МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1–40 05 01-03 Информационные системы и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

База данных «Магазин товаров бытовой химии» с использованием технологии применения мультимедийных технологий

Выполнил студент Хохрякова А. С.

(Ф.И.О.)

Руководитель проекта ассистент Колмаков М.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2021

# Реферат

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

Реферат

Лит.

Листов

1

БГТУ

Пояснительная записка курсового проекта содержит 39 страниц пояснительной записки, 20 иллюстраций, 7 источников литературы, 5 приложений.

При разработке использовались: ORACLE, ORACLE SQL DEVELOPER, ORACLE SQLPLUS.

Цель курсового проекта – разработка базы данных, которая обеспечивает быстрое взаимодействие между базой данных и сервером, что позволит пользователю оформить заказ, отменить заказ, а также узнать информацию об интересующих товарах, их категориях, получить полную историю своих заказов.

Вторая глава посвящена процессу проектирования системы и описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования проекта.

В четвёртой главе описано тестирование, проверка работоспособности и анализ данных системы.

В заключении приведены результаты проделанной работы.

# СОДЕРЖАНИЕ

Оглавление

[Реферат 2](#_Toc58846481)

[СОДЕРЖАНИЕ 3](#_Toc58846482)

[Введение 4](#_Toc58846483)

[1 Аналитический обзор схожих приложений 5](#_Toc58846484)

[2 Проектирование и разработка базы данных 8](#_Toc58846488)

[2.1 Спецификация функциональных требований 8](#_Toc58846489)

[2.2 Разработка модели базы данных 10](#_Toc58846490)

[2.3 Пользователи 12](#_Toc58846502)

[2.4 Хранимые процедуры 12](#_Toc58846503)

[2.4.1 Выборка данных для таблиц 12](#_Toc58846504)

[2.4.2 Удаление данных из таблиц 13](#_Toc58846505)

[2.4.3 Добавление данных в таблицы 13](#_Toc58846506)

[2.4.5 Экспорт и импорт таблиц в формат xml 13](#_Toc58846507)

[2.5 Индексы 13](#_Toc58846508)

[2.6 Пакеты 14](#_Toc58846509)

[2.7 Реализация последовательностей 16](#_Toc58846510)

[2.8 Триггеры 16](#_Toc58846511)

[2.9 Директории 17](#_Toc58846512)

[2.10 Реализация технологий 17](#_Toc58846513)

[3 Тестирование, проверка работоспособности и анализ данных 20](#_Toc58846514)

[Заключение 22](#_Toc58846515)

[Список литературы 23](#_Toc58846516)

[Приложение А 24](#_Toc58846517)

[Приложение Б 25](#_Toc58846518)

[Приложение В 29](#_Toc58846519)

[Приложение Г 31](#_Toc58846520)

[Приложение Д 37](#_Toc58846521)

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

Введение

Лит.

Листов

1

БГТУ

В**ведение**

В данной записке приведено описание базы данных, разработанной в соответствии с заданием на курсовое проектирование по теме «Магазин товаров бытовой химии» по дисциплине «Администрирование баз данных и приложений».

Задачей данной курсовой работы является разработка базы данных предназначенной для приложения «Магазин товаров бытовой». Приложение будет обладать следующими функционалом:

* Авторизация пользователей.
* Возможность добавления товаров в корзину.
* Возможность удаления товаров из корзины.
* Возможность редактирования корзины.
* Просмотр доступных товаров.
* Разграничение товаров по категориям.

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Для разработки и управления базой данных курсового проекта использовались система управления реляционными базами данных Oracle Database 12c, в связи с ее производительностью и надежностью, и графический интерфейс Oracle SQL Developer из-за интуитивной понятности и большого спектра возможности.

# Обзор технических решений и литературных источников

Для составления технических требований к проекту был проведен анализ похожих программных средств. Из открытого источника (интернет) были найдены web-ресурсы предоставляющее схожий функционал. Один из примеров – сайт ostrov-chistoty.by [1].

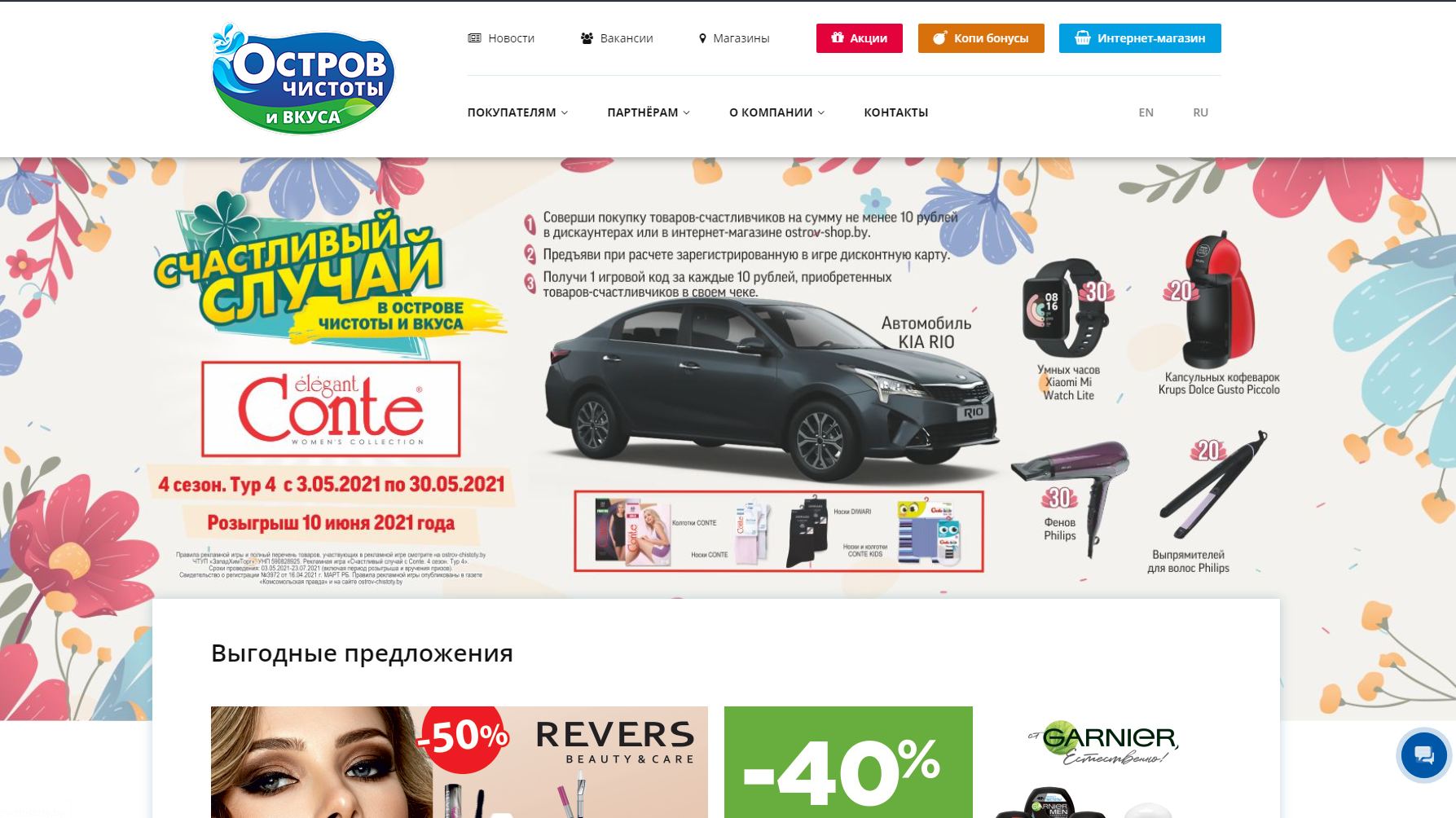


Рисунок 1.1 – Главная страница сайта ostrov-chistoty.by

Данный сайт обладает всем функционалом для облегчения выполнения всех необходимых операций, связанных с поиском товаров, добавления их в корзину и оформления заказов.

Наша задача спроектировать базу данных таким образом, чтобы весь функционал был полностью готов к задачам будущего приложения абсолютно любого магазина.

# Проектирование и разработка базы данных

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

2.Проектирование и разработка базы данных

Лит.

