|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ»

ДИСЦИПЛИНА «БАЗЫ ДАННЫХ»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

База данных "Снабжение"

Студент ИУ6-42 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бурлаков А.С.**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Скворцова М.А.**

(Подпись, дата) (И.О.Фамилия)

*2018 г.*

**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ6

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Пролетарский

« \_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине ***Базы данных***

Студент группы ИУ6-41

Гринберг Ханна Эдуардовна, Иванов Дмитрий Вячеславович

(Фамилия, имя, отчество)

Тема курсовой работы

База Данных «Туристическое агентство»

Направленность КР ***учебная***

Источник тематики ***кафедра ИУ6***

График выполнения КР: 25% к 3 нед., 50% к 7 нед., 75% к 10 нед., 100% к 14 нед.

Техническое задание Разработать базу данных “Туристическое агентство”, состоящую минимум из 5 таблиц, имеющую сложные запросы и отчеты. Разработать удобный и понятный пользовательский интерфейс. В качестве СУБД для разработки использовать Oracle Database Express Edition 11g, Oracle Database SQL Developer. Для проектирования схемы данных использовать Oracle SQL Developer Data Modeler. Для проектирования пользовательского интерфейса необходимо использовать Oracle APEX версии 5.1.

***Оформление курсовой работы:***

Расчетно-пояснительная записка на не менее 25 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания «10 » февраля 2017 г.

**Руководитель курсовой работы**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Скворцова

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Х.Э. Гринберг

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Д.В. Иванов

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**РЕФЕРАТ**

Расчетно-пояснительная записка 38 с., 16 рис., 8 табл., 7 источников.

Объектом разработки является база данных для снабжения склада и пользовательский интерфейс для неё.

Цель работы — создание удобного и доступного интерфейса для базы данных снабжения склада.

Поставленная цель достигается за счет средств Oracle APEX версии 5.1 и Oracle SQL Developer Data Modeler. В качестве СУБД для разработки используется Oracle Database Express Edition 11g, Oracle Database SQL Developer. Разрабатываемая база данных содержит 13 таблиц, взаимодействующих между собой при помощи связей. Пользовательский интерфейс реализует взаимодействие сотрудников склада, поставщиков и заказчиков за счет формирования заказов, просмотра информации по предыдущим заказам и формированию накладных

Ключевые слова - Oracle APEX, база данных, СУБД, Oracle SQL Developer Data Modeler, таблицы, триггеры, записи, поля.

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 5

1 Анализ и проектирование базы данных 6

1.1 Исследование предметной области 6

1.2 Создание модели в Oracle SQL Developer Data Modeler 7

2 Формирование базы данных 13

2.1 Создание пользователя для базы данных 13

2.2 Перенос базы данных в Oracle Database SQL Developer 13

2.3 Добавление данных в таблицы 14

3 Создание интерфейса в среде Oracle APEX версии 5.1 15

3.1 Создание страницы авторизации 15

3.2 Создание страницы регистрации 19

3.3 Создание страницы бронирования 21

3.4 Создание отчета для просмотра личного бронирования для

пользователя 23

3.5 Создание отчета для просмотра данных постояльцев

администратором 25

ЗАКЛЮЧЕНИЕ 29

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 30

Приложение А 31

**ВВЕДЕНИЕ**

* данной работе производится создание базы данных для снабжения склада и пользовательского интерфейса для неё. Она предназначена для формирования заказов. Интерфейс предоставляет возможность работнику склада просматривать информацию о заказе, создавать заказы как для заказчиков, так и для дозакупки товаров, удалять и изменять заказы.

Разработка такой базы данных является актуальной, поскольку даёт возможность администрации склада контролировать клиентскую базу и грамотно распределять рабочие ресурсы. Данная база данных упрощает работу, выполняемую работниками склада.

Интерфейс разрабатывался в соответствии со следующими требованиями:

* наличие пользователя-администратора и пользователя-работника;
* простота и понятность обычному пользователю;
* возможность просмотра личных данных пользователем-работником;
* возможность просмотра списка всех клиентов пользователем-

администратором;

* лаконичность интерфейса.

На данный момент существует множество аналогов такой базы данных. Можно найти множество различных интерфейсов для учета товаров на складе в интернете.

При разработке интерфейса для базы данных "Снабжение" учтены некоторые недостатки имеющихся аналогов, что позволило повысить универсальность разрабатываемого продукта.

**1 Анализ и проектирование базы данных**

**1.1 Исследование предметной области**

Предметной областью разрабатываемого программного продукта является формирование заказа для клиентов и формирование заказа для склада.

База данных «Снабжение» предназначена для упрощения обработки информации о клиентах, которые заказывают товары со склада.

Эта база данных предназначена для менеджеров (работников склада), которые заносят информацию в базу, регистрируя и работая с клиентами. Но также может использоваться людьми, имеющими непосредственное отношение к работе БД, такими как вышестоящее руководство.

База данных выполняет следующие функции:

1. Хранение информации о сотрудниках
2. Хранение информации о клиентах
3. Хранение информации о поставщиках
4. Хранение информации о заказах

Данные требования к разрабатываемой базе данных учтены при создании модели, а также доработаны в соответствии с возникающими проблемами при проектировании.

После рассмотрения предметной области было решено создать тринадцать таблиц: «Заказ», «Заказ склада», «Продукт», «Продукт на складе», «Продукт заказа», «Продукт заказа склада», «Магазин», «Пользователь», «Склад», «Сеть магазинов»

Схема связи таблиц представлена на рисунке 1.

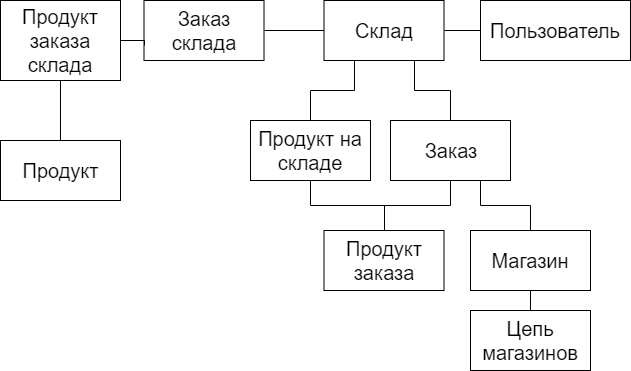


Рисунок 1 - Схема связи таблиц

**1.2 Создание модели в Oracle SQL Developer Data Modeler**

После исследования предметной области перешел к созданию модели в Oracle SQL Developer Data Modeler.

