МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

База данных для программного средства «La Rose» c технологией резервного копирования и восстановления.

Выполнил студент Почиковская Юлия Сергеевна

(Ф.И.О.)

Руководитель проекта асс. Колмаков М. В.

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

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

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

Консультанты асс. Колмаков М. В.

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

Нормоконтролер асс. Колмаков М. В.

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

Курсовой проект защищен с оценкой

Минск 2021

[**Введение** 3](#_Toc58591928)

[**Постановка задачи** 4](#_Toc58591929)

[**1. Разработка модели базы данных** 5](#_Toc58591930)

[**2. Разработка необходимых объектов** 8](#_Toc58591931)

[**2.1. Таблицы** 8](#_Toc58591932)

[**2.2. Пользователи** 8](#_Toc58591933)

[**2.3. Представления** 8](#_Toc58591934)

[**2.4. Процедуры** 9](#_Toc58591935)

[**2.5. Функции** 9](#_Toc58591936)

[**2.6. Синонимы** 10](#_Toc58591937)

[**2.7. Триггеры** 10](#_Toc58591938)

[**2.8. Индексы** 11](#_Toc58591939)

[**2.9. Исключения** 12](#_Toc58591940)

[**3. Описание процедур импорта и экспорта данных** 13](#_Toc58591941)

[**4. Технология резервного копирования и восстановления** 14](#_Toc58591942)

[**5. Тестирование производительности базы данных** 16](#_Toc58591943)

[**6. Руководство пользователя** 18](#_Toc58591944)

[**7. Проверка работоспособности** 19](#_Toc58591945)

[**Заключение** 20](#_Toc58591946)

[**Список использованной литературы** 21](#_Toc58591947)

[**Приложение А** 22](#_Toc58591948)

[**Приложение Б** 26](#_Toc58591949)

[**Приложение В** 29](#_Toc58591950)

[**Приложение Г** 31](#_Toc58591951)

[**Приложение Д** 66](#_Toc58591952)

[**Приложение Е** 72](#_Toc58591953)

# **Введение**

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

Основные функции СУБД:

* Определение структуры создаваемой базы данных, ее инициализация и проведение начальной загрузки;
* Предоставление пользователям возможности манипулирования данными (выборка необходимых данных, выполнение вычислений, разработка интерфейса ввода/вывода, визуализация);
* Обеспечение логической и физической независимости данных;
* Защита логической целостности базы данных;
* Защита физической целостности;
* Управление полномочиями пользователей на доступ к базе данных;
* Синхронизация работы нескольких пользователей;
* Управление ресурсами среды хранения;
* Поддержка деятельности системного персонала.

Обычно современная СУБД содержит следующие компоненты:

* Ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию;
* Процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода;
* Подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД [1].

СУБД существует огромное множество: Oracle, MS SQL Server, Microsoft Access, MySql и так далее. В данной работе будет использовано решение Oracle.

# **Постановка задачи**

На сегодняшний день существует большое количество систем управления базами данных. Одной из самых распространённых СУБД является Oracle.

Рассмотрим основные характеристики СУБД Oracle, которую я выбрала для выполнения данного курсового проекта.

* Возможность создания пользовательских типов данных;
* Множество диалектов для написания процедур и функций;
* Разные типы индексов;
* Полная поддержка транзакционности;
* Высокая производительность и надёжность;
* Возможность компиляции кода.

Целью данной курсовой работы является создание электронной системы, которая аккумулирует в себе функции социальной сети, новостного портала и библиотеки роз.

Задачи курсовой работы: определить необходимые сущности, создать на основе сущностей таблицы и установить связи между ними, инкапсулировать внутреннюю логику базы данных от пользователей, добиться приемлемой скорости работы базы данных, реализовать возможность резервирования базы данных в случае сбоев, реализовать экспорт и импорт данных в определенном формате.

База данных будет обладать следующими функционалом:

* регистрация пользователей в базе данных;
* создание личного профиля;
* добавление пользователей в друзья;
* отправка сообщений другим пользователям;
* просмотр и добавление новостей;
* создание альбомов и добавление в них фотографий;
* просмотр и добавление информации о розах;
* просмотр информации о питомниках;
* добавление комментариев под фотографиями;

# **Разработка модели базы данных**

Для базы данных были разработаны 12 таблиц. Диаграмма связей таблиц для необходимой базы данных представлена в приложении А.

Таблица Users, содержит пользователей(флористов), состоит из следующих столбцов:

* ID — уникальный идентификатор;
* FIRST\_NAME — имя пользователя;
* LAST\_NAME — фамилия пользователя;
* AGE — возраст пользователя;
* EMAIL — адрес электронной почты пользователя;
* PASS — пароль пользователя;
* INFO — краткая информация о пользователе;
* AVATAR — основная фотография пользователя.

Таблица Friends, содержит пары пользователей, которые являются друзьями, состоит из следующих столбцов:

* USER\_1 — уникальный идентификатор пользователя, внешний ключ на поле ID таблицы Users;
* USER\_2 — уникальный идентификатор друга, внешний ключ на поле ID таблицы Users.

Таблица Albums, содержит альбомы пользователей, состоит из следующих столбцов:

* ID — уникальный идентификатор;
* TITLE — название альбома;
* USER\_ID — идентификатор владельца, внешний ключ на поле ID таблицы Users;
* CREATED — дата и время создания.

Таблица Photos, содержит фотографии пользователей, состоит из следующих столбцов:

* ID — уникальный идентификатор;
* PHOTO — путь к фотографии;
* ALBUM\_ID — идентификатор альбома, внешний ключ на поле ID таблицы Albums;
* CREATED — дата и время создания;
* LONGITUDE — часть координаты по которой была сделана фотография(долгота);
* LATITUDE — часть координаты по которой была сделана фотография(широта).

Таблица Comments, содержит комментарии к фотографиям, состоит из следующих столбцов:

* ID — уникальный идентификатор;
* INFO — текст комментария;
* PHOTO\_ID — идентификатор фотографии, внешний ключ на поле ID таблицы Photos;
* USER\_ID — идентификатор отправителя, внешний ключ на поле ID таблицы Users;
* CREATED — дата и время создания.

Таблица Users\_To\_Dialogs, содержит пары пользователь-диалог, состоит из следующих столбцов:

* USER\_ID — идентификатор пользователя, внешний ключ на поле ID таблицы Users;
* DIALOG\_ID — идентификатор диалога, внешний ключ на поле ID таблицы Dialogs;

Таблица Dialogs, содержит информацию о диалоге, состоит из следующих столбцов:

* ID — уникальный идентификатор;
* CREATOR\_ID — идентификатор создателя диалога, внешний ключ на поле ID таблицы Users;
* TITLE — заголовок диалога;
* CREATED — дата и время создания.

Таблица Messages, содержит сообщения пользователей, состоит из следующих столбцов:

* ID — уникальный идентификатор;
* INFO — текст сообщения;
* USER\_ID — идентификатор отправителя, внешний ключ на поле ID таблицы Users;
* DIALOG\_ID — идентификатор диалога, внешний ключ на поле ID таблицы Dialogs;
* CREATED — дата и время создания.

Таблица News, содержит новости, состоит из следующих столбцов:

* ID — уникальный идентификатор;
* TITLE — заголовок новости;
* INFO — текст новости;
* USER\_ID — идентификатор создателя новости, внешний ключ на поле ID таблицы Users;
* PHOTO — фотография новости;
* CREATED — дата и время создания.

Таблица Roses, содержит информацию о розах, состоит из следующих столбцов:

* ID — уникальный идентификатор;
* NAME — название розы;
* INFO — информация о виде этой розы;
* USER\_ID — идентификатор создателя розы, внешний ключ на поле ID таблицы Users;
* PHOTO — фотография розы;
* CREATED — дата и время создания.

Таблица Farms, содержит информацию о питомниках где растут разные виды роз, состоит из следующих столбцов:

* ID — уникальный идентификатор;
* REGION — регион питомника;
* ADDRESS — адрес офиса;
* PHONE — телефонный номер;
* CREATED — дата и время создания;
* USER\_ID — идентификатор создателя питомника, внешний ключ на поле ID таблицы Users.

Таблица Logs, содержит информацию об изменениях в других таблицах, состоит из следующих столбцов:

* ID — уникальный идентификатор;
* TABLE\_NAME — название таблицы в которой произошли изменения;
* EXECUTED\_BY\_USER — пользователь совершивший изменение;
* EVENT\_TIME — время изменения;
* OPERATION — наименование операции вследствие которой произошло изменение;
* OLD\_VALUE — старое значение записи в таблице;
* NEW\_VALUE — новое значение записи в таблице.

Скрипты для создания всех таблиц базы данных представлены в приложении А.

# **Разработка необходимых объектов**

# **2.1. Таблицы**

Таблицы являются основой любой базы данных, именно в них хранится вся информация. При проектировании базы данных было создано 12 таблиц, которые подробно описаны ранее в разделе 1, а SQL-скрипты для их создания находятся в приложении А.

# **2.2. Пользователи**

Пользователь базы данных — это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации. На каждом этапе развития БД (проектирование, реализация, эксплуатация, модернизация и развитие, полная реорганизация) с ней связаны разные категории пользователей [2].

При проектировании базы данных понадобилось 2 пользователя (клиент, осуществляющий основные операции и администратор имеющий полный доступ ко всем объектам базы данных).

Скрипты для создания клиентов базы данных, прочие операции, а также выделение привилегий на осуществление определённых операций с базой данных представлены в приложении В.

# **Представления**

Представление (view), или виртуальная таблица в базе данных Oracle Database — это специфический образ таблицы или набора таблиц, определенный оператором SELECT. Представление не существует физически как обычная таблица, являющаяся частью табличного пространства. Фактически представление создает виртуальную таблицу или подтаблицу только с теми строками и/или столбцами, которые нужно показать пользователю [3].

Представление Oracle — результат хранимого запроса, поэтому в словаре данных сохраняется только определение представления.

Представления Oracle используются в приложениях по нескольким причинам, включая перечисленные ниже:

* уменьшение сложности;
* повышение безопасности;
* повышение удобства;
* переименование столбцов таблицы;
* настройка данных для пользователей;
* защита целостности данных.

В процессе выполнения курсового проекта были созданы представления для каждой таблицы.

Представления базы данных: USERS\_VIEW, ALBUMS\_VIEW, PHOTOS\_VIEW, COMMENTS\_VIEW, FRIENDS\_VIEW, DIALOGS\_VIEW, MESSAGES\_VIEW, NEWS\_VIEW, ROSES\_VIEW, FARMS\_VIEW.

Скрипты для создания всех представлений базы данных находятся в приложении Б.

# **Процедуры**

Нередко операция с данными представляет набор инструкций, которые необходимо выполнить в определенной последовательности. И в этом случае более оптимально будет инкапсулировать все эти действия в один объект — хранимую процедуру (stored procedure).

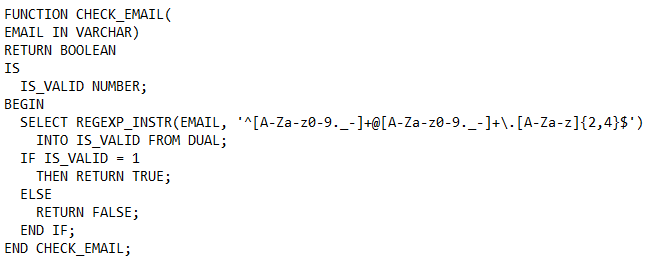
Хранимая процедура — объект базы данных, представляющий собой набор SQL — инструкций, который компилируется один раз и хранится на сервере. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML). Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения. В данном курсовом проекте хранимые процедуры предоставляют пользователю стандартный CRUD (Create, Read, Update, Delete) интерфейс для взаимодействия с таблицами. Помимо этого, есть специализированные процедуры для экспорта, импорта и логирования. Код хранимых процедур находится ниже в приложении Г.

# **Функции**

Функция — это подпрограмма, которая вычисляет значение.

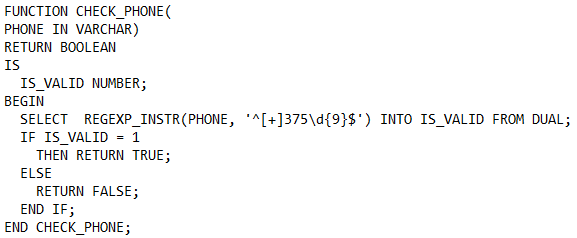
В процессе выполнения курсового проекта были созданы функции:

* CHECK\_EMAIL — функция проверки электронной почты на соответствие с регулярным выражением (листинг 2.1).



Листинг 2.1 – Функция проверки электронной почты

* CHECK\_PHONE — функция проверки номера телефона на соответствие с регулярным выражением (листинг 2.2).



Листинг 2.2 – Функция проверки номера телефона

Скрипты функций, которые взаимодействуют с конкретными таблицами приведены в приложении Г.

# **Синонимы**

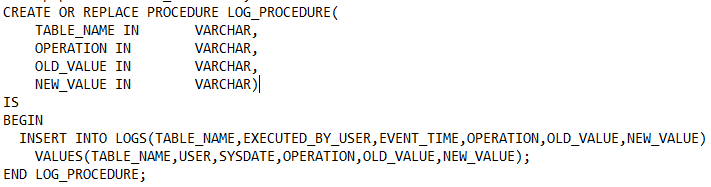
**Синонимы Oracle Database** — это псевдонимы объектов базы данных, которые служат в основном для облегчения пользователям доступа к объектам. Синонимы скрывают идентичность лежащих в их основе объектов и могут быть как приватными (private), так и общедоступными (public). Общедоступные синонимы доступны всем пользователям базы данных, а приватные синонимы являются составной частью схемы отдельного пользователя, и другим пользователям базы следует выдавать права доступа для использования приватных синонимов. Синонимы Oracle могут быть созданы для таблиц, представлений, материализованных представлений процедур [3].

В данном проекте публичные синонимы созданы для всех процедур и функций, которые доступны стандартному пользователю.SQL скрипты для создания синонимов находятся в приложении Г, рядом с объектами для которых они создаются.