Листов

1

БГТУ

При разработке курсового проекта понадобились следующие объекты:

1. Таблицы.
2. Хранимые процедуры.
3. Пакеты.
4. Последовательности.
5. Индексы.
6. Триггеры.
7. Хранимые директории.

## 2.1. Спецификация функциональных требований

Рассмотрим возможности, которые предоставляет разработанное программное средство. Для наглядности демонстрации использовались UML -диаграммы.

UML – уникальный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. [3]

На рисунке 2.1.1 показана UML-диаграмма, отображающая возможности приложения с точки зрения неавторизованного пользователя.

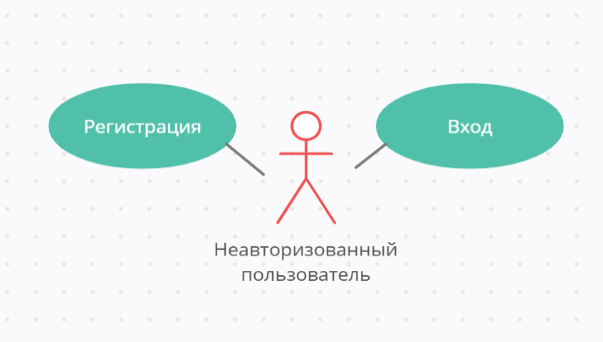


Рисунок 2.1.1 – UML диаграмма вариантов использования для неавторизованного пользователя

Авторизованный пользователь может управлять своими покупателями, осуществлять поиск товаров и добавление их в корзину, оформлять заказ, просматривать историю заказов, а также управление своими данными.

Также в приложение можно войти как администратор. Тогда открываются следующие возможности: добавление, удаление, изменение и просмотр всех таблиц.

## 2.2. Разработка модели базы данных

Для реализации базы данных для магазина бытовой химии было разработано 14 таблиц. Диаграмма базы со структурой связей представлена в приложении А, а в приложении Б представлен скрипт создания БД.

Логически можно вывести 15 основных таблиц: Customer, Cart, User, Customer\_number, Address, Phone\_book, Maker\_number, Maker, Product\_maker, Product, Storage, Category, Order, Order\_detail и Pictures.

Таблица User представляет перечень пользователей, состоит из столбцов:

−USER\_ID – идентификатор пользователя, тип int, первичный ключ, внешний ключ;

− LOGIN – уникальное имя пользователя для входа в систему, тип varchar(20);

− PASSWORD – пароль пользователя, тип varchar(20);

− TYPE\_OF\_USER – тип пользователя, тип varchar(20).

Таблица Customer представляет информацию о покупателях, состоит из столбцов:

−C\_ID– идентификатор покупателя, тип int, первичный ключ;

− FIRST\_NAME – имя покупателя, тип varchar(15);

− LAST\_NAME – фамилия покупателя, тип varchar(20);

− BIRTHDAY\_DATE– день рождения пользователя, тип date;

−ADDR\_ID – уникальный идентификатор адреса, тип int, внешний ключ;

− EMAIL– почта, тип varchar(25);

− GENDER – пол, тип varchar(2).

Таблица Customer\_number представляет перечень телефонных номеров пользователя, состоит из столбцов:

−C\_N\_ID – идентификатор контакта, тип int, первичный ключ;

− C\_ID– идентификатор пользователя, которому принадлежит данный контакт, тип int, внешний ключ;

− PH\_ID – идентификатор номера телефона, тип int, внешний ключ.

Таблица Phone\_book представляет перечень всех номеров телефонов, состоит из столбцов:

− PH\_ID – идентификатор номера телефона, тип int, первичный ключ;

− PHONE\_NUMBER– номер телефона, тип varchar(13).

Таблица Cart представляет корзину и состоит из столбцов:

−CART\_ID – идентификатор корзины, тип int, первичный ключ;

− P\_ID – идентификатор товара, тип int; вторичный ключ;

− С\_ID – идентификатор покупателя, тип int; вторичный ключ.

Таблица Address представляет адреса клиентов, состоит из столбцов:

−ADDR\_ID– идентификатор адреса, тип int, первичный ключ;

− CITY – название города, тип varchar(20);

− STREET – название улицы, тип varchar(20);

− BUILDING – номер дома, тип int;

− FLAT – номер квартиры, тип int.

Таблица Maker представляет информацию о производителе, состоит из столбцов:

−MAKER\_ID – идентификатор производителя, тип int, первичный ключ;

− MAKER\_NAME – название производителя, тип varchar(30);

− EMAIL – почта производителя, тип varchar(20).

Таблица Maker\_number представляет перечень телефонных номеров производителей, состоит из столбцов:

−M\_N\_ID – идентификатор контакта, тип int, первичный ключ;

− MAKER\_ID– идентификатор производителя, которому принадлежит данный контакт, тип int, внешний ключ;

− PH\_ID – идентификатор номера телефона, тип int, внешний ключ.

Таблица Product\_maker представляет перечень товаров и их производителей, состоит из столбцов:

−P\_M\_ID – идентификатор производителя определенного товара, тип int, первичный ключ;

− MAKER\_ID– идентификатор производителя, тип int, внешний ключ;

− P\_ID – идентификатор товара, тип int, внешний ключ.

Таблица Product представляет информацию о товарах, состоит из столбцов:

−P\_ID– идентификатор товара, тип int, первичный ключ;

− NAME – название товара, тип varchar(20);

− SHRT\_DSCR – краткое описание товара, тип varchar(30);

− PRICE – цена товара, тип decimal;

− CAT\_ID– идентификатор категории, тип int, внешний ключ.

Таблица Storage представляет информацию о товарах на складе, состоит из столбцов:

−P\_ID – идентификатор товара, тип int, внешний ключ;

− AMOUNT – количество товара на складе, тип int.

Таблица Pictures представляет фото и информацию о нем, состоит из столбцов:

−PIC\_ID – идентификатор картинки, тип int, первичный ключ;

− P\_ID – идентификатор товара, тип int, внешний ключ;

− PICTURE – фотография, тип blob;

Таблица Category представляет информацию о категориях товаров, состоит из столбцов:

−CAT\_ID– идентификатор категории товара, тип int, первичный ключ;

− NAME – название категории, тип varchar(20);

− SHRT\_DSCR – краткое описание категории, тип varchar(20).

Таблица Order представляет информацию о заказах, состоит из столбцов:

− O\_ID– идентификатор заказа, тип int, первичный ключ;

− C\_ID – идентификатор клиента, тип int, внешний ключ;

− P\_ID– идентификатор товара, тип int, внешний ключ;

− STATUS статус заказа, тип varchar(15);

− COUNT – количество товаров, тип int.

Таблица Order\_detail представляет дополнительную информацию о заказах, состоит из столбцов:

−O\_ID– идентификатор заказа, тип int, первичный ключ;

− DATE дата заказа, тип date;

− SUM – сумма заказа, тип decimal.

## 2.3. Пользователи

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации.

При проектировании базы данных были использованы 3 пользователя. Первый пользователь – с##client – обычный пользователь, имеет доступ к чтению и записи в таблицах, к которым имеет доступ.

Второй пользователь – manager – обладает более обширным перечнем прав. Имеет права для чтения из всех таблиц и изменения и записи во всех таблицах, к которым есть доступ у пользователя.

Третий пользователь – c##super – выполняет роль системы, которая держит на себе все объекты бд. Имеет все права.

## 2.4 Хранимые процедуры

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

При разработке курсового проекта было создано 36 процедур, хранящихся в 4 пакетах, для следующих целей:

1. Выборка данных для таблиц;

2. Удаление данных из таблиц;

3. Добавление данных в таблицы;

4. Изменение данных в таблицах;

5. Экспорт и импорт таблиц в формат xml;

Весь перечень созданных процедур будет представлен в Приложении В.

### 2.4.1 Выборка данных для таблиц

Для вывода данных из таблиц были написаны следующие процедуры: NUMBER\_BY\_CUSTOMER, GET\_CUSTOMERS, NUMBER\_BY\_MAKER, MAKER\_BY\_PRODUCT, CUSTOMER\_BY\_ID, PRODUCT\_BY\_ID, PICTURE\_BY\_PROD, GER\_PRODUCT, GET\_USERS, MAKER\_BY\_ID, USER\_BY\_ID, CART\_BY\_USER, PRODUCT\_BY\_CATEGORY и другие. Поскольку вывод таблицы размером в 100000 строк занимает довольно много времени, было принято решение создать индексы, что позволило уменьшить время вывода таблицы до минимума.

