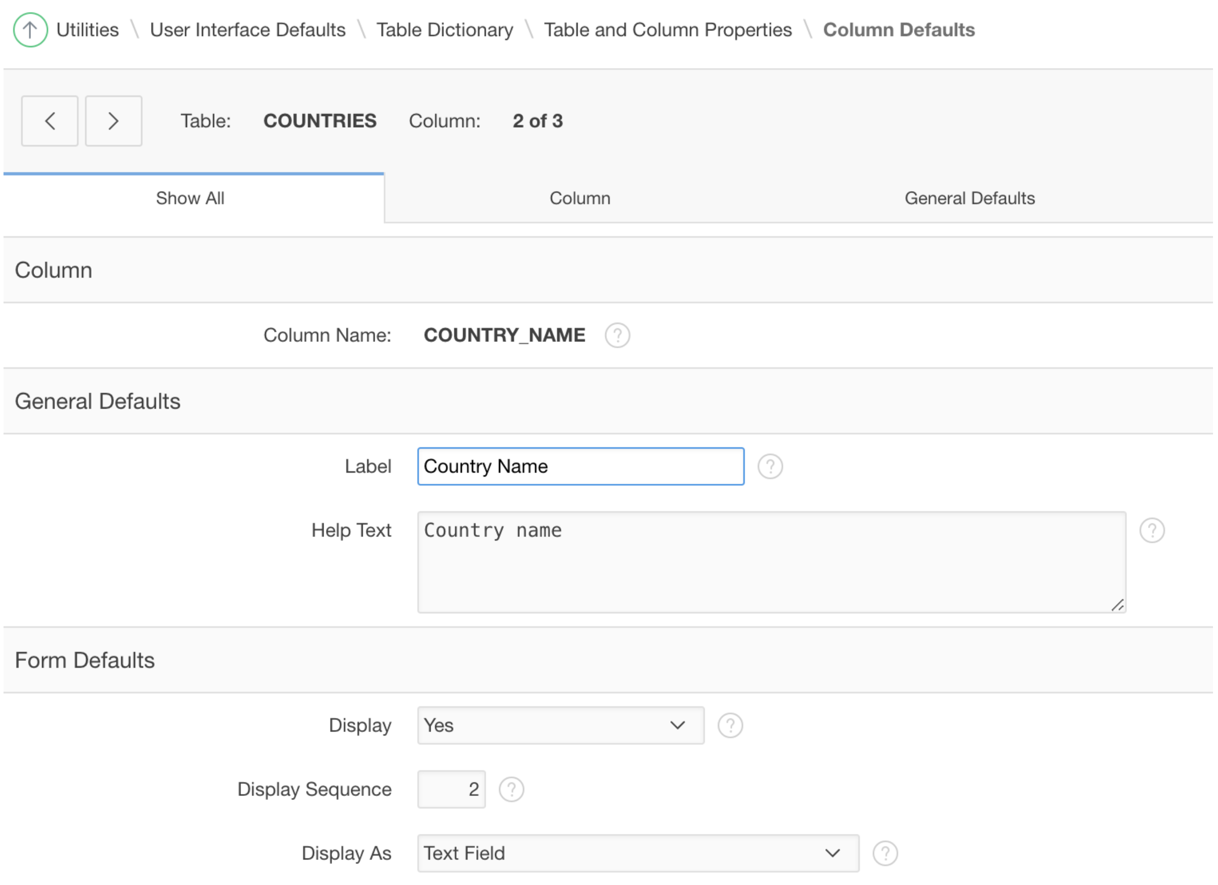

Utilities \ User Interface Defaults \ Table Dictionary \ Table and Column Properties \ Column Defaults

Table: **COUNTRIES** Column: **2 of 3**

Show All | Column | General Defaults

Column

Column Name:	COUNTRY_NAME

General Defaults

Label	Country Name
Help Text	Country name

Form Defaults

Display	Yes
Display Sequence	2
Display As	Text Field

Help text

Information about tables and columns (entered by the designer), can be very useful for the end users. So they can be used to generate help information. Ex. As popup windows displaying information about the tables and fields which are part of the current screen.

Contact information

The Models also contain information about the contact persons for the project. This is useful information which can be displayed in the APEX application. Ex. In the support page.

Best of all: by integrating the design model actively in the Application, we ensure that the model is kept up to date and will not get out of sync with the application over time. Thereby providing a valid system documentation for future supporters of the systems we develop.

Contact address:

Name
Martin B. Nielsen
MBNDATA
Amtmandsvej 2
4300 Holbæk
Denmark

Phone:	+45 40989637
Fax:	
Email	martin@mbndata.dk
Internet:	mbndata.dk