Для хранения баланса на складе, а так же учета сотрудников склада создана таблица «Склад». Она содержит информацию о балансе склада, администраторе склада, полученным из таблицы «Пользователь» и его названии.

Так как необходимо учитывать работников склада была создана таблица «Пользователь». Она содержит логин и пароль пользователя, с помощью которых он будет заходить в систему. Помимо этого таблица содержит данные о пользователе – его ФИО и текущий статус (работник склада «Р» или администратор «А»).

Основное назначение базы данных – формирование заказов, поэтому была создана таблица «Заказ». В таблице содержится информация о магазине, информация о котором получена из таблицы «Магазин», складе, с которого заказ будет приниматься, текущий статус заказа (Закрыт «C», Открыт «O», предоплата в 100% «П» - при заказе от 20к рублей и предоплата в 50% «H» - при заказе от 10к рублей). Помимо этого учитывается дата создания и закрытия заказа.

Для заказа так же необходимо учитывать продукты, которые в него входят. Для этого создана таблица «Продукт заказа», в которой содержится информация о продукте со склада, полученным из таблицы «Продукт склада», количестве заказанных продуктов, а так же номеру заказа, полученному из таблицы «Заказ», который необходим для реализации связи один-много.

Из предыдущей таблицы появляется необходимость создания следующей: таблицы «Продукт склада», которая содержит информацию о названии продукта, его количестве на складе, цене, а так же идентификатор склада, о продукте которого ведется описание. Идентификатор берется из таблицы «Склад».

Так как заказчиков у склада может быть много, решено создать таблицу «Цепь магазинов» которая объединяет некоторые магазины. В этой таблице хранится информация об названии цепи, например «Пятерочка».

Для учета заказчиков создана таблица «Магазин», в которой хранится информация о цепи магазинов, информация о которых берется из таблицы «Цепь магазинов», к которой магазин принадлежит, а так же его название и адрес.

Все следующие таблицы созданы для хранения информации о поставках на склад.

Таблица «Заказ склада» имеет информацию о складе, информация о котором берется из таблицы «Склад», на который завозят товары, текущем статусе заказа (Закрыт «C» и Открыт «О»), дате создания и закрытия заказа.

Таблица «Продукт заказа склада» содержит информацию о заказываемом продукте, информация о котором берется из таблицы «Продукт», количестве этого продукта, а так же информация о заказе, которая берется из таблицы «Заказ склада»

Таблица «Продукт» содержит информацию о названии и цене продукта.

Модель разработанной базы данных "Снабжение" представлена на рисунке

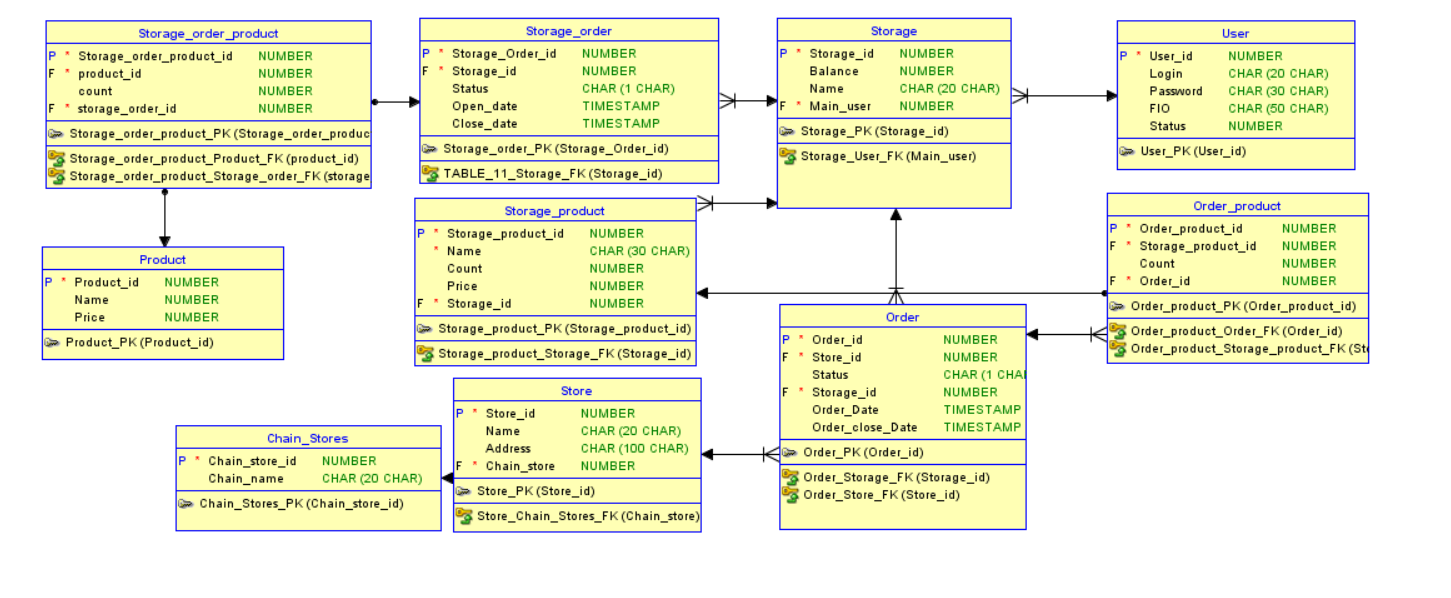


Рисунок 2 - Модель разработанной базы данных

Таблицей называется элемент, который представлен на рисунке 3.

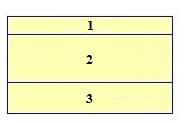


Рисунок 3 - Таблица

В данной таблице присутствуют следующие обозначения:

1. Название таблицы;
2. Описание столбцов таблицы (название и тип данных);
3. Описание ключей, связывающих данную таблицу с другими таблицами.

В представленной модели таблицы переименованы. Теперь их название:

* «Заказ» переименован в «Order»;
* «Заказ склада» переименован в «Storage\_order»;
* «Продукт» переименован в «Product»;
* «Продукт на складе» переименован в «Storage\_product»;
* «Продукт заказа» переименован в «Order\_product»;
* «Продукт заказа склада» переименован в «Storage\_order\_product»;
* «Магазин» переименован в «Store»;
* «Пользователь» переименован в «User»;
* «Склад» переименован в «Storage»;
* «Сеть магазинов» переименован в «Chain\_Stores».

На данном рисунке присутствует множество стрелок, соединяющих таблицы.

Стрелка, соединяющая таблицы «Chain\_stores» и «Store», является идентификацией первичного ключа «Chain\_store\_id» таблицы «Chain\_stores» и внешнего ключа «Chain\_store» таблицы «Store».