### 2.4.2 Удаление данных из таблиц

Для управления базой данных были созданы процедуры удаления DEL\_CART, DEL\_PRODUCT, DEL\_CUSTOMER, DEL\_CATEGORY, DEL\_CUS\_ORDER, DEL\_USER и DEL\_PICTURE. Они были разработаны для удаления строк из таблиц: CART, PRODUCT, CUSTOMER, USERS, CATEGORY, CUS\_ORDER и PICTURE.

### 2.4.3 Добавление данных в таблицы

Для управления базой данных были созданы процедуры добавления новых строк: ADD\_CATEGORY, ADD\_PRODUCT, ADD\_MAKER, ADD\_MNUMBER, ADD\_CNUMBER, ADD\_PICTURE, ADD\_NUMBER, ADD\_ADDRESS, ADD\_CART, ADD\_PRMAKER, ADD\_ORDER, ADD\_USER, ADD\_CUSTOMER. Они были разработаны для заполнения соответствующих таблиц.

**2.4.4 Изменение данных в таблицах**

Для управления базой данных были созданы процедуры обновления: UPDATE\_CART, UPDATE\_USER. Они были разработаны для обновления данных в соответствующих таблицах.

### 2.4.5 Экспорт и импорт таблиц в формат xml

Для экспорта таблицы CATEGORY в формате xml[5] была разработана процедура EXPORT\_CATEGORY\_TO\_XML. Для импорта таблицы CATEGORY в формате xml разработана процедура IMPORT\_CATEGORY\_FROM\_XML. Данные процедуры приведены в приложении Г и Д соответственно.

## 2.5 Индексы

Индекс [6] – объект базы данных, создаваемый с целью повышения производительности поиска данных [6]. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. В связи с необходимостью выборки таблиц, заполненных минимум на 100 000 строк, было принято решение разработать 9 индекса для всех таблиц.

Список индексов, созданных в рамках курсового проекта приведены ниже:

* ix\_product ON product (name,p\_id, cat\_id);
* ix\_ ix\_maker ON maker (maker\_name,maker\_id);
* ix\_category ON category (name,cat\_id);
* ix\_customer ON customer (c\_id, addr\_id);
* ix\_cus\_num ON customer\_number(c\_n\_id, c\_id, ph\_id);
* ix\_maker\_num ON maker\_number(m\_n\_id, maker\_id, ph\_id);
* ix\_prod\_m ON product\_maker(p\_m\_id, maker\_id, p\_id);
* ix\_cart ON cart(cart\_id,p\_id, c\_id);
* ix\_cus\_ord ON cus\_orders(o\_id,c\_id,p\_id);
* ix\_pictures ON pictures(pic\_id,p\_id).

Во время разработки использовалось 3 bitmap индекса для таблиц brand и model, так как данные таблицы не должны часто изменяться.

Для наглядной демонстрации повышения производительности поиска данных в таблице, ниже приведены рисунки с необходимой статистикой.





Рисунок 2.5.1 – Время и число операций чтения необходимые для поиска данных в таблице brand без индекса и с индексом

## 2.6 Пакеты

Пакет[5] Oracle PL/SQL — это объект схемы, который группирует логически связанные типы, элементы и подпрограммы. Пакеты обычно состоят из двух частей: спецификации и тела, хотя иногда тело не нужно.

Во время проектирования бд мне понадобилось создать 4 пакета для группировки процедур: USER\_PACK, ADMIN\_PACK, OBS\_PACK и XML\_PACKAGE.

USER\_PACK – хранит процедуры, которыми может пользоваться только пользователь (рисунок 2.6.1)



Рисунок 2.6.1 – спецификация пакета user\_pack

ADMIN\_PACK – хранит процедуры, которыми может пользоваться только администратор (рисунок 2.6.2)

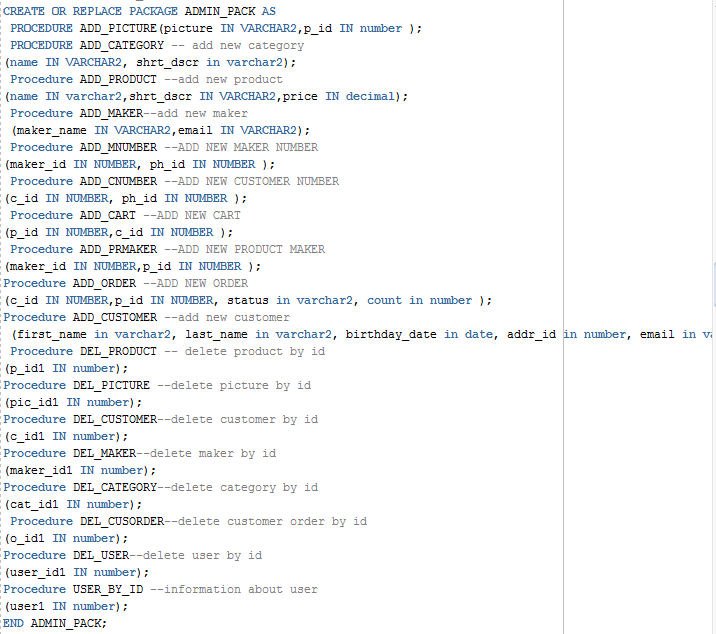


Рисунок 2.6.2 – спецификация пакета admin\_pack

OBS\_PACK – хранит процедуры, которыми могут пользоваться пользователь и администратор (рисунок 2.6.3)

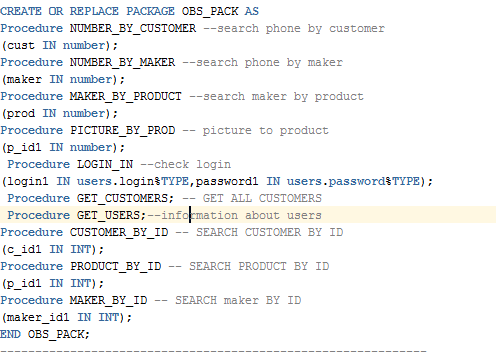


Рисунок 2.6.3 – спецификация пакета obs\_pack

XML\_PACKAGE – хранит процедуры, для работы с хмл (рисунок 2.6.4)

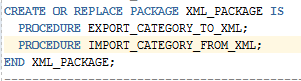


Рисунок 2.6.4 – спецификация пакета XML\_PACKAGE

## 2.7 Реализация последовательностей

Последовательность CREATE SEQUENCE [7] – это объект базы данных, который генерирует целые числа в соответствии с правилами, установленными во время его создания.

В процессе разработки проекта были использованы 10 последовательностей: customer\_s, users\_s, address\_s, maker\_s, cus\_num\_s, ph\_b\_s, mak\_num\_s, prod\_m\_s, cart\_s, category\_s, product\_s, pictures\_s, cus\_or\_s.

Примем создания последовательности приведен на рисунке 2.7.1

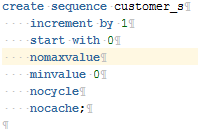


Рисунок 2.7.1 – создание последовательности

## 2.8 Триггеры

Триггеры PL/SQL - это именованные программные блоки, выполняемые в ответ на происходящие в базе данных Oracle события.

В процессе разработки проекта был создан триггер TR\_ADD\_LOGIN, который работает перед отработкой блока insert в таблицу users и проверяет на наличие дублирование логинов (рисунок 2.8.1).

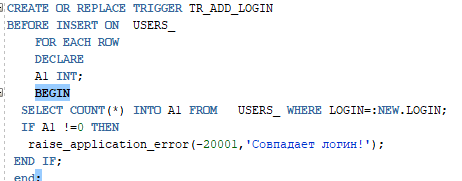


Рисунок 2.8.1 – создание триггера

## 2.9 Директории

Объект базы данных DIRECTORY - это ссылка на каталог сервера, на котором работает инстанс Oracle.

в ходе разработки бд были созданы 2 директории, для хранения xml и изображений: UTLDATA и PIC\_DIR.

Пример создания директории рисунок 2.9.1



Рисунок 2.9.1 – создание директории

## 2.10 Реализация технологий

В процессе разработки проекта была использована технология: мультимедийные типы данных в БД.

