МИНИСТЕРСТВО ОБРАЗОВНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Проектирование**

**информационных систем**

**с помощью CASE-средств**

**Отчет**

**По выполнению лабораторной работы №5**

**«Разработка генераторов и триггеров для ключевых атрибутов таблиц БД»**

Отчет составил:

Бакалавр группы ИВТВМбд-41

Захарычев Н.А.

Отчет принял:

Преподаватель кафедры ИКТРС

д.т.н Токмаков Г.П.

Ульяновск

УлГТУ

2017

1. **Цель выполнения работы**

Разработка с использованием языка макросов ERwin последовательностей, генераторов и триггеров для автоматического генерирования уникальных целых числовых значений и вставки их в поля ключевых атрибутов таблиц БД.

*Вариант 5.* Автоматизированная разработка БД АС для учета деятельности компьютерного магазина: поступление и реализация компьютеров.

1. **Описания содержания выполненных работ**
   1. **Краткие сведения о языке макросов Erwin**

Встроенный в ERwin макроязык содержит около 200 макросов, имена которых начинаются с символа «%». В текстах скриптов, создаваемых в рамках выполнения лабораторных работ, используются следующие макросы:

– %==(value1, value2) – оператор сравнения. Выдает значение 1 при совпадении значений value1 и value2 ,и 0 – при их несовпадении;

– %AttProp(nameUDP) – извлекает значение свойства c именем «nameUDP», подключенного к атрибуту;

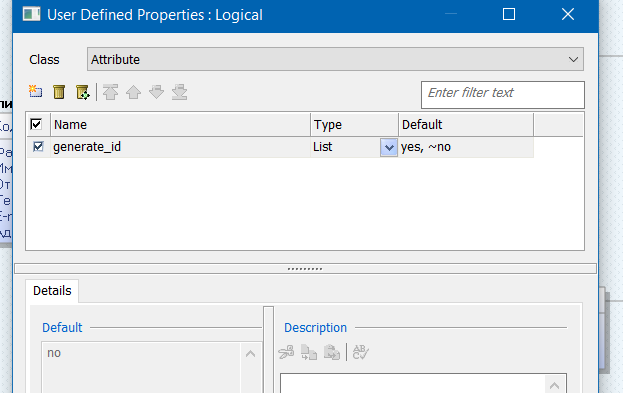
– %TableName – при раскрытии шаблона разворачивается в имя текущей таблицы;

– %ForEachAtt(%TableName) – макрос-оператор цикла. Осуществляет перебор всех атрибутов таблицы %TableName(). Например, конструкция %ForEachAtt%TableName{…} выполняет все, что находится внутри скобок {...}, для каждого атрибута таблицы %TableName;

– %if (проверка условия) –условный макрос является аналогом условного оператора if языков программирования. Например, конструкция %if (%==(%AttProp(generate\_id), yes)) {…} проверяет значение generate\_id атрибута (generate\_id – это свойство атрибута, устанавливаемое пользователем, описанное ниже), и если оно равно «yes», то выполняется все, что находится в фигурных скобках.

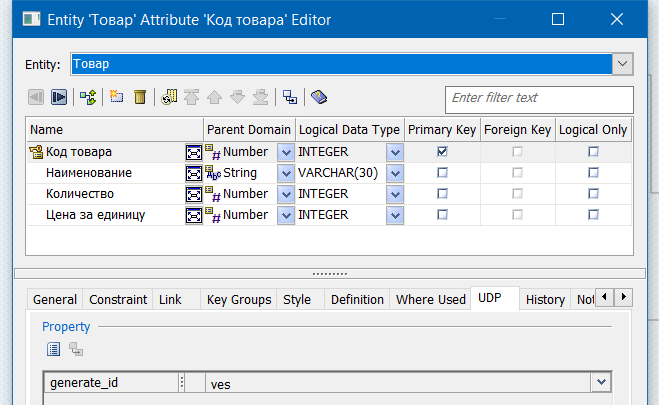
* 1. **Создание автоинкрементных ключей**

Для всех атрибутов создадим специальное пользовательское свойство (User Definition Property UDP), которое будет служить признаком генерирования значений. Для этого переходим в режим логической схемы и выбираем пункт меню Model\User Defined Properties. В диалоге свойств установим в списке Class объект Attribute, создадим свойство с именем generate\_id типа List и назначим ему допустимые значения по yes, ~no. Знак «~» определяет значение по умолчанию.