Стрелка, соединяющая таблицы «Store» и «Order», является идентификацией первичного ключа «Store\_id» таблицы «Store» и внешнего ключа «Store\_id» таблицы «Order».

Стрелка, соединяющая таблицы «Order» и «Order\_product», является идентификацией первичного ключа «Order\_id» таблицы «Order» и внешнего ключа «Order\_id» таблицы «Order\_product».

Стрелка, соединяющая таблицы «Order\_product» и «Storage\_product», является идентификацией первичного ключа «Storage\_product\_id» таблицы «Storage\_product» и внешнего ключа «Storage\_product\_id» таблицы «Order\_product».

Стрелка, соединяющая таблицы «Storage» и «Order», является идентификацией первичного ключа «Storage\_id» таблицы «Storage» и внешнего ключа «Storage\_id» таблицы «Order».

Стрелка, соединяющая таблицы «Storage» и «Storage\_product», является идентификацией первичного ключа «Storage\_id» таблицы «Storage» и внешнего ключа «Storage\_id» таблицы «Storage\_product».

Стрелка, соединяющая таблицы «Storage» и «Storage\_order», является идентификацией первичного ключа «Storage\_id» таблицы «Storage» и внешнего ключа «Storage\_id» таблицы «Storage\_order».

Стрелка, соединяющая таблицы «Storage\_order» и «Storage\_order\_product», является идентификацией первичного ключа «Storage\_order\_id» таблицы «Storage\_order» и внешнего ключа «storage\_order\_id» таблицы «Storage\_order\_product».

Стрелка, соединяющая таблицы «User» и «Storage», является идентификацией первичного ключа «User\_id» таблицы «User» и внешнего ключа «Main\_user» таблицы «Storage».

Стрелка, соединяющая таблицы «Storage\_order\_product» и «product», является идентификацией первичного ключа «product\_id» таблицы «product» и внешнего ключа «product\_id» таблицы «Storage\_order\_product».

1. **Формирование базы данных**

**2.1 Создание пользователя для базы данных**

Создание пользователя происходило с помощью графического интерфейса SQL Developer, т.к. данный способ создания позволяет видеть все права, которыми может обладать пользователь, и выбирать их из списка.

Было принято решение наделить пользователя следующими правами доступа:

* + возможность создания, изменения и удаления таблиц;
  + возможность создания, изменения и удаления процедур;
  + возможность создания, изменения и удаления триггеров;
  + возможность создания представлений;
  + возможность создания, изменения и удаления счетчиков;
  + возможность создания сессий;
  + возможность изменения пользователя.

**2.2 Перенос базы данных в Oracle Database SQL Developer**

Формирование базы данных началось с переноса схемы базы данных в Oracle Database SQL Developer. Для этого было необходимо получить схему базы данных в виде SQL кода и выполнить его в схеме пользователя, созданного предварительно в Oracle Database Express Edition 11g. SQL-код, сгенерированный SQL Data Modeler, и результат переноса представлены в приложении А.

**2.3 Добавление данных в таблицы**

Было решено первыми заполнить таблицы, которые не зависят от данных

* других таблицах. Сначала были заполнены таблицы «Product»,

«User », «Chain\_stores», «store», «Storage».

Для добавления данных в таблицы было решено использовать SQL.[6] В схеме пользователя был написан и отлажен оператор INSERT, который создаёт записи в таблице. В листинге 1 представлено добавление записи в таблицу "City" при помощи оператора INSERT:

Листинг 1 - Добавление записи в таблицу «*STORAGE\_PRODUCT*»

*INSERT INTO STORAGE\_PRODUCT (NAME, PRICE, COUNT, STORAGE\_ID)*

*VALUES ('Продукт 1', 35, 10, 1);*

* + данном примере команда INSERT добавляет записи в таблицу "Rooms".
* поля « ID\_CITY », «Name», « Country»

заносятся данные 1, 'Moscow', 'Russia'.

Команда INSERT за один вызов заносит в таблицу лишь одну строку, поэтому, чтобы заполнить нашу таблица нам понадобится вызвать команду столько раз, сколько строк мы планируем заполнить.

Для формирования идентификаторов у каждой таблицы необходимо создать последовательности. Для каждой таблицы были созданы последовательности, номер которых начинался с 0.

Листинг 2 – Добавление последовательности в таблицу order

*CREATE SEQUENCE order\_SEQ START WITH 0 INCREMENT BY 1 MINVALUE 0;*

Для автоматического заполнения идентификаторов таблиц были созданы триггеры, использующие последовательности для заполнения номеров.

Листинг 3 – Добавление триггера для таблицы order

CREATE OR REPLACE TRIGGER order\_TRG BEFORE

INSERT ON "Order" FOR EACH ROW

BEGIN

:NEW.order\_id := order\_SEQ.NEXTVAL;

END; **Создание интерфейса в среде Oracle APEX версии 5.1**

**3.1 Создание страницы авторизации**

Было решено создать страницу авторизации, на которой пользователю предлагается ввести свои логин и пароль.

Для осуществления авторизации была создана дополнительная таблица.

Её структура представлена на рисунке 4.

Рисунок 4 - Дополнительная таблица для создания авторизации

Для осуществления авторизации нового постояльца был создан пакет AUT. Его спецификация и тело представлено в листинге 2.

Листинг 2 - Авторизация нового постояльца

create or replace package AUT as function user\_aut (

U\_name in varchar2

,U\_password in varchar2 )

return boolean;

end;

create or replace package body "AUT" is function USER\_AUT(

U\_NAME IN VARCHAR2

,U\_PASSWORD IN VARCHAR2 ) return BOOLEAN as

i\_count integer;

begin

Select Count(\*) Into i\_count

From USER\_LOGIN

Where Upper(Trim(USER\_NAME)) = Upper(Trim(U\_NAME)) and Trim(USERPASS) = Trim(U\_PASSWORD);

If i\_count > 0 Then return True; Else return False; End If;

end USER\_AUT;

end "AUT";

Спецификация описывает прототип функции user\_aut, результатом выполнения которого является логическое значение, зависящее от наличия пользователя в таблице «USER\_LOGIN» (возвращает TRUE, если пользователь найден, FALSE - в противном случае).

* + теле функции осуществляется сравнение введенного имени пользователя в поле «Username» с полем «USER\_NAME» и введенного пароля
* поле «Password» с полем «USERPASS».