Oracle и PL/SQL поддерживают несколько разновидностей типов данных, предназначенных специально для работы с большими объектами (LOB[8], Large OBjects). Такие объекты позволяют хранить огромные (от 8 до 128 терабайт) объемы двоичных (например, графических) или текстовых данных.

В PL/SQL можно объявлять большие объекты четырех типов:

* BFILE— двоичный файл. Переменная этого типа содержит локатор файла, указывающий на файл операционной системы вне базы данных. Oracle интерпретирует содержимое файла как двоичные данные.
* BLOB— большой двоичный объект. Переменная этого типа содержит локатоp LOB, указывающий на большой двоичный объект, хранящийся в базе данных.
* CLOB — большой символьный объект. Переменная этого типа содержит локатор LOB, указывающий на хранящийся в базе данных большой блок текстовых данных в наборе символов базы данных.
* NLOB — большой символьный объект с поддержкой символов национальных языков (NLOB). Переменная этого типа содержит локатор LOB, указывающий на хранящийся в базе данных большой блок текстовых данных с национальным набором символов.

Большие объекты можно разделить на две категории: внутренние и внешние. Внутренние большие объекты (типы BLOB,CLOB и NCLOB) хранятся в базе данных и могут участвовать в транзакциях на сервере базы данных. Внешние большие объекты (тип BFILE) представляют двоичные данные, хранящиеся в файлах операционной системы вне таблиц базы данных. Они не могут участвовать в транзакциях, то есть вносимые в них изменения нельзя сохранить или отменить в зависимости от результата транзакции. Целостность данных обеспечивается только на уровне файловой системы. Кроме того, повторное чтение из BFILE может приводить к разным результатам — в отличие от внутренних больших объектов, соответствующих модели логической целостности чтения.

Код процедуры для добавления файла в бд:

CREATE OR REPLACE DIRECTORY PIC\_DIR AS 'C:/app/pictures/';

create or replace Procedure ADD\_PICTURE --add picture

(picture IN VARCHAR2, p\_id in number )

IS

pic\_id number;

L\_BLOB BLOB;

L\_BFILE BFILE;

BEGIN

SELECT pictures\_s.NEXTVAL INTO pic\_id FROM dual;

INSERT INTO pictures

( pic\_id,p\_id,picture)

VALUES

( pic\_id,p\_id,EMPTY\_BLOB())

RETURN picture INTO L\_BLOB;

L\_BFILE := BFILENAME('PIC\_DIR',picture);

DBMS\_LOB.FILEOPEN(L\_BFILE,DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADFROMFILE(L\_BLOB,L\_BFILE, DBMS\_LOB.GETLENGTH(L\_BFILE));

DBMS\_LOB.FILECLOSE(L\_BFILE);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

END;

Где: PICTURE IN VARCHAR2 – имя загружаемого файла,

P\_ID IN NUMBER – ID продукт, к которому относится данная картинка,

PIC\_ID IN NUMBER – ID данной картинки,

L\_BLOB BLOB – переменная, для хранения картинки во время исполнения кода PL/SQL,

L\_BFILE BFILE – переменная, для открытия файл, находящегося в директории.

Вставка в таблицу PICTURE:  
INSERT INTO pictures

( pic\_id,p\_id,picture)

VALUES

( pic\_id,p\_id,EMPTY\_BLOB())

RETURN picture INTO L\_BLOB;

EMPTY\_BLOB() – функция, которая заполняет колонку picture пустым значение типа BLOB. Дальше мы возвращаем это значение из колонки picture в переменную L\_BLOB.

L\_BFILE := BFILENAME('PIC\_DIR',PICTURE) – получаем путь файла с помощью созданной директории, и имени файла.

DBMS\_LOB.FILEOPEN(L\_BFILE,DBMS\_LOB.FILE\_READONLY) – открываем поток для чтения данных из указанного пути.

DBMS\_LOB.LOADFROMFILE(L\_BLOB,L\_BFILE, DBMS\_LOB.GETLENGTH(L\_BFILE)) – записываем данные из фалайла в переменную L\_BLOB.

DBMS\_LOB.FILECLOSE(L\_BFILE) – закрываем поток.

# 3 Тестирование, проверка работоспособности и анализ данных

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

4 Тестирование, проверка работоспособности и анализ данных

Лит.

Листов

1

БГТУ

Разработанное программное средство полностью соответствует созданному для него техническому заданию, также оно использует дополнительно разработанный функционал.

Для проверки корректности работы программного средства, продемонстрируем следующий скрипт

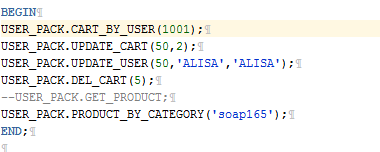


Рисунок 4.1 – Скрипт, демонстрирующий возможности пользователя

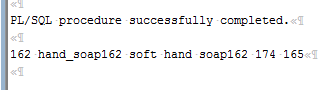


Рисунок 4.2 – вывод скрипта пользовотеля

Скрипт, демонстрирующий работоспособности администратора приведен ниже

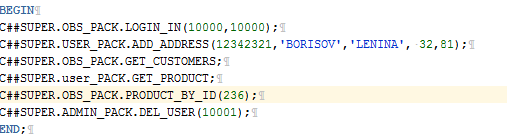


Рисунок 4.3 – Скрипт, демонстрирующий возможности администратора

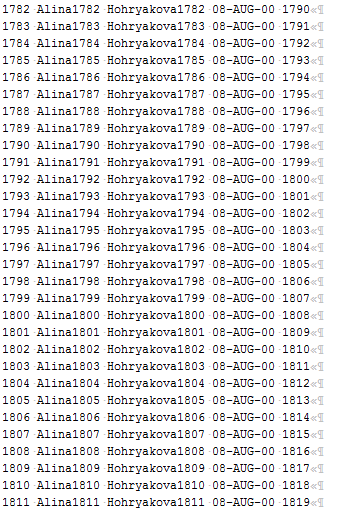


Рисунок 4.4 – вывод скрипта администратора

# Заключение

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

Заключение

Лит.

Листов

1

БГТУ

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных «Магазин товаров бытовой химии». Основной целью курсового проекта стало проектирование базы данных для дальнейшей интеграции с приложением/сайтом, которые помогут облегчить взаимодействие с базой данных посредством программного интерфейса. При разработке выполнены следующие пункты:

* Авторизация пользователей.
* Возможность добавления товаров в корзину.
* Возможность удаления товаров из корзины.
* Возможность редактирования корзины.
* Просмотр доступных товаров.
* Разграничение товаров по категориям.
* технология работы с мультимедийными типами данных;
* импорт и экспорт XML.

Приложение прошло тестирование при использовании в БД большого количество данных.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

# Список литературы

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

Список литературы

Лит.

Листов

1

БГТУ

1. Сайт ostrov-chistoty.by – https://ostrov-chistoty.by/.
2. Интернет ресурс для проектирования и создания UML-диаграмм – https://app.diagrams.net
3. Сайт habr.com – https://habr.com/ru/post/256655/.
4. Сайт oracle doc – https://docs.oracle.com/cd/B19306\_01/appdev.102/b14261/packages.htm.
5. Сайт oracle-dba.ru – https://oracle-dba.ru/docs/architecture/indexes/.
6. Сайт oracleplsql.ru– https://oracleplsql.ru/sequences.html
7. Сайт oracle-patches.com - https://oracle-patches.com/db/sql/3472-большие-объекты-данных-в-pl-sql-blob,-clob,-nclob-и-bfile