# **Триггеры**

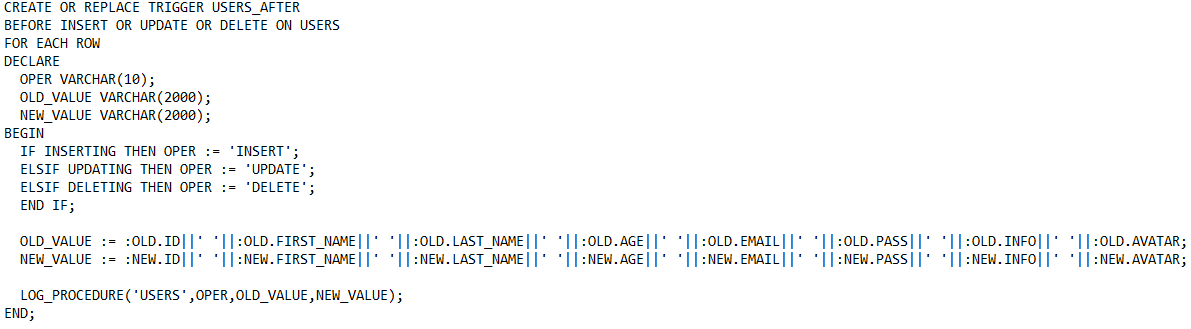
Триггер — [хранимая процедура](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0) особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определённом столбце заданной таблицы [реляционной базы данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Триггеры применяются для обеспечения целостности данных и реализации сложной [бизнес-логики](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0). Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в [транзакции](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции [4].

В данном курсовом проекте для всех таблиц были реализованы after триггеры, реагирующие на операции INSERT, UPDATE, DELETE. В результате вызова триггера он записывает произошедшие изменения в таблицу LOGS с помощью процедуры LOG\_PROCEDURE (листинг 2.3).



Листинг 2.3 – Процедура логирования

Пример SQL скрипта триггера для таблицы Users (листинг 2.4).

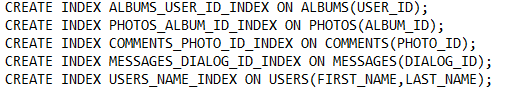


Листинг 2.4 – Триггер для таблицы Users

Все скрипты всех триггеров находятся в приложении Д.

# **Индексы**

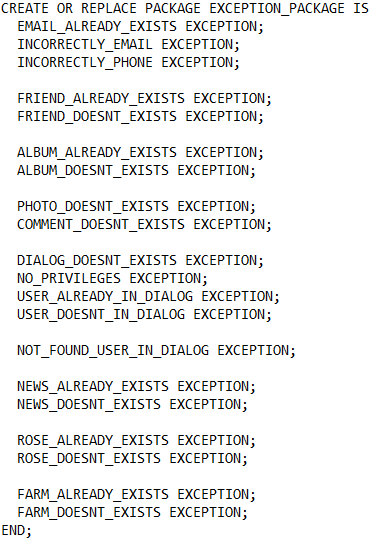
Индекс — объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. В связи с необходимостью выборки таблиц, заполненных минимум на 100 000 строк, было принято решение разработать индексы (листинг 2.5).



Листинг 2.5 – Индексы базы данных

# **Исключения**

В процессе выполнения процедур и функций могут возникать ошибки. При возникновении системной ошибки или ошибки в приложении — в PL/SQL инициируется исключение. В результате выполнение блока прерывается, и управление передается для обработки в раздел исключений текущего блока, если он имеется. После обработки исключения возврат в тот блок, где исключение было инициировано, невозможен, поэтому управление передается во внешний блок. Для более точного реагирования на ошибки в данном курсовом проекте был разработан пакет, содержащий пользовательские исключения (листинг 2.6).



Листинг 2.6 – Пакет исключений

# **Описание процедур импорта и экспорта данных**

XML — это формат данных, используемый для обмена данными в форме, которая может быть легко использована и распространена. Часто возникает необходимость импортировать и экспортировать XML-файлы в Oracle, в данной курсовой работе используются пакеты DBMS\_XMLPARSER и DBMS\_XSLPROCESSOR для импорта xml, и DBMS\_XMLDOM для экспорта данных в xml формат. Для работы с файлами так же использовался пакет DBMS\_LOB. Данные пакеты были выбраны, потому что они обладают очень широким функционалом и гибкой настройкой xml документа. В данном курсовом проекте функции экспорта, импорта используются для таблицы Users, так как эта таблица является важнейшей в базе данных. Скрипты функций импорта, экспорта представлены в приложении Е. Пример выходного xml файла (листинг 3.1).



Листинг 3.1. – Пример выходного xml файла

# **Технология резервного копирования и восстановления**

Резервное копирование — процесс создания копии данных на носителе, предназначенном для [восстановления данных](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) в оригинальном или новом месте их расположения в случае их повреждения или разрушения.

Для реализации данной технологии была использована утилита RMAN.

RMAN (Recovery Manager) — это менеджер резервного копирования и восстановления, поставляемый для [баз данных Oracle](https://en.wikipedia.org/wiki/Oracle_database) (начиная с версии 8). Он обеспечивает возможности [резервного копирования](https://en.wikipedia.org/wiki/Database_dump), восстановления и [восстановления](https://en.wikipedia.org/wiki/Data_recovery) [базы данных](https://en.wikipedia.org/wiki/Database_dump), решая проблемы высокой доступности и аварийного восстановления [5].

С помощью RMAN можно выполнять инкрементное резервное копирование. Выполнять инкрементное резервное копирование другим способом невозможно.

RMAN позволяет восстанавливать файл данных с несколькими поврежденными блоками данных в оперативном режиме, не прибегая к восстановлению из резервной копии.

В процессе работы экземпляр базы данных Oracle Database использует несколько групп файлов, которые следует архивировать для последующего восстановления:

* Файлы данных и табличных пространств (\*.DBF).
* Файлы конфигурации базы данных (\*.ORA).
* Управляющие файлы базы данных (\*.DBF).
* Файлы журналов транзакций (\*.LOG).

В данном курсовом проекте было выполнено полное горячее резервирование.

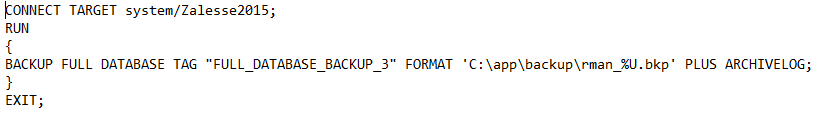
Горячее резервирование — это резервирование, при котором целевая база данных запущена в момент процесса резервирования (может выполняться только при включенном режиме архивирования журналов).

Полное резервирование включает все файлы данных, управляющий файл (controlfile) и файл серверных параметров (spfile).

Рекомендуемая частота резервного копирования составляет одну неделю.

Для более удобного управления процессами резервирования и восстановления RMAN позволяет писать командные файлы, в которых размещается вся логика.

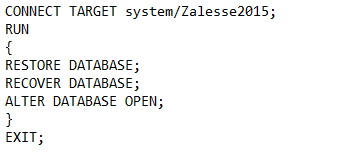
Содержание файла back.rcm который предназначен для полного резервирования базы данных (листинг 4.1).



Листинг 4.1 – Содержание файла back.rcm

В результате выполнения этих команд RMAN генерирует резервные файлы по указанному пути.

Для восстановления базы данных необходимо выполнить команды из файла rec.rcm предварительно закрыв базу данных (листинг 4.2).

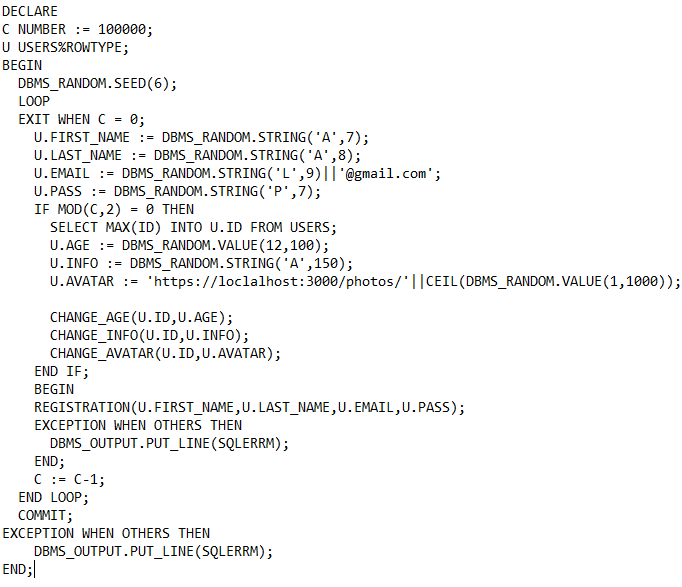


Листинг 4.2 – Содержание файла rec.rcm

# **5.** **Тестирование производительности базы данных**

Для тестирования производительности была взята за основу таблица Users.

Изначально таблица Users была заполнена на 100000 строк информации (листинг 5.1).



Листинг 5.1 – Заполнение таблицы

После этого был применён SELECT-запрос к данной таблице и при помощи стандартных средств SQL Developer оценена цена выборки к таблице (листинг 5.2).



Листинг 5.2 – Оценка времени запроса к таблице Users.

После проведения первоначальной оценки был построен B-Tree индекс к таблице Users по столбцу First\_Name и Last\_Name и проведена оценка такого же SELECT-запроса к таблице Users. B-tree строит индексы используя реализацию сбалансированного дерева. Полученные результаты во время оценки (листинг 5.3).



Листинг 5.3 – Оценка запроса к таблице с построенным B-Tree индексом.

Таким образом, постройка индекса к таблице была оправдана, так как мы получили прирост производительности и более выгодное распределение ресурсов, в зависимости от параметра, характеризующего запрос.

# **6. Руководство пользователя**

Для взаимодействия пользователя с базой данных были разработаны функции и процедуры. Список процедур и функций доступных пользователю представлен ниже, скрипты данных объектов базы данных расположены в приложении Г.

Процедуры и функции доступные стандартному пользователю: REGISTRATION – производит регистрацию нового пользователя;

CHANGE\_AGE – позволяет изменить возраст пользователя;

CHANGE\_INFO – позволяет изменить краткую информацию о пользователе;

CHANGE\_AVATAR – позволяет изменить фотографию;

LOGIN – авторизация пользователя;

GET\_USER\_BY\_ID – поиск пользователя по его идентификатору;

GET\_USERS\_BY\_NAME – поиск пользователя по его имени;

GET\_USER\_BY\_EMAIL – поиск пользователя по его почте;

ADD\_FRIEND – добавление другого пользователя в друзья, DELETE\_FRIEND\_BY\_ID – удаление пользователя из друзей;

GET\_FRIEND\_BY\_ID – поиск друзей по идентификатору;

GET\_FRIEND\_BY\_EMAIL – поиск друзей по почте;

ADD\_ALBUM – создание альбома;

DELETE\_ALBUM\_BY\_ID – удаление альбома по идентификатору;

DELETE\_ALBUM\_BY\_USER\_ID – удаление альбома по идентификатору пользователя который создал альбом;

DELETE\_ALBUM\_BY\_TITLE – удаление альбома по его названию;

DELETE\_ALBUM\_BY\_DATE – удаление альбома по дате создания;

UPDATE\_ALBUM – обновление данных альбома;

GET\_ALBUM\_BY\_ID – поиск альбома по идентификатору;

GET\_ALBUMS\_BY\_USER\_ID – поиск альбома по идентификатору создавшего его пользователя;

GET\_ALBUMS\_BY\_TITLE – поиск альбома по названию;

GET\_ALBUMS\_BY\_DATE – поиск альбома по дате создания;

ADD\_PHOTO – добавление фото;

DELETE\_PHOTO\_BY\_ID – удаление фото по идентификатору; DELETE\_PHOTO\_BY\_ALBUM\_ID – удаление фото по идентификатору альбома;

UPDATE\_PHOTO\_COORDS – изменение разрешения фото;

GET\_PHOTOS\_BY\_ALBUM\_ID – поиск фото по идентификатору альбома; GET\_PHOTO\_BY\_ID – поиск фото по его идентификатору; GET\_PHOTO\_BY\_USER\_ID – поиск фото по идентификатору пользователя, который добавил это фото;

ADD\_COMMENT – добавление комментария под фото; DELETE\_COMMENT\_BY\_ID – удаление комментария по идентификатору;

DELETE\_COMMENT\_BY\_PHOTO\_ID – удаление комментария по идентификатору фото;

GET\_COMMENTS\_BY\_PHOTO\_ID – поиск комментария по идентификатору фото;

GET\_COMMENT\_BY\_ID – поиск комментария по идентификатору;

ADD\_DIALOG – создания диалога;

DELETE\_DIALOG\_BY\_ID – удаление диалога по идентификатору;

INVITE\_USER\_TO\_DIALOG – добавление пользователя в диалог;

REMOVE\_USER\_OF\_DIALOG – удаление пользователя из диалога;

GET\_DIALOG\_BY\_ID – поиск диалога по идентификатору;

GET\_DIALOGS\_BY\_TITLE – поиск диалога по его названию;

GET\_DIALOGS\_BY\_CREATOR\_ID – поиск диалога по идентификатору создателя диалога;

ADD\_MESSAGE – отправка сообщения;

DELETE\_MESSAGE – удаление сообщения;

GET\_MESSAGES\_BY\_DIALOG\_ID – поиск сообщения по идентификатору диалога;

GET\_MESSAGE\_BY\_ID – поиск сообщения по его идентификатору;

GET\_MESSAGES\_BY\_DIALOG\_ID – поиск сообщения по идентификатору диалога;

GET\_MESSAGES\_BY\_SENDER\_ID – поиск сообщения по идентификатору отправителя

GET\_ROSES – поиск добавленных роз;

GET\_ROSE\_BY\_NAME – поиск роз по имени;

GET\_ROSE\_BY\_ID – поиск поз по идентификатору;

GET\_NEWS – поиск новостей;