Для авторизации необходимо было создать страницу, в которой присутствуют поля для ввода логина и пароля и кнопка «Login». Интерфейс страницы авторизации представлен на рисунке 5.



Рисунок 5 - Страница авторизации

Была создана схема авторизации на основе таблицы «USER\_LOGIN» отдельно для пользователя и для администратора. Это связано с тем, что администратор обладает следующими правами: возможность вносить коррективы в таблицы, исправлять данные и пр. Страница Main\_H имеет два интерфейса для пользователя и администратора.

**3.2. Создание страницы Администратора**

Для администратора системы предусмотрена страница управления пользователями (рис. 2), которая включают в себя выбор прав, достаточных для входа в приложение, отчёт со списком пользователей, которые имеют привилегию на вход в приложение, функцию создания таких пользователей, кнопки для добавления новых пользователей, для удаления пользователей и для сохранения изменений. При нажатии кнопки сохранения изменений новые данные заносятся в базу.

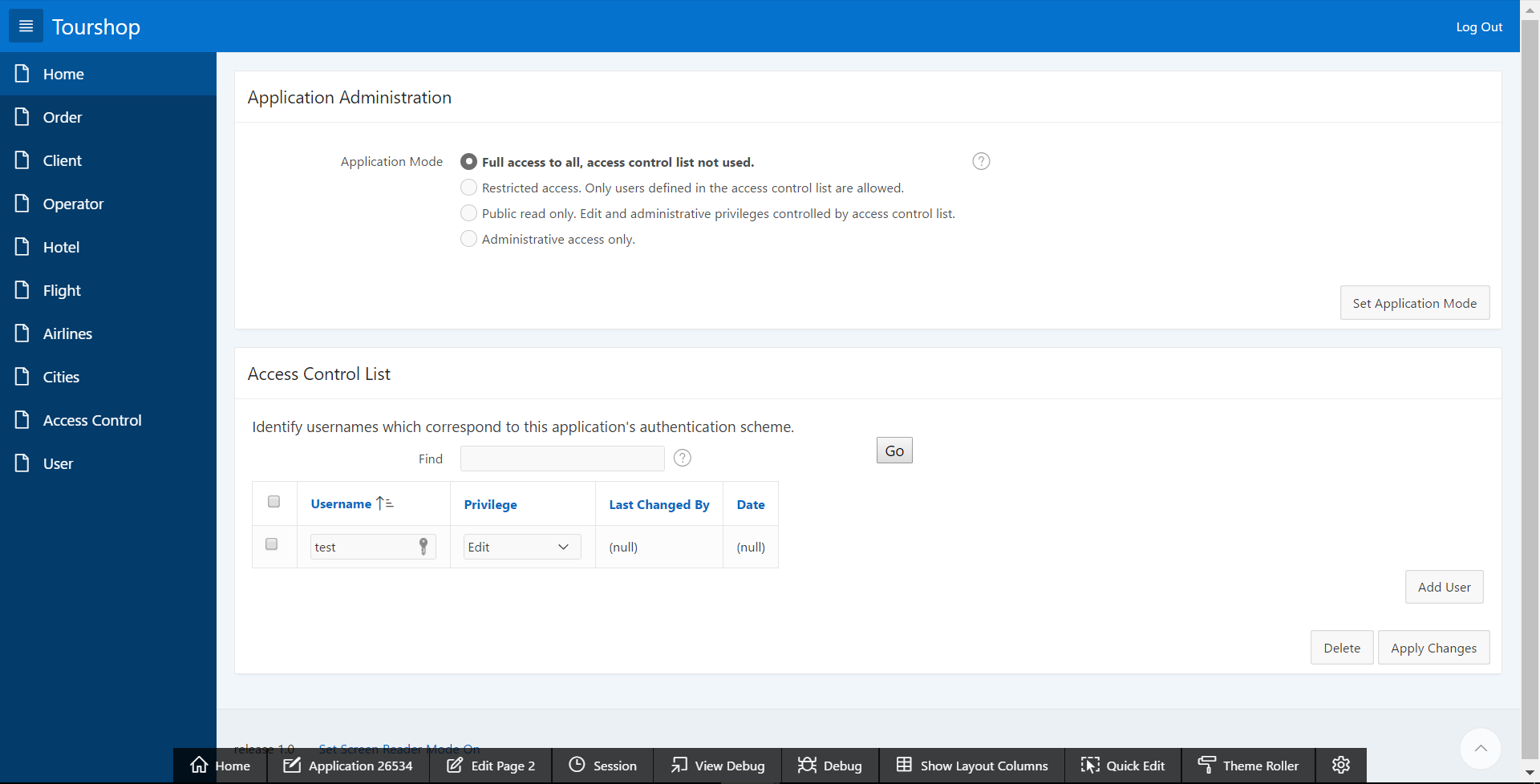
****

Рисунок 5 - Страница управления пользователями

**3.3. Страница входа**

Страница входа в систему (рис.3) состоит из формы для ввода имени учётной записи и пароля и выпадающего списка, позволяющего выбрать тип учётной записи, администратор или пользователь с ограниченными правами.

При выборе типа учётной записи срабатывает динамическое действие, при котором заполняются элементы <> и заносятся в текущую сессию. Это позволяет обеспечить заполнение страниц приложения в соответствии с правами пользователя.

Также страница содержит кнопку входа “Log In”, запускающая проверку введённых данных на существование в таблице пользователей. Если пользователя, соответствующего введённым данным, не существует, то вход не будет произведён. Если проверка пройдена, то пользователь получает доступ к приложению в рамках своих прав.

