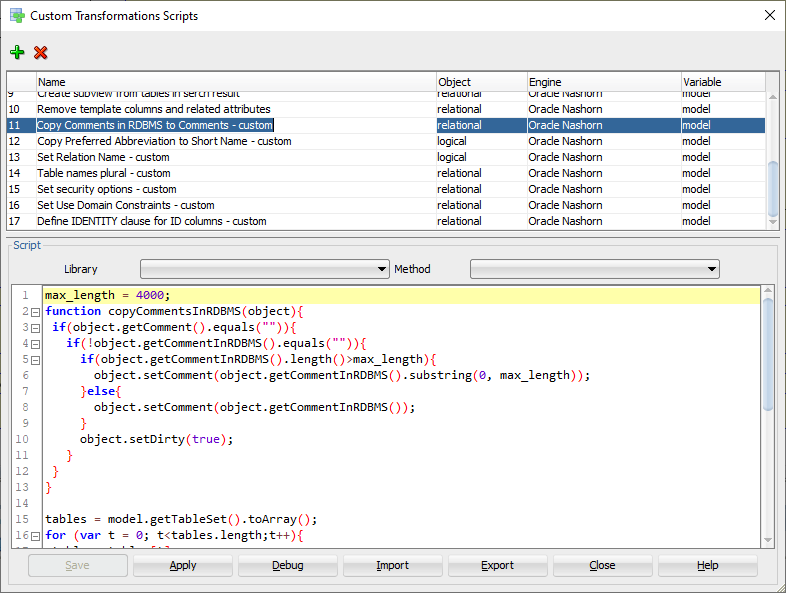
Oracle SQL Developer Data Modeler

# Custom Transformations Scripts



## Copy Comments in RDBMS to Comments - custom

max\_length = 4000;

function copyCommentsInRDBMS(object){

if(object.getComment().equals("")){

if(!object.getCommentInRDBMS().equals("")){

if(object.getCommentInRDBMS().length()>max\_length){

object.setComment(object.getCommentInRDBMS().substring(0, max\_length));

}else{

object.setComment(object.getCommentInRDBMS());

}

object.setDirty(true);

}

}

}

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

for (var t = 0; t<tables.length;t++){

table = tables[t]

copyCommentsInRDBMS(table);

columns = table.getElements();

size = table.getElementsCollection().size();

for (var i = 0; i < columns.length; i++) {

column = columns[i];

copyCommentsInRDBMS(column);

}

keys = table.getKeys();

for (var i = 0; i < keys.length; i++) {

key = keys[i];

if(!key.isFK()){

copyCommentsInRDBMS(key);

}else{

copyCommentsInRDBMS(key.getFKAssociation());

}

}

}

## Copy Preferred Abbreviation to Short Name – custom

function copyPreferredAbbreviation(object) {

if (object.getShortName().equals("")) {

var preferredAbbreviation = object.getPreferredAbbreviation();

if (!preferredAbbreviation.equals("")) {

object.setShortName(preferredAbbreviation);

object.setDirty(true);

}

}

}

entities = model.getEntitySet().toArray();

for (var e = 0; e < entities.length; e++) {

copyPreferredAbbreviation(entities[e]);

}

## Set Relation Name – custom

function setRelationName(object, name) {

if (!object.getName().equals(name)) {

object.setName(name);

object.setDirty(true);

}

}

relations = model.getRelationSet().toArray();

for (var r = 0; r < relations.length; r++) {

setRelationName(relations[r], relations[r].getSourceEntity().getPreferredAbbreviation() + '\_' + relations[r].getTargetEntity().getPreferredAbbreviation());

}

## Table names plural – custom

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

for (var t = 0; t<tables.length; t++){

var table = tables[t];

var tableName = table.getName();

if (tableName.endsWith("Y")) {

// Y -> IES

table.setName(tableName.slice(0, -1) + "IES");

table.setDirty(true);

} else if (!tableName.endsWith("S")) {

// . -> .S

table.setName(tableName + "S");

table.setDirty(true);

}

}

## Set security options – custom

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

for (var t = 0; t<tables.length;t++){

table = tables[t];

cols = table.getElements();

for(var c = 0; c < cols.length; c++){

if (cols[c].isContainsPII() != true) {

cols[c].setContainsPII(false);

}

if (cols[c].isContainsSensitiveInformation() != true) {

cols[c].setContainsSensitiveInformation(false);

}

}

table.setDirty(true);

}

## Set Use Domain Constraints – custom

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

for (var t = 0; t<tables.length;t++){

table = tables[t];

cols = table.getElements();

for(var c = 0; c < cols.length; c++){

if (cols[c].getDomain() != null &&

cols[c].getDomain().getName() != "Unknown" &&

cols[c].getUseDomainConstraints() != true) {

cols[c].setUseDomainConstraints(true);

table.setDirty(true);

}

}

}

## Define IDENTITY clause for ID columns – custom

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

for (var t = 0; t < tables.length; t++) {

table = tables[t];

cols = table.getElements();

for (var c = 0; c < cols.length; c++) {

if (cols[c].getName().equals("ID")) {

cols[c].setAutoIncrementColumn(true);

cols[c].setIdentityColumn(true);

cols[c].setAutoIncrementGenerateTrigger(false);

cols[c].setDirty(true);

// ApplicationView.log("Changed auto increment properties for column " + table.getName() + "." + cols[c].getName());

}

}

}

var clause = "IDENTITY\_CLAUSE"; // thank you SQL Data Modeler, &$^é\*"é\*^"$^é&ùù!

var tables = model.getStorageDesign().getTableProxySet().toArray();

for (var t = 0; t < tables.length; t++) {

var table = tables[t];

var cols = table.getColumns().toArray();

for (var c = 0; c < cols.length; c++) {

if (cols[c].getName().equals("ID")) {

cols[c].setAutoIncrementDDL(clause);

cols[c].setDirty(true);

}

}

}