# Приложение А

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

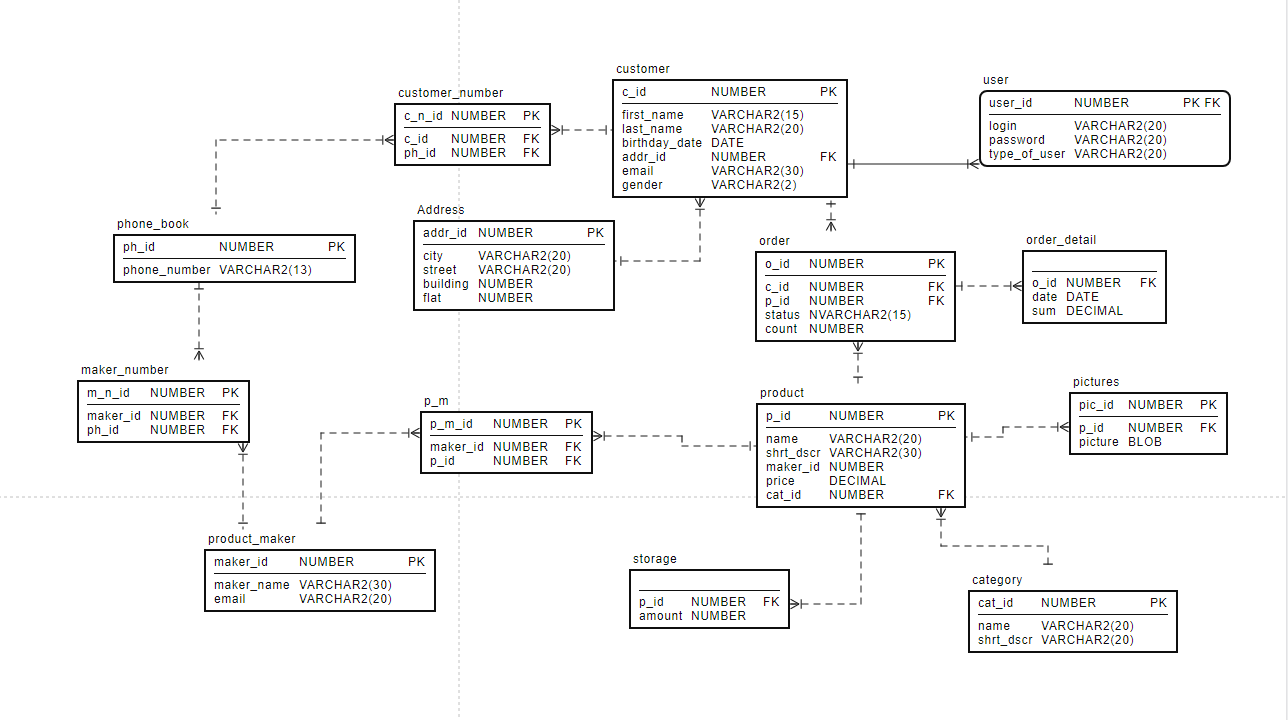
Приложение А

Лит.

Листов

1

БГТУ



# Приложение Б

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

Приложение Б

Лит.

Листов

1

БГТУ

CREATE TABLE address (

addr\_id NUMBER NOT NULL,

city VARCHAR2(20) NOT NULL,

street VARCHAR2(20) NOT NULL,

building NUMBER NOT NULL,

flat NUMBER NOT NULL,

PRIMARY KEY (addr\_id)

);

SELECT\* FROM category;

CREATE TABLE category (

cat\_id NUMBER NOT NULL,

name VARCHAR2(20) NOT NULL,

shrt\_dscr VARCHAR2(20) NOT NULL,

PRIMARY KEY (cat\_id)

);

CREATE TABLE maker (

maker\_id NUMBER NOT NULL,

maker\_name VARCHAR2(30) NOT NULL,

email VARCHAR2(20) NOT NULL,

PRIMARY KEY (maker\_id)

);

CREATE TABLE phone\_book (

ph\_id NUMBER NOT NULL,

phone\_number VARCHAR2(13) NOT NULL,

PRIMARY KEY (ph\_id)

);

CREATE TABLE users (

user\_id NUMBER NOT NULL,

login VARCHAR2(20) NOT NULL,

password VARCHAR2(20) NOT NULL,

type\_of\_user VARCHAR2(20) NOT NULL,

PRIMARY KEY (user\_id)

);

INSERT INTO users(user\_id,login, password, type\_of\_user) VALUES(0,'ADMIN','ADMIN','ADMIN');

CREATE TABLE customer (

c\_id NUMBER NOT NULL,

first\_name VARCHAR2(15) NOT NULL,

last\_name VARCHAR2(20) NOT NULL,

birthday\_date DATE NOT NULL,

addr\_id NUMBER NOT NULL,

email VARCHAR2(30) NOT NULL,

gender VARCHAR2(2) NOT NULL,

PRIMARY KEY (c\_id),

CONSTRAINT customer\_addr\_id\_Address\_addr\_id\_foreign FOREIGN KEY (addr\_id) REFERENCES address (addr\_id),

CONSTRAINT customer\_c\_id\_users\_user\_id\_foreign FOREIGN KEY (c\_id) REFERENCES users (user\_id)

);

CREATE TABLE maker\_number (

m\_n\_id NUMBER NOT NULL,

maker\_id NUMBER NOT NULL,

ph\_id NUMBER NOT NULL,

PRIMARY KEY (m\_n\_id),

CONSTRAINT maker\_number\_ph\_id\_phone\_book\_ph\_id\_foreign FOREIGN KEY (ph\_id) REFERENCES phone\_book (ph\_id),

CONSTRAINT maker\_number\_maker\_id\_product\_maker\_maker\_name\_foreign FOREIGN KEY (maker\_id) REFERENCES maker (maker\_id)

);

CREATE TABLE product (

p\_id NUMBER NOT NULL,

name VARCHAR2(20) NOT NULL,

shrt\_dscr VARCHAR2(30) NOT NULL,

price DECIMAL NOT NULL,

cat\_id NUMBER NOT NULL,

PRIMARY KEY (p\_id),

CONSTRAINT product\_cat\_id\_category\_cat\_id\_foreign FOREIGN KEY (cat\_id) REFERENCES category (cat\_id)

);

CREATE TABLE cart (

cart\_id NUMBER NOT NULL,

p\_id NUMBER NOT NULL,

c\_id NUMBER NOT NULL,

PRIMARY KEY (cart\_id),

CONSTRAINT Cart\_p\_id\_product\_p\_id\_foreign FOREIGN KEY (p\_id) REFERENCES product (p\_id),

CONSTRAINT Cart\_c\_id\_customer\_c\_id\_foreign FOREIGN KEY (c\_id) REFERENCES customer (c\_id)

);

CREATE TABLE cus\_orders (

o\_id NUMBER NOT NULL,

c\_id NUMBER NOT NULL,

p\_id NUMBER NOT NULL,

status NVARCHAR2(15) NOT NULL,

count NUMBER NOT NULL,

PRIMARY KEY (o\_id),

CONSTRAINT order\_p\_id\_product\_p\_id\_foreign FOREIGN KEY (p\_id) REFERENCES product (p\_id),

CONSTRAINT order\_c\_id\_customer\_c\_id\_foreign FOREIGN KEY (c\_id) REFERENCES customer (c\_id)

);

CREATE TABLE customer\_number (

c\_n\_id NUMBER NOT NULL,

c\_id NUMBER NOT NULL,

ph\_id NUMBER NOT NULL,

PRIMARY KEY (c\_n\_id),

CONSTRAINT customer\_number\_c\_id\_customer\_c\_id\_foreign FOREIGN KEY (c\_id) REFERENCES customer (c\_id),

CONSTRAINT customer\_number\_ph\_id\_phone\_book\_ph\_id\_foreign FOREIGN KEY (ph\_id) REFERENCES phone\_book (ph\_id)

);

CREATE TABLE pictures (

pic\_id NUMBER NOT NULL,

p\_id NUMBER NOT NULL,

picture BLOB NOT NULL,

PRIMARY KEY (pic\_id),

CONSTRAINT pictures\_p\_id\_product\_p\_id\_foreign FOREIGN KEY (p\_id) REFERENCES product (p\_id)

);

CREATE TABLE product\_maker (

p\_m\_id NUMBER NOT NULL,

maker\_id NUMBER NOT NULL,

p\_id NUMBER NOT NULL,

PRIMARY KEY (p\_m\_id),

CONSTRAINT p\_m\_p\_id\_product\_p\_id\_foreign FOREIGN KEY (p\_id) REFERENCES product (p\_id),

CONSTRAINT product\_maker\_maker\_id\_maker\_maker\_id\_foreign FOREIGN KEY (maker\_id) REFERENCES maker (maker\_id)

);

CREATE TABLE storage (

p\_id NUMBER NOT NULL,

amount NUMBER NOT NULL,

CONSTRAINT storage\_p\_id\_product\_p\_id\_foreign FOREIGN KEY (p\_id) REFERENCES product (p\_id)

);

CREATE TABLE details\_order (

o\_id NUMBER NOT NULL,

date\_of\_order DATE NOT NULL,

sum\_of\_order DECIMAL NOT NULL,

CONSTRAINT order\_detail\_o\_id\_order\_o\_id\_foreign FOREIGN KEY (o\_id) REFERENCES cus\_orders (o\_id)

);

# Приложение В

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С

Провер.