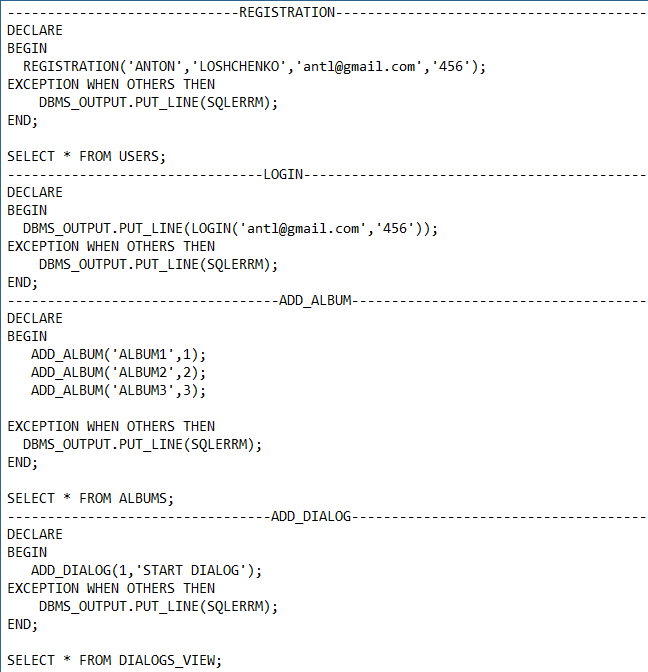
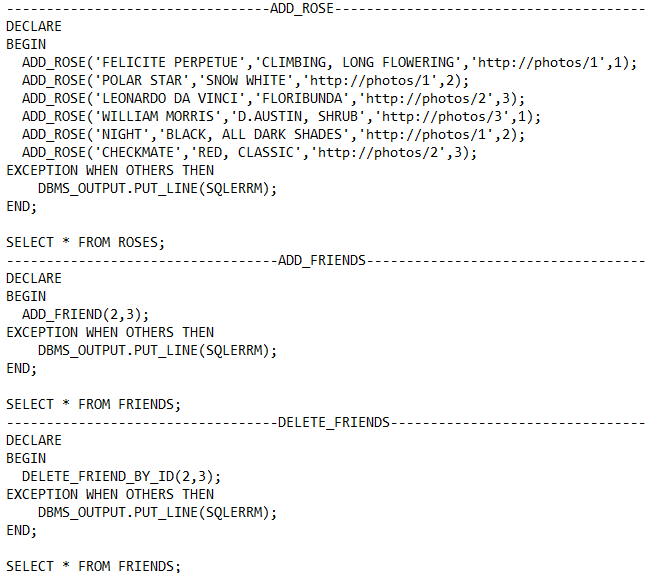
GET\_NEWS\_BY\_ID – поиск новостей по идентификатору;

GET\_NEWS\_BY\_TITLE – поиск новостей по названию.

В итоге функционал пользователя оказался достаточно прост и информативен для использования, так как он представляет собой стандартный CRUD интерфейс.

# **7. Проверка работоспособности**

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

Для проверки корректности работы программного средства был написан файл test\_db.sql, содержащий процедуры и функции из технического задания (листинг 7.1).

Листинг 7.1 – Содержимое файла test.sql.

# **Заключение**

В процессе выполнения курсовой работы была достигнута поставленная цель по созданию электронной системы, которая аккумулирует в себе функции социальной сети, новостного портала и библиотеки роз. Удалось обеспечить большую гибкость базы данных за счет исключения избыточности и несогласованности зависимости. Так же была реализована инкапсуляция внутренней структуры базы данных посредством создания пользователей, представлений и синонимов. Был выполнен анализ скорости работы базы данных и принято решение создать индексы, в результате этого скорость доступа к данным увеличилась в разы.

При разработке проекта выполнены следующие пункты:

* регистрация пользователей в базе данных;
* создание личного профиля;
* добавление пользователей в друзья;
* отправка сообщений другим пользователям;
* просмотр и добавление новостей;
* создание альбомов и добавление в них фотографий;
* просмотр и добавление информации о розах;
* просмотр информации о питомниках;
* добавление комментариев под фотографиями;

В результате выполнения курсового проекта были получены навыки по взаимодействию с документами в формате xml посредством системных пакетов Oracle, изучена технология резервного копирования и восстановления базы данных.

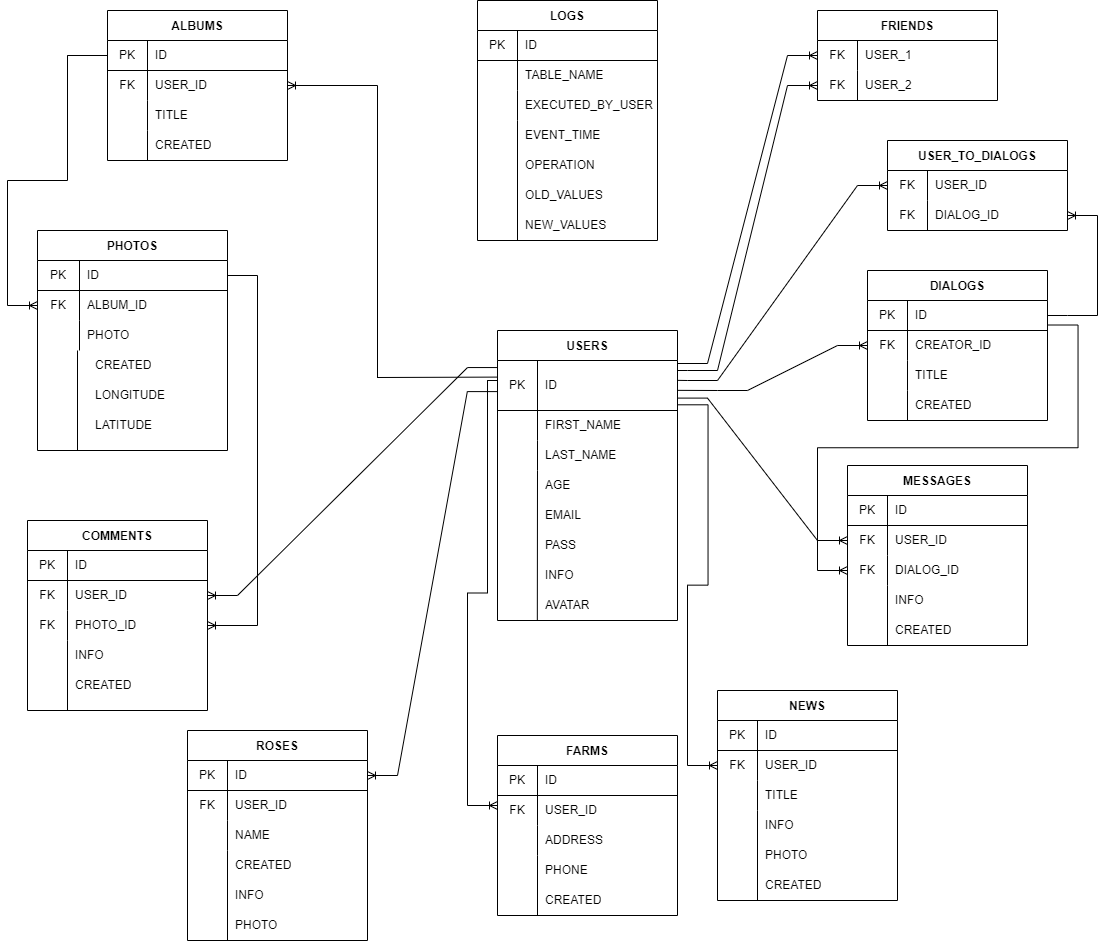
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

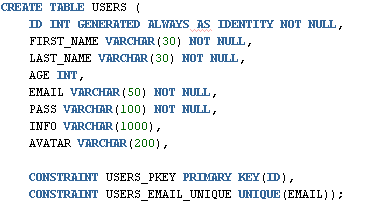
.

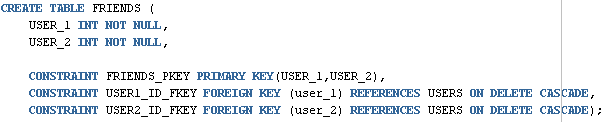
# **Список использованной литературы**

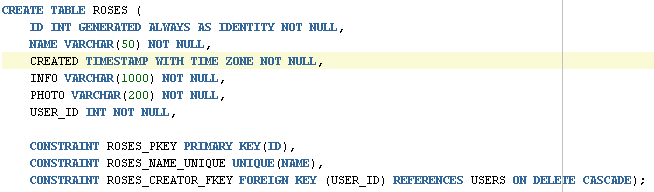
1. Lance A. Oracle Database Concepts, 12c Release 1 [Текст] / Tom K. – Chicago,Illinois, USA, 2014 – 550с.
2. Bert R. Oracle Database 2 Day DBA 12c Release 1 [Текст] – Chicago,Illinois, USA, 2014 – 285с.
3. Ravinder G. Mastering Oracle GoldenGate [Текст] – Chicago,Illinois, USA, 2016 – 649с.
4. Tom K. Expert oracle database architecture Third Edition [Текст]/ Darl K. – Москва, 2016 – 960с.
5. Oracle RMAN Commands [Электронный ресурс]: Дата доступа: 10.12.2020. Режим доступа <https://blogs.oracle.com/oraclemagazine/scripting-oracle-rman-commands>.