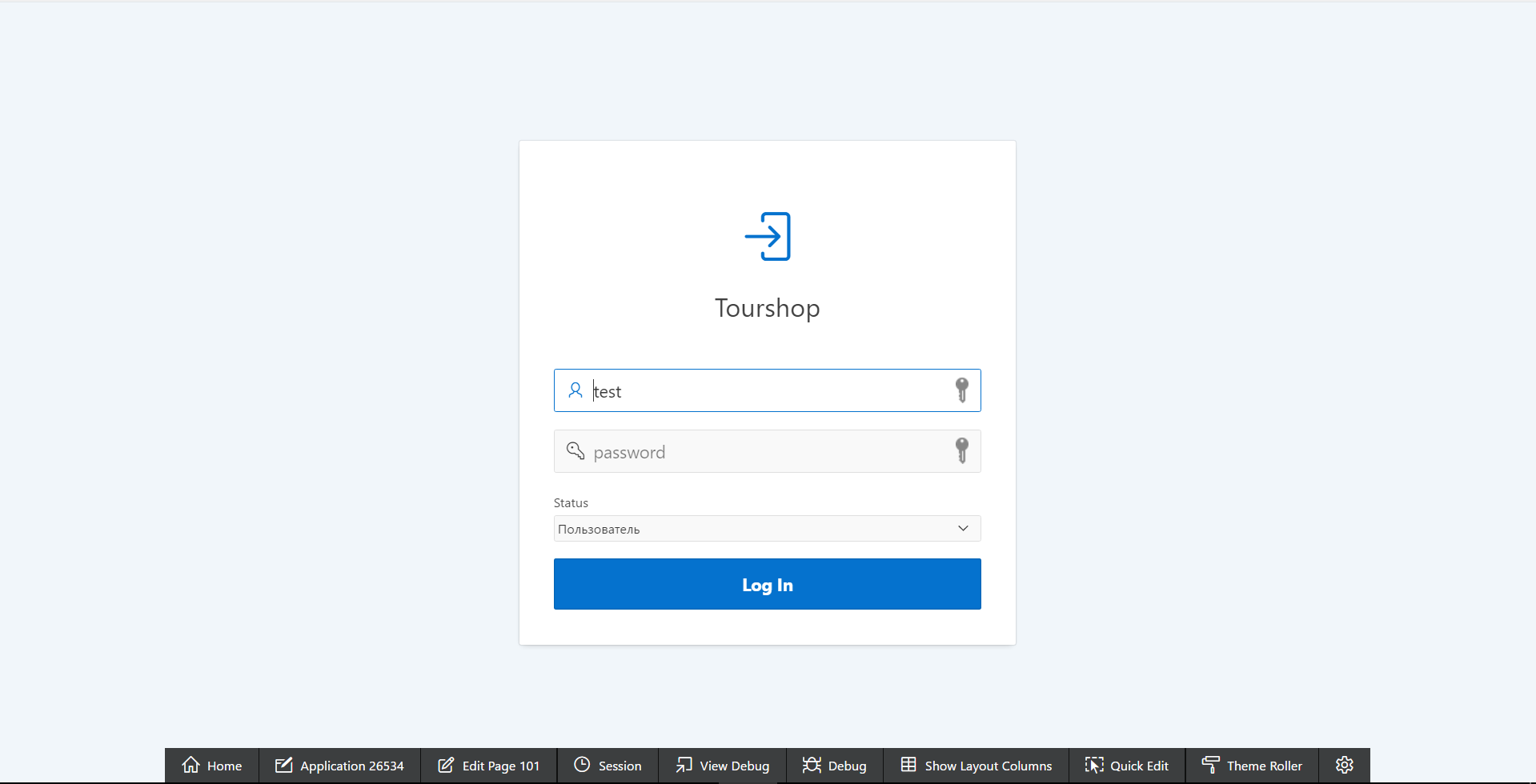


Рисунок 7 - Страница входа

**ЗАКЛЮЧЕНИЕ**

* результате выполнения курсового проекта была создана система заказа туров.

Данная система основана на базе данных, реализованной при помощи программных средств СУБД MySQL, и имеет пользовательский интерфейс, созданный на основе APEX Oracle.

В процессе реализации проекта были выполнены:

— Исследование предметной области, ознакомление с существующими аналогами и разработка концепции системы;

— Построение схемы объектов базы данных при помощи Oracle Data

— Генерация SQL-кодов по созданной модели, исправление ошибок генерации и необходимая доработка в соответствии с особенностями СУБД

— Заполнение базы данных;

— Реализация пользовательского интерфейса с помощью APEX Oracle;

— Проверка функционирования всех форм интерфейса.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Райордан Р, Основы реляционных баз данных [Текст] / Ребекка Райордан: Основы реляционных баз данных - Microsoft Press Русская редакция, 2012, 390с. - ISBN 5-7502-0150-3
2. Грофф Джеймс Р., SQL Полное руководство [Текст] / Джеймс Р. Грофф, SQL Полное руководство - Издательский дом "Вильямс", 2015, 390с. - ISBN 978-5-8459-1654-9
3. Дейт К. Дж., Введение в системы баз данных [Текст] / К. Дж. Дейт, Введение в системы баз данных, 8-е издание - Издательский дом

"Вильямс", 2005, 1316с. - ISBN 5-8459-0788-8

1. MySQL. Руководство администратора. - М.: Вильямс, 2005. - 621 с.
2. Иванова Г.С., Ничушкина Т.Н. Оформление текстовых документов. Методические указания по оформлению расчетно-пояснительных записок дипломных и квалификационных работ. - М.: Издательство МГТУ им.

Н.Э. Баумана, 2004. - 10 с.

1. Молинаро Э., SQL. Сборник рецептов [Текст] / Энтони Молинаро, SQL. Сборник рецептов - СПб: Символ-Плюс, 2009, 666с. - ISBN 5-93286-125-8
2. Борисов В.Ю. Методические рекомендации к выполнению курсовой работы / В.Ю. Борисов. - М.: Нобель Пресс, 2013. - 988 с

**Приложение А**

**SQL-код, сгенерированный SQL Data Modeler**

CREATE TABLE "SOC\_STATUS"

( "ID\_STATUS" NUMBER(\*,0) NOT NULL ENABLE,

"STATUS" CHAR(25),

CONSTRAINT "SOC\_STATUS\_PK" PRIMARY KEY ("ID\_STATUS")

USING INDEX ENABLE

)

/

CREATE TABLE "DOCUMENTS"

( "ID\_DOC" NUMBER(\*,0) NOT NULL ENABLE,

"NAME" CHAR(25),

CONSTRAINT "DOCUMENT\_PK" PRIMARY KEY ("ID\_DOC")

USING INDEX ENABLE

)

/

CREATE TABLE "CLIENT"

( "ID\_CLIENT" NUMBER(\*,0) NOT NULL ENABLE,

"SURNAME" CHAR(25),

"NAME" CHAR(20),

"PATRONYMIC" CHAR(25),

"BIRTH" DATE,

"ID\_DOC" NUMBER(\*,0) NOT NULL ENABLE,

"NUM\_DOC" NUMBER(\*,0),

"PHONE" CHAR(10),

"EMAIL" CHAR(25),

"SOC\_STATUS" NUMBER(\*,0) NOT NULL ENABLE,

CONSTRAINT "CLIENT\_PK" PRIMARY KEY ("ID\_CLIENT")

USING INDEX ENABLE

)

/

CREATE TABLE "OPERATOR"