Колмаков М.В.

Н. контр.

Утверд.

Приложение В

Лит.

Листов

1

БГТУ

# CREATE OR REPLACE PACKAGE USER\_PACK AS

# Procedure ADD\_NUMBER --ADD NEW NUMBER

# (ph\_id1 in number ,ph\_number IN VARCHAR2);

# Procedure ADD\_ADDRESS --add new address

# (addr\_id1 in number,city IN VARCHAR2, street IN VARCHAR2, building in number, flat in number);

# Procedure ADD\_USER --add new user

# (user\_id1 in number ,login IN VARCHAR2,password IN VARCHAR2);

# Procedure CART\_BY\_USER -- cart by user

# (user\_id IN number);

# Procedure UPDATE\_CART -- update cart

# (cust in number,prod in number );

# Procedure UPDATE\_USER -- update user

# (user\_id1 in number,log in varchar2,pass in varchar2 ) ;

# Procedure DEL\_CART -- delete from cart

# (cart\_id1 IN number);

# Procedure GET\_PRODUCT;--GET ALL PRODUCTS

# Procedure PRODUCT\_BY\_CATEGORY--SEARCH INFORMATION ABOUT PRODUCT BY CATEGORY

# ( name1 in varchar2);

# END USER\_PACK;

# drop package USER\_PACK;

# -------------------------------------------------------------

# CREATE OR REPLACE PACKAGE BODY USER\_PACK AS

# Procedure ADD\_NUMBER --ADD NEW NUMBER

# (ph\_id1 in number,ph\_number IN VARCHAR2)

# IS

# ph\_id number;

# BEGIN

# SELECT ph\_b\_s.NEXTVAL INTO ph\_id FROM dual;

# INSERT INTO phone\_book

# ( ph\_id, phone\_number)

# VALUES

# (ph\_id1,ph\_number);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_ADDRESS --add new address

# (addr\_id1 in number,city IN VARCHAR2, street IN VARCHAR2, building in number, flat in number)

# IS

# addr\_id number;

# BEGIN

# SELECT address\_s.NEXTVAL INTO addr\_id FROM dual;

# INSERT INTO address

# ( addr\_id, city, street, building, flat)

# VALUES

# (addr\_id1, city, street, building, flat);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_USER --add new user

# (user\_id1 in number,login IN VARCHAR2,password IN VARCHAR2)

# IS

# user\_id number;

# BEGIN

# SELECT users\_s.NEXTVAL INTO user\_id FROM dual;

# INSERT INTO users

# ( user\_id, login, password, type\_of\_user)

# VALUES

# (user\_id1, login, password, 'USER');

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure CART\_BY\_USER -- cart by user

# (user\_id IN number)

# IS

# CURSOR GET\_CART IS

# SELECT \* FROM cart WHERE user\_id=c\_id;

# C GET\_CART%ROWTYPE;

# BEGIN

# BEGIN

# OPEN GET\_CART;

# LOOP

# EXIT WHEN GET\_CART%NOTFOUND;

# FETCH GET\_CART INTO C;

# IF GET\_CART%FOUND THEN

# DBMS\_OUTPUT.PUT\_LINE(C.cart\_id|| ' ' ||C.p\_id|| ' ' ||C.c\_id);

# END IF;

# END LOOP;

# CLOSE GET\_CART;

# END;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure UPDATE\_CART -- update cart

# (cust in number,prod in number )

# IS

# BEGIN

# IF cust IS NULL THEN

# UPDATE cart

# SET p\_id=prod WHERE c\_id=cust;

# ELSE

# UPDATE cart

# SET p\_id=prod WHERE c\_id=cust;

# END IF;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure UPDATE\_USER -- update user

# (user\_id1 in number,log in varchar2,pass in varchar2 )

# IS

# BEGIN

# IF log IS NULL THEN

# UPDATE users

# SET login=log WHERE user\_id=user\_id1;

# ELSIF log is null then

# update users

# set password=pass where user\_id=user\_id1;

# ELSE

# update users

# SET login=log, password=pass WHERE user\_id=user\_id1;

# END IF;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure DEL\_CART -- delete from cart

# (cart\_id1 IN number)

# IS

# BEGIN

# DELETE cart WHERE cart\_id=cart\_id1;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure GET\_PRODUCT--GET ALL PRODUCTS

# IS

# CURSOR product IS

# SELECT \* FROM product;

# p product%ROWTYPE;

# BEGIN

# BEGIN

# OPEN product;

# LOOP

# EXIT WHEN product%NOTFOUND;

# FETCH product INTO p;

# IF product%FOUND THEN

# DBMS\_OUTPUT.PUT\_LINE(p.p\_id|| ' ' || p.name|| ' ' ||p.shrt\_dscr);

# END IF;

# END LOOP;

# CLOSE product;

# END;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure PRODUCT\_BY\_CATEGORY--SEARCH INFORMATION ABOUT PRODUCT BY CATEGORY

# ( name1 in varchar2)

# IS

# CURSOR GET\_COMMENT IS

# SELECT product.p\_id, product.name, product.shrt\_dscr,product.price,category.cat\_id FROM product,category WHERE product.cat\_id=category.cat\_id AND category.name=name1;

# U GET\_COMMENT%ROWTYPE;

# BEGIN

# BEGIN

# OPEN GET\_COMMENT;

# LOOP

# EXIT WHEN GET\_COMMENT%NOTFOUND;

# FETCH GET\_COMMENT INTO U;

# IF GET\_COMMENT%FOUND THEN

# DBMS\_OUTPUT.PUT\_LINE(U.p\_id|| ' ' || U.name|| ' ' ||U.shrt\_dscr|| ' ' ||U.price|| ' '||U.cat\_id);

# END IF;

# END LOOP;

# CLOSE GET\_COMMENT;

# END;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# 

# END USER\_PACK;

# --CONNECT C##CLIENT/12345678;

# --BEGIN C##SUPER.USER\_PACK.ADD\_ADDRESS('MINSK',NULL,NULL,NULL);END;

# --DROP PACKAGE ADMIN\_PACK;

# CREATE OR REPLACE PACKAGE ADMIN\_PACK AS

# PROCEDURE ADD\_PICTURE(picture IN VARCHAR2,p\_id IN number );

# PROCEDURE ADD\_CATEGORY -- add new category

# (name IN VARCHAR2, shrt\_dscr in varchar2);

# Procedure ADD\_PRODUCT --add new product

# (prod\_name in varchar2,name1 IN varchar2);

# Procedure ADD\_MAKER--add new maker

# (maker\_name IN VARCHAR2,email IN VARCHAR2);

# Procedure ADD\_MNUMBER --ADD NEW MAKER NUMBER

# (maker\_id IN NUMBER, ph\_id IN NUMBER );

# Procedure ADD\_CNUMBER --ADD NEW CUSTOMER NUMBER

# (c\_id IN NUMBER, ph\_id IN NUMBER );

# Procedure ADD\_CART --ADD NEW CART

# (p\_id IN NUMBER,c\_id IN NUMBER );

# Procedure ADD\_PRMAKER --ADD NEW PRODUCT MAKER

# (maker\_id IN NUMBER,p\_id IN NUMBER );

# Procedure ADD\_ORDER --ADD NEW ORDER

# (c\_id IN NUMBER,p\_id IN NUMBER, status in varchar2, count in number );

# Procedure ADD\_CUSTOMER --add new customer

# (first\_name in varchar2, last\_name in varchar2, birthday\_date in date, addr\_id in number, email in varchar2, gender in varchar2);