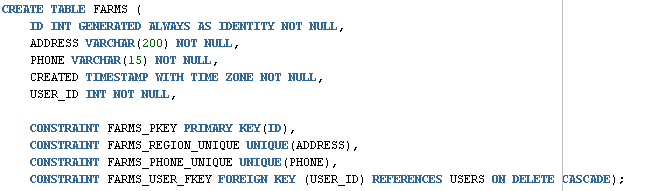
# **Приложение А**

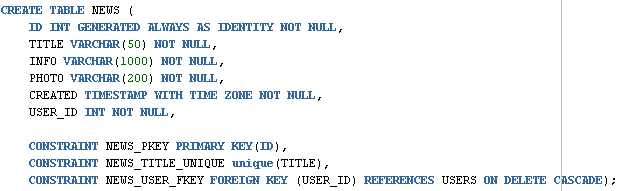


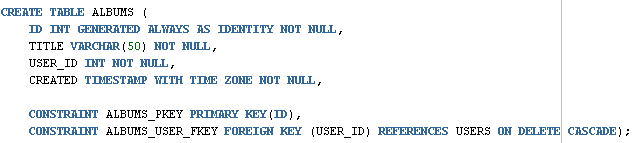


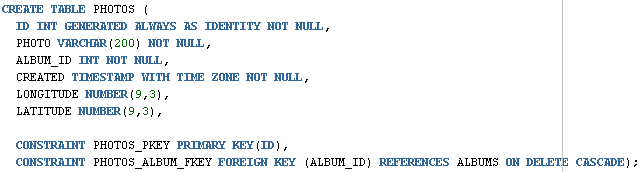


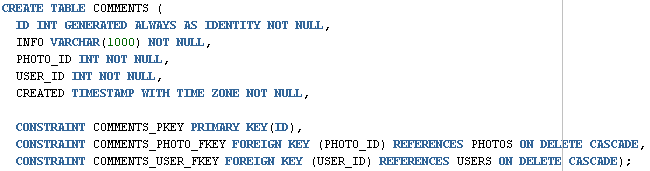


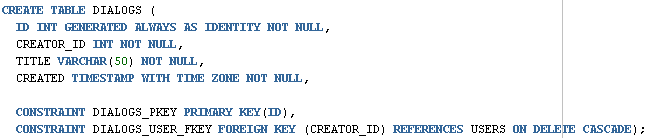


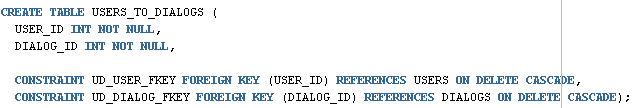


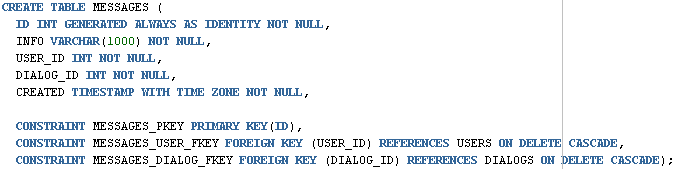


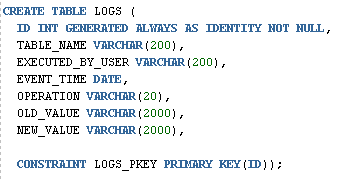




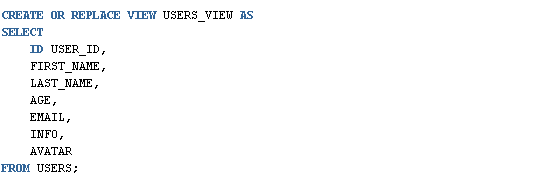


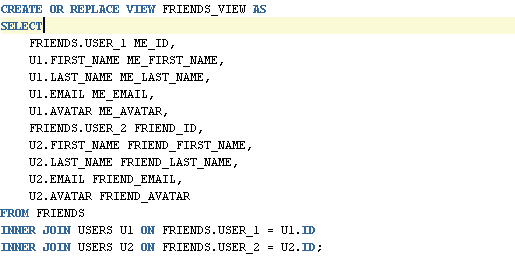


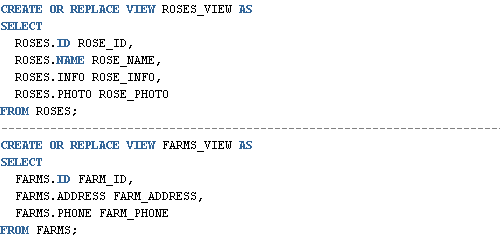


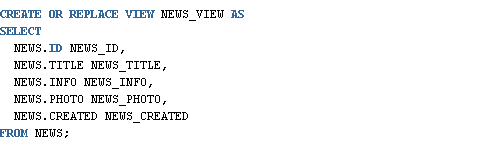


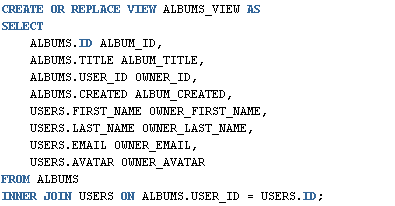
# **Приложение Б**

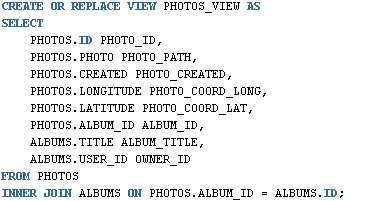


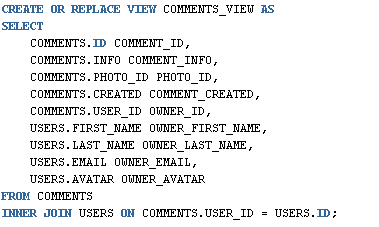
****

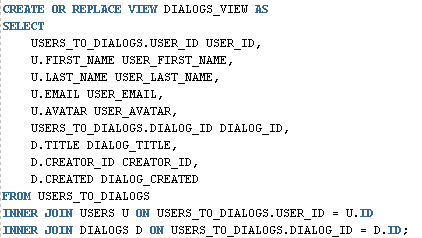
****

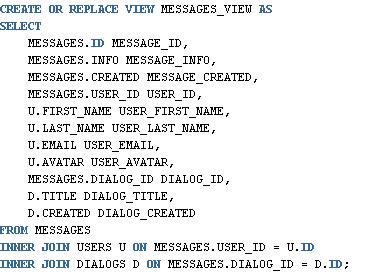
****

****

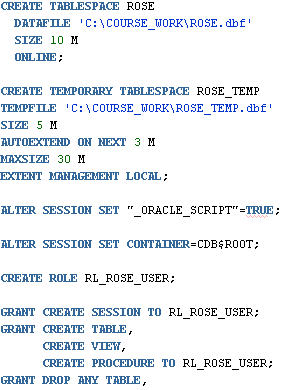
****

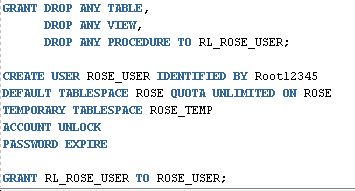
****

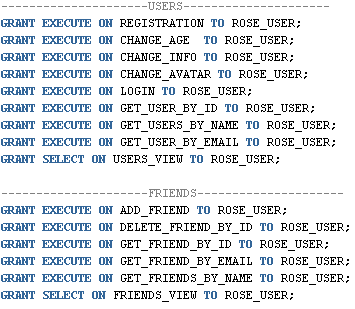
****

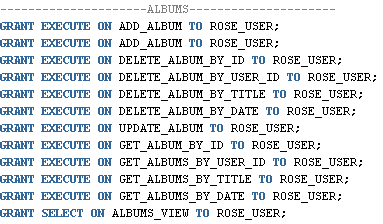
****

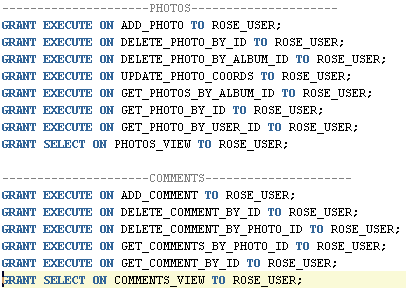
# **Приложение В**

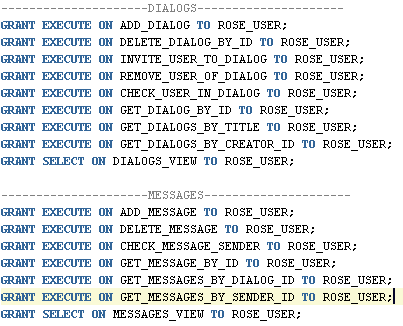


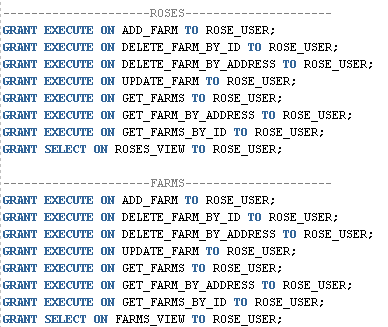


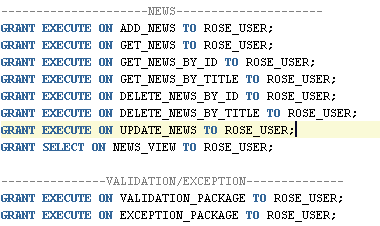