( "ID\_OPERATOR" NUMBER(\*,0) NOT NULL ENABLE,

"NAME" CHAR(20),

"SURNAME" CHAR(20),

"PATRONYMIC" CHAR(20),

"ID\_DOC" NUMBER(\*,0) NOT NULL ENABLE,

"NUM\_DOC" CHAR(15),

"REG\_NUMBER" CHAR(15),

"ADRESS" CHAR(25),

"EMAIL" CHAR(15),

CONSTRAINT "OPERATOR\_PK" PRIMARY KEY ("ID\_OPERATOR")

USING INDEX ENABLE

)

/

CREATE TABLE "AIRLINES"

( "ID\_AIRLINE" NUMBER(\*,0) NOT NULL ENABLE,

"NAME" CHAR(15),

CONSTRAINT "AIRLINES\_PK" PRIMARY KEY ("ID\_AIRLINE")

USING INDEX ENABLE

)

/

CREATE TABLE "FLIGHT"

( "ID\_FLIGHT" NUMBER(\*,0) NOT NULL ENABLE,

"NUMBER\_FLIGHT" CHAR(6),

"DATE\_FLIGHT" DATE,

"AIR\_ARRIVAL" NUMBER(\*,0) NOT NULL ENABLE,

"AIR\_DEPART" NUMBER(\*,0) NOT NULL ENABLE,

"AIRLINES" NUMBER(\*,0) NOT NULL ENABLE,

CONSTRAINT "FLIGHT\_PK" PRIMARY KEY ("ID\_FLIGHT")

USING INDEX ENABLE

)

/

CREATE TABLE "SALES"

( "ID\_SALE" NUMBER(\*,0) NOT NULL ENABLE,

"NUM\_SALE" NUMBER(\*,0),

"SOC\_STATUS" NUMBER(\*,0) NOT NULL ENABLE,

CONSTRAINT "SALES\_PK" PRIMARY KEY ("ID\_SALE")

USING INDEX ENABLE

)

/

CREATE TABLE "FOOD"

( "ID\_FOOD" NUMBER(\*,0) NOT NULL ENABLE,

"TYPE\_FOOD" CHAR(20),

"PRICE\_FOOD" NUMBER NOT NULL ENABLE,

CONSTRAINT "FOOD\_PK" PRIMARY KEY ("ID\_FOOD")

USING INDEX ENABLE

)

/

CREATE TABLE "HOTEL"

( "ID\_HOTEL" NUMBER(\*,0) NOT NULL ENABLE,

"NAME\_HOTEL" CHAR(20),

"TYPE\_HOTEL" NUMBER(\*,0),

CONSTRAINT "HOTEL\_PK" PRIMARY KEY ("ID\_HOTEL")

USING INDEX ENABLE

)

/

CREATE TABLE "ACCOMODATION"

( "ID\_ROOM" NUMBER(\*,0) NOT NULL ENABLE,

"TYPE\_ROOM" CHAR(20),

"NUM\_PERSON" NUMBER(\*,0),

"PRICE\_ROOM" NUMBER NOT NULL ENABLE,

CONSTRAINT "ACCOMADATION\_PK" PRIMARY KEY ("ID\_ROOM")

USING INDEX ENABLE

)

/

CREATE TABLE "Order"

( "ID\_ORDER" NUMBER(\*,0) NOT NULL ENABLE,

"ID\_OPERATOR" NUMBER(\*,0) NOT NULL ENABLE,

"ID\_CLIENT" NUMBER(\*,0) NOT NULL ENABLE,

"VISA" CHAR(9),

"DATE\_ORDER" DATE,

"ID\_HOTEL" NUMBER(\*,0) NOT NULL ENABLE,

"ID\_FOOD" NUMBER(\*,0) NOT NULL ENABLE,

"ID\_ROOM" NUMBER(\*,0) NOT NULL ENABLE,

"FLIGHT\_TO" NUMBER(\*,0) NOT NULL ENABLE,

"FLIGHT\_FROM" NUMBER(\*,0) NOT NULL ENABLE,

"DATE\_DEPART" TIMESTAMP (6),

"DATE\_ARRIVAL" TIMESTAMP (6),

"ID\_SALE" NUMBER(\*,0) NOT NULL ENABLE,

"DAYS" NUMBER NOT NULL ENABLE,

"TRIPS\_PRICE" NUMBER,

CONSTRAINT "ORDER\_PK" PRIMARY KEY ("ID\_ORDER")

USING INDEX ENABLE

)

/

CREATE TABLE "CITY"

( "ID\_CITY" NUMBER(\*,0) NOT NULL ENABLE,

"NAME" CHAR(25),

"COUNTRY" CHAR(25),

CONSTRAINT "CITY\_PK" PRIMARY KEY ("ID\_CITY")

USING INDEX ENABLE

)

/

CREATE TABLE "AIRPORT"

( "ID\_AIR" NUMBER(\*,0) NOT NULL ENABLE,

"NAME" CHAR(25),

"CITY" NUMBER(\*,0) NOT NULL ENABLE,

"IATA" CHAR(5),

CONSTRAINT "AIR\_PK" PRIMARY KEY ("ID\_AIR")

USING INDEX ENABLE

)

/

CREATE TABLE "APEX\_ACCESS\_SETUP"

( "ID" NUMBER,

"APPLICATION\_MODE" VARCHAR2(255),

"APPLICATION\_ID" NUMBER,

CONSTRAINT "APEX\_ACCESS\_SETUP\_PK" PRIMARY KEY ("ID")

USING INDEX ENABLE,

CONSTRAINT "APEX\_ACCESS\_SETUP\_UK" UNIQUE ("APPLICATION\_ID")

USING INDEX ENABLE

)

/

CREATE TABLE "APEX\_ACCESS\_CONTROL"

( "ID" NUMBER,

"ADMIN\_USERNAME" VARCHAR2(255),

"ADMIN\_PRIVILEGES" VARCHAR2(255),

"SETUP\_ID" NUMBER,

"CREATED\_BY" VARCHAR2(255),

"CREATED\_ON" DATE,

"UPDATED\_ON" DATE,

"UPDATED\_BY" VARCHAR2(255),

CONSTRAINT "APEX\_ACCESS\_CONTROL\_PK" PRIMARY KEY ("ID")

USING INDEX ENABLE,

CONSTRAINT "APEX\_ACCESS\_CONTROL\_UK" UNIQUE ("ADMIN\_USERNAME", "SETUP\_ID")

USING INDEX ENABLE

)

/

CREATE TABLE "HTMLDB\_PLAN\_TABLE"