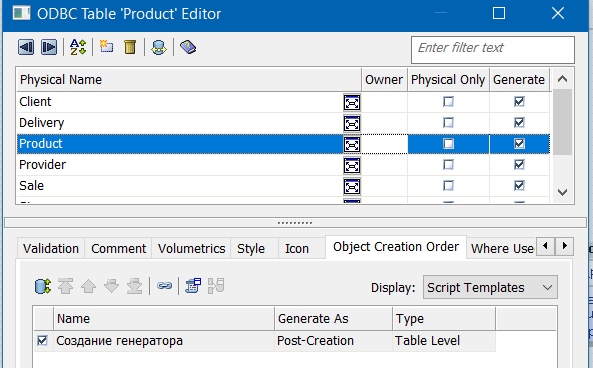
* 1. **Создание пользовательского свойства**

Пользовательское свойство будет видно у всех атрибутов модели, а значение его по умолчанию будет равно -no. Для генерируемых ключевых атрибутов поменяем это значение на -yes. С этой целью вызовем диалог Attribute Editor и на вкладке UDP поменяем значение свойства generate\_id c no на yes (на примере таблицы «Товар»).

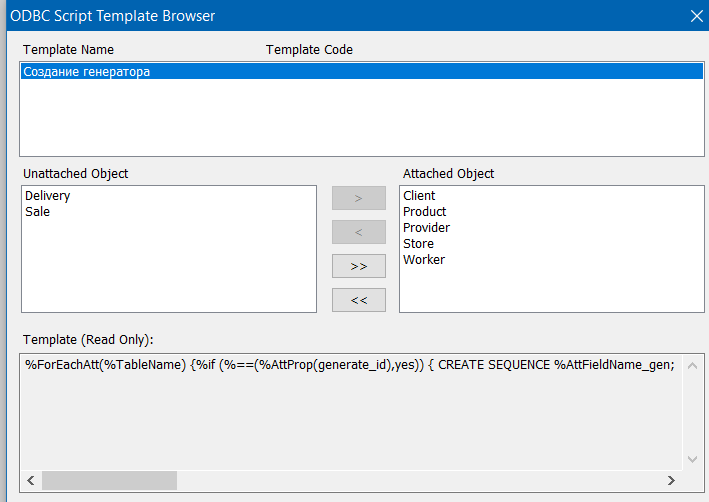


* 1. **Создание шаблона скрипта последовательности, генератора и триггера**

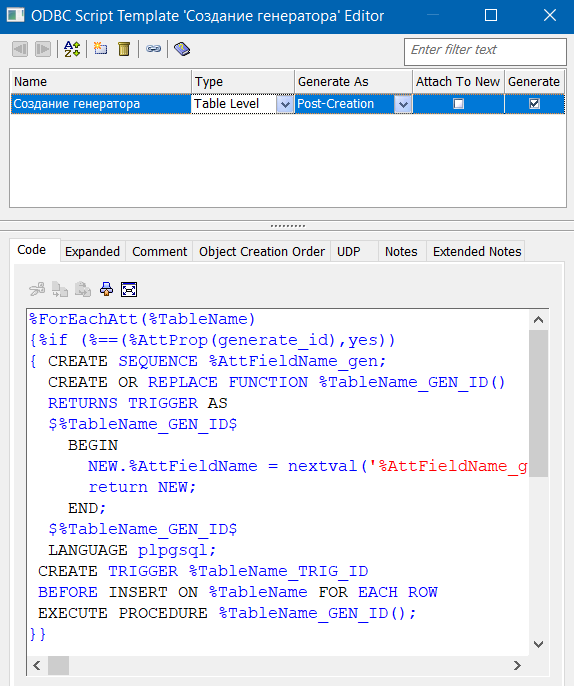
Для этого перейдем в режим физической модели, выберем Model\Tables и на панели инструментов закладки Object Creation Order нажмем кнопку Script Template Editor.



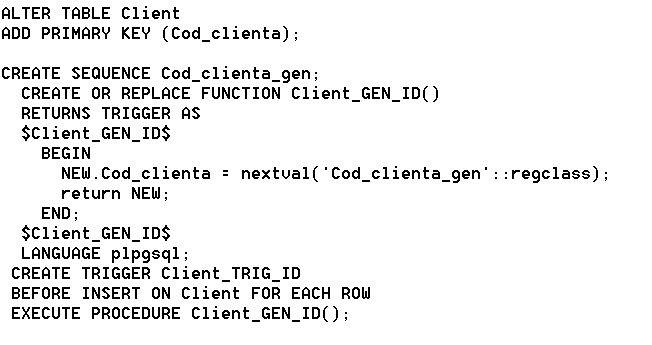
Создадим новый скрипт с названием «Создание генератора». Также в редакторе отметим именно те таблицы, к которым необходимо применить данный скрипт.