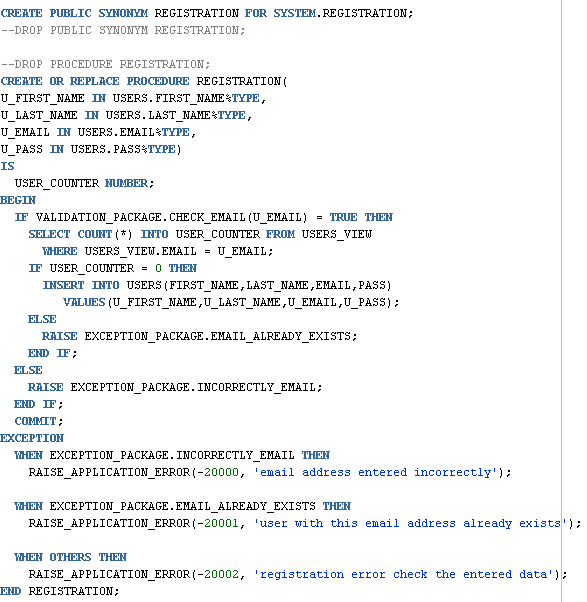


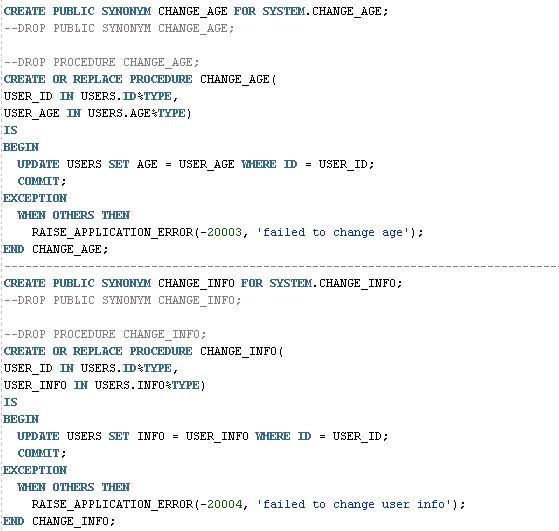


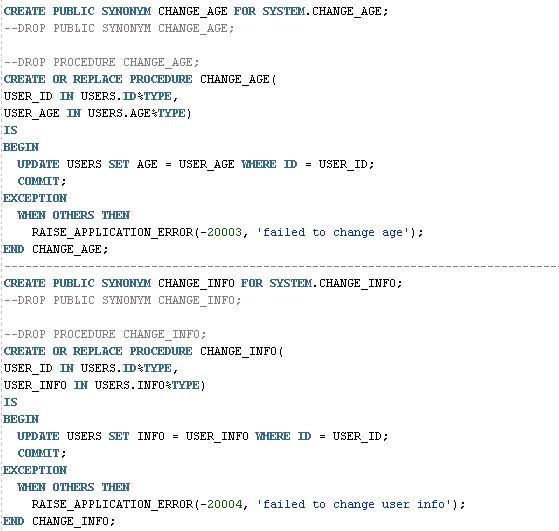


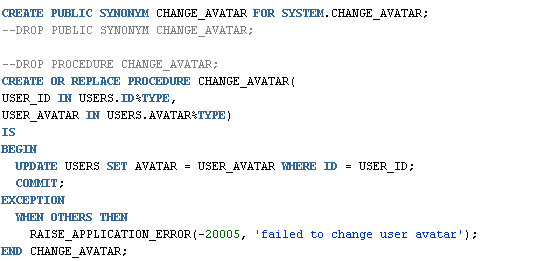


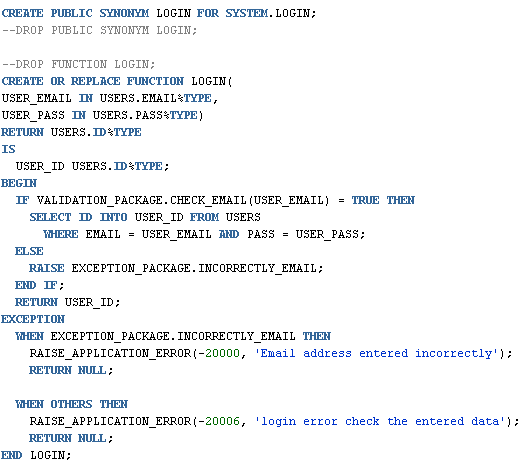
# **Приложение Г**

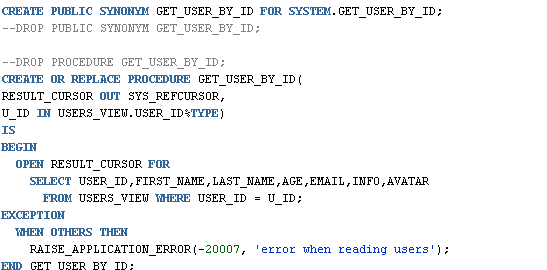


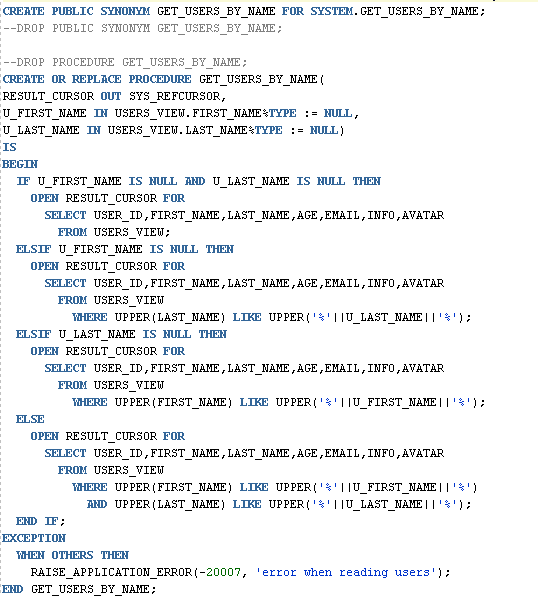


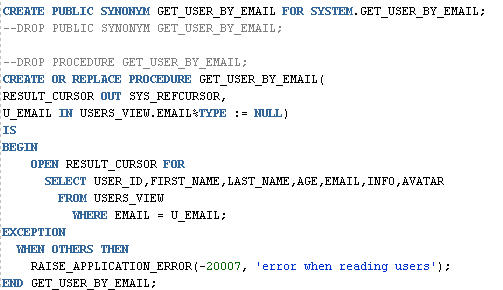


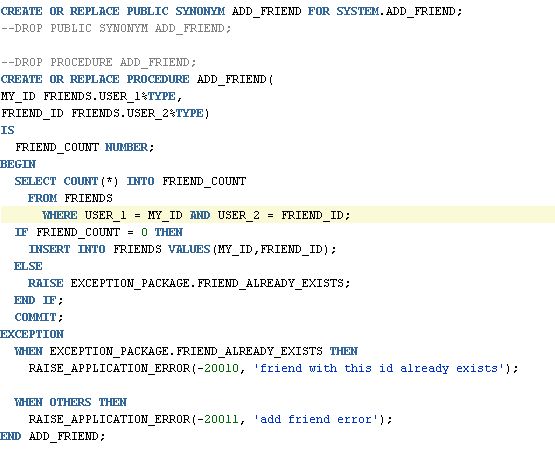


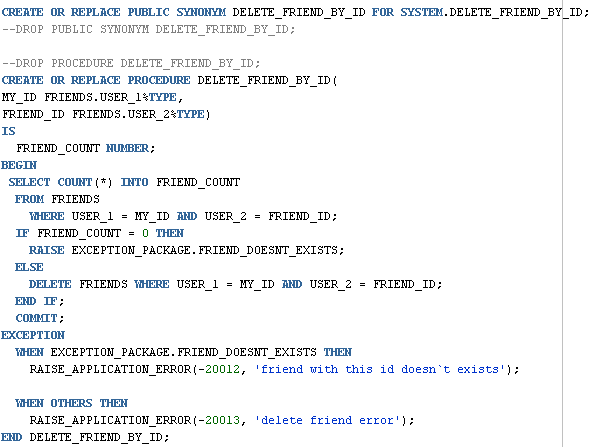


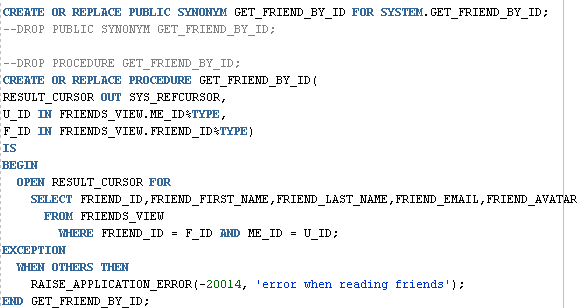


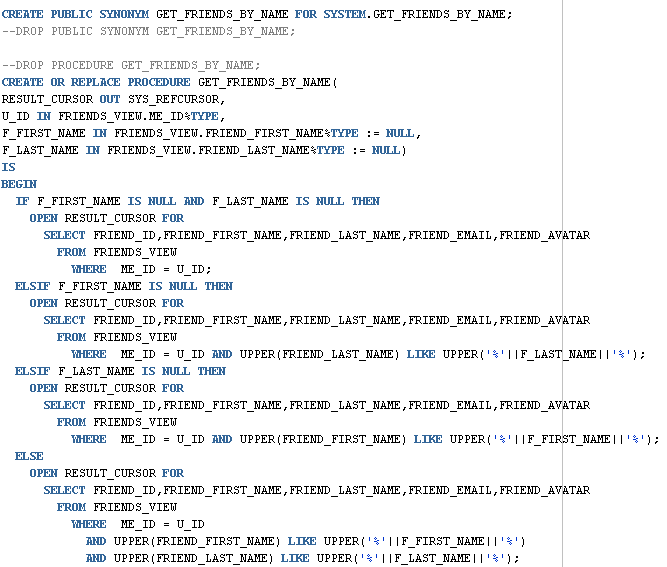


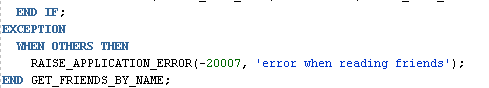


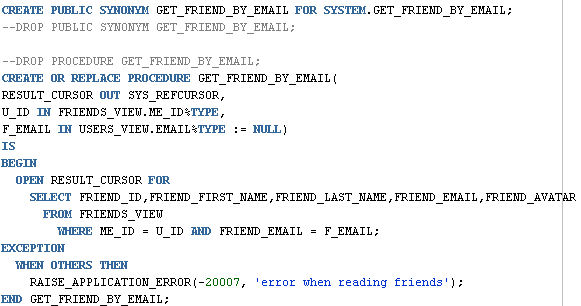


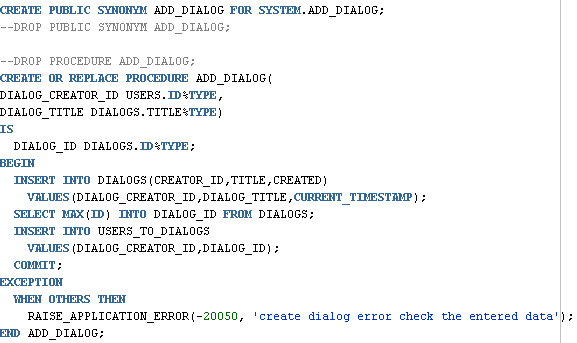


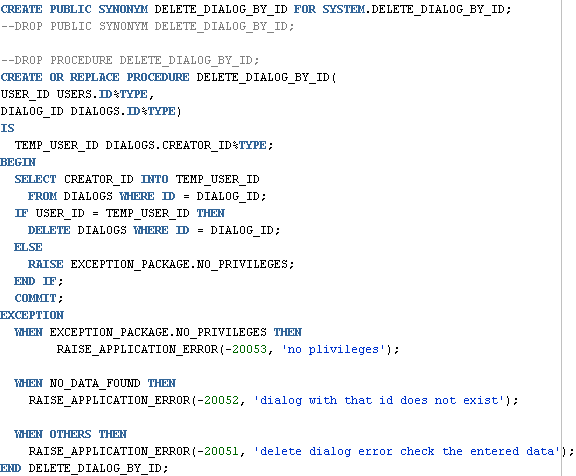


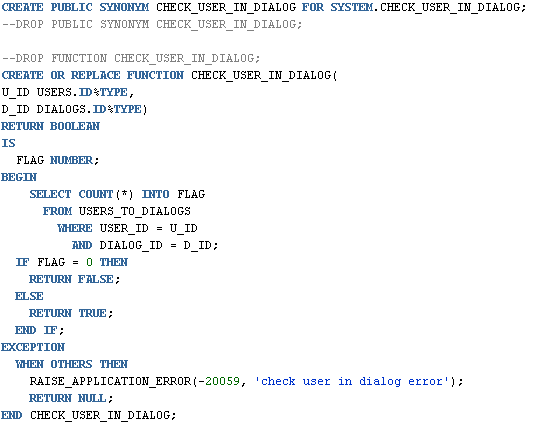


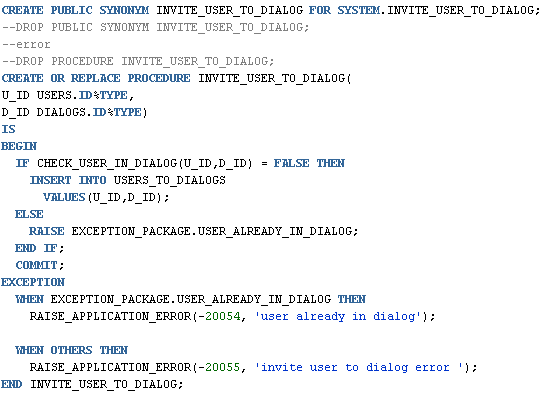


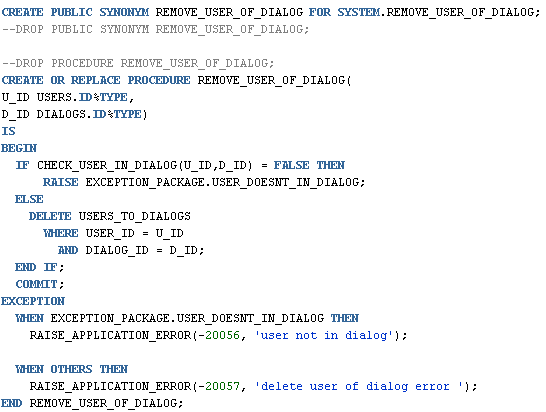


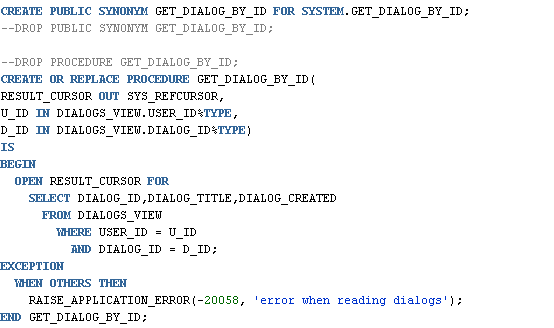


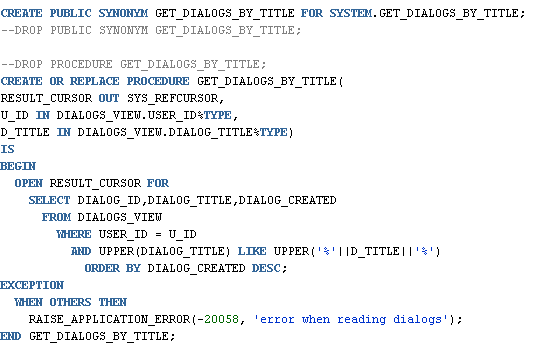


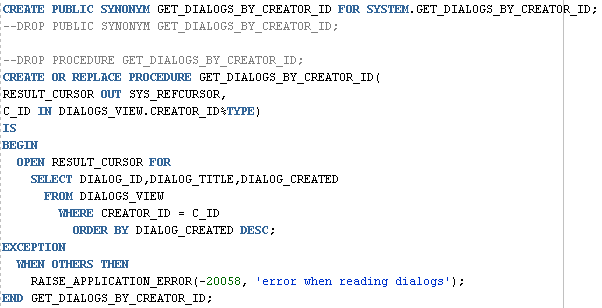


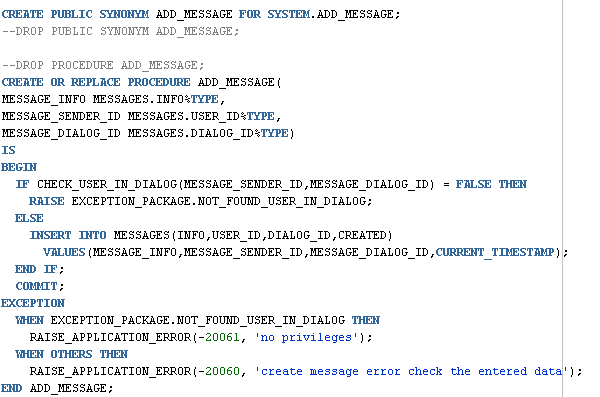


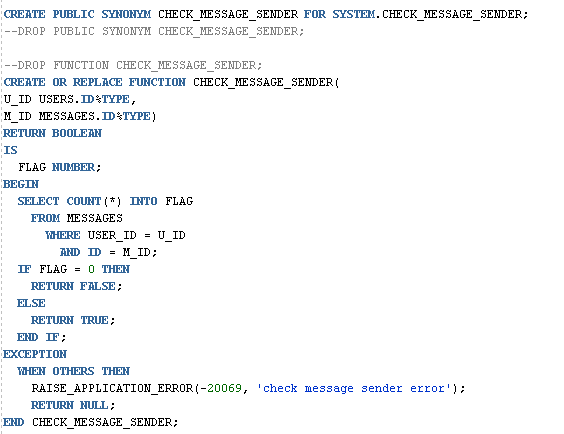


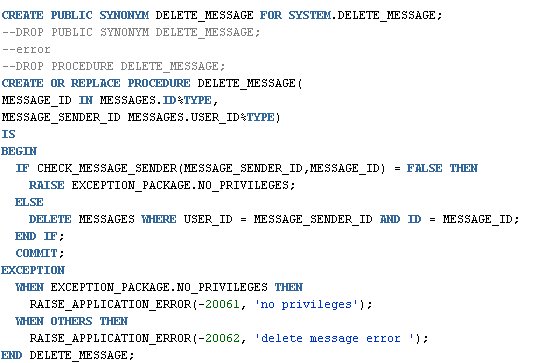


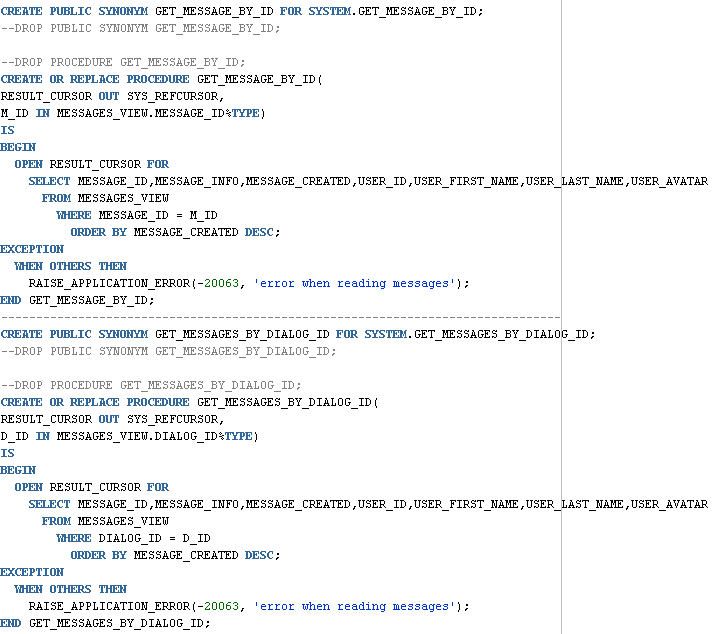
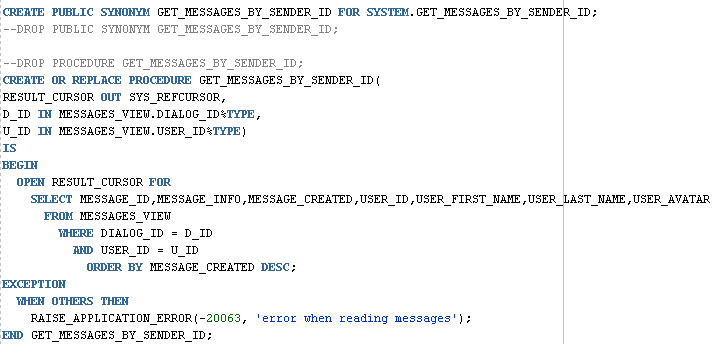