# Procedure DEL\_PRODUCT -- delete product by id

# (p\_id1 IN number);

# Procedure DEL\_PICTURE --delete picture by id

# (pic\_id1 IN number);

# Procedure DEL\_CUSTOMER--delete customer by id

# (c\_id1 IN number);

# Procedure DEL\_MAKER--delete maker by id

# (maker\_id1 IN number);

# Procedure DEL\_CATEGORY--delete category by id

# (cat\_id1 IN number);

# Procedure DEL\_CUSORDER--delete customer order by id

# (o\_id1 IN number);

# Procedure DEL\_USER--delete user by id

# (user\_id1 IN number);

# Procedure USER\_BY\_ID --information about user

# (user1 IN number);

# END ADMIN\_PACK;

# ------------------------------------------------------------------

# CREATE OR REPLACE PACKAGE BODY ADMIN\_PACK AS

# Procedure ADD\_PICTURE --add picture

# (picture IN VARCHAR2, p\_id in number )

# IS

# pic\_id number;

# L\_BLOB BLOB;

# L\_BFILE BFILE;

# BEGIN

# SELECT pictures\_s.NEXTVAL INTO pic\_id FROM dual;

# INSERT INTO pictures

# ( pic\_id,p\_id,picture)

# VALUES

# ( pic\_id,p\_id,EMPTY\_BLOB())

# RETURN picture INTO L\_BLOB;

# 

# L\_BFILE := BFILENAME('PIC\_DIR',picture);

# DBMS\_LOB.FILEOPEN(L\_BFILE,DBMS\_LOB.FILE\_READONLY);

# DBMS\_LOB.LOADFROMFILE(L\_BLOB,L\_BFILE, DBMS\_LOB.GETLENGTH(L\_BFILE));

# DBMS\_LOB.FILECLOSE(L\_BFILE);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_CATEGORY -- add new category

# (name IN VARCHAR2, shrt\_dscr in varchar2)

# IS

# cat\_id number;

# BEGIN

# SELECT category\_s.NEXTVAL INTO cat\_id FROM dual;

# INSERT INTO category

# ( cat\_id, name,shrt\_dscr)

# VALUES

# (cat\_id,UPPER(name),UPPER(shrt\_dscr));

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_PRODUCT --add new product

# (prod\_name in varchar2,name1 IN varchar2)

# IS

# p\_id number;

# cat\_id1 number;

# BEGIN

# SELECT product\_s.NEXTVAL INTO p\_id FROM dual;

# SELECT cat\_id INTO cat\_id1 FROM category WHERE category.name ='name1';

# INSERT INTO product

# (p\_id, name,cat\_id)

# VALUES

# (p\_id, prod\_name, cat\_id1);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_MAKER--add new maker

# (maker\_name IN VARCHAR2,email IN VARCHAR2)

# IS

# maker\_id NUMBER;

# BEGIN

# SELECT maker\_s.NEXTVAL INTO maker\_id FROM dual;

# INSERT INTO maker

# ( maker\_id, maker\_name, email)

# VALUES

# (maker\_id,maker\_name,email);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_MNUMBER --ADD NEW MAKER NUMBER

# (maker\_id IN NUMBER, ph\_id IN NUMBER )

# IS

# m\_n\_id NUMBER;

# BEGIN

# SELECT mak\_num\_s.NEXTVAL INTO m\_n\_id FROM dual;

# INSERT INTO maker\_number

# ( m\_n\_id, maker\_id, ph\_id)

# VALUES

# ( m\_n\_id, maker\_id, ph\_id);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_CNUMBER --ADD NEW CUSTOMER NUMBER

# (c\_id IN NUMBER, ph\_id IN NUMBER )

# IS

# c\_n\_id NUMBER;

# BEGIN

# SELECT cus\_num\_s.NEXTVAL INTO c\_n\_id FROM dual;

# INSERT INTO customer\_number

# ( c\_n\_id, c\_id, ph\_id)

# VALUES

# ( c\_n\_id, c\_id, ph\_id);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_CART --ADD NEW CART

# (p\_id IN NUMBER,c\_id IN NUMBER )

# IS

# cart\_id NUMBER;

# BEGIN

# SELECT cart\_s.NEXTVAL INTO cart\_id FROM dual;

# INSERT INTO cart

# ( cart\_id,p\_id, c\_id)

# VALUES

# ( cart\_id,p\_id, c\_id);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_PRMAKER --ADD NEW PRODUCT MAKER

# (maker\_id IN NUMBER,p\_id IN NUMBER )

# IS

# p\_m\_id NUMBER;

# BEGIN

# SELECT prod\_m\_s.NEXTVAL INTO p\_m\_id FROM dual;

# INSERT INTO product\_maker

# (p\_m\_id,maker\_id, p\_id)

# VALUES

# ( p\_m\_id,maker\_id, p\_id);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_ORDER --ADD NEW ORDER

# (c\_id IN NUMBER,p\_id IN NUMBER, status in varchar2, count in number )

# IS

# o\_id NUMBER;

# BEGIN

# SELECT cus\_or\_s.NEXTVAL INTO o\_id FROM dual;

# INSERT INTO cus\_orders

# (o\_id,c\_id, p\_id, status, count)

# VALUES

# ( o\_id,c\_id, p\_id, status, count);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure ADD\_CUSTOMER --add new customer

# (first\_name in varchar2, last\_name in varchar2, birthday\_date in date, addr\_id in number, email in varchar2, gender in varchar2)

# IS

# c\_id number;

# BEGIN

# SELECT customer\_s.NEXTVAL INTO c\_id FROM dual;

# INSERT INTO customer

# (c\_id, first\_name, last\_name, birthday\_date, addr\_id, email, gender)

# VALUES

# (c\_id, first\_name, last\_name, birthday\_date, addr\_id, email, gender);

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure DEL\_PRODUCT -- delete product by id

# (p\_id1 IN number)

# IS

# BEGIN

# DELETE product WHERE p\_id=p\_id1;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure DEL\_PICTURE --delete picture by id

# (pic\_id1 IN number)

# IS

# BEGIN

# DELETE pictures WHERE pic\_id=pic\_id1;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure DEL\_CUSTOMER--delete customer by id

# (c\_id1 IN number)

# IS

# BEGIN

# DELETE customer WHERE c\_id=c\_id1;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure DEL\_MAKER--delete maker by id

# (maker\_id1 IN number)

# IS

# BEGIN

# DELETE maker WHERE maker\_id=maker\_id1;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure DEL\_CATEGORY--delete category by id

# (cat\_id1 IN number)

# IS

# BEGIN

# DELETE category WHERE cat\_id=cat\_id1;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure DEL\_CUSORDER--delete customer order by id

# (o\_id1 IN number)

# IS

# BEGIN

# DELETE cus\_orders WHERE o\_id=o\_id1;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure DEL\_USER--delete user by id

# (user\_id1 IN number)

# IS

# BEGIN

# DELETE customer WHERE c\_id=user\_id1;

# COMMIT;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# Procedure USER\_BY\_ID --information about user

# (user1 IN number)

# IS

# CURSOR GET\_USER IS

# SELECT login,type\_of\_user FROM users WHERE users.user\_id=user1;

# U GET\_USER%ROWTYPE;

# BEGIN

# BEGIN

# OPEN GET\_USER;

# LOOP

# EXIT WHEN GET\_USER%NOTFOUND;

# FETCH GET\_USER INTO U;

# IF GET\_USER%FOUND THEN

# DBMS\_OUTPUT.PUT\_LINE(U.login|| ' ' || U.type\_of\_user);

# END IF;

# END LOOP;

# CLOSE GET\_USER;

# END;

# EXCEPTION

# WHEN OTHERS THEN

# raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

# END;

# END ADMIN\_PACK;

# Приложение Г

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

Приложение Г

Лит.