( "STATEMENT\_ID" VARCHAR2(30),

"PLAN\_ID" NUMBER,

"TIMESTAMP" DATE,

"REMARKS" VARCHAR2(4000),

"OPERATION" VARCHAR2(30),

"OPTIONS" VARCHAR2(255),

"OBJECT\_NODE" VARCHAR2(128),

"OBJECT\_OWNER" VARCHAR2(128),

"OBJECT\_NAME" VARCHAR2(128),

"OBJECT\_ALIAS" VARCHAR2(261),

"OBJECT\_INSTANCE" NUMBER(\*,0),

"OBJECT\_TYPE" VARCHAR2(128),

"OPTIMIZER" VARCHAR2(255),

"SEARCH\_COLUMNS" NUMBER,

"ID" NUMBER(\*,0),

"PARENT\_ID" NUMBER(\*,0),

"DEPTH" NUMBER(\*,0),

"POSITION" NUMBER(\*,0),

"COST" NUMBER(\*,0),

"CARDINALITY" NUMBER(\*,0),

"BYTES" NUMBER(\*,0),

"OTHER\_TAG" VARCHAR2(255),

"PARTITION\_START" VARCHAR2(255),

"PARTITION\_STOP" VARCHAR2(255),

"PARTITION\_ID" NUMBER(\*,0),

"OTHER" LONG,

"DISTRIBUTION" VARCHAR2(30),

"CPU\_COST" NUMBER(\*,0),

"IO\_COST" NUMBER(\*,0),

"TEMP\_SPACE" NUMBER(\*,0),

"ACCESS\_PREDICATES" VARCHAR2(4000),

"FILTER\_PREDICATES" VARCHAR2(4000),

"PROJECTION" VARCHAR2(4000),

"TIME" NUMBER(\*,0),

"QBLOCK\_NAME" VARCHAR2(128)

)

/

CREATE TABLE "USERS"

( "ID\_USER" NUMBER GENERATED BY DEFAULT AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOT NULL ENABLE,

"U\_NAME" VARCHAR2(20),

"U\_PASSWORD" VARCHAR2(20),

"U\_ACTIVATED" NUMBER,

CONSTRAINT "USERS\_PK" PRIMARY KEY ("ID\_USER")

USING INDEX ENABLE

)

/

ALTER TABLE "AIRPORT" ADD CONSTRAINT "AIR\_CITY\_FK" FOREIGN KEY ("CITY")

REFERENCES "CITY" ("ID\_CITY") ENABLE

/

ALTER TABLE "APEX\_ACCESS\_CONTROL" ADD CONSTRAINT "APEX\_ACCESS\_CONTROL\_FK" FOREIGN KEY ("SETUP\_ID")

REFERENCES "APEX\_ACCESS\_SETUP" ("ID") ENABLE

/

ALTER TABLE "CLIENT" ADD CONSTRAINT "CLIENT\_SOC\_STATUS\_FK" FOREIGN KEY ("SOC\_STATUS")

REFERENCES "SOC\_STATUS" ("ID\_STATUS") ENABLE

/

ALTER TABLE "CLIENT" ADD CONSTRAINT "Client \_Document\_FK" FOREIGN KEY ("ID\_DOC")

REFERENCES "DOCUMENTS" ("ID\_DOC") ENABLE

/

ALTER TABLE "FLIGHT" ADD CONSTRAINT "FLIGHT\_AIRPORT\_FK" FOREIGN KEY ("AIR\_ARRIVAL")

REFERENCES "AIRPORT" ("ID\_AIR") ENABLE

/

ALTER TABLE "FLIGHT" ADD CONSTRAINT "FLIGHT\_AIRPORT\_FKV2" FOREIGN KEY ("AIR\_DEPART")

REFERENCES "AIRPORT" ("ID\_AIR") ENABLE

/

ALTER TABLE "FLIGHT" ADD CONSTRAINT "FLIGHT\_АIRLINES\_FK" FOREIGN KEY ("AIRLINES")

REFERENCES "AIRLINES" ("ID\_AIRLINE") ENABLE

/

ALTER TABLE "OPERATOR" ADD CONSTRAINT "OPERATOR\_DOCUMENTS\_FK" FOREIGN KEY ("ID\_DOC")

REFERENCES "DOCUMENTS" ("ID\_DOC") ENABLE

/

ALTER TABLE "Order" ADD CONSTRAINT "ORDER\_ACCOMADATION\_FK" FOREIGN KEY ("ID\_ROOM")

REFERENCES "ACCOMODATION" ("ID\_ROOM") ENABLE

/

ALTER TABLE "Order" ADD CONSTRAINT "ORDER\_CLIENT\_FK" FOREIGN KEY ("ID\_CLIENT")

REFERENCES "CLIENT" ("ID\_CLIENT") ENABLE

/

ALTER TABLE "Order" ADD CONSTRAINT "ORDER\_FLIGHT\_FK" FOREIGN KEY ("FLIGHT\_TO")

REFERENCES "FLIGHT" ("ID\_FLIGHT") ENABLE

/

ALTER TABLE "Order" ADD CONSTRAINT "ORDER\_FLIGHT\_FKV1" FOREIGN KEY ("FLIGHT\_FROM")

REFERENCES "FLIGHT" ("ID\_FLIGHT") ENABLE

/

ALTER TABLE "Order" ADD CONSTRAINT "ORDER\_FOOD\_FK" FOREIGN KEY ("ID\_FOOD")

REFERENCES "FOOD" ("ID\_FOOD") ENABLE

/

ALTER TABLE "Order" ADD CONSTRAINT "ORDER\_HOTEL\_FK" FOREIGN KEY ("ID\_HOTEL")

REFERENCES "HOTEL" ("ID\_HOTEL") ENABLE

/

ALTER TABLE "Order" ADD CONSTRAINT "ORDER\_OPERATOR\_FK" FOREIGN KEY ("ID\_OPERATOR")

REFERENCES "OPERATOR" ("ID\_OPERATOR") ENABLE

/

ALTER TABLE "Order" ADD CONSTRAINT "ORDER\_SALE\_FK" FOREIGN KEY ("ID\_SALE")

REFERENCES "SALES" ("ID\_SALE") ENABLE

/

ALTER TABLE "SALES" ADD CONSTRAINT "SALES\_SOC\_STATUS\_FK" FOREIGN KEY ("SOC\_STATUS")