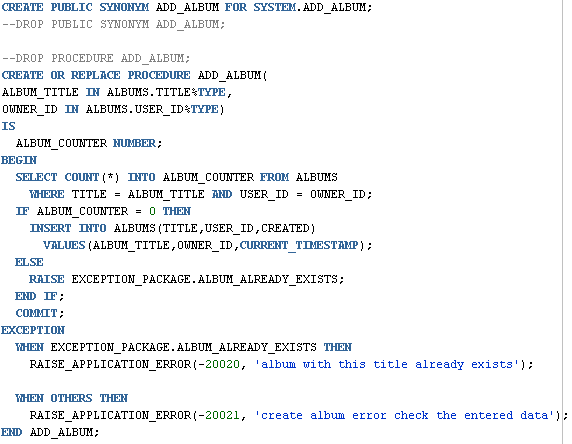


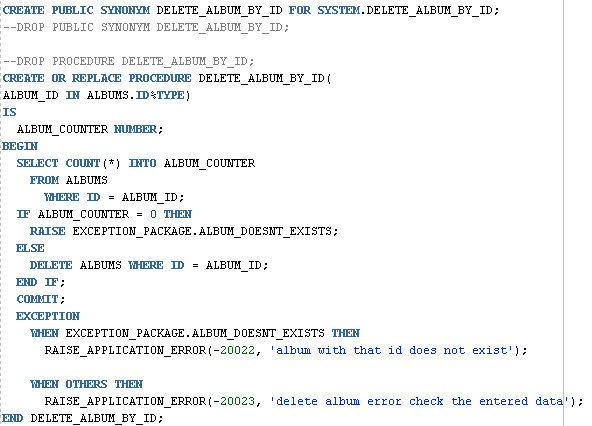


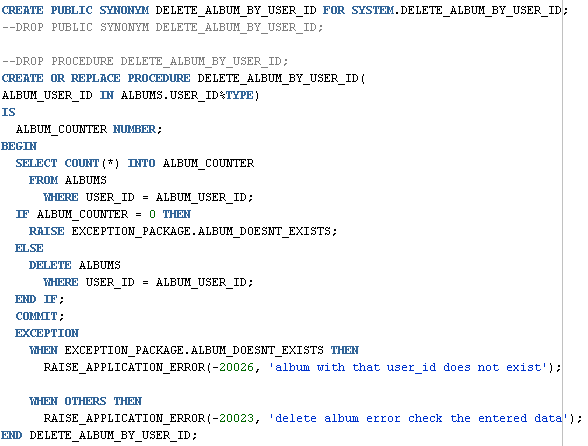


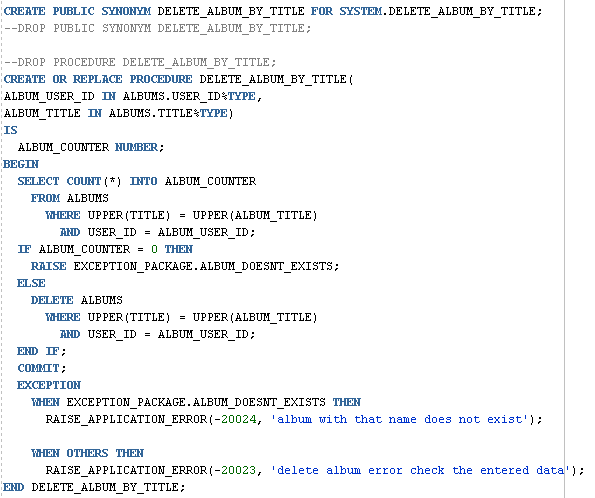


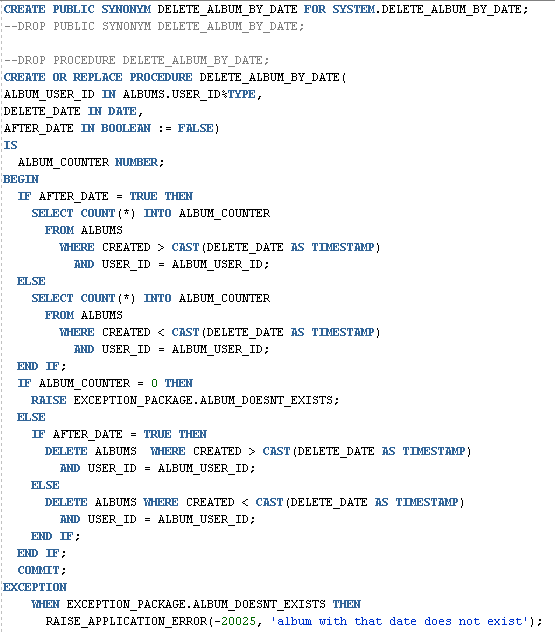


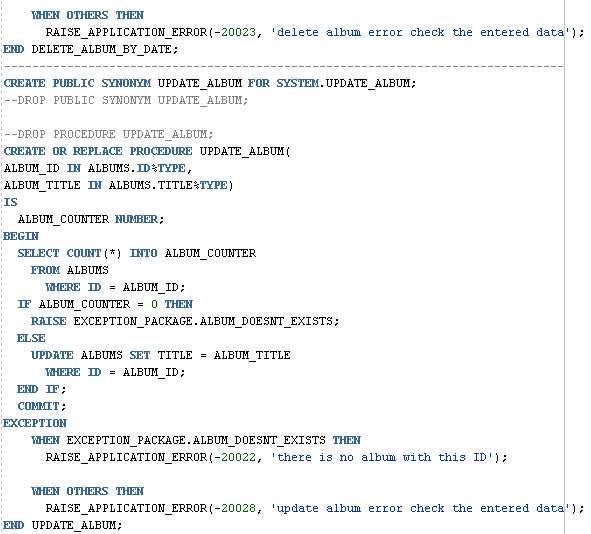


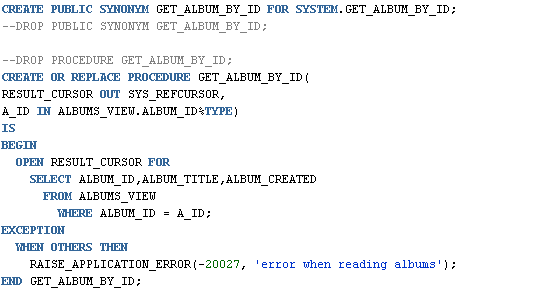


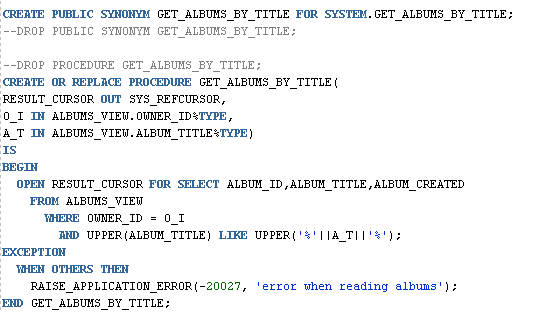


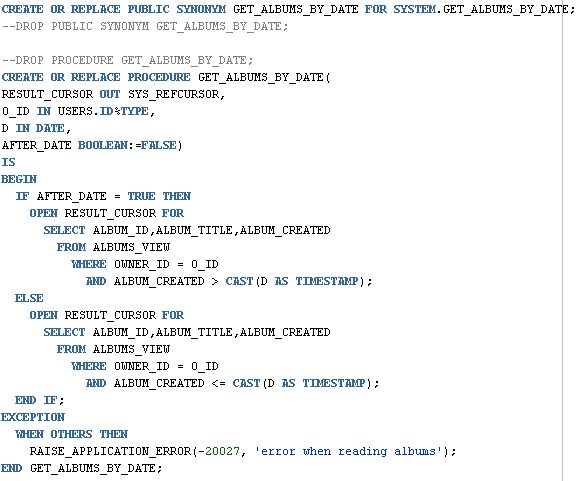




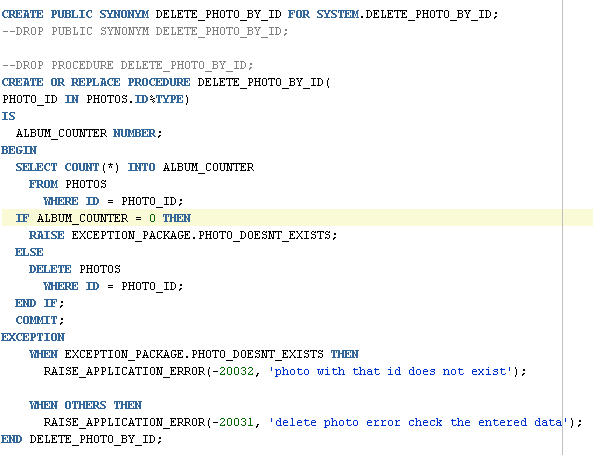


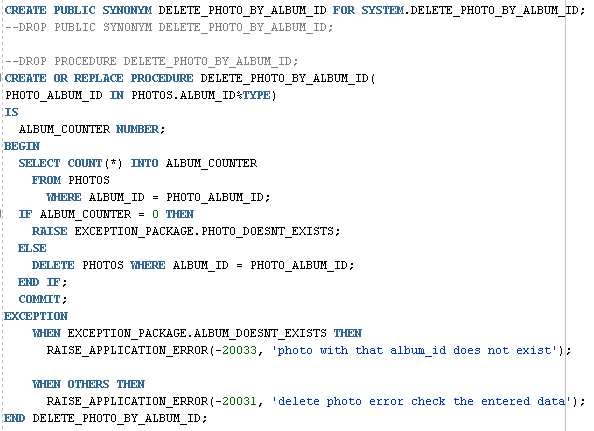


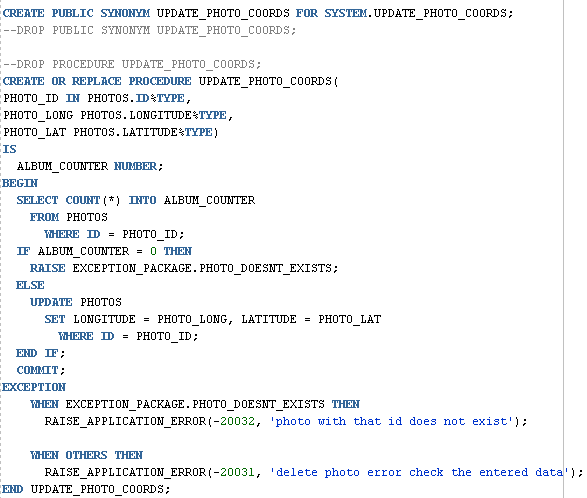


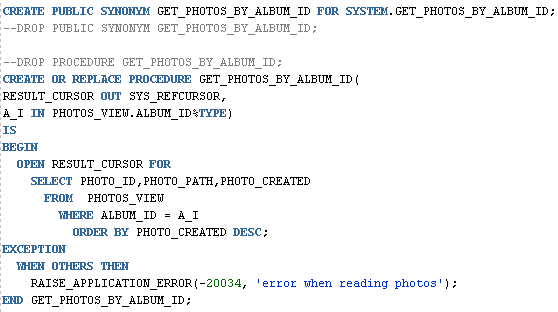


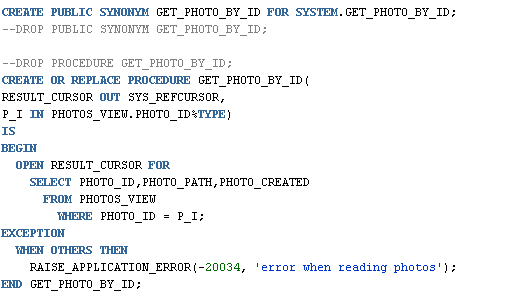


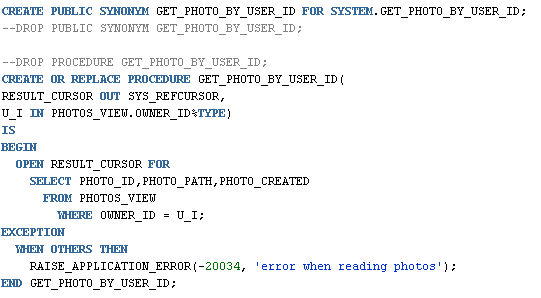


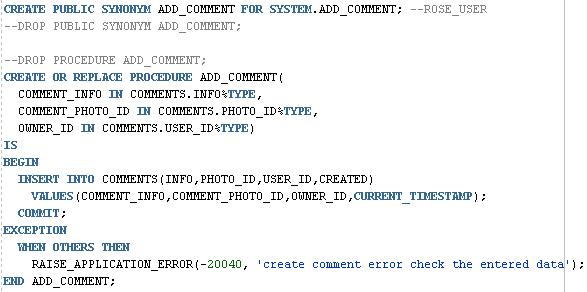


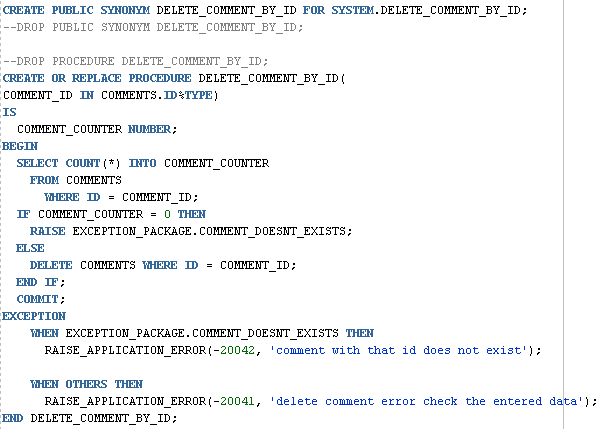


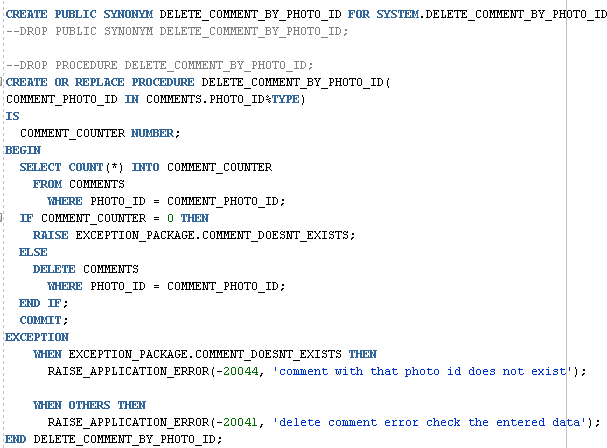


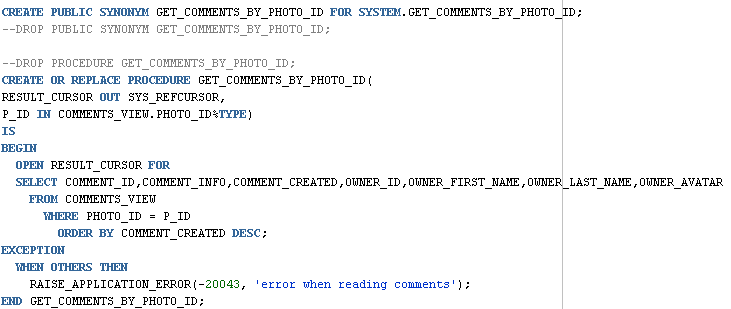


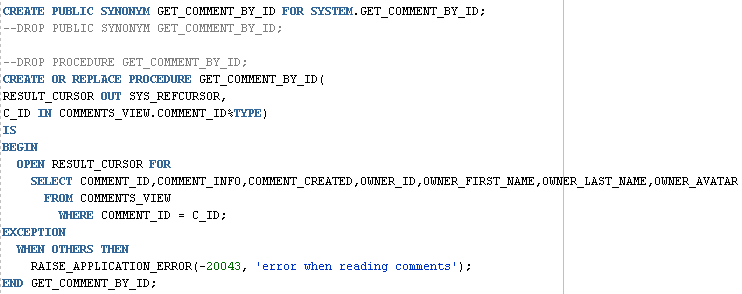
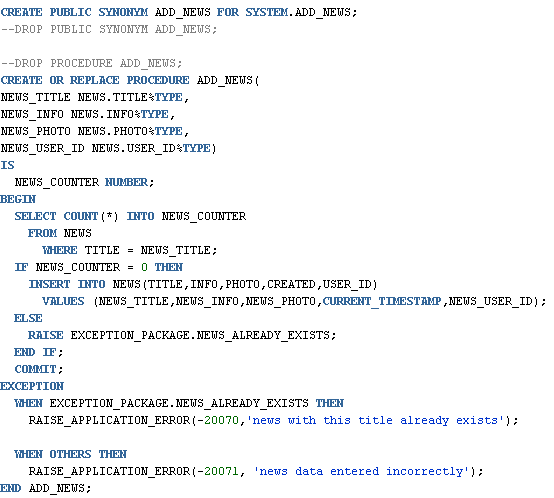