Убедимся в правильности скрипта и соблюдение типа «Table Level», а также галочки в столбце «Generate».



* 1. **Проверка результатов выполнения скриптов**



* 1. **Создание собственных триггеров для реализации процесса «Склад»**

Создадим триггер, который перед отпуском требуемого количества товара покупателю проверяет, достаточно ли для этого наличного товара на складе.

CREATE OR REPLACE FUNCTION public.check\_id\_prd()

RETURNS trigger AS

$BODY$

DECLARE prd\_count integer;

BEGIN

*1. Выборка записи из таблицы*

SELECT INTO prd\_count amount FROM store

WHERE cod\_product = NEW.cod\_product;

*2. Количество товара в таблице store меньше продаваемого*

IF prd\_count < NEW.amount

THEN RAISE EXCEPTION 'Количество наличного това-

ра %, невозможно продать % единиц товара', prd\_count,

NEW.amount;

*3. Количество товара в таблице store больше  продаваемого*

ELSE UPDATE store

*4. Уменьшение количества товара в таблице store на заданную величину*

SET amount = amount - NEW.amount

WHERE cod\_product = NEW.cod\_product;

RETURN NEW;

END IF;

END;

$BODY$

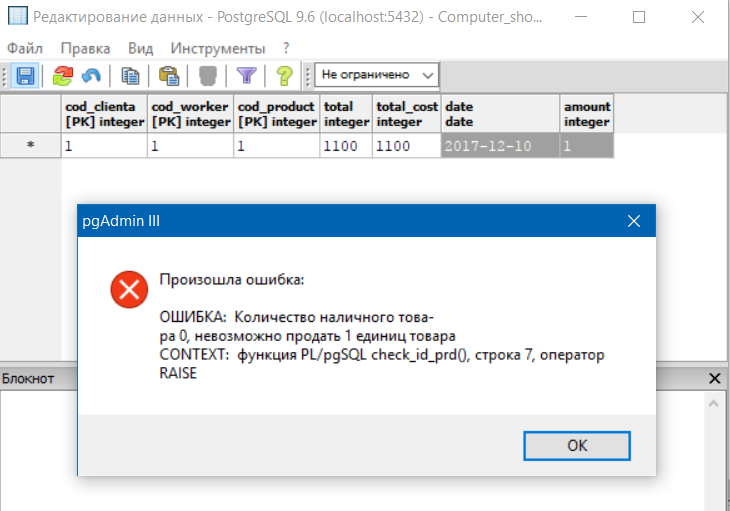
LANGUAGE plpgsql VOLATILE

COST 100;

ALTER FUNCTION public.check\_id\_prd()

OWNER TO postgres;

Если при заказе на складе не будет хватать нужного количества товара, то получим ошибку:



Если товара хватает, то при заказе число товара в таблице Store уменьшится.

Аналогичную процедуру сделаем и для добавления товара на склад. Если товара на складе не существует, то добавляем новый.

CREATE OR REPLACE FUNCTION public.check\_add\_id\_prd()

RETURNS trigger AS

$BODY$

DECLARE id\_prd integer;

BEGIN

SELECT INTO id\_prd cod\_product FROM product

WHERE cod\_product = NEW.cod\_product;

IF id\_prd IS NULL

THEN RAISE EXCEPTION ' Продукта с кодом % не существует',

NEW.cod\_product;

ELSE

SELECT INTO id\_prd cod\_product FROM store

WHERE cod\_product = NEW.cod\_product;

IF id\_prd IS NULL

THEN INSERT INTO store(cod\_product, amount)

VALUES(NEW.cod\_product, NEW.amount)

ELSE

UPDATE store

SET amount = amount + NEW.amount

WHERE cod\_product = NEW.cod\_product;

END IF;

RETURN NEW;

END;

$BODY$

LANGUAGE 'plpgsql' VOLATILE;

1. **Вывод**

В результате выполнения данной лабораторный работы, были получены навыки в создании последовательностей, генераторов и триггеров при помощи CASE-средства ERWIN, также были улучшены навыки в написании SQL скриптов.