REFERENCES "SOC\_STATUS" ("ID\_STATUS") ENABLE

/

CREATE SEQUENCE "ACCOMADATION\_ID\_ROOM\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "ACCOMODATION\_ID\_ROOM\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "AIRLINES\_ID\_AIRLINE\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 6 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "AIRPORT\_ID\_AIR\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "APEX\_ACCESS\_CONTROL\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "CITY\_ID\_CITY\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 8 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "CLIENT\_ID\_CLIENT\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 11 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "DEMO\_ORDER\_ITEMS\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 160 CACHE 20 NOORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "DEMO\_ORD\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 11 CACHE 20 NOORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "APEX$\_WS\_SEQ" MINVALUE 100 MAXVALUE 999999999999999999999999999 INCREMENT BY 1 START WITH 100 CACHE 20 NOORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "DEMO\_CUST\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 100 CACHE 20 NOORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "DEMO\_PROD\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 100 CACHE 20 NOORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "DEPT\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 50 CACHE 20 NOORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "DOCUMENTS\_ID\_DOC\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "EMP\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 8000 CACHE 20 NOORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "FLIGHT\_ID\_FLIGHT\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 12 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "FOOD\_ID\_FOOD\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "HOTEL\_ID\_HOTEL\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 5 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "OPERATOR\_ID\_OPERATOR\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 3 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "ORDER\_ID\_ORDER\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 2 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "SALES\_ID\_SALE\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "SOC\_STATUS\_ID\_STATUS\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE NOPARTITION

/

CREATE SEQUENCE "USERS\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOPARTITION

/

CREATE OR REPLACE EDITIONABLE TRIGGER "ACCOMODATION\_ID\_ROOM\_TRG" BEFORE

INSERT ON Accomodation FOR EACH ROW WHEN (NEW.ID\_Room IS NULL) BEGIN :NEW.ID\_Room := Accomodation\_ID\_Room\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "ACCOMODATION\_ID\_ROOM\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "AIRLINES\_ID\_AIRLINE\_TRG" BEFORE

INSERT ON Airlines FOR EACH ROW WHEN (NEW.ID\_Airline IS NULL) BEGIN :NEW.ID\_Airline := Airlines\_ID\_Airline\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "AIRLINES\_ID\_AIRLINE\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "AIRPORT\_ID\_AIR\_TRG" BEFORE

INSERT ON Airport FOR EACH ROW WHEN (NEW.ID\_Air IS NULL) BEGIN :NEW.ID\_Air := Airport\_ID\_Air\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "AIRPORT\_ID\_AIR\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "CITY\_ID\_CITY\_TRG" BEFORE

INSERT ON City FOR EACH ROW WHEN (NEW.ID\_City IS NULL) BEGIN :NEW.ID\_City := City\_ID\_City\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "CITY\_ID\_CITY\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "CLIENT\_ID\_CLIENT\_TRG" BEFORE

INSERT ON Client FOR EACH ROW WHEN (NEW.ID\_Client IS NULL) BEGIN :NEW.ID\_Client := Client\_ID\_Client\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "CLIENT\_ID\_CLIENT\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "DOCUMENTS\_ID\_DOC\_TRG" BEFORE

INSERT ON Documents FOR EACH ROW WHEN (NEW.ID\_Doc IS NULL) BEGIN :NEW.ID\_Doc := Documents\_ID\_Doc\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "DOCUMENTS\_ID\_DOC\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "FLIGHT\_ID\_FLIGHT\_TRG" BEFORE

INSERT ON Flight FOR EACH ROW WHEN (NEW.ID\_Flight IS NULL) BEGIN :NEW.ID\_Flight := Flight\_ID\_Flight\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "FLIGHT\_ID\_FLIGHT\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "FOOD\_ID\_FOOD\_TRG" BEFORE

INSERT ON Food FOR EACH ROW WHEN (NEW.ID\_Food IS NULL) BEGIN :NEW.ID\_Food := Food\_ID\_Food\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "FOOD\_ID\_FOOD\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "HOTEL\_ID\_HOTEL\_TRG" BEFORE

INSERT ON Hotel FOR EACH ROW WHEN (NEW.ID\_Hotel IS NULL) BEGIN :NEW.ID\_Hotel := Hotel\_ID\_Hotel\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "HOTEL\_ID\_HOTEL\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "OPERATOR\_ID\_OPERATOR\_TRG" BEFORE

INSERT ON Operator FOR EACH ROW WHEN (NEW.ID\_Operator IS NULL) BEGIN :NEW.ID\_Operator := Operator\_ID\_Operator\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "OPERATOR\_ID\_OPERATOR\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "ORDER\_ID\_ORDER\_TRG" BEFORE

INSERT ON "Order" FOR EACH ROW WHEN (NEW.ID\_Order IS NULL) BEGIN :NEW.ID\_Order := Order\_ID\_Order\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "ORDER\_ID\_ORDER\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "SALES\_ID\_SALE\_TRG" BEFORE

INSERT ON Sales FOR EACH ROW WHEN (NEW.ID\_Sale IS NULL) BEGIN :NEW.ID\_Sale := Sales\_ID\_Sale\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "SALES\_ID\_SALE\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "SOC\_STATUS\_ID\_STATUS\_TRG" BEFORE

INSERT ON Soc\_Status FOR EACH ROW WHEN (NEW.ID\_Status IS NULL) BEGIN :NEW.ID\_Status := Soc\_Status\_ID\_Status\_SEQ.NEXTVAL;

END;

/

ALTER TRIGGER "SOC\_STATUS\_ID\_STATUS\_TRG" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "BI\_APEX\_ACCESS\_CONTROL"

before insert or update on apex\_access\_control

for each row

begin

if inserting and :new.id is null then

select apex\_access\_control\_seq.nextval into :new.id from sys.dual;

end if;

if inserting then

:new.created\_by := v('USER');

:new.created\_on := sysdate;

end if;

if updating then

:new.updated\_by := v('USER');

:new.updated\_on := sysdate;

end if;

end;

/

ALTER TRIGGER "BI\_APEX\_ACCESS\_CONTROL" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "BI\_APEX\_ACCESS\_SETUP"

before insert or update on apex\_access\_setup

for each row

begin

if inserting and :new.id is null then

select apex\_access\_control\_seq.nextval into :new.id from sys.dual;

end if;

if :new.application\_id is null then

:new.application\_id := v('APP\_ID');

end if;

end;

/

ALTER TRIGGER "BI\_APEX\_ACCESS\_SETUP" ENABLE

/

CREATE OR REPLACE EDITIONABLE TRIGGER "BI\_USERS"

before insert on "USERS"

for each row

begin

if :NEW."ID\_USER" is null then

select "USERS\_SEQ".nextval into :NEW."ID\_USER" from sys.dual;

end if;

end;

/

ALTER TRIGGER "BI\_USERS" ENABLE

/