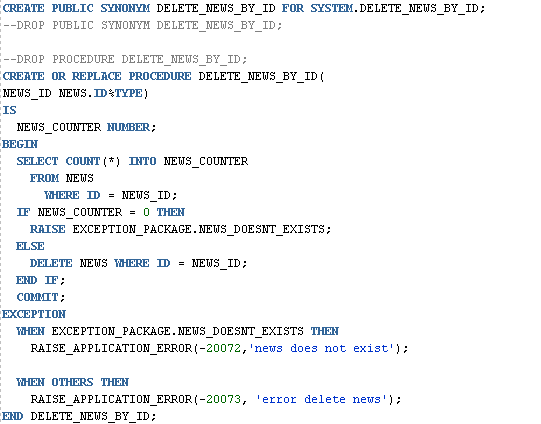


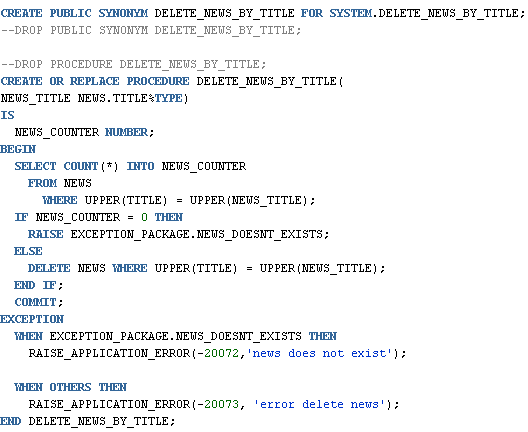


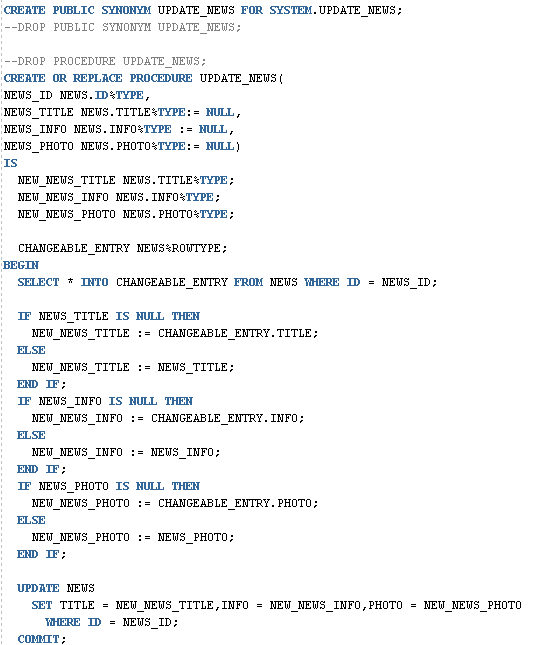


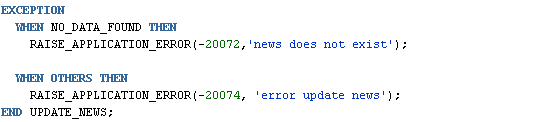


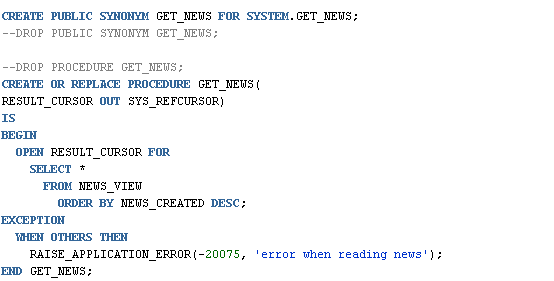


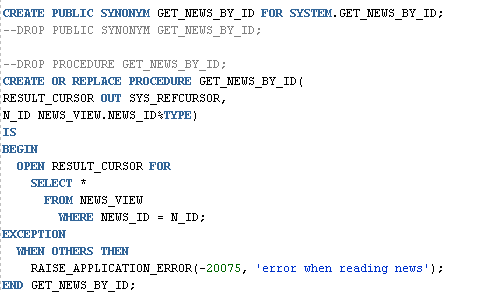


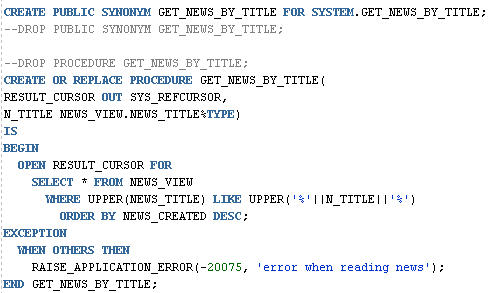


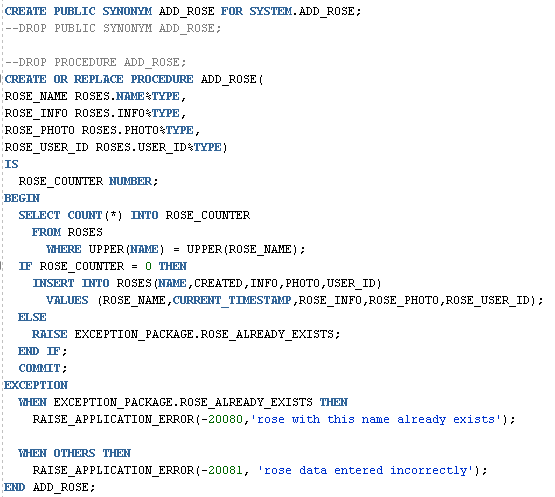


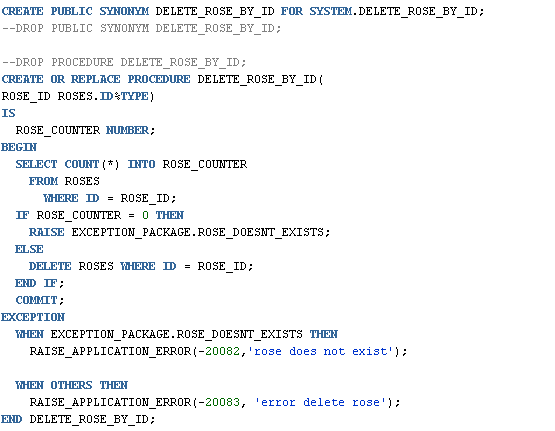


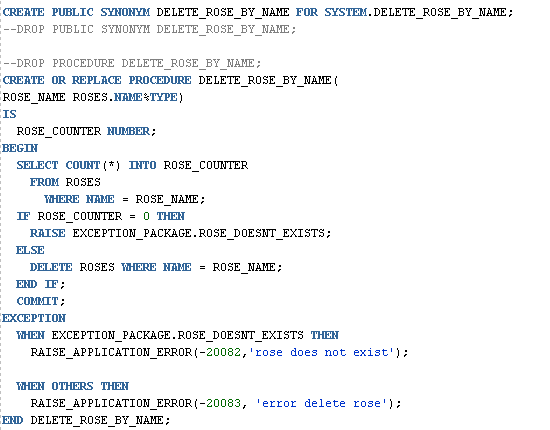


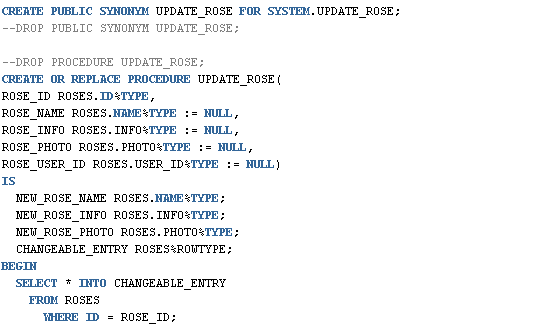


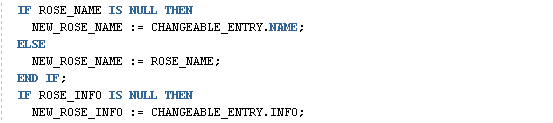


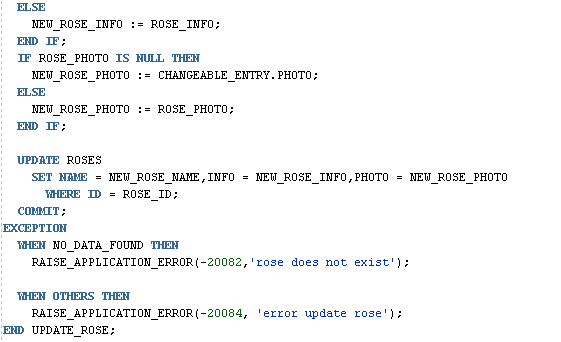


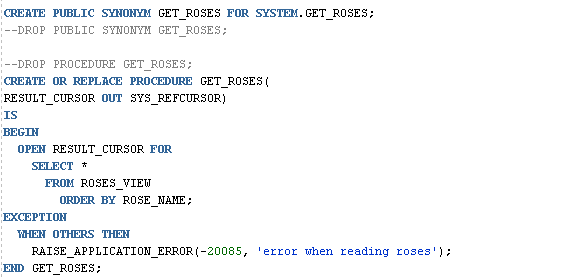


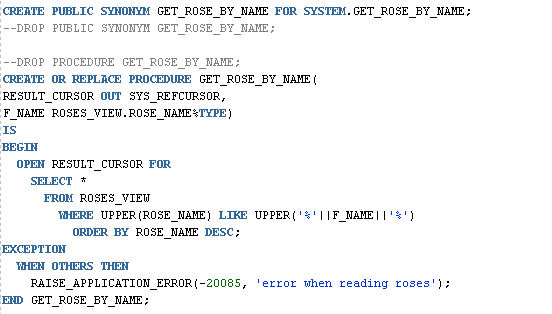


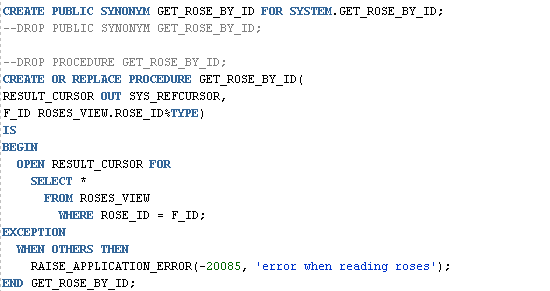


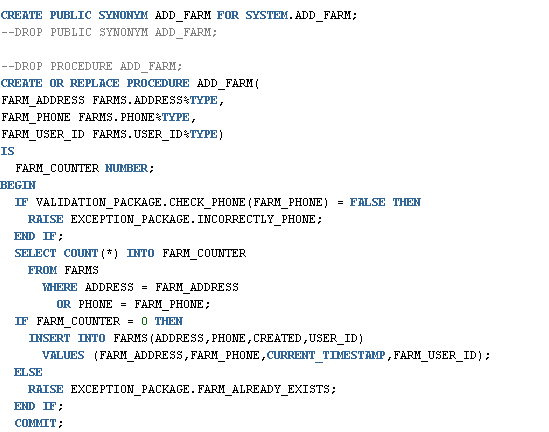


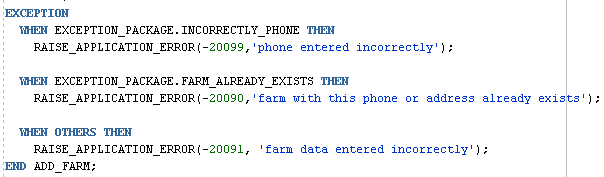


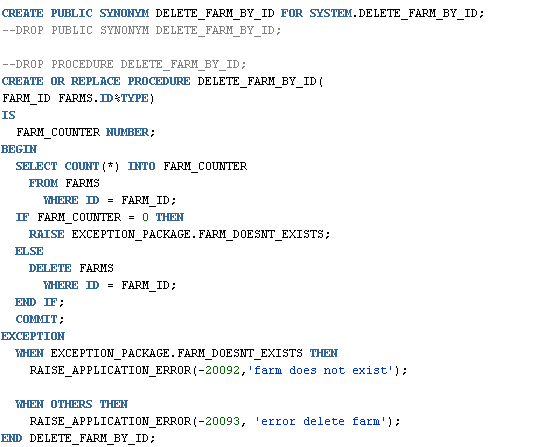


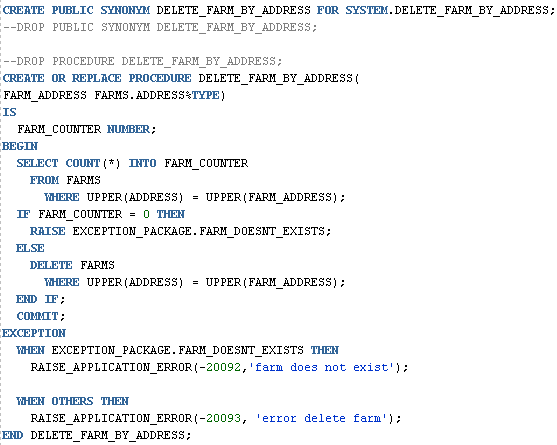


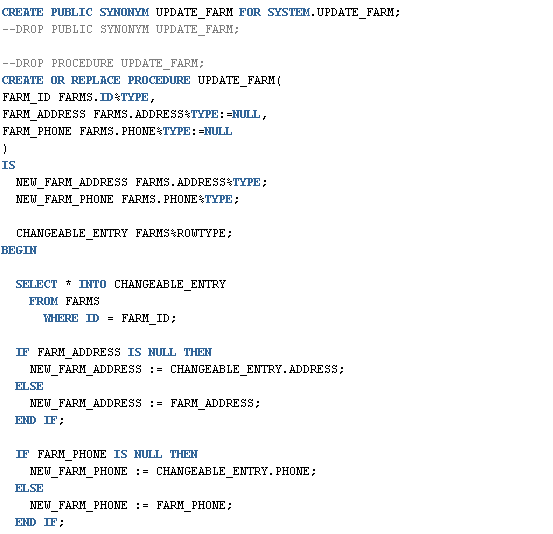


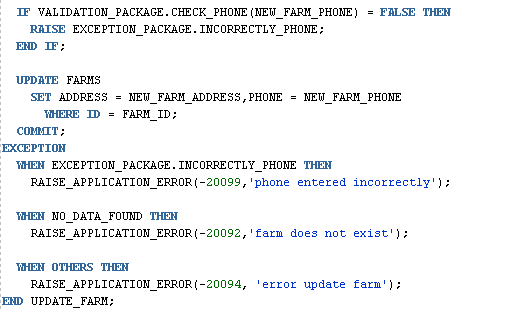


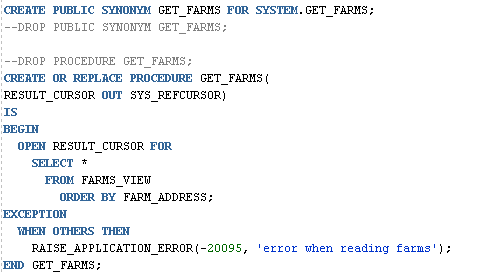


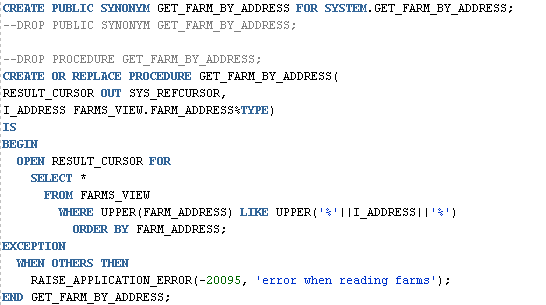


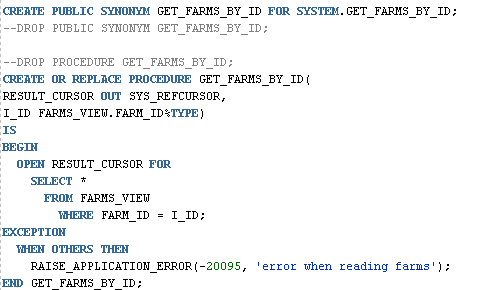