Листов

1

БГТУ

<category xmlns:xsi="http://www.w3.org/2001/XMLSchema" xsi:nonamespaceSchemaLocation="http://www.oracle.com/Employee.xsd">

<category>

<ID>217</ID>

<CATEGORY>soap217</CATEGORY>

</category>

<category>

<ID>218</ID>

<CATEGORY>soap218</CATEGORY>

</category>

<category>

<ID>219</ID>

<CATEGORY>soap219</CATEGORY>

</category>

<category>

<ID>220</ID>

<CATEGORY>soap220</CATEGORY>

</category>

<category>

<ID>221</ID>

<CATEGORY>soap221</CATEGORY>

</category>

<category>

<ID>222</ID>

<CATEGORY>soap222</CATEGORY>

</category>

<category>

<ID>223</ID>

<CATEGORY>soap223</CATEGORY>

</category>

<category>

<ID>224</ID>

<CATEGORY>soap224</CATEGORY>

</category>

<category>

<ID>225</ID>

<CATEGORY>soap225</CATEGORY>

</category>

<category>

<ID>226</ID>

<CATEGORY>soap226</CATEGORY>

</category>

<category>

<ID>227</ID>

<CATEGORY>soap227</CATEGORY>

</category>

<category>

<ID>228</ID>

<CATEGORY>soap228</CATEGORY>

</category>

<category>

<ID>229</ID>

<CATEGORY>soap229</CATEGORY>

</category>

<category>

<ID>230</ID>

<CATEGORY>soap230</CATEGORY>

</category>

<category>

<ID>231</ID>

<CATEGORY>soap231</CATEGORY>

</category>

<category>

<ID>232</ID>

<CATEGORY>soap232</CATEGORY>

</category>

<category>

<ID>233</ID>

<CATEGORY>soap233</CATEGORY>

</category>

<category>

<ID>234</ID>

<CATEGORY>soap234</CATEGORY>

</category>

<category>

<ID>235</ID>

<CATEGORY>soap235</CATEGORY>

</category>

<category>

<ID>236</ID>

<CATEGORY>soap236</CATEGORY>

</category>

<category>

<ID>237</ID>

<CATEGORY>soap237</CATEGORY>

</category>

<category>

<ID>238</ID>

<CATEGORY>soap238</CATEGORY>

</category>

<category>

<ID>239</ID>

<CATEGORY>soap239</CATEGORY>

</category>

<category>

<ID>240</ID>

<CATEGORY>soap240</CATEGORY>

</category>

<category>

<ID>241</ID>

<CATEGORY>soap241</CATEGORY>

</category>

<category>

<ID>242</ID>

<CATEGORY>soap242</CATEGORY>

</category>

<category>

<ID>243</ID>

<CATEGORY>soap243</CATEGORY>

</category>

<category>

<ID>244</ID>

<CATEGORY>soap244</CATEGORY>

</category>

<category>

<ID>245</ID>

<CATEGORY>soap245</CATEGORY>

</category>

<category>

<ID>246</ID>

<CATEGORY>soap246</CATEGORY>

</category>

</category>

# Приложение Д

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Хохрякова А.С.

Провер.

Колмаков М.В.

Н. контр.

Утверд.

Приложение Д

Лит.

Листов

1

БГТУ

CREATE TABLE empfromxml ( cat\_id NUMBER, name VARCHAR2(50 ));

CREATE DIRECTORY xmltest AS 'C:/DB/XML/';

INSERT INTO EMPFROMXML (cat\_id, name)

SELECT ExtractValue(Value(x),'//ID') as cat\_id,

ExtractValue(Value(x),'//CATEGORY') as name

FROM TABLE(XMLSequence(Extract(XMLType(bfilename('XMLTEST', 'category.xml'),

nls\_charset\_id('UTF8') ),'/category/category'))) x;

select \* from empfromxml;

CREATE OR REPLACE DIRECTORY UTLDATA AS 'C:\DB\XML\';

--DROP DIRECTORY UTLDATA;

--------------------------------------------------------------------------------

--DROP PACKAGE XML\_PACKAGE;

CREATE OR REPLACE PACKAGE XML\_PACKAGE IS

PROCEDURE EXPORT\_CATEGORY\_TO\_XML;

PROCEDURE IMPORT\_CATEGORY\_FROM\_XML;

END XML\_PACKAGE;

--------------------------------------------------------------------------------

CREATE OR REPLACE PACKAGE BODY XML\_PACKAGE IS

--------------------------------------------------------------------------------

PROCEDURE EXPORT\_CATEGORY\_TO\_XML

IS

DOC DBMS\_XMLDOM.DOMDocument;

XDATA XMLTYPE;

CURSOR XMLCUR IS

SELECT XMLELEMENT("category",

XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi",

'http://www.oracle.com/Employee.xsd' AS "xsi:nonamespaceSchemaLocation"),

XMLAGG(XMLELEMENT("category",

XMLELEMENT("ID",U.cat\_id),

XMLELEMENT("CATEGORY",U.name)

))

) FROM category U;

BEGIN

OPEN XMLCUR;

LOOP

FETCH XMLCUR INTO XDATA;

EXIT WHEN XMLCUR%NOTFOUND;

END LOOP;

CLOSE XMLCUR;

DOC := DBMS\_XMLDOM.NewDOMDocument(XDATA);

DBMS\_XMLDOM.WRITETOFILE(DOC, 'UTLDATA/category.xml');

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20102, 'EXPORT XML ERROR');

END EXPORT\_CATEGORY\_TO\_XML;

--------------------------------------------------------------------------------

PROCEDURE IMPORT\_CATEGORY\_FROM\_XML

IS

L\_CLOB CLOB;

L\_BFILE BFILE := BFILENAME('UTLDATA', 'category.xml');

L\_DEST\_OFFSET INTEGER := 1;

L\_SRC\_OFFSET INTEGER := 1;

L\_BFILE\_CSID NUMBER := 0;

L\_LANG\_CONTEXT INTEGER := 0;

L\_WARNING INTEGER := 0;

P DBMS\_XMLPARSER.PARSER;

V\_DOC DBMS\_XMLDOM.DOMDOCUMENT;

V\_ROOT\_ELEMENT DBMS\_XMLDOM.DOMELEMENT;

V\_CHILD\_NODES DBMS\_XMLDOM.DOMNODELIST;

V\_CURRENT\_NODE DBMS\_XMLDOM.DOMNODE;

U category%ROWTYPE;

BEGIN

DBMS\_LOB.CREATETEMPORARY (L\_CLOB, TRUE);

DBMS\_LOB.FILEOPEN(L\_BFILE, DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADCLOBFROMFILE (DEST\_LOB => L\_CLOB, SRC\_BFILE => L\_BFILE, AMOUNT => DBMS\_LOB.LOBMAXSIZE,

DEST\_OFFSET => L\_DEST\_OFFSET, SRC\_OFFSET => L\_SRC\_OFFSET, BFILE\_CSID => L\_BFILE\_CSID,

LANG\_CONTEXT => L\_LANG\_CONTEXT, WARNING => L\_WARNING);

DBMS\_LOB.FILECLOSE(L\_BFILE);

COMMIT;

-- Create XML Parser.

P := DBMS\_XMLPARSER.NEWPARSER;

-- Parse XML into DOM object

DBMS\_XMLPARSER.PARSECLOB(P,L\_CLOB);

-- Document Element

V\_DOC := DBMS\_XMLPARSER.GETDOCUMENT(P);

-- Root element

V\_ROOT\_ELEMENT := DBMS\_XMLDOM.Getdocumentelement(v\_Doc);

--- return Dbms\_Xmldom.Domnodelist

V\_CHILD\_NODES := DBMS\_XMLDOM.GETCHILDRENBYTAGNAME(V\_ROOT\_ELEMENT,'\*');

FOR i IN 0 .. DBMS\_XMLDOM.GETLENGTH(V\_CHILD\_NODES) - 1

LOOP

-- <USER> Node.

V\_CURRENT\_NODE := DBMS\_XMLDOM.ITEM(V\_CHILD\_NODES,i);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE,

'cat\_id/text()',U.cat\_id);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE,

'name/text()',U.name);

INSERT INTO category(cat\_id,name)

VALUES(I+1,U.name) ;

END LOOP;

DBMS\_LOB.FREETEMPORARY(L\_CLOB);

DBMS\_XMLPARSER.FREEPARSER(P);

DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_LOB.FREETEMPORARY(L\_CLOB);

DBMS\_XMLPARSER.FREEPARSER(P);

DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);

RAISE\_APPLICATION\_ERROR(-20101, 'IMPORT XML ERROR'|| SQLERRM);

END IMPORT\_CATEGORY\_FROM\_XML;

--------------------------------------------------------------------------------

END XML\_PACKAGE;

--------------------------------------------------------------------------------

BEGIN C##SUPER.XML\_PACKAGE.IMPORT\_CATEGORY\_FROM\_XML;END;

BEGIN C##SUPER.XML\_PACKAGE.EXPORT\_CATEGORY\_TO\_XML;END;

DROP PACKAGE XML\_PACKAGE;