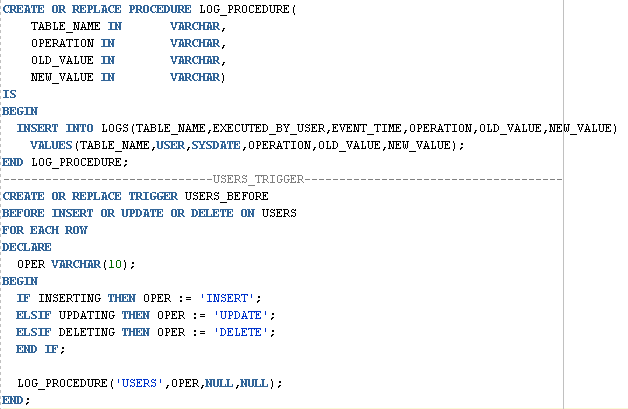


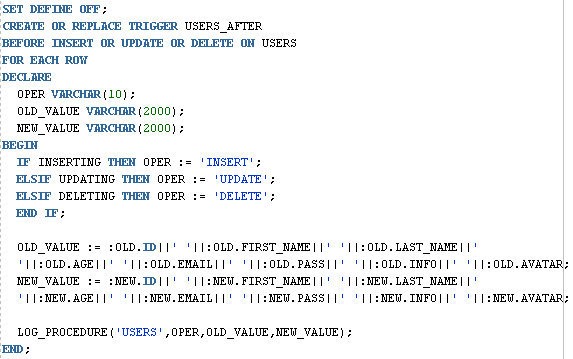


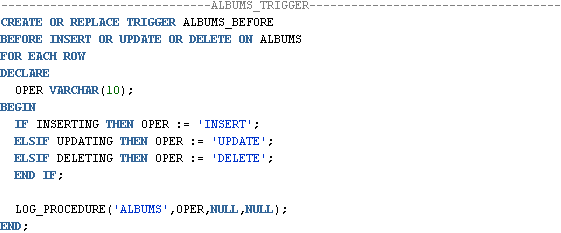


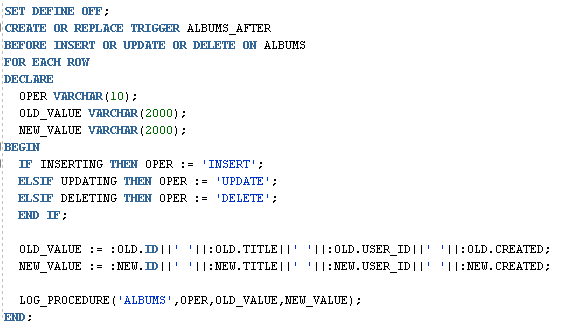


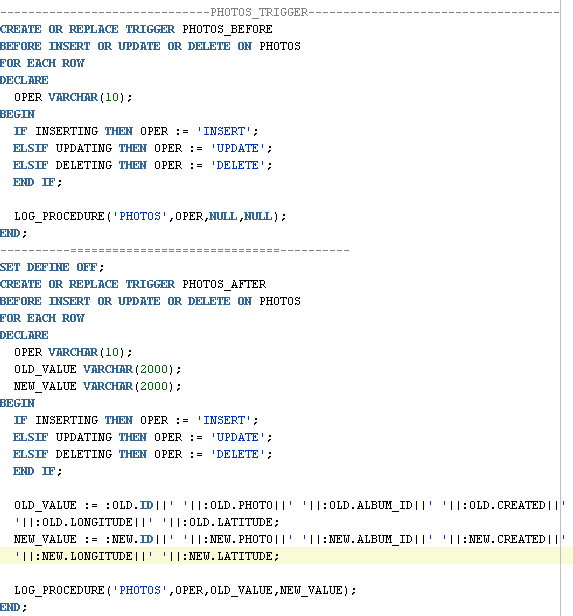
# **Приложение Д**

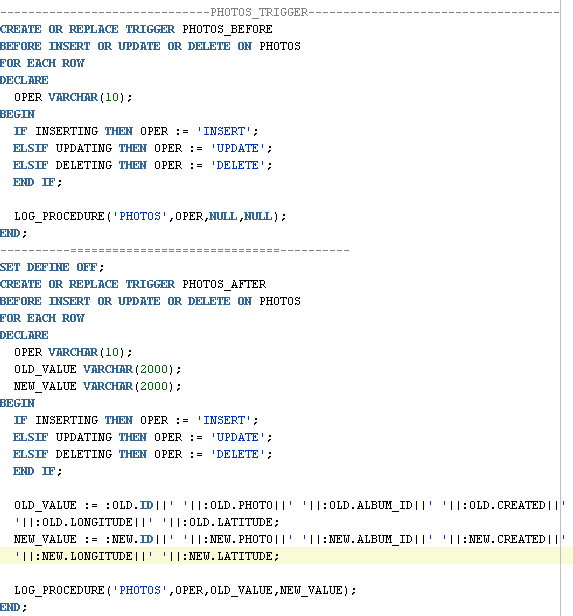


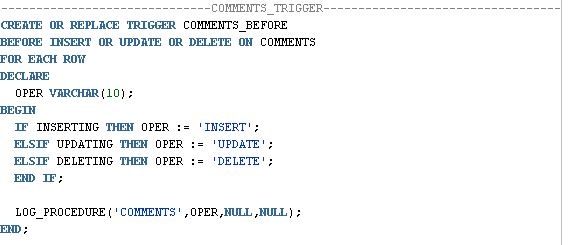


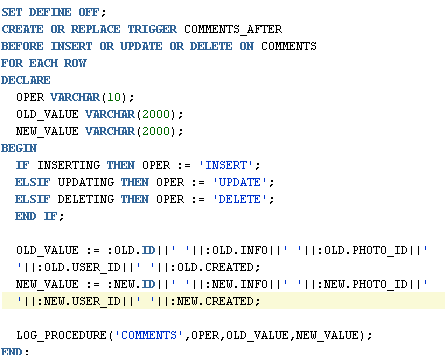


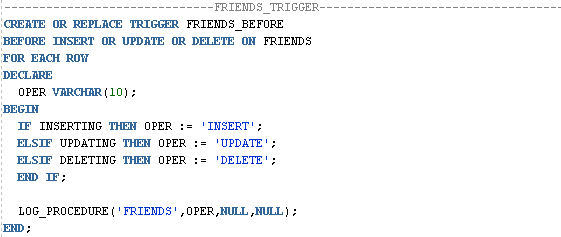


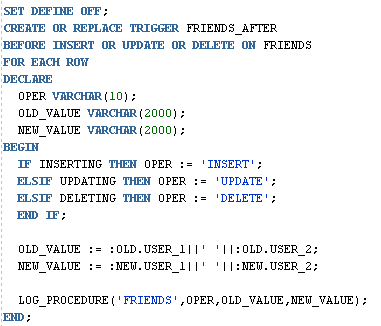


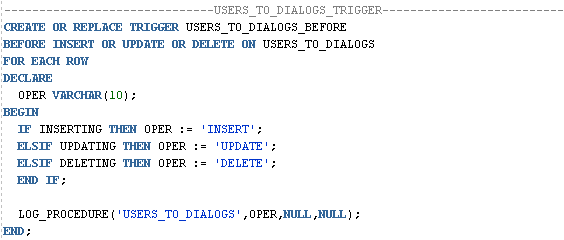


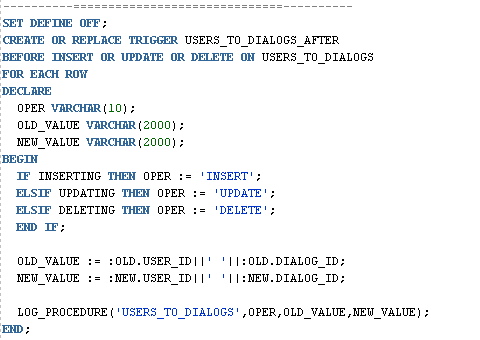


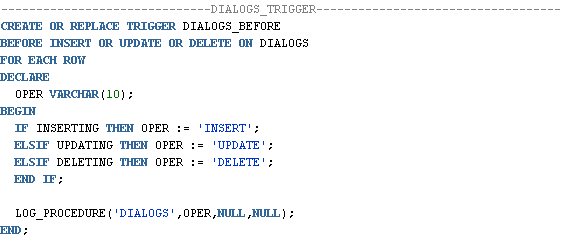


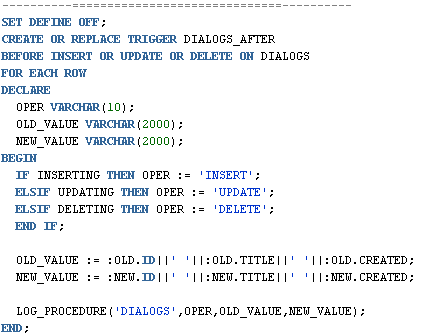


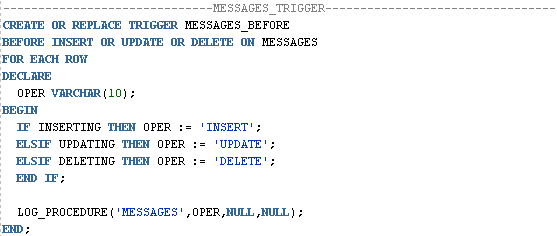


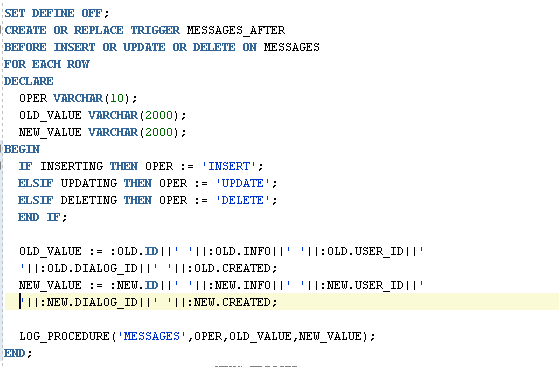


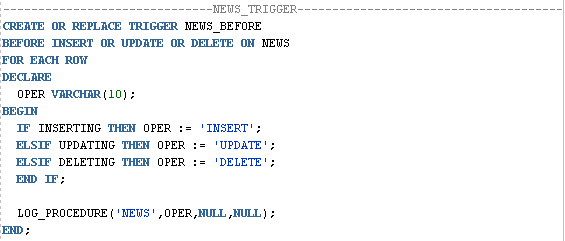


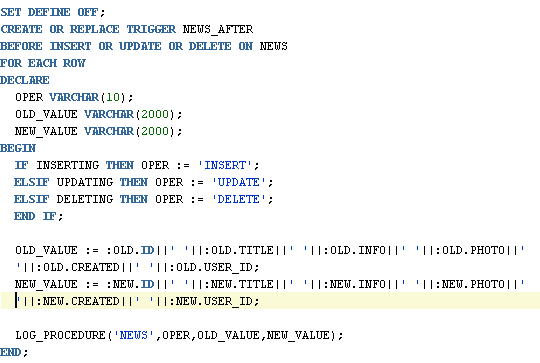


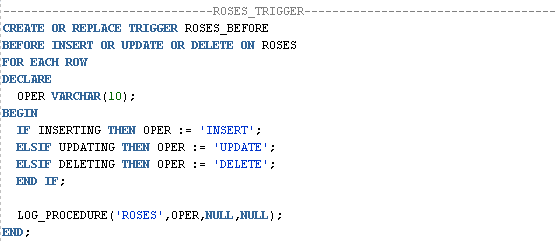


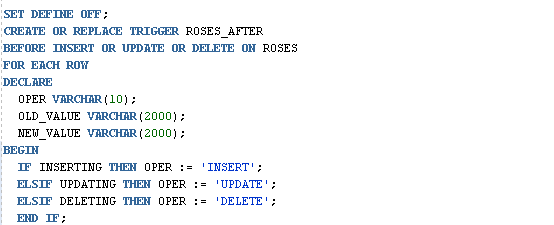


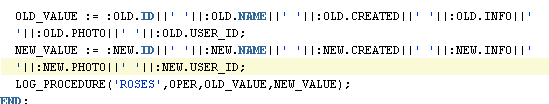


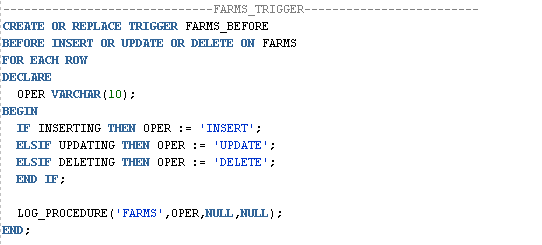


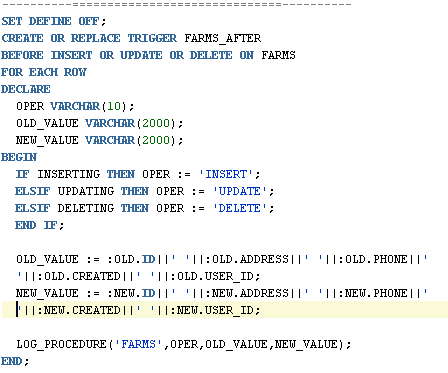












# **Приложение Е**



